プログラミング-言語・基礎・実践- 16－14
(1994. 3. 10)

並行論理型言語 NGHC の時相論理式による制約を用いた部分計算法

中尾 有志　　村上 昌己
岡山大学 工学部
岡山市津島中 3 丁目 1 番 1 号
E-mail: {nakao,murakami}@momo.it.okayama-u.ac.jp

あらまし

本稿では，並行論理型言語 Nested Guarded Horn Clauses(NGHC) で記述された応答型並行プロセスを部分計算する方法を述べる．本稿で述べる方法は，プロセスが受け取るメッセージについて成り立つ制約条件を時相論理式で表し，それを利用してプロセスを部分計算する方法である．この方法によりプロセスを実行環境で特殊化したより効率的なプロセスに変換できる．

和文キーワード　並列論理型言語 NGHC, 部分計算, 応答型プロセス, 時相論理式

Partial Evaluation of Nested Guraded Horn Clauses Programs using Temporal Logic Formulas

Yushi Nakao　　Masaki Murakami
Department of Information Technology,
Okayama University
3-1-1 Tsushimanaka, Okayama, Japan
E-mail: {nakao,murakami}@momo.it.okayama-u.ac.jp

Abstract

We propose a method of partial evaluation for reactive concurrent processes in Nested Guarded Horn Clauses (NGHC) with constraints on the environment processes. Constraints which are satisfied with messages recieved by processes are described with temporal logic formulas. This method can transform processes to more efficient processes specialized to the executing environment.

英文 key words　NGHC, Partial Evaluation, Reactive Process, Temporal Logic Formulas

1 はじめに

本稿では，並行論理型言語 Nested Guarded Horn Clauses (NGHC) プログラム [1] の部分計算法を述べる．

プログラムの部分計算法とは，プログラムをその実行環境についての情報を利用して，より実行効率のよいプログラムに特殊化する方法である [2] ．部分計算に関する研究は，従来からさまざまな分野で行われている．並行論理型言語の分野については [4, 6] で報告されている．

[4] の方法は，初期ゴールの入力変数に関する束縛情報によってプログラムを特殊化する方法が取られている．例えば，ゴール $append(X, Y, Z)$ の入力変数 Y を $Y = [1, 2, 3]$ という束縛情報を用いてプログラムを特殊化する方法である．つまり [4] の部分計算法は，計算の開始時にすべての入力を受け取り 終了時に出力をするという，変換型の立場に基づいて行われている．ところが並行論理型言語では，プログラムを変換型の立場で捉えることはしない．並行論理型言語におけるプログラムは，計算の開始時に受け取る入力のみならず，実行中にもその環境とメッセージを送受信しながら処理を続けていくという応答型の立場で捉えられる．したがって [4] は，並行論理型言語の部分計算として不完全である [6] ．

そこで [6] では，複数の入力チャネルによってメッセージを受け取る応答型プロセスにおいて，あるチャネルに届けられるメッセージが既知であるとして，その情報を用いて応答型プロセスを特殊化するという部分計算法を採用している．また [6] では，このような部分計算法を並行論理型言語に適用した例も報告している．

一方応答型プロセスとは，メッセージを受け取って処理をし，環境へ返答し，次の入力を受け取るプロセスになるものと定式化できる．したがって，最初のプロセスの何世代か後のプロセスが受け取るメッセージについて何らかの制約が成り立つ場合，その制約を用いてプロセスの動作効率を改善できるはずである．しかし，変換型の立場に基づく [4] の部分計算法では，実行の途中に届けられるメッセージについての制約情報を記述できない．また [6] の方法は，届けられるメッセージの値が既知であるものしか許さないので，ここでいう「何らかの制約」を利用して部分計算を行うことはできない．

そこで，プロセスが受け取るメッセージについて成り立つ制約を時間を表現する体系の一つである時相論理 [5] を用いて表し，関数表現のプロセスをその制約で部分計算した [7] が報告されている．しかし [7] では，関数表現のプロセスを対象としていたので，具体的な並行プログラミング言語への適用はされていない．そこで本稿では，この方法を並行論理型言語 NGHC に適用する．

本稿では，NGHC で記述される応答型プロセスが受け取るメッセージ，また，そのメッセージに対する応答メッセージとコンティニュエーションのプロセスの記述法を具体的に定める．これによって，NGHC で記述される応答型プロセスを関数表現することが可能となる．そして，このように関数表現された NGHC の応答型プロセスを，時相論理式を用いて記述した実行環境についての制約情報を用いて，新たな応答型プロセスへ特殊化する．さらに，部分計算によって得られる関数表現の応答型プロセスを NGHC のプログラムへ変換する．

以上の手法により，与えられた NGHC プログラムを，その実行環境における初期入力のみならず，その実行中に受け取るメッセージについての制約を用いてより動作効率のよい NGHC プログラムへ特殊化することができる．さらに，この部分計算法は，[6] の拡張であると考えられる．

本稿では以下，2 節で応答型プロセスと時相論理について述べる．3 節では NGHC で記述される応答型プロセスの関数表現法を述べ，その実行環境で成り立つ時相論理式による制約を用いた部分計算法について述べる．4 節では，結論と今後の課題を述べる．

2 応答型プロセスの環境と時相論理

並行計算の分野では 通常，「プログラムは起動時に入力を受け取るだけでなく，実行中でもその環境との間でメッセージの送受信を行いながら処理を続けて行き，必ずしも終了しない」という応答型のシステムとして捉えられる．

そこで本節では，応答型プロセス，また応答型プロセスの環境について成り立つ時相論理 [5] による制約について述べ，NGHC で記述されるプロセスが応答型プロセスであることを示す．

2.1 応答型プロセス

応答型プロセスとは，上述した応答型の立場で捉えたプロセスであり，次の図で摸式化できる．

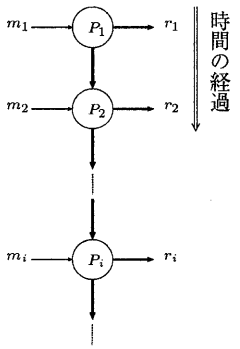

図 2.1 応答型プロセスの概念図

図 2.1 において，プロセスを P_j，各プロセスが受け取るメッセージを m_j，また環境への出力である応答メッセージを r_j としたとき，図 2.1 を利用して応答型プロセスのふるまいを次のように追うことができる．

プロセス P_1 はメッセージ m_1 を受け取り，応答メッセージ r_1 を返し，次の入力を受け取るプロセス P_2 となる．そして，プロセス P_2 も プロセス P_1 と同じようにメッセージ m_2 を受け取り，応答メッセージ r_2 を返し，次のプロセスとなり，以下同様に続いていく．

したがって応答型プロセスは，メッセージを受け取り応答メッセージとコンティニュエーションのプロセスの組を返す関数と考えることができる．そこで，プロセス P のメッセージ m に対する応答メッセージを $resp(P,m)$，またコンティニュエーションのプロセスを $cont(P,m)$ とすると，プロセス P は次のように表される．

$$P \stackrel{def}{=} \{m \rightarrow \langle resp(P,m), cont(P,m)\rangle\} \qquad (1)$$

またコンティニュエーションのプロセス $cont(P,m)$ も (1) の形式で定めることができる．ここで，受け取るメッセージが複数存在する場合は次のように表記する．

$$P \stackrel{def}{=} \{\ \begin{array}{lcl} m_1 & \rightarrow & \langle resp(P,m_1), cont(P,m_1)\rangle \\ \vdots & & \vdots \\ \|m_n & \rightarrow & \langle resp(P,m_n), cont(P,m_n)\rangle \end{array}\ \}$$

この省略記法として，I を添字の集合とした時，

$$P \stackrel{def}{=} \{\|_{i\in I}(m_i \rightarrow \langle resp(P,m_i), cont(P,m_i)\rangle)\}$$

という表記も用いる．

さらに，プロセス $P_1, P_2, \ldots, P_n$ が並行に実行されることを，プロセス間演算子 $\|$ を使って，以下のように表記する．

$$P_1 \| P_2 \| \cdots \| P_n$$

本稿では，プロセスを関数表現するとは，上記の形式でプロセスを表記することをいう．

2.2 時相論理

時相論理は，時間の進行に伴って変化して行く世界の状態記述を自然に行えるように通常の古典論理に時相記号と呼ばれる $\{\mathsf{X}, \mathsf{G}, \ldots\}$ を付加して拡張したものである．ここで，X, G という時相記号の意味は以下の通りである．

- $\mathsf{X}p$ ： 現在の次の時点で p が成り立つ．
- $\mathsf{G}p$ ： 現在から先の全ての時点で p が常に成り立つ．

このような時相記号を含む論理式の真偽は，以下のように定義されている [5] ．

定義 2.1 論理式 p が状態 s において，成立することを $s \models p$ と表す．また，S : 状態集合， $N : S \rightarrow S$ どの状態に対してもその次の状態を与える後継者関数，N^i でこの N の i 回の適用を表す．

- $s \models p$ (p が時相記号を含まない時) iff p が成立
- $s \models \mathsf{X}p$ iff $N(s) \models p$
- $s \models \mathsf{G}p$ iff 任意の $0 \leq i$ に対し $N^i(s) \models \mathsf{G}p$

2.3 応答型プロセスの環境と時相論理

本節では，応答型プロセスの環境と時相論理との関係を明確にする．

まず，定義 2.1 の状態集合 S を図 2.1 の応答型プロセス $P_j\,(1 \leq j)$ の集合に対応づけた場合，2.1 節で述べたように プロセス P_j の次の状態は P_{j+1} であるから， 定義 2.1 の 後継者関数に対して $N(P_j) = P_{j+1}$ であることが解る．したがって，応答型プロセスのふるまいは，時間の進行にともなって変化していく時間軸上のモデルに対応すると考えられる．

また，応答型プロセスの実行環境としては，各チャネルを通じてやりとりする通信相手のプロセス，または，その通信されるメッセージの系列の集合と考えることができる．これから，図 2.1 の応答型プロセスについて，次のような見方ができる．

プロセス P_1 を起動した場合 P_1 が受け取るメッセージ m_1 だけでなく，何世代か後のプロセス P_i が受け取るメッセージ m_i が解れば，さらにプロセスの処理を進めることができる．したがって，もしメッセージ m_i について成り立つ制約情報を記述できるなら，その制約を利用してプロセス P_i を特殊化できると考えられる．

このような見方をした場合，[4] では，初期入力の変数に関する束縛情報だけを用いていたので，実行の途中に届けられるメッセージについての制約情報を表現できない．よって，[4] で取られていた部分計算法は並行論理型言語の部分計算法として不十分だと考えられる．また [6] では，既知なメッセージしか用いることができないので，上述したような一般的な制約情報を記述することができない．

以上述べてきたことから，応答型プロセスの実行中の環境についての一般的な制約，つまり応答型プロセスが受け取るメッセージについての制約を時相論理式により記述できると考えられる．この時 $\mathsf{X}p$ という制約は，プロセス P のメッセージ m に対するコンティニュエーション $cont(P,m)$ が受け取るメッセージに関して制約 p が成り立つ，と考えることができる．

2.4 NGHCの応答型プロセスと時相論理

本節では，NGHCで記述されるプロセスが応答型プロセスと見なせることを述べ，さらにNGHCの応答型プロセスにおける環境と時相論理について述べる．

NGHCで記述されるプロセスは，次のように解釈できる．まず，プロセスがゴールアトム，ゴール(アトムの集まり)が並行プロセスであると見なされる．NGHCでは多くの場合，プロセスは再帰的プロセスとして定義される．また，ゴールアトム間で共有される変数を具体化することによって通信する．入力変数に対する束縛情報をプロセスが受け取るメッセージと見なす．

以下では，NGHCで記述されるプロセスを節集合WとゴールGの組(W,G)で表すことにする．

このように記述されるNGHCプロセスが，2.1節で述べた応答型プロセスであることは容易に確かめられる．したがって，関数表現の形式で表すことができる．本稿では，NGHCで記述される応答型プロセス$P=(W,G)$を関数表現する際必要となるPが受け取るメッセージm_i，そのメッセージに対するコンティニュエーションのプロセス$cont(P,m_i)$また，応答メッセージ$resp(P,m_i)$を次のように記述する．

m_i: プロセスPのゴールGをメッセージm_iで具体化したゴールG'が実行可能となるようなゴールの入力変数に対する束縛情報

$cont(P,m_i)$:
ゴールGをメッセージm_iで具体化して実行した際にサスペンドまたは成功して得られたプロセス

$resp(P,m_i)$:
ゴールGをメッセージm_iで具体化して実行した際に求められる 各ゴールの出力変数に関する 単一化アトムの系列

ただし，これらの具体的な求め方は，3.1節で示す．

さらに，NGHCで記述される応答型プロセスの実行環境としては，各変数を通じてやりとりする通信相手のプロセス，または，その通信されるメッセージの系列の集合と考えることができる．したがって，NGHCで記述される応答型プロセスの実行中の環境についての制約とは，そのプロセスが受け取るメッセージについての制約と考えられる．2.3節で述べたようにこの制約は時相論理式を用いて表現できる．

以下にNGHCに対応させた時相論理式の例を示す．

例 2.2 NGHCで記述されたある応答型プロセスの入力変数 X が無限列$[1,1,\cdots]$に束縛されるという制約を時相論理式を用いて以下のように記述できる。

$$\mathtt{X}=[1|\mathtt{Xs1}]\wedge \mathtt{G}(\_\mathtt{I}=[1|\_\mathtt{N}]\rightarrow \mathtt{X}(\_\mathtt{N}=[1|\_\mathtt{I}']))$$

(ただし,_I, _N は無名変数)

3 NGHCプログラムの部分計算法

本節では，時相論理による制約を利用したNGHCプログラムの部分計算法を述べる．本節で述べる部分計算法は，NGHCで記述される応答型の並行プロセスをその実行環境における初期入力だけでなく，実行中に受け取られるメッセージについての情報を用いてプロセスを特殊化する方法である．

本稿では，以下の手順でNGHCプログラムを部分計算する．

1. NGHCプログラムを関数表現の応答型プロセスに変換
2. 関数表現の応答型プロセスを時相論理式による制約で部分計算
3. 部分計算で得た応答型プロセスを実現するNGHCプログラムに変換

3.1 NGHCプログラムの関数表現

本節では，NGHCのプロセスを関数表現の形式で記述する際，プロセスが受け取るメッセージ，またそのメッセージに対する応答メッセージとコンティニュエーションのプロセスを求める方法を具体的に述べる．

ただし本稿では，NGHCで記述されるプロセスPをプログラム節の集合WとゴールGとの組(W,G)で表し，W中の各節は強標準形[1]であるとする．

3.1.1 プロセスが受け取るメッセージの記述

本節では，NGHCで記述されたプロセスが受け取るメッセージの求め方について述べる．

2.4節で述べたようにプロセス$P=(W,G)$が受け取るメッセージは，Gをそのメッセージで具体化したゴールG'が実行可能となるように 定めなければならない．

したがって，NGHCで記述されるプロセス$P=(W,G)$(Gは1つのアトムからなる)が受け取るメッセージm_iは

m_i: W中の節c_iの入力制約の変数をゴールGの対応する変数で置き換えた一方向単一化アトム

しかし，このようにプロセスが受け取るメッセージを記述できるのは，ゴールがただ1つだけのアトムからなる場合であることに注意されたい．

3.2節ではゴールが複数の場合，すなわち並行プロセ

スの場合について，プロセスが受け取るメッセージの記述法について述べる．

3.1.2 コンティニュエーションの記述

本節では，NGHCで記述されたプロセスが受け取るメッセージに対するコンティニュエーションのプロセスを記述する方法について述べる．

2.4節で述べたように，NGHCで記述されたプロセス P が受け取るメッセージ m に対するコンティニュエーション $cont(P,m)$ は，プロセス P をメッセージ m で具体化して実行した際にサスペンドまたは成功して得られたプロセスであると定式化した．

この立場に基づき以下では，プロセスのコンティニュエーションを具体的に求める方法を述べる．この方法はNGHCプログラムの展開規則 [1] に準じて行われる．

プロセス P のメッセージ m に対するコンティニュエーションのプロセス $cont(P,m)$ は，以下の手順で求められる．

コンティニュエーションの求め方

入力: プロセス $P=(W_P,G_P)$ &メッセージ m

出力: コンティニュエーション $cont(P,m)=(W_C,G_C)$

step c1

ゴール G_P をメッセージ m で具体化し，得られたゴールを $G_C=G_C^1,\ldots,G_C^k$ とする．

step c2

G_C の i 番目のゴールを G_C^i $(1\leq i\leq k)$ とする．また，**step c3** で更新される節集合 W_i，W_C の初期値を次のように決める．

$$W_i:=\emptyset,\ W_C:=W_P$$

step c3

I. 実行可能なゴール G_C^i が存在する場合：

1. ゴール G_C^i が W_C 中の

$$H\ :\text{-}\ I_1|O_1,\ldots,I_n|O_n\rightarrow B.$$

という形の節の n 番目の出力制約を評価して，その評価が成功しているとき以下の処理を行い **step c3** に戻る．

θ を G_C^i が節 $H\ :\text{-}\ I_1|O_1,\ldots,I_{j-1}|O_{j-1}$ を解いて得られた G_C^i の変数に関する解代入とする

G_C^i に対して

$G_C^i:=B\theta$ （G_C^i を $B\theta$ で置換）

$\forall G_C^j\in G_C$ $(G_C^i\neq G_C^j)$ に対して

$G_C^j:=G_C^j\theta$ （G_C^j を $G_C^j\theta$ で置換）

$$W_i:=W_i,\ W_C:=W_C$$

2. ゴール G_C^i が W_C 中の

$$H\ :\text{-}\ I_1|O_1,\ldots,I_n|O_n\rightarrow B.$$

という形の節の j $(j>1)$ 番目の入力制約を評価してサスペンドしているとき以下の処理を行い **step c3** に戻る．

W_C 中の節で $H\ :\text{-}\ I_1|O_1,\ldots,I_{j-1}|O_{j-1}$ というプレフィクスを持つ節を c_l $(1\leq l)$ とする．ただし，c_l は次のように表せるとする

$c_l: H\ :\text{-}\ I_1|O_1,\ldots,I_{j-1}|O_{j-1},$
$I_{l(j)}|O_{l(j)},\ldots,I_{l(n)}|O_{l(n)}\rightarrow B.$

各 c_l に対して

$c_l': H'\ :\text{-}\ I_{l(j)}|O_{l(j)},\ldots,I_{l(n)}|O_{l(n)}\rightarrow B.$

を作る．ただし H'は $\{X_1,\ldots,X_l\}$ $(=Var(H\cup I_1\cup O_1\cup\cdots\cup I_{j-1}\cup O_{j-1}))$ を引数にもつ，新たな述語記号を用いたヘッドである．さらに，

$W_i:=\bigcup_l\{c_l'\}$ (ただし $1\leq l$)

θ を G_C^i が節 $H\ :\text{-}\ I_1|O_1,\ldots,I_{j-1}|O_{j-1}$ を解いて得られた G_C^i の変数に関する解代入とする

$G_C^i:=H'\theta$

$G_C^j:=G_C^j\theta$ $(1\leq j\leq k,\ j\neq i)$

$W_C:=W_C\cup W_i$

II. $\forall G_C^i$ が，W_C 中のすべての節の 1 番目の入力制約を評価できずにサスペンドしている，または，$\forall G_C^i=true$ である場合：

プロセス P のメッセージ m に対するコンティニュエーションのプロセス $cont(P,m)$ を次のように決めて終了する．

$$cont(P,m)=(W_C,G_C)$$

以上の手順で求められるコンティニュエーションが，2.4節で述べたように，プロセス P のゴール G をメッセージ m で具体化して実行した際にサスペンド，または，成功して得られるプロセスであることは容易に確かめられる．

3.1.3 応答メッセージの記述

本節では，NGHCで記述されたプロセス P のメッセージ m に対する応答メッセージ $resp(P,m)$ の求める方法について述べる．

プロセス P のメッセージ m に対する応答メッセージ $resp(P,m)$ とは，環境への出力である．この応答メッセー

ジは 3.1.2 節で述べたコンティニュエーションを求める際に得られるゴールの出力変数に関する解代入を単一化アトムに置き換えることによって求まる.

3.2 並行に実行される NGHC プロセスの関数表現

多くの場合, NGHC で記述されるプロセスは並行に実行されるので, そのようなプロセスを関数表現する方法を具体的に求めることは意味のあることである.

本節では, プロセス P が n 個の並行プロセスである時 (すなわち, $P = P1 \| \cdots \| Pn$), P を

$$P \stackrel{def}{=} \{[]_{i \in I}(m_P^i \rightarrow \langle resp(P, m_P^i), cont(P, m_P^i)\rangle)\}$$

の形式で関数表現する方法について述べる. このように関数表現することで, $\|$ を用いてそのまま関数表現するよりも, 並行プロセスが受け取るメッセージに対する応答メッセージ, またコンティニュエーションのプロセスを捉えることが容易になる.

以下では, メッセージ m_P^i を記述する方法を定めるが, その前に次の 2 つの定義を導入する.

まず, プロセス $P1, P2$ が並行に実行される時, プロセス間で内部通信 (プロセス $P1$ とプロセス $P2$ との通信) が起こる可能性がある. 並行論理型言語では, 通信は共有変数を使って行われる. そこで, 次のような概念を定めることができる.

定義 3.1 プロセス $P1=(W_1, G_1), P2=(W_2, G_2)$ が

$$Pj \stackrel{def}{=} \{[]_{i_j \in I_j}(m_j^{i_j} \rightarrow \langle resp(Pj, m_j^{i_j}), cont(Pj, m_j^{i_j})\rangle)\} \quad (\text{ただし}, j = 1, 2)$$

と関数表現されていて, $P1 \| P2$ ($P1$ と $P2$ が並行に実行される) とする.

1. ゴール G_1, G_2 に共有変数 X が存在する
2. プロセス $P1$ が受け取れるメッセージ $m_1^k\,(k \in i_1)$ 中に $X \leq \_$ が存在し, かつ, プロセス $P2$ の応答メッセージ $resp(m_2^l, P2)\,(l \in i_2)$ に $X = \_$ または, $\_ = X$ が存在する

上の *1*, *2* が成り立つ時, メッセージ m_1^k 中の $X \leq \_$ を内部通信に関する制約といい, m_1^k を内部通信に関する制約を含むメッセージという.

定義 3.1 の 1,2 が成り立つ時, プロセス $P2$ からプロセス $P1$ へ内部通信が起こる可能性がある. 定義 3.1 は, n 個の並行プロセスについても自然の拡張として定義できる.

また, 次の概念も導入する.

定義 3.2 あるメッセージ m に対して, $(X \leq O), (X \leq P)$ が, 以下を満足する時

$$(X \leq O) \in m, \text{ かつ } (X \leq P) \in m,$$
$$\text{かつ } O \text{ と } P \text{が単一化できない}$$

$X \leq O, X \leq P$ を矛盾する制約の組という.

定義 3.2 は, $X \leq O, X \leq P$ という制約を含むメッセージが決して存在しないことを意味する.

以上のような準備をして, プロセス P が受け取れるメッセージ m_P^i の集合 M_P を次の手順で求めることができる.

並行プロセスが受け取るメッセージの求め方
入力: プロセス $P1, \ldots, Pn$ の各メッセージ
出力: 並行プロセス Pのメッセージの集合
$M_P = \{m_P^1, \cdots, m_P^h\}$

step f1

プロセス $P1, \cdots, Pn$ が受け取れるメッセージの集合をそれぞれ, $M_1, \cdots, M_n$ とする

step f2

step f1 の $M_1, \cdots, M_n$ それぞれに, 内部通信に関する制約を含むメッセージがあればその制約をメッセージから削除し, 得られたメッセージの集合を以下のように決める.

$$M_i' = \{m'^1_i, \cdots, m'^{t(i)}_i\} \quad (\text{ただし}, 1 \leq i \leq n)$$

step f3

I. **step f2** で得られた $M_1', \cdots, M_n'$ を利用して新しいメッセージの集合 M を以下のように求める.

$$M = \bigcup_{S \subseteq I} \prod_{i \in S} M_i' \quad (I\text{は添字の集合})$$

ここで, $S = \{1, \cdots, s\}$ とした時に,

$$\prod_{i \in S} M_i' = \{(m_1, \cdots, m_s) \mid m_1 \in M_1', \ldots, m_s, \in M_s', \; k, l \in S, k \neq l \text{に対して } m_k \neq m_l\}$$

II. I で得られたメッセージの集合 M 中に, 矛盾する制約の組を含むメッセージを M から 削除する. そのメッセージの集合を $M_\wedge$ とする.

step f4

step f2, step f3 で得られたメッセージより, プロセス P が受け取れるメッセージの集合を M_P とすると, 次のように記述できる.

$$\begin{aligned} M_P &= (M_1 \cap M_1') \cup \cdots \cup (M_n \cap M_n') \cup M_\wedge \\ &= \{m_P^1, m_P^2, \cdots, m_P^h\} \end{aligned}$$

したがって，プロセス $P = P1 \| \cdots \| Pn$ は，

$$P \stackrel{def}{=} \{ \begin{array}{ccc} m_P^1 & \rightarrow & \langle resp(P, m_P^1), cont(P, m_P^1) \rangle \\ \vdots & & \vdots \\ \|m_P^h & \rightarrow & \langle resp(P, m_P^h), cont(P, m_P^h) \rangle \} \end{array}$$

と表記できる．ただし，メッセージ m_P^j $(1 \leq j \leq h)$ に対するプロセス P の応答メッセージ $resp(P, m_P^j)$ とコンティニュエーションのプロセス $cont(P, m_P^j)$ を 3.1節の方法で求める．

3.3 時相論理式による制約を用いた部分計算法

本節で示す部分計算法は，[7] で定められた部分計算法を一部変更しNGHCで記述される応答型プロセスに対応させたものである．

NGHC で記述されたプロセス $P = (W, G)$ が

$$P \stackrel{def}{=} \{\|_{i \in I}(m_i \rightarrow \langle resp(P, m_i), cont(P, m_i) \rangle)\}$$

で定義されているとき，プロセス P のメッセージについて成り立つ制約 p による部分計算を $Part(P, p)$ と表記する．本節では，$Part(P, p)$ を求める方法を示す．まず，プロセス $Q = (W', G')$ を

$$Q \stackrel{def}{=} Part(P, p)$$

と定義して，右辺を以下の規則を用いて変換して行く．

- **part 規則** ：

  制約 p が時相記号を含まない制約の場合

  $$Part(P, p) \Rightarrow part(P, p)$$

  ただし，$part(P, p)$ は，従来から与えられた方法による部分計算であり，以下のようにして求める．

  $$part(P, p) \Rightarrow \{\|_{i \in I}((m_i)_p \rightarrow \langle resp(P, (m_i)_p), cont(P, (m_i)_p) \rangle)\}$$

  ここで，$(m_i)_p$ は メッセージ m_i を制約 p で特殊化することを示す．

- **$\|$ 規則** ：

  制約 p がプロセス P の入力変数にのみ関係する制約で，かつ，プロセス Q から P への通信が存在しない場合

  $$Part(P \| Q, p) \Rightarrow Part(P, p) \| Q$$

- **and 規則** ：

  $$Part(P, p \wedge q) \Rightarrow Part(Part(P, p), q)$$

- **or 規則** ：

  制約 p, q が時相記号を含まない制約の場合

  $$Part(P, p \vee q) \Rightarrow Part(P, p) \sqcup Part(P, q)$$

  ただし，

  $P \stackrel{def}{=} \{\|_{i \in I}(m_i \rightarrow \langle resp(P, m_i), cont(P, m_i) \rangle)\}$

  $Q \stackrel{def}{=} \{\|_{j \in J}(m_j \rightarrow \langle resp(Q, m_j), cont(Q, m_j) \rangle)\}$

  である時，

  $$P \sqcup Q \stackrel{def}{=} \{\|_{i \in I}(m_i \rightarrow \langle resp(P, m_i), cont(P, m_i) \rangle \; \|_{j \in J}(m_j \rightarrow \langle resp(Q, m_j), cont(Q, m_j) \rangle)\}$$

- **X 規則** ：

  $$Part(P, \mathsf{X}p) \Rightarrow \{\|_{i \in I}(m_i \rightarrow \langle resp(P, m_i), Part(cont(P, m_i), p) \rangle)\}$$

- **G 規則** ：

  $\mathsf{G}p \equiv p \wedge \mathsf{X}\mathsf{G}p$ という性質から，X 規則と **and** 規則より求まる

  $$Part(P, \mathsf{G}p) \Rightarrow Part(\{\|_{i \in I}(m_i \rightarrow \langle resp(P, m_i), Part(cont(P, m_i), \mathsf{G}p) \rangle)\}, p)$$

- **畳み込み規則** ：

  $Part(P, p)$ を変換して行くうちに，制約 p によるプロセス $P1 = (W, G\theta)$ (すなわち，プロセス $P1$ はプロセス Pの具体例) の部分計算 $Part(P1, p)$ が現れた場合

  $$Part(P1, p) \Rightarrow Q\theta$$

- **終了規則** ：

  $$Part(P, p) \Rightarrow P$$

ここで，$\|$ 規則を導入した理由は，$\|$ 規則を適用できる制約で部分計算する場合は，並行に実行されるプロセスを 3.2 節の方法で関数表現する必要がないと考えられるからである．

プロセス P のメッセージに成り立つ制約 p における部分計算 $Part(P, p)$ が終了するのは，上記の **part** 規則を適用して従来の部分計算に帰着されるか，畳み込み規則が適用されるか，あるいは，終了規則が適用されるかして終了する．

3.4 関数表現のプロセスを実現する NGHC プログラム

本節では，3.3 節の部分計算で求めた 関数表現のプロセスを実現するNGHCプログラムに変換する方法について述べる．

NGHC で記述されたプロセス P が，制約 p で部分計算され新たなプロセス Q になったとする．すなわち，$Q \stackrel{def}{=} Part(P,p)$ である．プロセス Q が

$$Q \stackrel{def}{=} \{[]_{i\in I}(m_i \rightarrow \langle resp(Q,m_i), cont(Q,m_i)\rangle)\}$$
$$(1 \leq I \leq n,\ cont(Q,m_i) = (W_i, G_i))$$

で関数表現されているとすると，プロセス Q を実現する NGHC のプログラムは，G_{new} とそれを定義するプログラム節 W_{new} を W_i に加えたプログラム節で実現される．ここで，W_{new} と G_{new} を以下のように定める．

- W_{new} :

  W_{new} は，関数表現形式のメッセージ m_i を NGHC プログラムの入力制約 I_i，応答メッセージ $resp(Q_i,m_i)$ を出力制約 O_i，そして，コンティニュエーションのゴール G_i をボディゴールに対応させて得られる節の集合である．すなわち，

  $$\bigcup_{i\in I}\{(p_{new}(Z_1,\ldots,Z_k) :- m_i \mid resp(Q,m_i) \rightarrow G_i.)\}$$

  である．ただし，ヘッド $p_{new}(Z_1,\ldots,Z_k)$ は，W_i に存在しない新たな述語記号に k 個の引数を適用したもの．ここで，変数 $Z_1,\ldots,Z_k$ は，

  $$\{Z_1,\ldots,Z_k\} = Var(m_i \cup resp(Q,m_i) \cup G_i) - Var(m_i \cap resp(Q,m_i)) - Var(m_i \cap G_i) - Var(resp(Q,m_i) \cap G_i)$$

  で定められる．ただし，$1 \leq i \leq n$

- G_{new} :

  W_{new} 中の節のヘッドをゴールにしたもので表される．すなわち，

  $$G_{new} \ :\ p_{new}(Z_1,\ldots,Z_k).$$

4 結論

本稿では，並行論理型言語 NGHC で記述された応答型プロセスが受け取るメッセージに成り立つ制約条件を時相論理式で表し，それを利用してプロセスをより動作効率の良いプロセスへ特殊化する方法を述べた．この方法により，起動時の初期入力のみならず，実行の途中でプロセスが受け取るメッセージについて成り立つ制約を利用しての部分計算が可能となった．

今後の課題としては，(NGHC プログラム ⇔ 関数表現) の変換技法の妥当性について議論する必要がある．すなわち，ここで述べた (NGHC プログラム ⇔ 関数表現) の変換技法がそれぞれのセマンティクスを変えることがないもの，ということが示されるべきである．そのために，NGHC と関数表現に共通な形式的意味論を与えることが必要である．また変換技法を実現し，この部分計算法の効率改善の度合を定量的に評価することも考えなければならないであろう．

さらに，NGHC プログラムと時相論理式から，部分計算された NGHC のプログラムを直接求める方法を構築することも興味深い．また [8] では，並行プロセス合成のために部分評価を応用する手法が提案されているが，本稿で述べた部分計算法が NGHC のプログラムの合成に用いることができるかどうかを検討するという課題も考えられる．

謝辞

本研究を行うにあたり，励ましと御指導を与えていただきました山崎教授に感謝いたします．また，熱心な御討論をいただいた知能情報処理工学研究室の皆様に感謝します．

参考文献

[1] M. Falaschi, M. Gabbrielli, G. Levi, and M. Murakami, "NESTED GUARDED HORN CLAUSES", *International Journal of Foundations of Computer Science Vol.1 No.3*, pp.249-263, (1990)

[2] 淵, 古川, 溝口, "プログラム変換", 共立出版, (1987)

[3] 淵, 古川, 溝口, "並列論理型言語 GHC とその応用", 共立出版, (1987)

[4] H.Fujita, "FGHC Partial Evaluator as a General Purpose Parallel Compiler", *ICOT Reseach Center, Technical Report*, (1988)

[5] 松本, 内平, 本位田, "時相論理とその応用", 情報処理学会, Vol.30 No.6, pp.651-657, (1989)

[6] 村上, "応答型並行プログラムの部分評価法", 情報処理学会, プログラミング-言語・基礎・実践-研究会 92-PRG-7-2, (1992)

[7] 村上, "応答型並行プロセスのための時相論理式による制約を用いた部分計算法", 日本ソフトウェア科学会研究報告, (1992)

[8] 村上, "並行プロセス合成のための部分評価法の応用", 日本ソフトウェア科学会研究報告, 第 10 回論文集, B10-3, (1993)