

「同期基本命令の公理的定義」の紹介

瀬川 清 (早大)

Martin, Alain J., "An Axiomatic Definition of Synchronization Primitive", Acta Informatica 16, P. 219-235 (1982) の概要を述べる。

1. はじめに

セマフォ演算 P/V のような, 組として用いられる二つの同期基本命令の意味を "boundness", "progress" と "fairness" と呼ばれる三つの公理による定める。また, slack と呼ばれる「自由度」により, 同期基本命令は次の三つに分類できる:

slack $\left\{ \begin{array}{l} \text{無限} \text{ --- 無限のバッファを持つもの (例 } P/V \text{)} \\ \text{有限} \text{ --- 有限のバッファを持つもの (例 } \text{send/receive} \text{)} \\ 0 \text{ --- バッファを持たないもの (例 } CSP \text{)} \end{array} \right.$

2. 二つの動作の同期 (The Synchronization of two Actions)

定義

X action: 命令 X は実行されると, X action をおこす。

sequential computation : action の有限 / 無限列。

concurrent computation : 幾つかの sequential computation から成る。

process : concurrent computation を構成する sequential computation を呼ぶ。

cX : computation が始まってからの, X action の実行個数 (正しくは, 終了個数)。

gX : 現在, 中断されている X action の個数。

tX : $cX + gX$ と定義される。

同期 : 命令の組 (X, Y) に対して, cX と cY の間に自明でない関係があるとき, X action と Y action は同期しているという。

同期基本命令 : X action と Y action が同期していて, $cX - cY$ が上限または下限を持つとき, X と Y は同期基本命令であるという (このことを, 厳密に後で定義する)。

3. 同期基本命令の意味合い (The Semantic Characterization of Synchronization Primitives)

同期基本命令の持つべき性質として、次のようなものが考えられる。そして、これらを同期基本命令の公理とすればよいと思われる。

X と Y が同期しているのならば、 X と Y の実行回数に大きな差はないと考えられるので、

boundness

R1: $cX - cY$ は上限または下限を持つ

という性質があげられる。R1 を満たさなくするよう、命令 X または Y の実行は中断 (suspend) される。次に、中断されっぱなしでは困るので、

progress

R2: 中断中の命令の集合はできる限り小さくする

という性質が欲しくなる。これから、系として次のことがあげられる:

命令の組 (X, Y) に対し、 X が中断されるなら Y は中断されないし、 Y が中断されるなら X は中断されない。

中断中の action を含む process は、その action の所で、delay しているという。X で中断している process があるのに、Y が実行されることなくなると、deadlock が生じる。

一方、 Y が実行されても、中断中のままであるときは、starvationが生じる。starvationは、もちろん困るので、

fairness

$R3$: process が delay しているとき、中断している命令を含む、命令の組の実行回数は有限である

という性質があげられる。

$R1$ と $R2$ を満足するものを、弱い同期基本命令 (weak synchronization primitive) と呼び、さらに $R3$ も満足するものを、強い同期基本命令 (strong synchronization primitive) と呼ぶ。

4. 代数的形式化 (An Algebraic Formulation of the Synchronization Requirements)

3. で述べた三つの性質 ($R1 \sim R3$) を形式化する。

Boundness Requirement

$R1$: kX, kY を整数として

① kX, kY の少なくとも一つは有限

② $-kX \leq cX - cY \leq kY$

系. $kX, kY \geq 0$

Progress Requirement

$$\begin{aligned}
 R2: & (gX=0 \vee cX=cY+kX) \wedge \\
 & (gY=0 \vee cY=cX+kY) \wedge \\
 & (gX=0 \vee gY=0)
 \end{aligned}$$

R2 の注釈

$gX=0 \vee cX=cY+kX$, するわけ, $gX \neq 0$, つまり, X で中断するときは, $cX=cY+kX$. これは, X で中断するのは, $R1$ を満足しなくなるまで待たなければならない, ということを現わす。

$gX=0 \vee gY=0$, するわけ, X と Y の両方から中断することはない, ということを現わす。

Fairness Requirement

process i に対して次のような d_i を考える

$$d_i = \begin{cases} 0 & \text{process } i \text{ は delay 中でない。} \\ m & \text{process } i \text{ が } X \text{ action で delay しているとき,} \\ & \text{その delay 中に (他の process により) 実行} \\ & \text{(終了) された } X \text{ の数。} \end{cases}$$

R2: 各 process i に対して, d_i は有限

次の「弱い同期基本命令に関する定理」が容易に導かれる:

(X, Y) が弱い同期基本命令



$$cX = \min(tX, tY + kX) \wedge$$

$$cY = \min(tY, tX + kY)$$

5. 同期基本命令の三つの型 (The Three Types of Synchronization Primitives)

$K = kX + kY (\geq 0)$ を slack と呼び、その値により、同期基本命令を類別する。

$K = 0$ と $R1, R2$ に代入したものを考える。逆に、それらにより、同期基本命令の一つを定義する。

定義

命令 X と Y は次の二つの条件を満たすとき、バッファ無し通信命令 (unbuffered communication primitive) といい:

$$R1: cX = cY$$

$$R2: \#X = 0 \vee \#Y$$

ここでは $R3$ (fairness) を考えていない。すなわち、弱い同期基本命令の類別も行なっていない。

$K = \infty$ の場合は, $kX < \infty$, $kY = \infty$ とすると,

$$R1: cX \leq cY + kX$$

$$R2: (fX = 0 \vee cX = cY + kX) \wedge fY = 0$$

となり, ここで $s = kX - cX + cY$ とおくと

$$R1: s \geq 0$$

$$R2: (fX = 0 \vee s = 0) \wedge fY = 0$$

となる。 $kY = \infty$ より, $fY = 0$ となるので, 次の定義が得られることになる。

定義

次の条件を満たす P と V をセマフォ命令と呼ぶ:

S : 初期値 S_0 の整変数 (セマフォと呼ぶ)

$$A_0: S \geq 0$$

$$A_1: cP + S = cV + S_0$$

$$A_2: fP = 0 \vee S = 0$$

$K < \infty$ の場合も考えると次の定義が導かれた。

定義

次の条件を満たす P と V を、**互排 P/V 命令** と呼ぶ:

S : 初期値 S_0 のセマフォ

S_m : 正整数

$$B_0: 0 \leq S \leq S_m$$

$$B_1: CP + S = CV + S_0$$

$$B_2: \{ P=0 \vee S=0$$

$$B_3: \{ V=0 \vee S=S_m$$

6. 古典的定理 (Some Classical Theorem)

ここで示されているいくつかの定理の内容は、我々が常識だと思っていることである。そのようなものが、先に定義したもののから導かれるということは、「定義」自身が妥当なものであるということも現わすことになる。

相互排除に関する定理

Theorem. Consider an arbitrary number of concurrent processes, each consisting for what concerns synchronization in a strict alternation of P and V operations, starting with a P and ending with a V , on any of the n ($n > 0$) semaphores s_i , $0 \leq i < n$. If np is the number of processes having completed a P and not yet completed the following V , then $np \leq \sum_i s_{0i}$, where s_{0i} is the initial value of s_i .

Corollary. If $\sum_i s_{0i} = 1$, there is at most one process at a time inside the program part enclosed by a P and the following V .

生産者 / 消費者問題に関する定理

Theorem. Given an arbitrary number of concurrent processes of the "producer" type defined by the program text:

$$*[\dots P(s); \text{PUT}; V(r); \dots]$$

and an arbitrary number of concurrent processes of the "consumer" type defined by the program text:

$$*[\dots P(r); \text{GET}; V(s); \dots],$$

where PUT and GET are arbitrary atomic actions, then:

$$-r_0 \leq c\text{PUT} - c\text{GET} \leq s_0$$

holds, where s_0 and r_0 are the initial values of s and r , respectively.

deadlock に関する定理

Theorem. Given n processes defined by the program text:

$$P(a); P(b); V(a); V(b)$$

and m processes defined by the program text:

$$P(b); P(a); V(b); V(a)$$

with $m+n>0$, the computation is deadlock-free if the following relation holds on the initial values a_0 and b_0 of a and b :

$$a_0 > 0 \wedge b_0 > 0 \wedge (a_0 > n \vee b_0 > m).$$

8. 対称 P / V 命令と生産者 / 消費者問題型命令

生産者 / 消費者問題で次のように置きかえを行なう:

$$t := c\text{PUT} - c\text{GET} + r_0$$

$$P'(t): P(r); \text{GET}; V(s)$$

$$V'(t): P(s); \text{PUT}; V(r)$$

Theorem. If $P'(t)$ and $V'(t)$ are considered as indivisible actions, the following relations are invariantly true:

- C0: $0 \leq t \leq s_0 + r_0$
- C1: $cP' + t = cV' + r_0$
- C2: $qP' = 0 \vee t = 0$
- C3: $qV' = 0 \vee t = s_0 + r_0$.

すなわち、生産者/消費者問題のための同期基本命令を考えると、対応 P/V 命令となる。

9. P/V 命令の実現 (An Implementation of P and V operations)

二進セマフォ系により、一般のセマフォ系を次のように実現すると、先の $A_0 \sim A_2$ を満たすことがわかる。

```

P(s): P(m);
    [s > 0 → s, p := s - 1, p + 1
    □ s = 0 → q := q + 1; V(m); P(z)
    ];
    V(m).
V(s): P(m);
    [q = 0 → s, r := s + 1, r + 1
    □ q > 0 → q, p, r := q - 1, p + 1, r + 1; V(z); P(m)
    ];
    V(m).

```

Initially: $s = s_0, \quad m, z = 1, 0 \quad p, q, r = 0, 0, 0$.

10. 結論

セマフォ命令の公理として、先の $A_0 \sim A_2$ が必要十分であるというのは、仮説であるが、妥当なものであると思われる。