Theoretical Computer Science 24 (1983) 1-20
North-Holland Publishing Company

AN ITERATIVE AND STARVATION-FREE SOLUTION FOR A GENERAL CLASS OF DISTRIBUTED CONTROL PROBLEMS BASED ON INTERACTION PRIMITIVES

Horst WEDDE

Gesellschaft für Mathematik und Datenverarbeitung mbH Bonn, D-5205 St. Augustin 1, Fed. Rep. Germany

Communicated by M. Nivat
Received December 1981
Revised July 1982

Abstract. For an independent and graphical representation of the constraints in distributed system parts the formalism of *Loosely Coupled Systems* is recalled. The events in these structures are formally derived from symmetrical bilateral restrictions. How the interaction between neighbouring parts influences processes in these parts is then adequately described by a symmetrical transitional structure (*slack of behaviour*) in each part. In order also to represent asymmetrical types of influence *local excitement relations* are introduced by means of which we can determine directions of flow and can force processes to leave a given local state. The formally extended system structures are called *Interaction Systems* (IS). A solution of the Dining Philosophers' Problem given by Dijkstra in [5] is briefly discussed. In order to demonstrate the flexibility and representational power of our graphical tools we then derive a starvation-free solution for that problem in a *stepwise* procedure. (We do not assume a global finite-delay property.) We reconsider a general problem for distributed processes which access shared resources [2, 14]. The solution scheme for the special case of Dijkstra's problem applies at once to the general case. The starvation-freeness of the solution is proved.

Introduction

The theory of *Loosely Coupled Systems* (LCS) (see [10, 11, 13]) is solely based on the specification of the interaction between (distributed) system parts. As the events in LSCs are completely derived from the specified interaction structure (constraints) given by *coupling relations* we can model an independent aspect of behaviour in these terms. The internal behaviour structure of the system parts is regarded to be induced by interaction with environmental parts. In [10] a formal concept was defined by means of which one can express these relationships conveniently. Altogether we can so model and analyze a large number of synchronization mechanisms in distributed systems [10].

The interaction primitives in LCSs (*coupling relations*) specify mutual exclusion between states of subsystems. The internal behaviour structure which is induced by coupling relations is symmetrical, being mainly the slack of behaviour which is left to the parts under the specified constraints.

In order also to deal with asymmetrical influences we introduce additional (asymmetrical) interaction primitives in Section 1, so-called *excitement relations*. The extended formal system structures are called *Interaction Systems*. They model *local* influences between parts in such a way that there is a certain equilibrium between the *rights* of the influencing part and its *responsibilities* for the activities in the influenced part. Although such a principle is often enough not observed (e.g. an interrupt which has not yet been processed can be cancelled by an overlying interrupt: So the influencing state may disappear although the influenced action has not yet occurred) we found it very helpful to define a clean formal concept of induced behavioral structures, including *directions of flow* and *local forces*. Also, for the 'rendezvous' between two processes in ADA (see [7]) or in CSP (see [6]) balancing principles similar to ours are observed. (Modeling the CSP process synchronization by use of Interaction Systems has been worked out in a separate paper ([9]).

In [5] Dijkstra offered a new solution for the Dining Philosophers' Problem in terms of *atomic actions*. Because of the implicit use of mutual exclusion between activities of different processes the program structures do not reflect the (cyclic) dependency structure with respect to the shared resources. Thus the solution – which is deadlock-free – can immediately be transferred to a much more general problem where the mutual dependency structure is arbitrary. In Section 2 we give a short argument which makes it very doubtful whether or not starvation-freeness can be achieved for Dijkstra's program without assuming a global finite-delay property. The argument reveals at the same time that one cannot model a starvation-free solution under distributed control solely on the basis of mutual exclusion (coupling) relations. In order to demonstrate the flexibility of our extended formal tools we then build a solution for that problem by stepwise constructing a suitable Interaction System which is starvation-free. We do not assume a global finite-delay property. In Section 3 we extend this solution to the general case where the mutual dependency structure (with respect to accessing common resources) is an arbitrary graph. We prove that this solution is starvation-free. In Section 4 we discuss our results and compare them with other existing solutions [1, 2].

1. Interaction systems

We shall briefly recall some basic notions and properties of Loosely Coupled Systems. For more details see [10] and [13].

In order to formalize the interdependence between the components of a real system we do not assume anything about the system parts except that they are in exactly one state or section of activity at any time. We call these states *phases*. Let B be the (finite) set of parts, P the set of all phases. Our assumption then has the following form:

$$\forall_{b \in B} \; b \subseteq P; \qquad \bigcup_{b \in B} b = P; \qquad \forall_{\substack{b_1, b_2 \in B \\ b_1 \neq b_2}} b_1 \cap b_2 = 0. \tag{1.1}$$

The interdependence between the parts b_1 and b_2 is at first reduced to the mutual exclusion of some of their states (phases), the corresponding relation is denoted by $K\langle b_1 b_2\rangle$ and is called *coupling relation* between b_1 and b_2. The relation K which contains all pairs of mutually exclusive phases is the union of the $K\langle b_1 b_2\rangle$ and of all relations $K\langle bb\rangle$, $b \in B$, where $(p_1, p_2) \in K\langle bb\rangle$ iff $p_1, p_2 \in b$ and $p_1 \neq p_2$. K is symmetrical.

Global system situations are called *cases*. A case is a subset c of phases such that

$$\text{(a)} \quad \forall_{b \in B} |c \cap b| = 1; \qquad \text{(b)} \quad \forall_{p_1, p_2 \in c} (p_1, p_2) \notin K. \tag{1.2}$$

The set of all cases is denoted by C. *Elementary events* in a Loosely Coupled System are phase transitions in a single part which lead from one case to another one. They are therefore represented by a pair (c_1, c_2), $c_1, c_2 \in C$, such that $|c_1 - c_2| = |c_2 - c_1| = 1$. Events are defined by the following postulate:

Postulate 1.1. Every event in an LCS can be decomposed into a sequence of elementary ones.

An LCS is then denoted by a quadruple (P, B, C, K). A simple example is found in Fig. 1.

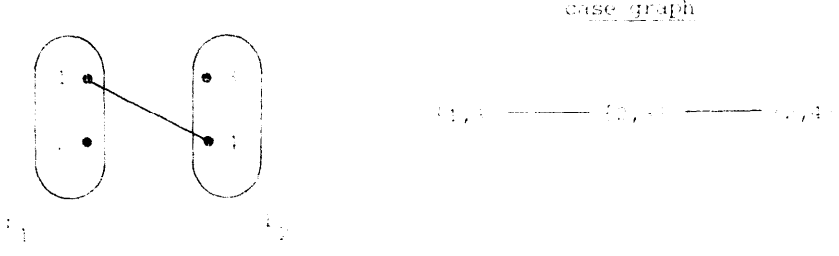


Fig. 1.

The two parts have two phases each. The coupling relation is represented by the undirected edge between 1 and 4. The event structure is found in the *case graph* which has the cases as nodes and the elementary events as (undirected!) edges. We mention briefly that two phase transitions $p_1 \to q_1$ and $p_2 \to q_2$ are called *concurrent in an (initial) case* c iff each of them may occur in c and $(q_1, q_2) \notin K$. By that, concurrency in LCSs is a basic and local property. It means that two events may occur in arbitrary order and even simultaneously (compare [10]). A *slack phase with respect to a subsystem* S is a phase which is not coupled to any phase in a part belonging to S. Starting with the case {1, 3} in Fig. 2 we can see that no event can occur in this situation because there is no elementary event activated. By adding the slack phase 5 (with respect to the subsystem which contains b_2 only) as shown in Fig. 3 we can now pass along the following sequence: {1, 3}, {5, 3}, {5, 4}, {2, 4}. Thus the slack of b_1 with respect to b_2 was enlarged.

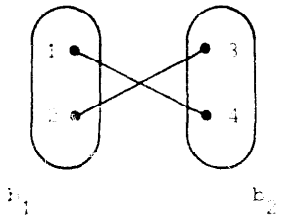


Fig. 2.

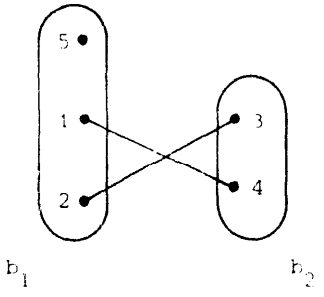


Fig. 3.

In [10] a formal concept had been developed in which the influence upon a part coming from its environment can be modelled and analyzed. If we want to understand the influence upon b_1 in Fig. 4 which comes from the coupling to b_2 we learn from the case graph of this LCS that e.g. from phase 1 one cannot immediately reach 3 (similarly for phases 2, 3, 4). Altogether one can step around in b_1 as indicated by the undirected edges in Fig. 5 if we assume that certain intermediate steps in b_2 can occasionally occur between two transitions in b_1. But this is no problem since no restriction has been specified which prevents b_2 from acting in the indicated manner.

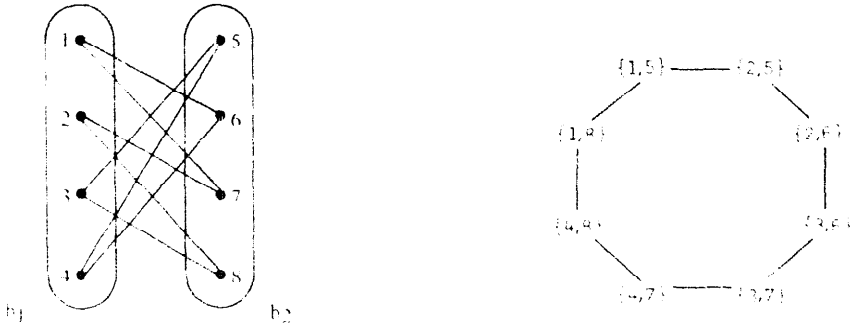


Fig. 4.

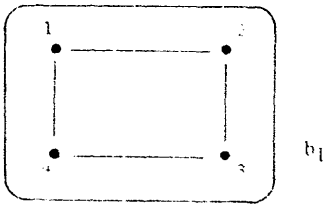


Fig. 5.

All relational structures, i.e. the coupling relations and the induced relations in the parts (see Fig. 5) are symmetrical. In order to refine our language we introduce new specification elements:

Definition 1.1. Let $E \subseteq (B \times B) - \mathrm{id}(B)$ with $K \cap (E \cup E^{-1}) = \emptyset$. We call E an *excitement relation* iff for $p \in b_1$, $q \in b_2$ where $b_1, b_2 \in B$ and $b_1 \neq b_2$, and for $c \in C$ the pair $(p, q) \in E$ means:

(1) In c, b_2 is forced by b_1 to leave q;
(2) As long as b_2 has not left q, b_1 cannot leave p.

With the notations above q is an *excited* or *unstable* phase, p is an *exciting* phase. A case c is called *unstable* iff it contains an excited (and thus an exciting) phase. Otherwise c is called *stable*. As a typical example one may consider a job which sends a request for resource allocation: Under normal circumstances the requested resources will be allocated after some while but as long as this has not been done the job has to wait. As another example, if an ADA task reaches an accept statement it has to wait until another task calls the corresponding entry in the task head. Then a rendezvous occurs after which the calling task is free after execution of the accept statement. The interaction between tasks in ADA is completely based on this rendezvous technique (see [7]).

Given a case $c \in C$ and $p \in b$ we call p *free with respect to* c iff $(p, p') \notin K$ for $b' \neq b$ and $\{p'\} := c \cap b'$. (A free phase with respect to c is occasionally reachable from c by a single phase transition.)

Assuming distributed control or partial autonomy of the system parts we cannot expect that a *local* influence from b_1 on b_2 (like an excitation) will have an observable effect (because b_2 might successfully resist to leave the excited phase). In order to define the *global* effect of the local excitations we formulate some behavioral rules which we regard as a kind of *local stability axioms*:

Axiom 1.1. If $p \in b$ is excited in c and if phases in b are free with respect to c then b will go after some while into one of these free phases unless they are excluded by phase transitions in parts $b' \neq b$.

Axiom 1.2. If $p \in b$ is excited in c then b will go to another excited phase only if no stable phase is free with respect to c.

Axiom 1.3 (Induced forces). If b is forced in c to leave an excited phase p and if no other phase in b is free w.r.t. c then b forces the parts $b' \neq b$ to leave their corresponding phase p' in $c \cap b'$ in case that p' excludes a transition from p.

In order to keep track of the partial autonomy of the parts we introduce an explicit concept of (*local*) *decisions*:

D1. A part b in a phase p may *decide* to go to an arbitrary phase $q \in b$ in the sense that b would go to q without a decision only in case b was forced to leave p and q was a free phase. This decision may be cancelled before b goes to q, by one of the following three reasons:

(a) b is forced to leave p, q is not free while $q' \neq q$ is free;
(b) b is forced to leave p, q is free but not stable while $q' \neq q$ is stable and free;
(c) q is free but b cannot leave p because it excites $q' \in b'$.

In case of (a) and (b), b would have to cancel its decision and react to the force.

D2. When the decision to go to q has been cancelled b may take a new decision unless b would have to react to a force.

D3. There may be parts b which are not able to make decisions to go to a phase q. They are called *inert*.

(Typical examples for inert parts would be purely functional units (hardware components). Let BI *be the subset of inert parts in* B. We shall mainly use them in order to construct formal control mechanisms.)

D4. If a non-inert part b has decided to go to a phase q, b cannot go to $q' \neq q$ unless the decision to go to q has been cancelled before. After a decision to go to q, b would go after some while unless the decision has been cancelled.

In order to trace the local decisions in a formal behaviour concept one could introduce—for each $b \in B$—mappings of the form

$$m\langle b\rangle : b \to b \cup \{0, 1\}$$

such that

$$m\langle b\rangle(p) \neq p \text{ for } p \in b \quad \text{and} \quad m\langle b\rangle(p) \neq 0 \Rightarrow m\langle b\rangle(p') = 0 \text{ for } p' \neq p.$$

$m\langle b\rangle(p) = 0$ would indicate that b is not in p, $m\langle b\rangle(p) = q$ would indicate that b has decided to go to q; $m\langle b\rangle(p) = 1$ would mean that b is in p but has not (yet) decided to leave p. (Such a mapping could be called a *marking of* b.)

However, the (induced) process structures in the parts which are regarded in this paper are simple enough for neglecting such formalistic details. Instead, we shall be more concerned here with explaining and exemplifying the ideas which give rise to such a formal framework.

Axiom 1.4. If a part b which is not inert decides to go into a phase p which is currently excluded by a phase p' in a part b' then b forces b' to leave p'.

Axiom 1.5. If a force is induced such that b' leaves $p' \in c$ (as described in Axiom 1.3 or 1.4) then b' will behave as if p' were an excited phase.

A 6-tuple (P, B, BI, C, K, E) where the components are defined as above is called an *Interaction System*.

We want to give some small comments and examples regarding the meaning of the axioms.

If $E = \emptyset$ there would be no explicit notion for forces (and therefore no need to regard the aspect of inertia). So it is convenient then to assume $BI = \emptyset$, and we end up with the special case of Loosely Coupled Systems.

In a business or a public administration it is an essential difference for an employee whether the execution of a specific task is left to him (delegation of decision) or whether it is ordered by his boss. A system model in which one wants to discuss

delegation of decision or of responsibility as a *basic* aspect should consequently have a basic and explicit notion for *local* forces (and for their absence). (In our case this is the excitement relation.) So our formal framework is not only related to a mainly physical context of (local) forces but even more to fundamental behavioral aspects of socio-technical systems.

In a mechanical system the parts tend to enter a stable equilibrium state (regarding the potential energy). This is a partial motivation for Axiom 1.2. Axioms 1.3, 1.4 and 1.5 specify how forces are propagated in case that a stable situation cannot be reached at once in an excited part. Due to Axiom 1.3 forces might be propagated over coupled parts. The effect of such a propagation wave might even induce a force in the initially exciting parts—which is often a highly undesirable phenomenon in real systems. One main goal in our theory is therefore to find and investigate formal methods by which such phenomena could be detected or avoided in a formal specification.

An idea behind Axioms 1.2 and 1.4 is that a part b which is not inert will *follow his own decisions* (and in this way perform 'spontaneous' actions for an outside observer). This slack of decision is limited only in cases where b is forced to leave a phase: Even if b had decided to go to a follower phase q the decision would have to be cancelled in case q is not free or 'appropriate' with respect to Axiom 1.2 (compare D1(a) and D1(b) and Axioms 1.1 and 1.2). With this context in mind it is not very difficult to see that there is no contradiction between the axioms.

Under the assumptions of Axiom 1.1 some other events in parts $b' \neq b$ may occur before p is left. (These may even prevent b from leaving p!). Due to Definition 1.1(2) this has no influence on the excitation of p. Axiom 1.1 may be regarded as a *weak and local form of a finite-delay property* because after a finite (local) time b would always have to enter one of the free phases (even if b had previously decided to enter a phase which currently is not free). Axiom 1.1 is weak in the sense that b has only a very weak influence on the other parts (among which some might exclude follower phases in b).

The example in Fig. 6 is the standard construction for LCSs by which the transition $p \to q$ is absolutely excluded (as well as $q \to p$). Replacing the coupling edge between q and 2 by the excitement arrow $(2, q)$ we see (Fig. 7) that $p \to q$ is possible now (if we start from the stable case $\{p, 2\}$) but q has to be left after some while (and will do so due to Axiom 1.1).

If we reverse the arrow in Fig. 7 and define b' to be inert we come to the Interaction System in Fig. 8 (by underscoring b' we indicate that b' is inert) in which $p \to q$ may occur starting from $\{p, 2\}$ (after b has decided to go to q). Afterwards q excites 2, and after some while b' will go to 1 (Axiom 1.1) and remain there (because of its inertia); b can leave q but $q \to p$ cannot occur. Thus b can go from p to q—this is not enforced—but it cannot directly go back to p. In more detail: If b being in q would decide to go back to p then b' would be forced to go to 2 again (Axiom 1.1), and hence b could not leave q; b could then cancel the former decision (D1(c)) and take a different one in order to leave q.

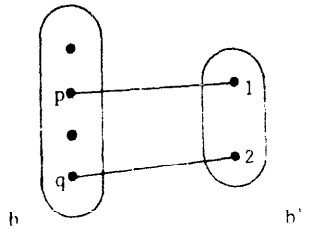


Fig. 6.

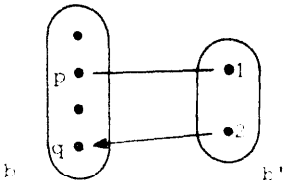


Fig. 7.

If we start in {1, 3, 6} in Fig. 9 where b_2 is inert then b_2 is forced to go to 4; 4 is not free with respect to {1, 3, 6} as 4 is coupled to 6. So b_3 is forced to go to 5 (Axiom 1.3). This will happen after some while (Axioms 1.1 and 1.5).

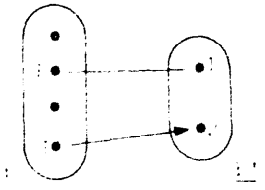


Fig. 8.

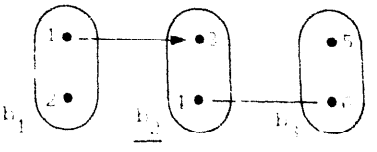


Fig. 9.

Summarizing up the discussion along Fig. 7, Fig. 8 and Fig. 9 we see that in Interaction Systems a phase transition *may* occur only if it is compatible with the mutual exclusion structure and if it does not violate Definition 1.1(2). A transition may be enforced or not, and the reverse transition may be excluded (so we can induce directions of activity in the parts).

Notation 1.1. The case graph of an Interaction System has the cases as nodes and an arrow from one case to another one iff a phase transition *may* occur which transforms the first into the second case. The corresponding arrow will be crossed by a little bar iff by the represented transition an excited phase p is left, i.e. where $(q, p) \in E$ for some q in the case to be transformed by the discussed transition. An arrow will be dotted iff it describes a phase transition in an inert part. (Note that such a transition may occur only if it is enforced by an excitement arrow or an induced force (through coupling edges).)

Different from other formal systems theories we have an explicit notion of (local) forces in our formal stuctures. The case graph as determined in Notation 1.1 traces the possible changes in a system without describing the effect of forces and their propagation. In addition to this aspect of behaviour the Axioms 1.1 to 1.5 together with D1, D2, D3, D4 help us to discuss the new dynamic aspect of forces and so to predict which changes eventually *will* occur in Interaction Systems. (In a formal concept of behaviour including markings of parts as mentioned above the cases with phases $p \in b$ would be replaced by values $n(b)(p)$ labelled with p, thus representing a 'decision state' of the Interaction System.) In order to display this

to some extent we shall briefly discuss two extensions of the LCS in Fig. 4 which will also be of special interest in Section 2. They are to be found in Fig. 10 and Fig. 11, respectively, together with their case graphs. If we want to describe the influence which b_2, by its connection to b_1 in Fig. 10, imposes on the behaviour of b_1, we easily derive, on the basis of our axioms, the internal transitional structure in Fig. 12 from the case graph in Fig. 10. (The meaning of a crossed arrow in Fig. 12 is that the corresponding transition is enforced by an environmental part.) Looking at the case graph in Fig. 11 we see that b_1 is not excited to execute $1 \rightarrow 2$.

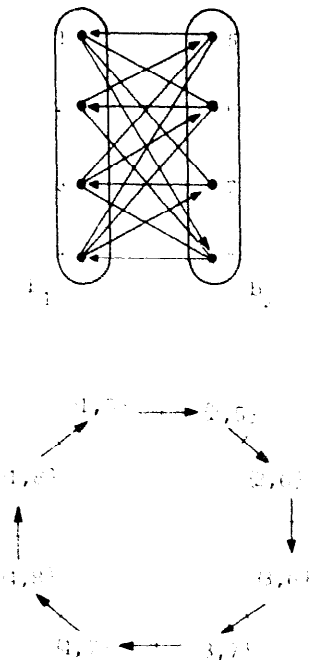


Fig. 10.

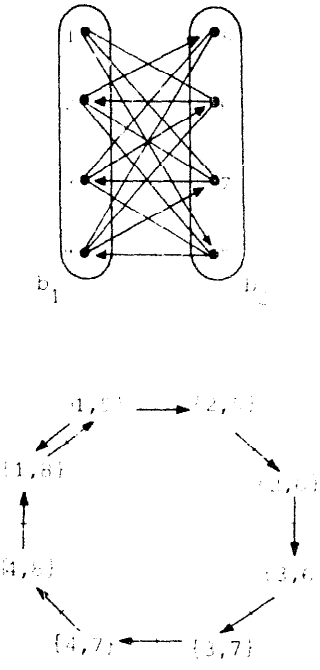


Fig. 11.

Also $2 \rightarrow 1$ cannot occur immediately after $1 \rightarrow 2$. Also, b_2 may decide in 5 to enter 8. After reaching 8, b would be forced to go back to 1. However, if we regard b_2 to be inert then $5 \rightarrow 8$ will never occur in this system (see the corresponding case graph in Fig. 13). Consequently we would in this new Interaction System end up with the pattern of behaviour which is found in Fig. 14.

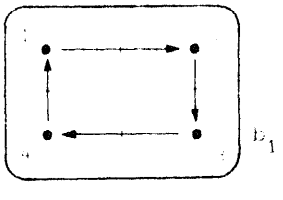


Fig. 12.

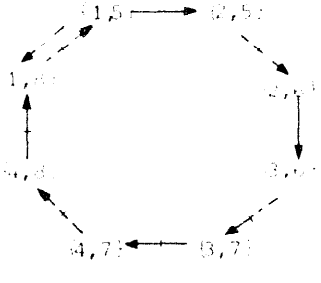


Fig. 13.

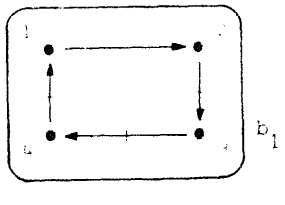


Fig. 14.

2. The dining philosophers

In a sequence of famous problems [3, 4] the Dining Philosophers' problem was to be solved *without a central control mechanism, without any assumption on relative speeds of processes or on geographical or timely distances.* In [5] Dijkstra describes a solution of this problem by a little program related to the graphical structure in Fig. 15. Here the nodes correspond to philosophers, and two nodes are connected

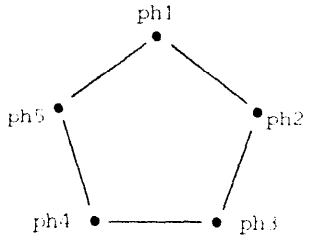


Fig. 15.

iff the corresponding philosophers are neighbours. The programs are cycles of the following form:

do THINK;
A: ⟨direct outgoing arrows towards all non-thinking neighbours⟩;
B: ⟨**await** no outgoing arrows **then** EAT⟩;
C: ⟨remove all incoming arrow heads of incident edges⟩;
od

These programs are very short and elegant because of the use of the *atomic actions* A, B, and C. In [5] it is proved that the solution is deadlock-free. Considering the possibility of individual starvation let us assume that initially all philosophers are thinking. After some while $ph1$, $ph2$, $ph3$ come close to the end of their thinking section while $ph4$ and $ph5$ are assumed to remain thinking for the rest of our discussion.

Let $ph2$ be very fast compared with his neighbours $ph1$ and $ph3$. So he will become hungry very soon and will enter his section A while $ph1$ and $ph3$ are still thinking. As A is an atomic action we are sure that during its execution neither $ph1$ nor $ph3$ will enter their section A: Otherwise $ph2$ would have to interrupt the execution of his section A and direct outgoing arrows either to $ph1$ or to $ph3$. Thus $ph2$ excludes $ph1$ and $ph3$ from entering their section A. After $ph2$ has executed A, $ph1$ and $ph3$ might leave their thinking section. But it is conceivable that $ph2$ is fast enough to successively pass through its sections B and C and even through THINK again before $ph1$ and $ph3$ have left their THINK section. So our argumentation starts to be repeatable, and hence $ph1$ and $ph3$ would starve under these special assumptions on the relative speeds. Another argumentation could easily be based on the assumption of a coalition of $ph1$ and $ph3$ against $ph2$ which

would lead to a starvation of $ph2$. In both cases the mutual exclusion of the sections A makes it impossible to avoid starvation under distributed control, without further constraints on the interaction of the philosophers. By use of our new interaction primitives we shall stepwise construct an Interaction System which meets the problem requirements.

The first step is found in Fig. 16. Here the relevant sections of activities (phases) of the philosophers are chosen to be thinking (t) and eating (e) between which they may alternate. This system is clearly deadlock-free. But again one easily finds

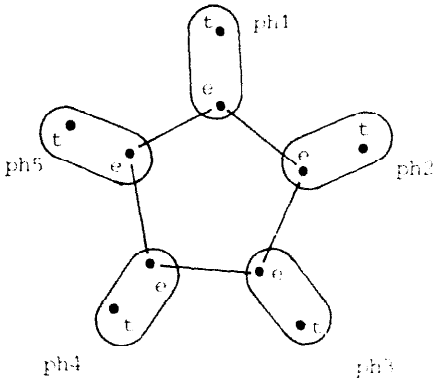


Fig. 16.

out that $ph2$ will starve once $ph1$ and $ph3$ form a coalition against him. In order to guarantee that a philosopher who has indicated his interest to eat cannot be prevented from doing so after a finite time we introduce phases rgi and ci for each phi, plus an internal (induced) stucture as it is shown in Fig. 17 (in direct analogy

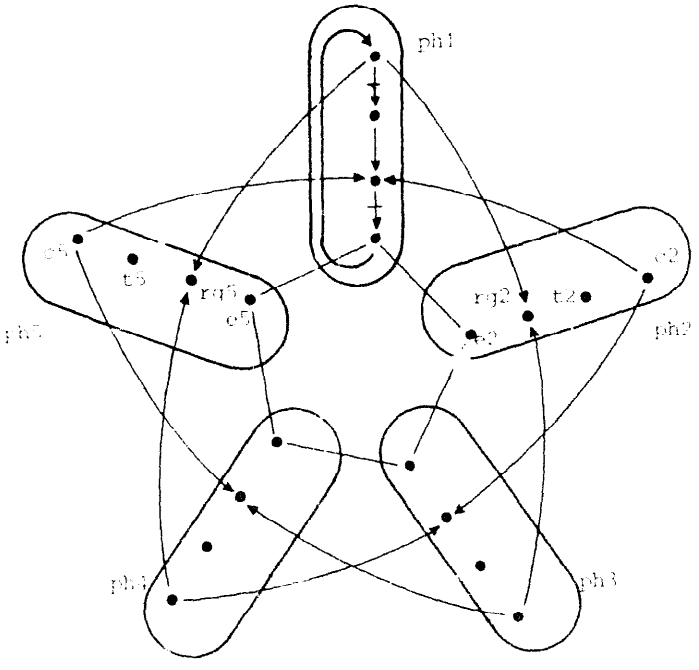


Fig. 17.

to Dijkstra's program structure). *rgi* is called the *registration phase*, *ci* is called the *clearing phase* for *phi*. The internal structure in *phi* corresponds to that in Fig. 14. As explained there it could be realized by connecting an additional inert part to *phi* as shown in Fig. 11. (Note that *phi* is never forced to leave *ti*!)

Whenever *phi* has reached its phase *ci* in Fig. 17 the effect of the excitement arrows going out of *ci* is that *phi* has to wait until the neighbour parts have occasionally left their phases $rg(i-1)$ and $rg(i+1)$, respectively ($i-1$ and $i+1$ are taken modulo 5). So a hungry philosopher *phi* can (in *rgi*) indicate his interest to eat at any time (!), and a coalition against him could not work in the way as described above.

Starting in the case $\{t1, t2, \ldots, t5\}$ we assume the following development: $ph3$, $ph4$, $ph5$ remain in their initial phase while $ph1$ be much faster than $ph2$ on the way to the eating phase. After some while $ph2$ will be in $rg2$. When $ph1$ has reached $c1$ under these circumstances it will be stopped. Let us assume that $ph1$ becomes very slow now (after so many efforts) whereas the process in $ph2$ starts to rush through its cycle. Then it is conceivable that $ph2$ has reached $rg2$ again before $ph1$ has left $c1$. Consequently $ph1$ could starve now. In order to escape this danger we introduce excitement arrows from the eating to the neighbour clearing phases as shown in Fig. 18.

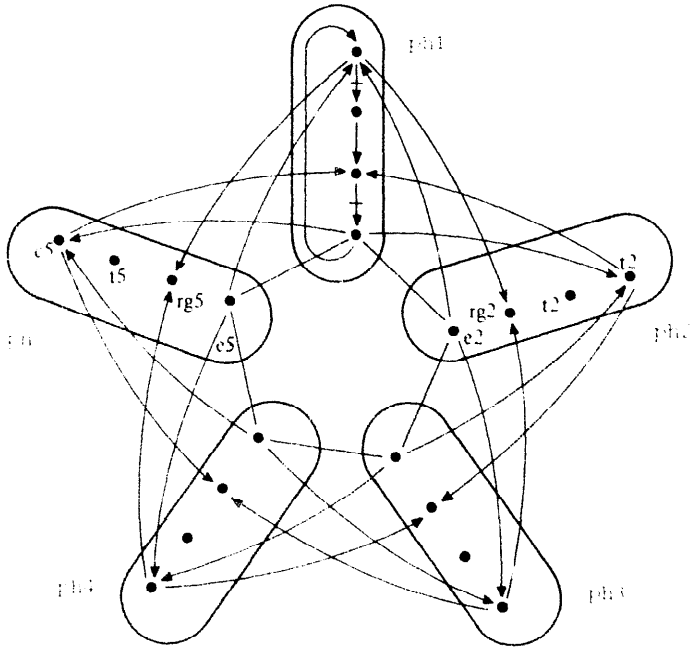


Fig. 18.

Continuing the discussion above we see that $ph2$, after starting from $rg2$, cannot leave the next phase $e2$ unless $ph1$ has left $c1$. Thus there is no danger for $ph1$ to starve any more. Even if the process in $ph2$ would become very slow now and if the process in $ph1$ would begin to rush through its cycle again there is no danger

for $ph2$: Before $ph1$ could enter $e1$, $ph2$ has to leave $e2$, due to the coupling edge between $e1$ and $e2$.

There is an uncomfortable regulation for $ph2$ at this point of discussion: Occasionally $ph2$ would be bound to remain in $e2$ (eating) until $ph1$ (or maybe $ph3$) has left $c1$ (or $c3$), respectively. In order to prevent the philosophers to be gorged one could split the eating section of each *phi* into three phases *ei*, *rli*, *c'i* where *rli* is to be a *relax phase* and *c'i* is to be another clearing activity (see Fig. 19).

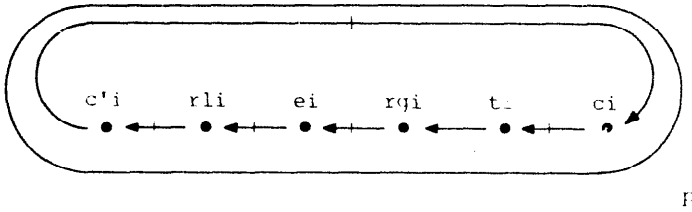


Fig. 19.

(The extended internal structure in Fig. 19 can be realized by an extended standard construction corresponding to that in Fig. 11.) An intuitive idea for this step is to separate the functions of the previous eating phase *ei* (eating *and* exciting $c(i-1), c(i+1)$) by distributing them among the 'particles' *ei* and *c'i* (which border the section $ei \leftrightarrow rli \leftrightarrow c'i$). *ei* would be coupled to its neighbour eating phases. In the same way *c'i* would be coupled to its neighbour phases of the form *c'j*. Thus the access to the sections $ei \leftrightarrow rli \leftrightarrow c'i$ would still be mutually exclusive for neighbours (as well as the capability to leave this section). (In this way the processes are occasionally prevented from passing a neighbour.) In order to allow a part *phi* to leave his eating phase at any time one needs one additional phase between *ei* and *c'i*, just *rli*: This phase is not coupled to any phase (and therefore a *slack phase* (see Section 1)). There would be excitement arrows from *c'i* to the neighbour phases of the form *cj*. Finally, *rgi* would excite the neighbour phases of the form *rlj*, *ci* would excite the neighbour phases *rgj* (as before).

If we understand *ci* to be the activity in which all forks are released by *phi* we are led to the more detailed description where we have – instead of *ci* – two phases $ci(i-1)$ and $ci(i+1)$ ($i-1$ and $i+1$ regarded modulo 5) in which *phi* releases the fork for $ph(i-1)$ or $ph(i+1)$, respectively. As *phi* would then interact with one neighbour only in each of these phases we would 'split up' the corresponding interaction edges as shown in Fig. 20. (The new internal structure in *phi*—which is displayed for $ph1$ in Fig. 20—will be explained in the next section.)

3. The generalized problem

It is not very difficult to see that for an arbitrary initial situation in Fig. 20 each philosopher can (or will) enter the next phase after some while. In this section we want to point out that we have so far already modeled the solution scheme for a

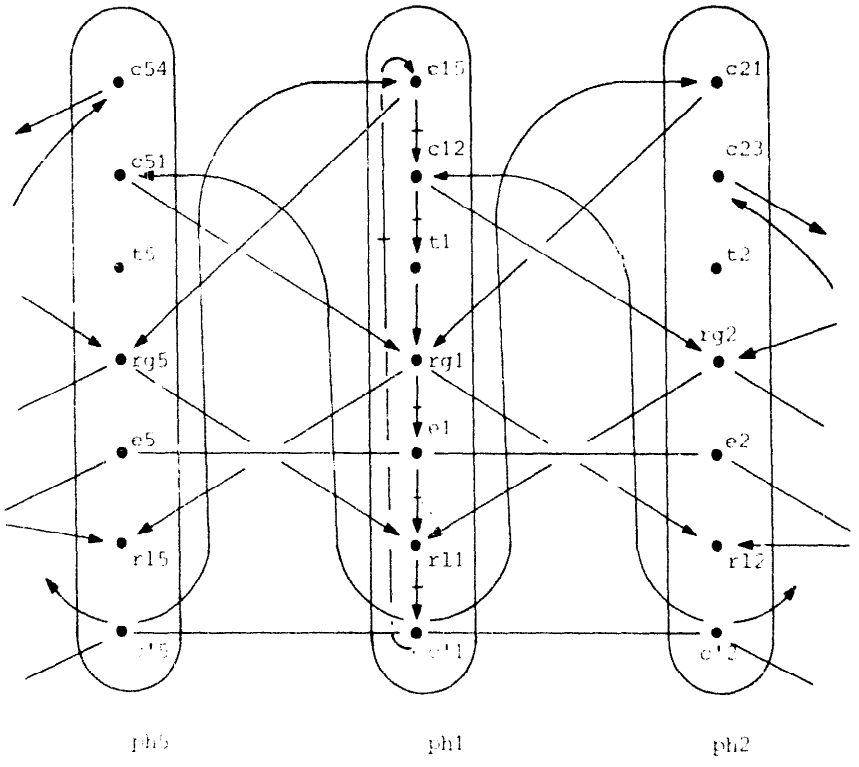


Fig. 20.

considerably more general interaction problem under distributed control. This problem was already addressed or discussed earlier (e.g. [1, 2]). We shall reconsider the problem and then prove the correctness of our proposed solution.

We consider a set of n subsystems $ph1, \ldots, phn$. In each of these parts there is a critical section in which the processes occasionally use shared resources in a mutually exclusive way. Under the constraints which were listed at the beginning of Section 2 for the special case of the 5 philosophers, a starvation-free solution is solicited.

Let us denote the critical section of phk by ek, the 'remainder section' by tk. If we define two parts to be neighbours if they use common resources the simple cycle for the special problem case in Fig. 15 is turned now into an arbitrary graph structure. In particular, we denote the neighbourhood phk by $N(phk)$ and write

(a) $\quad N(phk) =: \{phk(k1), phk(k2), \ldots, phk(ki_k)\}.$

In direct analogy to the special case in Section 2, we introduce auxiliary phases in phk such that

(b) $\quad phk =: \{ck(k1), \ldots, ck(ki_k), tk, rgk, ek, rlk, c'k\}.$

Corresponding to Fig. 20 we introduce an internal structure in phk such that

(c) $\quad ck(k1) \mapsto \cdots \mapsto ck(ki_k) \mapsto tk \to rgk \mapsto ek \mapsto$

$$\mapsto rlk \mapsto c'k \mapsto ck(k1).$$

In our language this is realized by connecting *phk* to an inert control part *stk* in a standard way as shown in Fig. 21. The dotted edges here represent local compatibility between *phk* and *stk* (the local complement of the coupling relation $K\langle phk, stk\rangle$), the arrow heads indicate excitations. (Note that there is no arrow into *tk*, also that *stk* is not in $N(phk)$!)

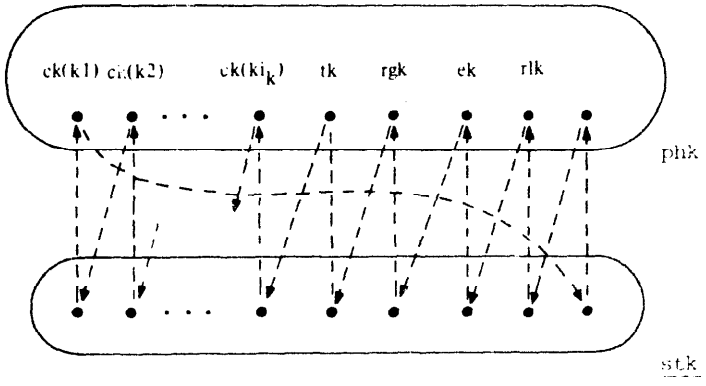


Fig. 21.

Let $p \in phk$. For the purpose of simplicity we only define connections between the *phk*, $k \in \{1, \ldots, n\}$ (taking the connections between *phk* and *stk* as known from Fig. 21):

(d) $$\tilde{K}(p) := \begin{cases} \emptyset, & p \in \{ek, c'k\}, \\ \{e(kl) \mid 1 \leq l \leq i_k\}, & p = ek, \\ \{c'(kl) \mid 1 \leq l \leq i_k\}, & p = c'k; \end{cases}$$

(e) $$\tilde{E}(p) := \begin{cases} \emptyset, & p \in \{tk, ek, rlk\}, \\ \{rg(kl)\}, & p = ck(kl),\ 1 \leq l \leq i_k, \\ \{rl(kl) \mid 1 \leq l \leq k\}, & p = rgk, \\ \{c(kl)k \mid 1 \leq l \leq k\}, & p = c'k. \end{cases}$$

Comparing (d) and (e) with the relations in Fig. 20 we can at once see that in the general case we have just the same bilateral relational schemes as in the special case. With these definitions we formulate the main result.

Theorem 3.1. *For the Interaction System* $(P, B, \mathrm{BI}, C, K, E)$ *where* $B = \{phk \mid k = 1, \ldots, n\} \cup \{stk \mid k = 1, \ldots, n\}$ *and* $\mathrm{BI} = \{stk \mid k = 1, \ldots, n\}$ *and where* K *and* E *are defined in* (d), (e) *and by Fig.* 21, *the following holds*:

An arbitrary part phk being in an arbitrary phase p cannot be prevented forever from leaving p.

Proof. (0) *phk*, $1 \leq k \leq n$, is obviously never prevented by a part *phj* from leaving *tk* or *ek*: The follower phase *rgk* (or *rlk*) can be entered at any time, and $\tilde{E}(\{tk, ek\}) = \emptyset$.

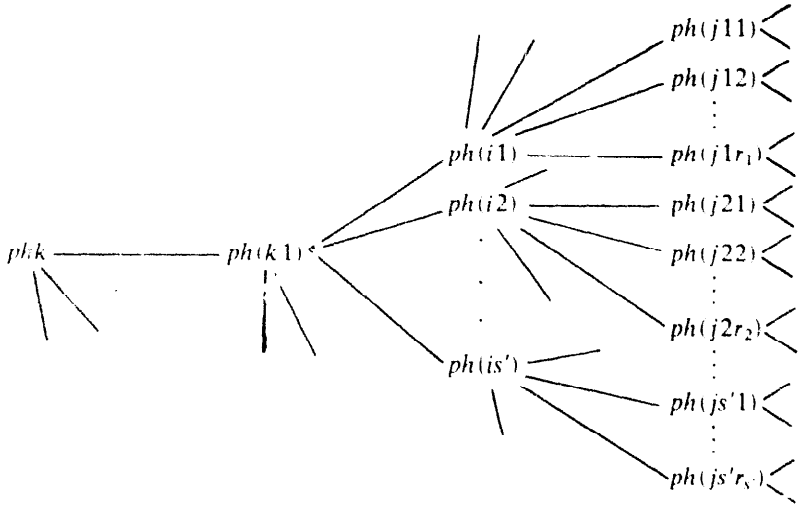
(1) Let us assume that *phk*, $1 \leq k \leq n$, prevents a part in $N(phk)$, without loss of generality $ph(k1)$, from leaving one of its phases. *phk* would be in one of the

phases $ck(k1)$, rgk, ek, rlk, $c'k$. Correspondingly $phk(k1)$ would be in $c'(k1)$, $c(k1)k$, $rg(k1)$, $rg(k1)$, $rl(k1)$, respectively. Whenever phk has then left one of the mentioned phases it cannot return to the same phase (i.e. after completing the internal cycle) unless $ph(k1)$ has left the initially locked phase: phk would at latest be stopped in rlk, $c'k$, $ck(k1)$, $ck(k1)$, rgk, respectively (due to the definition of K and E).

(2) Assume that for $1 \leq k \leq n$, phk is in $ck(k1)$ and cannot leave this phase in a case $c_0 \in C$. Due to the definition of E we know that for $ph(k1)$: $rg(k1) \in c_0$.

(3) If $ph(k1)$ cannot leave $rg(k1)$ then there are parts $ph(i1), ph(i2), \ldots, ph(is) \in N(ph(k1))$, $s \leq i_{(k1)}$ such that c_0 contains either $e(il)$ or $rl(il)$, $l = 1, \ldots, s$. Because of (0) $ph(il)$ can leave $e(il)$ at any time. So we may restrict ourselves to the situation where $ph(il)$ is in the phase $rl(il)$, $l = 1, \ldots, s$, and cannot leave $rl(il)$ for $l = 1, \ldots, s'$ and $s' < s$ (without loss of generality).

(4) Then we may assume that for every $ph(il)$ we have just $ph(jl1), ph(jl2), \ldots, ph(jls'_l) \in N(ph(il))$ such that $c'jl1, c'jl2, \ldots, c'jls'_l \in c_0$. As in (2) we may assume that $ph(jl1), ph(jl2), \ldots, ph(jlr_l)$, $r_l \leq s'_l$, cannot leave $c'jl1, c'jl2, \ldots, c'jlr_l$. Therefore we have, for every l and $1 \leq m \leq r_l$: $ph(glm1), ph(glm2), \ldots, ph(glmt_m) \in N(ph(jlm))$ such that $c(glm1)(jlm)$, $c(glm2)(jlm), \ldots, c(glmt_m)(jlm) \in c_0$. As our assumption was that $ph(k1)$ is locked in $rg(k1)$ we know that $ph(k1) \notin \{ph(jlm) \mid 1 \leq m \leq r_l, 1 \leq l \leq s\}$. Consequently $ck(k1) \notin \{c(glm1)(jlm), c(glm2)(jlm), \ldots, c(glmt_m)(jlm)\}$.



(5) Here we have reached the end of the causal chain under the assumption that in c_0 phk cannot leave $ck(k1)$: The only phase which is excited by $c(gkmf)(jlm)$, $f = 1, \ldots, t_m$, is $rg(jlm)$ which is *not* in c_0 (see (4)). As the follower phases of $c(glmf)(jlm)$ are accessible at any time $ph(glmf)$ will go to the next phase after some while. One of the possibly remaining parts in $N(ph(jlm))$, say $ph(glmt'_m)$, $t'_m \notin \{1, \ldots, t_m\}$, may succeed to go to $c(glmt'_m)(jlm)$ before $ph(jlm)$ has left $c'(jlm)$ [compare (4)]. This does not cause any complication: None of the $ph(glmf)$, $f \in \{1, \ldots, t_m, t'_m\}$, can enter $c(glmf)(jlm)$ again unless $ph(jlm)$ has left $c'(jlm)$

(according to (1)). As a result $ph(jlm)$ cannot be prevented from leaving $c'(jlm)$, and due to its internal structure it will do so. (The follower phase—which is of the form $c(jlm)(ly)$—is accessible at any time.)

(6) Due to (1) the $ph(jlm)$, $1 \leq m \leq r_l$, would not be able to again prevent $ph(il)$ from leaving $rl(il)$ unless $ph(il)$ has left this phase in a follower case of c_0. If in the meantime one of the remaining parts in $N(ph(il))$, say $ph(jlm')$, $m' \notin \{1, \ldots, r_l\}$ [compare (4)], has entered $c'(jlm')$ and is prevented from leaving this phase then we repeat the argumentation in (4) and (5) and so convince ourselves that $ph(jlm')$ will be allowed after some while to leave $c'(jlm')$ (and will do so). Altogether $ph(il)$, $1 \leq l \leq s'$, cannot be prevented after some while from leaving $rl(il)$, and it will do so due to its internal structure.

(7) In the meantime some parts of the form $ph(il')$, $l' \notin \{1, \ldots, s'\}$, might have entered their phase $rl(il')$ and be prevented from leaving this phase. (Consequently $ph(k1)$ would not yet be released.) But then we may argue along (3), (4), (5), (6), and we end up with the result that after some while $ph(il')$ can no longer be prevented from leaving $rl(il')$: This can be done because the argumentation above is in general terms and does not contain any special assumption on the parts (e.g. regarding cyclic connections). Due to (1) we conclude that eventually $ph(k1)$ cannot be locked in $rg(k1)$ forever. According to (1) each of the $ph(il)$ is stopped at latest in $c(il)(k1)$ as long as $ph(k1)$ is in $rg(k1)$. So, as a consequence of an assumption in (3), none of the $ph(il)$ can eventually enter $e(il)$ as long as $ph(k1)$ has not left $rg(k1)$: Thus $e(k1)$ is eventually accessible, and $ph(k1)$ will enter it according to its internal structure induced by $st(k1)$.

(8) As $ph(k1)$ is the only part which may prevent phk from leaving $ck(k1)$ we learn from (7) that phk will eventually leave $ck(k1)$ – also according to (1) and to the internal structure induced by stk.

(9) If phk is in one of the phases different from $ck(k1), ck(k2), \ldots, ck(i_k), tk, ek$, i.e. phk is in $gk, rlk, c'k$, then we begin an argumentation in corresponding terms at (3), (3), (4), respectively. After (7), (6), (5), respectively we can conclude then that phk cannot be prevented forever from leaving the mentioned phases.

This completes the proof. □

4. Discussion and conclusion

Because of its detailed discussion the proof of starvation-freeness looks a bit lengthy. However, the main idea behind is extremely simple (and, by the way, easily derivable from the graphical representation in Fig. 20): We check the longest possible causal chain by which phk may be prevented from leaving a given phase. *By construction* this chain has only 4 members.

Chang's solution of the Dining Philosophers' Problem [1] does not use a *global control* mechanism. It is based on message passing, so one needs at most some kind of *local* authority (arbiter) in order to implement the algorithm (e.g. in order to decide

whether a message got lost or not). On the other hand, the solution concept heavily depends on the structure of the neighbourhood graph in that also those philosophers who are not neighbours of two given connected philosophers get involved in the regulation of bilateral relationships. This is even more so for the generalized problem.

Devillers and Lauer offered a solution for the generalized problem of Section 3 [2] by introducing asymmetrical *allowance counters*. There is clearly a close familiarity between the relations which are induced in [2] between the states of neighbours by use of the allowance counters, and our interaction relations between two parts. Except for the initialization phase of the distributed system in [2]—where e.g. a cycle of controllers has to be avoided—one does not need a global control mechanism: The strategy of each process depends only on its neighbours. This is also the case in our solution because the model of Interaction Systems is completely based on *local interaction* primitives: We have two different types of *static* bilateral relations and axioms (Axioms 1.1 to 1.5) which determine their operational semantics. (Hence we have a graphical modeling tool including even behavioral aspects completely.)

We wanted to point out how transparent and flexible our formal modeling instruments are. So we designed, in Section 2, a solution for the 5 philosophers in a *stepwise* procedure. Although we have an interaction language we are able to model (induced) flow structures in the parts. There seems to be no special advantage to model just a simple cycle in the *phk* in this manner. (How the concept of induced patterns of behaviour can be used in general is worked out to some extent in [8].) However, in this induced structure it is left to the parts *phk* to leave their thinking phase or to stay there (while other transitions are forced to occur: compare also the discussion after Axiom 1.5). This would be conceivable if the 'normal' job of the processes in *phk* would be done in the phase *tk* and if only in some worst case—which would arise e.g. by influences from a non-visible part of the environment—*phk* would be forced to go to *ek*. In the solution represented in Fig. 20, Axiom 1.1 implies that a philosopher has no *internal* difficulties to leave a given phase once he is forced from outside to do so. This is a very *weak and local* form of the finite-delay property. As the interaction between any two philosophers in section 2 does not depend on the special neighbourhood graph—if one neglects the multiplicity of the phases *ck(il)*—the interaction scheme is unchanged (and so an *invariant*) when we consider the general problem case in section 3.

The discussion around Dijkstra's program in Section 2 gives already the feeling that for interaction problems of the type which is dealt with here one cannot at the same time guarantee starvation-freeness and avoid global control mechanisms in an implementation, purely on the basis of *mutual exclusion* (semaphores). This was confirmed during stimulating discussions which the author had with Michael Rabin in 1981, and it led to another motivation for our additional excitement relations. Moreover, in this way the discussion about implementation of the interaction structures is typically a discussion about the widest possible distribution of control. (We have already pointed out in [9] how the excitement primitives can be

implemented in a multiprocessor environment by use of operations like COMPARE AND SWAP in the IBM assembler code.) With the help of a recent distributed algorithm by Winkowski [14] we have meanwhile succeeded (with J. Winkowski and A. Maggiolo-Schettini) to design an implementation scheme for the model of Interaction Systems essentially in terms of local message passing between interconnected partners and under minimal local control (basically an arbiter function, e.g. in order to decide whether a message got lost or not). This is matter of a separate publication and of experimental investigations.

Besides that, there is an ongoing work to use our methodology of interaction as a tool which admits a *formal* specification of design requirements even at a very early design stage in case of large socio-technical systems. This is a consequence of the fact that our 'language' is based on *negative* and *local* interaction relations in terms of which many of the early known or fixed requirements for such systems (e.g. resource constraints) can conveniently be modelled in a flexible (interactive) way. This is beyond the scope of a purely theoretical discussion and will be published elsewhere.

Acknowledgment

I would like to appreciate the very constructive and careful comments from one of the unknown referees. Also, the concepts presented here could considerably be clarified after discussions with my colleague Eike Best.

References

[1] E. Chang, *n*-Philosophers: an exercise in distributed control, *Comput. Networks* (2) (1980) 71–76.
[2] R.E. Devillers and P.E. Lauer, A general mechanism for avoiding starvation with distributed control, *Information Processing Lett.* **7** (3) (1978) 156–158.
[3] E.W. Dijkstra, Co-operating sequential processes, in: F. Genuys, Ed., *Programming Languages* (Academic Press, New York, 1968) 43–112.
[4] E.W. Dijkstra, Hierarchical ordering of sequential processes, *Acta Informat.* **1** (2) (1971) 115–138.
[5] E.W. Dijkstra, Aspects of reasoning effectively about distributed systems (EWD 625); also in: B. Shaw (Ed.), *Proc. Joint IBM/University of Newcastle upon Tyne Seminar on Distributed Systems*, (1978).
[6] C.A.R. Hoare, Communicating sequential processes, *Comm. ACM* **21** (8) (1978) 666–677.
[7] J.D. Ichbiah, Rationale for the design of the ADA programming language, *SIGPLAN Notices* **14** (6) (1980).
[8] N. Kraft and H. Wedde, Inducing patterns of behaviour in distributed system parts, Lecture Notes in Computer Science **88** (Springer, Berlin, 1980) 375–386.
[9] N. Kraft and H. Wedde, Modeling principles of formal communication by use of interaction systems, Technical Report GMD-ISF 80.08 GMD Bonn (1980).
[10] A. Maggiolo-Schettini, H. Wedde and J. Winkowski, Modelling a solution for a control problem in distributed systems by restrictions, *Theoret. Comput. Sci.* **13** (1980) 61–83.
[11] H. Wedde, Lose Kopplung von Systemkomponenten, *Berichte der GMD* **96** (1975).

[12] H. Wedde, A starvation-free solution for the dining philosophers' problem by use of interaction systems, *Proc. MFCS'81 Symposium*, Lecture Notes in Computer Science **118** (Springer, Berlin, 1981) 534–543.

[13] H. Wedde and J. Winkowski, Determining processes by violations, *Proc. MFCS'77 Symposium*, Lecture Notes in Computer Science **53** (Springer, Berlin, 1977) 549–559.

[14] J. Winkowski, Protocols of accessing overlapping sets of resources, *Information Processing Lett.* **12** (5) (1981) 239–243.