談話室

"PV 操作について" へのコメント

斎　藤　信　男*

はじめに

"PV 操作について" の徳田氏ら**の指摘どおり，文献 1) の PV 操作の定義は，明らかにミスであります．非負整数変数型，または，整数変数型のどちらかに統一しない限り，永久に block してしまうことがあり得ます．この点については，慶大の土居氏より以前に指摘があり，たとえば，京大大型計算機センタの第6回研究セミナ報告の中の筆者のレポートでは，整数変数型の定義に統一しています．無用の混乱を生じた方には，おわび致します．

以下に，徳田氏らの提案に対して若干のコメントを申し述べ，訂正させていただきます．

（I） さて，腕木システムの定義の仕方は，いろいろあり，混乱が生じ易いのですが，その一因は，定義の中で，更に，下のレベルの同期基本命令を用いていることです．腕木システムの形式的意味を論ずるためには，むしろ，busy form of waiting を用いた方がよいと思われます．たとえば，文献 2) に，筆者は，同期基本命令の一般形を，busy form of waiting を用いて定義しています（図-1 参照）．ここで，消費操作の*で，たとえば，block をし，生産操作の * で wakeup をすることにすれば，下のレベルの同期操作を利用でき，能率のよい実現が可能になるわけです．（勿論，図-1 でも，非可分操作を実現するという意味では，下のレベルの同期操作を利用しています．）

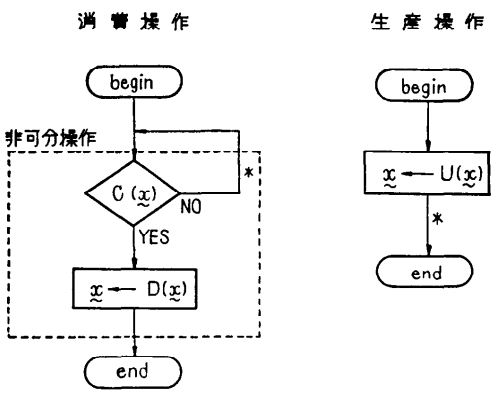

図-1　同期基本命令の一般形

"PV 操作について" の中では，非負整数型の場合を（1），（2）式を用いて定義していますが，（2）では，Q_s が空でないことを判断して，wakeup をし，s の値を1増加することはしていません．これは，たまたま，（1）のP操作でSの値が1減少することがわかっているので，相互に余計な加減算や代入をしないで済ませられるのです．しかし，PV Multiple や，PV Chunk では，一つの V 操作によって，どの P 操作が通過可能になるのかは，単純な関係では定義できません．したがって，wakeup された後で，もう一度，通過条件$C(x)$ の判定を行わせることにすれば，実現が簡単になるわけです．このようにすれば，V操作の中で，どのプロセスを wakeup すべきかを決定する必要はなく，通過できそうなプロセスを全て wakeup してやればよいことになります．勿論，こうすれば，単一の処理装置では能率が悪くなりますが，非常に多数の処理装置があるときは，うまくゆきそうです．

なお，"PV 操作について" の（3），（4）式で定義した整数型の場合，図-2（次頁参照）に示すような単純な busy form of waiting の形式をとることはできません．これは，下降関数による代入を，先に行ってしまうからです．このため，たとえば，図-3（次頁参照）に示すように，下降関数の代入の効果を打ち消して，再び，先頭から入ってくるという busy form of waiting の形式になります．このとき，消費操作の**印の点で，block をし，生産操作の**印の点で，wakeup することになります．（3）のように，Sへの代入が一回で済むのは，V 操作によって，必ず，1つは wakeup できることが，たまたま保証されているからです．PV Chunk や，PV Multiple の場合のように，V 操作と P 操作との関係が単純でない場合

* 筑波大学電子・情報工学系
** 情報処理 Vol. 17, No. 5, pp. 111～113

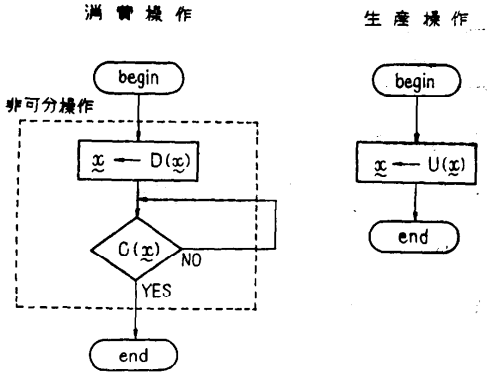

図-2　ビジー待ち形式

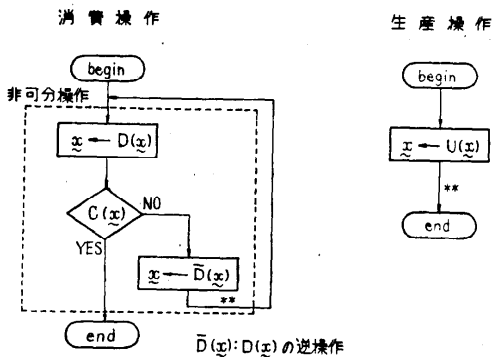

図-3　ビジー待ちの別形式

は，このように簡単にはいかないでしょう．

（II） PV Multiple の定義は，確かに複雑なものになり，実際に効率よく実現するのは，難しい問題となるでしょう．この場合，何らかの形の busy form of waiting は避けられないものと思われます．

"PV 操作について" で定義されている方法（conditional critical region を用いたもの）は，かなり簡潔に記述されています．この場合，**var** *e*: **event** *v* を介してプロセス間の交信をしていますが，V Multiple 内の cause (*e*) によって，*e* に対する queue 内の全てのプロセスが起動されて，*v* に対する queue 内に移されます．したがって，*e* をどのように選んで，プロセス間の連絡をつけるかということが問題になります．

システム全体で一つしか *e* を設けなければ，起動すべきプロセスの数はふえてしまいます．また，各腕木変数に一つずつ *e* を設ければ，文献 3) で定義された方法に近くなります．システムにあらわれる PV Multiple があらかじめ全てわかっていれば，どの V Multiple が，どの P Multiple と関係するのかわかりますので，必要にして十分な event をあらかじめ決めることができますが，多重アクセス・システムのように，新しいプロセスがシステムに任意の時点に到着してくる場合には，そのような関連を前もってつけておくことは不可能です．

参　考　文　献

1) 斎藤：OS の基礎理論（1），情報処理，Vol. 15, No. 11, pp. 887～899 (1974).
2) 斎藤：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).
3) L. Presser: Multiprogramming Coordination, Computing Surveys, Vol. 7, No. 1 (1975).

（昭和 50 年 11 月 13 日受付）