自然言語処理 74－1
（1989. 9. 21）

意味計算II

－制約付変数とユーザー定義オブジェクト－

赤間　清
北海道大学工学部情報工学科

自然言語の理解過程を解釈の逐次更新の過程として明快に実現するには，文の各部分の解釈を適切に表現し，高速に更新しなければならない．prolog が提供する基本オブジェクト（アトム，変数）だけではその目的のためには不十分である．PAL が prolog に追加した集合束縛変数は，文の解釈の表現，更新に重要な役割を果す新しいオブジェクトである．それは単純で有用な制約を高速に扱うことによって，自然言語処理に貢献する．本論文では，一般的な制約によって規定される新しいオブジェクトをユーザーが自由に定義できる枠組みを導入する．新しいオブジェクトは制約付き変数とよばれる．それは集合束縛変数を含むばかりか，自然言語処理の対象領域に適した多くのオブジェクトを自由に表現可能であり，それを利用すれば，自然言語処理をさらに明快にプログラミングすることができる．

A Semantic Calculus II

Restricted Variables and User defined Objects

Kiyoshi Akama
Dept. of Information Eng., Faculty of Eng., Hokkaido University

In this paper we introduce restricted variables into the extended prolog PAL with set bound variables (SBVs). They enable us to define freely new objects using more general type of constraints than the membership constraints of SBVs. Defining useful objects that we may have in mind in the process of understanding natural language sentences, we can write more elegant programs for natural language processing than we use only built-in objects (atoms and variables) in prolog or SBVs in PAL.

1．　まえがき

自然言語の理解過程を解釈の逐次更新の過程[赤間 89b]として明快に実現するには，文の各部分の解釈を適切に表現し，それを高速に更新しなければならない．通常の prolog が提供する基本オブジェクト（アトム，変数）は低レベルのオブジェクトであり，解釈を表現し，高速に更新するための基本要素としては不十分である．

PAL が prolog に追加した集合束縛変数，とくに，C型の集合束縛変数は「概念」に対応するオブジェクトである．それは，単純で有用な制約を高速に扱うことによって，自然言語処理において文の解釈の表現，更新に重要な役割を果すことができる[赤間 89a]．

C型の集合束縛変数の重要な点の１つは，概念とその階層構造を指定することによって，ユーザーが自然言語処理の対象領域に応じたオブジェクトを導入できる点である．その意味でC型の集合束縛変数はユーザーが定義するオブジェクトである．これに対して，通常の prolog で使えるオブジェクト（アトム，変数）はすべて組込のものばかりであり，そこではユーザーは与えられたオブジェクトだけを使って自然言語処理をプログラミングしなければならない．

この差は重大である．自然言語処理は人間の思考に出現する対象を扱う必要がある．人間の頭の中では，赤い服や，親切な人や，赤い服を着た太郎や，恋人の紀子や，愛犬のポチや，かわいいペットなど，さまざまな対象のイメージが生き生きと動いている．自然言語処理を高度化するためには，おそらくそれらのイメージを模擬するような，オブジェクトを導入していくことが必要であろう．

prolog では複雑な対象をコンス（項）で表現する．しかしコンスはここで求めるようなオブジェクトにはならない．仮にコンスで対象を組み上げたとしても，それは述語呼出しのユニフィケーションの際には，コンスのルールに従って動くにすぎない．自然言語処理において必要なオブジェクトの多くはコンスではない．もし人間をコンスで「表現」したなら，ユーザーが述語を呼出して，いちいちそのコンスを人間のルールに従って動かしてやらねばならない．

オブジェクトに関して重要な点が少なくとも２つある．１つは，オブジェクトが自分に相応しい行動をとることである．紀子のイメージは「かわいい娘」にはなるが犬や机にはならない．イメージは，あたかも自分が何物か知っているかのように，現実世界のルールにそって変化する．２つ目は，オブジェクトはユーザーが陽に述語を使って変形しなくても，述語呼出しの際のユニフィケーションで陰に変更されるべきことである．これは，人間のイメージが自分で変化していくことに類似している．イメージを変化させるために，人間が意識的に変更ルーチンを起動する必要は必ずしもないのである．

しかし自然言語処理で必要なオブジェクトの詳細については今後の研究で段階的に明らかにして行かなければならない．したがってユーザーが必要に応じて自由に新しいオブジェクトを導入できる枠組みが自然言語処理の研究を加速する上で重要と考えられる．

本論文では，ユーザーが一般的な制約によって規定される新しいオブジェクト（制約付き変数）を自由に導入できる枠組みを導入する．制約付き変数は豊かな表現力を持つ．したがってユーザーはいろいろなオブジェクトを自由に定義して，それを直接の対象とした論理プログラミングを行なうことができる．これは知識情報処理に対して，なかでも自然言語処理の研究と実用化に対して基礎的な役割を果すものである．

2．　制約付き変数

2．1　制約付き変数の定義

PAL の集合束縛変数は，変数ｖと集合ｓのペアである．集合ｓは変数ｖがｓの範囲の対象であるという制約を表現している．集合束縛変数の例を図１に示す．これらの例において，{fis

hing swimming}は fishing と swimming からなる集合を，[20 30]は20 以上 30 未満の数の集合を，(+ dog cat)はすべての犬または猫からなる集合を意味している．

```
F型  *hobby^{fishing swimming}
  釣，泳ぎのどちらかである *hobby
I型  *age^[20 30]
  20 以上 30 未満である *age
C型  *animal^(+ dog cat)
  犬または猫である *animal
```

図１　集合束縛変数の例

このように PALシステム version 4.9では，F型，I型，C型の集合束縛変数（マルチクラスを含む）が導入されている．集合束縛変数としては，F型，I型，C型だけでなくいろいろ考え得る．しかし version 4.9では，他の新しい集合束縛変数をユーザーが導入するための一般的枠組みは提供していない．また，いろいろな制約を扱いたいという観点から見れば，集合束縛変数は，その定義が示すように，「ある集合に属する」という型以外の制約を扱えないという限界を持つ．

PAL version 5.0 は，ユーザー定義述語によってユニフィケーションを制御する機構を提供している．これによって集合束縛変数は制約付き変数に一般化される．制約付き変数は上記の限界を取除くことができる．

制約付き変数は，変数ｖと制約ｓのペアであるが，集合束縛変数とちがって制約ｓは任意のものが許される．また制約ｓの内部表現もユーザーが選択する任意のＳ式によって行なうことができる．

ユーザーは次の２つの述語を定義することによって，制約付き変数を導入する．

Ointersect … ２つの制約付き変数の間の unification を定義する

Omember … アトムやコンスと制約付き変数との unification を定義する．

２つの制約付き変数 *1~s1 と*2~s2 のユニフィケーションは，述語 Ointersect は，

```
(Ointersect s1 s2 *s)
```

という述語呼出しが成功するとき成功し，両者はそれによって求まる *s を制約とする制約付き変数 *1~*sに変化する．変数でないＳ式（アトムまたはコンス）x と制約付き変数 *~s のユニフィケーションは，

```
(Omember x s)
```

という述語呼出しが成功するとき成功し，* は x となる．

これらの述語によって制約付き変数の本質的な部分は定義できるが，PAL ではさらに次の述語をユーザーが指定できる．

Oprint … 制約付き変数の印字方法を指定する

Oprint を定義した場合，制約付き変数 *~s は，*~ という印字のあと，

```
(Oprint s)
```

で決る印字を行なう（Oprint が示す任意の実行が可能）．Oprint を定義しなければ，システムはデフォールトで princ という組込述語を用いる．このときは制約の内部表現がそのまま出力される．

２．２　制約付き変数の例

制約付き変数を説明するための最も判りやすいやり方の１つは，集合束縛変数を制約付き変数で実現する方法を示すことであろう．I型の集合束縛変数を制約付き変数で実現した例を図２に示す．ここでは，PALで *age^[20 30]のように表現したＩ型集合束縛変数を，制約の内部表現 (In 20 30) を用いて*age~(In 20 30)のように表わすと約束している．また，:= は数の計算をする述語，format は変換（/d は数字出力用の変換）をして印字する述語である．この定義の下で，

```
(and (= *1~(In 2 5) *2~(In 3 8))
     (print *1))
```

を実行すれば，*1~[3 5] が印刷される．

```
(as (Ointersect (In *fromx *tox)
                (In *fromy *toy)
                (In *fromz *toz))
    (:= *fromz (max *fromx *fromy))
    (:= *toz   (min *tox *toy))
    (< *fromz *toz))

(as (Omember *x (In *from *to))
    (numberp *x)
    (<= from *x)
    (< *x *to))

(as (Oprint (In *from *to))
    (format "[/d /d]" *from *to))
```

図２　制約付き変数による
　　Ｉ型集合束縛変数の実現例

２．３　定オブジェクトの導入

制約付き変数を使って，変化しないオブジェクトを導入することもできる．たとえば，

```
(as (Ointersect (P *x *y)
                (P *x *y)
                (P *x *y)))
(as (Omember *a (P *x *y)) (false))
```

と定義すれば，*~(P 3 5) は，２項組(3 5)を表わす１つの定オブジェクトを表わす．*~(P 3 5)はコンス(3 5)とは違う．もしコンスなら（*x . *y）にマッチするが，*~(P 3 5) は(*x . *y) にマッチしない．つまり *~(P 3 5) はユニフィケーションによって内部が覗けないひと塊りのものである．

このような対象をユーザーが新しく導入することができることは，たとえば自然言語処理で次のような効用がある．「角の右の金を５－６に動かせ」という例文を考える．この文の PAL (version 4.9)/TALKにおける意味表現は，図３のようになる．このうち，

```
(object *1^kin  …)
```

の部分は，「角の存在する場所の右の場所にある金を」に対応する表現になっている．また，

```
(place *5^place  …)
```

の部分は，「ｘ座標が５，ｙ座標が６のペアデータによって決る場所に」に対応している．

図３は図４のシンタックスを満足している．

```
(place *5^place  …)
```

に出現するペアデータ：

```
(*6^pair (x 5) (y 6))
```

もまた図４のシンタックスを満たしている．しかしペアデータの部分は全体の意味表現とは異質のまとまりをなしており，図４のシンタックスに従うべきではない．このペアデータは，上で定義した制約付き変数を用いて

```
*6~(P 5 6)
```

と表現するほうが望ましい．

```
(*0^sentence
 (verb move)
 (mod 命令)
 (object *1^kin
         (==>place *2^place
                   (<==right *3^place (==>exist *4^kaku))))
 (place *5^place
        (==>data *6^pair (x 5) (y 6))))
```

図３　PAL(ver. 4.9)/TALK が出力する「角の右の金を５－６に動かせ」の意味表現
　　（注意　==>data　などは１つの atom であり，==>などには特別な意味はない）

```
〈対象〉 = （〈分子〉 ・ 〈関対リスト〉）
〈分子〉 = 〈アトム〉 | 〈集合束縛変数〉 | 〈通常変数〉
〈関対リスト〉 = （） | （〈関対〉 ・ 〈関対リスト〉）
〈関対〉 = （〈関係〉 ・ 〈対象〉）
〈関係〉 = 〈述語〉
```

図４　PAL/TALK における文，対象の表現

このことは，「コンス(a (b c))」に対する意味表現を考えればもっとよく理解できよう．この意味表現を，もし

```
(*1^cons
  (==>data *2^cons_data
           (value . (a (b c)))))
```

と書くなら，これは図４のシンタックスにあわない．図４ではコンスは〈分子〉になれない（こう決めた理由は紙面の都合上割愛する）．かといって，コンスデータを，上述のペアデータのように表現すれば，関係名が際限なく必要になる．ここは

```
(as (0intersect (C *x) (C *x) (C *x)))
(as (0member *a (C *x)) (false))
```

によって定義した制約付き変数を用いて，

```
*2~(C (a (b c)))
```

という風に書きたい．この表現はひとかたまりのコンスデータをうまく表現している．またコンスデータであることを先頭の C によって一瞬のうちに知ることができる．

以上のような表現を可能にするために，図４の〈分子〉の定義は，

```
〈分子〉 = 〈アトム〉
        | 〈集合束縛変数〉
        | 〈制約付き変数〉
        | 〈通常変数〉
```

と改訂しておけばよい．

1m と 100cm のように，同一のものに対して複数の表現が可能な場合がよくある．制約付き変数による定オブジェクトの表現は，このような表現の違いを吸収するためにも利用できる．たとえば，1mと 100cm をそれぞれ，

```
1m  …  *1~(Q 1 m)
100cm …  *2~(Q 100 cm)
```

と表現することにして，制約付き変数の定義に図５を追加する．このとき，

```
*1~(Q 1 m) と *2~(Q 100 cm)
```

はunifyする．

```
(as (0intersect (Q *nx *ux)
                (Q *ny *uy)
                (Q *nx *ux))
    (equiv *nx *ux *ny *uy))

(as (equiv *nx *ux *ny *uy)
    (unit *ux *sort *kx)
    (unit *uy *sort *ky)
    (:= (x *nx *kx) (x *ny *ky)))

(as (unit mm length 1))
(as (unit cm length 10))
(as (unit  m length 1000))
(as (unit km length 1000000))
(as (unit mg weight 1))
(as (unit  g weight 1000))
(as (unit kg weight 1000000))
```

図５　量に対する制約付き変数

３．　セマンティック・マッチング

セマンティック・マッチングは，２つの対象記述が同一視しうるか否か判定し，同一視しうる場合に２つの対象記述の含む情報を統合したより詳細な対象記述を得ることである[田中 85, 井佐原86, 赤間 87]．ここでは制約付き変数を用いてセマンティック・マッチングを実現する方法について述べ，その特徴について論じる．

３．１　セマンティック・マッチング

まず[赤間 87]に基づいた例をあげる．

```
(*h1^男性
    (名前 *n1^{佐藤　石田})
    (年齢 *a1^[30 40])
    (上司 *j1^人間
            (名前 加藤)))    ------①
```

は，「名前が佐藤あるいは石田，年齢が３０代，加藤と言う名前の人を上司に持つ男性 *h1」であり，

```
(*h2^人間
    (名前 佐藤)
    (年齢　37)
    (上司 *j2^男性))          ------②
```

は，「名前が佐藤，年齢が３７，男性の上司を持つ人間 *h2」である．これらをセマンティック・マッチングするプログラムをここでは s_match （[赤間 87]で unify としたもの）としよう．上で挙げた対象をつかって，

```
(s_match ①　②　*match)
```

と s_match 述語を起動すれば，*match として

```
(*h1^男性
    (名前 佐藤)
    (年齢　37)
    (上司 *j1^男性
            (名前 加藤)))
```

が得られる．

３．２　制約付き変数による実現例

これと同様のことを，制約付き変数で行なう１つの方法は，もちろん，全体のＳ式の構造を変えないまま上記の集合束縛変数を制約付き変数で置き換えることである．しかしここでは，Ｓ式全体を１つの制約付き変数で表現することを考える．そのとき，①，②は以下の③，④にかえればよい．

```
*h3~((is_a 男性)
     (名前 *n3~((member (佐藤 石田))))
     (年齢 *a3~((In 30 40)))
     (上司 *j3~((is_a 人間)
                (名前 加藤))))    ------③
*h4~((is_a 人間)
     (名前 佐藤)
     (年齢　37)
     (上司 *j4~((is_a 男性))))    ------④
```

これらの表現で，各「制約述語」（ここの例では is_a，In，名前，上司 など）は第１引数を省略しているものと考える．たとえば④は，

```
(is_a *h4 人間)  … *h4 は人間
(名前 *h4 佐藤)  … *h4 の名前は佐藤
(年齢 *h4  37)   … *h4 の年齢は37
```

```
(as (0intersect *listx *listy
                ((*pred . *argsz) . *listzz))
    (append *headx
            ((*pred . *argsx) . *tailx)
            *listx)
    (append *heady
            ((*pred . *argsy) . *taily)
            *listy)
    (cut)
    (if (IntersectPred *pred *predI)
        (*predI *argsx *argsy *argsz)
        (= *argsx *argsy)
        (= *argsx *argsz))
    (append *headx *tailx *listxx)
    (append *heady *taily *listyy)
    (0intersect *listxx *listyy *listzz))
(as (0intersect *listx *listy *listz)
    (append *listx *listy *listz))

(as (0member *x *list) (exec_x *x *list))

(as (exec_x *x ()))
(as (exec_x *x ((*pred . *args) . *rest))
    (*pred *x . *args)
    (exec_x *x *rest))
```

図６　セマンティック・マッチングのための 0intersect,0member の定義（トップレベルのみ）

```
(上司 *h4 *j4)   … *h4 の上司は *j4
(is_a *j4 男性) … *j4 は男性
```

の条件をすべて満たす *h4 を意味する（*j4は存在束縛で考える）．

このような対象（制約付き変数）を unify するために，0intersect や 0member などを図６のように定義する．セマンティック・マッチングは，対象記述の中に用いられる「制約述語」の「意味」に依存して結果が変るが，図６は個々の「制約述語」には依存しない部分のプログラムである．

図６に加えて，各「制約述語」の持つ意味を定義しておかねばならない．たとえば，「制約述語」is_a に対しては，

```
(as (IntersectPred is_a is_a_))
(as (is_a_ (*c1) (*c2) (*c1))
    (super *c1 *c2) (cut))
(as (is_a_ (*c1) (*c2) (*c2))
    (super *c2 *c1))

(as (super * *) (cut))
(as (super *des *ans)
    (ASC *des *parent)
    (super *parent *ans))

(as (ASC man human))
(as (ASC woman human))
(as (ASI adam man))
(as (ASI eve woman))

(as (is_a *instance *class)
    (ASI *instance *parent)
    (super *parent *class))
```

のような定義（これはＣ型集合束縛変数の最簡略版である）が必要である．ここで，

```
(as (IntersectPred is_a is_a_))
```

はis_aのための「intersect」計算に述語is_a_を使うことを宣言している．is_a の定義は0memberの計算のときに使われる．

member に関する定義は次のようなものが考えられる．member 自体は通常どおりなので割愛している．

```
(as (IntersectPred member member_))
(as (member_ (*listx)
             (*listy)
             ((*a . *listz)))
    (intersect *listx
               *listy
               (*a . *listz)))

(as (intersect () ? ()) (cut))
(as (intersect (*a . *x) *y (*a . *z))
    (member *a *y)
    (cut)
    (intersect *x *y *z))
(as (intersect (? . *x) *y *z)
    (intersect *x *y *z))
```

制約述語 In に関する定義は，図２と似ていることもあり省略する．これらの定義の下で，

```
(= ③ ④)
```

と unification を起動すれば，

```
*h3~((is_a 男性)
     (名前 佐藤)
     (年齢 37)
     (上司 *j3~((is_a 男性)
                (名前 加藤))))
```

が得られる．

３．３　２つの方法の比較

s_match を用いる方法と，制約付き変数を用いる方法はそれほどの差がないように見えるかもしれない．しかし，それらには重要な違いがある．s_match では

```
(s_match ①  ②  *match)
```

としても，*match に得られる対象は①や②とは別のコンスである．s_match の過程で①や②のコンスの構造は変化しない（集合束縛変数で書いてある部分は変化する）．しかし制約付き変数を用いた方法では，

```
(= ③ ④)
```

によって，③，④が本当に同一の記述に変化する．この効果は全く別のところに出現している *h3，*h4 も瞬時に伝播する．

また，①，②をセマンティック・マッチングするには，ユーザーが陽に s_match 述語を起動しなければならない．しかし，③，④のセマンティック・マッチングは，unification 述語 = で陽に起動できる他に，述語呼出し時のユニフィケーションによって自動的に達成される．この場合セマンティック・マッチングとはユニフィケーションそのものに他ならない．

4．　制約付き変数の実現方法

PAL(version 4.9) では，集合束縛変数を実現する基礎として，変数に任意のＳ式（変数を含むことを許す）を結びつける枠組みがすでに作られている．この枠組みを用いて，[滝川 88] では制約付き変数を提供する言語 PIEP をメタプログラミングを用いて実現している．

本論文ではさらに，制約付き変数をメタプログラミングによらずに実現し，効率の大幅な改善を達成した．そのためにPAL (version 5.0) では，CからPALの述語を呼出し，その述語呼出しがつくる束縛を通常の述語呼出しと同様に利用することのできる関数 eval が用いられている．

5．　むすび

本論文では，PAL に制約付き変数を導入した．それは変数に任意の制約が付加された変数である．制約付き変数は大きな表現力を持つ．これによってユーザーはいろいろないろいろなオブジェクトを自由に定義して，それらのオブジェクトが相互作用(unify)する規則を作り，それらのオブジェクトを直接の対象とした論理プログラミングを行なうことができる．PAL を用いれば，アトムと変数しかオブジェクトのない無味乾燥な世界ではなく，より複雑な対象を模擬するオブジェクトの住む豊かな世界で宣言的にプログラムを記述できる．

これは自然言語処理にとって重要である．人間の頭の中では，さまざまな対象のイメージが生き生きと相互作用している．高度な自然言語処理を達成するためには，人間の思考に出現するイメージのような対象を計算機の中で明快に扱う必要があるだろう．

制約付き変数は豊かな表現力を持つ．他の種類の応用例については，別の論文で述べたい．また制約付き変数の機能を特殊化してユーザーが手軽に利用できる枠組みがすでに PAL に導入されていることを付加えておく．

謝　辞

制約付き変数の実装に貢献していただいた滝川雅巳（北海道大学大学院文学研究科修了，現在 ㈱ビー・ユー・ジー），坪山徳保（北海道大学工学部情報工学科Ｍ２）の各氏に感謝する．

文　献

[赤間 87] 赤間清：集合束縛変数に基づく意味表現とユニフィケーション，情報処理学会，自然言語処理研究会資料，62-8，pp.53-60 (1987)

[赤間 89a] 赤間清：集合束縛変数とその自然言語処理への応用，人工知能学会誌，Vol.4，No.2，pp.177-184 (1989)

[赤間 89b] 赤間清：意味計算Ｉ，情報処理学会，自然言語処理研究会資料，89-22，pp.57-64 (1989)

[井佐原 86] 井佐原均，石崎俊，半田剣一：文脈解析システムにおける概念表現とその照合法，情報処理学会第３２回全国大会講演論文集，6M-2，pp.1283-1284 (1986)

[高木 87] 高木朗，伊東幸宏：自然言語の処理，丸善 p.200 (1987)

[滝川 88] 滝川雅巳：宣言型知識表現言語の構築，北海道大学修士論文，p.60 (1988)

[田中 85] 田中穂積，奥村学，小山晴生：オブジェクトの同一性判定及び同一化アルゴリズムとその応用，日本ソフトウェア科学会第２回大会論文集，2A-2，pp.37-40 (1985)

[坪山 89] 坪山徳保，赤間清，宮本衛市：制約付き変数とその応用，電気関係学会北海道支部連合大会，(1989)