Vol. 27 No. 1 情報処理学会論文誌 Jan. 1986

並行処理のための関数型計算モデル concurrent HFP†

宮地利雄†† 片山卓也††

本論文では，階層的関数型の並行計算モデル concurrent HFP を提案し，これを利用した並行プロセス系の記述を行うとともに，その動作の時制論理による形式化を行った．この計算モデルでは，並行プロセス群が，通信ポートを経由して相互に同期しデータを交換しあう計算木の集まりとして表現される．計算木の節点を構成しているモジュールと呼ぶ抽象構造体や通信ポートはそれぞれ決められた属性の集合をもち，各属性には計算過程の中で値が割り当てられる．計算全体は，並行して行われる計算木の成長，属性値の評価，ランデブーの3種類の動作を通して遂行される．計算木を成長させ属性値を評価する方法は展開規則により定義されている．属性値の評価は，副作用をもたない純関数のみによってデータ駆動原理に従って進められる．さらに，並行処理システムがもつ非決定的動作や同期を関数システムの性質と調和させながら記述するために，同期条件および順序条件と呼んでいる規則を伴った通信ポート間のランデブーの概念が導入されている．

1. まえがき

制御の流れに基づいて実行が行われる従来型の言語と比較し，関数型言語を用いたプログラミングでは，データの流れを中心に計算が進むため，記述の自然さ，検証の容易さ，また並列に実行できる部分を検出しやすいことから並列処理計算機上で高速に実行できる可能性が注目されている．しかしながら，データの到着を待ち合わせることが唯一の同期方法である純粋な関数型言語では，並列処理システム内で不可避的に現れる共有資源管理などに伴った同期を自然に記述することができないことから適用可能範囲が著しく限定されてしまう．

本論文では，属性文法を基礎とした階層的関数型言語として提案された HFP[1] によるモジュール階層を中核とし，並列処理システムの記述を目的として，同期のための基本機構をこれに付加して拡張した関数型計算モデル concurrent HFP（以降 cHFP と記す）の提案を行う．この計算モデルでは，並行プロセス群が，通信ポートを経由して相互に同期しデータを交換しあう計算木の集まりとして表現される．計算木の節点を構成しているモジュールと呼ぶ抽象構造体や通信ポートはそれぞれ決められた属性の集合をもち，各属性には計算過程の中で値が割り当てられる．計算全体は，並行して行われる計算木の成長，属性値の評価，ランデブーの3種類の動作を通して実現される．計算木を成長させ属性値を評価する方法は展開規則により定義されている．属性値の評価は，副作用をもたない純関数のみによってデータ駆動原理に従って進められる．さらに，並行処理システムがもつ非決定的動作や同期を関数システムの性質と調和させながら記述するために，同期条件および順序条件と呼んでいる規則を伴った通信ポート間のランデブーの概念が導入されている．

本稿の以下では，次章で提案する計算モデルの構造と直感的な意味を与え，続く3章でこれを用いた並行システムの表現例を掲げる．さらに，4章では本モデルの動作の時制論理を用いた形式化を行う．なお，本稿では，提案する計算モデルの動作機構を正確に述べることを主眼としており，記述言語としての構文上の洗練や便法についてはあえて考慮していない．

2. 計算モデル

2.1 定　義

われわれの計算モデルは，属性文法を基礎とした階層的関数型言語として提案された HFP を拡張したもので，次の四つ組として与えられる：

(M, C, P, E_r)

ここで，M はモジュールの集合，C は通信ポートの集合である．C の各要素は対をなしており，$c \in C$ と対になっているものを $\bar{c} \in C$ により表す．M と C の各要素 n には，それぞれ入力属性の集合 $I(n)$ と出力属性の集合 $O(n)$ とが与えられている．入力属性および出力属性は，通常のプログラム言語の変数に相当するもので単に属性とも呼ばれ，E_r 中の属性値定義に基づいて計算過程のなかで値が，属性のインスタンスのそれぞれに対して結合される．ただし，$I(n) \cap O(n) = \phi$．また，通信ポート $c, \bar{c} \in C$ については，$I(c) =$

† Concurrent HFP: A Functional Computation Model for Parallel Processing by TOSHIO MIYACHI and TAKUYA KATAYAMA (Department of Computer Science, Faculty of Engineering, Tokyo Institute of Technology).

†† 東京工業大学工学部情報工学科

$O(\bar{c})$ かつ $O(c)=I(\bar{c})$ である.

P はプロセス集合で，各プロセス $p\in P$ に対して初期モジュール $i_p\in\{m|m\in M, I(m)=\phi\}$ が1個ずつ決められている．各プロセスは，計算の進行に伴い E_r に従って，初期モジュール i_p を根とし，モジュールや通信ポートのインスタンスを節点とする計算木を構成する.

E_r は展開規則の集合であり，個々の展開規則の一般形式は次のとおりである：

$X_0\rightarrow X_1,\cdots,X_t$; 属性値定義

when 展開条件　　**synch** 同期条件

order 順序条件

ここで，$X_0\in M$ を展開条件の左辺と呼び，$M\cup C$ の要素からなるリスト（同じ要素の重複も，また空のリストも許される）$X_1,\cdots,X_t$ を展開規則の右辺と呼ぶことにしよう．また，$X_0, X_1,\cdots,X_t$ は同じ名前の複数の出現を適宜に添字をつけて区別し，それぞれをモジュールまたは通信ポートの出現 (occurence) と呼ぶ.

属性値定義は，$I(X_0)$ と $O(X_i)$ に属する属性の値と関数を用いて $O(X_0)$ と $I(X_i)$ に属するそれぞれの属性の値を決める定義式の集まりである（ただし，$1\leqq i\leqq t$)．「代入」と異なり，われわれのモデルでは属性値は一度定義されるとその後は別の値に再定義されることがなく，また定義式中に現れる関数には副作用をもたない純粋な関数のみを許しているので，属性値の評価順序が直接に計算結果に影響を及ぼすことがない.

展開条件はその展開規則左辺の入力属性集合 $I(X_0)$ に属する属性上の述語であって，これが真になるときに限って，その展開規則を適用することができる.

同期条件は通信ポートの集合 $\{X_1,\cdots,X_t\}\cap C$ の部分集合であって，ここで指定された通信ポートがインスタンスの生成後ただちにそれぞれ対をなす通信ポートと結合して属性値の交換を行うこと（ランデブー）が可能であるときのみに，その展開条件の適用を限っている.

順序条件は，$\{X_1,\cdots,X_t\}$ 上の半順序関係であって，この展開規則の適用によって新しく作られる子節点を根とする部分木内の通信ポートのランデブーの時間的な順序を規定している.

2.2 動　　作

われわれの計算モデルにおける計算の過程は，計算木の展開，属性値の評価，通信ポート間のランデブーの3種類の動作を軸として進行する.

（a） 計算木の展開

計算木の形成は，モジュール，通信ポート，初期モジュールをそれぞれ非終端記号，終端記号，開始記号とみなし，展開規則の $X_0\rightarrow X_1,\cdots,X_t$ を生成規則とみなしたときの導出木の形成と同様の方法で行われる．すなわち，計算木の葉の位置にあるモジュールは，左辺がそのモジュール名と一致しているような展開規則が存在するときには，その右辺を構成しているモジュールや通信ポートの新しいインスタンスを作り，自身の新しい子節点として計算木に加えることを繰り返しながら初期モジュールから始めて計算木を成長させていく．しかしながら，導出木の場合と異なり，適用する展開規則を任意に選ぶことができるのではなく，展開条件と同期条件をともに満足していることが計算木を拡大するための必要条件とされる．条件を満足する展開規則が複数個存在する場合には，そのうちの一つを非決定的に選ぶ．展開条件が，その左辺のモジュールの入力属性の値により計算木の展開を制御するプロセス内的な条件であるのに対し，同期条件は，他のプロセスの通信ポートとの相互関係により決定されるプロセス間的な条件である.

（b） 属性値の評価

計算木の展開の際に作られたモジュールや通信ポートの新しいインスタンスの上のすべての属性は，値が初め未定義状態になっている．そして，展開後，展開規則の中の属性値定義に従って各属性の値が評価されていく．この評価過程は，副作用なしに，完全にデータ駆動的にすすめられる.

Id[2] や Concurrent Prolog[3] では並行プロセス間の交信を stream を利用して表現しているために本質的に stream 型変数を必要としているが，本モデルにおいては，構造をもったデータであるにせよ属性値は単一の値であって，計算の進行に伴って部分の値が定まっていく構造値である stream 型の取り扱いは考えていない.

（c） ランデブー

異なるプロセスの計算木に属する1組の対をなす通信ポートのインスタンスが結合し，その属性値の交換を行う動作を「ランデブー」と呼ぶ．本モデルにおけるプロセス間の交信はすべてランデブーとして表現される．各通信ポートはそれぞれ1回限りのランデブーに関与するのみで，ランデブーをいったん開始したものが，再び別の組み合わせを作ってランデブーするこ

とがない.

通信ポートのインスタンスを作り出した展開規則の同期条件の指定に含まれているような通信ポートでは，展開によりインスタンスが作られるときに，それ以外の通信ポートではインスタンスが作られた適当な時間の後に対をなす通信ポートとの対応関係が成立したときにランデブーを開始する．その後，属性名の対応により二つの通信ポートの間で値の受け渡しが行われ，通信ポートの上の属性の値がすべて定義されたときにランデブーが完了する．

属性値の評価がデータの依存関係のみにより制御されているのに対し，ランデブー動作は実行順序が展開規則の順序条件をもとに生成される半順序関係により拘束されている．すなわち，展開規則 $e \equiv E_r$ の順序条件で指定された半順序関係を $<_e$ で表すとき，計算木内の通信ポートのインスタンスの集合の上の半順序関係 $<$ が次式により定義される：

$$\text{for all } c_1, c_2 \in C$$
$$c_1 < c_2 \equiv$$
$$\exists e \in E_r,\ \exists n_1, n_2 \in M \cup C:$$
$$\left[\begin{array}{l} c_1 \in \textbf{subtree}(n_1) \wedge \\ c_2 \in \textbf{subtree}(n_2) \wedge \\ n_1 <_e n_2 \end{array}\right]$$

ここで，**subtree**(n) は節点 n を根とする部分木を表す．そして，$c_1 < c_2$ であるときには，c_1 のランデブーの完了以前に c_2 のランデブーが開始されないという時間的順序関係を要請している．

3. 応　用　例

この章では cHFP により代表的な並行処理の問題について具体的な記述を行い，cHPF の表現能力を示すとともに，その動作の概略の直感的理解を得るための助けとする．

3.1 生産者消費者問題

最初に簡単な例題として，100 以下の自然数とそれに対する階乗の値を順に生成し通信ポートを経由して他のプロセスにこれを供給するプロセスを例として次に掲げる．

［例 1］ 階乗の生産者消費者問題

$M = \{E, F, C\}$
$I(F) = \{n, f\}$
$C = \{P, \bar{P}\}$
$I(P) = O(\bar{P}) = \{n, f\}$
$P = \{\text{Prod, Cons}\}$

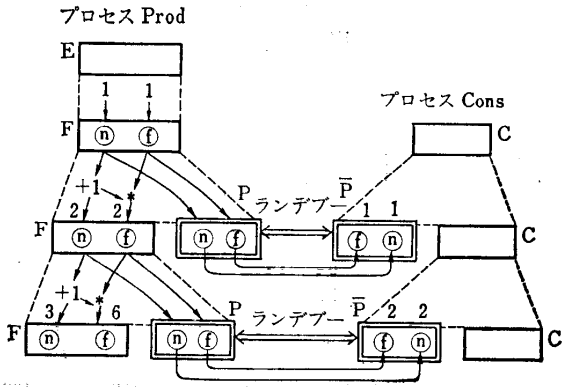

図 1　階乗の生産者・消費者問題に対する計算木
Fig. 1　Computation tree for factorial producer-consumer problem.

initial-module (Prod)$=E$
initial-module (Cons)$=C$

E_r :
$E \rightarrow F$; $n.F=1, f.F=1.$
$F_0 \rightarrow P, F_1$; $n.F_1 = n.F_0 + 1,$
$f.F_1 = f.F_0 * (n.F_0+1),$
$n.p = n.F_0,$
$f.p = f.F_0$
when $n.F_0 < 100$
order $\{P < F_1\}$
$C_0 \rightarrow \bar{P}, C_1$; **synch** $\{\bar{P}\}$

Prod と Cons がそれぞれ階乗の値の生産者と消費者を表すプロセスで両者が通信ポート P と $\bar{P}$ とを経由してその上の属性 n と f によりデータ授受を行う．動作の様子を示すために計算木のスナップショットを**図 1** に示す．図中では，通信ポートを二重の四角形で，ランデブーの組み合わせを二重線の矢印により示す．

なお，2 番目の展開規則中の順序条件 **order** $\{P < F_1\}$ を省略した場合には生成された値が供給される順序が定義されないことになり，またその順序条件を同期条件 **synch** $\{P\}$ に置き換えたときには，プロセス Prod の計算木の成長がランデブーと歩調を合わせ進行するようなシステムが表現される．

3.2 容量制限付きバッファ

次の例題は，容量制限付きバッファの動作を記述する問題である．バッファはプロセス Buffer として表現され，2 種類の通信ポート $\overline{GET}, \overline{PUT}$ を備えている．バッファを利用するプロセスは下の記述の中では明示していないが，通信ポート PUT, GET によるランデブーによりバッファからのデータの取り出し，バ

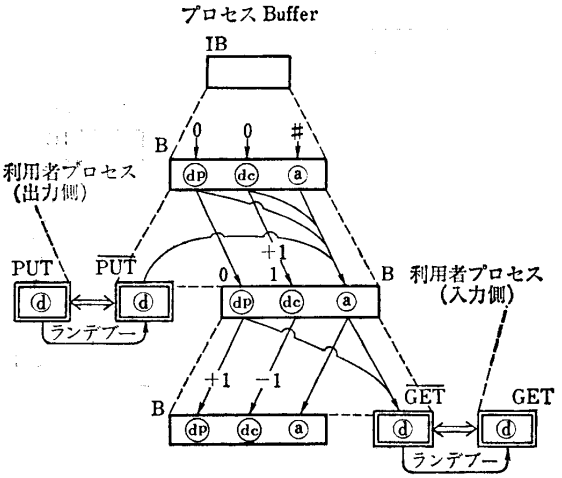

図 2 容量制限つきバッファ問題に対する計算木
Fig. 2 Computation tree for bounded buffer problem.

ッファへのデータの出力をそれぞれ行うことができる．次に記述例を掲げ，**図 2** に計算木のスナップショットを示す．

［例 2］ 容量制限付きバッファ

$M=\{IB, B, \cdots\}$

$I(B)=\{a, d_p, d_c\}\cdots$

$C=\{GET, \overline{GET}, PUT, \overline{PUT}\}$

$I(\overline{GET})=O(GET)=\{d\}$

$O(\overline{PUT})=I(PUT)=\{d\}$

$P=\{\text{Buffer}, \cdots\}$

initial-module (Buffer)$=IB$

.....

E_r :

$IB\rightarrow B$; $d_p.B=0$, $d_c.B=0$, $a.B=\#$.

$B_0\rightarrow\overline{GET}, B_1$; $d.\overline{GET}=a.B_0$ $[d_p.B_0]$,
$d_p.B_1=d_p.B_0+1$ **mod** N,
$d_c.B_1=d_c.B_0-1$,
$a.B_1=a.B_0$
when $d_c.B>0$ **synch** $\{\overline{GET}\}$

$B_0\rightarrow PUT, B_1$; $d_c.B_1=d_c.B_0+1$,
$d_p.B_1=d_p.B_0$,
$a.B_1=a.B_0[(d_p.B_0+d_c.B_0+1$ **mod** $N)\backslash d.\overline{PUT}]$
when $d_c.B<N$ **synch** $\{\overline{PUT}\}$

バッファは，配列で実現された N 個のエントリからなる環状バッファとして実現されていて，バッファが空または空き領域がないことに伴う同期は，展開条件 **when** $d_c.B>0$, **when** $d_c.B<N$ によりそれぞれ表現されている．なお，属性 a のデータ型は，配列 **array** $[0\cdots N-1]$ であり，$a[i]$ で i 番目の要素を，$a[i\backslash x]$ で配列 a の i 番目の要素の値だけを x で置き換えて得られる配列を表している．また，# は初期値として与えられる配列型の任意の値を示すために使われている．

3.3 Partitioning Sets

次に扱う問題は，整数からなる N 個の集合 $\{S_n\}$ が与えられたとき，$\bigcup_{n=1}^{N} S_n$ を $(i\leqq j\wedge d_i\in S_i'\wedge d_j\in S_j'\supset d_i\leqq d_j$ および $|S_n|=|S_n'|$ を満足する N 個の部分集合 $\{S_n'\}$ に分割する問題である．この問題と解は $N=2$ の場合について Dijkstra[4] により取り扱われたものである．

整数の各集合 S_n は各プロセス P_n に一つずつ分配され，各プロセスは自分がもっている整数集合 S_n 中の最小および最大の整数を，問題に与えられた条件が満足されるようになるまでそれぞれ隣接するプロセス P_{n-1}, P_{n+1} と交換しあうことによって解を得る．

$M=\{IM_n, M_n \mid 0\leqq n\leqq N\}$

$I(M_n)=\{s, c_l, c_r\}$

$C=\{E_n, \overline{E_n} \mid 0\leqq n\leqq N\}$

$I(E_n)=O(\overline{E_n})=\{m_r, c_r\}$

$O(E_n)=I(\overline{E_n})=\{m_l, c_l\}$

$P=\{P_n \mid 0\leqq n\leqq N\}$

initial-module $(P_n)=IM_n$

E_r :

$0\leqq n\leqq N$ に対して

$IM_n\rightarrow M_n$; $s.M_n=S_n$,
$c_l.M_n=(n=0)$,
$c_r.M_n=(n=N)$.

$0\leqq n<N$ に対して

$M_{n_0}\rightarrow M_{n_1}, E_n$;
$m_l.E_n=\max(s.M_{n_0})$,
$s.M_{n_1}=s.M_{n_0}-\{m_l.E_n\}\cup m_r.E_n$
$c_l.M_{n_1}=c_l.E_n$
$=c_l.M_{n_1}$ **and** $(m_r.E_n=m_l.\overline{E_n})$ **or** $N=0$
$c_r.M_{n_1}=c_l.E_n$ **and** $(m_l.E_n=m_r.E_n)$.
when not $c_r.M_{n_0}$

$0<n\leqq N$ に対して

$M_{n_0}\rightarrow\overline{E_n}, M_{n_1}$
$m_r.\overline{E_n}=\max(s.M_{n_0}\cup m_l.\overline{E_n})$,
$s.M_{n_1}=s.M_{n_0}\cup m_l.\overline{E_n}-m_r.\overline{E_n}$,

$c_l.M_{n_1}=c_r.\overline{E_n}$ **and** $(m_l.\overline{E_n}=m_r.\overline{E_n})$,
$c_r.M_{n_1}=c_r.\overline{E_n}$
$=c_r.M_{n_0}$ **and** $(m_l.\overline{E_n}=m_r.\overline{E_n})$
or $n=N$
when not $c_l.M_{n_0}$

4. 動作の形式化

本章では，概略を直感的に2章で述べた cHFP の動作について，より精密な定義を与えよう．並行システムの動的な振るまいを詳細に論ずるための道具として知られているものの中から本稿では時制論理を使う．本モデルは，データ値の操作に関しては純粋に関数的であって，いったん定義された属性値が後になって変更されるようなことがなく，属性値評価がデータ駆動的に行われることから，次のように比較的容易に時制論理を用いた動作の形式化を行うことができる．

ここでの形式化に際して利用される時制論理は，通常の1階述語論理に時制演算子として□と○の2種類を加えたもので，線形時間論理のクラスに属している．線形時間論理において時間は枝分かれのない一次元的な流れとして考えられる．論理式 p に対して，□p により現在（すなわち注目している時点）以降のすべての時間において p が成立することを，○p により現在の直後の時点で p が成立することを表す．また，単に p と書かれたときには現在において p が成立するものと解釈される．以上直感的に述べた内容は，公理および推論規則として次のように表現される．

L_1. A，ただし A は1階述語論理における定理
L_2. ○$(A\supset B)\supset(\bigcirc A\supset\bigcirc B)$
L_3. $\Box A\supset\bigcirc\Box A$
L_4. $\Box A\supset A$
L_5. $\Box(A\supset\bigcirc A)\supset\Box A$
L_6. □A，ただし A は公理

RMP. $\dfrac{A,\ A\supset B}{B}$

与えられた cHFP の記述の意味は，その cHFP 記述に対応する完全で無矛盾な公理を追加することによって行われる．すなわち，与えられた cHFP の動作についての表明が正しいことが，時制論理の公理 L_1〜L_6 と推論規則および追加された cHFP の公理とを用いてその表明を証明できることと同値になるような体系を与えるのである．本稿では，任意に与えられた cHFP の記述に対応して追加すべき公理群を与える規則を定めることによって cHFP の意味を一般的かつ形式的に定義する．

まず初めに，表記法をいくつか導入しておく．プロセス名の後ろに自然数の列をつなげた $w\in P\cdot N^*$（N は自然数の集合）によって計算木の節点を識別する．

$X\in M\cup C$ のとき，節点 w が X のインスタンスであることを exist (X,w)，節点 w 上の X の各属性の値が値割り当て関数 A で与えられるものとなっていることを $X(A,w)$ と記す．なお，属性値の割り当てを表すために，属性集合と同じ記号を用いる．

このとき，任意に与えられた cHFP の記述に対応して追加される公理を以下に示す．

[プロセスの初期モジュールの公理] 各 $p\in P$ に対して，それぞれの初期モジュールを $i_p\in M$ と書くとき：

I. exist (i_p, p).

[計算木展開の公理] 各 $e_r\in E_r$ に対して，e_r が次の形式 $X_0\to X_1,\cdots,X_t\ ;\ AD_1,\cdots,AD_s$ **when** $P(A_0)$ **synch** S_y **order** S_q であるとき：

E. $\forall A_0,A_1,\cdots,A_t,w$:
([exist (X_0,w)
$\wedge$ $(X_0(A_0,w)\ \wedge\ P(A_0))$
$\wedge$ [$\forall k$ ([$X_k\in S_y$
$\wedge$ exist $(X_k,w\cdot k)$]
$\supset\exists w'$: rendez $(X_k,w\cdot k,w')$)]]

$\supset$ [exist $(X_1\cdot 1)\ \wedge\ \cdots$
$\cdots\ \wedge$ exist $(X_t,w\cdot t)\ \wedge$
$\bigwedge_{i=1}^{S}$ ([$\bigwedge_{(AD_i\cap A_j)\neq\phi}$ $X_j(A_j,w\cdot j)$]$\supset$ AD_i)
$\wedge\ \forall\ X_mX_n$
[$(X_m,X_n)\in S_q\supset$
$\forall\ w_1w_2\in N^*$:
[$w\cdot m|w_1<w\cdot n|w_2$]]])

上の公理では，2行めでモジュールの存在を，3行目で分割条件の充足を，4〜6行めで同期条件の充足を掲げ，以上を条件として，7行めで新たなモジュールや通信ポートのインスタンスを付加して計算木が成長することを，8〜9行めで属性値の関係を，11行め以降では順序条件をもとに後出の ready において参照される関係“<”を与えている．なお A_i は X_i の属性集合に対する値の割り当てを，$AD_i\cap A_j$ は属性集合のうち属性定義式 AD_i に現れるものの集合を，

$u \cdot a$ は列 u の最後に a を付加した列を，$u|v$ は列 u と v とを結合して得られる列を表す．これまでに掲げた二つの公理ⅠとⅡは，cHFP の動作から論理式への忠実な直訳になっている．

1組の通信ポート，節点 w_1 上の C との節点 w_2 上の $\bar{C}$ とが，ランデブーの状態にあることを couple (C, w_1, w_2) で表し，現在はランデブーの状態にないが直後の時点ではランデブーの状態に移ることを rendez (C, w_1, w_2) で表す．また，通信ポートに関する二つの述語 ready と candid を次のように定義する：

ready $(C, w) \equiv$ exist (C, w)
$\wedge\ \forall\ w'\ [\sim$couple $(C, w, w')]$
$\wedge\ \vee\ C'\ w'\ [(w' < w \wedge$ exist $(C', w'))$
$\supset\ \exists\ w''$: couple $(C', w', w'')]$

candid $(C, w_1, w_2) \equiv$
ready $(C, w_1) \wedge$ ready $(\bar{C}, w_2)$

直感的には，ready (C, w) により節点 w 上の通信ポート C が新たにランデブーできる状態にあること，すなわち，w 上に C が存在しており（第1項），現時点でランデブー状態になく（第2項），順序条件を満足している（第3項）ことを表し，また，candid (C, w_1, w_2) により節点 w_1 上の C と節点 w_2 上の $\bar{C}$ とが新たにランデブーに入ることができる通信ポート対であることを表している．

以上の準備の上で，ランデブー一般に関する公理として次のものを加える．

[ランデブーの公理]

R_1. $\forall\ C\ w_1\ w_2$ [rendez (C, w_1, w_2)
$\supset$ candid (C, w_1, w_2)]

R_2. [$\exists\ C\ w_1\ w_2$: candid (C, w_1, w_2)]
$\supset$ [$\exists\ C\ w_1\ w_2$: rendez (C, w_1, w_2)]

R_3. $\forall\ C\ w\ w_1\ w_2$:
([rendez (C, w, w_1)
$\wedge$ rendez (C, w, w_2)]
$\supset\ w_1 = w_2$)

R_4. $\forall\ C\ w_1\ w_2$:
[rendez $(C, w_1, w_2) \supset$
rendez $(\bar{C}, w_2, w_1)$]

R_5. $\forall\ C\ w_1\ w_2$ [rendez $(C, w_1, w_2) \supset$
$\bigcirc\ \Box$ couple (C, w_1, w_2)]

R_6. $\forall\ C\ w_1\ w_2$ [couple $(C, w_1, w_2) \supset$
$\exists\ A : (C(A, w_1) \wedge \bar{C}(A, w_2))$]

ただし，A は C の属性集合に対する値の割り当てを表す．R_1 はランデブーの必要条件（直後にランデブーに入る（rendez）ためには candid が真であることが必要）を，R_2 は必然性（candid が真である通信ポート対が存在すれば，直後にランデブーに入る（rendez）少なくとも1対の通信ポートが存在すること）を，R_3 は競合の回避，すなわち，どの通信ポートもそれぞれ唯一つの通信ポートとしかランデブーしえないことを，R_4 はランデブーの対称性を，R_5 は結合の確立（rendez が真であれば，その直後の時点以降ランデブーの状態が持続すること）を，R_6 はランデブーによる結合対の間の属性値の交換を表現した公理となっている．

5. まとめ

階層的関数型の並行計算モデル cHFP を提案し，これを利用した並行プロセス系の記述を行うとともに，その動作の形式化を行った．本モデルは，値について純粋に関数的な動作を行うとともに，並行プロセスの記述に必須の同期機構としてランデブーを導入している点に特徴がある．関数型言語の枠組の中で並行処理を記述する従来の試みでは[2),3)]，時間的に次々に処理されるデータの流れを stream 型のデータ値として抽象化し，その上の merge をはじめとする基本演算子を導入して並列処理システムの動作を表現したものがほとんどであった．そのようなアプローチは繰り返しを主とする数値計算システムの記述においては直接的で自然な方法と言うことができるが，われわれのモデルと比較するとき，より広い応用に対して同期を制御する能力が十分でなく，同期制御が値の評価と複雑にからみあうという欠点をもっている．

なお，提案した記述法をプログラミング言語として見るときわめて使いにくいものとして評価されるかもしれないが，これは本稿ではモデルの基本機能に焦点を当てるために原形に近い形式での記法を用いているためである．この点に関しては今後の研究課題であると考えている．

参考文献

1) Katayama, T.: HFP: A Hierarchical and Functional Programming Based on Attribute Grammar, 5th International Conference on Software Engineering, p. 343 (1981).
2) Kim, A., Gostelow, P. and Plouffe, W.: The (Preliminary) Id Report: An Asynchronous Programming Language and Computing Machine, Tech. Rep. 114 Dept. of Inf. and Comp. Science, Univ. of California, Irvine (1978).

3) Shapiro, E. V.: A Subset of Concurrent Prolog and Its Interpreter, ICOT Tech. Rep. TR-003 (1982).
4) Dijkstra, E. W.: A Correctness Proof for Communicating Processes—A Small Exercise, EWD-607, Burroughs, Nuenen, The Netherlands (1977).

(昭和59年1月10日受付)
(昭和60年7月18日採録)