

## The Choice Coordination Problem

Michael O. Rabin

Department of Mathematics, The Hebrew University of Jerusalem  
and Division of Applied Mathematics, Aiken Computation Laboratory,  
Harvard University, Cambridge, MA 02138, USA\*

**Summary.** In the course of a concurrent computation, processes  $P_1, \dots, P_n$  must reach a common choice of one out of  $k$  alternatives  $A_1, \dots, A_k$ . They do this by protocols using  $k$  shared variables, one for each alternative. If the range of the variables has  $m$  values then  $\frac{1}{2} \sqrt[3]{n} \leq m$  is necessary, and  $n + 2 \leq m$  is sufficient, for deterministic protocols solving the choice coordination problem (C.C.P.). We introduce very simple randomizing protocols which, independently of  $n$ , solve the C.C.P. by use of a fixed alphabet. A single-byte (256-valued) alphabet permits a solution with non-termination probability smaller than  $2^{-127}$ . Many software and hardware tasks involving concurrency can be interpreted as choice coordination problems. Choice coordination problems occur also in nature.

### 1. Introduction

Let  $P_1, \dots, P_n$  be  $n$  processes engaged in some concurrent computation. Assume that in the course of that computation some or all of the processes come upon  $k$  possible alternatives  $A_1, \dots, A_k$ , and that for the global computation to proceed the processes must choose one and only one of these alternatives. It does not matter which  $A_i$  will be agreed upon, but a coordinated choice must be arrived at.

If the alternatives  $A_1, \dots, A_k$  are identically named by all processes, say by words  $w_1, \dots, w_k$ , then there is an easy solution. Each  $P_s$  participating in the choice will traverse all the  $A_1, \dots, A_k$  in some order and choose the  $A_i$  such that  $w_i = \min w_j$ . If, however, each  $P_s$  has his own system of names for  $A_1, \dots, A_k$  then a different approach to choice coordination is required.

Assume that each  $A_i$  has an associated variable  $v_i$  which is shared by  $P_1, \dots, P_n$ . A process  $P_s$  arriving at  $A_i$  can *test and set*  $v_i$ , i.e. in one in-

---

\* This research was supported in part by NSF grants: MCS77-02474 at Washington University, Seattle, MCS80-12716 at University of California at Berkeley. Presented at the Specker Symposium in Zürich, January 1980

divisible step and without interruption from any other process, read the current value of  $v_i$  and possibly change it. Choosing  $A_i$  will be signaled by assigning  $v_i := e$  where  $e$  is a distinguished value in the range of  $v_i$ .

A solution for the choice coordination problem (C.C.P.) is a system of protocols for  $P_1, \dots, P_n$  such that for every order in which the processes are activated, eventually one and only one of  $v_1, \dots, v_k$  will satisfy  $v_i = e$ .

We assume that the ranges of  $v_1, \dots, v_k$  are  $\Sigma = \{0, 1, \dots, m-2, e\}$  and that all the shared variables are initialized  $v_i := 0$ . Our main concern will be the size  $|\Sigma| = m$  of the coordination alphabet, as a function of the number  $n$  of processes.

The coordination problem first arose in a study by M. Fischer and the present author [2], of an algorithm for concurrent search of a data structure by many processors. The structure is presented as a collection of nodes or cells, where each node contains a number of pointers to other nodes. A subprogram of that algorithm involved a part  $L$  of the structure which is a simple closed loop consisting of cells  $C_1, \dots, C_k$  where from  $C_i$  there is a unique pointer to  $C_{i+1}$ ,  $1 \leq i \leq k$ , ( $k+1 = 1 \bmod k$ ). Some of the processes  $P_1, \dots, P_n$  enter  $L$  at various cells and start traversing it in the order imposed by the pointers. In order that the global search algorithm may proceed, the processes must arrive at a common choice of a cell  $C_i$  at which to "break"  $L$ .

M. Fischer and the present author have devised, for the case that there is no common system of names for the cells, a coordination protocol which uses  $n+2$ -valued shared variables.

It is not easy to reduce the size of the coordination alphabet below  $n$  and still find a solution for the C.C.P. On the other hand it is not even obvious that a fixed alphabet  $\Sigma_k$  (depending on  $k$ ) will not suffice for the C.C.P. for any collection  $P_1, \dots, P_n$ , where the protocols may of course depend on  $n$ .

M. Ben-Or [1] found, for  $k=2$ , coordination protocols for  $n$  processes using about  $n/2+2$  letters. This lays to rest the obvious conjecture that if there are more processes than letters ( $|\Sigma| < n$ ) then coordination is impossible.

The main result of this paper (Theorem 4) is that for  $k=2$ , if we use  $m$  letters to coordinate  $n$  processes and  $8m^3 \leq n$  then for every  $P_1, \dots, P_n$  there exists a schedule  $S$  (sequence of activation of the processes) so that the processes do not achieve C.C. when computing under  $S$ . In short, if  $8|\Sigma|^3 \leq n$  there does not exist a solution for the C.C.P. for  $n$  processes.

Thus if  $n+2 \leq m$  there exists a solution, and for  $m \leq \frac{1}{2} \sqrt[3]{n}$  there does not exist a solution for the C.C.P. for  $n$  processes. In terms of numbers of bits for representing the coordination alphabet,  $O(\log n)$  bits are necessary and sufficient for a solution of the C.C.P.

Next we introduce the idea of using randomization in the coordination protocol and get a surprising result. In a randomized protocol or program each  $P_i$ , in its turns according to the schedule  $S$ , performs an atomic action which may depend on the value of a randomly chosen number  $r$ ,  $1 \leq r \leq R$ . Given  $0 < \varepsilon < 1$ , we consider a system  $P_1, \dots, P_n$  of such randomizing protocols to be an  $1-\varepsilon$  solution for the C.C.P. if for every schedule  $S$ , the probability for the processes to reach C.C. is at least  $1-\varepsilon$ . We do not assume a probability distribution on the schedules  $S$ , but rather achieve a highly reliable solution

which is effective for every schedule. For this point of view concerning the introduction of randomization into algorithms see [5]. It turns out that for  $k = 2$  and a fixed alphabet with  $m$  letters we can formulate, for any  $n$ , protocols  $P_1, \dots, P_n$  which will achieve C.C. with probability at least  $1 - 1/2^{m/2}$ . Thus with  $m = 256$ , i.e. using 8 bits, we get a solution with reliability greater than  $1 - 1/2^{128}$ . This method easily generalizes to arbitrary  $k$ .

William Bossert has pointed out to me an interesting example of choice coordination in nature. The mite of the genus *Myrmoysus* parasites the ear membrane of moths of the family *Phaenidae*. If both ears of a moth are infected, it does not hear the sonar of bats that prey on it and is in greater danger of being devoured together with its colony of mites. The mites employ an ear-choice coordination protocol involving chemical markings of trails, see [6].

## 2. Basic Concepts

We shall phrase our definitions and results for the case  $k = 2$  of choice coordination for two alternatives  $A_1, A_2$ . The reader is referred to the Introduction for the intuitive meaning of the formal definitions.

Let  $\Sigma = \{0, 1, \dots, m-2, e\}$ , where  $m$  is an integer and  $e$  is a marker, be an  $m$ -letter alphabet. Let  $T, E$  stand respectively for transfer and exit. As usual, if  $B$  is a set then  $B^*$  will denote the set of all finite words (sequences) on  $B$ .

*Definition 1.* A process or protocol  $P$  using  $\Sigma$ , is a mapping

$$P: \Sigma^* \rightarrow (\Sigma - \{e\}) \times \{T\} \cup \{(e, E)\} \quad (1)$$

such that

$$P(we) = (e, E), \quad w \in \Sigma^*. \quad (2)$$

Note that if  $P(w) = (\tau, X)$  and  $\tau \neq e$  then  $X = T$  must hold.

The intended interpretation is that  $P$  operates on the pair  $(\alpha, \beta)$  of cells. At any given time,  $P$  is positioned either on the left or on the right cell and the pair contains letters  $(\alpha, \beta)$ ,  $\alpha \in \Sigma$ ,  $\beta \in \Sigma$ . Assume that  $P$  is positioned on the left and about to perform an atomic action. If  $P$  has seen, in its active stages, the sequence  $w = \sigma_1 \sigma_2 \dots \sigma_k$  of symbols (hence  $\sigma_k = \alpha$ ) and  $P(w) = (\tau, X)$ , then  $P$  will replace  $\alpha$  by  $\tau$ . For  $X = T$  the process  $P$  will transfer to the right, and on  $X = E$  it will exit (leave) the computation.

However, for the sake of uniform description of the computation, we shall adopt the convention that on  $E$  the process  $P$  stays on the same side. Since  $X = E$  only when  $\tau = e$ , and because of (2), it follows that in this case the atomic action of  $P$  will be  $(e, E)$  in all subsequent activations of  $P$ .

The manner in which the activities of the processes interlace in any particular computation attempting choice coordination is given by a schedule.

*Definition 2.* A schedule or live sequence is a pair  $(S, p)$  where  $S = i_1 i_2 \dots$ ,  $1 \leq i_j \leq n$ , is an infinite sequence and  $p: \{1, \dots, n\} \rightarrow \{L, R\}$ .

If  $i_j=i$  we say that  $Pi$  is active at time  $j$ . If  $p(i)=L$  we say that  $Pi$  is initially positioned on the left-hand cell, and similarly for  $p(i)=R$ .

We shall now define the computation performed by processes according to a schedule  $(S, p)$ . In the following,  $A$  denotes the empty word,  $\phi$  is the empty set, and  $A-B$  is the set of elements in  $A$  but not in  $B$ .

Note that in our notion of schedule, the time-sequence starts with  $t=1$ . We shall take  $t=0$  to mean the instant of initialization of the computation.

Let  $P_1, \dots, P_n$  be processes using  $\Sigma$ , and let  $(S, p)$  be a schedule.

*Definition 3.* A history for a process  $P$  is a word  $w \in \Sigma^*$ . A configuration  $C$  is a pair  $(\lambda, \rho)$ ,  $\lambda, \rho \in \Sigma$ , a sequence  $(w_1, \dots, w_n)$  of histories, one for each process, and two sets  $LF, RT$ , of processes. If  $Pi \in LF$  ( $Pi \in RT$ ) we say that in  $C$  process  $Pi$  is positioned on the left (right).

*Definition 4.* The computation  $\Gamma$  by the processes according to the schedule  $(S, p)$ ,  $S = i_1 i_2 \dots$ , is the sequence  $C_0, C_1, \dots$ , of configurations, where

$$C_t = \langle (\lambda_t, \rho_t), (w_1(t), \dots, w_n(t)), LF(t), RT(t) \rangle$$

is called the configuration after time  $t$ . The computation  $\Gamma$  is defined inductively as follows. Initialize

$$\begin{aligned} \lambda_0 = \rho_0 = 0, \quad w_j(0) = A, \quad 1 \leq j \leq n, \\ LF(0) = \{Pi | p(i) = L\}, \quad RT(0) = \{Pi | p(i) = R\}. \end{aligned}$$

Assume that  $C_t$  is already defined and let  $i_{t+1} = i$  so that  $Pi$  is the next process active in  $S$ .

If  $Pi \in LF(t)$  and  $Pi(w_i(t)\lambda_t) = (\tau, X)$  then  $(\lambda_{t+1}, \rho_{t+1}) = (\tau, \rho_t)$  and

$$w_i(t+1) = w_i(t)\lambda_t\tau, \quad w_j(t+1) = w_j(t) \quad \text{for } j \neq i. \quad (3)$$

Furthermore, if  $X = T$  then

$$LF(t+1) = LF(t) - \{Pi\}, \quad RT(t+1) = RT(t) \cup \{Pi\},$$

and if  $X = E$  (in which case  $\tau = e$ ) then  $LF(t+1) = LF(t)$  and  $RT(t+1) = RT(t)$ .

The definition for the case  $Pi \in RT(t)$  runs similarly.

*Remark.* It is important to note that the history  $w_\ell(t)$  of any process  $P_\ell$  after time  $t$  is the sequence of symbols that  $P_\ell$  "saw" at the times it was active according to  $S$ ; a process does *not* continuously examine the content of a cell even when it is positioned on the side of that cell. This fact is formalized in (3) by the difference in definition of  $w_\ell(t+1)$  between  $w_i(t+1)$  for the active  $Pi$  and  $w_j(t+1)$ ,  $j \neq i$ .

With the above notations, we can distinguish three possible outcomes of the computation by  $P_1, \dots, P_n$  according to a schedule  $(S, p)$ .

I. For every  $t$ ,  $\lambda_t \neq e$  and  $\rho_t \neq e$ . We shall say that  $\Gamma$  has *not terminated*.

II. For some  $t$ ,  $\lambda_t = \rho_t = e$ . We shall say that  $\Gamma$  led to *contradiction*.

III. Neither I nor II; for some  $t$ ,  $\lambda_t = e$  or  $\rho_t = e$ , but for no  $t < s$ ,  $(\lambda_s, \rho_s) = (e, e)$ . In this case we say that  $\Gamma$  resulted in *choice coordination*.

*Definition 5.* Processes (protocols)  $P_1, \dots, P_n$  are called a *solution for the choice coordination problem (C.C.P.)* if for every schedule  $(S, p)$ , the computation according to this schedule results in choice coordination.

### 3. A Solution for the Choice Coordination Problem

**Theorem 1** (M. Fischer, M. Rabin). *For every  $n$  there exist protocols  $P_1, \dots, P_n$  solving the C.C.P. and using  $n+2$  letters  $\Sigma = \{0, 1, \dots, n, e\}$ .*

*Proof.* We shall present the argument for the case  $k=2$  of two alternatives. Informally the behavior of  $P_i$ ,  $1 \leq i \leq n$ , is described as follows. When  $P_i$  first enters, if it sees 0 it prints  $i$ , and if it sees  $1 \leq j \leq n$  it prints  $\min(i, j)$ ; in either case  $P_i$  transfers sides. Later on, if  $m$  is the smallest non-zero integer in  $P_i$ 's history and  $P_i$  currently sees  $0 \leq j \leq n$ , then  $P_i$  leaves  $j < m$  unchanged and transfers, prints 0 and transfers if  $m < j$ , and prints  $e$  if  $j = m$ .

In short,  $P_i$  always "becomes"  $P_m$  for the smallest  $1 \leq m \leq n$  it has seen, replaces by 0 any  $j > m$  it sees, and marks  $e$  when it sees its current name  $m$  for the second time.

Using the standard notation: Let  $w = x_1 x_2 \dots x_\ell \in \Sigma^*$ ,  $1 \leq \ell$ , and  $m = \min_{0 < x_i} x_i$ ,

then

$$\begin{aligned} P_i(0) &= (i, T), & P_i(j) &= (\min(i, j), T) & \text{for } 0 < j \leq n, \\ P_i(e) &= P_i(we) = (e, E), \\ P_i(wj) &= \begin{cases} (j, T) & 0 \leq j < m \\ (0, T) & m < j \leq n \\ (e, E) & j = m. \end{cases} \end{aligned}$$

To prove termination, let  $(S, p)$  be a schedule and recall that  $S = i_1 i_2 \dots$  is infinite. Assume by way of contradiction that the computation  $\Gamma$  does not terminate (i.e. that  $(e, x)$  or  $(x, e)$  never appears; see Definition 4 and the terminology following it). Let  $i$  be the minimal index of a process appearing in  $S$ . Assume that  $t$  is the first time that  $P_i$  appears in  $S$  and that  $p(i) = L$ . At time  $t+1$  the content is  $(i, \rho)$ . Because  $i \leq i_s$  for every  $i_s \in S$ , the value  $i$  on the left is never changed subsequent to time  $t+1$ .

Let  $P_j$  be a process which is active an infinite number of times in  $S$ . Let  $t+1 < t_1 < t_2 < t_3$  be three consecutive times at which  $P_j$  is active in  $S$ , such that at time  $t_1$  the process is on the left (seeing  $i$ ). Then at time  $t_3$  process  $P_j$  will replace  $i$  by  $e$ . Thus termination is established.

Assume next that some schedule  $(S, p)$  leads to  $(e, e)$ . Let  $i$  be the last content of the left-hand side  $L$  before the change to  $e$ , and similarly for  $j$  on the right-hand side  $R$ . Since  $i \neq 0$ ,  $j \neq 0$ , we must have  $i \neq j$  (by induction on computations), so that w.l.g.  $i < j$ . Let  $P_\ell$  and  $P_r$  be the processes which respectively change the  $L$ -side and the  $R$ -side into  $e$ . Let  $t$  be the time at which  $P_\ell$  has seen or written  $i$  in  $L$ . The contents of  $L$  will now remain  $i$  until the change to  $e$ .

Let  $t < u$  be the next time that  $P\ell$  was active (on the right). Then immediately after time  $u$  the contents was  $(i, 0)$ .

Thus  $Pr$  must visit the  $R$ -side after time  $u$  and before marking the  $R$ -side by  $e$ . Hence  $Pr$  must also visit  $L$  after time  $u$  and before  $L$  is  $e$ . But then the contents of  $L$  is  $i$  so that  $Pr$  "becomes"  $Pi$ . This contradicts the assumption that later  $Pr$  change  $j > i$  into  $e$  on the  $R$ -side.

The above algorithm and the proof of correctness apply also in the case of choice coordination for any number  $k$  of alternatives.

#### 4. The Lower Bound

The task of establishing a lower bound on the number of letters necessary for coordinating  $n$  processes is made difficult by the generality of our notion of a program or process. The atomic action of a process at a particular time within a computation depends on its entire history up to that time, and we make no assumptions on the nature of that dependency. The proof for the lower bound will be effected by combining a pigeon-hole argument with some graph theoretic results.

We shall be interested in what a process  $P$  is *about* to do at a given time. This is expressed in the following.

*Definition 6.* We say that in configuration  $C$  (see Definition 3)  $Pi$  is *primed on the left side (right side)* to change  $\alpha$  into  $\beta$  (primed to do  $\alpha \rightarrow \beta$ , for short) if

$$Pi \in LF \quad (Pi \in RT) \text{ and } Pi(w, \alpha) = (\beta, X).$$

If  $\Gamma$  is a computation of  $P1, \dots, Pn$ , then  $Pi$  is primed on the left side to do  $\alpha \rightarrow \beta$  at time  $t$  if it is so primed in the configuration  $C(t-1)$  immediately after time  $t-1$ .

Informally,  $Pi$  primed on the left side to do  $\alpha \rightarrow \beta$  at time  $t$  means that according to the schedule  $Pi$  will next be active at time  $t_1 \geq t$ , and at that time the contents will be  $(\alpha, \sigma)$ , then just after time  $t_1$  the contents will be  $(\beta, \sigma)$ . Note, however, that the contents at time  $t_1$  could be  $(\gamma, \sigma)$ ,  $\gamma \neq \alpha$ .

Let  $(S, p)$  be a schedule,  $\Gamma$  the computation according to  $(S, p)$ . After time  $t-1$ , let  $G \subseteq LF(t-1)$  be a set of processes. With  $G$  we associate a directed graph  $(\Sigma, E)$  on  $\Sigma = \{0, 1, \dots, m-2, e\}$  as follows. For each  $Pi \in G$  specify one pair  $\alpha, \beta$  such that  $Pi$  is primed to do  $\alpha \rightarrow \beta$  at time  $t$ , and put the directed edge  $\langle \alpha, \beta \rangle$  into  $E$ . Even though the associated graph is not unique, we shall denote it by  $G$  because in any given context it will be clear which edges are chosen.

If  $G = (V, E)$  is a directed graph we say that vertex  $\alpha$  is *connected* to vertex  $\beta$  if  $\alpha = \beta$  or there exists a path  $\alpha = \alpha_1, \dots, \alpha_k = \beta$  such that  $\langle \alpha_j, \alpha_{j+1} \rangle \in E$ ,  $1 \leq j < k$ . If  $\alpha$  is connected to  $\beta$  and  $\beta$  is connected to  $\alpha$  we say that  $\alpha$  and  $\beta$  are *strongly connected* and write  $\alpha \sim \beta$ .

The relation  $\sim$  is an equivalence relation on the set of all vertices of  $G$ . The equivalence classes under  $\sim$  are called the *strongly connected components* of  $G$ .

Let  $K \subseteq V$  be a strongly connected component of  $G$ . We call  $K$  a *terminal component* of  $G$  if  $\langle \alpha, \beta \rangle \in E$  and  $\alpha \in K$  imply  $\beta \in K$ . Every directed graph has some terminal components. In the extreme case that  $E = \emptyset$  (there are no edges) the terminal components consist of single vertices. If  $G$  is strongly connected then  $V$  is a terminal component.

Consider the following process of adding edges to a graph with set  $0 \in V$  of vertices. The graph  $G(0) = (V, \emptyset)$  has no edges. If  $G(i) = \langle V, E_i \rangle$  is not strongly connected, then  $G(i+1) = (V, E_{i+1})$  is obtained by choosing a terminal strongly connected component  $K$  of  $G(i)$ , an  $\alpha \in K$  and a  $\beta \notin K$ , and setting  $E_{i+1} = E_i \cup \{\langle \alpha, \beta \rangle\}$ .

**Lemma 2.** *If the graphs  $G(0), G(1), \dots, G(M)$  are a sequence of the above type and if the set  $V$  of vertices has  $m$  elements, then  $M \leq 2m - 2$ .*

*Also, if at every stage  $k$  in the construction of the sequence  $G(0), \dots$ , the terminal strongly connected component  $K$  of  $G(k)$  for which  $\alpha \in K$  is connected from 0, i.e. there is a directed path in  $G(k)$  from 0 to  $\alpha$ , and if  $\langle \alpha, \beta \rangle$  is added to obtain  $G(k+1)$ , then the strongly connected component  $K'$  which contains  $\beta$  in  $G(k+1)$  is a terminal component.*

*Proof.* Let  $S(i)$  denote the number of strongly connected components and let  $T(i)$  denote the number of terminal components in  $G(i)$ . Initially  $S(0) + T(0) = 2m$ .

Each step from  $G(i)$  to  $G(i+1)$ ,  $0 \leq i < M$ , reduces  $T(i)$  or  $S(i)$  by at least 1. Thus  $2 \leq S(M) + T(M) \leq 2m - M$ . Hence  $M \leq 2m - 2$ .

We leave the proof of the second assertion to the reader.

We shall need another combinatorial result.

**Lemma 3.** *Let  $H = X_1 X_2, \dots, X_{2k}$ ,  $X_i \in \{L, R\}$  be a sequence of even length and let  $F(x)$  be the reversal function such that  $F(R) = L$ ,  $F(L) = R$ . There exists an index  $1 \leq i \leq 2k$  such that the sequence*

$$H_1 = X_1 \dots X_{i-1} F(X_i) F(X_{k+1}) \dots F(X_{2k}) \tag{4}$$

*contains an equal number of L's and R's.*

*Proof.* By induction on  $k$ . The case  $k=1$  is obvious. Assume the result true for all sequences of length  $2k-2$  and let  $H = X_1 \dots X_{2k}$ . If  $X_j = L$ ,  $1 \leq j \leq 2k$ , or  $X_j = R$ ,  $1 \leq j \leq 2k$  then choose  $i = k+1$ . Otherwise there exists a  $1 \leq j \leq 2k$  such that  $X_j = L$ ,  $X_{j+1} = R$  or vice-versa.

Let  $i$  be an index such that applying  $F$  to the sequence  $H' = X_1 \dots X_{j-1} X_{j+2} \dots X_{2k}$  from  $X_i$  onwards will produce  $H'_1$  with the desired property. Here  $1 \leq i \leq j-1$  or  $j+2 \leq i \leq 2k$ . The same  $i$  will work for  $H$  to produce the  $H_1$  of (4).

**Theorem 4.** *Let  $P_1, \dots, P_n$  be processes on the alphabet  $\Sigma = \{0, 1, \dots, m-2, e\}$ . If  $8m^3 \leq n$  then these processes are not a solution for the choice coordination problem.*

*Proof.* The overall plan is to define a finite schedule  $(S, p)$ ,  $S = i_1 i_2 \dots i_{t-1}$ , where  $p: \{i_1, \dots, i_{t-1}\} \rightarrow \{L, R\}$ , so that after time  $t-1$  the cells have contents

$(\alpha, \alpha)$ . Furthermore, there will exist symbols  $\lambda_1 = \alpha, \dots, \lambda_\ell = e, \rho_1 = \alpha, \dots, \rho_k = e$  and processes  $Pj_1, \dots, Pj_{\ell-1}, Pm_1, \dots, Pm_{k-1}$ , such that at time  $t$  process  $Pj_i, 1 \leq i \leq \ell-1$ , is primed on the left to do  $\lambda_i \rightarrow \lambda_{i+1}$ , and  $Pm_i, 1 \leq i \leq k-1$ , is primed on the right to do  $\rho_i \rightarrow \rho_{i+1}$ .

The construction of a finite schedule with these properties will turn out to be possible unless we shall encounter at some time  $s < t-1$  a hitherto unused process  $P_i, i \notin \{i_1, \dots, i_s\}$ , and a side  $X \in \{L, R\}$  with the following behavior. The computation under the schedule  $(S', p')$  will be nonterminating (see the terminology following Definition 4), where  $S' = i_1 i_2 \dots i_s i i i \dots, p'(i) = X$ , and  $p'(j) = p(j)$  for  $j \neq i$ . In this case  $P1, \dots, Pn$  are not a solution for the C.C.P.

The existence of a finite schedule as above also entails that  $P1, \dots, Pn$  are not a solution. Namely, under the finite schedule  $(i_1 i_2 \dots i_{t-1} j_1 \dots j_{\ell-1} m_1 \dots m_{k-1}, p)$ , the computation by  $P1, \dots, Pn$  produces  $(e, e)$ .

To simplify indices we shall first assume  $16m^3 \leq n$  and at the end improve to  $8m^3 \leq n$ . The construction of the finite schedule is achieved in stages. At stage  $k \leq 2m$  we have a schedule  $(S_k, p_k)$ , where  $S_k$  is of length  $t_k$  and the computation by  $P1, \dots, Pn$  according to this schedule produces a configuration after time  $t_k$  (see Definition 4) with the following properties.

1. The contents of the cells is  $(\alpha_k, \alpha_k)$ .
2. The schedule  $(S_k, p_k)$  invokes just the processes  $P1, \dots, P8m^2k$ . In particular, the domain of  $p_k$  is  $\{1, \dots, 8m^2k\}$ .
3. There exist pairwise disjoint subsets  $GL_i(k) \subseteq LF(t_k)$  and  $GR_i(k) \subseteq RT(t_k), 1 \leq i \leq 4m-2k$  such that their associated graphs, also denoted by  $GL_i(k)$  and  $GR_i(k)$ , satisfy
  - a) All of these graphs on  $\Sigma$  have the same strongly connected components.
  - b) The symbol  $\alpha_k$  lies in a *terminal* strongly connected component, call it  $K$ , of these graphs.

Recall that a subset  $G \subseteq LF(t_k)$  is viewed as a graph on  $\Sigma$  by specifying for every  $Pi \in G$  a pair  $\alpha, \beta \in \Sigma$  so that  $Pi$  is primed on the left at time  $t_k+1$  to do  $\alpha \rightarrow \beta$ .

At stage 0 we define  $\alpha_0 = 0, (S_0, p_0) = (A, \phi), GL_i(0) = GR_i(0) = \phi, 1 \leq i \leq 4m$ . Assuming that  $(S_k, p_k)$  is already defined and  $k < 2m$  we shall define  $(S_{k+1}, p_{k+1})$  or, failing to do so, get that  $P1, \dots, Pn$  are not a solution. We need a preliminary observation.

Let  $\beta, \gamma \in K$ , where  $K$  is the strongly connected component in 3.b), and let  $GL_i(k)$  be any one of the graphs in 3. Since  $K$  is a component of  $GL_i(k)$ , there exists a sequence of *pairwise different* symbols  $\lambda_1 = \alpha_k, \lambda_2, \dots, \lambda_\ell = \beta$  (if  $\alpha_k = \beta$  then  $\ell = 1$ ), and processes  $Pj_1, \dots, Pj_{\ell-1} \in GL_i(k)$  so that  $Pj_r, 1 \leq r \leq \ell-1$  is primed on the left to do  $\lambda_r \rightarrow \lambda_{r+1}$ . Thus the schedule  $(S_k j_1 j_2 \dots j_{\ell-1}, p_k)$  will lead to the contents  $(\beta, \alpha_k)$ . A similar statement holds for the right-hand side. Thus an appropriate extension  $(S', p_k)$  of  $(S_k, p_k)$  would lead to  $(\beta, \gamma)$ . In particular this implies that if  $e \in K$  then  $P1, \dots, Pn$  are not a solution, so that we may assume  $e \notin K$ .

Consider the processes  $P8m^2k+1, \dots, P8m^2k+8m^2$ , which have not been used in  $(S_k, p_k)$ . Since  $k < 2m$  and (for now)  $16m^3 \leq n$  we have  $8m^2k+8m^2 \leq n$ . At time  $t_k+1$  start  $P8m^2k+1$  on side  $L$  and run it. In other words, construct



the schedule  $(S, p)$  where  $S = S_k(8m^2k+1)(8m^2k+1)\dots$ , and  $p$  extends  $p_k$  by  $p(8m^2k+1) = L$ . In the computation according to  $(S, p)$  the contents of the cells after times  $t_k, t_k+1, t_k+2, \dots$  will be  $(\alpha_k, \alpha_k), (\lambda_1, \alpha_k), (\lambda_1, \rho_1), \dots$ . Two cases are possible. Either  $\lambda_i, \rho_i \in K$  - the terminal component containing  $\alpha_k$ , for  $i = 1, 2, \dots$ , (since  $e \notin K$ , in this case  $(S, p)$  leads to a non-terminating computation), or there is a smallest  $i$  such that  $\lambda_i \notin K$  or  $\rho_i \notin K$ . In this case truncate  $S$  to a finite sequence  $S'$ , stopping just at the point when  $P8m^2k+1$  is primed to do  $\lambda_i \rightarrow \lambda_{i+1}$  where  $\lambda_i \in K, \lambda_{i+1} \notin K$ , or  $\rho_i \rightarrow \rho_{i+1}$  where  $\rho_i \in K, \rho_{i+1} \notin K$ , as the case may be. For example if  $\lambda_1 \notin K$  then  $S' = S_k$ , and  $P8m^2k+1$  will be primed on the left to do  $\alpha_k \rightarrow \lambda_1$ . And if  $\lambda_1, \rho_1, \lambda_2 \in K$  but  $\rho_2 \notin K$  then  $S' = S_k(8m^2k+1)(8m^2k+1)(8m^2k+1)$ ; after time  $t_k+3$ , process  $P8m^2k+1$  will be primed on the right to do  $\rho_1 \rightarrow \rho_2$ .

Next start  $P8m^2k+2$  on side  $L$  and extend  $(S', p')$  to  $(S'', p'')$  in the above manner so that after the computation according to  $(S'', p'')$ , process  $P8m^2k+2$  is primed either on the left or on the right to do  $\beta \rightarrow \gamma$  where  $\beta \in K, \gamma \notin K$ . Continuing in this manner, starting every  $P8m^2k+i, 1 \leq i \leq 8m^2$ , on the  $L$ -side, we extend  $(S_k, p_k)$  to  $(S^{(8m^2)}, p^{(8m^2)})$  so that after the computation according to this schedule, every  $P8m^2k+i, 1 \leq i \leq 8m^2$ , is primed to do  $\beta(i) \rightarrow \gamma(i)$ .

There are at most  $m$  possible values for  $\gamma(i)$ , so there must exist  $8m$  indexes  $8m^2k+1 \leq j_1 < j_2 < \dots < j_{8m} \leq 8m^2(k+1)$ , for which  $\gamma(j_1) = \gamma(j_2) = \dots = \gamma(j_{8m}) = \gamma$ . Thus at the end of our schedule,  $Pj_\ell, 1 \leq \ell \leq 8m$ , is primed on side  $X_\ell \in \{L, R\}$  to do  $\beta(j_\ell) \rightarrow \gamma$  where  $\beta(j_\ell) \in K$  and  $\gamma \notin K$ .

We would like to add one  $Pj_\ell$  to each  $GL_i(k)$  and each  $GR_i(k), 1 \leq i \leq 4m-2k$ , thereby creating the desired  $GL_i(k+1), GR_i(k+1)$  and their associated graphs. This cannot be directly done because the  $X_\ell$  need not be evenly distributed between  $L$  and  $R$ . To rectify the situation we use Lemma 3.

According to Lemma 3 there exists an  $1 \leq i \leq 8m$  such that the sequence  $X_1 \dots X_{i-1} F(X_i) \dots F(X_{8m})$ , where  $F(L) = R$  and  $F(R) = L$ , has an equal number of  $L$ 's and  $R$ 's. Denote by  $t$  the time in the computation according to  $(S^{(8m^2)}, p^{(8m^2)})$  after which  $Pj_{i-1}$  is primed (at time  $t+1$ ) on side  $X_{i-1}$  to do  $\beta(j_{i-1}) \rightarrow \gamma$ . Let the contents of the cells after time  $t$  be  $(\lambda, \rho)$  where  $\lambda, \rho \in K$ . Actually  $\lambda = \beta(j_{i-1})$  if  $X_{i-1} = L$  and  $\rho = \beta(j_{i-1})$  if  $X_{i-1} = R$ , but this fact is not used.

By our definition of  $S^{(8m^2)}$  it has the form

$$S^{(8m^2)} = S_k(8m^2k+1) \dots j_{i-1}(j_{i-1}+1) \dots 8m^2(k+1),$$

where the displayed occurrence of  $j_{i-1}$  is at time  $t$  in the computation. Denote by  $\bar{S}$  the initial segment of  $S^{(8m^2)}$  up to and including the displayed  $j_{i-1}$ ; the length of  $\bar{S}$  is  $t$ . Since  $\lambda, \rho \in K$ , we can activate, subsequently to  $\bar{S}$ , processes in  $GL_{4m-2k}$  on the left-side and in  $GR_{4m-2k}$  (on the right-side), to transform the contents of the cells from  $(\lambda, \rho)$  to  $(\rho, \lambda)$ . This is done in the manner previously detailed in the proof just after the definition of  $(S_k, p_k)$ . The order of activations of those processes extends  $\bar{S}$  to  $\bar{S}\ell_1 \dots \ell_s$ . Define now

$$S' = \bar{S}\ell_1 \dots \ell_s(j_{i-1}+1) \dots 8m^2(k+1).$$

Here  $P\ell_1, \dots, P\ell_s$  are the processes used to effect the flip-over of the contents of the cells; we have  $\ell_j \leq 8m^2k, 1 \leq j \leq s$ . Note that in the computation accord-

ing to  $S'$ , after time  $t+s$  the contents of the cells is  $(\rho, \lambda)$ . Also,  $j_i, \dots, j_{8m}$  appear in  $S'$  after  $\ell_s$ .

Change  $p^{(8m^2)}$  into  $p_{k+1}$  by setting  $p_{k+1}(h) = p^{(8m^2)}(h)$ ,  $1 \leq h \leq j_{i-1}$ , and  $p_{k+1}(h) = R$  for  $j_{i-1} + 1 \leq h \leq 8m^2(k+1)$ . Recall that  $p^{(8m^2)}(h)$  was  $L$ ,  $j_{i-1} + 1 \leq h \leq 8m^2(k+1)$ . Let us examine the effect of the computation according to the schedule  $(S', p_{k+1})$ . After time  $t+s$ , i.e. at the end of the  $\bar{S}\ell_1 \dots \ell_s$  segment, the contents of the cells is  $(\rho, \lambda)$ . At times  $t+s+1, t+s+2, \dots$ , the computation will run exactly like the computation according to  $(S^{(8m^2)}, p^{(8m^2)})$  at times  $t+1, t+2, \dots$ , except that left and right are interchanged in the sense that everything that occurred in the computation according to  $(S^{(8m^2)}, p^{(8m^2)})$  on the  $L$ -side will occur in the computation according to  $(S', p_k)$  on the  $R$ -side, and similarly with  $L$  and  $R$  interchanged. Since  $Pj_i, \dots, Pj_{8m}$ , were first activated in  $(S^{(8m^2)}, p^{(8m^2)})$  after time  $t$ , and in  $(S', p_{k+1})$  after time  $t+s$ , it follows that at the end of the computation according to  $(S', p_{k+1})$  these processes will be primed on sides  $F(X_i) \dots F(X_{8m})$  to do  $\beta(j_i) \rightarrow \gamma, \dots, \beta(j_{8m}) \rightarrow \gamma$ . Thus at the end of this computation exactly  $4m$  of the  $Pj_1, \dots, Pj_{8m}$ , will be primed on the left and exactly  $4m$  of these processes will be primed on the right.

Also, at the end of  $(S', p_{k+1})$  the contents of the cells will be  $(\sigma_1, \sigma_2)$ , where  $\sigma_1, \sigma_2 \in K$ . Assume, without loss of generality, that  $Pj_1$  ends up primed on the left and  $Pj_2$  ends up primed on the right. Since  $\beta(j_1), \beta(j_2) \in K$ , we can use processes in  $GL_{4m-2k-1}(k)$  on the  $L$ -side and processes in  $GR_{4m-2k-1}(k)$  on the  $R$ -side, to extend  $S'$  to  $S''$  so that at the end of the computation according to  $(S'', p_{k+1})$  the contents is  $(\beta(j_1), \beta(j_2))$ .

Define  $S_{k+1} = S''j_1j_2$ . Since  $Pj_1$  was primed on the left side to do  $\beta(j_1) \rightarrow \gamma$ , and similarly for  $Pj_2$  on the right side, at the end of the computation according to  $(S_{k+1}, p_{k+1})$  the contents is  $(\gamma, \gamma)$ . Define  $\alpha_{k+1} = \gamma$ .

That  $(S_{k+1}, p_{k+1})$  has the properties 1–2 of  $(S_k, p_k)$  is obvious. In order to establish property 3, add to each of  $GL_i(k)$ ,  $1 \leq i \leq 4m-2k-2$ , (a different) one of the  $Pj_3, \dots, Pj_{8m}$  which is primed on the  $L$ -side and call the resulting graph  $GL_i(k+1)$ ; proceed similarly with the  $GR_i(k)$ . Ba 3.a), all the  $GL_i(k)$ ,  $GR_i(k)$ , had the same strongly connected components. Every  $Pj_\ell$  adds a new edge  $\langle \beta(j_\ell), \alpha_{k+1} \rangle$  where  $\beta(j_\ell) \in K$  – a common strongly connected component. It follows that all  $GL_i(k+1)$ ,  $GR_i(k+1)$ ,  $1 \leq i \leq 4m-2(k+1)$ , have the same strongly connected components. By the second assertion in Lemma 2, the process of obtaining  $GL_i(k+1)$  from  $GL_i(k)$  insures that  $\alpha_{k+1}$  is in a terminal strongly connected component of  $GL_i(k+1)$ . Thus 3.a), 3.b), are established for  $(S_{k+1}, p_{k+1})$ .

To get by with  $8m^3 \leq n$ , observe that at stage  $k+1$  of the construction we want to add processors to just  $GL_i(k)$ ,  $GR_i(k)$ ,  $1 \leq i \leq 4m-2(k+1)$ , i.e., to  $8m-4(k+1)$  sets and not  $8m$  sets. Thus it will suffice to activate  $(8m-4k)m$  new processors at this stage, which brings the total to  $8m^3$ .

To conclude the proof of Theorem 4, observe that the sequence of graphs  $GL_1(0), GL_1(1), \dots$  satisfies the conditions of Lemma 2 so that it has length at most  $2m-2$ . Thus for some  $k \leq 2m-2$ , the above construction cannot be continued. This means that either we have encountered at stage  $k$  a schedule leading to a non-terminating computation, or at stage  $k$  we had  $e \in K$  for the strongly connected component  $K \ni \alpha_k$ . But in the latter case we had a schedule leading to a contradiction  $(e, e)$ . End of proof!  $\square$

The author does not see a way of using fewer than  $O(m^2)$  new processes at each stage, and no argument insuring fewer than  $O(m)$  stages. Neither does there seem to be a way, within the strategy of this proof, to reuse primed processes  $Pj \in GL_i(k)$  once they were activated to do a transition  $\alpha \rightarrow \beta$ . For it was essential that with respect to every process that was introduced and brought to a primed state in the construction of  $(S_{k+1}, p_{k+1})$  from  $(S_k, p_k)$ , we had the freedom to initiate it on the  $L$ -side or on the  $R$ -side. But once a process is used, the side is determined. Thus calling for  $O(m^2)$  new processes at each stage seems unavoidable.

On the other hand, examination of the proof reveals that we have established a claim somewhat stronger than Theorem 4. Namely, if  $8m^3 \leq n$  then there exists a schedule  $(S, p)$  leading to a non-terminating computation by  $P1, \dots, Pn$ , where for some  $1 \leq i \leq n$ ,  $S$  has the form

$$S = i_1 i_2 \dots i_s iii \dots, \quad (5)$$

or there exists a schedule according to which  $P1, \dots, Pn$  compute  $(e, e)$ .

The special form (5) of the schedule  $(S, p)$  leading to non-termination, suggests the possibility that the assumption  $8m^3 \leq n$  may be too strong.

As was pointed out in the Introduction, in terms of bit-count, which is the significant measure for implementations, we are not far from optimal: An alphabet  $\Sigma$  requiring at least  $1/3 \log_2 n$  bits is necessary, and requiring  $\log_2 n$  bits is sufficient, for a solution of the C.C.P. for  $n$  processes. The next significant step is to reduce the number of bits to say 8, for every number of processes, by use of randomized protocols.

## 5. Randomizing Protocols for the C.C.P.

The difficulty in choice coordination arises out of the initial symmetry of the contents of the cells, a symmetry that is impossible to break unless sufficiently many symbols are available in the synchronization alphabet. This suggests the idea of using randomizing protocols that will break the symmetry with very high probability. The same idea of randomization was successfully used for various synchronization problems [3, 4] and seems to be a generally applicable method in this area.

Let us assume that we have an  $m+2$ -valued alphabet  $\Sigma = \{0, 1, 2, \dots, m, e\}$  where  $m$  is even. The numbers  $1, \dots, m$  are viewed as grouped in pairs  $\{1, 2\}, \{3, 4\}, \dots, \{m-1, m\}$ . Each of the processes  $P1, \dots, Pn$ , is able to make a random binary choice between two items. By random  $\{i, i+1\}$  we mean one of the two numbers  $i, i+1$  chosen with equal probabilities. All the processes have the same program. If, upon first entering, say on the  $L$ -side,  $P$  sees 0 it writes random  $\{1, 2\}$  and transfers sides. In general, if the last letter that  $P$  saw is  $\alpha$  and it currently sees  $\beta$  then if  $\alpha < \beta$  it transfers side, if  $\beta < \alpha$  process  $P$  marks the cell, i.e. changes the  $\beta$  into  $e$ , and if  $\alpha = \beta < m-1$  then  $P$  replaces  $\beta$  by random  $\left\{ 2 \left\lceil \frac{\beta}{2} \right\rceil + 1, 2 \left\lceil \frac{\beta}{2} \right\rceil + 2 \right\}$  i.e., by random  $\{i, i+1\}$  where  $\{i, i+1\}$  is the next pair after the pair containing  $\beta$ . Thus if  $P$  last saw 13 on the  $L$ -side and

now sees 13 on the  $R$ -side (this does *not* mean that the current contents of the cells is (13, 13)) then it will write random  $\{15, 16\}$  on the  $R$ -side. Formally, for  $0 \leq \alpha, \beta \leq m, w \in \Sigma^*$ ,

$$P(we) = (e, E), \quad P(0) = (\text{random } \{1, 2\}, T), \quad P(\alpha) = (\alpha, T) \quad \text{for } 1 \leq \alpha \leq m,$$

$$(6) \quad P(w\alpha\beta) = \begin{cases} (\beta, T) & \alpha < \beta \\ (e, E) & \beta < \alpha \\ (\text{random}(i, i+1), T), & i = 2 \lceil \frac{\beta}{2} \rceil + 1, \alpha = \beta < m - 1. \end{cases}$$

Note that  $P$  is not always defined, for example  $P(7, 13, m-1, m-1)$  is undefined.

The notions of a schedule and of a computation  $\Gamma$  by randomizing processes  $P_1, \dots, P_n$  according to a schedule, is exactly as in Definitions 2-4. Any particular computation  $\Gamma$  involves specific outcomes of random  $\{i, i+1\}$  whenever this operator was used. To a given schedule  $\pi = (S, p)$  there correspond many computations according to  $\pi$ . The  $1/2$  probabilities of the outcomes  $i$  and  $i+1$  in random  $\{i, i+1\}$  entail a probability  $Pr_\pi(\Gamma)$  for a computation  $\Gamma$  according to  $\pi$ . Note that  $Pr_\pi$  is not conditional probability, since  $\pi$  is fixed. It is easy to see that an event such as  $\Gamma$  non-terminating (see I following Definition 4), has a well defined probability.

**Theorem 5.** *If each of the processes  $P_1, \dots, P_n$  on the alphabet  $\Sigma = \{0, 1, \dots, m, e\}$  is defined by (6), then for every schedule  $(S, p)$  a computation  $\Gamma$  according to this schedule will never lead to a contradiction (i.e., contents  $(e, e)$ ), and will terminate with probability greater than  $1 - 1/2^{m/2}$ .*

*Furthermore, if  $P_i$  is active in  $S$   $2k \leq m$  or more times, then the probability that by the  $2k^{\text{th}}$  activation  $P_i$  will find or write  $e$  is greater than  $1 - 1/2^k$ .*

*Proof.* Let us start by showing that  $(e, e)$  never arises. Let  $\Gamma = C_0, C_1, \dots$ , be a computation according to the schedule  $(S, p)$  and let

$$(0, 0), (\lambda_1, \rho_1), \dots, (\lambda_t, \rho_t), \dots$$

be the sequence of contents of the pair of cells in  $C_0, C_1, \dots$ . Since a change of a contents of a cell involves randomly drawing from the next pair  $i, i+1$ , we have  $\lambda_1 \leq \lambda_2 \dots$  and  $\rho_1 \leq \rho_2 \dots$ . Assume by way of contradiction that  $\lambda_t = \rho_t = e$  and that  $P_i$  and  $P_j$  were the processes that, respectively, changed the contents of the left-hand cell at time  $u+1$  and the contents of the right-hand cell at time  $s+1$  into  $e$ . Let  $P_i$ 's history before time  $u$  be  $a\rho_v$  where  $a \in \Sigma^*$  and  $v \leq u$ , and  $P_j$ 's history before time  $s$  be  $b\lambda_w$   $b \in \Sigma^*$ ,  $w \leq s$ . Since at time  $u+1$  process  $P_i$  changes the  $L$ -side into  $e$  we must have  $\rho_v > \lambda_u$ , and similarly  $\lambda_w > \rho_s$ .

Assume now  $s < u$ . Process  $P_i$  must have last visited the  $R$ -side before time  $s+1$  because at time  $s+1$  process  $P_j$  was there, and after time  $s+1$  the contents were  $(\lambda_{s+1}, e)$ ; hence  $s \geq v$ . Thus  $\lambda_w > \rho_s \geq \rho_v > \lambda_u$  and  $\lambda_w > \lambda_u$ . But  $w \leq u$ , a contradiction.

The only way in which  $\Gamma$  will not terminate is if for some  $t$  the contents  $(\lambda_t, \rho_t)$  are  $(m-1, m-1)$  or  $(m, m)$ . For this to occur there must be times

$t_1 < t_2 < \dots < t_{m/2} = t$  such that  $\lambda_{t_1} = \rho_{t_1} \in \{1, 2\}$ ,  $\lambda_{t_2} = \rho_{t_2} \in \{3, 4\}$ ,  $\dots$ . Thus  $m/2$  times, independent choices random  $\{i, i+1\}$  on the  $L$ -side and on the  $R$ -side have produced the same value. The probability of a single such event is  $1/2$  and the probability of  $m/2$ -fold repetition is  $1 - 1/2^{m/2}$ .

The assertion concerning waiting-time for the individual process  $Pi$  until C.C. is proved similarly.

If we take  $m+2=256$  so that 8 bits suffice for the alphabet, then choice coordination will be achieved with probability  $1 - 1/2^{127}$ . Also if we count each time that a process  $Pi$  visits the  $L$ -side cell as a round, then the expected number of rounds by  $Pi$  in a computation  $\Gamma$  before C.C. is 2.

By slightly modifying the protocol (6) we can reduce the expectation and variance of the waiting time until C.C. of any process participating in the schedule. Assume that  $|\Sigma|=1002$  so that 10 bits suffice for implementation. Divide the integers  $1 \leq i \leq 1000$  into groups of 100,  $\{1, \dots, 100\}$ ,  $\{101, \dots, 200\}$ ,  $\dots$ . The random draw in case of equality will be from the next group of 100, so that we use random  $\{i+1, \dots, i+100\}$  for  $i=0, 100, \dots, 900$ . The probability of not breaking the symmetry at each stage is  $1/100$  so that the probability for non-termination is smaller than  $1/100^{10} = 10^{-12}$ . The probability of a process  $Pi$  making, say, two rounds without C.C. is at most  $100^{-2}$ .

Could we improve the result concerning randomizing protocols and obtain a solution involving a fixed alphabet for the C.C.P. with the properties that we never get a contradiction and the probability of non-termination is 0? If we generalize Definition 1 of a process by introducing randomness into (1), we get the general notion of a randomizing process. A careful reading of the proof of Theorem 4 shows that for randomizing processes we get

**Theorem 6.** *Let  $P1, \dots, Pn$  be randomizing processes on an  $m$ -letter alphabet such that for every schedule  $\pi=(S, p)$ : (1) No computation  $\Gamma$  according to  $\pi$  leads to the contradiction (e.e.). (2) The probability for a computation  $\Gamma$  to be non-terminating is 0. Then we must have  $1/2 \cdot \sqrt[3]{n} \leq m$ .*

## 6. Conclusions

The C.C.P. lends itself to many interpretations both in hardware and in software situations. For example, in the course of a computation,  $k$  almost identical versions  $A_1, \dots, A_k$  of a text are being generated. Processes  $P1, \dots, Pn$  have to agree on one of these as the commonly used version. Thus a C.C.P. arises.

Our analysis delineates what can be done by classical deterministic processes to solve the C.C.P. It turns out that to solve the C.C.P. for  $n$  deterministic processes, an alphabet  $\Sigma$  requiring  $O(\log_2 n)$  bits is necessary and sufficient. In terms of bit-count the disparity between our upper and lower bound results is small.

We suggest the approach employing randomization as a very practical and convenient paradigm for solving the C.C.P. and in fact other problems of synchronization and coordination. As indicated in Sect. 5, there are many

possible variations of the randomization method. Thus one can tailor a version of randomizing protocols to suit a particular problem.

It is very important that the protocols  $P_1, \dots, P_n$  for the randomizing solution for the C.C.P. are all identical, use a very small alphabet, and the whole thing is independent of  $n$ . One can envision situations where  $n$  will be very large, and where the set of processes participating in the choice coordination computation is not known in advance. Thus trying to impose different protocols depending on  $n$ , as is necessary in the classical solution, becomes cumbersome.

In practice the C.C. protocol of Theorem 5 does not even require use of a random number generator by  $P$ . We can, so to speak, randomize the whole technology in advance. Assume that we produce many microprocessors on chips and that we know that during the lifetime of these processors certain subsets of the ensemble will have to participate in up to a billion billions ( $10^{18}$ ) choice coordination computations. During production we can incorporate into each chip a different randomly generated 127-bit sequence which codes a random choice of one element from each of the pairs  $\{1, 2\}, \dots, \{253, 254\}$ . Each processor will play in every encounter in which it participates, using its fixed random sequence. Under the reasonable assumption that choice coordination tasks and the schedules of activation of the processors will be independent of the preprepared random sequences, it follows from Theorem 5 that the probability of failure to reach C.C. in one or more of the  $10^{18}$  possible encounters is smaller than  $10^{18} \cdot 2^{-127} \leq 2^{-87}$ . Thus we have high reliability for absence of even one breakdown for the whole lifetime of the system. The coordination alphabet in this example requires just one byte.

Finally, Theorems 5 and 6 taken together illustrate an interesting phenomenon. If we are willing to tolerate the practically negligible  $2^{-127}$  probability of failure then a fixed 256-letter alphabet and a very simple protocol will solve the C.C.P. independently of the number of processes. But if we insist on probability 0 of failure, then complexity goes up as  $\sqrt[3]{n}$  with the number  $n$  of processes. Perfectionism, it seems, does not pay!

## References

1. Ben-Or, M.: Private communication
2. Fischer, M., Rabin, M.O.: Concurrent search of a large data-structure. In preparation
3. Lehmann, D., Rabin, M.O.: On the advantages of free choice: A symmetric and fully distributed solution to the dining philosophers problem. Submitted for publication
4. Rabin, M.O.:  $N$ -process synchronization by  $4 \cdot \log_2 N$ -valued shared variables. Proceedings of the 21st IEEE Annual Symp. on Foundations of Computer Science (1980). To appear JCSS
5. Probabilistic algorithms: Algorithms and complexity, New Directions and Recent Trends (J.F. Traub Ed). Academic Press: New York (1976), pp. 21-39
6. Treat, A.: Experimental control of ear choice in the moth ear mite. XI. Internationaler Kongress für Entomologie. Wien (1960), pp. 619-621