

Communicating Sequential Processes の数学モデル

数理解析研究所 萩野 達也

Communicating Sequential Processes[1] (以下 CSP と略す) の  
 提唱者である C.A.R. Hoare の論文 A Model for Communicating  
 Sequential Processes (On the Construction of Programs, R.M.  
 McKeag and A.M. Macnaghten, Cambridge University Press, 1980  
 pp.229-254) の内容を紹介する。この論文では、CSP で書か  
 れたプログラムの正当性を証明する際の基礎となる、CSP の  
 数学的モデルを与えている。

1. 記法および基本的概念

まず process は、それが関与した event を表わす記号の有  
 限列 (関与した順序に並ぶ) である trace の集合として定義  
 される。process  $P$  に対して  $\bar{P}$  は、 $P$  の関与できる event の  
 全体を表わす。まず、CSP の単純な構文に対しては、

$$\text{ABORT} = \{ \langle \rangle \} \quad (\langle \rangle \text{ は空列})$$

$$c \rightarrow P = \{ \langle \rangle \} \cup \{ \langle c \rangle s \mid s \in P \}$$

$$P \parallel Q = P \cup Q \quad \dagger$$

と定義する。process の定義は、しばしば再帰的に行われる。これは、BNF 文法と同様に取扱う。

## 2. 複合文

2 つの process  $P$  と  $Q$  の複合実行を取り扱う。まず、並列に実行する場合には、互に共通に関与する event は同期をとらないといけないので

$$P \parallel Q = \{ s \in (\bar{P} \cup \bar{Q})^* \mid (s \uparrow \bar{P}) \in P \text{ か } (s \uparrow \bar{Q}) \in Q \}$$

となる。ここで、 $X^*$  は  $X$  の元からなる有限列全体の集合を、 $s \uparrow X$  は列  $s$  より  $X$  以外の元を取除いた列を、それぞれ表わす。

次に、 $P$  の実行の後に  $Q$  を行う場合を考える。そのために、特別な記号  $\checkmark$  を導入する。 $\checkmark$  はすべての process に共通な event であり、process が (successfully に) 停止したことを示す。<sup>†</sup>

$$P ; Q = \{ s \in P \mid s \text{ は } \checkmark \text{ を含まない} \} \cup \{ st \mid s < \checkmark > \in P, t \in Q \}$$

program を小さくする ( $\checkmark$  を program 中に書かない) ために、

process SKIP =  $\{ < >, < \checkmark > \}$  と定義し、 $d \rightarrow \text{SKIP}$  を  $d$  と略する。

## 3. Alphabet Transformation

Alphabet Transformation  $f$  は、記号の変換で、process  $P$  に対して  $f(P)$  は、trace 中の記号すべてに  $f$  を適用してえられ

た (trace をもつ) process を表わす。重要な例として、process の名前付けがある。m と名前付けられた process P は、Alphabet Transformation  $\text{prefix}_m(x) = m.x$  によつて、 $m:P = \text{prefix}_m(P)$  と定義される。

次に、局所的な event を外部から隠すために、

$$[P \parallel Q] = (P \parallel Q) \setminus (P \cap \bar{Q} - \{\sqrt{\quad}\})$$

と定義する。† ここで、 $P \setminus X$  は  $\{s \uparrow (P - X) \mid s \in P\}$  である。

#### 4. 入出力

まずはじめに、名前なしの入出力を定義する。 $!t, ?t$  は、それぞれ t の出力、入力を示す event である。入力に対する program を書きやすくするために、

$$?x:T \rightarrow P_x = \{\langle \rangle\} \cup \{\langle ?t \rangle s \mid t \in T, s \in P_x\}$$

と定義する。T は入力の型で、 $\alpha$  はダミー変数である。

process P の出力をすべて process Q に送る場合には、

$$P \gg Q = [\text{strip}!(P) \parallel \text{strip}?(Q)]$$

となる。ここで、 $\text{strip}!(x)$  は  $\alpha$  が  $!t$  なら  $!$  を取除き  $t$  とする Alphabet Transformation である。 $\text{strip}?$  も同様。これによつて、P の出力と Q の入力は同一 event となり、同期がとられる。

次に、名前付き process の入出力は、同様に、 $m!t(m?t)$

を、 $m$ と名前付けられた process に出力（から入力） $t$ を送る（受取る）ことを表わす event とする。同期をとるために、 $\text{prefix}_m$  の定義に、

$$\text{prefix}_m(?t)=m!t \quad \text{prefix}_m(!t)=m?t$$

$$\text{prefix}_m(n?t)=\text{prefix}_n(m!t)=n.m!t$$

を付け加える。 $n.m!t$  は process  $n$  から process  $m$  へ  $t$  が通信されたことを表わす event である。

## 5. Sharing と例

program を簡単にするために、

$$[\|x:X]P_x = P_u \| P_v \| \dots \quad X=\{u,v,\dots\}$$

と定義する。 $[\|x:X]P_x$  も同様に定義する。<sup>†</sup>

以上のような定義をもとに、この論文では、A Multiprogrammed Batch Processing System の記述例をあげている。これは、複数の card reader からの job 入力を、複数の processor で処理を行い、その結果を複数の line printer に出力する system である。

## 6. 離散事象の Simulation

最後に、離散事象の simulation を行うモデルを取扱う。ここでは、process は現在の時間を知り、たり、ある時間になる



- 2) 再帰的な定義に対して、もうすこし考察する必要があるのではないか。
- 3) process は、途中状態の trace も含む集合として定義されているが、最終結果だけではないのか。それによつて、 $\checkmark$  はいらなくならないのか。
- 4) 名前なし process, 入出力があるために、従来の CSP より使いやすい。さらに、計時 process 等では、ANY という、process 全体の集合をもちいている。
- 5) 離散事象の simulation で、repeat timer?x until x=8 のような busy wait はどうなるのか。
- 6) このモデルでは、通信、同期のみを取扱っており、代入文、if 文、while 文の取扱いはどうするのか。

## 10. 参考文献

- 1) C.A.R.Hoare, Communicating Sequential Processes. CACM 21,8,1978.pp.666-677.
- 2) N.Francez, C.A.R.Hoare, D.J.Lehmann and W.P.de Roever, Semantics of Nondeterminism, Concurrency and Communication. JCSS 19,1979.pp.290-308.
- 3) K.R.Apt, N.Francez and W.P.de Roever, A Proof System for Communicating Sequential Processes. ACM TOPLAS 2,3,1980. pp.359-385.
- 4) G.M.Levin and D.Gries, A Proof Technique for Communicating Sequential Processes. Acta Informatica 15,1981.pp.281-302.