LFCS

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

The Autosynchronisation of Leptothorax Acervorum (Fabricius) Described in WSCCS

by

Chris Tofts

LFCS Report Series ECS-LFCS-90-128

LFCS December 1990

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

The Autosynchronisation of *Leptothorax Acervorum* (Fabricius) Described in WSCCS.

Chris Tofts,
Laboratory for the Foundations of Computer Science,
Department of Computer Science,
University of Edinburgh.
e-mail:cmnt@uk.ac.ed.lfcs

December 13, 1990

Abstract

We present an introduction to the calculus WSCCS. The calculus is then used to study a probabilistic synchronisation algorithm; which may underly the synchronous behaviour of a particular species of ant. We demonstrate that the cyclicity is stable and that the system must eventually synchronise. We present several estimates of how long it will take such a system to synchronise. The stability of the algorithm to some forms of error is discussed, and we present a derived algorithm which has a high tolerance of error. Some experimental results are reported, which show that our explanation may be a reasonably accurate account of the interactions underlying the biological phenomena.

1 Introduction.

There are many physical systems, consisting of large numbers of individuals, that exhibit synchronised behaviour without refrence to any external signal or clock. These include both bio-chemical systems [NP77] and many biological systems [May73, PDG87, DGPFL87, WH88, See87]. Of particular interest in this paper is the ant species *Leptothorax Acervorum* [FB87, FBGH89], which exhibits 'bursts' of activity within the nest. Individual workers spend aprroximately 70% of their time inactive and then are generally all active together.

This phenomenom is of particular interest owing to the following properties;

- it is unaffected by individuals dying;
- it is highly decentralised, there is no identifiable essential ant;

- it is unaffected by ants leaving the colony and then rejoining.

Whilst some may argue that the queen is an identifiable and unique ant within a colony, the phenomenom still occurs in her absence[FB87].

Due to the complexity of the system the cycles of activity will *not* be a simple sine-wave with a fixed period. They will have a period (hopefully) distributed according to some simple probabilistic definition given by the interactive behaviour of the individuals.

The main reason for describing such a system within a process algebra [Mil89, BBK86, Hoa85], is that we can express precisely the behaviour of an individual and from that deduce precisely the behaviour of a collection of individuals interacting. The language we shall be using to describe ants is the calculus WSCCS [Tof90], a probabilistic extension of Milner's SCCS [Mil83].

We start be giving a brief description of two other models of this acitivity and then give an introduction to the language WSCCS. For a fuller comparison of these models see [THF90]. The rest of the paper consists of the analysis of the behaviour of a simple system of interacting ants described in WSCCS.

1.1 Simulation Model.

In [GD88] a model of the behaviour of colonies based on autocatalysis is presented. This model presupposes the following behaviour of an individual;

- once it falls asleep it will stay asleep for some fixed period whence it will become wakeable;
- at each instant a wakeable ant can wake spontaneously with some fixed probability;
- a woken ant will move at random, waking any wakeable ants it 'runs' into;
- at each instant an awake ant can fall asleep with some fixed probability.

A collection of ants, obeying the constarints above, were simulated using a small computer. The simulation has bursts of activity similar to those seen in real nests.

1.2 Energy Model.

In [HBF89] a model based on the amount of energy within a nest is presented. It relies on describing the amount of energy (E) held within the nest and the number of active ants (A) as differential equations;

$$\frac{dE}{dt} = aA - bE \qquad (a, b > 0)$$

$$\frac{dA}{dt} = \frac{c}{h+E}(eA^2 + fNA + N^2)(N - A) - gA$$

where N is the total number of ants within the nest. For particular values of the constants a, b, c, e, f, g, h these equations show cyclic behaviour of the number of active ants A.

2 The Language WSCCS.

Our language WSCCS is an extension of Milner's SCCS [Mil83] a language for describing synchronous concurrent systems. 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 $\mathcal{P}$ augmented with the usual infinite object ω, and a set of process variables Var.

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 $S(\mathbf{1}) = \mathbf{1}$ and $\overline{S(a)} = S(\overline{a})$, action sets $A \subseteq Act$, with $\mathbf{1} \in A$, and arbitrary *finite* indexing sets I:

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

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 experments 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 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 $1a + 2b$. 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 $\mu x.ax$.

2.1 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 $E \stackrel{a}{\longrightarrow} F$ 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 processes 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.

2.1.1 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.

Definition 2.1 *Let S be a set of processes then:*

- $P \stackrel{w}{\longmapsto} S$ *with* $w = \sum\{w_i | P \stackrel{w_i}{\longmapsto} Q$ *for some* $Q \in S\}$; [2]
- $P \stackrel{a}{\longrightarrow} S$ *iff there exists* $Q \in S$ *and* $P \stackrel{a}{\longrightarrow} Q$.

[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.

[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.

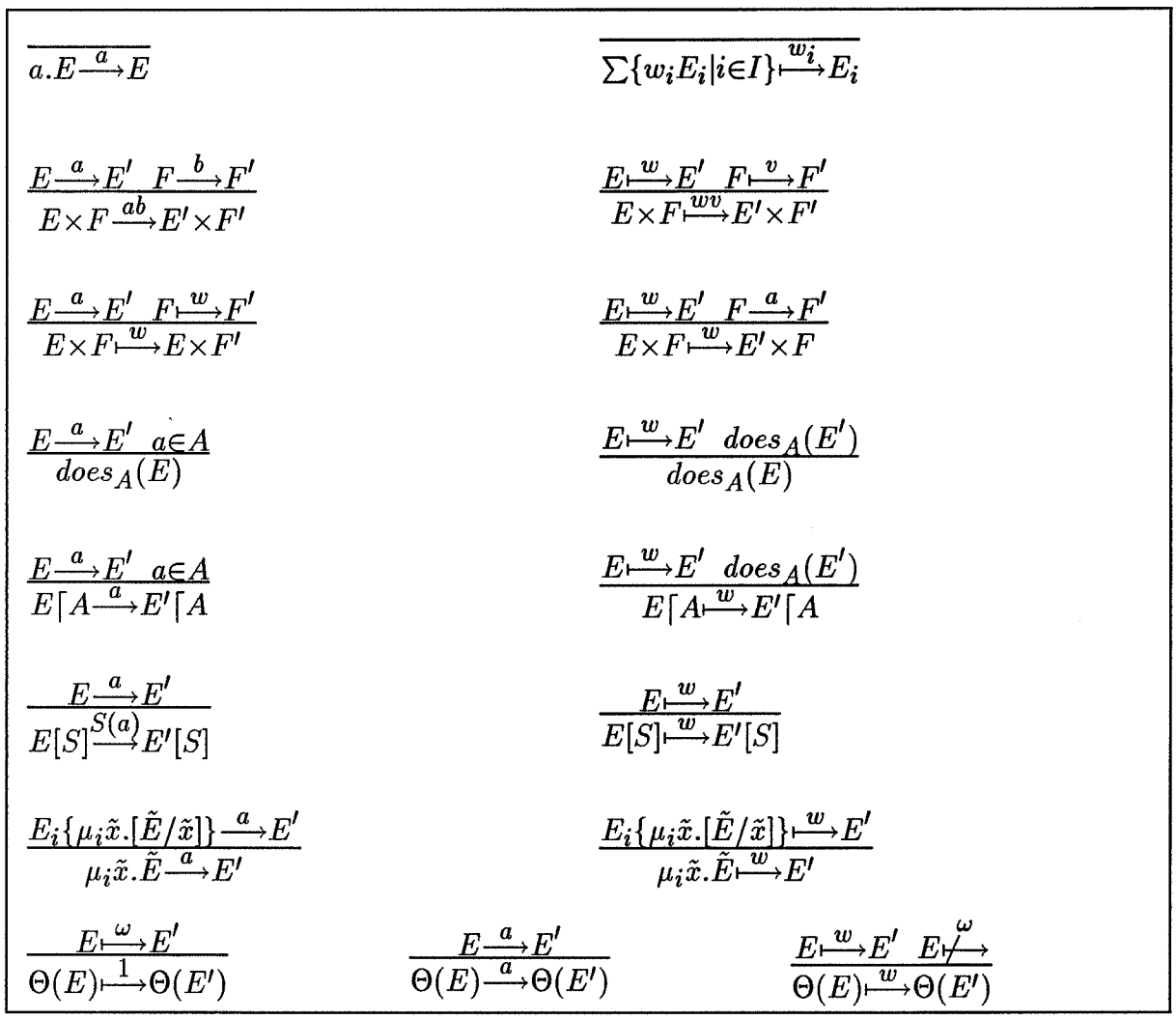

Figure 1: Operational Rules for WSCCS.

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 2.2 *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 $w \in \mathcal{W}$, $P \stackrel{w}{\longmapsto} S$ iff $Q \stackrel{w}{\longmapsto} S$;*
- *for all $a \in Act$, $P \stackrel{a}{\longrightarrow} S$ iff $Q \stackrel{a}{\longrightarrow} S$.*

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

Definition 2.3

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

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

Lemma 2.4 *Let P and Q be processes such that $P \stackrel{d}{\sim} Q$. Then for all action sets A, $does_A(P)$ iff $does_A(Q)$.*

Proposition 2.5 *Direct equivalence is substitutive for finite processes. Thus, given $P \stackrel{d}{\sim} Q$ and $P_i \stackrel{d}{\sim} Q_i$ for all $i \in I$ then:*

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

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

Definition 2.6 *Let $\tilde{E}$ and $\tilde{F}$ be expressions containing variables at most $\tilde{X}$. Then we will say $\tilde{E} \stackrel{d}{\sim} \tilde{F}$ if for all process sets $\tilde{P}$, $\tilde{E}\{\tilde{P}/\tilde{X}\} \stackrel{d}{\sim} \tilde{F}\{\tilde{P}/\tilde{X}\}$.*

Proposition 2.7 *If $\tilde{E} \stackrel{d}{\sim} \tilde{F}$ then $\mu_i \tilde{X}.\tilde{E} \stackrel{d}{\sim} \mu_i \tilde{X}.\tilde{F}$.*

[3] We denote the equivalence class of a process 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.

2.1.2 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 definition.

Definition 2.8 *We say an equivalence relation $R \subseteq Pr \times Pr$ is a* relative bisimulation *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 $w, v \in \mathcal{W}$, $P \stackrel{w}{\longmapsto} S$ iff $Q \stackrel{v}{\longmapsto} S$ and $c_1 w = c_2 v$;*
2. *for all $S \in Pr/R$ and for all $a \in Act$, $P \stackrel{a}{\longrightarrow} S$ iff $Q \stackrel{a}{\longrightarrow} S$.*

Two processes are relative bisimulation equivalent, *written $P \stackrel{r}{\sim} Q$ 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 2.9

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

Proposition 2.10 *Let P and Q be processes such that $P \stackrel{d}{\sim} Q$, then $P \stackrel{r}{\sim} Q$.*

Definition 2.11 *Let $\tilde{E}$ and $\tilde{F}$ be expressions containing variables at most $\tilde{X}$. Then we will say $\tilde{E} \stackrel{r}{\sim} \tilde{F}$ if for all process sets $\tilde{P}$, $\tilde{E}\{\tilde{P}/\tilde{X}\} \stackrel{r}{\sim} \tilde{F}\{\tilde{P}/\tilde{X}\}$.*

Proposition 2.12 *$\stackrel{r}{\sim}$ is a congruence for finite and infinite processes.*

We would like a notion of equivalence that permits us to disregard the structure of the choices and just look at the total chance of reaching any particular state. This is known *not* to produce a congruence [SST89], but is a useful notion of equivalence.

Definition 2.13 *We define an abstract notion of evolution as follows;*

$$P \xrightarrow{a[w]} P' \text{ iff } P \xrightarrow{w_1} \mapsto \ldots \xrightarrow{w_n} \mapsto \xrightarrow{a} P' \text{ with } w = w_1 \ldots w_n.$$

In order to define an equivalence which uses such transitions we need a notion of accumulation.

Definition 2.14 *Let S be a set of processes then:*

$$P \xrightarrow{a[w]} S \text{ iff } w = \sum\{w_i | P \xrightarrow{a[w_i]} Q \text{ for some } Q \in S\};$$ [4]

We can now define an equivalence that ignores the choice structure but not the choice values.

Definition 2.15 *We say an equivalence relation $R \subseteq Pr \times Pr$ is an* abstract bisimulation *if $(P, Q) \in R$ implies that:*

there are $c_1, c_2 \in \mathcal{P}$ such that for all $S \in Pr/R$ and for all $w, v \in \mathcal{W}$, $P \xrightarrow{a[w]} S$ iff $Q \xrightarrow{a[v]} S$ and $c_1 w = c_2 v$.

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

2.2 Equational Characterisation of WSCCS.

We present some equational laws over WSCCS processes in Figure 2, these form a sound and complete equational system over the finite processes in WSCCS. We shall write $p = q$ for $p \overset{r}{\sim} q$.

Definition 2.16 *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 $d_A(a.E)$;*
- *If there exists $i \in I$ with $d_A(E_i)$ then $d_A(\Sigma_{i \in I} w_i E_i)$.*

[4] 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.

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

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

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

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

$$(\Theta_1)\ \Theta(a.E) = a.\Theta(E)$$

$$(\Theta_2)\ \Theta(\Sigma_{i\in I} w_i E_i) = \begin{cases}\Sigma_{j\in J} 1.\Theta(E_j) \text{ where } J = \{i \in I | w_i = \omega\} \text{ and } J \neq \emptyset,\\ \Sigma_{i\in I} w_i \Theta(E_i) \text{ if } J = \emptyset\end{cases}$$

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

Figure 2: Equational rules for WSCCS.

3 Ants in WSCCS.

We assume a description of the ant as follows (note this is a simplification of the description given in [GD88]:

- It will sleep for a determined period.
- It will then become wakeable at which time it may wake up.
- If it wakes up it will wake any other wakeful ant.
- Having woken up the ant will then fall asleep again.

With this intention we end up with the following process describing an ant.

$$\begin{aligned}
&Sleep(k) = \mathbf{1}.\mathbf{1}.Sleep(k-1)\\
&Sleep(0) = m.\mathbf{1}.Wakeable + n.\mathbf{1}.Woken\\
&Woken(z) = \omega.\overline{a}^{z}.Sleep(s) + 1.(\omega.\overline{a}^{z-1}.Sleep(s) + \cdots 1.(\omega.\overline{a}.Sleep(s) + 1.\mathbf{1}.Sleep(s)))\\
&Wakeable = \underbrace{1 \ldots 1.}_{z-times}(1.a.Sleep(s) + 1.\mathbf{1}.Sleep(0))
\end{aligned}$$

The free parameters in the above set of proceses have the following meaning:

- z is the number of ants in the nest.
- s is the sleeping time.

- $\frac{n}{m+n}$ is the probability of any individual wakeable ant waking,
- and hence $\frac{m}{m+n}$ is the probability of it staying wakeable.

There are two points of interest in the above process; one is the use of odd and even time steps together with ticks to stop the bi-conditioning of the wakeup point; the multiple 1 guards on the wakeup signal receiver are there for the same purpose.

An ant nest is the following process:

$$Ant_i = Sleep(i)$$
$$Nest(z) = \Theta((Ant_{i1} \times Ant_{i2} \times \cdots \times Ant_{iz}) \lceil\{\mathbf{1}\})$$

where the $i1, \cdots, iz$ parameters specify how long it will be before a particular ant can wake up for the first time.

4 Properties of the Colony.

4.1 Synchronisation is stable.

The particular case of a nest which is synchronised is of greatest interest and that is the process;

$$Snest(z,s) = \Theta((Ant_s \times Ant_s \times \cdots \times Ant_s) \lceil\{\mathbf{1}\}).$$

We will now show that once a nest is synchronised it will stay synchronised.

$$Snest(z,s)(\xrightarrow{\mathbf{1}})^{2s-times} Snest(0,s)$$
$$Snest(z,0) = \Theta((Sleep(0) \times \cdots \times Sleep(0))\lceil\{\mathbf{1}\})$$
$$Snest(z,0) = m^z.\mathbf{1}.\underbrace{1\ldots1}_{z-times}.\mathbf{1}.Snest(z,0) + \binom{z}{1} m^{(z-1)} n.\mathbf{1}.\underbrace{1\ldots1}_{z-times}.\mathbf{1}.Snest(z,s)$$
$$+ \cdots + \binom{z}{z} n^z.\mathbf{1}.\underbrace{1\ldots1}_{z-times}.\mathbf{1}.Snest(z,s)$$

Equally we could represent the above as a transition matrix with the transitions being pairs of ticks, and for a colony with 7 sleep states (numbered by the amount of time left to wake up) we obtain the following.

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 - \frac{m^z}{(m+n)^z} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{m^z}{(m+n)^z} \end{pmatrix}$$

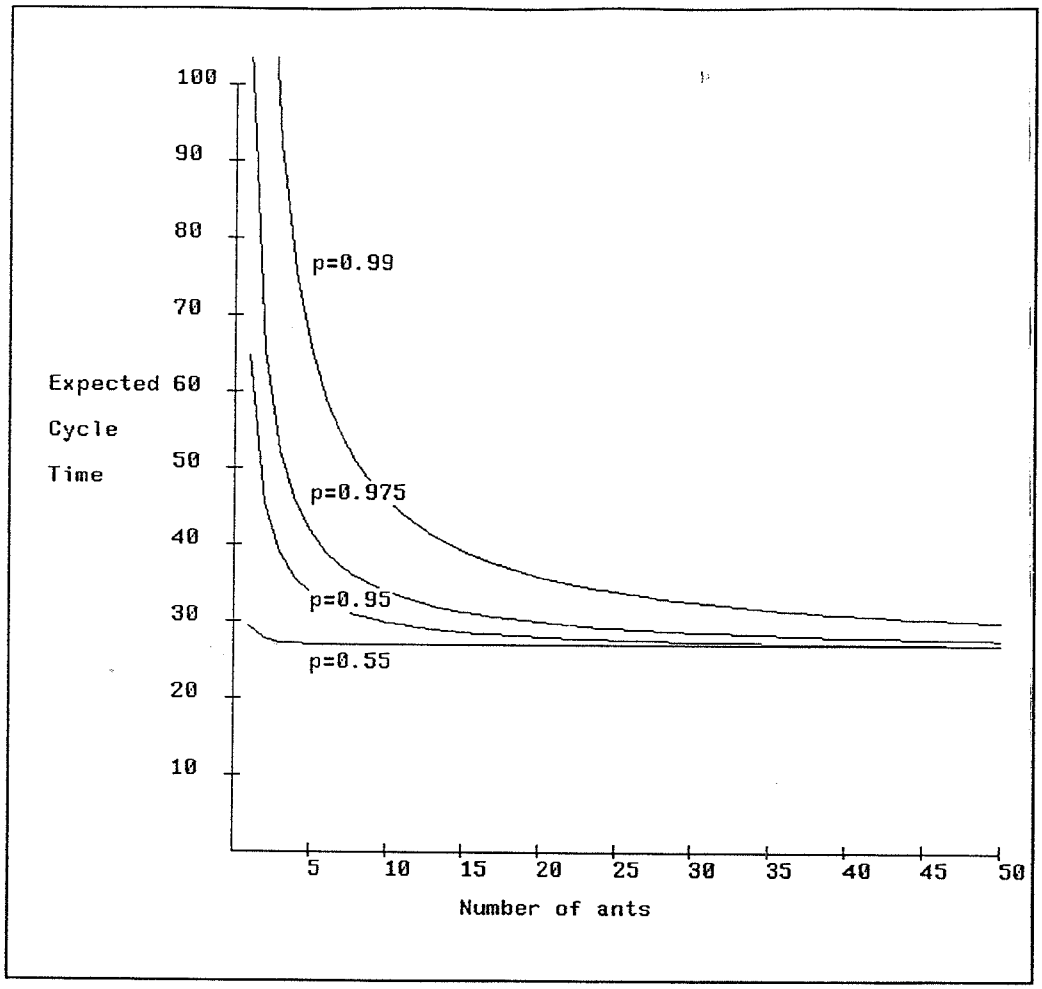

Figure 3: Expected cycle length against Colony size.

In other words a synchronised nest behaves like a single ant, excepting that the probability (p) of a nest staying asleep at each turn once it can potentially wake up is $\frac{m^z}{(n+m)^z}$ as we would expect.

The expected cycle time and variance in cycle time, in terms of the probability p of a nest staying asleep once it is wakeable, are given respectively by the formulae;

$$s + \frac{1}{1-p} \qquad \frac{p}{(1-p)^2}$$

In figure 3 we demonstrate that the precise probability of an invdividual waking has little effect on the cycle time of resonably sized colonies.

4.2 A Colony Always Synchronises.

We have shown that the synchronised state is absorbing. We will now demonstrate that there are no other absorbing cycles or states establishing that the nest must always eventually synchronise.

Definition 4.1 *The usual* monus *operator is defined as follows:*

$$n \dot{-} m = \begin{cases} n - m \text{ if } m \leq n \\ 0 \text{ otherwise} \end{cases}$$

Consider the evolutions of the following process;

$$\Theta((Ant(0) \times \ldots Ant(0) \times Ant(n_i) \times \ldots \times Ant(n_k))\lceil\{\mathbf{1}\})$$

with $0 \leq n_i \leq n_k$. Now this has the following evolution path. We will use relativised weights in this section to save time (writing the probability of an individual ant staying asleep as p).

$$\begin{array}{c} \Theta((Ant(0) \times \ldots Ant(0) \times Ant(n_i) \times \ldots \times Ant(n_k))\lceil\{\mathbf{1}\}) \\ \downarrow p^{(i-1)} \\ \downarrow \mathbf{1} \\ \downarrow \mathbf{1} \\ \Theta((Ant(0) \times \ldots Ant(0) \times Ant(n_i \dot{-} 1) \times \ldots \times Ant(n_k \dot{-} 1))\lceil\{\mathbf{1}\}) \\ \vdots (n_k \dot{-} 1 \text{ times}) \\ \Theta((Ant(0) \times \ldots Ant(0) \times Ant(0) \times \ldots \times Ant(0))\lceil\{\mathbf{1}\}) \end{array}$$

Hence we can deduce that the probability of a state with at least one ant wakeable in it synchronising (without our particular ant waking) is,

$$p^{(i-1)n_k} p^{(n_k - n_i)} \ldots p^{(n_{(k-1)} - n_k)}$$

which is non-zero for non-zero sleep probability.

Now consider a more general state,

$$\Theta((Ant(n_1) \times Ant(n_2) \times \ldots \times Ant(n_k))\lceil\{\mathbf{1}\})$$

with $0 \leq n_1 \leq n_2 \cdots \leq n_k$. This can only evolve through pairs of ticks until the following state is reached,

$$\Theta((Ant(0) \times \ldots Ant(0) \times Ant(n_i - n_1) \times \ldots \times Ant(n_k - n_1))\lceil\{\mathbf{1}\})$$

which is precisely the state we considered above.

In figures 4 5, we include two simulations of six ants with 10 sleep states with differing wake up probabilities to demonstrate the eventual synchronisation of the nest.

Whilst the size of a transition matrix for such a system is large, having approximately[5] $(s+1)^{2k}$ entries, we include a small example to indicate its structure we will use 3 ants with two sleep states. Numbering the states of

[5] We do not take account of states that are equivalent with respect to associativity and commutivity of times.

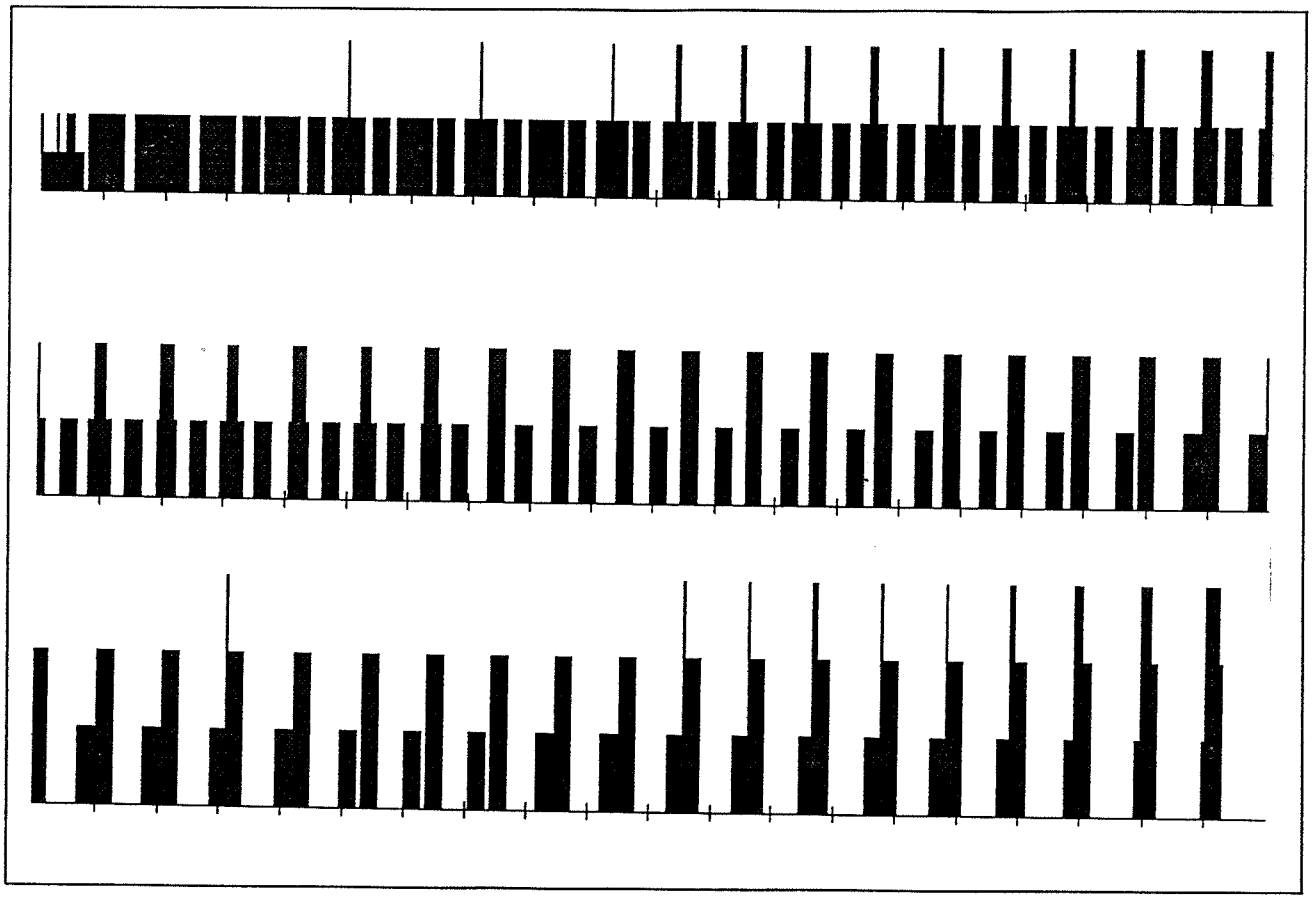

Figure 4: Six ants synchronsing I

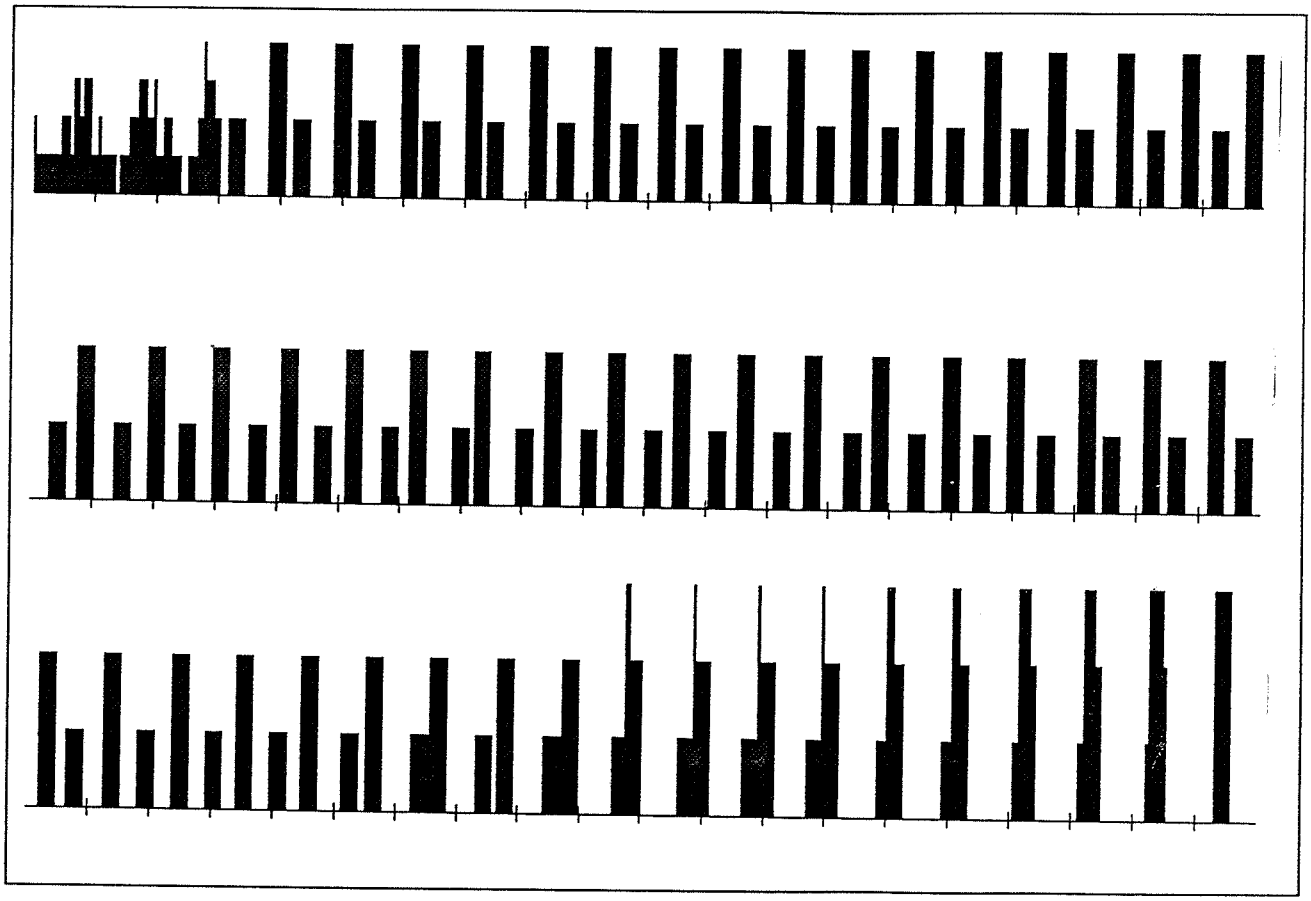

Figure 5: Six ants synchronsing II

$$\Theta((Ant(x) \times Ant(y) \times Ant(z))\lceil\{\mathbf{1}\})$$

as (2,2,2),(1,1,1),(0,0,0),(1,2,2),(0,1,1),(1,1,2),(0,0,1),(0,0,2),(0,2,2) and (0,1,2) respectively.

$$\begin{pmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1-p^3 & 0 & p^3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & p & 0 & 0 & 0 & 1-p & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & p^2 & 0 & 0 & 0 & 0 & 0 & 1-p^2 & 0 \\
0 & 0 & 0 & 1-p^2 & 0 & 0 & p^2 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & p & 1-p & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & p & 0 & 0 & 0 & 1-p
\end{pmatrix}$$

Larger versions have a similar structure, with a closed cycle, and other states that can only evolve towards states that can possibly synchronise. With that in mind we can consider the reduced matrix were we only look a synchronisation versus non-synchronised, and do not attempt to keep the transit times between states equal. Thus the above would only have the following states (0,0,0),(0,1,1),(0,0,1),(0,0,2),(0,2,2) and we obtain the following transition matrix.

$$\begin{pmatrix}
1 & 0 & 0 & 0 & 0 \\
p & 0 & 0 & 1-p & 0 \\
p^2 & 0 & 0 & 0 & 1-p^2 \\
p^4 & (1-p^2) & 0 & 0 & p^2(1-p^2) \\
p^2 & 0 & 0 & 1-p^2 & 0
\end{pmatrix}$$

The transition matrices for these systems are very sparse, the language WSCCS gives us an efficient representation of such as we only give those transistions that exist thus avoiding all the zero transitions.

4.3 Exact time to Synchronise for 2 ants.

We should like to know how long it takes two of our ants starting an arbitary distance apart to synchronise. If we consider only those states with at least one zero sleep time then we obtain the following derivations.

$$\begin{array}{ccc}
\Theta((Ant(0) \times Ant(1))\lceil\{\mathbf{1}\}) & & \Theta((Ant(0) \times Ant(1))\lceil\{\mathbf{1}\}) \\
\downarrow p & & \downarrow 1-p \\
\downarrow \mathbf{1} & & \downarrow \mathbf{1} \\
\downarrow \mathbf{1} & & \downarrow \mathbf{1} \\
\Theta((Ant(0) \times Ant(0))\lceil\{\mathbf{1}\}) & & \Theta((Ant(0) \times Ant(s))\lceil\{\mathbf{1}\})
\end{array}$$

For a more general state ($k \leq s$),

$$\begin{array}{c} \Theta((Ant(0) \times Ant(k))\lceil\{\mathbf{1}\}) \\ \downarrow p \\ \downarrow \mathbf{1} \\ \downarrow \mathbf{1} \\ \Theta((Ant(0) \times Ant(k-1))\lceil\{\mathbf{1}\}) \end{array} \qquad \begin{array}{c} \Theta((Ant(0) \times Ant(k))\lceil\{\mathbf{1}\}) \\ \downarrow 1-p \\ \downarrow \mathbf{1} \\ \downarrow \mathbf{1} \\ \Theta((Ant(k-1) \times Ant(s))\lceil\{\mathbf{1}\}) \\ \left.\begin{array}{l} \downarrow \mathbf{1} \\ \downarrow \mathbf{1} \end{array}\right\} k-1 \text{ times} \\ \Theta((Ant(0) \times Ant(s-k+1))\lceil\{\mathbf{1}\}) \end{array}$$

Thus writing $E(k)$ as the expected time to synchronise for two ants starting a sleep time k apart we obtain the following simultaneous equations.

$$\begin{aligned} E(0) &= 0 \\ E(1) &= 2p + (1-p)(E(s)+2) \\ E(k) &= p(2+E(k-1)) + (1-p)(E(s-k+1)+2k) \end{aligned}$$

We now present the solutions for when we have 3 and 4 sleep states (writing $q = 1-p$ for brevity).

$$\begin{aligned} E(1) &= 2p + qE(3) + 2q \\ E(2) &= 2p + pE(1) + qE(2) + 4q \\ E(3) &= 2p + pE(2) + qE(1) + 6q \end{aligned}$$

$$E(1) + E(2) + E(3) = 6p + qE(3) + E(2) + E(1) + 12q.$$

Using the above we obtain,

$$\begin{aligned} E(1) &= 2 + \tfrac{1}{p}(q(6p+12q)) \\ E(2) &= \tfrac{1}{p}(2 + p(2 + \tfrac{1}{p}(q(6p+12q))) + 2q) \\ E(3) &= \tfrac{1}{p}(6p+12q). \end{aligned}$$

Similarly for 4 sleep states we obtain,

$$\begin{aligned} E(1) &= 2 + \tfrac{1}{p}(q(8p+20q)) \\ E(2) &= \tfrac{1}{p}(2 + q(8p+20q) + 4q + 4q^2) \\ E(3) &= \tfrac{1}{p}(2 + q(8p+20q) + 4q + 4q^2) + 2 + 4q \\ E(4) &= \tfrac{1}{p}(8p+20q) \end{aligned}$$

the solutions to the above equations are presented in graphical form in figure 6.

In general we cannot solve for all the times but we can solve for the time to synchronise starting as far apart as possible. Observe that given s sleep states,

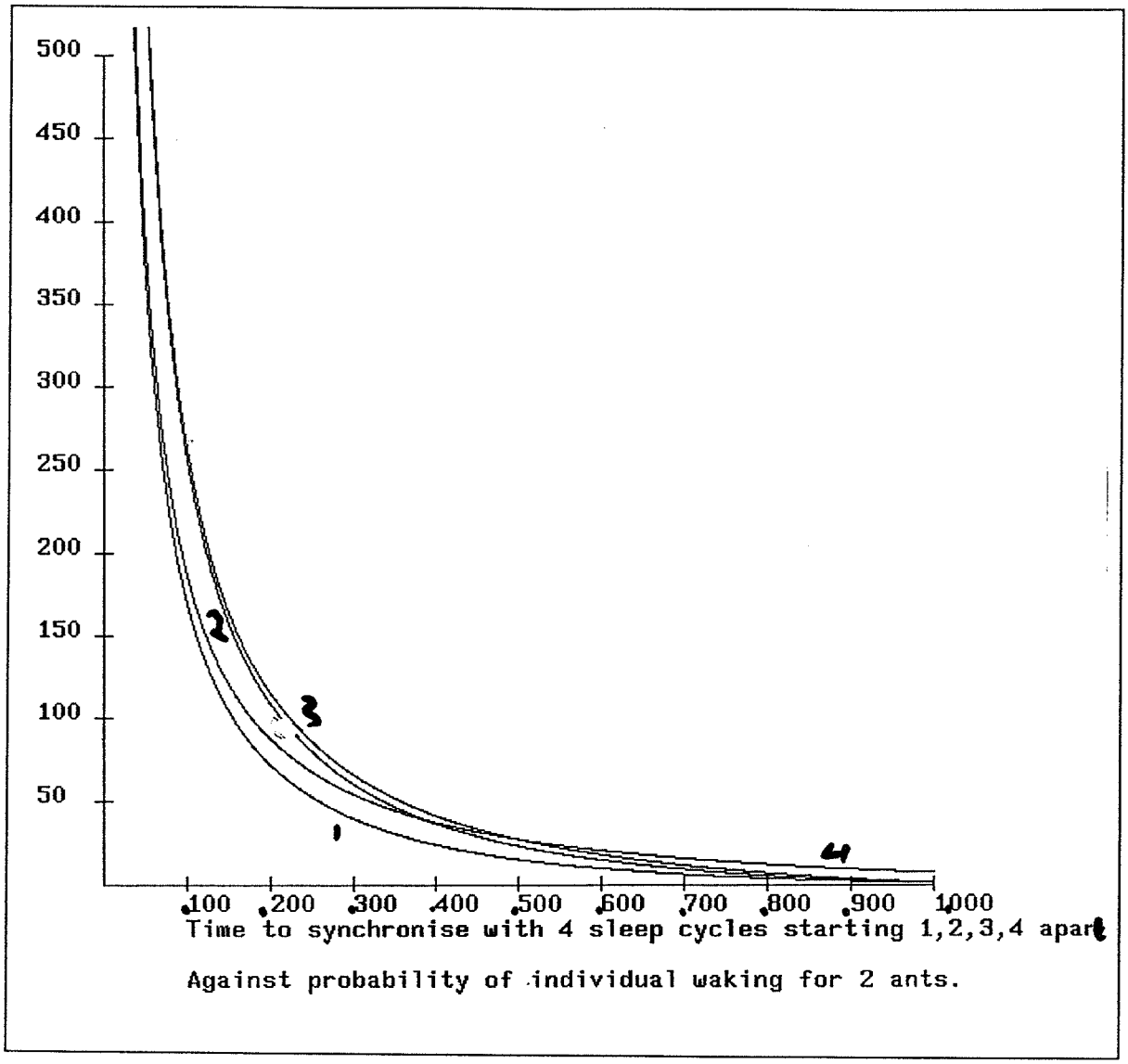

Figure 6: Expected time to syncronise given 4 sleep states.

$$E(1)+\cdots+E(s)=2sp+E(1)+\cdots+qE(s)+s(s+1)q$$

and thus we obtain,

$$E(s)=\tfrac{1}{p}(2sp+s(s+1)q)$$

We can examine the ratio between time to synchronise and cycle time with respect both to the number of sleep states, and the probability of an individual staying asleep. These results are presented in figures 7 8 respectively.

4.4 Use of groups.

We will change our defintion of the ant slightly to the following;

$$\begin{aligned}
&Sleep(k)=\mathbf{1}.\mathbf{1}.Sleep(k-1)\\
&Sleep(0)=m.\mathbf{1}.Wakeable+n.\mathbf{1}.Brd(z)\\
&Brd(j)=\omega.\overline{a}^{z}.Sleep(s)+1.Brd(j-1)\\
&Brd(1)=\omega.\overline{a}.Sleep(s)+1.\mathbf{1}.Sleep(s)\\
&Wakeable=\underbrace{1\ldots1.}_{z-times}(1.a.Sleep(s)+1.\mathbf{1}.Sleep(0))\\
&Ant=Sleep(s).
\end{aligned}$$

Also defining an alternative ant as;

$$\begin{aligned}
&Slp(k)=\mathbf{1}.\mathbf{1}.Slp(k-1)\\
&Slp(0)=m^{i}.\mathbf{1}.Wkable+((n+m)^{i}-m^{i}).\mathbf{1}.Brod(z)\\
&Brod(j)=\omega.\overline{a}^{z}.Slp(s)+1.Brod(j-1)\\
&Brod(1)=\omega.\overline{a}.Slp(s)+1.\mathbf{1}.Slp(s)\\
&Wkable=\underbrace{1\ldots1.}_{z-times}(1.a.Sleep(s)+1.\mathbf{1}.Slp(0))\\
&Ant'=Slp(s).
\end{aligned}$$

Definition 4.2 *The i'th power of a process P, written P^{i}, is the process;*

$$\underbrace{P\times P\times\cdots\times P}_{i-times}$$

We would like to establish that the processes, Ant^{z} and Ant', are essentially the same. It is clear they they are not immediately equivalent since Ant^{i} can produce the action $\overline{a}^{zi}$ whilst the highest power of $\overline{a}$ that Ant' can produce is $\overline{a}^{z}$. If we define the following set of actions,

$$A=\{\overline{a}^{z},\cdots,\overline{a},a^{z},\cdots,a,\mathbf{1}\}$$

and consider the processes $Ant^{i}\lceil A$ and $Ant'\lceil A$. The interesting case to consider is $(Sleep(0)^{i})\lceil A$ and $(Slp(0))\lceil A$. Examining the evolution pair

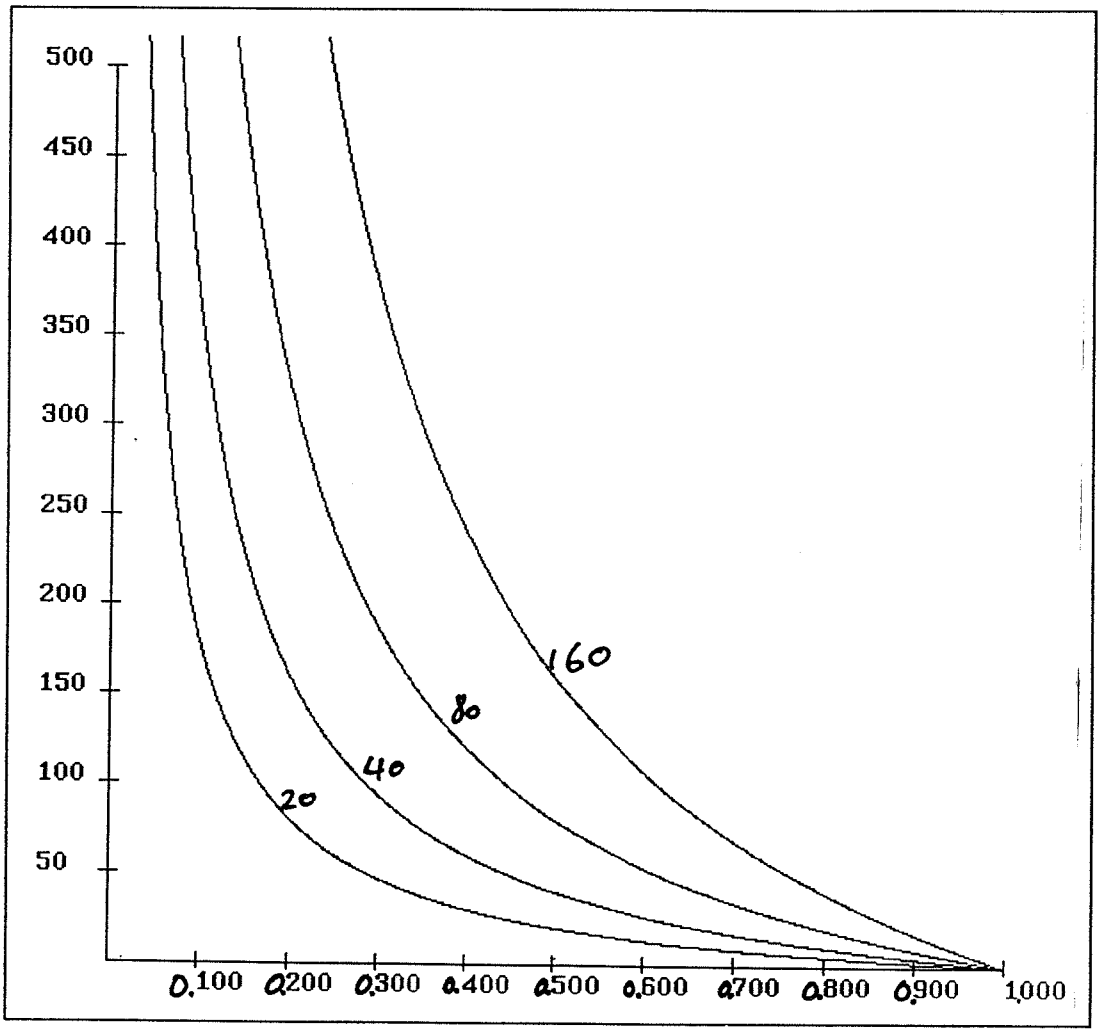

Figure 7: Time to synchronise for various numbers of sleep states against probability of individual sleeping.

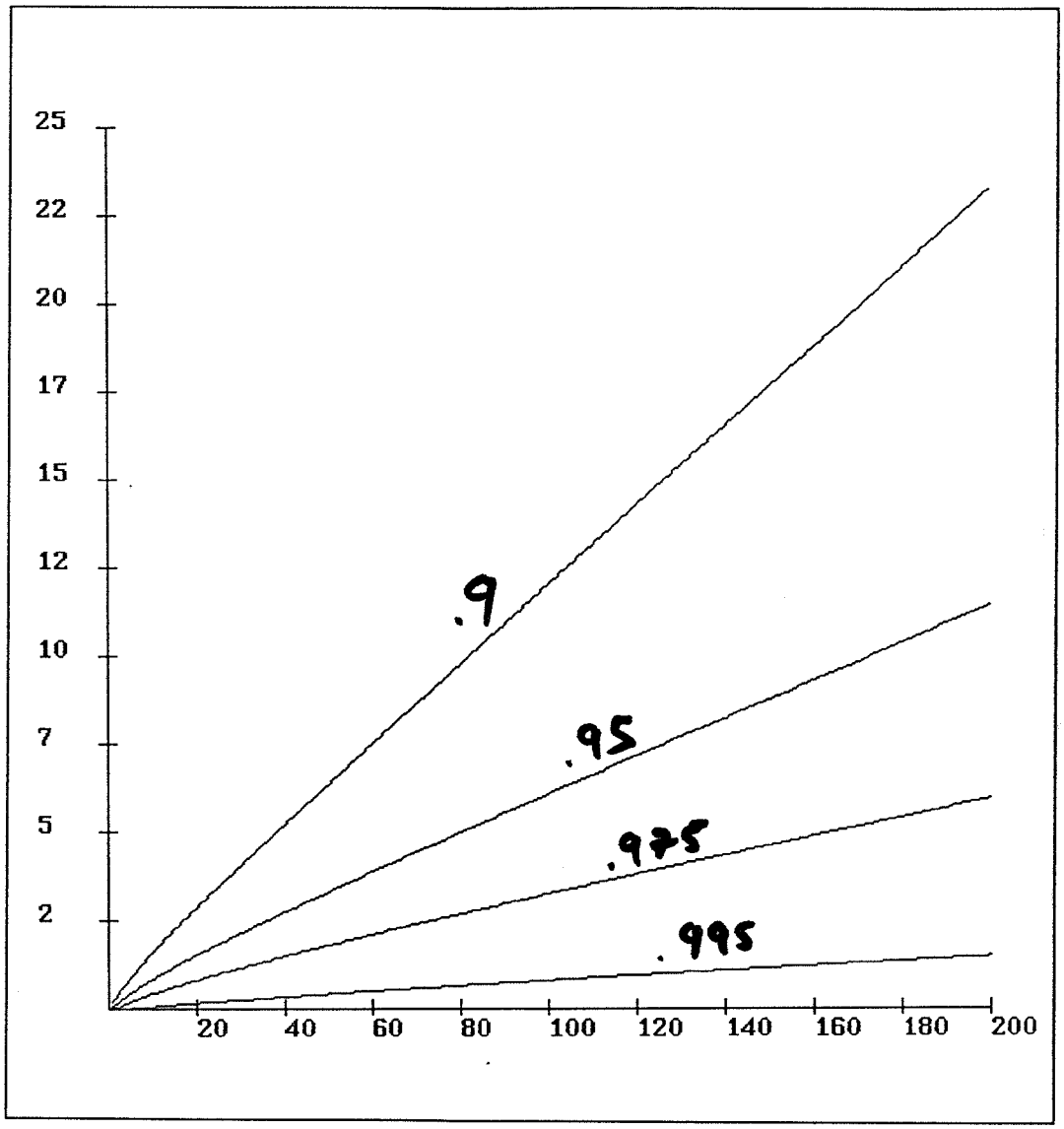

Figure 8: Time to synchronise for various sleeping probabilities against number of sleep states.

$$\begin{array}{c} (Sleep(0)^i)\lceil A \\ \downarrow m^i \\ \downarrow \mathbf{1} \\ (Wakeable^i)\lceil A \end{array} \qquad \begin{array}{c} (Slp(0))\lceil A \\ \downarrow m^i \\ \downarrow \mathbf{1} \\ (Wkble)\lceil A \end{array}$$

We see that these can only be equivalent if they are in a context, where if a process can recieve an $\overline{a}$ action then we can recieve an $\overline{a}^i$ action. To ensure that subsequently we evolve to an identifiable state. Similarly if we consider the waking up behaviour we find a difference in the priority structure that requires certain properties to be true of the context in which we place the processes $Ant^i\lceil A$ and $Ant'\lceil A$. Accordingly we will try to establish the equivalence of,

$$\Theta((Ant^i \times Sleep(n_1) \times \cdots Sleep(n_k))\lceil\{\mathbf{1}\})$$

and

$$\Theta((Ant' \times Sleep(n_1) \times \cdots Sleep(n_k))\lceil\{\mathbf{1}\}).$$

Since the behaviour in non-zero sleep states is trivial we examine the behaviour at the wake up point. That is the processes

$$\Theta((Sleep(0)^i \times Sleep(n_1) \times \cdots Sleep(n_k))\lceil\{\mathbf{1}\})$$

and

$$\Theta((Slp(0)' \times Sleep(n_1) \times \cdots Sleep(n_k))\lceil\{\mathbf{1}\}).$$

There are two cases;

- none of the n_i's are zero. In this case the first process is equivalent to the following;

$$\begin{array}{l} m^i.\mathbf{1}.\underbrace{1\ldots1}_{z-times}.\mathbf{1}.\Theta((Sleep(0)^i \times Sleep(n_1-1) \times \cdots Sleep(n_k-1))\lceil\{\mathbf{1}\})+ \\ \binom{\mathrm{i}}{1} m^{(i-1)}n.\mathbf{1}.\underbrace{1\ldots1}_{z-times}.\mathbf{1}.\Theta((Sleep(s)^i \times Sleep(n_1-1) \times \cdots Sleep(n_k-1))\lceil\{\mathbf{1}\}) \\ +\cdots+ \\ \binom{\mathrm{i}}{\mathrm{i}} n^i.\mathbf{1}.\underbrace{1\ldots1}_{z-times}.\mathbf{1}.\Theta((Sleep(s)^i \times Sleep(n_1-1) \times \cdots Sleep(n_k-1))\lceil\{\mathbf{1}\}), \end{array}$$

which we can rewrite as;

$$\begin{array}{l} m^i.\mathbf{1}.\underbrace{1\ldots1}_{z-times}.\mathbf{1}.\Theta((Sleep(0)^i \times Sleep(n_1-1) \times \cdots Sleep(n_k-1))\lceil\{\mathbf{1}\})+ \\ ((m+n)^i - m^i).\mathbf{1}.\underbrace{1\ldots1}_{z-times}.\mathbf{1}.\Theta((Sleep(s)^i \times Sleep(n_1-1) \times \cdots Sleep(n_k-1))\lceil\{\mathbf{1}\}), \end{array}$$

and the second is the following;

$$m^i.\mathbf{1}.\underbrace{1\ldots1}_{z-times}.\mathbf{1}.\Theta((Slp(0)' \times Sleep(n_1-1) \times \cdots Sleep(n_k-1))\lceil\{\mathbf{1}\})+$$
$$((m+n)^i - m^i).\mathbf{1}.\underbrace{1\ldots1}_{z-times}.\mathbf{1}.\Theta((Slp(s)' \times Sleep(n_1-1) \times \cdots Sleep(n_k-1))\lceil\{\mathbf{1}\}).$$

- One or more of the $n_i'.s$ are zero, to simplify the expansion we will consider the case where only n_1 is zero and all the other n_i are greater than or equal to one. In this case the first process is;

$$m^{(i+1)}.\mathbf{1}.\underbrace{1\ldots1}_{z-times}.\mathbf{1}.\Theta((Sleep(0)^i \times Sleep(0) \times \cdots Sleep(n_k-1))\lceil\{\mathbf{1}\})+$$
$$m^i n.\mathbf{1}.\underbrace{1\ldots1}_{z-times}.\mathbf{1}.\Theta((Sleep(s)^i \times Sleep(s) \times \cdots Sleep(n_k-1))\lceil\{\mathbf{1}\})+$$
$$\binom{i}{1} m^i n.\mathbf{1}.\underbrace{1\ldots1}_{z-times}.\mathbf{1}.\Theta((Sleep(s)^i \times Sleep(s) \times \cdots Sleep(n_k-1))\lceil\{\mathbf{1}\})+$$
$$\binom{i}{1} m^{(i-1)} n^2.\mathbf{1}.\underbrace{1\ldots1}_{z-times}.\mathbf{1}.\Theta((Sleep(s)^i \times Sleep(s) \times \cdots Sleep(n_k-1))\lceil\{\mathbf{1}\})+$$
$$+\cdots+$$
$$\binom{i}{i} m n^i.\mathbf{1}.\underbrace{1\ldots1}_{z-times}.\mathbf{1}.\Theta((Sleep(s)^i \times Sleep(s) \times \cdots Sleep(n_k-1))\lceil\{\mathbf{1}\})+$$
$$\binom{i}{i} n^{(i+1)}.\mathbf{1}.\underbrace{1\ldots1}_{z-times}.\mathbf{1}.\Theta((Sleep(s)^i \times Sleep(s) \times \cdots Sleep(n_k-1))\lceil\{\mathbf{1}\}),$$

which can be simplified to,

$$m^{(i+1)}.\mathbf{1}.\underbrace{1\ldots1}_{z-times}.\mathbf{1}.\Theta((Sleep(0)^i \times Sleep(0) \times \cdots Sleep(n_k-1))\lceil\{\mathbf{1}\})+$$
$$((m+n)^{(i+1)} - m^{(i+1)}).\mathbf{1}.\underbrace{1\ldots1}_{z-times}.\mathbf{1}.\Theta((Sleep(s)^i \times Sleep(s) \times \cdots Sleep(n_k-1))\lceil\{\mathbf{1}\}).$$

For the second we have,

$$m^{(i+1)}.\mathbf{1}.\underbrace{1\ldots1}_{z-times}.\mathbf{1}.\Theta((Slp(0)' \times Sleep(0) \times \cdots Sleep(n_k-1))\lceil\{\mathbf{1}\})+$$
$$m^i n.\mathbf{1}.\underbrace{1\ldots1}_{z-times}.\mathbf{1}.\Theta((Slp(s)' \times Sleep(s) \times \cdots Sleep(n_k-1))\lceil\{\mathbf{1}\})+$$
$$((m+n)^i - m^i)m.\mathbf{1}.\underbrace{1\ldots1}_{z-times}.\mathbf{1}.\Theta((Slp(s)' \times Sleep(s) \times \cdots Sleep(n_k-1))\lceil\{\mathbf{1}\})+$$
$$((m+n)^i - m^i)n.\mathbf{1}.\underbrace{1\ldots1}_{z-times}.\mathbf{1}.\Theta((Slp(s)' \times Sleep(s) \times \cdots Sleep(n_k-1))\lceil\{\mathbf{1}\}),$$

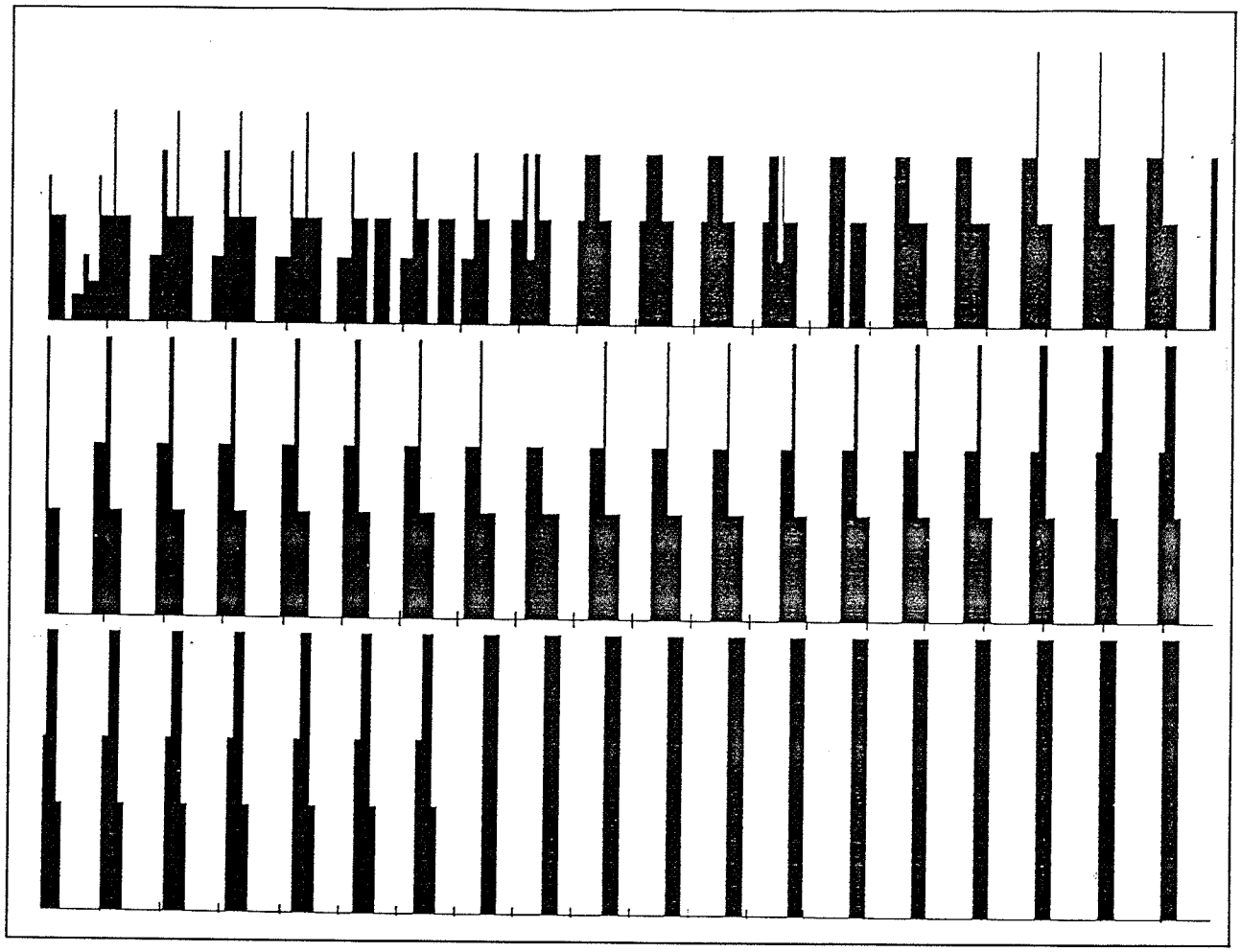

Figure 9: Six synchronised groups of sizes 8,5,3,3,1,1 synchronising.

which can in turn be simplified to;

$$m^{(i+1)}.\mathbf{1}.\underbrace{1\ldots1}_{z-times}.\mathbf{1}.\Theta((Slp(0)' \times Sleep(n_1-1) \times \cdots Sleep(n_k-1))\lceil\{\mathbf{1}\}) +$$
$$((m+n)^{(i+1)} - m^{(i+1)}).\mathbf{1}.\underbrace{1\ldots1}_{z-times}.\mathbf{1}.\Theta((Slp(s)' \times Sleep(s) \times \cdots Sleep(n_k-1))\lceil\{\mathbf{1}\}).$$

Whilst the above is not sufficient to demonstrate that the two versions are directly equivalent, it does suffice to demonstrate that they are abstractly equivalent. So they cannot be distinguished by just observing them. We used the above property to produce a simulation of a nest with more than 6 ants by using 6 groups of differing sizes and the result is presented in figure 9.

4.5 Creep Estimate of Synchronisation time.

When we have ants with differing sleep probabilities then they will have different expected cycle times. We can use this difference to *estimate* how long they will take to synchronise. Letting the expected cycle time be E_s and E_l, with $E_s < E_l$, then in each cycle we would expect the gap between the process to close by an amount $E_l - E_s$. Since we cannot start them more then E_s apart we thus get a maximum expected time to synchronisation of

$$\frac{E_s^2}{(E_l - E_s)}$$

This is only a first order approximation, and should be an overestimate in general. This is especially useful after the observation above when we have synchronised groups with different numbers of ants in them. For instance in estimating the time to synchronise for the situation in figure 10. We use the above formual to estimate the time taken to synchronise for groups of identical ants of sizes 2,4,6 with a group of size 20, for various values of the sleep probability. The result is given in figure 11.

4.6 Approximating Synchronisation Time.

A further method of approximating synchronisation time is to take a distribution of unsynchronised states. We should like a distribution that is stable over the transisition matrix; in the sense that we recover the same distribution but with some addition of the synchronised state. Unfortunately finding such a distribution requires us to find an eigenvector of a matrix of size $(s!/(s\text{-}n)!)^2$ (for s sleep states and n ants), which is too large for any practical example. So instead we have assumed that a linear superposition of unsynchronised states will be *reasonably* constant over the transition matrix. It is relatively easy to calculate the probability of this family of states synchronising and to calculate the expected number of cycles until synchronisation can be attempted again. Essentially we are using the final transition matrix given in the example of the 3 ants synchronising with 2 sleep states.

This method gives a very rough approximation, we have taken estimates for both the stable superposition and the expected time to next synchronisation attempt, and we cannot hope to have more than an order of magnitude approximation. This technique has been used to estimate the time to synchronise for 10 ants with 10 and 20 sleep states, for sleep probabilities ranging from 0.9 to 0.99 (for an individual). The results are presented in figure 12. For comparison the expected cycle times for both of these systems are approximately 10.4 and 20.4 respectively.

5 Coping With Errors.

We consider the performance of our system of ants when individuals can "mis-count" how long to stay asleep. We add an ant that will sleep for some number of ticks less than its

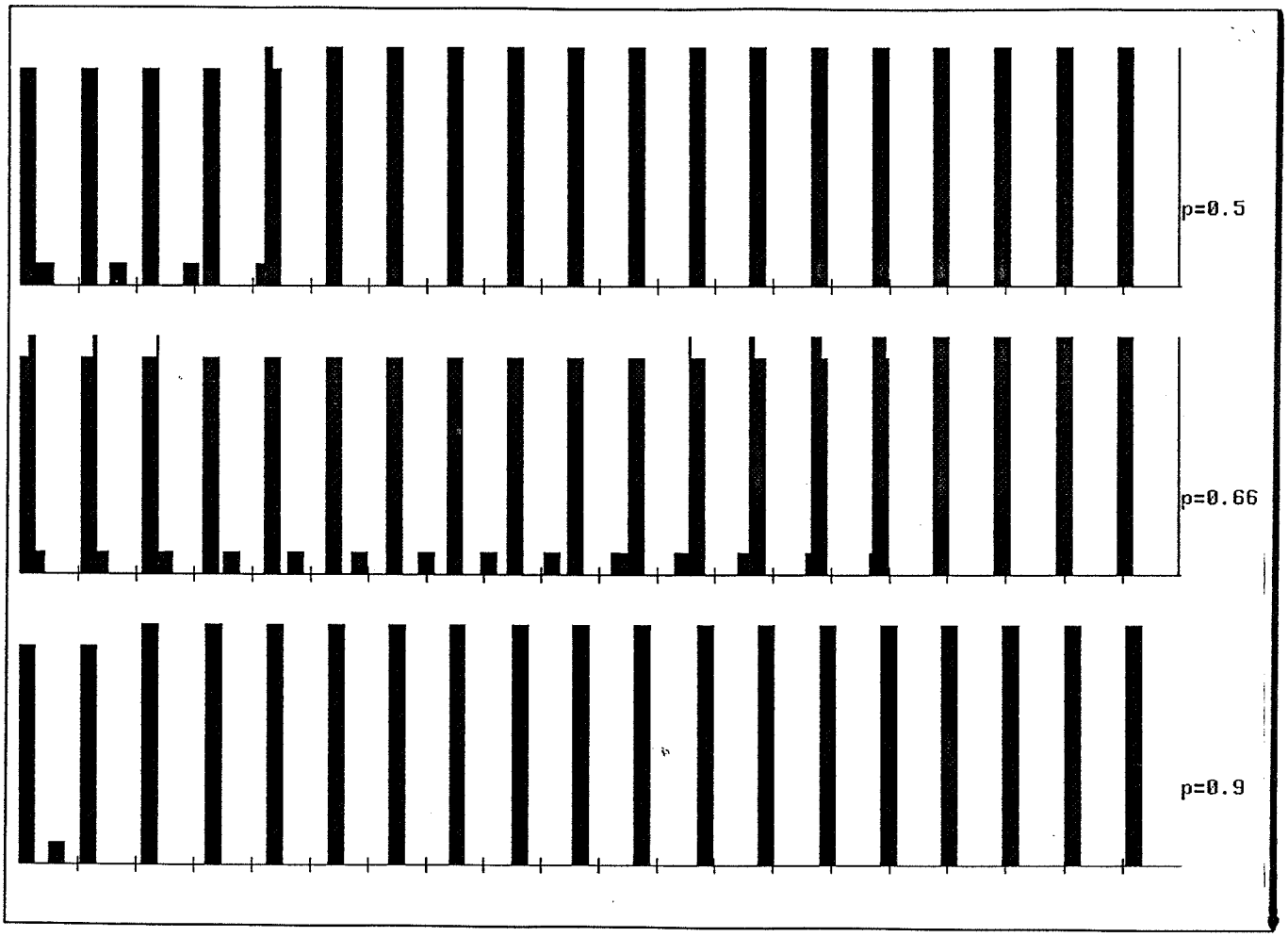

Figure 10: 1 ant synchronising with 10.

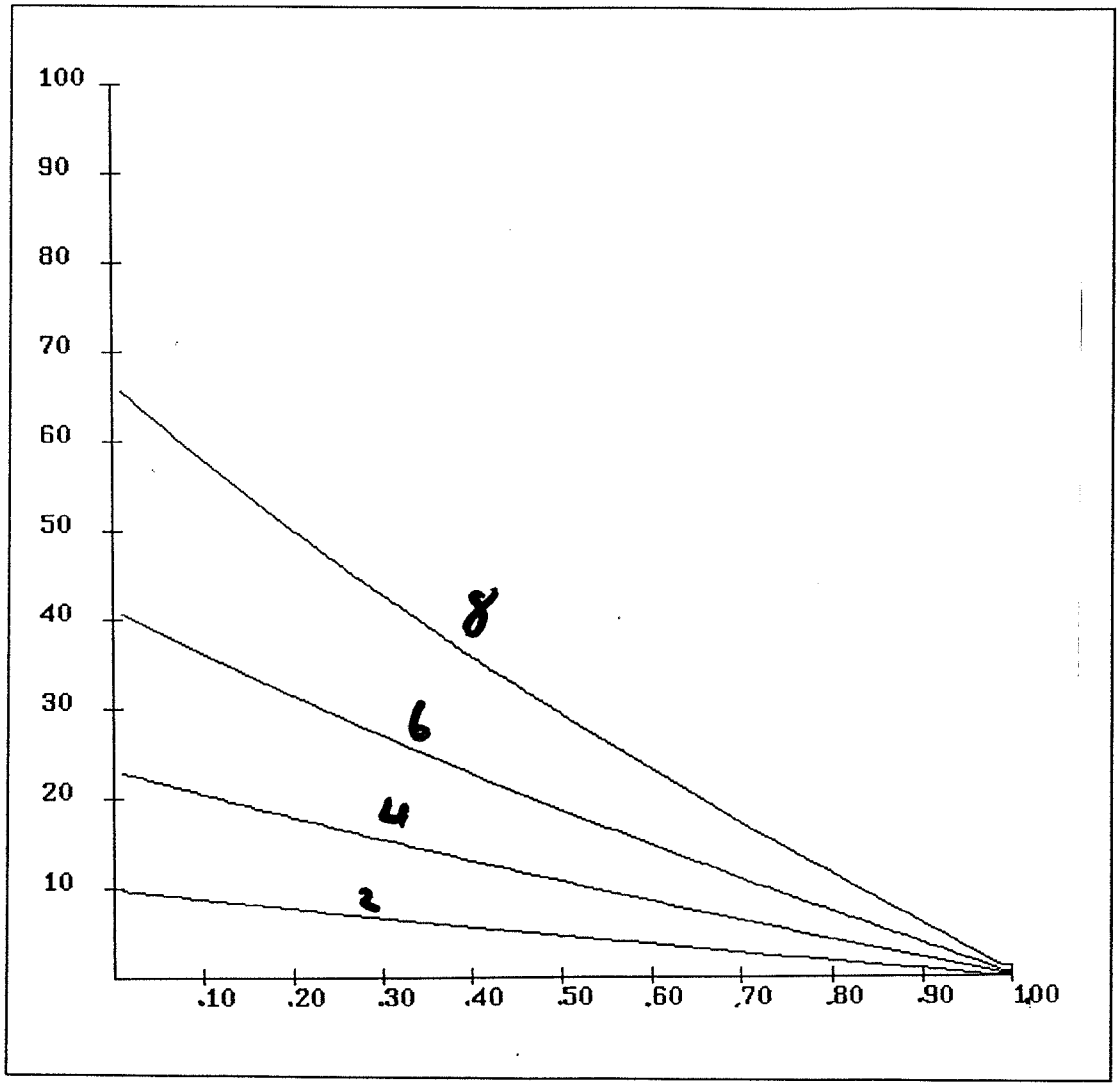

Figure 11: Aproximate synchronisation time against sleep probability for 20 ants with groups of various sizes.

P(Sleep)	10 Sleep states	20 Sleep States
0.90	36	12225
0.91	30	7016
0.92	25	3770
0.93	19	1896
0.94	15	895
0.95	11	397
0.96	8	167
0.97	5	68
0.98	4	25
0.99	2	9

Figure 12: Approximate time to synchronise for 10 ants.

fellows; that is to say an ant described by the same process as before but with a different value of the sleep parameter s.

$$
\begin{aligned}
&Esleep(k) = \mathbf{1}.\mathbf{1}.Esleep(k-1)\\
&Esleep(0) = Edecide\\
&Edecide = m.Ewakeable + n.Ebroad(z)\\
&Ebroad(k) = \omega.a^{k}.Esleep(s-e) + 1.Ebroad(k-1)\\
&Ebroad(0) = 1.\mathbf{1}.Esleep(s-e)\\
&Ewakeable = 1.a.Esleep(s-e) + 1.\mathbf{1}.Edecide
\end{aligned}
$$

So we can describe a nest with one of these error prone ants as follows (where we have fixed the parameter s).

$$\Theta((Ssleep(k) \times Ant_{i1} \times \ldots \times Ant_{ik})\lceil\{\mathbf{1}\})$$

The behaviour of such a nest differs as e is positive or negative. For e positive and less than s. If the nest starts off synchronised (that is all the ants are in their respective initial sleep states), we reach the following process after $2(s-e)$ ticks:

$$\Theta((Edecide \times Sleep(e) \times \ldots \times Sleep(e))\lceil\{\mathbf{1}\})$$

We have (essentially) two evolution paths for the above process, given the probability of an individual sleeping is p, the first is;

$$\Theta((Edecide \times Sleep(e) \times \ldots \times Sleep(e))\lceil\{\mathbf{1}\})$$
$$\downarrow p$$

$$
\begin{array}{c}
\Theta((Edecide \times Sleep(e-1) \times \ldots \times Sleep(e-1))\lceil\{\mathbf{1}\}) \\
\Big\downarrow \mathbf{1} \\
\Big\downarrow \mathbf{1} \\
\left.\begin{array}{l}\Big\downarrow p \\ \Big\downarrow \mathbf{1} \\ \Big\downarrow \mathbf{1}\end{array}\right\} (e-1) \text{ times} \\
\Theta((Edecide \times Decide \times \ldots \times Decide)\lceil\{\mathbf{1}\})
\end{array}
$$

and the second is;

$$
\begin{array}{c}
\Theta((Edecide \times Sleep(e) \times \ldots \times Sleep(e))\lceil\{\mathbf{1}\}) \\
\Big\downarrow 1-p \\
\Big\downarrow \mathbf{1} \\
\Big\downarrow \mathbf{1} \\
\Theta((Esleep(s-e) \times Sleep(e-1) \times \ldots \times Sleep(e-1))\lceil\{\mathbf{1}\})
\end{array}
$$

Which we can see will stay synchronised with the other ants with probability p^e. However in the case of e negative starting with a synchronised nest, we reach a slightly different state:

$$\Theta((Ssleep(-e) \times Sleep(0) \times \ldots \times Sleep(0))\lceil\{\mathbf{1}\})$$

This again can essentially evolve in two ways, the first being:

$$
\begin{array}{c}
\Theta((Ssleep(-e) \times Sleep(0) \times \ldots \times Sleep(0))\lceil\{\mathbf{1}\}) \\
\Big\downarrow p^{(z-1)} \\
\Big\downarrow \mathbf{1} \\
\Big\downarrow \mathbf{1} \\
\Theta((Ssleep(1-e) \times Sleep(0) \times \ldots \times Sleep(0))\lceil\{\mathbf{1}\}) \\
\left.\begin{array}{l}\Big\downarrow p^{(z-1)} \\ \Big\downarrow \mathbf{1} \\ \Big\downarrow \mathbf{1}\end{array}\right\} (1-e) \text{ times} \\
\Theta((Ssleep(0) \times Sleep(0) \times \ldots \times Sleep(0))\lceil\{\mathbf{1}\})
\end{array}
$$

or alternatively:

$$
\begin{array}{c}
\Theta((Ssleep(-e) \times Sleep(0) \times \ldots \times Sleep(0))\lceil\{\mathbf{1}\}) \\
\Big\downarrow 1-(p^{(z-1)}) \\
\Big\downarrow \mathbf{1} \\
\Big\downarrow \mathbf{1} \\
\Theta((Ssleep(1-e) \times Sleep(s) \times \ldots \times Sleep(s))\lceil\{\mathbf{1}\})
\end{array}
$$

So we see, with a total of z ants, the nest will only stay synchronised with probaility $p^{(z-1)e}$. These two situations can be summarised as follows.

- e positive, the local clock is fast, and we have a good chance of staying synchronised.
- e negative, the local clock is slow, and we have a poor chance of staying synchronised.

It should be noted that the ants that do not make a counting error stay synchronised (Section 4.4). We now present an slightly more complex ant which can cope with the problem of a slow clock more effectively. The new description of the ant is as follows.

- an ant will sleep for a period;
- it will then become wakeable but *will not* awake spontaneously;
- it then can either wake spontaneously or be woken, and it chooses to wake with some fixed probability;
- once it has awoken it will then fall asleep immediately.

This description amounts to the following WSCCS process for each individual.

$$
\begin{aligned}
&Dsleep(k) = \mathbf{1}.\mathbf{1}.Dsleep(k-1)\\
&Dsleep(0) = Ddoze(s')\\
&Ddoze(k) = \mathbf{1}.(1.a.Dsleep(s) + 1.\mathbf{1}.Ddoze(k-1)\\
&Ddoze(0) = Ddecide\\
&Ddecide = m.Dwakeable + n.Dbroad(z)\\
&Dbroad(k) = \omega.a^{k}.Dsleep(s) + 1.Dbroad(k-1)\\
&Dbroad(0) = \omega.\mathbf{1}.Dsleep(s)\\
&Dwakeable = 1.a.Dsleep(s) + 1.\mathbf{1}.Ddecide
\end{aligned}
$$

The colony is defined as usual by:

$$
\begin{aligned}
&Nant_{(i)} = Dsleep(i)\\
&Ncolony = \Theta((Nant_{(i_1)} \times \ldots \times Nant_{(i_z)})\lceil\{\mathbf{1}\})
\end{aligned}
$$

The process $Ncolony$ has the same synchronisation properties of the earlier colony. An interesting observation is that if we make the parameter s zero then as soon as one ant wakes up the whole colony will synchronise. Choosing s zero and letting the ant and colony definitions be as follows:

$$
\begin{aligned}
&Fant_{(i)} = Ddoze(i)\\
&Fcolony = \Theta((Fant_{(i_1)} \times \ldots \times Fant_{(i_z)})\lceil\{\mathbf{1}\})
\end{aligned}
$$

This process has the following evolution (we order the ants so that the lowest value of the i_k's is first):

$$\begin{array}{c}
\Theta((Fant_{(i_1)} \times \ldots \times Fant_{(i_z)})\lceil\{\mathbf{1}\}) \\
\downarrow \mathbf{1} \\
\downarrow \mathbf{1} \\
\Theta((Fant_{(i_1-1)} \times \ldots \times Fant_{(i_z-1)})\lceil\{\mathbf{1}\}) \\
\left.\begin{array}{l}\downarrow \mathbf{1} \\ \downarrow \mathbf{1}\end{array}\right\} (i_1 - 1) \text{ times} \\
\Theta((Ddecide \times \ldots \times Fant_{(i_z-i_1)})\lceil\{\mathbf{1}\}) \\
\downarrow 1-p \\
\downarrow \mathbf{1} \\
\downarrow \mathbf{1} \\
\Theta((Fant_{s'} \times \ldots \times Fant_{s'})\lceil\{\mathbf{1}\})
\end{array}$$

When the first ant wakes up then as all the others are dozing they will wake and thus on the first waking the colony will become synchronised.

The only reason we should not do this is that in some sense listening for a wake up signal all the time is not resting. So we can build a colony identical to our earlier one but that will synchronise on the first ant waking by taking the following description of an individual.

$$\begin{array}{l}
Hsleep(k, s') = \mathbf{1}.\mathbf{1}.Hsleep(k-1, s') \\
Hsleep(0, s') = Hdoze(s', s') \\
Hdoze(k, s') = \mathbf{1}.(1.a.Hsleep(s, 0) + 1.\mathbf{1}.Hdoze(k-1, s') \\
Hdoze(0, s') = Hdecide \\
Hdecide = m.Hwakeable + n.Hbroad(z) \\
Hbroad(k) = \omega.a^k.Hsleep(s, s') + 1.Hbroad(k-1) \\
Hbroad(0) = \mathbf{1}.Hsleep(s, s') \\
Hwakeable = 1.a.Hsleep(s, s') + 1.\mathbf{1}.Hdecide
\end{array}$$

and taking the initial definitions of the individual and nest as

$$\begin{array}{l}
Hant_j = Hdoze(j, 0) \\
Hcolony1 = \Theta((Ant_{(i_1)} \times \ldots \times Ant_{(i_z)})\lceil\{\mathbf{1}\})
\end{array}$$

Unfortunately this has the same error behaviour as before, but if we retain some dozing period, then we can afford the local clock to run slow by that period and the system will stay synchronised. Whilst retaining the high probability that if the clock runs fast we will stay synchronised.

So we define an ant that makes counting errors much as before:

$$Ehsleep(k, s') = \mathbf{1}.\mathbf{1}.Ehsleep(k-1, s')$$
$$Ehsleep(0, s') = Ehdoze(s', s')$$
$$Ehdoze(k, s') = \mathbf{1}.(1.a.Ehsleep(s, 0) + 1.\mathbf{1}.Ehdoze(k-1, s')$$
$$Ehdoze(0, s') = Ehdecide$$
$$Ehdecide = m.Ehwakeable + n.Ehbroad(z)$$
$$Ehbroad(k) = \omega.a^k.Ehsleep(s-e, s') + 1.Ehbroad(k-1)$$
$$Ehbroad(0) = \mathbf{1}.Ehsleep(s-e, s')$$
$$Ehwakeable = 1.a.Ehsleep(s-e, s') + 1.\mathbf{1}.Ehdecide$$

The definition of a nest is as follows:

$$Hant_j = Hsleep(j, s')$$
$$Hcolony = \Theta((Ehsleep(i_1) \times Hant_{(i_2)} \times \ldots \times Hant_{(i_z)})\lceil\{\mathbf{1}\})$$

Once again we consider starting with all the ants in their maximal sleep state. Then with e positive but less than s' we arrive in the state below which has the described derivation:

$$\Theta((Ehdecide \times Hdoze(s'-e, s') \times \ldots \times Hdoze(s'-e, s')\lceil\{\mathbf{1}\})$$
$$\downarrow 1-p$$
$$\downarrow \mathbf{1}$$
$$\downarrow \mathbf{1}$$
$$\Theta((Ehsleep(s-e, s') \times Hsleep(s, s') \times \ldots \times Hsleep(s, s'))\lceil\{\mathbf{1}\})$$

or alternatively

$$\Theta((Ehdecide \times Hdoze(s'-e, s') \times \ldots \times Hdoze(s'-e, s'))\lceil\{\mathbf{1}\})$$
$$\downarrow p$$
$$\downarrow \mathbf{1}$$
$$\downarrow \mathbf{1}$$
$$\Theta((Ehdecide \times Hdoze(s'-e-1, s') \times \ldots \times Hdoze(s'-e-1, s'))\lceil\{\mathbf{1}\})$$

but in either case the nest will stay synchronised. With e negative but less than s' then we arrive at the state below which has the desribed subsequent derivations.

$$\Theta((Ehdoze(s'-e, s') \times Hdecide \times \ldots \times Hdecide(s'-e, s'))\lceil\{\mathbf{1}\})$$
$$\downarrow 1-p^{(z-1)}$$
$$\downarrow \mathbf{1}$$
$$\downarrow \mathbf{1}$$
$$\Theta((Ehsleep(s-e, s') \times Hsleep(s, s') \times \ldots \times Hsleep(s, s')\lceil\{\mathbf{1}\})$$

or alternatively

$$\Theta((Ehdoze(s'-e,s' \times Hdecide \times \ldots \times Hdecide\lceil\{\mathbf{1}\})$$
$$\downarrow p$$
$$\downarrow \mathbf{1}$$
$$\downarrow \mathbf{1}$$
$$\Theta((Ehdoze(s'-e-1,s') \times Hdecide \times \ldots \times Hdecide)\lceil\{\mathbf{1}\})$$

Thus as long as the error is less than the dozing period the ants will stay synchronised. The ratio between the amount the ant sleeps and the amount it dozes can be considered as a measure of how disturbed a colony is. The lower this ratio the faster the nest can respond as a whole to the behaviour of others. This ratio could be controlled by some external action but for simplicity is omitted.

6 Experimental Results.

To test the model Melanie Hatcher[6] has used activity information gathered using a digitized camera image. Activity levels for two nests have been monitored for periods of 112 hours, and the cycles lengths estimated from that data. Cycles with mean period 1309 and 1307 seconds were found, these estimates being accurate to 34 and 32 seconds with a confidence of 95%. The cycle lengths are then tested against the prediction that they should arrise from an exponential decay with mean matched to that given by the model. Ms Hatcher has found that, when a Chi-squared test for fit is used a 'best' values of 11.38 and 8.64 were obtained. Since this system has 10 degrees of freedom this is well within the 99% confidence limits that the data matches the predictions, at least for cycle length.

7 Conclusions and Further Work.

We have presented a model of autosynchronisation that has the essential property we should require of always achieving synchronisation. The synchronised state is both simple and elegant. The predictions about this state have been applied to relatively undisturbed ant nests and have matched up to the limit of the technology being used to measure the nests. Moreover we have both exact (for 2 ants) and approximate measures of how long such a system will take to synchronise.

It is of interest that there is little 'trade off' in accuracy of cycle time versus efficeincy of synchronisation. The exponential nature of many ants synchronising together allows us to have a very low probability of an individual waking (and thus a very fast synchronisation time) whilst having a very high probability of a collection of individuals waking. We have shown that a derived algorithmn can be highly stable to errors in counting of individuals, and thus provides a highly error tolerant synchronisation technique.

In biological terms our model is both highly abstract and contains some unphysical assumptions. We have basically assumed that ants can produce a signal which travels an

[6]The Department of Animal Physiology and Ecology, School of Biological Sciences, University of Bath

infinite distance in a finite time. However this does not seem to lead to any detectably erroneous deductions. Also the assumption that all ants will stay asleep for the same amount of time is not true; workers of differing castes appear to sleep for differing times. Some simulations of this situation were run, having changed the processes appropriately, they showed some of the 'messiness' of the original activity data (see figure 13). One possible solution to this is to introduce a monitor process with a definition similar to below,

$$Mon = \omega.(1.aw^{21}.Amon + \ldots + 1.aw^{30}.Amon) + 1.\mathbf{1}.Mon$$
$$Amon = \overline{awake}.\underbrace{\mathbf{1}\ldots\mathbf{1}}_{n-times}.Mon$$

and then consider the process,

$$\Theta((Mon \times Colony)\lceil\{\mathbf{1}, \overline{awake}\})$$

taking the cycle length to be the duration between the $\overline{awake}$ actions. A similar technique can be used with colonies whose individual ants make mistakes: i.e. do not respond to wake up signals; or to colonies with only finite transmission ranges of wake up signals.

8 Acknowledgments.

Chris Tofts is supported by a BP Venture Research Grant. We would like to thank Melanie Hatcher, Nigel Franks, Jean Louis Deneubourg and Simon Goss for discussions on the biological content of this work, especially Melanie for doing all the experiments. On the computational side we would like to thank Mathew Morley and Fallen Moron[7] for their constructive comments, Glenn Bruns and the other members of the concurrency club for their useful discussion.

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

[DGPFL87] J.L. Deneubourg, S. Goss, J.M. Pasteels, D. Fresneau, J.P. Lachaud, Self-organisation mechanisms in Ant Societies (II), In: From Indivivual to Collective Behaviour in Social Insects, J.M. Pasteels and J.L. Deneubourg eds, pp 177-196, Experienta Supplementum 54, Birkhauser Verlag, 1987

[FB87] N. Franks, S. Bryant, Rythmical Patterns of Activity within the Nests of Ants, in Chemistry and Biology of Social Insects, J. Eder, H. Rembold eds, pp 122-123, Verlag J. Peperny, 1987.

[7] Copyright Master David Pym.

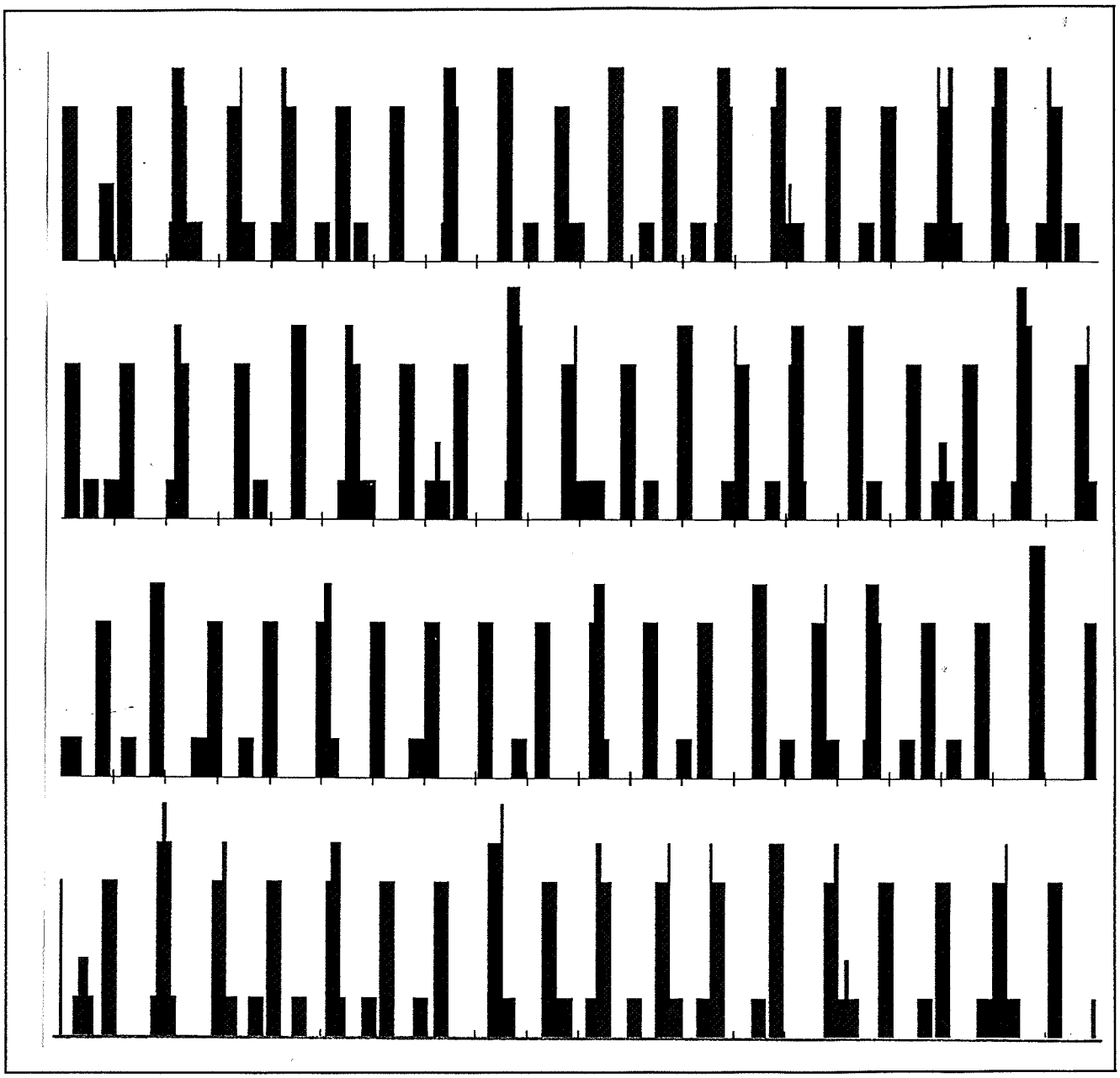

Figure 13: Ants with Differing Sleep Times.

[FBGH89] N. Franks, S. Bryant, R. Griffiths, L. Hemerik, Synchronisation of the Behaviour Within Nests of the Ant *Leptothorax Acervorum (Fabricius)* I, Discovering the Phenomenon and its Relation to Levels of Starvation, Bulletin of Mathematical Biology, Vol. 51, 1989.

[GD88] S. Goss, J.L. Deneubourg, Autocatalysis as a Source of Synchronised Rythmical Activity in Social Insects, Insectes Sociuax, Paris, Vol. 35, number 3, pp 310-315, 1988.

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

[HBF89] L. Hemerik, N.F. Britton, N. Franks, Synchronisation of the Behaviour Within Nests of the Ant *Leptothorax Acervorum (Fabricius)* II, Modelling the Phenomenon and Predictions from the Model. Bulletin of Mathematical Biology, Vol. 51, 1989.

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

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

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

[May73] R.M. May, Stability and Complexity in Model Ecosystems, Princeton University Press 1973.

[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.

[NP77] G. Nicolis, I. Prigogine, Self-Organisation in Non-equilibrium Systems, Wiley, 1977

[PDG87] J.M.Pasteels,J.L. Deneubourg, S. Goss, Self-organisation mechanisms in Ant Societies (I), In: From Indivivual to Collective Behaviour in Social Insects, J.M. Pasteels and J.L. Deneubourg eds, pp 155-175, Experienta Supplementum 54, Birkhauser Verlag, 1987

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

[See87] T.D. Seeley, Foraging by Honeybee Colonies: a Case Study of Decenteralised Control in Insect Societies, in Chemistry and Biology of Social Insects, J. Eder, H. Rembold eds, pp 489-491, Verlag J. Peperny, 1987.

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

[THF90] C. Tofts, M.J.Hatcher, N. Franks, Autosynchronisation in Leptothorax Acervorum; Theory, Testability and Experiment, In preperation.

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

[WH88] E.O. Wilson, B. Holldobler, Dense Heterachies and Mass Communication as the Basis and Organisation in Ant Colonies, TREE 3, pp 65-68, 1988

A Swb.

This is a simple 'algebraists assistant' written in ML, which was used to produce the traces for the simulations in this document.

A.1 Syntax of Swb.

The syntax of process is as follows (operators in binding order):

- process variables start with a capital letter;
- the empty process is written 'Nil';
- the action tick is written 't';
- priority operator is written !( )
- renaming is written P#(a1/a,a2/b) will rename a by a1 b by a2;
- permission is written P|(a1,a2,a3) brackets are important.
- prefixing is written a.P;
- product is written P * Q;
- finally summation is written P:Q:R and will check that all the process have an initial weight guard.

The syntax of actions is as follows:

- a lower case name (possibly including numbers but not starting with them);
- raising an action to a power is written '^' e.g. a^5;
- product is written '&' as a & b denotes the product of a and b.

The syntax of weights is as follows:

- either a positive integer;

- or an '@' denoting the prioritised selection.

To summarise in BNF;

$$
\begin{aligned}
&\mathit{action} = \mathbf{name}|\mathit{action}\hat{}\mathbf{number}|\mathit{action}\&\mathit{action}\\
&\mathit{weight} = \mathbf{number}|@\\
&\mathit{Sumterm} = \mathit{weight}.\mathit{Process}\\
&\mathit{perm} = (\mathit{action},\ldots,\mathit{action})\\
&\mathit{rename} = (\mathit{action}/\mathit{action},\ldots,\mathit{action}/\mathit{action})\\
&\mathit{Process} = \mathit{action}.\mathit{Process}|\mathit{Process}*\mathit{Process}| \qquad \mathit{Process}|\mathit{perm}|\\
&\qquad\qquad \mathit{Sumterm}:\mathit{Sumterm}\ldots:\mathit{Sumterm} \quad |\mathit{Process}\#\mathit{rename}|\\
&\qquad\qquad !\mathit{Process}|(\mathit{Process})|\mathbf{Processname}
\end{aligned}
$$

A file is a set of bindings of the form

Process name = Process ;

the semi-colon is the process expression terminator.

A.2 Commands.

The following commands are avilable:

- cp(), clears the current process environment;
- rp "filename", reads the file into the process environment using the bindings contained in the file;
- gp "processname", puts that process into the current process buffer;
- dn n, chooses the n'th state from the normalised current process;
- trace n, runs the current process through n steps printing the actions;
- opf "filename" opens a file to output all the text written by the system, it also still appears on the screen;
- cf(), closes the current text file.

A.3 Example.

The syntax of the processes for the Swb used to produce the simulation of the groups of ants.We have abbreviated by removing some of the intermediate sleep states in the later ant group defintions. Note that test actions have been added to the process to aid the animation.

```
Sleepz20=aw^8.aw^8.Sleepz19;
Sleepz19=aw^8.aw^8.Sleepz18;
Sleepz18=aw^8.aw^8.Sleepz17;
Sleepz17=aw^8.aw^8.Sleepz16;
Sleepz16=aw^8.aw^8.Sleepz15;
Sleepz15=aw^8.aw^8.Sleepz14;
Sleepz14=t.t.Sleepz13;
Sleepz13=t.t.Sleepz12;
Sleepz12=t.t.Sleepz11;
Sleepz11=t.t.Sleepz10;
Sleepz10=t.t.Sleepz9;
Sleepz9=t.t.Sleepz8;
Sleepz8=t.t.Sleepz7;
Sleepz7=t.t.Sleepz6;
Sleepz6=t.t.Sleepz5;
Sleepz5=t.t.Sleepz4;
Sleepz4=t.t.Sleepz3;
Sleepz3=t.t.Sleepz2;
Sleepz2=t.t.Sleepz1;
Sleepz1=t.t.Decidez;
Decidez=43.t.Wakeablez:57.t.Broadcastz;
Wakeablez=1.1.1.1.1.1.(1.t.Decidez:1.a.Sleepz20);
Broadcastz=Broadz5;

Broadz5=@.'a^5.Sleepz20:1.Broadz4;
Broadz4=@.'a^4.Sleepz20:1.Broadz3;
Broadz3=@.'a^3.Sleepz20:1.Broadz2;
Broadz2=@.'a^2.Sleepz20:1.Broadz1;
Broadz1=@.'a.Sleepz20:1.t.Sleepz20;

Sleepb20=aw^5.aw^5.Sleepb19;
.
.
Sleepb1=t.t.Decideb;
Decideb=59.t.Wakeableb:41.t.Broadcastb;
Wakeableb=1.1.1.1.1.1.(1.t.Decideb:1.a.Sleepb20);
Broadcastb=Broadb5;

Broadb5=@.'a^5.Sleepb20:1.Broadb4;
.
.
Broadb1=@.'a.Sleepb20:1.t.Sleepb20;
```

```
Sleepa20=aw^3.aw^3.Sleepa19;
.
.
Sleepa1=t.t.Decidea;
Decidea=73.t.Wakeablea:27.t.Broadcasta;
Wakeablea=1.1.1.1.1.1.(1.t.Decidea:1.a.Sleepa20);
Broadcasta=Broada5;

Broada5=@.'a^5.Sleepa20:1.Broada4;
.
.
Broada1=@.'a.Sleepa20:1.t.Sleepa20;


Sleep20=aw.aw.Sleep19;
.
.
Sleep1=t.t.Decide;
Decide=9.t.Wakeable:1.t.Broadcast;
Wakeable=1.1.1.1.1.1.(1.t.Decide:1.a.Sleep20);
Broadcast=Broad5;

Ant=Sleepz20;
Ant1=Sleepa10;
Ant2=Sleepa15;
Ant3=Sleepb12;
Ant4=Sleep4;
Ant5=Sleep7;

Broad5=@.'a^5.Sleep20:1.Broad4;
.
.
Broad1=@.'a.Sleep20:1.t.Sleep20;

Cola=!((Ant * Ant1 * Ant2 *Ant3 * Ant4 * Ant5)|
      (t,aw,aw^2,aw^3,aw^4,aw^5,aw^6,aw^7,aw^8,aw^9,aw^10,aw^11,aw^12,
       aw^13,aw^14,aw^15,aw^16,aw^17,aw^18,aw^19,aw^20,aw^21,aw^22));
```

B Edges.

The event driven graphical system is a simple method of demonstrating the evolution of processes graphically. For each state change that the animator wishes to represent a test action can be added to the process. The system takes a trace of observable actions and an event driven graphics program, and animates the trace using the program.

B.1 Events.

An event consists of two parts;

- name, consisting of the usual alphabet. Case is significant.
- value, (optional) this is an ascii number.

B.2 Traces.

Permitted trace files are sequences of events, seperated by commas or carriage returns. This can be generated using the command *random* in the version of the workbench in ~fm/lib/cw.exp, this function generates a random trace of given length of observable actions.

B.3 The Language.

This is structured in blocks, and has the following grammer.

Var ::= *String* | *String*(*Exp*)
Exp ::= *Var* | *Exp*+*Exp* | *Exp*−*Exp* | *Exp*∗*Exp*.
Com ::= *Var*:=*Exp* |
loop *Exp* *Combl* | **line** *Exp*,*Exp*,*Exp*,*Exp*,*Exp* |
rect *Exp*,*Exp*,*Exp*,*Exp*,*Exp* |
frect *Exp*,*Exp*,*Exp*,*Exp*,*Exp* |
text *String*,*Exp*,*Exp*,*Exp* | **num** *Exp*,*Exp*,*Exp*,*Exp*
Combl ::= *Com*;*Combl* | **end**
Event ::= *String* | *String*(*String*) | *String*(*Number*)
Block ::= **on** *Event* **do** *Combl*

Note: that a string consists only of upper and lower case letters, and underscore, space cannot be used nor can numbers. Parsing of expressions is from right to left with no operator precedence or brackets, if this proves unhelpful I'm sorry, and will probably fix in later implementations. A carriage return can be used instead of a semi-colon as a command seperator.

This syntax is interpreted as follows; if a mathching event is found in the trace then the block of commands associated with it is executed. If the event is specified as *start*(z) then the value associated with the event *start* will be assigned to the variable z. An event of the form *start*(5) will only match an action *start* with the value 5.

B.3.1 Parameters.

For the graphics functions the parameters are as follows

$$x1, y1, x2, y2, c$$

for line this means draw from $(x1, y1)$ to $(x2, y2)$ in colour c, (currently 0=white, 1=black). For the rectangles they specify top left hand and bottom left right corner. (Note in sunview top left hand corner of the screen is $(0, 0)$ so y coords decrease going up the screen, i.e an allowable rectangle is $(0, 0, 200, 100)$.

In the text function the three expressions are x-location, y-location and font respectively. In the num function the four expressions are respectively the expression to print up, followed by location and font.

The expression in the loop command controls the number of executions of the following commands.

B.3.2 Errors.

Error reporting is fairly detailed, an error name (which should be explanatory) is printed followed by the next twenty characters in the command file. One standard error is caused by the inclusion of numbers in strings, especially in the construction of actions. The text *on begin5* is *not* a valid block header but *on begin*(5) is.

B.4 Example.

The following program was used to illustrate the ant simulation. The *start* and *new* actions were added to the trace from the swb by hand.

```
on start do
   x:=10
   y:=200
   line 10,200,810,200,1
   line 10,400,810,400,1
   line 10,600,810,600,1
   line 10,800,810,800,1
   t:=50
   loop 19
     line t,205,t,195,1
     line t,405,t,395,1
     line t,605,t,595,1
     line t,805,t,795,1
     t:=t+40
   end
end
```

```
on new do
   x:=10
   y:=y+200
end

on t do
   x:=x+1
end

on aw do
   line x,y,x,y-15,1
   x:=x+1
end

on aw^(n) do
   z:=15*n
   line x,y,x,y-z,1
   x:=x+1
end
```

B.5 Usage.

There are only four commands provided:

- **simulate("tracefile","programfile");** will simulate the trace using the program specified.
- **gprog "commandfile";** will load the command file.
- **gtrace "tracefile";** loads a trace file.
- **gdo();** executes the loaded program on the loaded trace.

All of these commands are executed in a graphics window of size 1000*800.