ソフトウェア基礎論　２５－４
（1988. 5. 26）

無限プロセスを含む並列論理型プログラムの宣言的意味論

A Declarative Semantics of Parallel Logic Programs with Perpetual Processes

村上　昌己
Masaki MURAKAMI

（財）新世代コンピュータ技術開発機構研究所
Institute for New Generation Computer Technology,

概要：flatＧＨＣのようなHorn論理に基づく並列言語の宣言的意味論について述べる。本論文では通常の純Horn論理型プログラムの宣言的セマンティクスにおける Herbrand 基底に代わって、入出力履歴（I/O history:I/O 履歴)の領域を導入し，プログラムの表示（denotation）は、I/O 履歴の集合により与えられる。ゴール節及びコミット・オペレーターを含むHorn節の集合について真／偽の概念を導入し、プログラムの意味はその節の集合を真とするI/O 履歴の集合として定義される。さらにこのように定義されたセマンティクスが、プログラムの構文から定まる関数の最大不動点によって特徴づけられることを示す。このアプローチによりガード／コミット機構によって制御されながら無限に計算を続けるプロセスを含むプログラムの性質の議論が可能となる。

ABSTRACT: A declarative semantics of a parallel programming language bassed on Horn logic such as flat GHC is presented. The domain of input/output histories (I/O histories) is presented. The denotation of a program is defined as a set of I/O histories. The notion of truth is re-defined for goal clauses and sets of guarded clauses. The semantics of a program is defined as the maximum model of the program. We also show that the semantics is characterized as gratest fixedpoint of the function obtained from the program. The properties of programs which contain perpetual computation controlled by guard/commit mechanisms can be discussed using the semantics.

１．はじめに

近年、Horn論理に基づく並列プログラミング言語としてPARLOG [Clark 86]、 Concurrent Prolog [Shapiro 86]、ＧＨＣ [Ueda 88]などが提案されている。これらの言語では、入出力用のストリームを用いて通信を行いながら無限に計算を続けるプロセスを自然に表現することができる。このようなプログラム言語のセマンティクスについての形式的な議論の方法については、いくつかの結果が報告されている。[Takeuchi 86, Saraswat 85, 87, Ueda 86, Shibayama 87] これらの結果は、いずれも操作的アプローチに基づいたものである。ゆえにこれらの結果はどちらかといえば、処理系の形式的な仕様といった趣のものである。実際プログラムの検証、変換などの手法の論理的な基礎としては何らかの宣言的な手法によるセマンティクスが与えられることが求められる。

純Horn論理型プログラミング言語の場合、[Apt 82, Lloyd 84]らによる宣言的なセマンティクスが既に報告されている。このアプローチではプログラムのセマンティクスはそのプログラムを記述する節集合の最小モデル，すなわちプログラム自身と`等価`な単位節の集合によって与え

られる。また，このような単位節の集合はプログラム節の集合から定まる関数の最小不動点によって与えられる。このアプローチによってプログラムの実行メカニズムとは独立に，プログラムの論理的帰結として導かれるものを解の集合として特徴づけることができる。したがって，このようなアプローチは論理プログラムの明解さを生かすにはもっとも適したアプローチといえる。また領域を完備Herbrand空間に拡張することにより，有限時間で停止しないような計算の定式化への拡張なども試みられてる。[Lloyd 84, Sakakibara 85]

しかしながらこれら結果は純Horn論理型プログラムについてのものであり，このままではガード／コミット機構をもつような並列プログラミング言語に適用することはできない。すなわち，ガード／コミット機構を用いてのゴールの反駁において，（コミット記号を論理積とみなした場合）解としてえられるものは節集合の論理的帰結になってはいるが，論理的帰結となるものがすべて得られる可能性があるわけではない。すなわちコミット記号を含む節集合のセマンティクスとしては，最小モデルはその手続き的な解集合に比べて大き過ぎる。また[Falaschi 87] で指摘された通り，全て基底原子式で特徴づけるという方法も適切ではない。したがって，プログラムの計算結果を論理的帰結として議論するためには，flat ＧＨＣの計算規則のようなコミット記号を含む節を用いたゴールの反駁が証明手続きとしてある意味で完全かつ無矛盾とするようなモデル理論を再構築することが必要となる。

そこでこのアプローチの並列言語への拡張が[Levi 87 88, Falaschi 88]等で試みられた。[Levi 88]ではflat ＧＨＣプログラムのセマンティクスを，ＧＨＣにおける単位節にあたるガード付原子式の集合によって議論する方法が報告されている。ガード付原子式とは例えば次のような形をしたガード付節である。

$$p(X, Y) :- X = \tau \mid Y = \tau'$$

この意味は直観的には，ＧＨＣプログラムのガード付節としての意味に等しい。すなわち，$p(\tau, Y)$という形をしたゴールが実行されると，$Y = \tau'$という解代入が得られる事を意味する。これによって従来の純Horn論理型プログラムで用いられた宣言的意味論がコミット機構を含むプログラムの変数を含むゴールについての動作を議論することが可能になった。例えば次のようなプログラムについて：

```
p(X, Y) :- X = [X1|Y1] | q(X1, Y1, Y).
q(X, Y, Z) :- true | Z = [X|X1], r(Y, X1).
r(X, Y) :- X = [X1|Y1] | Y = [X1].
s(X, Y) :- X = [X1|Y1] | Y = [Yb|Y2], h(Yb).
h(Y) :- true | Y = b.
t(X, Y) :- true | p([A|X], Y), h1(A), s(Y, X)
h1(A) :- true | A = a.
```

このプログラムのdenotationのうち，q, p, tに関するものは次のようなものとなる。

```
q(X, Y, Z) :- Y = [X1|Y1] | Z = [X|X1].
p(X, Y) :- X = [X1, X2|Y1] | Y = [X1|X2].
t(X, Y) :- true | Y = [a|b], X = [b|Y1]
```

しかしながら，ここではガード付原子式はゴールが成功した時に得られる出力代入と入力代入の関係を記述しているのみである。[Takeuchi 86]に報告されているように，このような関係のみを用いて入出力関係だけでは動作を記述することが困難であり，計算の中間結果をも用いることが必要となる例が存在する。例えば，この方法では依然として次のような二つのプログラムp1, p2を区別することはできない。

```
p1 :  p(X, Y) :- X = [A|X] |
                     Y = [A|Z1], pp(X1, Z1)
      pp(X, Y) :- X = [A|X1] |
                     Y = [A|N], nil(N).

p2 :  p(X, Y) :- X = [A|X1] | p2(X1, Y, A).
      p2(X, Y, A) :- X = [B|X1] |
                     Y = [A|Y1], p22(B, Y1).
      p22(B1, Y1) :- true | Y1 = [B1|N], nil(N).

      nil(N) :- true | N = nil.
```

この二つのプログラムは，pについてガード付原子式を用いてdenotationを与える限り同じものである。しかし，両者は出力を具体化するタイミングが異なる。このようなプログラムが非決定的なプログラムと並列に実行された場合に，どちらのプログラムを用いるかによって全体の動作が異なるものになる可能性があることは，[Brock, 81, Takeuchi 86] 等によって指摘されている。このようにflatＧＨＣプログラムの動作を十分に記述するためにはプログラムの実行途中の振舞について必要にして十分な情報を記述しうるような記法を用いたセマンティクスが必要となる。さらに，無限に計算が続くプロセスの動作を議論するのは，ゴールが成功したときの得られる代入によって議論するのは不可能である。無限に計算が続くプロセスを含むようなプログラム（例えば素数をつぎつぎとあるストリームに出力するようなプログラム）の場合には，実行によって外部からなんらかの計算結果が観測できるのは，正に計算の途中の振舞からであり，この意味においてはプログラムの実行途中の（外部から観測できる）振舞を記述できるセマンティクスこそが望むべきものである。

本論文では，flatＧＨＣ風の言語で記述された無限プロセスを含むようなプログラムに対して宣言的な，すなわち純Horn節におけるモデル論的セマンティクスの拡張としてのセマンティクスを与える。すなわち通常の単位節の概念に代わって、入出力履歴（input /output history: I/O 履歴)の概念を導入する。I/O 履歴の集合はHerbrand基底に対応する。またI/O 履歴はガード付原子式の拡張である。すなわち，I/O 履歴はあるプログラムがデッドロックも失敗もせずに計算が進んだときにたどるpath （または木）を外側から観測したものになっている。ここではさらにゴールまたはプログラムが（宣言的に）真であるということをI/O 履歴の概念をもとに再定義する。プログラムのセマンティクスはそのプログラムを真とするI/O 履歴の最大の集合として与えられる。ここで導入するゴール節の真／偽の概念は，即反駁の成功／不成功には対応しない。例えば，無限に実行を続けるプロセスを起動するゴール節は正常な動作を続けるものであるかぎり真とされる。また始めからゴールが十分に具体化されて起動されるのではなく，実行途中に外部から（例えば他のプロセス）から入力を受け取って実行を継続するようなプロセスを表わすゴールは，途中でサスペンドするものであっても，その原因が入力待ちであるような場合には真とみなされる。

さらにここではこのように定義されたプログラムのdenotationが，プログラムの構文から定まる関数の最大不動点によって特徴付けられることがしめされる。

2．ガード付ストリーム

本節では，I/O 履歴 を構成する基本的な成分であるガード付ストリームについてのべる。まず準備として幾つかの基本的な定義を示す。本稿では議論を簡単にするために，{a, b} という領域の上のリストを扱うプログラムに対象を限定する。

[定義 1] Var を可算無限個の変数の集合，Fun = {a, b, nil, cons} を関数記号の集合とする。Funの各要素に対してarityを a, b, nil/0, cons/2のように定める。

ここではVarの元を大文字で始まる文字列（X, Y, Z, X1, X2, …)であらわすことにする。

[定義 2] Term を次のように定まる項の集合とする。

(i) $\tau \in$ Var またはτがa, b, nilの場合，$\tau \in$Term.

(ii) $\tau 1$, $\tau 2 \in$ Term のとき，

$$\mathrm{cons}(\tau 1, \tau n) \in \mathrm{Term}$$

[定義 3] 項τが単純であるとは，τがa, b, nilのいずれかであるか，cons(X, Y) の形をしていてX, Yが異なる変数である場合をいう。

[定義 4] 写像σ : Var → Terms が次の条件を充たすとき代入と呼ばれる。

$$\Sigma = \{X \mid \sigma X \neq X,\ X \in \mathrm{Var}\}$$

とするとき，Σは有限集合。

ここで，代入の定義域をVarからTermへ拡張する。

[定義 5] $\tau \in$ Term について$\sigma\tau$は次のように定義される。

$$\begin{aligned}\sigma\tau = \ &\text{if } \tau = X \in \mathrm{Var} \text{ then } \sigma X\\ &\text{else if } \tau \in \{a, b, nil\} \text{ then } \tau\\ &\quad\text{else if } \tau = \mathrm{cons}(\tau 1, \tau 2) \wedge\\ &\qquad \tau 1, \tau 2 \in \mathrm{Term}\\ &\qquad\quad \text{then } \mathrm{cons}(\sigma\tau 1, \sigma\tau 2)\end{aligned}$$

[定義 6] Vを変数の集合，Σを写像σから[定義 4]の通りに定まる集合とする。$\Sigma \subset V$ であるときσはVに制限されているという。また$\Sigma \cap V = \phi$のとき，σはVについて不変であるという。

[定義 7] $\tau, \tau' \in$ Term とする。τがτ'の呼び換えであるとは次のような代入σおよびσ'が存在し，$\tau = \sigma\tau'$ かつ $\tau' = \sigma'\tau$ となることをいう。

[定義 8] 単純な代入式

$X \in$ Var, τを単純な項とするとき，次のような式を単純な代入式という。

$$X = \tau$$

特に$X = X$という式はtrueと表記する。単純な代入式の有限集合を用いて代入を表現することができる。しかしながら一般に単純な代入式の有限集合が常に代入を定めるわけではない。

例 1) {X = cons(Y, Z), Y = a} は代入を定義する。
一方 {X = a, X = b} は代入を定義しない。 //

以下，代入式の集合とそれが定める代入とを同一視する。また以下では慣習にしたがいcons(X, Y) は[X|Y] で，nil は [] で表わす。

本稿で提案するI/O 履歴は，純Horn論理型プログラムの単位節の概念にあたるものであり，[Levi 88]のガード付原子式の概念の拡張である。また，[Takeuchi 86] の derivation tree からプロセスの内部変数についての情報をとりのぞいたものとも考えられる。すなわちI/O履歴はプロセスの呼ばれたときの形を表わすヘッドＨとプロセスのある実行における入出力の関係を示すボディ部ＧＵを用いて次のように表わされる。

H :- GU

ここでHは互いに異なる変数に述語記号を適用したもの，GUは単純な代入式の集合で表現された代入σとボディ部での単一化の実行を表わす式 Ub の対<σ | Ub> の集合である。直観的にはHという形のゴールの引数がσによって具体化されるとUb という単一化が行われることを意味する。たとえば次のプログラムについて，

```
p1(X, Y) :- X = [A|X1] |
            Y = [B|Y1], p(A, B), p1(X1, Y1).
p(A, B) :- A = a | B = b.
p(B, A) :- B = b | A = a.
```

次のI/O履歴はp1の入力ストリームXにaが入力され（すなわちXの第1要素が aに具体化され）出力ストリームYにb が出力され続いてbが入力されると a が出力される様子を記述している。

```
p1(X, Y) :- {<{X = [a|X1]}|Y = [b|Y1]>,<{X = [a|X1],
            X1 = [b|X2]}|Y1 = [a|Y2]>, …}
```

Hの一つのI/O履歴はプロセスHの可能な実行の一つを表わす。したがって，実行中に異なる節にコミットする可能性がある場合は，異なるI/O履歴が存在する。また，同じ節にコミットした実行でも，並列に走っているプロセスと異なるスケジューリングの違いによって異なるI/O履歴が存在する可能性がある。

実際に考えられるGHCのプログラムの正常な実行を考えた場合，I/O 履歴のボディ部は幾つかの特徴を備えている。例えばこの例からわかるように，<σ1 | Ub1>, <σ2 | Ub2> ∈GUについて，Ub2はUb1 より先に実行可能となるならばσ1 ⊂ σ2 のような関係がある。一般にGUの上で，このような実行順序について半整列順序が定義される。

さらにGHCプログラムの正常な実行には次のような特徴がある。すなわち，GHCにおける変数は論理的な変数であり，ある変数への単純な具体化はプロセスの実行中には高々一度しかおこなわれず（実行が正常に続く限り）2度目以降はテスト・ユニフィケーションとなり，異なる項との単一化は成功しない。また，あるプロセスが変数Xをτに具体化する場合，他のプロセスがその変数を同じ項に具体化するのを待つ必要はない。またある変数が異なる項に具体化されるのを待つことはない。

本節の残りの部分ではではこのような特徴を反映した集合，ガード付ストリームについて述べる。

[定義 9] X ∈ Var，τを単純な項とするとき，次のような式を単純な判定式という。

X ?= τ

uni(X，τ) は X = τという代入式またはX ?= τという判定式を表わす。

[定義 10] 代入σ，変数X，単純な項τについて，<σ |uni(X，τ)> をガード付代入とよぶ。このときσを<σ |uni(X，τ)>のガード部，uni(X，τ)を能動部とよぶ。

直観的にはuni(X，τ)が代入式であったときは実際にXを具体化する単一化を，判定式であった場合はテスト・ユニフィケーションをあらわす。

[定義 11] 与えられたガード付代入の集合GUの上に次のような関係 <を定義する。<σ1 | u1>, <σ2 | u2> ∈ GUとする。ある代入θが存在しθσ1 = σ2 となりかつ σ1 = θ'σ2 となるθ'が存在しないとき，

<σ1 | u1> < <σ2 | u2>

とする。このように定義された関係 < が半整列順序となることは容易に示せる。

[定義 12] ガード付代入の集合GUが次の条件をみたすとき，GUをガード付ストリームとよぶ。

1) <σ1 | u1>, <σ2 | u2> ∈ GU，
<σ1 | u1> ≠ <σ2 | u2> かつu1とu2が同じ左辺をもつならば少なくとも一方は判定式であり，右辺は互いに等しい。またu1が代入式でu2が判定式であったとき

<σ2 | u2> < <σ1 | u1>

でない。

2) <σ | u> ∈ GU ならばσは，どんな<σ' | X = τ> ∈ GU についてもXを具体化せず，かつどんな<σ' | X ?= τ> ∈ GU についてもτ ≠ τ' についてX = τ' ∈ σ とはならない。

3) <σ | u>, <σ' | u'> ∈ GU，X = τ ⊂ σ，X = τ' ∈ σ' ならばτ = τ'。

例 2) 次のような集合GU1，GU2について，GU1はZにaの列を次々と読みこんでb の列をYに出力するプロセスの，GU2はX にaの列を，Yにbの列を読みこんで両者をマージし，Zに出力するプロセスの動作載1例をそれぞれあらわしている。

GU1 = {gu1i (1≦i)}，
GU2 = {gu2j (1≦j)}

```
gu11 = <{Z = [A|Z1], A = a} | Y = [B|Y1]> ,
gu12 = <{Z = [A|Z1], A = a} | B = b> ,
gu13 = <{Z = [A|Z1], A = a,
         Z1 = [A1|Z2], A1 = a} | Y1 = [B1|Y2]>
gu14 = <{Z = [A|Z1], A = a,
         Z1 = [A1|Z2], A1 = a} | B1 = b>
       . . . . . .

gu21 = <{X = [A0|X1], A0 = a} | Z = [A|Z1]>
gu22 = <{X = [A0|X1], A0 = a} | A = a>
gu23 = <{X = [A0|X1], A0 = a,
         Y = [B|Y1], B = b} | Z1 = [A1|Z2]>
gu24 = <{X = [A0|X1], A0 = a,
         Y = [B|Y1], B = b} | A1 = b>
gu25 = <{X = [A0|X1], A0 = a, Y = [B|Y1],
         B = b, Y1 = [B1|Y2], B1 = b} |
         Z2 = [B2|Z3]>
gu26 = <{X = [A0|X1], A0 = a, Y = [B|Y1],
         B = b, Y1 = [B1|Y2], B1 = b} |
         B2 = b>
         . . . . . .                    //
```

Fun, Varから定まるすべてのガード付ストリームの集合をStrで表わす。並列に走る複数のゴールを含む節の真/偽を議論するには次の概念が重要である。

[定義 13] $1 \leq i \leq n$ について $GUi \in Str$ とするとき

$$
\begin{aligned}
Gu1 = \{&\langle \sigma \mid u\rangle \mid \\
&\exists \langle \sigma \mid u\rangle \in GUi \text{ for some} \\
&\forall X = \tau \in \sigma, \\
&\neg \forall j \ \langle \sigma' \mid X = \tau\rangle \in GUj\}
\end{aligned}
$$

$$
\begin{aligned}
Guk+1 = Guk \cup \\
\{&\langle \sigma \mid u\rangle \mid \exists \langle \sigma' \mid u\rangle \in GUi \text{ for some } i, \\
&\forall X = \tau \in \sigma' \\
&((\neg \exists j \ \langle \sigma'' \mid X = \tau\rangle \in GUj) \vee \\
&\langle \sigma'' \mid X = \tau\rangle \in Guk) \wedge \\
&\sigma = (\sigma' - \{X = \tau \mid \langle \sigma'' \mid X = \tau\rangle \in Guk\}) \\
&\cup \{\sigma'' \mid \langle \sigma'' \mid X = \tau\rangle \in Guk\}\}
\end{aligned}
$$

とおくとき,

$$GU = \bigcup_{i \to \infty} Gui$$

とする。このGUがガード付ストリームになり，かつ次のU1とU2が一致するとき,

$$U1 = \{u \mid \langle \sigma \mid u\rangle \in GU\}$$
$$U2 = \{u \mid \langle \sigma \mid u\rangle \in GUi \text{ for some } i\}$$

このGUを GU1, GU2, … , GUn の同期付マージとよび, GU1 ‖ GU2 ‖ … ‖ GUn と表わす。

n = 1 の場合同期付マージは常に定義でき結果はGU1自身と等しくなる。

例 3) {<{X = a} | Y = b>} と {<{Y = b} | X = a>} は同期付マージ不可能であるが, {<{X = a} | Y = b>} と {<{Y = b} | X ?= a>} はマージ可能であり結果は {<{X = a} | Y = b>, <{X = a} | X ?= a>} となる。また {<{true} | X = a>} と {<{X = a} | Y = b>, <{X = a} | X ?= a>} とのマージの結果は {<{true} | Y = b>, <{true} | X = a>, <{true} | X ?= a> } となる。

//

例 4) 先の例 2)に述べたガード付ストリームGU1, GU2 は同期付マージできない。すなわち,

```
Gu1 = {gu21, gu22},
Gu2 = Gu1 ∪
      {<{X = [A0|X1], A0 = a} | Y = [B|Y1]>,
                <{X = [A0|X1], A0 = a} | B = b>}
Gu3 = Gu2 ∪
     {<{X = [A0|X1], A0 = a, A1 = a} | Y1 = [B1|Y2]>,
      <{X = [A0|X1], A0 = a, A1 = a} | B1 = b>,
      <{X = [A0|X1], A0 = a, A1 = a} | Z1 = [A1|Z2]>
      <{X = [A0|X1], A0 = a, A1 = a} | A1 = b>}
           . . . . . .
```

ここでGu1 ∪ Gu2 ∪ Gu3 (= Gu3) はガード付ストリームとはならない。すなわちA1は能動部で b に具体化されているにもかかわらずガード部分で a に具体化されているので, [定義 12] の2) に反する。

//

3. モデル論的セマンティクス

本節では先に述べた入出力履歴の概念をもとに，純Horn論理型プログラムにおける Herbrand基底，単位節等の概念に相当するものを並列論理型言語にたいして導入する。この節ではGHCの計算規則にそって実行される簡単な並列プログラミング言語を示す。この言語は議論を単純にするためにGHCの極端に簡単なサブセットを採用している。しかし，この制限が一般性を失うものではないことは容易に示される。本質的には [Levi 88] のstrong normal form への変換アルゴリズムの拡張によって示される。

述語記号の集合をPredで表わす。Predの元にはそれぞれarityが定められているものとする。ここではPredは有限

集合であるとし，小文字の並びでその元を表わす。Predの元（n-ary)をｎ個の項に適用したものを原子式という。

[定義 14] H，B1，　B2，…，Bn を原子式，かつHの引数部に出現するのは全て異なる変数，また各Bi(1≦i≦n）は原子式，Ug，Ub を単純な代入式ととき，

H :- Ug | Ub, B1,　B2, …, Bn

をガード付節とよぶ。ガード付節の集合をプログラムとよぶ。

以下では，Hがｐ(X1，X2，…，Xk)のとき，Var(H) = {X1，X2，…，Xk} とする。
[定義 15] 与えられた Var，Fun，Pred について，擬I/O履歴ｔとは次のようなものである。

p(X1，X2，…，Xk) :- GU

ここでｐ∈Pred，X1，X2，…，Xk はすべて互いに異なるVarの元，またＧＵ ∈ Str でありかつ，ＧＵの各元の能動部に出現する代入式および判定式の左辺およびガード部の代入によって具体化される変数は，以下のように定まる変数の集合 ∪ Vi に含まれる。
i→∞

V0 = Var(p(X1，X2，…，Xk))
Vi+1 = {X | ∃Y ∈ Vi, ⟨σ|U⟩ ∈GU,
かつU = uni(Y，τ）または
Y = τ ∈ σ であり，Xはτに出現する。}

ここで，p(X1，X2，…，Xk) をｔのヘッド部分，ＧＵをボディ部分とよぶ。
直観的にはＧＵにはヘッド部分から”見える”変数しか出現しない。したがって，プロセスの内部変数についての情報は棄てられている。すなわち，ボディ部のガード付ストリームが表わしているのは「このプロセスが何をするか」であり「このプロセスはどのように計算を行うか」ではない。この意味では計算木[Takeuchi 86] によるアプローチより，抽象化されたセマンティクスを定義している。

擬I/O履歴は並列言語における単位節の概念に相当する。しかしながら擬I/O履歴では本質的に同じ計算に対して複数の記法が可能となる。すなわちヘッドに出現する変数以外の変数については，いかなる変数が用いられようと外から観測する限りにおいては同じ計算を表現しているものとみなすことができる。そこで擬I/O履歴の領域に適当な同値関係を導入し，その同値類でHerbrand基底にあたるものを定義する。
[定義 16] 与えられたｎ個の擬I/O履歴：ｔ1，…，ｔn が変数互換であるとは，いかなるｔi，ｔj についてもｔiとｔj に共通に出現する変数の集合は，次に定義される変数の変数の集合: Common(ｔi，ｔj)と等しいことをいう。
Hi，Hj をそれぞれｔi，ｔj のヘッド部，ＧＵi，ＧＵj をボディ部とするとき，

COM 1(ｔi，ｔj) = Var(Hi) ∩ Var(Hj)

COM k+1(ｔi，ｔj) = COM k ∪
{X| ∃ｇui ∈ＧＵiかつ∃ｇuj ∈ ＧＵj
でありｇui，ｇujのいずれもガード部または能動部にuni(Y，τ）という形の式を含む極小の元であり，Y∈ COM k(ｔi，ｔj)，かつτはX，[X|Z] または [Z|X]の形をしている。}

Common(ｔi，ｔj) = ∪ COM k
k→∞

明らかに ｎ = 1 の場合，ｔ1 は変数互換である。
[定義 17] 写像σ:Var → Var （必ずしも代入になっているとは限らない）について，σσ'= σ'σが恒等写像となるσ'が存在するとき，σ（およびσ'）を呼び換え写像とよぶ。

入出力履歴ＧＵと呼び換え写像σから定まる入出力履歴σＧＵを次のように定義する。

σＧＵ = {σｇｕ | ｇｕ ∈ ＧＵ}

ここで，ｇｕ = ⟨θ | uni(Y，τ')⟩ のときσｇｕ = ⟨σ∗θ | uni(σY，στ')⟩，ただしσ∗θ は次のような集合定義される代入。

σ∗θ = {σX = στ | X = τ∈ θ}

ＧＵが入出力履歴であったとき，σＧＵも入出力履歴となることは容易に示せる。
[定義 18] 擬I/O履歴ｔ1，ｔ2 について，ｔ1 : p(X1，X2，…，Xk) :- ＧＵ1，ｔ2 : p(X1，X2，…，X k) :- ＧＵ2 すなわち，ｔ1，ｔ2 はおなじ原子式をヘッドにもち，かつＧＵ1 に出現する変数の集合からＧＵ2 に出現する変数の集合への Var(p(X1，X2,…,Xk))について不変な呼び変え写像σが存在し，σＧＵ1 = ＧＵ2 となるとき

ｔ1 ～ ｔ2

とする。

関係 ～が同値関係となることは容易に示せる。以下では FUN, VAR, PREDから定まるすべての擬I/O履歴の集合の ～ による商集合をＩ／O-histと表記しその元をI/O履歴とよぶ。以下ではＩ／O-histはHerbrand 基底に代わるものとしての役割を果たす。

[定義 19] Ｉ／O-histの任意の部分集合を解釈と呼ぶ。

[定義 20] Fun, VAR, PRED が定められているものとする。ｔをこれらから得られるI/O履歴，ｇをゴールとするとき，ｔがｇのトレースであるとは，ｔのインスタンスはＨ :- ＧＵという形をしており，あるσについてσＨ = ｇ かつ，任意の$\langle\theta|U\rangle \in GU$ についてσ'θ = σ'σとなるσ' が存在し，かつＵが判定式Ｘ ?= τならばσＸ = στ，Ｕが代入式ならばσはＸを具体化しないことである。

ここで代入σが変数Ｘを具体化しないとは，σＸ = Ｙ ∈ VAR であり，かつＸ以外にσＺ = Ｙ となる変数Ｚが存在しないことをいう。

[定義 21] Fun, VAR, PRED が定められているものとする。Ｉをこれらから得られる解釈，ｇをゴールとするとき，ｇがＩで真であるとはI/O履歴：ｔ∈Ｉが存在し，ｔがｇのトレースとなることである。

例 5) Ｉ = {r(X) :- {<{true}|X ?= a>}, r(X) :- {<{true}|X = a>}} とするときr(a), r(X) はＩで真であり，r(b), r([a|Y]) は真でない。

//

[定義 22] FUN, VAR, PRED, Ｉを上の通りとする。ｇ1, …, ｇn をFUN, VAR, PRED から得られるゴール節とするとき，ｇ1, …, ｇn がＩで真であるとは，各ｇi (1≦i≦n)のトレースｔiがＩに含まれ，ｔiの変数互換なインスタンスのボディ部分の同期付マージが定義できることである。また空（すなわち ｎ = 0 の場合）なゴール節は常に真であるとする。

この定義がwell definedであるためには，ｇ1, …, ｇn のそれぞれのトレースのインスタンスのボディ部分の同期付マージが定義できるか否かが，インスタンスの選び方に依存しないことである。これは次の命題より容易に示すことができる。

[命題] 入出力履歴ＧＵ1,…,ＧＵnと呼び換え写像σについて，ＧＵ1,…,ＧＵnの同期付マージが定義できるならばσＧＵ1, …,σＧＵn についても定義できる。

証明は同期付マージの定義よりstraight forward である。

例 6) Ｉ = {p(X, Y) :- {<{X = a}|Y = b>},
q(X, Y) :- {<{Y = b}|X = a>}}

とするとき，p(X, Y), q(X, Y) というゴール節は真とならない。一方，

Ｉ' = {p(X, Y) :- {<{X = a}|Y = b>},
p(X, Y) :- {<{X = a}|Y ?= b>},
q(X, Y) :- {<{Y = b}|X = a>},
q(X, Y) :- {<{Y = b}|X ?= a>}}

では p(X, Y), q(X, Y) というゴール節は真である。 //

あるゴールｇが真であることと，ｇのみからなる長さ1のゴール節が真であることが同値なことは容易に示せる。

次にプログラムが解釈Ｉで真であるということを定義する。これによってプログラムを真とする解釈すなわちプログラムのモデルの概念が導入される。

[定義 23] ＧＵをガード付ストリーム，Ｖを変数の有限集合とする。ここでＧＵのＶによる制限ＧＵ↓Ｖとは次のような集合である。

V0 = V,
Vi+1 = {Ｘ | ∃Ｙ ∈ Vi, $\langle\sigma|U\rangle \in GU$,
Ｕ = uni(Ｙ, τ)またはＹ = τ ∈ σ であり，
Ｘはτに出現する。}
とするとき，

$GU\downarrow V = \{\langle\sigma|U\rangle \mid \langle\sigma|U\rangle \in GU,$ かつＵの左辺は $\bigcup_{i\to\infty}$ Viに含まれる.$\}$

ＧＵがガード付ストリームのとき，ＧＵ↓Ｖもガード付ストリームとなる。

[定義 24] ＧＵをガード付ストリーム，Ｕを単純な代入式とするとき，ＵとＧＵから定まる次の集合がやはりガード付ストリームとなるとき，

$\{\langle\sigma|Ub\rangle|\ \langle\sigma'|Ub\rangle \in GU,\ \sigma = \{U\} \cup \sigma'\}$

これをＧＵ＄Ｕで表わす。

[定義 25] ガード付節の集合Ｄについて，解釈ＩがＤのモデルであるとは，Ｉの任意の元ｔについてある節 Ｈ :- Ｕg | Ｘ = τ , Ｂ1, …, Ｂk が存在して，ある代入σについて，ｔのインスタンスが次のような形をしていることである。

Ｈ :- {<{Ｕg} | uni(Ｘ, τ)>}∪
((ＧＵ1 ‖…‖ ＧＵk)＄Ｕg)↓Var(Ｈ)

ここでGUiはσBiというゴールのトレース（∈I）のインスタンスのボディ部，σはσ = {Ug, X = τ} ∪ σ'という形でありかつ，V = Var(H)∪{X}とおくときσはVに制限された代入であり，任意の<θ|U> ∈ (GU1 ‖…‖ GUk)$Ug についてθ < σ またはθ = σとなる。

例 7) Dを次のようなガード付節の集合とする。

```
p(X, Y) :- X = a | Y = b.
q(X, Y) :- Y = b | X = a.
t(X, Y) :- X = a | true, p(X, Y), q(X, Y).
```

このとき次のIはDのモデルである。

```
I = {t(X, Y) :- {<{X = a}|true>, <{X = a}|Y = b>,
     <{X = a}|X ?= a>} p(X, Y) :- {<{X = a}|Y =
     b>}, p(X, Y) :- {<{X = a}|Y ?= b>}, q(X, Y)
     :- {<{Y = b}|X = a>}, q(X, Y) :- {<{Y = b}
     |X ?= a>}}                                  //
```

[定義 25]によれば，Dのモデルは複数存在しうる。明らかに空集合は任意のDについてモデルとなる。また次の命題はモデルの定義より明らかに成り立つ。

[命題] 添え字の集合Indについて，節集合Dのモデルの族Mi(i∈Ind) を考える。このとき

$$\bigcup_{i \in Ind} Mi$$

はやはりDのモデルである。

上の命題よりDの全てのモデルの集合和をとったものはやはりDのモデルであり，これはDの最大のモデルとなる。節集合Dのセマンティクスとは Dの最大モデルによって定義される。直観的には最大モデルとはDの上で実行したすべての正常な計算のパスを集めたものと考えられる。

例 8) 次のプログラムPを考える。

```
merge(X , Y,  Z) :- X = [A1|X1] |
                    Z = [A1|Z1], merge(X1, Y, Z1).
merge(X , Y,  Z) :- Y = [B1|Y1] |
                    Z = [B1|Z1], merge(X, Y1, Z1).
inva(Z, Y) :-  Z = [A|Z1] |
               Y = [B|Y1], inval(A, B, Z1, Y1, Y).
inval(A, B, Z1, Y1, Y) :-
               A = a | B = b, inva(Z1, Y1).
inval(A, B, Z1, Y1, Y) :-
               A = b | true,  inva(Z1, Y).
```

Pの最大モデルは例 3) の GU1, GU2 について，

```
inva(Z, Y) :- GU1
merge(X, Y, Z) :- GU2
```

という形の元を含む。すなわちこれらのゴールは単独に実行した場合，GU1, GU2に示されるような動作をする可能性がある。しかしながら次のようなゴール節として同時に実行した場合，それぞれのゴールが上のような動作をすることはない。

```
inva(Z, Y), merge(X, Y, Z)
```

一般に上の様なゴール節においてXにaの列を無限に与え実行を続けた場合，Zの具体化された部分ではbの数はa の数を決して上回ることはない。このことは，このセマンティクスにおいてそれぞれのゴールの任意のトレースの同期付マージの結果についてb がaより先行するような結果がありえないことを示すことによって議論できる。このような議論は [Levi 88]の方法や，純Horn論理型プログラムを完備Herbrand空間を導入したアプローチ[Sakakibara 85]では不可能なものである。 //

与えられたプログラム節の集合Dのモデルで真となるゴール節は，[Ueda 85]に述べられたGHCの計算規則のようなガード／コミットを考慮した並列入力反駁の規則もとで”正常に動作する”ゴール節を定めることを目的としている。しかしながらここでいう”正常な動作”は必ずしも有限時間で成功するゴールを意味しない。すなわち先に述べたように，無限に結果を出力しながら計算を続けるプロセスはここでは正常な動作をするものと見做し，そのようなプロセスを表わすゴール節は真となる。

また，サスペンドするようなゴール（またはゴール節）についても場合によっては真となる。すなわち次のようなプログラムについて，

```
p(X, Y) :- X = a | Y = b.
q(X, Y) :- Y = b | X = a.
t(X, Y) :- X = a | true, p(X, Y), q(X, Y).
```

このプログラムのもとではt(X, Y)というゴールはサスペンドするが，X が a に具体化されることにより実行がさらに進む。この場合，Dの最大モデルには

```
t(X, Y) :- {<{X = a}|true>,
            <{X = a}|Y = b>, <{X = a}|X ?= a>}
```

という元が含まれ，したがってこのようなゴールは真となる。しかしながら，プログラムの３行目のtの定義を次のように変更した場合を考える。

```
t :- true | true, p(X, Y), q(X, Y).
```

この場合ゴール t を実行するとp(X, Y), q(X, Y) というゴール節が呼び出されたのちサスペンドし，以後実行が進む可能性はない。このようなゴールは先の定義では偽となる。

4．不動点的セマンティクス

本節では先に定義したモデル論的セマンティクスがDから定義される関数の最大不動点によって特徴づけられることをしめす。

Ｉ／Ｏ-histが与えられたとき，全ての解釈の集合ＩＰは集合の包含関係のもとで完備束となり，最大元はＩ／Ｏ-hist，最小限は空集合φである。

[定義 26] ガード付節の有限集合Ｄから定まる関数ΦD：ＩＰ → ＩＰ を次のように定義する。

ΦD（S） = S∩ {t | tのインスタンスは，

H :- {<{Ug} | uni(X, τ)>}∪
((GU1 ‖ … ‖ GUk)$Ug)↓Var(H)

という形をしており，

∃H :- Ug | X = τ, B1, …, Bk ∈D,

GUiはσBiというゴールのトレース（∈S）のインスタンスのボディ部，σはσ = {Ug, X = τ} ∪ σ'という形でかつ，V = Var(H)∪ {X} とおくときσはVに制限された代入であり，任意の<θ|U> ∈ (GU1 ‖ … ‖ GUk)$Ug についてθ < σ または θ = σ}

鎖si：s1 ⊃ s2 ⊃ … … について，siの最大下界を∩siで表わすことにする。

[定義 27] Ｌを完備束とする。関数ｆ：Ｌ → Ｌが，任意の鎖si：s1 ⊃ s2 ⊃ …について

∩{f(si)|0 ≦i} = f(∩{si|0 ≦i})

となるとき，ｆは下向きに連続であるという。

よく知られているように，関数ｆが連続ならばｆは単調である。すなわちS1 ⊃ S2 ならばf(S1) ⊃ f(S2)である。また次の２つの命題はよく知られたものである。

[命題] 完備束Ｌの最大元とＴとするとき，ｆが下向きに連続であるならばｆの最大不動点 gfpｆは次のような式で特徴づけられる。

gfpf = ∩{fn(T) | n > 0}

ここでf0(T) = T, fn+1(T) = f(fn(T)) とする。

[命題] ｆが単調な関数のとき，ｆの最大不動点は次の集合の上限に等しい。

{x | f(x) ⊃ x}

以下に，ΦDの性質を幾つか示す。証明は省略する。

[命題] ΦD は下向きに連続である。

[命題] ΦD（Ｉ）⊃ Ｉ iff ＴはＩのモデル。

以上により次の定理が得られる。

[定理]

節集合Ｄの最大モデルMDはΦD の最大不動点であり，次の式で表わされる。

MD = ∩ {ΦDn(Ｉ／Ｏ-hist) | n > 0}

5．結び

本稿では，flatＧＨＣのサブセットについてその最大モデルによる宣言的セマンティクスを与え，さらに最大不動点による特徴付けを行った。ここで導入されたセマンティクスによって，従来の方法では不可能であった，ガード／コミット機構によって制御される無限計算の途中結果として得られる解をプログラム節の集合の論理的帰結としての特徴付けることが可能となった。

謝辞：本研究をすすめるにあたり，有益な議論をして下さったＩＣＯＴ古川次長，第１研究室の皆様，およびPisa大学 Levi教授に感謝します。

参考文献：

[Apt 82] K. Apt, and M. H. Van Emden, Contributions to the theory of logic programming, J. Assoc. Comput. Mach. 29, (1982)

[Brock 81] J. D. Brock, W. B. Ackermann, Scenarios: A Model of Non- determinate Computation, Lecture Notes in Computer Science, No. 107 Springer, 1981

[Clark 86] K. L. Clark and S. Gregory, PARLOG: Parallel programming in logic, ACM Trans. on Programming Language and Systems 86, 1986

[Falaschi 87] M. Falaschi, G. Levi, M. Martelli, and C. Palamidessi, A more general Declarative Semantics for Logic Programming Languages, submitted to Theoret. Comp. Sci.
[Falaschi 88] M. Falaschi, and G. Levi, Operational and fixpoint semantics of a class of commited-choice logic languages, unpublished memo.
[Levi 87] G. Levi and C. Palamidessi, An Approach to the Declarative Semantics of Synchronization in Logic Language, Proc. of International Conf. on Logic Programming 87, 1987
[Levi 88] G. Levi, A new declarative semantics of Flat Guarded Horn Clauses, Tech. Rep. of ICOT, to appear, 1988
[Lloyd 84] J. W. Lloyd, Foundations of logic programming, Springer-Verlag, 1984
[Maher 87] M. J. Maher, Logic Semantics for a Class of Commited-Choice Programs, Proc. of International Conf. on Logic Programming 87, 1987
[Park 69] D. Park, Fixpoint induction and proofs of program properties, Machine Intelligence 5, Edinburgh University Press, Edinburgh, 1969
[Sakakibara 85] Y. Sakakibara, A Fixpoint Characterization of Stream Parallelism in Logic Programs, Proc. of 2nd Cof. JSSST, in Japanese, 1985
[Saraswat 85] V. A. Saraswat, Partial Correctness Semantics for CP [↓, |, &], Lecture Notes in Comp. Sci., No. 206, 1985
[Saraswat 87] V. A. Saraswat, The Concurrent logic programming CP: definition and operational semantics, Proc. of ACM Symp. on Principles of Programming Languages, 1987
[Shapiro 86] E. Y. Shapiro, Concurrent Prolog: A progress report, Lecture Notes in Comp. Sci. No. 232, 1986
[Shibayama 87] E. Shibayama, A Compositional Semantics of GHC, Proc. of 4th Cof. JSSST, 1987
[Takeuchi 86] A. Takeuchi, Towards a Semantic Model of GHC, Tech. Rep. of IECE, COMP86-59, 1986
[Ueda 88] K. Ueda, Guarded Horn Clauses, MIT Press, 1988
[Ueda 86] K. Ueda, On Operational Semantics of Guarded Horn Clauses, Tech. Memo of ICOT, TM-0160, 1986