THE CHOICE COORDINATION PROBLEM

by

Michael O. Rabin

Memorandum No. UCB/ERL M80/38

August 1980

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

THE CHOICE COORDINATION PROBLEM

by

Michael O. Rabin
Department of Mathematics
The Hebrew University of Jerusalem

and

Department of Computer Science
Harvard University

Abstract

In the course of a concurrent computation, processes $P1,\ldots,Pn$ must reach a common choice of one out of k alternatives $A_1,\ldots,A_k$. They do this by protocols using k shared variables, one for each alternative. If the range of the variables has m values then $\frac{1}{2}\sqrt[3]{n} \leqslant m$ is necessary, and $n + 2 \leqslant m$ 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.

This research was supported in part by NSF grants: MCS77-02474 at Washington University, Seattle, MCS80-12716 at University of California at Berkeley.

The Choice Coordination Problem

by

Michael O. Rabin

1. Introduction

Let $P1,\ldots,Pn$ 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 $A_1,\ldots,A_k$, 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 $A_1,\ldots,A_k$ are identically named by all processes, say by words $w_1,\ldots,w_k$, then there is an easy solution. Each Ps participating in the choice will traverse all the $A_1,\ldots,A_k$ in some order and choose the A_i such that $w_i = \min w_j$. If, however, each Ps has his own system of names for $A_1,\ldots,A_k$ then a different approach to choice coordination is required.

Assume that each A_i has an associated variable v_i which is shared by $P1,\ldots,Pn$. A process Ps arriving at A_i can <u>test and set</u> v_i, i.e. in one indivisible 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 $P1,\ldots,Pn$ such that for every order in which the processes are activated, eventually one and only one of $v_1,\ldots,v_n$ will satisfy $v_i = e$.

We assume that the ranges of $v_1,\ldots,v_n$ are $\Sigma = \{0,1,\ldots,m-2,e\}$ 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 $C_1,\ldots,C_k$ 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 $P1,\ldots,Pn$ 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 Σ_k (depending on k) will not suffice for the C.C.P. for any collection $P1,\ldots,Pn$, 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 $P1,\ldots,Pn$ there exists a schedule S (sequence of activation of the processes) so that the processes do not acheive 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 $n \leq m$ 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 Pi, 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 P1,...,Pn 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 P1,...,Pn 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.

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 $\Sigma = \{0,1,\ldots,m-2,e\}$, 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 Σ, is a mapping

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

such that

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

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 $(\ ,\ )$ of cells. At any given time, P is positioned either on the left or on the right cell and the pair contains letters (α,β), $\alpha \in \Sigma$, $\beta \in \Sigma$. 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 $w = \sigma_1\sigma_2 \ldots \sigma_k$ of symbols (hence $\sigma_k = \alpha$) and $P(w) = (\tau,X)$, then P will replace α by τ. 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 $S = i_1 i_2 \ldots$, $1 \leq i_j \leq n$, is an infinite sequence and $p: \{1,\ldots,n\} \to \{L,R\}$.

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, Λ denotes the empty word, ϕ 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 $P1,\ldots,Pn$ be processes using Σ, 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 (λ,ρ), $\lambda,\rho \in \Sigma$, a sequence $(w_1,\ldots,w_n)$ 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 Γ by the processes according to the schedule (S,p), $S = i_1 i_2 \ldots$, is the sequence $C_0,C_1,\ldots$, of configurations, where

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

is called the configuration after time t. The computation Γ is defined inductively as follows. Initialize

$\lambda_0 = \rho_0 = 0$, $w_j(0) = \Lambda$, $1 \leq j \leq n$,

$LF(0) = \{Pi \mid p(i) = L\}$, $RT(0) = \{Pi \mid p(i) = R\}$.

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

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

Furthermore, if $X = T$ then

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

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 contents 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 $P1,\ldots,Pn$ according to a schedule (S,p).

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

II For some t, $\lambda_t = \rho_t = e$. We shall say that Γ 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 Γ resulted in choice coordination.

Definition 5. Processes (protocols) P1,...,Pn 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 P1,...,Pn solving the C.C.P. and using $n+2$ letters $\Sigma = \{0,1,...,n,e\}$.

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

In short, Pi always "becomes" Pm 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 $w = x_1 x_2 \ldots x_\ell \in \Sigma^*$, $1 \leq \ell$, and $m = \min_{0<x_t} x_t$, then

$$Pi(0) = (i,T), \quad Pi(j) = (\min(i,j),T) \quad \text{for } 0 < j \leq n,$$

$$Pi(e) = Pi(we) = (e,E)$$

$$Pi(wj) = \begin{cases} (j,T) & 0 \leq j < m \\ (0,T) & m < j \leq n \\ (e,E) & j = m \end{cases}$$

To prove termination, let (S,p) be a schedule and recall that $S = i_1 i_2 \ldots$ is infinite. Assume by way of contradiction that the computation Γ does not terminate (i.e. that (e,x) or (x,e) never appear); 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 Pi appears in S and that $p(i) = L$. At time $t+1$ the content is (i,ρ). 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 Pj 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 Pj is active in S, such that at time t_1 the process is on the left (seeing i). Then at time t_3 process Pj 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 contents 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 $P\ell$ amd Pr 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 $P\ell$ "becomes" Pi. This contradicts the assumption that later $P\ell$ 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 processor 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 α into β (primed

to do $\alpha \to \beta$, for short) if

$$Pi \in LF \quad (Pi \in RT) \quad \text{and} \quad Pi\ (w_1\alpha) = (\beta, X).$$

If Γ is a computation of $P1,\ldots,Pn$, then Pi is primed on the left side to do $\alpha \to \beta$ <u>at time</u> 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 $\alpha \to \beta$ at time t means that if according to the schedule Pi will next be active at time $t_1 \geq t$, and at that time the contents will be (α,σ), then just after time t_1 the contents will be (β,σ). Note, however, that the contents at time t_1 could be (γ,σ), $\gamma \neq \alpha$.

Let (S,p) be a schedule, Γ 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 (Σ,E) on $\Sigma = \{0,1,\ldots,m-2,e\}$ as follows. For each $Pi \in G$ specify <u>one</u> pair α,β such that Pi is primed to do $\alpha \to \beta$, 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 α is <u>connected</u> to vertex β if $\alpha = \beta$ or there exists a path $\alpha = \alpha_1,\ldots,\alpha_k = \beta$ such that $\langle\alpha_j,\alpha_{j+1}\rangle \in E$, $1 \leq j < k$. If α is connected to β and β is connected to α we say that α and β are <u>strongly connected</u> 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 <u>strongly connected components</u> 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 $E = \phi$ (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 V of vertices. The graph $G(0) = (V,\phi)$ 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 $G(0), G(1), \ldots, G(M)$ are a sequence of the above type and if the set V of vertices has m elements, then $M \leqslant 2m-2$.

Also, if at every stage k in the construction of the sequence $G(0),\ldots$, 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 α, and if $\langle\alpha,\beta\rangle$ is added to obtain $G(k+1)$, then the strongly connected component $K' \ni \beta$ 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 $G(i+1)$, $0 \leqslant i < M$, reduces $T(i)$ or $S(i)$ by at least 1. Thus $2 \leqslant S(M) + T(M) \leqslant 2m - M$. Hence $M \leqslant 2m - 2$.

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

We shall need another combinatorial result.

Lemma 3. Let $H_1 = X_1X_2,\ldots,X_{2k}$, $X_i \in \{L,R\}$ 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

$$(4) \quad H_1 = X_1 \ldots X_{i-1}\, F(X_i)\, F(X_{k+1}) \ldots F(X_{2k})$$

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 $H = X_1 \ldots X_{2k}$. 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 $H' = X_1 \ldots X_{j-1}\, X_{j+2} \ldots X_{2k}$ 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 P1,...,Pn be processes on the alphabet $\Sigma = \{0,1,\ldots,m-2,e\}$. 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), $S = i_1i_2 \ldots i_{t-1}$, where $p: \{i_1,\ldots,i_{t-1}\} \to \{L,R\}$, so that after

time $t-1$ the cells have contents (α,α). Furthermore, there will exist symbols $\lambda_1 = \alpha,\ldots,\lambda_\ell = e$, $\rho_1 = \alpha,\ldots,\rho_k = e$ and processes $Pj_1,\ldots,Pj_{\ell-1}$, $Pm_1,\ldots,Pm_{k-1}$, such that at time t 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 be possible unless we shall encounter at some time $s < t-1$ a hitherto unused process P_i, $i \notin \{i_1,\ldots,i_s\}$, and a side $X \in \{L,R\}$ with the following behavior. The computation under the schedule (S',p') will be non-terminating (see the terminology following Definition 4), where $S' = i_1 i_2 \ldots i_s iii\ldots$, $p'(i) = X$, and $p'(j) = p(j)$ for $j \neq i$. In this case $P1,\ldots,Pn$ are not a solution for the C.C.P.

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

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 $P1,\ldots,Pn$ 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 (α_k,α_k).
2. The schedule (S_k,p_k) invokes just the processes $P1,\ldots,P4m^2k$. In particular, the domain of p_k is $\{1,\ldots,4m^2k\}$.
3. There exist pairwise disjoint graphs $GL_i(k) \subseteq LF(t_k)$ and $GR_i(k) \subseteq RT(t_k)$, $1 \leq i \leq 4m-2k$ such that
   a) All of these graphs on Σ have the same strongly connected components.

b) The symbol α_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 Σ 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 $\alpha \to \beta$.

At stage 0 we define $\alpha_0 = 0$, $(S_0,p_0) = (\Lambda,\phi)$, $GL_i(0) = GR_i(0) = \phi$, $1 \leq i \leq 4m$. 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,...,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 $\lambda_1 = \alpha, \lambda_2, \ldots, \lambda_\ell = \beta$ (if $\alpha_k = \beta$ then $\ell = 1$), and processes $Pj_1,\ldots,Pj_{\ell-1} \in GL_i(k)$ so that Pj_r, $1 \leq r \leq \ell-1$ is primed on the left to do $\lambda_r \to \lambda_{r+1}$. Thus the schedule $(S_k j_1 j_2 \ldots j_{\ell-1}, p_k)$ will lead to the contents (β,α_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 (β,γ). In particular this implies that if $e \in K$ then P1,...,Pn are not a solution, so that we may assume $e \notin K$.

Consider the processes $P4m^2k+1,\ldots,P4m^2k+4m^2$, which have not been used in (S_k,p_k). Since $k < 2m$ and $8m^3 \leq n$ we have $4m^2k + 4m^2 \leq n$. At time t_k+1 start $P4m^2k+1$ on side L and run it. In other words, construct the schedule (S,p) where $S = S_k(4m^2k+1)(4m^2k+1)\ldots$, and p extends p_k by $p(4m^2k+1) = L$. In the computation according to (S,p) the contents of the cells after times t_k, t_k+1, t_k+2,... will be (α_k,α_k), (λ_1,α_k), (λ_1,ρ_1),... . Two cases are possible. Either $\lambda_i,\rho_i \in K$ - the terminal component containing α_k, for $i = 1,2,\ldots$.

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 $P4m^2k+1$ is primed to do $\lambda_i \to \lambda_{i+1}$ where $\lambda_i \in K$, $\lambda_{i+1} \notin K$, or $\rho_i \to \rho_{i+1}$ 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 $P4m^2k+1$ will be primed on the left to do $\alpha_k \to \lambda_1$. And if $\lambda_1, \rho_1, \lambda_2 \in K$ but $\rho_2 \notin K$ then $S' = S_k(4m^2k+1)(4m^2k+1)(4m^2k+1)$; after time t_k+3, process $P4m^2k+1$ will be primed on the right to do $\rho_1 \to \rho_2$.

Next start $P4m^2k+2$ on side L and extend (S',p') to (S'',p'') in the above manner so that after the computation according to (S'',p''), process $P4m^2k+2$ is primed either on the left or on the right to do $\beta \to \gamma$ where $\beta \in K$, $\gamma \notin K$. Continuing in this manner, starting every $P4m^2k+i$, $1 \leq i \leq 4m^2$, on the L-side, we extend (S_k,p_k) to $(S^{(4m^2)},p^{(4m^2)})$ so that after the computation according to this schedule, every $P4m^2k+i$, $1 \leq i \leq 4m^2$, is primed to do $\beta(i) \to \gamma(i)$.

There are at most m possible values for $\gamma(i)$, so there must exist $4m$ indexes $4m^2k+1 \leq j_1 < j_2 < \ldots < j_{4m} \leq 4m^2(k+1)$, for which $\gamma(j_1) = \gamma(j_2) = \ldots = \gamma(j_{4m}) = \gamma$. Thus at the end of our schedule, Pj_ℓ, $1 \leq \ell \leq 4m$, is primed on side $X_\ell \in \{L,R\}$ to do $\beta(j_\ell) \to \gamma$ where $\beta(j_\ell) \in K$ and $\gamma \notin K$.

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

According to Lemma 3 there exists an $1 \leq i < 4m$ such that the sequence $X_1 \ldots X_{i-1}\ F(X_i) \ldots F(X_{4m})$, 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 $(S^{(4m^2)},p^{(4m^2)})$ after which Pj_{i-1} is primed (at time $t+1$) on side X_{i-1} to do $\beta(j_{i-1}) \to \gamma$. Let the contents of the cells after time t be (λ,ρ) 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 $S^{(4m^2)}$ it has the form

$$S^{(4m^2)} = S_k\ (4m^2k+1)\ \ldots\ j_{i-1}\ (j_{i-1}+1)\ \ldots\ 4m^2(k+1),$$

where the displayed occurence of j_{i-1} is at time t in the computation. Denote by $\overline{S}$ the initial segment of $S^{(4m^2)}$ up to and including the displayed j_{i-1}; the length of $\overline{S}$ is t. Since $\lambda,\rho \in K$, we can activate, subsequently to $\overline{S}$, processes in GL_{2m-2k} on the left-side and in GR_{2m-2k} (on the right-side), to transform the contents of the cells from (λ,ρ) to (ρ,λ). 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 $\overline{S}$ to $\overline{S}\ell_1\ldots\ell_s$. Define now

$$S' = \overline{S}\ell_1\ \ldots\ell_s\ (j_{i-1}+1)\ \ldots\ 4m^2(k+1)$$

Here $P\ell_1,\ldots,P\ell_s$ are the processes used to effect the flip-over of the contents of the cells; we have $\ell_j \leqslant 4m^2k$, $1 \leqslant j \leqslant s$. Note that in the computation according to S', after time $t+s$ the contents of the cells is (ρ,λ). Also, $j_1,\ldots,j_{4m}$ appear in S' after ℓ_s.

Change $p^{(4m^2)}$ into p_k by setting $p_k(h) = p^{(4m^2)}(h)$, $1 \leqslant h \leqslant j_{i-1}$, and $p_k(h) = R$ for $j_{i-1}+1 \leqslant h \leqslant 4m^2(k+1)$. Recall that $p^{(4m^2)}(h)$ was L, $j_{i-1}+1 \leqslant h \leqslant 4m^2(k+1)$. Let us examine the effect of

the computation according to the schedule (S',p_k). After time $t+s$, i.e. at the end of the $\overline{S}\ell_1\ldots\ell_s$ segment, the contents of the cells is (ρ,λ). At times $t+s+1$, $t+s+2,\ldots,$ the computation will run exactly like the computation according to $(S^{4m^2)},p^{(4m^2)})$ at times $t+1$, $t+2$, ..., except that left and right are interchanged in the sense that everything that occured in the computation according to $(S^{(4m^2)},p^{(4m^2)})$ 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 $Pj_1,\ldots,Pj_{4m}$, were first activated in $(S^{(4m^2)},p^{(4m^2)})$ after time t, and in (S',p_k) after time $t+s$, it follows that at the end of the computation according to (S',p_k) these processes will be primed on sides $F(X_1)\ldots F(X_{4m})$ to do $\beta(j_1)\rightarrow\gamma,\ldots,\beta(j_{4m})\rightarrow\gamma$. Thus at the end of this computation exactly $2m$ of the $Pj_1,\ldots,Pj_{4m}$, will be primed on the left and exactly $2m$ of these processes will be primed on the right.

Also, at the end of (S',p_k) the contents of the cells will be (σ_1,σ_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 $GL_{2m-2k-1}(k)$ on the L-side and processes in $GR_{2m-2k-1}(k)$ on the R-side, to extend S' to S'' so that at the end of the computation according to (S'',p_k) 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 (γ,γ). 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 $GL_i(k)$, $1 \leq i \leq 2m - 2k - 2$,

(a different) one of the $Pj_3,\ldots,Pj_{4m}$ which is primed on the L-side and call the resulting graph $GL_i(k+1)$; proceed similarly with the $GR_i(k)$. By 3.a), all the $GL_i(k)$, $GR_i(k)$, had the same strongly connected components. Every Pj_ℓ adds an edge $<\beta(j_\ell),\alpha_{k+1}>$ where $\beta(j_\ell) \in K$ - a common strongly connected component. It follows that all $GL_i(k+1)$, $GR_i(k+1)$, $1 \leq i \leq 2m - 2(k+1)$, 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 α_{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 conclude the proof of Theorem 4, observe that the sequence of graphs $GL_1(0)$, $GL_1(1)$,... 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!

By being more stingy with the use of new processes in the construction of the schedules (S_k,p_k), we could reduce the bound in Theorem 4 to $2m^3 \leq n$. 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 $\alpha \to \beta$. 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 $8m^3 \leqslant n$ then there exists a schedule (S,p) leading to a non-terminating computation by $P1,\ldots,Pn$, where for some $1 \leqslant i \leqslant n$, S has the form

$$(5) \qquad S = i_1 i_2 \ldots i_s iii\ldots ,$$

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

The special form (5) of the schedule (S,p) leading to non-termination, suggests the possibility that the assumption $8m^3 \leqslant n$ 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 Σ 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 suggest 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 $\Sigma = \{0,1,2,\ldots,m,e\}$ where m is even. The numbers $1,\ldots,m$ are viewed as grouped in pairs $\{1,2\}$, $\{3,4\},\ldots,\{m-1,m\}$. Each of the processes $P1,\ldots,Pn$, 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 α and it currently sees β then if $\alpha < \beta$ it transfers side, if $\beta < \alpha$ process P marks the cell, i.e. changes the β into e, and if $\alpha = \beta < m-1$ then P replaces β by random$\{2\lceil\frac{\beta}{2}\rceil+1,\ 2\lceil\frac{\beta}{2}\rceil+2\}$ i.e., by random$\{i,i+1\}$ where $\{i,i+1\}$ is the next pair after the pair containing β. Thus if P last saw 13 on the L-side and now sees 13 on the R-side (this does <u>not</u> mean that the current contents of the cells is (13,13)) then it will write random $\{15,16\}$ on the R-side. Formally, for $0 \leqslant \alpha, \beta \leqslant m$, $w \in \Sigma^*$,

$$P(we) = (e,E),\quad P(0) = (\text{random}\{1,2\},T),\quad P(\alpha) = (\alpha,T) \text{ for } 1 \leqslant \alpha \leqslant m, \tag{6}$$

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

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 Γ by randomizing processes P1,...,Pn according to a schedule, is exactly as in Definitions 2 - 4. Any particular computation Γ 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 π. 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 Γ according to π. Note that Pr_π it not conditional probability, since π is fixed. It is easy to see that an event such as Γ non-terminating (see I following Definition 4), has a well defined probability.

Theorem 5. If each of the processes P1,...,Pn on the alphabet $\Sigma = \{0,1,...,m,e\}$ is defined by (6), then for every schedule (S,p) a computation Γ 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 Pi is active in S $2k \leq m$ or more times, then the probability that by the $2k^{th}$ activation Pi will find or write e is greater than $1-1/2^k$.

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

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

be the sequence of contents of the pair of cells in $C_0,C_1,...$. Since a

change of a contents of a cell involves randomly drawing from the next pair $i,i+1$, we have $\lambda_1 \leq \lambda_2 \ldots$ and $\rho_1 \leq \rho_2 \ldots$. Assume by way of contradiction that $\lambda_t = \rho_t = e$ and that Pi and Pj 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 Pi's history after time u be $a\rho_v$ where $a \in \Sigma^*$ and $v \leq u$, and Pj's history after time s be $b\lambda_w$ $b \in \Sigma^*$, $w \leq s$. Since at time $u+1$ process Pi changes the L-side into e we must have $\rho_v > \lambda_u$, and similarly $\lambda_w > \rho_s$.

Assume now $s < u$. Process Pi must have last visited the R-side before time $s+1$ because at time $s+1$ process Pj was there, and after time $s+1$ the contents was (λ_{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 Γ will not terminate is if for some t the contents (λ_t, ρ_t) is $(m-1,m-1)$ or (m,m). For this to occur there must be times $t_1 < t_2 < \ldots < t_{m/2} = t$ such that $\lambda_{t_1} = \rho_{t_1} \in \{1,2\}$, $\lambda_{t_2} = \rho_{t_2} \in \{3,4\}, \ldots$. 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 Γ 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 $1 \leqslant i \leqslant 1000$ into groups of 100, $\{1,\ldots,100\}$, $\{101,\ldots,200\},\ldots$. The random draw in case of equality will be from the next group of 100, so that we use $\text{random}\{i+1,\ldots,i+100\}$ for $i = 0, 100,\ldots, 900$. 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 $P1,\ldots,Pn$ be randomizing processes on an m-letter alphabet such that for every schedule $\pi = (S,p)$: (1) No computation Γ according to π leads to the contradiction (e.e). (2) The probability for a computation Γ to be non-terminating is 0. Then we must have $1/2 \cdot \sqrt[3]{n} \leqslant m$.

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 $A_1,\ldots,A_k$ of a text are being generated,

Processes P1,...,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 Σ 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 Section 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 P1,...,Pn 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},...,{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 $10^{18}.2^{-127} \leqslant 2^{-87}$. 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!

BIBLIOGRAPHY

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, October 1980.

5. ---, Probabilistic algorithms, Algorithms and Complexity, New Directions and Recent Trends, J.F. Traub Ed., Academic Press, N.Y. (1976), pp. 21-39.