TERMINATION DETECTION FOR DIFFUSING COMPUTATIONS *

Edsger W. DIJKSTRA
Burroughs, 5671 AL Nuenen, The Netherlands

C.S. SCHOLTEN
Philips Research Laboratories, 5656 AA Eindhoven, The Netherlands

Received 2 June 1980

Concurrency, termination detection, distributed control, separation of concerns, message-based systems, correctness proving, diffusing computations, networks, cornets, activation tree, nondeterminacy

The following seems to capture the quintessence of a situation that is not unusual in distributed processing. Consider a finite, directed graph. If the graph contains an edge from node A to node B, we call B 'a successor of A', and A 'a predecessor of B'. We assume the existence of a node without incoming edges; this node will be called 'the environment' (because it acts as such with respect to the rest of the graph). The other nodes will be called 'the internal nodes'.

For each node its initial state will be called 'the neutral state'. A so-called 'diffusing computation' is started when the environment sends – of its own accord, so to speak – a message to one or more of its successors; it is supposed to do this just once. After reception of its first message, an internal node is free to send messages to its successors. It is this feature that inspired the name 'diffusing computation'.

We shall confine our attention to diffusing computations for which it can be proved that also each internal node will send only a finite number of messages. For such a computation eventually each node will reach the situation in which it neither sends nor receives any more messages; when all nodes have reached that state, the whole graph is as dead as a doornail and the diffusing computation is defined to have terminated.

Our problem is the design of a signalling scheme – to be superimposed on the diffusing computation proper – such that, when the diffusing computation proper has thus terminated, the fact of this completion will eventually be signalled back to the environment. Besides a node's ability to receive messages from its predecessors and to send messages to its successors, we assume each node also to be able to receive 'signals' from its successors and to send 'signals' to its predecessors; in other words, each edge is assumed to be able to accommodate two-way traffic, but only messages of the computation proper in the one direction and signals in the opposite direction. We shall impose that in the total computation – i.e. from the moment that the environment sent its messages to the rest of the graph until it has received the completion signal – each edge will have carried as many messages in the one direction as it has carried signals in the opposite direction.

For each edge we define its 'deficit' as the number of messages transmitted along it minus the number of signals returned along it; because no node is supposed to be equipped with the clairvoyance that it would need to predict how many more messages of the computation proper it is going to receive, we impose, to begin with, the invariant

P0: each edge has a non-negative deficit,

a relation which is obviously satisfied initially.

The obligation to keep P0 invariant does not constrain the sending of messages; a signal, however, may only be sent by a node with at least one incoming

* This manuscript was finished on 26 January 1979.

edge that has a positive deficit.

For each node we define C = the sum of the deficits of its incoming edges, and relation P0 will be kept invariant by keeping

P1: $C \geqslant 0$

invariant for each node, complemented by the understanding – see later – that a node sending a signal will select for it an incoming edge with an initially positive deficit. (This selection is possible: because sending a signal implies for its sender C := C – 1, the invariance of P1 implies the initial validity of $C \geqslant 1$, i.e. the existence of at least one such incoming edge.)

Similarly we define for each node D = the sum of the deficits of its outgoing edges. From P0 we conclude that we have for each node

P2: $D \geqslant 0$.

For the sequel we redefine 'neutral state' to mean any state with C = 0 **and** D = 0 (which holds initially).

In our proposal the sending of messages and signals is further constrained by the required invariance of

P3: $C > 0$ **or** $D = 0$

for each internal node. (We further postulate that no node is infinitely lazy.) Because the sending of a message implies D := D + 1 for its sender, the invariance of P3 excludes spontaneous message sending by an internal node in its neutral state; furthermore the invariance of P3 may prevent an internal node from reducing its C to zero, i.e. from sending the last signal currently due to its predecessors.

We observe that it suffices when each sending internal node keeps P3 invariant for *itself*, because

(1) the sending of a message then keeps P3 invariant for *all* internal nodes: for the sending node by virtue of its construction, for the receiving successor by virtue of the fact that its C is increased by 1, and for all other nodes because their C's and D's remain unaffected, and

(2) the sending of a signal then keeps P3 invariant for *all* internal nodes: for the sending node by virtue of its construction, for the receiving predecessor by virtue of the fact that the accompanying decrease of its D by 1 can never destroy the truth of D = 0 on account of P2, and for all other nodes because their C's and D's remain unchanged.

A node in such a state of the computation proper that it has messages to send should maintain $C > 0$; otherwise, because the sending of a signal includes for its sender C := C – 1, the invariance of P1 **and** P3 requires by the axiom of assignment that the act of signalling be guarded by

G: $(C - 1 \geqslant 0)$ **and** $(C - 1 > 0$ **or** $D = 0)$

which is equivalent to

G: $C > 1$ **or** $(C = 1$ **and** $D = 0)$.

When the computation proper has terminated, no C is increased anymore; the ensuing signalling, as guarded in each internal node by G, will terminate because each sending of a signal decreases the sum of all the C's over the graph, a sum that is bounded from below on account of P1. Hence, when the computation proper has terminated, the system will reach the 'ultimate state' in which neither messages nor signals are sent anymore. From the fact that no more signals are sent we conclude that in the ultimate state **non** G holds for each internal node, which under the truth of P1 and P2 reduces to

$C = 0$ **or** $(C = 1$ **and** $D > 0)$.

For the environment, which has no predecessors, we always have

$C = 0$ **and** $D \geqslant 0$.

Hence, in the ultimate state we have

$C \leqslant D$

for all nodes. Because the sum of the C's over the whole graph equals the sum of the D's over the whole graph, we have in the ultimate state

$C = D$

for all nodes. Because the environment always has C = 0, we conclude our

Theorem 1. A bounded number of steps after the diffusing computation has terminated, the environment will have returned to the neutral state.

The above theorem tells us that after termination of the diffusing computation the moment will come that the environment has returned to the neutral state. Conversely we would like to conclude from the fact that the environment has returned to the neutral state

that, indeed, the diffusing computation has terminated. Calling a node that is not in its neutral state – i.e. a node with $C > 0$ **or** $D > 0$ – 'an engaged node', we propose

P4: all engaged internal nodes are reachable from the environment via directed paths, all edges of which have positive deficits,

because under the truth of P4, a neutral environment – which has no outgoing edges with a positive deficit! – implies that all internal nodes are in the neutral state as well and, therefore, the diffusing computation has terminated (by definition, as, on account of P3, no internal node will then send anymore messages).

Up till now, a node allowed to send a signal was constrained in its selection of predecessor to receive the signal only by the requirement that on all its incoming edges the deficit should remain non-negative. In the following we shall show how, by further restriction of the selection of predecessor to receive the signal, the invariance of P4 can be maintained.

Up till now each node's signalling obligation can be characterized by a bag, such that the deficit of the edge from A to B equals the number of occurrences of the name of A in the bag of B. A node in the neutral state has an empty bag, each reception by B of a message from A causes the name of A to be added to B's bag, which by this mechanism can be filled with names of predecessors of B, and the transmission of a signal from B to its predecessor A is accompanied by the removal of one occurrence of A's name from B's bag (in which the name of A could occur several times). Note that for each node, C equals the number of elements in its bag.

We can ensure P4 by replacing each bag by what we have dubbed 'a cornet'. The name 'cornet' has been chosen because, like in a pointed bag, one element contained in it enjoys the special status of being 'the oldest element': whereas a stack is characterized by 'last in, first out', a cornet is characterized by the much weaker 'very first in, very last out'.

Let 'the edge from A to B is an engagement edge' mean 'the name of A is the oldest element in B's cornet'. We observe:

(a) each engagement edge connects two engaged nodes (because it has a positive deficit and, hence, leads from a node with $D > 0$ to a node with $C > 0$);

(b) engagement edges do not form cycles (because, when the edge from A to B became an engagement edge, B was initially neutral and, hence, had no outgoing engagement edge);

(c) each engaged internal node has one incoming engagement edge (on account of P3 and because its bag has been replaced by a cornet).

From (a), (b), and (c) we conclude that the engagement edges form a rooted tree – with the environment as its root – to which each engaged node, but no neutral node belongs. Hence its edges provide the paths whose existence implies the truth of P4, and for our system with cornets instead of bags we have proved

Theorem 2. When the environment has returned to the neutral state, the diffusing computation has terminated.

This concludes the description and correctness proof of our signalling scheme.

Concluding remarks

Note that an internal node, while it needs to keep track of the deficits of its incoming edges, does not need to keep a record of to which of its successors it has sent messages, nor from which of its successors it has received signals: for the implementation the counter D suffices.

Note further that our signalling scheme is perfectly general in the sense that we have made no assumption about the topology of the rest of the graph: in particular, neither merging, nor even cyclic paths have been excluded.

Note further that our signalling scheme is also perfectly general in the sense that it can be superimposed on *any* diffusing computation fired from a single environment. In particular, because we have not excluded that, in the course of a single diffusing computation, internal nodes switch back and forth between the neutral and the engaged state several times, our highly non-deterministic diffusing computation is 'free' to behave like any more specific one (ultimately even like a fully deterministic one). In other words, we can appreciate the study of a single highly non-deterministic algorithm as an effective way of studying a whole class of algorithms: the non-deter-

ministic algorithm emerges when, abstracting from their mutual differences, we concentrate on what the many algorithms of the class have in common.

Acknowledgements

We are, foremost and all, indebted to the members of The Tuesday Afternoon Club: in one session an instance of our problem statement was born, in subsequent sessions our successive generalizations were followed closely, and our texts were screened carefully. We are indebted to Dr. P.M. Merlin from the Technion-Israel Institute of Technology, Haifa, who, in a lecture he gave at the University of Newcastle-upon-Tyne, inspired the problem. We are also grateful to C.A.R. Hoare for his encouragement and for the opportunity he provided – by organizing under the auspices of the Science Research Council a Workshop on Distributed Processing – for presenting our solution, before this final text was written.