ON A SOLUTION TO THE CIGARETTE SMOKERS' PROBLEM

(without conditional statements)

D. L. Parnas

July, 1972

ABSTRACT

This report discusses a problem first introduced by Patil [1]. Patil has presented a proof that the problem cannot be solved using the P and V operations introduced by Dijkstra [3] unless conditional statements are used. This report contains a solution to the problem as defined by Patil and shows that Patil's proof overlooks important capabilities for P and V which were used by Dijkstra in his original report. This report also discusses the need for the generalized operators suggested by Patil.

Introduction

In a widely circulated and referenced memorandum [1,2], Suhas Patil has introduced a synchronization problem entitled "The Cigarette Smokers' Problem". He claims that the problem cannot be solved using the P and V primitives introduced by Dijkstra [3] unless conditional statements are also used. He supports that claim with an elaborate proof in terms of Petri Nets. On the basis of that proof, Patil concludes that the P and V primitives are not sufficiently powerful and that more complex operations are needed. In this paper we present a solution to the Cigarette Smokers' Problem, discuss the "flaw" in Patil's proof, and discuss the need for additional operations.

The Problem

To avoid the possibility of misinterpretation of the problem, and because the problem is defined with some quite arbitrary restrictions, we quote the following statement of the problem directly from the paper by Patil [1].

"The Cigarette Smokers' Problem

Three smokers are sitting at a table. One of them
has tobacco, another has cigarette papers, and the
third one has matches -- each one has a different
ingredient required to make and smoke a cigarette but
he may not give any ingredient to another. On the table
in front of them, two of the three ingredients will be
placed, and the smoker who has the necessary third
ingredient should pick up the ingredients from the table,
make a cigarette and smoke it. Since a new set of
ingredients will not be placed on the table until this
action is completed, the other smokers who cannot make
and smoke a cigarette with the ingredients on the table
must not interfere with the fellow who can. Therefore,
coordination is needed among the smokers. The actions
of the smokers without the coordination are as follows.

X - the smoker with tobacco	Y - the smoker with paper
α_x: pick up the paper pick up the match roll the cigarette light the cigarette smoke the cigarette return to α_x	α_y: pick up the tobacco pick up the match roll the cigarette light the cigarette smoke the cigarette return to α_y

Z - the smoker with matches

α_z: pick up the tobacco
pick up the paper
roll the cigarette
light the cigarette
smoke the cigarette
return to α_z

To inform the smokers about the ingredients which are
placed on the table, three semaphores a, b and c
representing tobacco, paper and match respectively,
are provided. On placing an ingredient on the table,
the corresponding semaphore is incremented by performing
a V[ ] operation. On the smokers' side semaphores X,
Y and Z are used to signal that a cigarette has been
smoked. The smoker who completes smoking a cigarette
performs the operation V[ ] on the corresponding
semaphore.

The smokers' problem is, then, to define some
additional semaphores and processes, if necessary, and
to introduce necessary P and V statements in these

processes so as to attain the necessary cooperation
among themselves required to ensure continued smoking
of cigarettes without reaching a deadlock.   There is,
however, a restriction that the process which supplies
the ingredients cannot be changed and that no conditional
statements may be used.   The first restriction is placed
because the smokers are seeking cooperation among them-
selves and therefore should not change the supplier, and
the second restriction is justified because P and V
primitives were introduced to avoid having to coordinate
processes by repeatedly testing a variable until it
changes its value, and because the operation of making
and smoking a cigarette has no conditional actions.
It will be seen that the cigarette smokers' problem has
no solution.

     To give a precise statement of the problem, we
will present below the processes involved. The set of
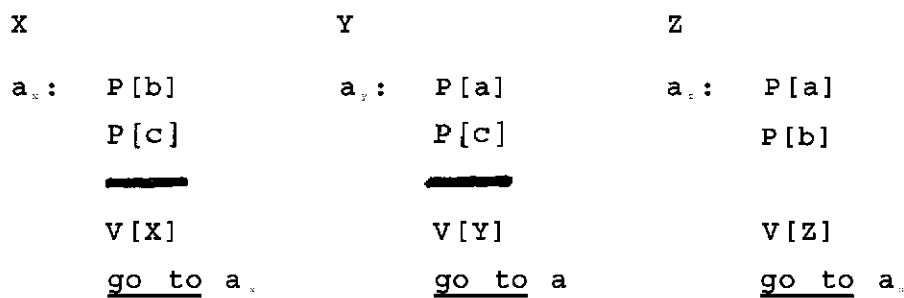processes in the enclosed area when taken together,
represent the agent who puts the ingredients on the table.
It should be recalled that semaphores a, b, and c
represent tobacco, paper and match respectively, and the
action of putting these ingredients on the table is
represented by V[ ] operation on the corresponding
semaphores.   Among the processes representing the agent,
such actions are performed by the processes R_a, R_b

and R_c which represent the three possible actions of

the agent. Initially semaphore s is 1 and the
processes R_a, R_b and R_c compete to perform P[s],

and the process which succeeds puts the ingredients
on the table by performing V[ ] on two of the semaphores.
Processes β_x, β_y and β_z detect the completion of

smoking of a cigarette; β_x performs V[s] when smoker

X indicates completion of smoking the cigarette by
performing V[X]. The semaphore s serves to signal
that it is time for the agent to place a new set of
ingredients on the table.

     On the smokers' side an attempt is made to
represent their actions by means of the processes X,
Y and Z.   These processes, however, do not correctly
represent their activity because of the possibility
of a deadlock. For example, in one case, tobacco and
paper are placed on the table, and process Y may
perform P[a] before process Z, the process which was
supposed to perform P[a], and thereby create a
deadlock in which no process is able to proceed.
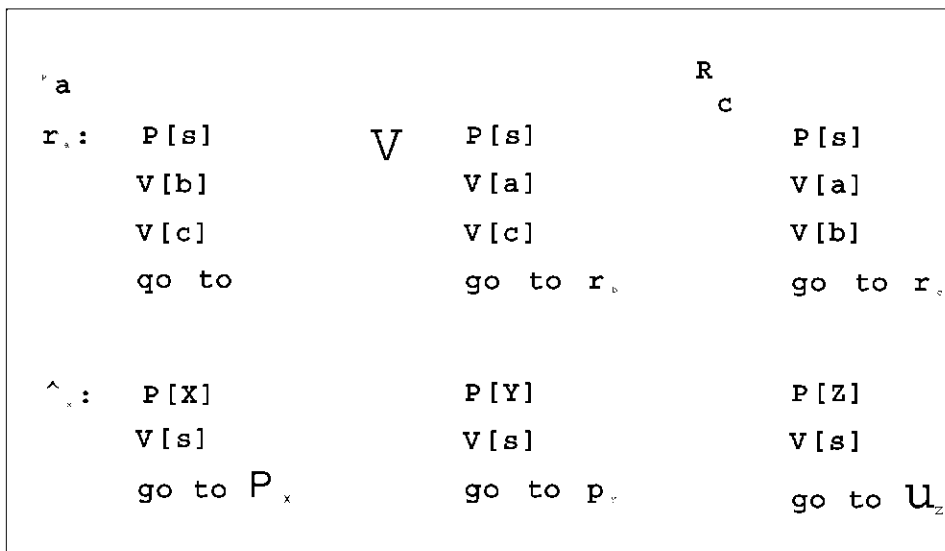
```
 a                                              R           the agent
                                                  c
 r :    P[s]            V      P[s]              P[s]
        V[b]                   V[a]              V[a]
        V[c]                   V[c]              V[b]
        go  to                 go  to  r         go  to  r      initially  s = 1
                                                               and  a,b,c,X,Y
                                                               and  Z = 0

 ^  :   P[X]                   P[Y]              P[Z]
  x
        V[s]                   V[s]              V[s]
        go to  P               go to  p         go to  U
               x                                       z
```

```
 X                     Y                     Z
 a  :   P[b]           a  :   P[a]           a  :   P[a]
  x                     y                     z
        P[c]                   P[c]                   P[b]
        ━━━━                   ━━━━                                the smokers
        V[X]                   V[Y]                   V[Z]
        go  to  a              go  to  a              go  to  a
```

The blank stands for some operation performed by
the process (in the case of the smoker this
operation is that of smoking the cigarette) .

The smokers' problem is to define additional sema-
phores and processes and to introduce appropriate P
and V statements so as to make them deadlock free.
No alteration may, however, be made to the processes
defining the agent."

End of Quotation

The Solution (processes and semaphores additional to the agent)

semaphore mutex; (initially 1)

integer t; (initially 0)

semaphore array S[1:6]; (initially 0)

δ_a: P (a);
    P (mutex);
    t ← t + 1;
    V(S[t]);
    V (mutex);
    go to δ_a;

δ_b: P (b);
    P (mutex);
    t ← t + 2;
    V(S[t]);
    V (mutex);
    go to δ_b;

δ_c: P (c);
    P (mutex);
    t ← t + 4
    V(S[t]);
    V (mutex);
    go to δ_c;

      X
      bc
α_x: $\overline{P(S[6])}$;

    t ← 0;
    ■■■
    V (X);

    go to α_x;

      Y
      ac
α_y: $\overline{P(S[5])}$;

    t ← 0;
    ■■■
    V (Y);

    go to α_y;

      z
      ab
α_z: $\overline{P(S[3])}$;

    t ← 0;
    ■■■
    V (Z);

    go to α_z;

Optional: if overflow is a problem

δ_1: P(S[1]);
    go to δ_1;

δ_2: P(S[2]);
    go to δ_2;

δ_3: P(S[4]);
    go to δ_3;

Comments

The last three processes are superfluous unless one worries about the ill-defined problem of semaphore overflow. The solution works by simulating a six branch case statement using a semaphore array and simple arithmetic operations. The solution is simpler than the solution given by Patil using conditionals.

On Patil's Proof

Patil [1] gives a method of representing cooperating processes as Petri nets. The scheme does not appear to be adequate when semaphore arrays are considered. Patil states that "the 'transition' representing the instruction P[S] has an additional arc from the place corresponding to semaphore S and from the transition corresponding to the instruction V[S], there is an additional arc to the place corresponding to the semaphore S." Every element of a semaphore array must be represented by its own place. The transitions corresponding to P and V operations may now place markers in any one (but only one) of these places. Patil's description of the action of a Petri net states that a transition places markers on all of its output places.

It appears then that Patil has used (but not stated) an assumption that there are no semaphore arrays. Since these arrays appeared in the earliest literature on the subject [3], the limitations reported by Patil are limitations of his primi-

tives, but they are not limitations on the primitives described by Dijkstra.

On a Complication Arising from the Introduction of Semaphore Arrays

The use of semaphore arrays does introduce one minor complication which has not been discussed in the literature. In a call of the form "P(S[E])" (where E represents an arbitrary integer valued expression), the evaluation of E must take place only once and <u>before</u> execution of the body of "P". The evaluation is not considered a part of the "primitive" P. (In other words, we must be able to consider the state of the system during evaluation of the expression, whereas we have no information about the state.of the system during the execution of a "Primitive" such as "P" or "V". Those interested in programming languages might be intrigued by the fact that the classical parameter passing modes found in ALGOL 60 are inadequate for this purpose. Were we to want to write an appropriate "P" algorithm in ALGOL 60, it would require use of the format P(s,E) where s is a semaphore array. Only in this way could we call E by value while being able to refer to s by name. We consider this minor difficulty to be a quirk in the design of ALGOL 60 rather than any limitation on the concept of the semaphore.

On the Yet Unsolved Problem

It is not the purpose of this paper to suggest that there
are no limitations to the capabilities of Dijkstra's semaphore
primitives. Our only conclusion is that the limitation reported
by Patil is not a limitation of the semaphore as introduced by
Dijkstra. There may well be other limitations of the semaphore
operations. While we disagree with the result presented by
Patil, we applaud his goal of a precise and substantial evaluation
of the Dijkstra primitives.

It is important, however, that such an investigation not
investigate the power of these primitives under artificial
restrictions. By artificial we mean restrictions which cannot
be justified by practical considerations. In this author's
opinion, restrictions prohibiting conditionals or semaphore
arrays are artificial.

The justifications given by Patil for eliminating conditional
statements are not valid. Some characteristics of the definition
of the agent are also artificial. In fact, the agent could be
modified to make the problem more difficult by removing the
restriction that the agent throws down only two resources and
waits until they have been taken before adding more [6]. Although
we do not have a solution to that more difficult problem (without
conditional statements) we do not conclude that the problem is
unsolvable.

Such an investigation is made more difficult because (1)
we have no firm definition of the set of problems which we wish

to solve, and (2) in using the PV system we have great freedom in the way that we assign tasks to processes and the way that we assign interpretations to the semaphores in use. Often problems which appear unsolvable are easily solved when additional processes or semaphores are introduced. It would be artificial to rule out such solutions in any investigation of the capabilities of semaphore primitives.

"P" and "V" are deliberately designed so that when there are several processes waiting for a "V" operation on a given semaphore, the choice of the process to be released by a "V" is not specified. This "non-determinism" is advantageous in writing programs where schedulers are subject to change or sometimes likely to be moved to another machine. On the other hand, it makes the solution of "priority problems" such as [4] more difficult. Perhaps some are impossible. The question is still open.

On More Powerful Primitives

Patil's paper ends with the introduction of a more powerful primitive than the "P" and "V" primitives. He suggests that the necessity of such a primitive is supported by the inability of "P" and "V" to solve the smokers' problem. Such proposals are not new -- a similar proposal was motivated by the "insufficiency" of "P" and "V" for problems similar to those solved in [4].

Such arguments are invalid since the problems can be solved. Further, we note that the generalized operation can be programmed in a straightforward way using P and V (and conditionals).

In the author's opinion, the use of the word "primitive" to describe routines which might have been called "monitor routines" or "operating system service calls" suggests that the programs implementing the operations be small and quickly executed. There are obvious practical advantages to placing such restrictions on a code which is the uninterruptable "heart" of an operating system. "P" and "V" have the requisite properties, the generalized operations do not.

There is no doubt that the generalized operations can be useful in describing certain processes. If one wants to describe such processes, one should build the operations, but they need not be built as "primitives". This attitude has been taken by the authors of [5] where some interesting new upper level operations are suggested.

The operations suggested by Patil have been called "parallel" operations because they simulate simultaneous activities on many semaphore variables. They make it easy to describe a single process which does tasks which could have been done by several cooperating sequential processes. Often one finds possibilities for parallel execution within such processes; these possibilities cannot be exploited by a multiprocessor system because one does not assign two processors to the same process simultaneously.[*] Often, because of the potential parallelism within the process, the program describing the process becomes very complex. In this sense, the generalized operation introduced by Patil and others

---

[*]Such a restriction is the operational definition of "process" or "task" in most operating systems.

is akin to the "go to" statement in programming languages. Both
add nothing to the set of soluble problems; both make it easier
to write programs which should not be written.

Acknowledgement

I am indebted to Wing Hing Huen for helping me to strengthen
one of the arguments in this paper. I am also grateful to P.
Wodon and P. J. Courtois for helpful comments.

References

[1] Patil, S. S., "Limitations and Capabilities of Dijkstra's
Semaphore Primitives for Coordination Among Processes",
Project MAC, Computational Structures Group Memo 57,
February, 1971.

[2] Project MAC - Progress Report, 1970-1971.

[3] Dijkstra, E. W., "Co-operating Sequential Processes" in
Programming Languages, F. Genuys Ed., Academic Press, New
York, 1968 [First published by T. H. Eindhoven, Eindhoven,
The Netherlands, 1965].

[4] Courtois, P. J., Heymans, F. and Parnas, D. L., "Concurrent
Control with Readers and Writers", CACM, October, 1971.

[5] Vantilborgh, H., van Lamsweerde, A., "On an Extension of
Dijkstra's Semaphore Primitives", Report R192, MBLE,
Laboratoire de Recherches, Brussels.

[6] P. J. Courtois; private communication.

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)	2a. REPORT SECURITY CLASSIFICATION
Computer Science Department	UNCLASSIFIED
Carnegie-Mellon University	2b. GROUP
Pittsburgh, Pa. 15213	

3. REPORT TITLE

ON A SOLUTION TO THE CIGARETTE SMOKERS' PROBLEM (without conditional statements)

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

Scientific   Final

5. AUTHOR(S) (First name, middle initial, last name)

D. L. Parnas

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
July, 1972	15	6
8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)	
WF-15-241-601		
b. PROJECT NO.		
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)	
d.		

10. DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
TECH OTHER	Information Systems
	Naval Research Laboratory
	Washington, D. C. 20390

13. ABSTRACT

This report discusses a problem first introduced by Patil 1 . Patil has presented a proof that the problem cannot be solved using the P and V operations introduced by Dijkstra 3 unless conditional statements are used. This report contains a solution to the problem as defined by Patil and shows that Patil's proof overlooks important capabilities for P and V which were used by Dijkstra in his original report. This report also discusses the need for the generalized operators suggested by Patil.

DD FORM 1473 NOV 65

DOCUMENT CONTROL DATA R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report la classified)

1. ORIGINATING ACTIVITY *(Corporate author)*	2a. REPORT SECURITY CLASSIFICATION
Computer Science Department	UNCLASSIFIED
Carnegie-Mellon University	2b. GROUP
Pittsburgh, Pa. 15213	

3. REPORT TITLE

ON A SOLUTION TO THE CIGARETTE SMOKERS' PROBLEM (without conditional statements)

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
Scientific   Final

5. AUTHOR(S) *(First name, middle Initial, la at name)*

D. L. Parnas

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
July, 1972	15	6

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
F44620-70-C-0107	
b. PROJECT NO.	
9769	
	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
C.61102F	

10. DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
TECH OTHER	Air Force Office of Scientific Rsch (NM)
	1400 Wilson Blvd.
	Arlington, Va. 22209

13. ABSTRACT

This report discusses a problem first introduced by Patil [1]. Patil has presented a proof that the problem cannot be solved using the P and V operations introduced by Dijkstra [3] unless conditional statements are used. This report contains a solution to the problem as defined by Patil and shows that Patil's proof overlooks important capabilities for P and V which were used by Dijkstra in his original report. This report also discusses the need for the generalized operators suggested by Patil.

DD FORM 1473