Short Communications
Operating Systems

Self-stabilizing Systems in Spite of Distributed Control

Edsger W. Dijkstra
Burroughs Corporation

Key Words and Phrases: multiprocessing, networks, self-stabilization, synchronization, mutual exclusion, robustness, sharing, error recovery, distributed control, harmonious cooperation, self-repair
CR Categories: 4.32

The synchronization task between loosely coupled cyclic sequential processes (as can be distinguished in, for instance, operating systems) can be viewed as keeping the relation "the system is in a legitimate state" invariant. As a result, each individual process step that could possibly cause violation of that relation has to be preceded by a test deciding whether the process in question is allowed to proceed or has to be delayed. The resulting design is readily—and quite systematically—implemented if the different processes can be granted mutually exclusive access to a common store in which "the current system state" is recorded.

A complication arises if there is no such commonly accessible store and, therefore, "the current system state" must be recorded in variables distributed over the various processes; and a further complication arises if the communication facilities are limited in the sense that each process can only exchange information with "its neighbors," i.e. a small subset of the total set of processes. The complication is that the behavior of a process can only be influenced by that part of the total current system state description that is available to it; local actions taken on account of local information must accomplish a global objective. Such systems (with what is quite aptly called "distributed control") have been designed, but all such designs I was familiar with were not "self-stabilizing" in the sense that, when once (erroneously) in an illegitimate state, they could—and usually did!—remain so forever. Whether the property of self self-stabilization—for a more precise definition, see below—is interesting as a starting procedure, for the sake of robustness or merely as an intriguing problem, falls outside the scope of this article. It could be of relevance on a scale ranging from a worldwide network to common bus control. (I have been told that the first solution shown below was used a few weeks after its discovery in a system where two resource-sharing computers were coupled via a rather primitive channel along which they had to arrange their cooperation.)

Author's address: Burroughs Corporation, Plataanstraat 5, Nuenen-4565, The Netherlands.

We consider a connected graph in which the majority of the possible edges are missing and a finite state machine is placed at each node; machines placed in directly connected nodes are called each other's neighbors. For each machine one or more so-called "privileges" are defined, i.e. boolean functions of its own state and the states of its neighbors; when such a boolean function is true, we say that the privilege is "present." In order to model the undefined speed ratios of the various machines, we introduce a central daemon—its replacement by a distributed daemon falls outside the scope of this article—that can "select" one of the privileges present. The machine enjoying the selected privilege will then make its "move"; i.e. it is brought into a new state that is a function of its old state and the states of its neighbors. If for such a machine more than one privilege is present, the new state may also depend on the privilege selected. After completion of the move, the daemon will select a new privilege.

Furthermore there is a global criterion, telling whether the system as a whole is in a "legitimate" state. We require that: (1) in each legitimate state one or more privileges will be present; (2) in each legitimate state each possible move will bring the system again in a legitimate state; (3) each privilege must be present in at least one legitimate state; and (4) for any pair of legitimate states there exists a sequence of moves transferring the system from the one into the other.

We call the system "self-stabilizing" if and only if, regardless of the initial state and regardless of the privilege selected each time for the next move, at least one privilege will always be present and the system is guaranteed to find itself in a legitimate state after a finite number of moves. For more than a year—at least to my knowledge—it has been an open question whether nontrivial (e.g. all states legitimate is considered trivial) self-stabilizing systems could exist. It is not directly obvious whether the local moves can assure convergence toward satisfaction of such a global criterion; the nondeterminacy as embodied by the daemon is an added complication. The question is settled by each of the following three constructs. For brevity's sake most of the heuristics that led me to find them, together with the proofs that they satisfy the requirements, have been omitted and—to quote Douglas T. Ross's comment on an earlier draft, "the appreciation is left as an exercise for the reader." (For the cyclic arrangement discussed below the discovery that not all machines could be identical was the crucial one.)

In all three solutions we consider $N + 1$ machines, numbered from 0 through N. In order to avoid avoidable subscripts, I shall use for machine *nr* . *i*:

L: to refer to the state of its lefthand neighbor, machine *nr* . $(i - 1)\mathbf{mod}(N + 1)$,
S: to refer to the state of itself, machine *nr* . *i*,
R: to refer to the state of its righthand neighbor, machine *nr* . $(i+1)\mathbf{mod}(N+1)$.

In other words, we confine ourselves to machines placed in a ring (a ring being roughly the sparsest connected graph I could think of); machine *nr* . 0 will also be called "the bottom machine," machine *nr* . *N* will also be called "the top machine." For the legitimate states, I have chosen those states in which exactly one privilege is present. In describing the designs we shall use the format:

if *privilege* **then** *corresponding move* **fi**".

Solution with K-state Machines (K > N)

Here each machine state is represented by an integer value S, satisfying $0 \le S < K$. For each machine, one privilege is defined, viz.
for the bottom machine:

```
if $L = S$ then $S := (S+1)\mathbf{mod}\ K$ fi
```

for the other machines:

```
if $L \neq S$ then $S := L$ fi
```

Note 1. With a central daemon the relation $K \geq N$ is sufficient.

Note 2. This solution has been generalized by C.S. Scholten [1] for an arbitrary network in which the degree of freedom in the legitimate state is that of the special Petri-nets called "event graphs": along each independent cycle the number of privileges eventually converges toward an arbitrary predetermined constant.

Solution with Four-state Machines

Here each machine state is represented by two booleans *xS* and *upS*. For the bottom machine *upS* = **true** by definition, for the top machine *upS* = **false** by definition: these two machines are therefore only two-state machines. The privileges are defined as follows:
for the bottom machine:

```
if $xS = xR$ and non $upR$ then $xS :=$ non $xS$ fi
```

for the top machine:

```
if $xS \neq xL$ then $xS :=$ non $xS$ fi
```

for the other machines:

```
if $xS \neq xL$ then $xS :=$ non $xS$; $upS :=$ true fi;
if $xS = xR$ and $upS$ and non $upR$ then $upS :=$ false fi
```

The four-state machines may enjoy two privileges. The neighbor relation between bottom and top machines is not exploited; we may merge them into a single machine, which is then also a four-state machine for which also two privileges have been defined.

Solution with Three-state Machines

Here each machine state is represented by an integer value S, satisfying $0 \le S < 3$. The privileges are defined as follows:
for the bottom machine:

```
if $(S+1)\mathbf{mod}\ 3 = R$ then $S := (S-1)\mathbf{mod}\ 3$ fi
```

for the top machine:

```
if $L = R$ and $(L+1)\mathbf{mod}\ 3 \neq S$ then $S := (L+1)\mathbf{mod}\ 3$ fi
```

for the other machines:

```
if $(S+1)\mathbf{mod}\ 3 = L$ then $S := L$ fi;
if $(S+1)\mathbf{mod}\ 3 = R$ then $S := R$ fi
```

Again the machine $nr \cdot i$ with $0 < i < N$ may enjoy two privileges: the neighbor relation between bottom and top machine has been exploited.

Acknowledgments are due to C.S. Scholten who unmasked an earlier effort as fallacious and since then has generalized the first solution, to C.A.R. Hoare and M. Woodger whose fascination by the first two solutions was an incentive to find the third one, and to the referees whose comments regarding the presentation of these results have been most helpful.

Received November 1973; revised June 1974

References
1. Scholten, C.S. Private communication.

Short Communications
Scientific Applications

An On-Site Data Management System Application in Field Archaeology

J.A. Brown and Bernard Werner
Northwestern University

Key Words and Phrases: APL-PLUS, archaeology, Koster prehistoric site, on-site data management, retrieval systems
CR Categories: 3.36, 3.70, 3.73, 3.89

Recent expansion of the scale and scope of large multi-disciplinary archaeological programs in the United States has necessitated a corresponding development of data base management systems in a time-sharing computer technology.

The computerized data processing and information retrieval system described here is specifically designed to meet the requirements of current investigations by

This research was supported by NSF grant GS-33865. Authors' addresses: J. A. Brown, Northwestern University, Archeological Program, 2000 Sheridan Rd., Evanston, IL 60201; B. Werner, Scientific Time Sharing Corp., Monroeville Mall 229, Monroeville, PA 15146.