ＣＩＬにおける言語情報の構造と操作のモデル

4C-3

奥西稔幸　　向井国昭

新世代コンピュータ技術開発機構

1. はじめに

CILは，レコード構造を部分項(PST:Partially Specified Term)として表現レベルで組込んだPrologの拡張言語である．

LFGやGPSG，PATR-Ⅱなどの最近の単一化文法に見られるように，言語情報処理においてレコード構造が重要な役割を果たすことが明らかになってきた．LFGの機能構造，GPSG の素性集合，PATR-ⅡのDAGはいずれもレコード構造とみなせる．レコード構造が有用であるのは，それが様々な見方ができるからである：

1) 部分関数
2) 有向グラフ・有向木
3) 有限状態オートマトン
4) 属性－値対リスト
5) 連想リスト
6) 継承的メンバシップ関係を持った集合
7) 添字付き集合
8) 結合律・可換律・冪等律が成り立つ代数的構造
9) 無限ストリーム

またHerbrand項が次のようにレコードの退化表現とみなせる点で，レコード構造をもつCILはDCGの自然な拡張とみなせる：

1) 次数が固定
2) 引数が番号に指定

本論文では，最初にCILのレコード構造のモデル(PTT: Partially Tagged Tree)を示し，次にPTTのパターン表現としてのPSTを説明する．さらにCILで組込述語として提供しているPTT操作述語の紹介後，幾つかの応用により，言語情報解析におけるPSTの有効性を示す．

CILは汎用日本語処理系LTB[1]の基本言語として利用されている．なお，本稿は[2]の抜粋である．

2. 言語情報の構造と基本操作のモデル

レコード構造の論理プログラミングへの素直な導入のために，PTTという無限木からなる領域を考える．

部分的にタグ付けされた木をPTTと呼ぶ．ここで木というのはprefix 演算で閉じている有限列からなる非空集合のことである．木の各節点は無限の枝をもつことができる．

PTT上での基本操作はマージである．PTTの集合がマージに関して可換律で結合律で冪等律が成り立つ半群であるのは明らかである．さらにPTT領域は，［付録．２］の部分等式理論に関して，compactかつsatisfaction completenessである［付録．１］．この性質はCILにおけるSLD-resolutionやnegation-as-failureの完全性や健全性の証明に使われる．

3. 言語情報の表現と部分単一化

PTTのパターン表現として次のような表記をもつデータ表現PSTを導入し，これを部分項を呼ぶ．

$$\{a_1/x_1,\ldots,a_n/x_n\}$$
但し $(n>=0,\ i\neq j \Leftrightarrow a_i\neq a_j)$

a_iはラベル，x_iは値と呼ばれ，値には部分項が許される．レコード構造（部分項）は帰納的に定義できる．

これに伴い，単一化が拡張されている．直観的には，２つの部分項ともに存在するラベルに関しては，ラベルに対応するそれぞれの値の単一化を行う．

```
?- {a/{c/1}, b/X} = {b/2, a/Y}.
    ⇒ X = 2, Y = {c/1}
?- {a/1} = {a/2}    ⇒ no
```

一方の部分項にしか存在しないラベルに関しては，そのラベルと値からなる対を，新たに，他方の部分項に追加する．

```
?- X = {a/1}, Y = {b/1}, X = Y.
   ⇒ X = {a/1, b/1},
     Y = {a/1, b/1}
```

単一化とは，与えられた等式の集合に公理系［付録．２］を適用していくことで，それを含む最小の集合を求めることである．

4. PTT操作用組込述語

PTT操作用組込述語およびシンタックスシュガを説明する．

(1) 組込述語

属性が持つ名前でその値を操作できる(role, locate) ことが部分項の利点の１つであるが，部分項の中身を別の方法で知るようなユーティリティを用意する．現在は，部分項が持つラベルを全て集めてきたり(setOfkeys)，部分項内のデータをレコード形式に変換したり(record)，部分項の対をバックトラックベースで全て調べる(getRole) ことができる述語を揃えている[3]．

```
?- getRole({a/1,b/2},K,V).
   ⇒ K=a,V=1;
     K=b,V=2.
?- role(K,X#{a/1,b/2},3),K=c.
   ⇒ X = {a/1,b/2,c/3}.
```

部分項は意味ネットワークなどの一種のグラフ表現と考えることも出来る．そのためにまず，部分項を別の部分項に併合する述語がある(merge, d_merge, t_merge)．併合された方の部分項だけしか拡張されない点で単一化と異なる．

```
?- d_merge(X#{c/d,a/4}, Y#{a/5}).
   ⇒ X = {a/4,c/d},
     Y = {a/5,c/d}.
```

部分項の拡張は単一化に組込まれているが、削除に関しては，ラベルを指定して削除する述語(delete)と，部分項のパターンを指定して削除する述語(masked_merge)がある．

```
?- delete(a, {a/1,b/2}, R). ⇒ R = {b/2}
```

この他にも、２つの部分項に共通な部分を調べたり(meet)，共通な部分の単一化をする(glue)述語がある．

```
?- meet({a/1,b/2},{b/3,c/4},A-[]).
   ⇒ A = [(b,2,3)].
```

また２つの部分項が包含関係にあるかどうかを調べる述語(subpat,t_subpat)や，包含関係になるように，片方の部分項を拡張する述語(extend) が準備されている．

```
?- t_subpat({a/{b/Y}},{b/1,a/{c/2,b/3}}).
   ⇒ yes
```

(2) シンタックスシュガー

x ! y　部分項のラベル指定による値の参照記法

```
?- {a/1, b/X}!b = 3. ⇒ X = 3
```

x : y　条件付き項

条件yを満たすx．条件には述語が書ける．

```
?- T = X:(father_of(X, 太郎),father_of(X,次郎)).
```

x # y　同格項

Model of structure and operation for linguistic information in CIL

Toshiyuki Okunishi and Kuniaki Mukai (ICOT)

```
    ?- X# {a/1, b/X!a} = Y.
       ⇒ X = {a/1, b/1}
         Y = {a/1, b/1}
```

x ?　遅延実行項
xが未定義の場合，具体化されるまでxを含む述語の実行を中断する．

```
    ?- print(X?), X = ok. ⇒ ok と表示される．
```

x @ y　遅延評価型条件付き項

```
    ?- X = {a/b}, Y = {a / @print}, X = Y. は，
       ⇒ b と表示され、
         X = {a/b},
         Y = {a/b} となる．
```

5. 言語情報解析への応用

言語情報処理への応用として，PSTとその操作述語を用いたプログラム例を示す．

(1) 状況意味論のオブジェクトのモデル

状況意味論におけるオブジェクトが PSTを用いて簡潔に記述・操作できる．［付録．3］に，CILのPST表現による文法・辞書，談話状況(discourse_situation)ならびに事態(soa)の記述を示す．

(2) 心的状態の表現

BarwiseとPerryの態度理論によれば，態度（心的状態）はparametric objectであるframeとsetting（アンカー）の対で表現される．次のような信念があったとする．

ジャック：私は太郎が花子をぶったと思う．
ベティ：私は花子が太郎をぶったと思う．

この場合，二人の心的状態はsettingが異なるだけで，同一のframeで表現できる．今，ジャックのsettingは

```
beater ⊢→ taro
beaten ⊢→ hanako
```

となる．ジャックとベティの心的状態は次のようになる．

```
believe(jack, {frame/beat(beater, beaten),
                beater/taro,
                beaten/hanako}).
believe(betty, {frame/beat(beater, beaten),
                 beater/hanako,
                 beaten/taro}).
```

CILのPTT操作述語を用いた質問応答ができる．

Q1. ぶったのが太郎だと思っているのは誰ですか．

```
  ?- believe(X, {beater/taro}). ⇒ X = jack
```

Q2. ジャックは太郎が何をしたと思っているのですか．

```
  ?- believe(jack, M#{frame/F}), getRole(M, A, taro).
   ⇒ A = beater,
     F = beat(beater, beaten),
     M = {frame/beat(beater, beaten),
          beater/taro, beaten/hanako}
```

6. おわりに

本論文では，言語情報処理にレコード構造が非常に有効な構造であることを述べ，CILで導入したPTTと呼ばれるレコード構造およびその操作述語を紹介し，それがどのように利用されるかを報告した．CILにはこれ以外にも遅延評価に基づく制約解決機構を組込んでいる．

現在，PSI-Ⅱの上で総合的なプログラム開発環境を備えた処理系が稼働している(CIL第3版)．そこでは部分項や遅延評価，シンタックスシュガといったCILの機能に対応するためのスクリーンデバッガを提供している[4]．またPSTに関しては，Herbrand 項への展開による高速化や，それを利用した効率的な継承機能の実現などを検討中である．

［謝辞］
天沼，鈴木両氏にはCIL第3版を実装して頂きました．近藤，今村両氏には貴重な議論をして頂きました．ここに感謝致します．

［参考文献］

[1] 杉村他:汎用日本語処理系LTBの構成，第37回情報処理学会全国大会論文集，1988.
[2] K. Mukai:Partially Specified Term in Logic Programming for Linguistic Analysis, ICOT-TR (to appear), 1988.
[3] 向井他:ＣＩＬ言語マニュアル，ICOT-TM 242, 1986.
[4] 天沼他:CILプログラミング環境，第2回人工知能学会全国大会論文集，1988.

［付録．1］ PTT領域の性質

% t, t_1 はPTT, α, β は節点

・加法（＋）…マージ演算
単位元： $\varepsilon + t = t$.
単位元： $t + \varepsilon = t$.
結合律： $(t_1 + t_2) + t_3 = t_1 + (t_2 + t_3)$.
可換律： $t_1 + t_2 = t_2 + t_1$.
冪等律： $t + t = t$.

・乗法（＊）…接ぎ木演算
単位元： $\langle\rangle t = t$.
結合律： $(\alpha\beta)t = \alpha(\beta t)$
分配律： $\alpha(t_1 + t_2) = \alpha t_1 + \alpha t_2$.
無限分配律： $\alpha(t_1+\ldots+t_\lambda+\ldots) = \alpha t_1+\ldots+\alpha t_\lambda+\ldots$

［付録．2］ 部分等式理論

% x, y, z は変数，u_1, v_1 は任意の表現，α は節点
% f, g はファンクタ，p, q は部分項

(1) x ～ x.
　もし x ～ y ならば y ～ x.
　もし x ～ y かつ y ～ z ならば x ～ z.
(2) もし f ≠ g ならば ¬ f(_,_,...) ～ g(_,_,...).
(3) p ～ p.
　もし p ～ q ならば q ～ p.
(4) もし $f(u_1,\ldots,u_n)$ ～ $f(v_1,\ldots,v_n)$ ならば
　u_1 ～ v_1 かつ ... かつ u_n ～ v_n.
(5) もし p ～ q かつ p/α と q/α がともに存在するならば p/α ～ q/α.
(6) もし x ～ p かつ x ～ y ならば y ～ p.
(7) もし x ～ p かつ x ～ q ならば p ～ q.

［付録．3］

```
  % 状況(discourse_situation)と事態(soa)
{sit/S, sp/I, hr/ You, dl/ Here, exp/ Exp}:
      (member(soa(speaking, (I, Here), yes), S),
       member(soa(addressing, (You, Here), yes), S),
       member(soa(utter, (Exp, Here), yes), S).

  % 文法と辞書
sentence({ip/SOA, ds/DS}) -->
       noun({ip/Ag, ds/ DS}),
       verb({ip/SOA, ds/DS, ag/Ag, obj/ Obj}),
       noun({ip/Obj, ds/ DS}).

noun({ip/jack}) --> [jack].          % Jack
noun({ip/betty}) --> [betty].        % Betty
noun({ip/X, ds/{sp/X}}) --> [i].     % I
noun({ip/X, ds/{hr/X}}) --> [you].   % You
verb({ip/ soa(love, (X, Y, Loc), yes),
       ds/ {dl/Loc}, ag/ X, obj/Y}) --> [love].
```