ショート・ノート

腕木システムの一般化*

斎　藤　信　男**

Abstract

Semaphore system proposed by E. W. Dijkstra is generally used for describing synchronization problems in concurrent process system. P-primitive is allowed to count down the value of a semaphore variable only when the pass condition is satisfied, while V-primitive is always allowed to count it up. In this sense, the semantics of these primitives are asymmetric.

This paper proposes a new pair of primitives $\bar{P}$ and $\bar{V}$. Both of them are required to satisfy the pass condition before modifying the value of a semaphore variable. Therefore, the semantics of these new primitives are considered to be symmetric. It is shown that the producer-consumer problem with a finite number of buffers can be conveniently described by using these new primitives.

1. はじめに

非同期処理システムにおける同期問題を記述する同期命令は，いろいろの体系が提案されているが，E. W. Dijkstra[1] の提案した腕木システム (semaphore system) が最も一般的なものと考えられ，その後，腕木システムの拡張体系が幾つか提案されている[2]．腕木システムにおいては，相反する意味を持つ P 命令と V 命令が使用されるが，V 命令は腕木変数の値をいつでもカウント・アップできるのに対し，P 命令は，ある通過条件を満すときだけカウント・ダウンできる[3]．その意味では，P 命令と V 命令は，非対称的である．この原則は，拡張された腕木システムにおいても守られている．

本稿では，カウント・ダウンをいつでも可能にし，カウント・アップに際しては，通過条件を遵守させるという意味を与える場合，また，カウント・ダウン，カウント・アップの両者に対して，通過条件を遵守させるという対称的な意味を与える場合について考察し，その応用例を示す．

2. 通過条件の変更

標準腕木システムにおいては，P 命令及び V 命令の意味は，次のように定義される．

規則 1

- P(s): (s は腕木変数)

  repeat: 〔**if** $s>0$ **then** $s:=s-1$

  eles go to repeat;〕

- V(s):

  〔$s:=s+1$;〕

但し，〔 〕内の処理は，途中で割り込みによって中断されることのないような非可分操作として実行されなければならない．S の値を，ある資源の数と考え，P 命令を資源の消費操作，V 命令を資源の生産操作と考えると，規則 1 は，

「存在しないものは，消費できない．」

という原則に従っていることになる．そこで，これと反対の原則

「存在するものは，生産できない．」

を用いて，$\bar{P}$ 及び $\bar{V}$ 命令の意味を定義すると，次のようになる．

規則 2

- P(s): (s は腕木変数)

  〔$s:=s-1$〕

---

* Genralization of Semaphore System by Nobuo SAITO (The University of Tsukuba Institute of Electronics and Information Science)

** 筑波大学電子・情報工学系

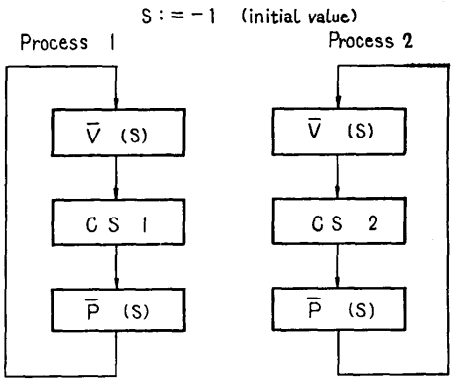

Fig. 1 mutual exclusion problem.

• $\bar{V}(s)$:
repeat: 〔**if** $s<0$ **then** $s:=s+1$
else go to repeat;〕

$\bar{P}(s)$ 及び $\bar{V}(s)$ 命令を用いても，相互排除問題[2]は，**Fig. 1** のように，標準腕木システムと全く同様にして記述することができる．また，生産者一消費者問題[2]も，P と $\bar{V}$，V と $\bar{P}$ とを入れ換えれば，全く同様に記述できる．この理由は，規則2を見てもわかるように，S の値が負の範囲で動いていると考えれば，通過条件と，カウント・アップ，カウント・ダウンとの組み合わせが標準腕木システムと等価になっているからである．

標準腕木システムにおいては，P 及び V 命令の実行回数に関し，次のような基本関係式が成り立つ[4]．

$$npp(s)=\min[np(s), C(s)+nv(s)] \quad (1)$$

ここに

$npp(s)$: P 命令で通過できた回数
$np(s)$: P 命令を実行した回数
$nv(s)$: V 命令を実行した回数
$C(s)$：腕木変数 S の初期値

通過条件を変更した $\bar{P}$ 及び $\bar{V}$ 命令に関しては，次のような基本関係式が成り立つことがわかる．

$$-n\bar{v}p(s)=\max[-n\bar{v}(s), C(s)-n\bar{p}(s)] \quad (2)$$

または，

$$n\bar{v}p(s)=\min[n\bar{v}(s), -C(s)+n\bar{p}(s)] \quad (3)$$

但し，$n\bar{v}p(s), n\bar{v}(s), n\bar{p}(s), C(s)$ は，(1)式で用いた意味に準ずる．

3. 通過条件の一般化

P 及び V 命令と，$\bar{P}$ 及び $\bar{V}$ 命令の体系においては，どちらか一方に通過条件を遵守させており，2つの命令の意味は，非対称的に定義されている．また，通過条件における S の値の比較値は0である．そこで，これらを一般化し，通過条件との組み合わせを対称的に定義し，また，通過条件において S と比較すべき値も，一般的に指定できるようにする．すなわち，次のような原則

「ある制限値以下しか存在しないものは，消費できない．」

「ある制限以上存在するものは，生産できない．」

を実現する P* 及び V* 命令を，次のように定義する．

規則 3

• P*(s, m): (s は腕木変数，m は整数)
repeat: 〔**if** $s>m$ **then** $s:=s-1$
else go to repeat;〕

• V*(s, n): (s は腕木変数，n は整数)
repeat: 〔**if** $s<n$ **then** $s:=s+1$
else go to repeat;〕

このとき，(1)に対応する基本関係式は，次のようになる．

$$np^*p(s,m)=\min[np^*(s,m), C(s)-m+nv^*p(s,n)] \quad (4)$$

$$-nv^*p(s,n)=\max[-nv^*(s,n), C(s)-n-np^*p(s,m)] \quad (5)$$

(5)式は，次のように変形できる．

$$nv^*p(s,n)=\min[nv^*(s,n), n-C(s)+np^*p(s,m)] \quad (6)$$

但し $np^*p(s,m)$, $nv^*p(s,n)$, $np^*(s,m)$, $nv^*(s,n)$, $C(s)$ は，(1)式で用いた意味に準ずる．

$C(s)=k$ とすると，n, m, k の大小関係により，S の値の変化は，次の6つの場合に分類される．

1) $m\leqq k\leqq n$

k が，それぞれの制限値の間にあり，P*, V* 共に，任意の順序で実行される．但し，$k=m$ のときは，最初は V* が，$k=n$ のときは，最初は P* が実行されなければならない．また，$m=n=k$ のときは，どの命令も通過できず，S の値は不変である．

2) $m\leqq n\leqq k$

S の値が $(n-1)$ になる迄，P* が続行され，以後，P*, V* が任意にあらわれる．$m=n$ のときは，S の値が n になると，以後，どの命令も通過できず，S の値は不変である．

3) $k\leqq m\leqq n$

S の値が $(m+1)$ になる迄 V* 命令が続行され，以

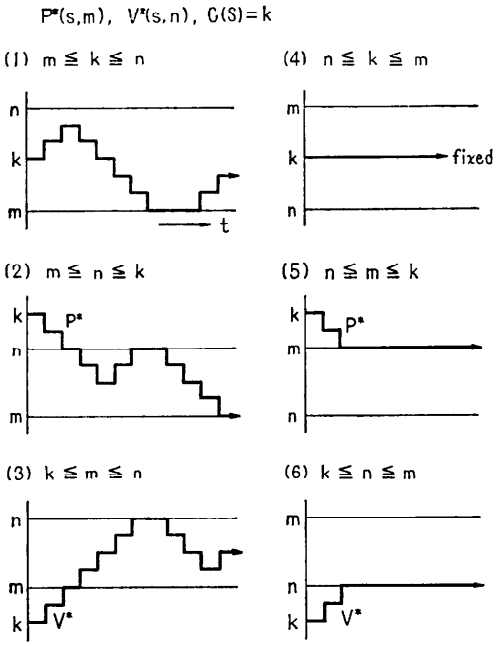

Fig. 2 behavior of semaphore variable for P* and V* primitives.

後，P*, V* が任意にあらわれる．$m=n$ のときは，S の値が m になると，以後，不変となる．

4) $n \leqq k \leqq m$

この状態においては，P*, V* 共に通過できず，したがって，S の値は不変である．

5) $n \leqq m \leqq k$

S の値が m になる迄，P* が続行され，以後，どの命令も通過できず，S の値は不変である．

6) $k \leqq n \leqq m$

S の値が n になる迄，V* が続行され，以後，どの命令も通過できず，S の値は不変である．

以上の状況を **Fig. 2** に示す．

P* 及び V* 命令を用いると，P, V 命令及び $\hat{\mathrm{P}}$, $\overline{\mathrm{V}}$ 命令は，次のように定義できる．

$$\begin{cases} \mathrm{P}(s) \equiv \mathrm{P}^*(S, 0) \\ \mathrm{V}(s) \equiv \mathrm{V}^*(S, +\infty) \end{cases}$$

$$\begin{cases} \hat{\mathrm{P}}(s) \equiv \mathrm{P}^*(S, -\infty) \\ \overline{\mathrm{V}}(s) \equiv \mathrm{V}^*(s, 0) \end{cases}$$

4. 応 用 例

2. で定義した $\hat{\mathrm{P}}$ 及び $\overline{\mathrm{V}}$ 命令は，本質的に P 及び V 命令と等価であり，相互排除問題，生産者一消費者問題などが記述できた．**3.** で定義した P* 及び V* 命令を用いると，有限バッファを持った生産者一消費者問題[1] が簡潔に記述できる．

```
begin
  integer numbuf; numbuf: =n;
  semaphore position, lock; position: =0;
                            lock: =1;
  parbegin
  producer: begin
    again 1: produce next portion;
             P* (lock, 0);
             add portion to buffer;
             V* (lock, 1);
             V* (position, numbuf);
           go to again 1
           end;
  consumer: begin
    again 2: P* (position, 0);
             P* (lock, 0);
             take portion from buffer:
             V* (lock, 1);
             process portion taken;
             go to again 2
           end
  parend
end;
```

生産者一消費者問題は，OS に頻繁にあらわれてくるが，有限のバッファしか持たないのが現実的である．したがって，これが簡潔に書けることは，有用であると思われる．

5. む す び

腕木システムの P 及び V 命令における通過条件の指定法を一般化し，対称的な意味を持つ P* 及び V* 命令を提案した．従来の P 及び V 命令は，その特殊な場合として定義できる．一般化した腕木システムを用いると，OS に実際によくあらわれる有限個のバッファを持った生産者一消費者問題が簡潔に記述できることを示した．また，新しい命令体系において成り立つ基本関係式を示した．今後，このような一般化を，PV Chunk, PV Multiple, up/down システムなどの拡張された腕木システムに対して行い，その有用性を検討することが必要であろう．

参 考 文 献

1) E. W. Dijkstra: Co-operating Sequential Pro-

cesses, **Programming Languages**, (ed. F. Genuys), Academic Press, New York, pp. 43～112 (1968).
2) たとえば，斎藤信男：OS の基礎理論(1)，情報処理，Vol. 15, No. 11, pp. 887～899 (1974).
3) 斎藤信男：同期基本命令の実現について，情報処理，Vol. 15, No. 11, pp. 841～849 (1974).
4) A. N. Habermann: Synchonization of Communicating Processes, CACM, Vol. 15, No. 3, pp. 171～176 (1972).

(昭和 50 年 5 月 13 日受付)