

〔研究〕

General Tracer の研究(II)--G.H.C.のabstract interpreter--

A Study of General Tracer (II) --Abstract Interpreter of G.H.C.--

大谷木 重 夫
S. OHYAGI

We use Guarded Horn Clause (G.H.C.) as means to describe Pascal-like Languages. Then there arises a problem what semantics G.H.C. has. G.H.C. is a subset of Prolog. So you may think there is no problem. However G. H. C. has very simple control structure compared to Prolog and it must be shown explicitly. In this paper a property of G. H. C. is derived from the description of Prolog and then abstract interpreter of G.H.C. is given.

§ 1 はじめに

プログラミング言語、特に Pascal 系言語に対しては λ -Calculus により抽象的に意味を与えることができる。更にそれは Scott の意味付けにより、 $P()$ の関数と解釈される。Prolog も λ -Calculus を媒介にして意味を与られている。このように λ -Calculus による意味記述は数学的には大変望ましいものである。

しかしプログラミング言語の意味を Prolog のような論理型の言語で与えると、プログラムを論理型データベースに変換することができ、例えば質疑応答システムをドッキングさせられるといった便利な面がある。それだけではなく Prolog では local variable の情報を捨てなければ過去の情報を保存しており、Prolog で定義された言語上のトレーサーを Prolog で動かせば back-ward にももどれるトレーサーができる可能性がある。

このような実用的見地からプログラム言語の意味を Prolog のような論理型言語で定義するのが望ましいが、通常の Pascal 系言語を定義するには Prolog でなくても G.H.C. で十分であり、G.H.C. の方が制御構造がはるかに簡単になるので、G.H.C. を採用したい。

G.H.C. の semantics は Prolog の意味記述から導かれると考えられるが、その簡単な制御構造は G.H.C. のプログラムの性質より導かれるもので本論文ではその性質をあきらかにするとともに G.H.C. の簡単な制御構造を Abstract interpreter により示したい。

§ 2 G.H.C. の定義

Guarded Horn Clause は次のような Backus Naur Form により定義される。

```
Program ::= Clause_groups
Clause_groups ::= Clause_group | Clause_groups Clause_group
Clause_group ::= Head Clauses
Head ::=  $\Lambda$ 
Clauses ::= Clause | Clause Clauses
Clause ::= Clause_head :- Guard_part, !,
Execution_block.
Clause_head ::= Predicate_func
Guard_part ::= terms
Execution_block ::= terms
terms ::=  $\Lambda$  | term | term terms
term ::= Predicate_func
```

Program は Clause 群の集まりであり、Clause 群は Clause の集まりである。Clause は Clause_head と Guard part ！実行部 からなっており、それぞれは term 及び term の並びである。term は述語及びスコラム関数からなっている。

例えばリストの最小値を与える min は

```
min([x],x) :-!.
min([x,y],y) :- (x>=y),!.
```


$$=m(s) \left[\left[\text{And_part} \right] \right] (n) (\text{next}^{k_0}(s))$$

より

$$= \text{set}(!\text{fail}) (\text{next}^{k_0}(s))$$

$$\text{Clause_exhausted}(m(s) \left[\left[\text{Clause} \right] \right] (n) (\text{next}^{k_0}(s)))$$

が成立するから

$$m(s) \left[\left[\text{Clause} \text{ Clauses}' \right] \right] (n) (\text{next}^{k_0}(s))$$

$$= m(s) \left[\left[\text{Clause} \right] \right] (n) (\text{next}^{k_0}(s))$$

従って

$$m(s) \left[\left[\text{Clauses} \right] \right] (n) (s)$$

$$= \text{set}(!\text{fail}) (\text{next}^{k_0}(s))$$

以上より次の定理が得られる。

[定理 1]

全ての Clause の And_part について

$$\text{And_part} \xrightarrow{*} \text{terms},!$$

を仮定すれば

$$\text{i) } m(s) \left[\left[\text{Clauses} \right] \right] (1) (s)$$

$$= m(s) \left[\left[\text{Clauses} \right] \right] (1) (\text{next}^{k_0}(s))$$

$$\text{ii) } m(s) \left[\left[\text{Clauses} \right] \right] (n) (s)$$

$$= \text{set}(!\text{fail}) (\text{next}^{k_0}(s))$$

但し $n \geq 2$

$$\text{iii) } m(s) \left[\left[\text{Clauses} \right] \right] (n) (s)$$

$$= \text{next}^l(s)$$

但し l は Clause の数

ここで k_0 は

$$k_0 = \min \{ k | \text{Suc}(\text{head}(\text{Clause}))(\text{cur}(\text{next}^k(s))) \}$$

[定義 1]

$$\text{And_part}_1 \xrightarrow{*} \text{terms}_1$$

$$\text{And_part}_2 \xrightarrow{*} \text{terms}_2$$

のとき

$$m(s) \left[\left[\text{terms}_1 \mid \right] \right] (n) (s)$$

$$\sim m(s) \left[\left[\text{terms}_2 \mid \right] \right] (n) (s)$$

とは $m(s) \left[\left[\text{And_part}_1 \right] \right] (n) (s)$

$$= m(s) \left[\left[\text{And_part}_2 \right] \right] (n) (s)$$

のことに定義する。

次の定理が証明すべき定理である。

[定理 2]

$$m(s) \left[\left[\text{terms},!, \text{term}_1,!, \dots, \text{term}_k,! \right] \right] (n) (s)$$

$$\sim m(s) \left[\left[\text{terms},!, \text{term}_1, \dots, \text{term}_k \right] \right] (n) (s)$$

(証明)

まず

$$m(s) \left[\left[\text{terms},!, \text{term}_1,!, \dots, \text{term}_k,! \right] \right] (n) (s)$$

$$\sim m(s) \left[\left[\text{terms},!, \text{term}_1, \text{term}_2,!, \dots, \text{term}_k,! \right] \right] (n) (s)$$

を示す。

順方向へ進むかぎり! は単に通過するだけだから

$n=1$ のときは $m(s) \left[\left[\text{terms},!, \text{terms}' \right] \right]$ の定義から

$$m(s) \left[\left[\text{terms},!, \text{term}_1,!, \text{term}_2,!, \dots, \text{term}_k,! \right] \right] (1) (s)$$

$$\sim m(s) \left[\left[\text{terms},!, \text{term}_1, \text{term}_2,!, \dots, \text{term}_k,! \right] \right] (1) (s)$$

は明らか。

$n \geq 2$ のときには

$$!\text{fail}(m(s) \left[\left[\text{terms},!, \text{term}_1,!, \text{term}_2,!, \dots, \text{term}_k,! \right] \right] (n) (s))$$

が成立するから

$$\text{And_part}_1 \xrightarrow{*} \text{terms},!, \text{term}_1,!, \text{term}_2,!, \dots, \text{term}_k,!$$

とすると

$$m(s) \left[\left[\text{And_part}_1 \right] \right] (n) (s) = \text{set}(!\text{fail})(s).$$

$$\text{And_part}_2 \xrightarrow{*} \text{terms},!, \text{term}_1, \text{term}_2,!, \dots, \text{term}_k,!$$

とする。

$$\text{Suc_process_status}(m(t) \left[\left[\text{term} \mid \text{terms}' \right] \right] (r+1) (m(s)))$$

$$\left[\left[\text{terms},! \mid \text{term terms}' \right] \right] (k) (s)$$

但し $\text{terms}' = \text{term}_2,!, \dots, \text{term}_k,!$

$k =$ これまでの成功通過の回数

$r =$ 最近の成功通過の回数

が成立するとすると

$$k=1, r \geq 1.$$

$\text{Clause_group}(\text{term} \xrightarrow{*} \text{terms},!)$ を仮定すると定理 1 より

$$m(s) \left[\left[\text{Clause_group}(\text{term}) \right] \right] (r+1) (s')$$

$$= \text{set}(!\text{fail})(\text{next}^{k_0}(s'))$$

であるから

$$m(t) \left[\left[\text{term} \mid \text{terms}' \right] \right] (r+1) (s')$$

$$= \text{decrement_index}(\text{set}(\text{fail})(s'))$$

これは矛盾である。定義より

$$m(s) \left[\left[\text{terms},!, \text{term}_1, \text{terms}' \right] \right] (n) (s)$$

$$= m(s) \left[\left[\text{terms},! \mid \text{term}_1 \text{ terms}' \right] \right] (k+1) (s)$$

これから

$$m(s) \left[\left[\text{And_part}_2 \right] \right] (n) (s)$$

$$= \text{set}(!\text{fail})(s)$$

これより

$$m(s) \left[\left[\text{terms},!, \text{term}_1,!, \dots, \text{term}_k,! \right] \right] (n) (s)$$

$$\sim m(s) \left[\left[\text{terms},!, \text{term}_1, \text{term}_2,!, \dots, \text{term}_k,! \right] \right] (n) (s)$$

が成立。

次に

$$m(s) \left[\left[\text{terms},!, \text{term}_1,!, \dots, \text{term}_{l-1},! \mid \dots, \text{term}_k,! \right] \right] (n) (s)$$

$$\sim m(s) \left[\left[\text{terms},!, \text{term}_1, \dots, \text{term}_{l-1} \mid \dots, \text{term}_k,! \right] \right] (n) (s)$$

の成立を仮定する。

$\text{terms}'' = \text{terms},!, \text{term}_1, \dots, \text{term}_{l-1},!$ とおくと上記より

$$m(s) \left[\left[\text{terms},!, \text{term}_1,!, \dots, \text{term}_l,! \mid \dots, \text{term}_k,! \right] \right] (n) (s)$$

$$\sim m(s) \left[\left[\text{terms},!, \text{term}_1,!, \dots, \text{term}_{l-1},!, \text{term}_l \mid \right. \right.$$

$$\left. \dots, \text{term}_k, \dots, ! \right] (n) (s)$$

$$\text{Suc_process_status}(m(t) \left[\left| \text{term}_t \mid \text{term}_{t+1},!, \dots, \text{term}_k,! \right| \right] \\ (r+1)(m(s) \left[\left| \text{terms},!, \text{term}_1,!, \dots, \text{term}_{t-1},! \mid \text{term}_t, \text{term}_{t+1},!, \dots, \text{term}_k,! \right| \right](k)(s))$$

の場合

$$= m(t) \left[\left| \text{term}_t \mid \text{term}_{t+1},!, \dots, \text{term}_t! \right| \right] (r+1)(\\ m(s) \left[\left| \text{terms},!, \text{term}_1,!, \dots, \text{term}_{t-1},! \mid \text{term}_t, \text{term}_{t+1},!, \dots, \text{term}_k,! \right| \right] (k)(s))$$

いま

$$m(s) \left[\left| \alpha!, \beta \mid \text{term},!, \gamma \right| \right] \\ \sim m(s) \left[\left| \alpha!, \beta \mid \text{term } \gamma \right| \right]$$

を仮定すると

$$\sim m(t) \left[\left| \text{term}_t \mid \text{term}_{t+1},!, \dots, \text{term}_k,! \right| \right] (r+1)(\\ m(s) \left[\left| \text{terms},!, \text{term}_1,!, \dots, \text{term}_{t-1},! \mid \text{term}_t, \text{term}_{t+1},!, \dots, \text{term}_k,! \right| \right] (k)(s)) \\ \sim m(t) \left[\left| \text{term}_t \mid \text{term}_{t+1},!, \dots, \text{term}_k,! \right| \right] (r+1)(\\ m(s) \left[\left| \text{terms},!, \text{term}_1,!, \dots, \text{term}_{t-1},! \mid \text{term}_t,!, \text{term}_{t+1},!, \dots, \text{term}_k,! \right| \right] (k)(s))$$

帰納法の仮定より

$$\sim m(t) \left[\left| \text{term}_t \mid \text{term}_{t+1},!, \dots, \text{term}_k,! \right| \right] (r+1)(\\ m(s) \left[\left| \text{terms},!, \text{term}_1,!, \dots, \text{term}_{t-1},! \mid \text{term}_t,!, \text{term}_{t+1},!, \dots, \text{term}_k,! \right| \right] (k)(s))$$

再び

$$m(s) \left[\left| \alpha, \beta!, \mid \text{term},!, \gamma \right| \right] \sim m(s) \left[\left| \alpha,!, \beta \mid \text{term } \gamma \right| \right]$$

を使って

$$\sim m(t) \left[\left| \text{term}_t \mid \text{term}_{t+1},!, \dots, \text{term}_k,! \right| \right] (r+1)(\\ m(s) \left[\left| \text{terms},!, \text{term}_1,!, \dots, \text{term}_{t-1},! \mid \text{term}_t, \text{term}_{t+1},!, \dots, \text{term}_k,! \right| \right] (k)(s))$$

定義より

$$\sim m(s) \left[\left| \text{terms},!, \text{term}_1,!, \dots, \text{term}_{t-1},! \mid \text{term}_t, \text{term}_{t+1},!, \dots, \text{term}_k,! \right| \right] (n)(s)$$

$$\text{Fail_process_status}(m(s) \left[\left| \text{terms},!, \text{term}_1,!, \dots, \text{term}_{t-1},! \mid \text{term}_t, \text{term}_{t+1},!, \dots, \text{term}_k,! \right| \right] (k+1)(s))$$

のとき

$$m(s) \left[\left| \text{terms},!, \text{term}_1,!, \dots, \text{term}_{t-1},! \mid \text{term}_t, \text{term}_{t+1},!, \dots, \text{term}_k,! \right| \right] (k+1)(s) \\ \sim m(s) \left[\left| \text{terms},!, \text{term}_1,!, \dots, \text{term}_{t-1},! \mid \text{term}_t,!, \text{term}_{t+1},!, \dots, \text{term}_k,! \right| \right] (k+1)(s) \\ \sim m(s) \left[\left| \text{terms},!, \text{term}_1,!, \dots, \text{term}_{t-1},! \mid \text{term}_t,!, \text{term}_{t+1},!, \dots, \text{term}_k,! \right| \right] (k+1)(s) \\ \sim m(s) \left[\left| \text{terms},!, \text{term}_1,!, \dots, \text{term}_{t-1},! \mid \text{term}_t, \text{term}_{t+1},!, \dots, \text{term}_k,! \right| \right] (k+1)(s)$$

より

$$\text{Fail_process_status}(m(s) \left[\left| \text{terms},!, \text{term}_1,!, \dots, \text{term}_{t-1},! \mid \text{term}_t, \text{term}_{t+1},!, \dots, \text{term}_k,! \right| \right] (k+1)(s))$$

で

$$m(s) \left[\left| \text{terms},!, \text{term}_1,!, \dots, \text{term}_t,! \right| \right]$$

$$\text{term}_{t+1},!, \dots, \text{term}_k,! \left| \right] (n)(s)$$

$$\sim m(s) \left[\left| \text{terms},!, \text{term}_1,!, \dots, \text{term}_{t-1},! \mid \text{term}_t, \text{term}_{t+1},!, \dots, \text{term}_k,! \right| \right] (n)(s)$$

が成立する。

それ以外の場合

$$m(s) \left[\left| \text{terms},!, \text{term}_1,!, \dots, \text{term}_{t-1},! \mid \text{term}_t,!, \text{term}_{t+1},!, \dots, \text{term}_k,! \right| \right] (n)(s) \\ \sim m(s) \left[\left| \text{terms},!, \text{term}_1,!, \dots, \text{term}_{t-1},! \mid \text{term}_t, \text{term}_{t+1},!, \dots, \text{term}_k,! \right| \right] (n)(s) \\ \sim m(t) \left[\left| \text{term}_t \mid \text{term}_{t+1},!, \dots, \text{term}_k,! \right| \right] (1)(\\ m(s) \left[\left| \text{terms},!, \text{term}_1,!, \dots, \text{term}_{t-1},! \mid \text{term}_t, \text{term}_{t+1},!, \dots, \text{term}_k,! \right| \right] (1)(s)) \\ \sim m(t) \left[\left| \text{term}_t \mid \text{term}_{t+1},!, \dots, \text{term}_k,! \right| \right] (1)(\\ m(s) \left[\left| \text{terms},!, \text{term}_1,!, \dots, \text{term}_{t-1},! \mid \text{term}_t, \text{term}_{t+1},!, \dots, \text{term}_k,! \right| \right] (1)(s)) \\ \sim m(s) \left[\left| \text{terms},!, \text{term}_1,!, \dots, \text{term}_{t-1},! \mid \text{term}_t, \text{term}_{t+1},!, \dots, \text{term}_k,! \right| \right] (n)(s)$$

以上から

$$m(s) \left[\left| \alpha,!, \beta \mid \text{term},!, \gamma \right| \right] (n)(s) \\ \sim m(s) \left[\left| \alpha,!, \beta \mid \text{term } \gamma \right| \right] (n)(s)$$

を証明すればよい。

[補題]

$$m(s) \left[\left| \alpha,!, \beta \mid \text{term},!, \gamma \right| \right] (n)(s) \\ \sim m(s) \left[\left| \alpha,!, \beta \mid \text{term } \gamma \right| \right] (n)(s)$$

(証明)

n=1 のとき

$$\text{Suc_process_status}(m(s) \left[\left| \alpha,!, \beta \mid \text{term},!, \gamma \right| \right] (n)(s))$$

であれば $m(s) \left[\left| \alpha,!, \beta \mid \text{term},!, \gamma \right| \right] (n)(s)$ は $\alpha,!, \beta$ を通過した時点での状態の値であるから $\text{term},!, \gamma$ には関係なく決まり従って

$$m(s) \left[\left| \alpha,!, \beta \mid \text{term},!, \gamma \right| \right] (n)(s) \\ \sim m(s) \left[\left| \alpha,!, \beta \mid \text{term } \gamma \right| \right] (n)(s)$$

となることは明白である。

$$\text{Fail_process_status}(m(s) \left[\left| \alpha,!, \beta \mid \text{term},!, \gamma \right| \right] (n)(s))$$

であれば $m(s) \left[\left| \alpha,!, \beta \mid \text{term},!, \gamma \right| \right] (n)(s)$ は fail で終わり、これまた $\text{term},!, \gamma$ に関係なくこの状態が決まるから

$$m(s) \left[\left| \alpha,!, \beta \mid \text{term},!, \gamma \right| \right] (n)(s) \\ m(s) \left[\left| \alpha,!, \beta \mid \text{term } \gamma \right| \right] (n)(s)$$

n≥2 のとき

βの長さについての帰納法をもちいる。

|β|=1 のとき

定義より

!Fail_process_status(
 m(s) [[α , ! γ]](n)(s))

が成立する。

| $\beta \mid \leq u$ で任意の δ に対し

!Fail_process_status(
 m(s) [[α , ! $\beta \mid \delta$]](n)(s))

が成立するとする。

$\delta = \text{term } \delta'$ にとると

!Fail_process_status(
 m(s) [[α , ! $\beta \mid \text{term } \delta'$]](n)(s))

より

m(s) [[α , ! $\beta \text{ term} \mid \delta'$]](n)(s)
 = m(s) [[α , ! $\beta \mid \text{term } \delta'$]](n)(s)

が成立。

従って

!Fail_process_status(
 m(s) [[α , ! $\beta \text{ term} \mid \delta'$]](n)(s))

が成立。

帰納法が完結して

!Fail_process_status(
 m(s) [[α , ! $\beta \mid \delta$]](n)(s))

が成立。

故に $\delta = \text{term}, !, \gamma$ 及び $\delta = \text{term } \gamma$ にとって

m(s) [[α , ! $\beta \mid \text{term}, !, \gamma$]](n)(s)
 \sim m(s) [[α , ! $\beta \mid \text{term } \gamma$]](n)(s)

が成立。

Q.E.D.

§ 4 G.H.C. の Abstract Interpreter

まず簡単な interpreter を pascal 風を書いてみよう。

いまプログラムが

$d_1 \dots d_n$.

$d_i = 'p_i :- p_1^i, \dots, !, \dots, p_s^i$

と与えられているものとする。 d_i に対しその Head p_i の述語名を $\text{head}(d_i)$, d_i に対し同じ head をもつ次の Clause を得る関数を $\text{next}(d_i)$ 手前の Clause を $\text{pre}(d_i)$ とする。又 p_{ix}^i を $p(d_i, i, x)$, p_i を $p(d_i)$ と書くことにする。

Stack は一本であり, Stack の top は current state を成している。 Stack の data format は

```
process_status : <Clause_name, stack_pointer_
  to_caller, subprocess_status>
subprocess_status : list_of <restart_index,
  stack_pointer, logical_state_1,
  logical_state_2>
```

述語 p を証明する過程をプロセスと呼ぶことにしよう。

すると process_status はこの process に最初に適用された Clause 名, この process を呼んだ process への pointer, この process の subprocess の状態記述からなる。 subprocess の状態記述は次に適用すべき Clause 名 (=restart_index) この subprocess の process_status の記述への pointer 及び変数の binding 情報を表す logical_state の記述からなっている。

入力文を $p(f(x))$ とすると process_status は Clause_name $\leftarrow p(f(x))$, caller はないから自分自身即ち

Stack_pointer_of_caller = 1.1

subprocess_status は

logical_state_1 = $p(x)$.

logical_state_2 = \perp .

restart_index = first(p).

stack_pointer = \perp .

であるから初期状態は

Stack : $\ll \leftarrow p(f(x)), 1.1, \ll p(x), \perp, \text{first}(p), \perp \gg \gg \gg$

である。

いま stp で Stack のある位置を表すものとする。このとき stp 番地の subprocess_status への index ix が与えられているとして Stack(stp) は

$\ll \ll d, n, \ll \ll \dots, \ll 1_1, 1_2, d', p, \dots \ll \gg \gg \gg$

└──────────┬──────────┘
 ix

である。

最初の Clause を Clause 群 q から選んで進むルーチンを $m(q, \text{stp}, ix)$ とするとき main ルーチンは

main:

initial_set;

RTN(m(p, 1, 1));

end;

である。

ルーチン $m(q, \text{stp}, ix)$ は

sub1: m(q, stp, ix)

Do

if $q = !$ then RTN(success);

$d_1 \leftarrow \text{Stack}(\text{stp}).\text{Clause_name};$

$d \leftarrow \text{Stack}(\text{stp}).\text{subprocess_status}(ix).$

restart_index;

Stack(stp).subprocess_status(ix).

restart_index \leftarrow next(d);

if $p(d) \cap p(d, ix)(p^{-1}(\text{Stack}(\text{stp}).$

subprocess_status(ix).logical_state1))

$\neg = \phi$

then

```

begin
  if m(d,stp,ix) = success
    then RTN(success);
  end;
end;
untill next(d) =  $\phi$  ;
RTN(fail);
end;

```

と定義される。即ち、まず $\text{first}(\text{tail}(d_i))$ なる Clause を適用して unification がとれるかどうか調べる。 unification がとれればこの Clause に従って subprocess を起動していく。 unification がとれなければ次々と進み最初の unification がとれる Clause を選択してその Clause に従って subprocess を起動していく。その結果が成功であれば成功、失敗であれば次の selection を求めていく。

次に Clause の適用とそれが生み出す process の管理を行うルーチンを定めるとしよう。

Clause 名^g

$d = 'p :- p_1, \dots, p_i, \dots, p_r'$

と書かれるものとする。このとき

$\text{length}(d) = r$

$\text{Pred_name_of}(p_i) = p_i$ の述語名か!

= 未定

$p(d, i) = p_i$

$\text{first}(!) = !$

なる関数を定める。すると目的とするルーチンは

```

sub2 : m(d,stp,ix)
  d1 <- Stack(stp).Clause_name;
  if length(d)=0
  then
  begin
    Stack(stp).subprocess_status(ix).
    logical_state2
    ←  $p \ p(d_1, ix)^{-1} \ (p(d) \cap (d_1, ix) p^{-1} ($ 
      Stack(stp).subprocess_
      status(ix).logical_state1));
  end;
  Push(<d,stp,ix,
    <<p p(d)-1 (p(d) ∩ p(d1, ix) p-1(
      Stack(stp).subprocess_status(ix).
      logical_state1))
      ,  $\perp$ , first(Pred_name_of(p1)),  $\perp$ >>,
    << $\perp$  ,  $\perp$ , first(Pred_name_of(p2)),  $\perp$ >>,
    .
    .
    .
    << $\perp$  ,  $\perp$ , first(Pred_name_of(pr)),  $\perp$ >>>);

```

```

stp <- length(Stack);
ix <- 1;
r <- length(d);
While ix ≤ r
Do
  q <- head(Stack(stp).subprocess_status(ix).
    restart_index);
  if m(q,stp,ix)=fail
  then
  begin
    if ix < ix(d) then
    begin
      Pop(length(Stack)-stp);
      Pop;
      RTN(fail);
    end;
    if ix ≥ ix(d) then
    begin
      stp1.ix1 <- Stack(stp).
        stack_pointer_to_caller;
      Pop(length(Stack)-stp);
      Pop;
      Stack(stp1).subprocess_
        status(ix1).restart_
        index <-  $\phi$ ;
      RTN(fail);
    end;
  end;
  if ix < r then
  begin
    ix <- ix+1;
    Stack(stp).subprocess_status(ix).
    logical_state1
    <- Stack(stp).subprocess_status(
      ix-1).logical_state2;
  end;
end;
stp1.ix1 <- Stack(stp).stack_pointer_to_
  caller;
Stack(stp1).subprocess_status(ix1).
  logical_state2 <-
  pp(d1, ix1)-1 p(d) p-1(Stack(stp).
  subprocess_status(r).
  logical_state2)
  ∩ Stack(stp1).subprocess_
  status(ix1).logical_state1;

```

```
RTN(success);  
end;
```

§ 5 おわりに

Guarded Horn Clauses [Ueda 1987] は並列計算機用の言語として設定されたわけであるが、その記述力の高さからいって sequential machine 上の言語としても通用するものと考えられる。本論文では G . H . C . の抽象的な interpreter を作成したが、これに従って具体的な interpreter が作成されることがのぞましい。

謝 辞

本論文は情報アーキテクチャ部分散システム研究室で筆者が行った研究をまとめたものである。研究室長であった塚本亨治氏及び研究室の皆様には公私にわたり大変お世話になりました。ここに厚く御礼申し上げます。また論文の作成に当たりましては大蒔情報アーキテクチャ部長にアドバイスを頂きました。また内容につきましては木下算譜言語論代表世話人にお世話になりました。

参 考 文 献

- 1) K.Ueda "Guarded Horn Clauses" Concurrent Prolog M.I.T. 1 (MIT, 1987) 140-156.
- 2) D.Scott "λ-Calculus and recursion theory" Proc. 3rd Scandinavian Logic Symp. (North Holland) 154-193.
- 3) S.Ohyagi "General Tracer の研究(I) -denotational description of Prolog-" 本シリーズ.

(1998.3.12 受付)