Compositional Performance Analysis

C. Tofts*

School of Computer Studies, Leeds Unversity, Leeds, LS2 9JT. cmnt@scs.leeds.ac.uk, http::/www.scs.leeds.ac.uk/chris/

Abstract. Recent extensions to process algebras can be used to describe performance or error rate properties of systems. We examine an abstract approach to the representation of time costs within these algebras that permits the efficient calculation of performance bounds on systems. In particular we avoid the 'state explosion' caused by the parallel composition of the representations of probabilistic time distributions. A major advantage of one of our approaches is its uniformity, which allows the eventual approximation level to be easily predicted from quality of the approximations to the underlying distributions.

1 Introduction

Recently there has been considerable interest in the use of formal methods to derive performance predictions for concurrent systems including [Tof90, GSST90, SS90, Han91, HM93, HR94]. One particular approach is to exploit a process algebraic description of the system under consideration [Mil80, Mil83, Mil90, Tof90, Han91, VW92, Tof93, Tof94, Tof95]. Whilst it is possible to describe large scale systems using such methods, the efficient derivation of system properties remains difficult, as a result of the large number of states required to describe such systems. In this paper we examine how costs within a simple model can be considered compositionally thus permitting us to reason effectively over large scale systems.

Consider the following two processes in some prototypical asynchronous probabilistic process description language:

$$P \stackrel{def}{=} \overline{a}.D_P.\overline{b}.P \qquad\qquad Q \stackrel{def}{=} a.D_Q.b.c.Q$$

the intention of the above is that the actions $\overline{a}$ and $\overline{b}$ in the process P are separated in time by an amount given by the distribution D_P, and likewise for the process Q. Considering a possible parallel composition of the above two processes: $S \stackrel{def}{=} (P|Q)\backslash a, b$ clearly we should expect that this process is observationally equivalent to the process: $R \stackrel{def}{=} D_?.c.R$. Since the two processes will synchronise on the a action, then both spend some time, given by their respective distributions, and then synchronise on the b action[2], with the process Q

* This work is supported by an EPSRC Advanced Fellowship.

[2] This is similar to the account of asynchronous actions with duration given by Castellani and Hennessy [CH87].

producing a c action. Since the processes are asynchronous if the b action (or its dual) becomes enabled in one process before the other we would demand that it wait for the dual action to become available before continuing. However a question remains, what is the distribution $D_?$, and can it be derived efficiently? Clearly, from the simple intended semantics given above, $D_?$ is a function of the maximum of the two distributions D_P and D_Q.

A possible solution to the above problem is to search for a simply defined family of distributions (that is one whose definition rests on a small number of parameters) which are preserved under the compositions of a process algebra. We could subsequently use this distribution family to approximate the distributions in the model system. In terms of simplicity the primary candidate is the family of geometric distributions, as they require but a single parameter to describe each particular distribution. This approach has been followed by Hillston [Hil94] but, as we shall demonstrate later, this family of distributions is **not** closed under the operations of a process algebra.

A further possibility is the use of the very general **phase distributions** [Nel95] which are known to be closed under the maximum operator. However, as we shall observe, whilst providing a well formed closed family under the operations of a process algebra, that these distributions do not provide an abstraction[3]. Indeed we are obliged to maintain as much information to describe the phase distribution as is present within the original description.

Throughout the sequel we shall use the Weighted Synchronous Calculus of Communicating Systems (WSCCS) to describe these composition problems. It may seem unreasonable to use a synchronous calculus to describe an asynchronous problem but as Milner [Mil83, Mil90] demonstrated these calculi actually contain asynchronous calculi. Furthermore, the underlying simplicity of the synchronous approach often clarifies difficult timing problems, and permits greater insight.

In the next section we present a brief introduction to the calculus WSCCS, this can be regarded as a prototypical probabilistic process algebra. Any other synchronous probabilistic calculus could be used [GSST90, SS90, Han91]. We then present how distributions can be represented in WSCCS and the representation of their composition. In section 5 we present the notion of abstraction over distributions, this is similar to notions of observational equivalence in CCS [Mil80,Mil90]. In section 6 we present an example calculation of the performance of a prototypical processor.

2 WSCCS

The language WSCCS [Tof90,Tof94] is an extension of Milner's SCCS [Mil83] a language for describing synchronous concurrent systems. Here we provide an introduction to the syntactic constructs that underlie WSCCS but omit the for-

[3] For instance WSCCS actually defines the appropriate phase distributions, when represented via a Markov Chain.

mal semantics and algebraic properties as they have a full description elsewhere [Tof94].

To define the language we presuppose a free abelian group Act over a set of atomic action symbols with identity $\surd$ and the inverse of a being $\overline{a}$. To distinguish multiplications over extended action names we use the symbol $\#$ to denote multiplication. As in SCCS, the complementary actions a (conventionally input) and $\overline{a}$ (output) form the basis of communication. Within our group we define that $\overline{a} = a^{-1}$. The action $\surd$ denotes the performance of a communication, taking one period of time, alternatively it can describe a one period delay.

2.1 Expressions

We define a set of expressions.

Definition 2.1 *A* relative frequency expression (RFE) *is formed from the following syntax, with x ranging over a set of variable names VRF, and c ranging over a fixed field (such as $\mathcal{N}$ or $\mathcal{R}$):*

$$e ::= x|c|e + e|e * e$$

Further we assume that the expressions form an abelian field.

In the sequel we shall omit the $*$ in expressions, denoting expression multiplication by juxtaposition. It should be noted that unlike other calculi with expressions [Mil90] the value of our expressions can have **no effect** on the structure of the transition graph of our system. Hence we should not expect that adding this extra structure to our probabilistic process algebra will cause any new technical difficulties.

2.2 Weights

We also take a set of weights $\mathcal{W}$, denoted by w_i, which are of the form[4] $e\omega^k$ with e from the relative frequency expressions and the ω^k (with $k \geq 0$) a set of infinite objects, with the multiplication and addition rules (assuming $k \geq k'$):

$$e\omega^k + f\omega^{k'} = e\omega^k = f\omega^{k'} + e\omega^k \qquad e\omega^k + f\omega^k = (e+f)\omega^k = f\omega^k + e\omega^k$$
$$e\omega^k * f\omega^{k'} = (ef)\omega^{k+k'} = f\omega^{k'} * e\omega^k$$

2.3 The Calculus

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

[4] Here e is the relative frequency with which this choice should be taken and k is the priority level of this choice. The choice of notation is based in [Tof90] arising from the observation that priority is similar to infinite weight.

$E ::= X \mid a : E \mid \sum\{w_i.E_i | i \in I\} \mid E \times E \mid E\lceil A \mid \Theta(E) \mid E[S] \mid \mu_i \tilde{x}\tilde{E}.$

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

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; $\sum\{w_i.E_i | i \in I\}$ 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 experiments the process E_i being chosen with a relative frequency of $\frac{w_i}{\sum_{i \in I} w_i}$. $E \times F$ 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; $\Theta(E)$ represents taking the most prioritised parts of the process E only; $E[S]$ 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 $\sum\{1_1.a : \mathbf{0},\ 2_2.b : \mathbf{0} | i \in \{1,2\}\}$ as $1.a + 2.b$. 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 $\mu x.a : x$. The weight n is an abbreviation for the weight $n\omega^0$, and the weight w^k is an abbreviation for the weight $1\omega^k$.

For a full description of the operational semantics, equivalences, and the algebra of WSCCS see [Tof90,Tof94].

3 Distributions in WSCCS

We start by describing three simple distributions in WSCCS to illustrate how we describe such costs and we shall then generalise these distributions.

Example 1. The linear distribution between 1 and n can be generated by the following process:

$$\begin{aligned}
L_0(n) &\stackrel{def}{=} 1.start\#\overline{finish} : L_0(n) + (n-1).start : L_1(n) + 1.\surd : L_0(n) \\
L_1(n) &\stackrel{def}{=} 1.\overline{finish} : L_0(n) + (n-2).\surd : L_2(n) \\
L_k(n) &\stackrel{def}{=} 1.\overline{finish} : L_0(n) + (n-k-1).\surd : L_{(k+1)}(n) \\
L_{(n-1)}(n) &\stackrel{def}{=} 1.\overline{finish} : L_0(n)
\end{aligned}$$

As a simpler example for $n = 3$ we obtain the following:

$$L_0(3) \stackrel{def}{=} 1.start\#\overline{finish} : L_0(3) + 2.start : L_1(3) + 1.\surd : L_0(3)$$
$$L_1(3) \stackrel{def}{=} 1.\overline{finish} : L_0(3) + 1.\surd : L_2(3)$$
$$L_2(3) \stackrel{def}{=} 1.\overline{finish} : L_0(3)$$

Notice that in the above two processes the distribution is 'initiated' by a *start* action upon which the environment **will** insist, and indicates its termination with a $\overline{finish}$ action which we shall oblige the environment to accept. After the $\overline{finish}$ action the process waits to be started again. All of our distribution processes will follow this format.

Definition 3.1 *A process P is a* distribution process *iff all of its transitions are in one of the following forms:*

$$P \stackrel{\surd}{\rightarrow} P \qquad P \stackrel{start\#\overline{finish}}{\longrightarrow} P \qquad P \stackrel{start}{\longrightarrow} (\stackrel{\surd}{\rightarrow})^* \stackrel{\overline{finish}}{\longrightarrow} P$$

Definition 3.2 *We define $P(t, P)$ to be the probability that a distribution takes time t to execute the actions start and $\overline{finish}$, we take the time of $start\#\overline{finish}$ to be 1, since it requires one tick to execute. Where it is clear from the context we shall omit the process, simply writing $P(t)$*

Proposition 3.3 *The probability of seeing a $\overline{finish}$ action at any time in the set $\{1, \ldots, n\}$ after the start action in the process $L(n)_0$ is: $\frac{1}{n}$. That is $L_0(n)$ is a correct implementation of the linear distribution.*

The above proposition can be demonstrated by direct calculation of the probability that a path between a *start* and a $\overline{finish}$ action has within the WSCCS calculus. Since in this, and the following example, that path is unique the calculation is straightforward.

Example 2. The Geometric distribution can be defined by the following process:

$$Geo_0(p) \stackrel{def}{=} p.start\#\overline{finish} : Geo_0(p) + (1-p).start : Geo_1(p) + 1.\surd : Geo_0(p)$$
$$Geo_1(p) \stackrel{def}{=} p.\overline{finish} : Geo_0(p) + (1-p).\surd : Geo_1(p)$$

Proposition 3.4 *The process $Geo_0(p)$ implements a geometric distribution, in other words: $P(t, Geo_0(p)) = p(1-p)^{(t-1)}$*

Finally we present an example of a fixed time distribution.

Example 3. A fixed time $t > 1$ between the start and finish actions:

$$Fix(t)_0 \stackrel{def}{=} 1.start : Fix(t)_1 + 1.\surd : Fix(t)_0$$
$$Fix(t)_1 \stackrel{def}{=} 1.\surd : Fix(t)_2$$
$$Fix(t)_k \stackrel{def}{=} 1.\surd : Fix(t)_{(k+1)}$$
$$Fix(t)_t \stackrel{def}{=} 1.\overline{finish} : Fix(t)_0$$

3.1 General Distributions

Definition 3.5 *A generalised distribution process using n states is given by the following process expression:*

$$G_0 \stackrel{def}{=} p_0.start\#\overline{finish} : G_0 + \sum_{i=1} np_{0i}.start : G_i + 1.\surd : G_0$$

$$G_k \stackrel{def}{=} p_k.\overline{finish} : G_0 + \sum_{i=1} np_{ki}.\surd : G_i$$

Notice that in the above we have a large number of free parameters $\{p_0, \ldots, p_n\}$ and for $\{p_{ki} | 1 \leq i, k \leq n\}$, we omit them formally from the definition for brevity, but assume that for any particular general distribution there is a fixed set of such parameters.

Proposition 3.6 *The process G_0 is a distribution process.*

Theorem 1. *The family of distributions following the format of G_0 is the discrete phase distributions. That is the distribution given by the time to absorbtion of a discrete time Markov chain with a unique absorbing state [Nel95, pp421-423].*

Corollary 3.7 *WSCCS is sufficient to express any discrete phase distribution.*

An interesting observation about the properties of such distributions and in part motivation for our abstraction work is the following.

Theorem 2. *Any bounded distribution, that is one where we can find a time limit l such that for all $t > l$, $P(t) = 0$ and $P(t = l) > 0$, requires a process with* at least *l states to describe it.*

An immediate corollary is that any $Fix(t)$ distribution requires at least t states to represent it. Hence, to represent distributions of the form wait t and then behave as a geometric, a fairly common cost form, we may need a large number of states.

The observation that fixed or limited distributions require a large number of states to describe their behaviour is a severe limitation in attempts to model large scale systems. The growth in the number of states in a concurrent system is exponential in the number states required to describe the components. Hence, if we wish to make use of bounded time costs at the same time as variability then we must find appropriate abstractions. It is clear that direct modelling, in this style, will be intrinsically intractable. By contrast, using continuous time models all fixed times have to be approximated *ab intio* by some distribution, usually with the same mean.

4 Composing Distributions

To generate more natural distributions, whilst not incurring a heavy state cost, Erlang [Kle75,pp119-147, Nel95, pp153-280] considered sequential and probabilistic choice[5] compositions of identical geometric distributions. These give rise to two simple two parameter families of distributions. For our purposes we wish to consider three forms of composition on our distributions: sequential, nondeterministic and parallel. We shall define these compositions and then examine the possibility of finding families of distributions, or abstractions upon distributions that are maintained by these compositions. In particular we wish to demonstrate that our compositions are closed with respect to our basic distribution family.

4.1 Sequential Composition

We can sequentially compose two distributions as follows. Firstly, we need an auxiliary process to ensure that a $\overline{finish}$ occurs before the next *start* can be accepted:

$$Se \stackrel{def}{=} 1.start\#\overline{s}\#f\#\overline{finish} : Se + 1.start\#\overline{s} : Se1 + 1.\surd : Se$$
$$Se1 \stackrel{def}{=} 1.f\#\overline{finish} : Se + 1.\surd : Se1$$

The auxiliary process above exploits the known properties of the distribution processes it intends to compose. It takes the *start* request from the environment and itiates the first of the two underlying distribution processes. We shall exploit renaming to use the $\overline{finish}$ action of the first component of the sequence to initiate the second. Finally, when the second component terminates, which is renamed to $\overline{f}$, the system has completed and a $\overline{finish}$ signal can be sent to the environment. The reason we cannot simply leave the *start* action of the first component and use its $\overline{finish}$ to initiate the second component is that the system would then be capable of prematurely accepting a further *start* action, before the external $\overline{finish}$, violating the definition of a distribution process.

Now we can define a sequential composition of distribution processes:

Definition 4.1 *Given* $G1_0$ *and* $G2_0$ *are generalised distribution processes, their* sequential composition $G1_0 \,;\, G2_0$ *is defined by*

$$(G1_0[s/start, a/finish] \times Se \times G2_0[f/finish, a/start])\lceil\{start, finish\}$$

Proposition 4.2 *If* $G1_0$ *and* $G2_0$ *are the initial states of two generalised distribution processes, then* $G1_0 \,;\, G2_0$ *is a generalised distribution process*

Example 4. Let $E_1(p) \stackrel{def}{=} Geo(p)_0$ and $E_k(p) \stackrel{def}{=} E_1(p) \,;\, E_{(k-1)}(p)$, then the family $E_k(p)$, $k \geq 1$, are the Erlang k distributions.

Example 5. Earlier we stated that an interesting distribution family is the geometric after a fixed time. This can be formed as $Fix(t)_0 \,;\, Geo(p)_0$.

[5] Although he thought of his choice construction in terms of a parallel composition.

4.2 Probabilistic Choice

We give the definition for choosing between two distributions. Again, an auxiliary process for the construction:

$$\begin{aligned} Chse(p) &\stackrel{def}{=} p.start\#\overline{sL}\#f\#\overline{finish} : Chse(p) + p.start\#\overline{sL} : CW(p) \\ &\quad +(1-p).start\#\overline{sR}\#f\#\overline{finish} : Chse(p) \\ &\quad +(1-p).start\#\overline{sR} : CW(p) + 1.\surd : Chse(p) \\ CW(p) &\stackrel{def}{=} 1.f\#\overline{finish} : Cgse(p) + 1.\surd : CW(p) \end{aligned}$$

Comparing with the role of the auxiliary process for sequence, this again ensures that after a *start* action the $\overline{finish}$ action must proceed any further *start*.

Definition 4.3 *Let $G1_0$ and $G2_0$ be generalised distribution processes, then we can define their* non-deterministic composition $G1_0 +_p G2_0$ *thus:*

$$(G1_0[sL/start, f/finish] \times G2_0[sR/start, f/finish] \times Chse(p))\lceil\{start, finish\}$$

Proposition 4.4 *If $G1_0$ and $G2_0$ are the initial states of two generalised distribution processes, then $G1_0 +_p G2_0$ is a generalised distribution process.*

Example 6. Erlang's [Kle75] distribution family $H_k(p)$ is the choice between k identical geometric distributions, hence we can define

$$H_2(p) \stackrel{def}{=} Geo_0(p) +_{\frac{1}{2}} Geo_0(p)$$

4.3 Parallel Composition

We present the parallel composition of two distributions using two auxiliary processes. These respectively control the start and finish of the parallel composition:

$$\begin{aligned} PS &\stackrel{def}{=} 1.start\#\overline{s1}\#\overline{s2} : PS + 1.\surd : PS \\ PF &\stackrel{def}{=} 1.f1\#f2\#\overline{finish} : PF + 1.f1 : PF_1 + 1.f2 : PF_2 + 1.\surd : PF \\ PF_1 &\stackrel{def}{=} 1.f2\#\overline{finish} : PF + 1.\surd : PF_1 \\ PF_2 &\stackrel{def}{=} 1.f1\#\overline{finish} : PF + 1.\surd : PF_2 \end{aligned}$$

Again, these processes will ensure that both of the components have completed their activity before a second *start* action is permitted.

Definition 4.5 *Let $G1_0$ and $G2_0$ be two generalised distribution processes, then we can define their* parallel composition $G1_0 \,||\, G2_0$ *thus:*

$$(PS \times G1_0[s1/start, f1/finish] \times G2_0[s2/start, f2/finish] \times PF)\lceil\{start, finish\}$$

Proposition 4.6 *If $G1_0$ and $G2_0$ are the initial states of two generalised distribution processes then $G1_0 \,||\, G2_0$ is a generalised distribution process.*

Example 7. Consider the parallel composition of two geometric distributions $Geo(p)_0 \,||\, Geo(p)_0$. This is equivalent to the process:

$$\begin{aligned} G_0^2(p) &\stackrel{def}{=} p^2.start\#\overline{finish} : G_0^2(p) + 2p(1-p).start : G_2^2(p) \\ &\quad +(1-p)^2.start : G_1^2 + 1.\surd : G_0^2(p) \\ G_1^2(p) &\stackrel{def}{=} p^2.\overline{finish} : G_0^2(p) + 2p(1-p).\surd : G_2^2(p) \\ &\quad +(1-p)^2.start : G_1^2 \\ G_2^2(p) &\stackrel{def}{=} p.\overline{finish} : G_0^2(p) + (1-p).\surd : G_2^2(p) \end{aligned}$$

Notice that in general there is no q for which $Geo_0(p) \,||\, Geo_0(p)$ is even similar to $Geo(q)$, let alone identical.

4.4 Definedness

When we compose a collection of distributions together, then given that the properties of the constituent distributions are well defined, the properties of the compositions should be well defined.

Example 8. We define a distribution that is well defined in terms of probability and yet does not have a mean. Let $\frac{1}{\alpha} = \sum_{i=0}^{i=\infty} \frac{1}{i^2} = \frac{\pi}{6}$, and consider the following distribution:

$$P(t,G) \stackrel{def}{=} \begin{cases} \frac{\alpha}{k^2} \text{ if } t = 10^k \text{ for integer } k \\ 0 \quad \text{Otherwise} \end{cases}$$

Simple arithmetic demonstrates that $\sum_{t=1}^{\infty} P(t,G) = 1$, and $\sum_{t=1}^{\infty} tP(t,G) = \infty$.

Definition 4.7 *A distribution G is defined for weight k iff given $\epsilon > 0$ there exists an n such that $\sum_{t=n}^{\infty} t^k P(t,G) < \epsilon$*

Theorem 3. *Let $G1$ and $G2$ be two distributions which are defined for weight k then so are the distributions $G1\,;\,G2$, $G1 +_p G2$, and $G1 \,||\, G2$.*

5 Abstraction

In order that we can calculate efficiently over compositions of distributions, we wish to abstract representative information and calculate only with that. We start by defining a general notion of abstraction as a map between algebras. If we can find destination or abstraction algebras for which a homomorphism exists with our compositional system then we can form a congruence. Four forms of abstraction algebra that permit different kinds of calculation are then presented.

Definition 5.1 *We observe that our compositions form an algebra with sort functions $\mathcal{N} \to \mathcal{R}$, and operations $;$, $||$, and $+_p$. We call this algebra $\mathcal{DA}$.*

Definition 5.2 *An* abstraction *A is an algebra morphism $A : \mathcal{DA} \to X$ where X is an arbitrary algebra.*

Lemma 4. *An abstraction A induces a congruence over distributions iff it is an algebra homomorphism [MT91],in other words:*

$$\begin{aligned} A(D_1; D_2) &= A(;)(A(D_1), A(D_2)) \\ A(D_1 +_p D_2) &= A(+_p)(A(D_1), A(D_2)) \\ A(D_1 || D_2) &= A(||)(A(D_1), A(D_2)) \end{aligned}$$

Definition 5.3 *Two distribution processes are* abstraction equivalent *if the distributions they define are equivalent up to some abstraction A.*

Lemma 5. *If an abstraction induces a congruence with respect to distributions then it induces a congruence with respect to distribution processes.*

The obvious abstraction to take is that of means. However the following example shows that means are not preserved by parallel composition and therefore we will be unable to find an appropriate map.

Example 9. Consider a two point distribution with equal probability of taking 10 or 20 ticks (the unit of time). The mean for this distribution is 15 units of time. Composing two of these distributions in parallel we have two systems with the same mean. However the parallel distribution takes 10 with probability $\frac{1}{4}$ and 20 with probability $\frac{3}{4}$ and has a mean of 17.5. Now compare this with the parallel composition of two point distributions of 15, the parallel distribution has mean of 15 ticks. Hence, knowing the means of two distributions is insufficient information to predict the mean of their parallel composition and consequently this cannot (unfortunately) form an appropriate abstraction.

5.1 Min and Max

The abstraction which uses the minimum and maximum values a distribution can take, forms a congruence inducing abstraction. This is similar to duration analysis [Han94].

Definition 5.4 *Take as destination algebra* $(\mathcal{N}^2; ;_{mm}, +_{pmm}, ||_{mm})$ *with the operators defined as follows:*

$$\begin{aligned} ;_{mm}((mi_1, ma_1), (mi_2, ma_2)) &= ((mi_1 + mi_2, ma_1 + ma_2)) \\ +_{pmm}((mi_1, ma_1), (mi_2, ma_2)) &= (min(mi_1, mi_2), max(ma_1, ma_2)) \\ ||_{mm}((mi_1, ma_1), (mi_2, ma_2)) &= (max(mi_1, mi_2), max(ma_1, ma_2)) \end{aligned}$$

with the obvious implied sorts.

Now we define the homormorphism by mapping the operations to their analogues given above, and the distributions by taking the ordered pair formed from the least non-zero probability time and the maximum non-zero probability time.

Lemma 6. *The algebra morphism defined above is a homomorphism.*

Whilst this forms a congruence it does not contain any information about the variability of the system. Clearly in the above analysis we could take just the minima or the maxima individually and form a congruence.

5.2 Truncated distributions

Definition 5.5 *Consider a distribution D and a natural T (the limit time) then form $D\lceil T$ in the following fashion:*

$$D\lceil T(t) = \begin{cases} D(t) & \text{if } t \leq T \\ 0 & \text{otherwise} \end{cases}$$

The sort of $D\lceil T : [0,T] \to \mathcal{R}$ which we shall name Trun.

We can form an algebra of our truncated distribution with suitable operators.

Definition 5.6 *Consider three operators $;_T$, $+_{pT}$ and $||_T$:*

$$\begin{aligned} ;_T(T_1,T_2)(t) &= \sum_{i=0}^{i=t} T_1(i)T_2(t-i) \\ +_{pT}(T_1,T_2)(t) &= pT_1(t) + (1-p)T_2(t) \\ ||_T(T_1,T_2)(t) &= T_1(t)T_2(t) + T_1(t)\sum_{i=0}^{i<t} T_2(t-i) + T_2(t)\sum_{i=0}^{i<t} T_1(t-i) \end{aligned}$$

with the obvious sorts. We can form an algebra $(Trun; +_{pT}, ;_T, ||_T)$ and an abstraction A_T by mapping distributions to their truncations and operators to their truncated equivalents.

Lemma 7. *The algbera morphism defined above is a homomorphism.*

We can use the estimates for various parameters for the original distribution from the truncated distribution. In particular we can define the mean of this distribution.

Definition 5.7 *The mean μ_T of a truncated distribution $D\lceil T$:*

$$\mu_T = \sum_{t=0}^{T} tD(t)$$

Lemma 8. *If μ is the mean distribution D, with a well defined mean, then $\mu_T < \mu$ where μ_T is the mean of $D\lceil T$. Furthermore $lim_{T \rightsquigarrow \infty} \mu_T = \mu$.*

Hence the means derived from truncated distributions are lower bounds on the mean of the complete distribution. Similar results can be obtained for the other order weights of the system, allowing analysis of variance and other distribution properties.

5.3 Subsampled Truncated

Consider the following abstraction over a distribution.

Definition 5.8 *Let D be a distribution with $\alpha > 1$ and T a multiple of α naturals; we define a truncated subsampled $D_\alpha\lceil T$ as follows:*

$$D_\alpha\lceil T(t) = \begin{cases} \sum_{i=(t-1)\alpha+1}^{t\alpha} D(i) \text{ if } \alpha t \leq T \\ 0 \text{ otherwise} \end{cases}$$

the sort of $D_\alpha\lceil T : [0, \frac{T}{\alpha}] \to \mathcal{R}$ *which we shall name* $Trun_\alpha$.

We can now define an algebra using operators defined in an identical fashion as those for truncated distributions.

Lemma 9. *There is an algebra homomorphism between* $\mathcal{DA}$ *and the algbera* $(Trun_\alpha; +_{pT}, ;_T, ||_T)$.

Definition 5.9 *The mean* μ_{T_α} *of a subsampled truncated distribution* $D_\alpha\lceil T$*:*

$$\mu_{T_\alpha} = \sum_{t=0}^{\frac{T}{\alpha}} t\alpha D(t)$$

Lemma 10. *If* μ *is the mean of an aperiodic first weight defined distribution* D *then there exists* α *and* T *such that for all* $\alpha' \geq \alpha$ *and* $T' \geq T$ $\mu_{T'_{\alpha'}} \geq \mu$ *where* $\mu_{T'_{\alpha'}}$ *is the mean of* $D_{\alpha'}\lceil T$.

Hence the means derived from subsampled truncated distributions are in the limit upper bounds on the mean of the complete distribution.

5.4 Bounding Probability

One problem of the above descriptions is that we lose information in the sequence constructions. Since above the limit we can only calculate a *lower* bound on the probability of an event occuring at that time. If we could maintain all of the distribution then we could predict the information we needed about the components in order to calculate a certain limit on the distribution of a system.

Definition 5.10 *Given a distribution* D *then a* bounding probability *is a relation* $DB : (\mathcal{N} \to \mathcal{R}) \times (\mathcal{R} \times \mathcal{N})$ *such that* $(D, (p, t)) \in DB(D)$ *if* $\sum_{i=0}^{t} D(i) \geq p$

In other words, we can put (p, t) in the bounds of D, if the probability of the distribution D taking less than t is a least p. We can construct DB as a function with repsect to t insisting that a single choice of p is made for each t.

Definition 5.11 *We can define three operators* $;_{DB}$, $+_{pDB}$ *and* $||_{DB}$*:*

$$\begin{aligned} ;_{DB}((p_1, t_1), (p_2, t_2)) &= (p_1 p_2, t_1 + t_2) \\ +_{pDB}((p_1, t_1), (p_2, t_2)) &= (p_1 p_2, max(t_1, t_2)) \\ ||_{DB}((p_1, t_1), (p_2, t_2)) &= (pp_1 + (1-p)p_2, max(t_1, t_2)) \end{aligned}$$

with the obvious sorts.

Lemma 11. *There is a homomorphism between $\mathcal{DA}$ and the algebra* $((\mathcal{R} \times \mathcal{N}); +_{pDB}, ;_{DB}, ||_{DB})$.

Lemma 12. *Let D be a distribution formed from a system of n components (that is to say n applications of either sequence, parallel or choice) then to obtain $DB(D, p^n)$ we need to know $DB(D_i, p)$ for each component distribution D_i.*

For example given a system of 10 components and needing to know the 95% limit on the system performance we need to know to within 0.995 (since $0.995^{10} > 0.95$) the performance of each of the components.

Obviously we can define a cummulative distribution defined as a set of pairs of distribution bounds. Unfortunately, the behaviour of means derived from this abstraction can only be shown to tend to the true mean (when all times are sampled) it is unknown in what manner it tends to this limit. However, performance requirements are frequently expressed in terms of such bounds, so their efficient calculation is important.

6 Example

As a large scale example (in this case 19 components) consider a 3 level pipelined asynchronous processesor as described in Figure 1. We assume that each cycle requires a fetch, a decode and an execute. We assume that there is no contention for the fetch unit, although it is used 3 times in the system. In the diagram the costs for each of the processing elements are labelled.

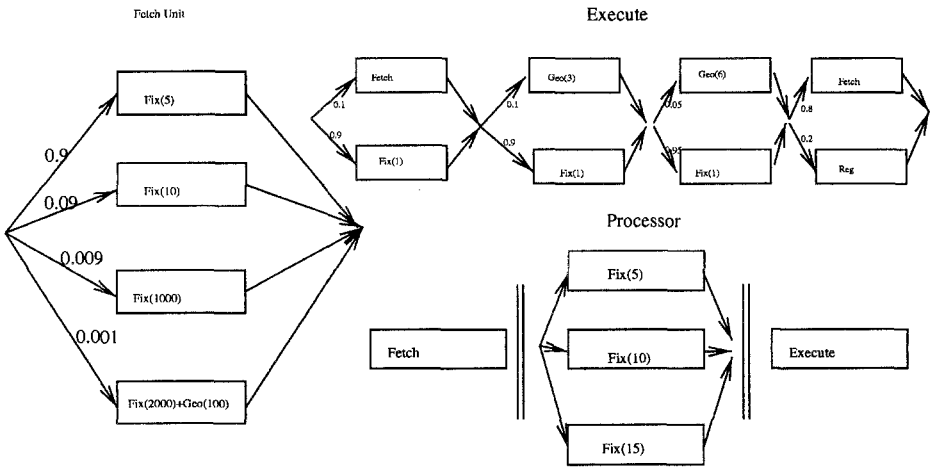

Fig. 1. The time costs of the components of the fetch and execute units of a prototypical processor, the final costs of the decode unit are included in the final composition.

The system presented would require 2001 states to describe the fetch unit, which appears independently three times in the design. The decode unit (between the fetch and the execute) requires 15, and the execute unit at least 2001^2 for the two fetch units it uses. The total number of states, as a phase distribution would be $120 * 10^6$. At this point the distribution would still have to be generated from the description, and given the number of states, it is unlikely that this distribution can be derived by exhaustive calculation. Hence, the distribution would have to be derived by some form of simulation, this rendering the formal model somewhat pointless.

The probability distribution for the first 5000 ticks of the system has been calculated. The data covers 99.99999% of the distribution and took 10 minutes of Sparc II time to calculate, the mean is 23.08. For comparison, the system was simulated, by generating samples from each of the underlying distributions, for 30 minutes and a mean of 22.53 obtained after 100000 samples were taken. Note that the calculational approach has achieved 100 times the accuracy in $\frac{1}{3}$ of the time.

Given the simulation was run 100000 times we should not expect to see events of greater rarity than 1 in 10^5. In the simulation of processors whose costs tend to be dominated by rare events, such as paging, the need to accurately represent the presence of these events is crucial. The calculational approach has covered events of order 1 in 10^7. It should be noted form the distributions that the occasional cost of 4000 ticks has a freqeuncy in the range 0.00001, and hence we could not expect a simulation run of 100000 repetitions to correctly estimate its consequences. Clearly a simulation is going to require prohibitive number of repetitions in order that it will achieve a reasonably accurate estimate of the performance of this system.

7 Conclusions and Further Work

We have demonstrated abstraction techniques that permit simple calculation of performance bounds on large scale systems. Furthermore, when dealing with unbounded distributions, it is possible to predict in advance what quality of bound can be obtained given a certain degree of knowledge of the performance of the components of the system. We have shown two truncation methods that give us respectively upper and lower bounds on the average case performance of a system. Whilst we presented our abstractions in the context of discrete distributions, they would work equally well if the underlying distributions were continuous. This work underpins, by providing an efficient calculation strategy on distribution compositions, work using a greater level of abstraction to describe systems. In this presentation we have omitted details of how data dependency can be modelled. Whilst this can be achieved in the usual way [Mil90] we leave it for a more abstract approach in which both data dependency and conflict can be modelled.

The notion of equivalence between distributions processes here, clearly has applications in the wider context of a probabilistic calculus. In particular ab-

stracting over the particular paths between two events and concentrating on the distribution of the time. Furthermore, the ability to perform symbolic calculations with distributions permits a far greater flexibility in addressing performance problems and may be achievable with further abstractions justified by these underlying calculational techniques. The general framework for defining abstractions over distributions which we have presented permits the efficient study of other possible abstraction methods. Calculations over the two truncated distribution methods have been automated, alongside a direct simulator for the same systems permitting comparison, this program (in SML) can be obtained from the author.

8 Bibliography.

[CH87] I. Castellani and M. Hennessey, Distributed Bisimulation, Report Sussex University 5/87, July 1987.

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

[Han91] H. Hannsson, Time and probability in Formal Design of Distributed Systems PhD Thesis, Department of Computer Systems Uppsala University Uppsala Sweden. TR DoCS 91/27. 1991

[Han94] M.R. Hansen, Model Checking Discrete Duration Calculus, FACS 6A:826-845, 1994.

[Hil94] J. Hilston, A Compositional Approach to Performance Modelling, PhD Thesis, Department of Computer Science, University of Edinburgh, 1994.

[HJ94] H. Hansson and B. Jonsson, A Logic for Reasoning about Time and Reliability, FACS (6):512-535, 1994.

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

[HM93] J. Hilston and F. Moller Proceedings of 1st Conference on Process Algebra and Performance Modelling, Edinburgh Computer Science Departmental Report, 1993

[HR94] U. Herzog and M. Rettlebach, Proceedings of 2nd Conference on Process Algebra and Performance Modelling, 1994

[Kle75] L. Kleinrock, Queueing Systems, Volumes I and II, John Wiley, 1975.

[Mil80] R. Milner, Calculus of Communicating System, LNCS92, 1980.

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

[Mil90] R. Milner, Communication and Concurrency, Prentice Hall, 1990.

[MT91] K. Meinke and J.V. Tucker, Universal Algebra, pp189-411 in Hanbook of Logic in Computer Science, ed S. Abramsky, D. Gabbay and T. S. E. Maibaum, Oxfor University Press, 1991.

[Nel95] K. Nelson, Probability, Stochastic Processes and Queueing Theory, Springer Verlag, 1995.

[Paz71] A. Paz, Introduction to Probabilistic Automata, Academic Press, 1971.

[SS90] S. Smolka and B. Steffen, Priority as Extremal Probability, Proceedings Concur '90, LNCS 458, 1990.

[Tof90] C. Tofts, A Synchronous Calculus of Relative Frequency, CONCUR '90, Springer Verlag, LNCS 458.

[Tof93] C. Tofts, Exact Solutions to Finite State Simulation Problems, Research Report, Department of Computer Science, University of Calgary, 1993.

[Tof94] C. Tofts, Processes with Probabilities, Priorities and Time, FACS 6(5): 536-564, 1994.

[Tof95] C. Tofts, Exact and Approximate Analytic Solutions of Properties of Probabilistic Processes, Proceedings first TACAS 1995, LNCS 1017.

[VW92] S. F. M. van Vlijmen, A. van Waveren, An Algebraic Specification of a Model Factory, Research Report, University of Amsterdam Programming Research Group, 1992.