Vocalog：語彙を用いたジェネリックな概念記述が可能な拡張論理型プログラミング言語†

後藤 文太朗†† 田中 譲††

自然言語における語彙には，(1)対象や制約の概念を語で表す機能，(2)語を合成して複雑な対象や制約の概念を表現する機能，(3)合成語に対して名前付けを行い，この名前を新たに語として用いることを許す語彙構築機能がある．これらの機能は，人と人との間における高度なコミュニケーションや，概念モデルの構築を可能にしている．本論文では，自然言語における語彙の持つこれらの特徴を Prolog に導入したシステム Vocalog を提案する．Vocalog では，名詞と形容詞を定義でき，それらを用いることでそれぞれ対象の概念や対象間の関係概念をジェネリックに記述できる．語構成子を用いて名詞や形容詞を組み合わせて名詞句や形容詞句を作ることができる．名詞句や形容詞句を用いることでそれぞれ複雑な対象の概念や対象間の関係概念を表現できる．語彙構築機能を用いることで，基本的な語彙をもとにして派生語彙を増やしていくことができる．導入した名詞や形容詞を述語定義や質問で利用する機構としてリスト述語を導入している．

1. はじめに

自然言語における語彙には，(1)対象の概念や対象間の関係概念を語で表す機能，(2)語を合成して複雑な対象の概念や複雑な対象間の関係概念を表現する機能，(3)合成語に対して名前付けを行い，この名前を新たに語として用いることを許す語彙構築機能がある．これらの機能は，人と人との間における高度なコミュニケーションや，概念モデルの構築を可能にしている．

著者らは，上述のような特徴を持つ語彙をデータベースへの問い合わせへ導入すること[1),2)]，一階述語論理で記述された知識ベースへ導入すること[3),4)]を行ってきた．本論文では，論理型プログラミング言語 Prolog に語彙を導入し利用できるようにした Vocalog[5)] (Vocabulary-Based Logic Programming Language) を提案する．語彙を Prolog に導入する目的は，知識表現言語としての記述力を高めることである．特に，基本的な概念をコンテキストから取り出して名詞や形容詞としてジェネリックに定義し，これらの合成により，より高次の概念をコンテキストから独立に定義することを目的としている．

Vocalog では，名詞と形容詞を定義でき，それらを用いることでそれぞれ対象の概念や対象間の関係概念をジェネリックに記述できる．語構成子を用いて名詞や形容詞を組み合わせて名詞句や形容詞句を作ることができる．名詞句や形容詞句を用いることでそれぞれ複雑な対象の概念や対象間の関係概念を表現できる．語彙構築機能を用いることで，基本的な語彙をもとに，派生語彙を増やしていくことができる．導入した名詞や形容詞を述語定義や質問で利用する機構としてリスト述語を導入している．

論理型プログラミング言語における項を拡張して，レコード的な構造をもつオブジェクトを扱えるようにしたシステムとして，KBL[6)]，LOGIN[7)]，CIL[8)]，CRL[9)] などが提案されている．データベースの分野では，コンプレックス・オブジェクトを扱うための論理体系として，O-logic[10)]，F-logic[11)]，C-logic[12)] などが提案されている．これらのシステムには，実際のコンプレックス・オブジェクト中で，オブジェクト間の制約を記述する機能がある．Vocalog では，オブジェクト間の制約を，実際のコンプレックス・オブジェクトとは独立に記述できる形容詞を導入している．形容詞を組み合わせた形容詞句という形で，複雑なオブジェクト間の制約をジェネリックに記述できる．

以下，本論文では，2章で Vocalog の概略について述べる．3章では Vocalog における語彙の導入方法に関して説明する．4章で Vocalog によるプログラム例を示す．

2. Vocalog の概略

Vocalog では名詞と形容詞を導入している．語構成子を用いることで名詞や形容詞を合成して名詞句と形容詞句を作ることができる．さらに，語彙構築機能により，名詞句や形容詞句に対して名前付けを行い，こ

† Vocalog: An Extended Logic Programming Language Capable of Generic Concept-Description Using a Vocabulary by FUMITARO GOTO and YUZURU TANAKA (Department of Electrical Engineering, Faculty of Engineering, Hokkaido University).

†† 北海道大学工学部電気工学科

の名前を新たに名詞や形容詞として用いることができる．以下では，これらについて説明する．

2.1 名　　詞

「名前が 'Taro' という人の血液型 X を求めよ」という質問を Prolog と Vocalog で記述した例はそれぞれ次のようになる．

```
Prolog   ?-person('Taro', _, _, _, X).
Vocalog  ?-[first_name:='Taro',
                                blood_type:=X].
```

Prolog の利用者は述語 person が5引数であり，第1引数から第5引数がそれぞれ，名前，姓，性別，誕生日，血液型であることを知っている必要がある．そして，第1引数に定項 'Taro'，第5引数に変数 X を与え，他の引数には無名変数を与えなくてはならない．Vocalog の利用者は，リスト中で，first_name と blood_type という名詞と対応させて，それぞれ定項 'Taro' と変数 X を与えるだけでよい．

Prolog では，$p(t_1, \cdots, t_m)$ のように述語を記述するが，述語名 p により，引数間の関係は示せるが，各引数の役割は明示されない．Prolog では，固定された順番で，引数を与えなくてはならない．

Vocalog では，述語の引数の役割を表すラベルとして，基本名詞というものを導入している．m 引数の述語 p の第1引数から第 m 引数の役割を表すものとして，基本名詞 $n_1, \cdots, n_m$ を導入することにより，述語 $p(t_1, \cdots, t_m)$ を $[n_1:=t_1, \cdots, n_m:=t_m]$ と記述できる．

リスト述語において名詞を利用することにより，

（1） 基本名詞と対応づけて引数を与えるので，リスト中の順番は任意でよい．

（2） 省略された引数には，システムが自動的にユニークな変数を与えるので，指定したい引数だけを与えればよい．

というように表現が簡便になるという利点がある．

2.2 形 容 詞

「名前が 'Taro' という人の，親の血液型を求めよ」という質問を Prolog と Vocalog で記述した例はそれぞれ次のようになり，答えは変数 Z に求めることができる．

```
Prolog   ?-person('Taro', U, _, _, _),
           is_parent_of([X, Y], ['Taro', U]),
           person(X, Y, _, _, Z).
Vocalog  ?-[first_name:='Taro',
            parent@blood_type:=Z].
```

Prolog の利用者は，2引数の述語 is_parent_of が親子関係を表していて，第1引数が親の名前と姓のリストで，第2引数が子供の名前と姓のリストであることを知っている必要がある．さらに共有変数を用いて，述語 person に与える引数と関係づけを行わなければならない．Vocalog では，リスト中で，名詞 first_name と対応させて引数 'Taro' を与え，形容詞 parent* で名詞 blood_type を parent@blood_type のように修飾したものと引数 Z を対応づけて与えるだけでよい．

Prolog では，共有変数を用いた複数の述語の連言により，複雑な関係を表現できるが，利用者はどの述語のどの引数と，どの述語のどの引数とを共有変数を用いて関係づけるかを知っている必要がある．

Vocalog では，対象間の関係概念がジェネリックに形容詞を用いて定義される．リスト述語中で，名詞を形容詞で修飾するだけで対象間に成り立つべき条件を付加できるので，表現が簡便になるという利点がある．

2.3 語 の 合 成

2.1節と2.2節では，語彙を導入することによる利点として，リスト述語で名詞や形容詞という語彙を用いることで表現が簡便になることを述べた．語彙を導入することによる別の利点として，語彙レベルで複雑な概念や関係概念をジェネリックに記述できるというものがある．語構成子というものを使って名詞や形容詞を合成した名詞句や形容詞句を用いて，複雑な概念や関係概念をジェネリックに記述できる．名詞句や形容詞句はリスト述語で利用することができる．

例えば，親であるという関係を表す形容詞 parent，子供であるという関係を表す形容詞 child，自分自身であるという関係を表す形容詞 self があれば，兄弟であるという関係は，「親の子供であって，自分自身ではない」というものであるので，'(child : parent)&(−self)' と記述できる．そして，この形容詞句を用いて，「'Taro' という名前の人の兄弟の名前 X を求めよ」という質問を次のように記述できる．

```
?-[first_name:='Taro',
    ((child : parent)&(−self))@first_name:=X].
```

2.4 語 彙 構 築

Vocalog では，基本名詞と基本形容詞を述語と対応

---

* Vocalog において対象間の親子関係が定義されていて，それに対して parent というラベル付けがされているということである．本論文中では，"形容詞 a" と記述してある場合は，Vocalog においてある関係に対して a というラベルが与えられていることを表している．このラベル a は，対象間の関係を表すものであり，自然言語では形容詞あるいは関係名詞の役割をするものである．Vocalog ではこれらをまとめて形容詞と呼ぶ．

させて導入する．さらにこれらの基本語彙を組み合わせた名詞句や形容詞句に対して名前付けを行い，この名前を新たに名詞や形容詞として用いることができる．これを語彙構築と呼び，語彙構築で導入された名詞と形容詞をそれぞれ派生名詞と派生形容詞と呼ぶ．

Vocalog において，導入された語彙は基本語彙と派生語彙に分けられる．基本語彙はプログラムと対応させて導入されるのに対し，派生語彙は基本語彙をもとにレキシカルに定義されるのでプログラムとは独立である．

語彙構築により派生語彙を導入する利点として，

（1） 複雑な概念や関係概念のジェネリックな定義が可能

（2） 複数プログラムでの同一語彙の利用が容易

というものがある．（1）は語彙レベルで複雑な概念や関係概念を定義できるからである．（2）は基本語彙とプログラムとの対応だけをとりさえすれば派生語彙を利用できるからである．

3. 語彙の導入方法

この章では，前章で示した語彙の Prolog への導入方法について説明する．導入された語彙は，リスト述語において利用することができる．以下の節では，Vocalog における語彙に含まれる語句のシンタックスとリスト述語のシンタックスを示し，語彙の定義方法とリスト述語の評価規則を示す．

3.1 Vocalog のシンタックス

Vocalog における語彙には名詞，形容詞，名詞句，形容詞句がある．名詞句と形容詞句は語構成子を用いて名詞と形容詞を組み合わせたものである．名詞句を作るための語構成子としては，代入（':='），論理演算（'+', '&', '−'），グループ・バイ演算（'/'），射影演算（'≪'），関数，リスト化，形容詞句がある．形容詞句を作るための語構成子としては，逆演算（'~'），論理演算（'+', '&', '−'），合成演算（':'），推移閉包演算（'*'）がある．名詞句と形容詞句のシンタックスを図1のように定義する．

名詞と形容詞は，さらに基本名詞と派生名詞，基本形容詞と派生形容詞に分けられる．これらの語については次節以降で説明する．

リスト述語は，語彙を述語定義や質問で利用するためのものである．リスト述語は，名詞句からなるリストであり，$N_1, \cdots, N_k$ を名詞句とすると，リスト述語は $[N_1, \cdots, N_k]$ という形になる．

3.2 基本名詞

基本名詞は，述語の引数の役割を表すラベルとして導入している．m 引数の述語 p の第1引数から第 m 引数の役割を表すラベルとして基本名詞 $n_1, \cdots, n_m$ を導入するには次の形の式を用いる．

$$\text{basic\_predicate}(p) ::= [n_1, \cdots, n_m]. \qquad (1)$$

これにより，述語 $p(t_1, \cdots, t_m)$ を $[n_1 := t_1, \cdots, n_m := t_m]$ と記述できる．また，引数は，基本名詞と対応させて与えればよいので，リスト中の順番は任意でよい．リスト中で，m 個の基本名詞のうち i 個だけが使われ，i 個の引数だけが指定されたときは，残りの $(m-i)$ 個の引数には，自動的にユニークな変数が割り当てられる．

Vocalog システムでは，（1）式を次の形の事実に変換してシステムに格納する．

$$\text{varg}(n_1, x_1, p(x_1, \cdots, x_m))\leftarrow.$$
$$\cdots$$
$$\text{varg}(n_m, x_m, p(x_1, \cdots, x_m))\leftarrow.$$

ただし，$x_1, \cdots, x_m$ は互いに異なる変数である．

このようにして定義された基本名詞を用いたリスト述語の評価規則は次のようになる．

$$[N_1 := X_1, \cdots, N_k := X_k]\leftarrow \text{varg}(N_1, X_1, P), \cdots, \text{varg}(N_k, X_k, P), \text{call}(P).$$
$$[]\leftarrow.$$

ここで，N_i $(i=1, \cdots, k)$ は基本名詞，X_i $(i=1, \cdots, k)$ は変数である．call(P) は，P を述語として評価する述語である．

```
<名詞句> ::= <名詞>
           |<名詞句>:=<項>
           |<名詞句>+<名詞句>
           |<名詞句>&<名詞句>
           |-<名詞句>
           |<名詞句>/<名詞句>
           |<名詞句><<<名詞句>
           |<関数名>(<名詞句>,…,<名詞句>)
           |[]|[<名詞句>,…,<名詞句>]
           |<形容詞句>@<名詞句>
<形容詞句> ::= <形容詞>
           |<形容詞句>~
           |<形容詞句>+<形容詞句>
           |<形容詞句>&<形容詞句>
           |-<形容詞句>
           |<形容詞句>:<形容詞句>
           |<形容詞句>*
```

図1 名詞句と形容詞句のシンタックス
Fig. 1 Syntax for noun phrases and adjective phrases.

3.3 名詞句

この節では，形容詞句以外の語構成子について説明し，それを用いた名詞句を含むリスト述語がどのようにして評価されるかを示す．

代入

代入は，名詞句の取る値を指定するのに用いる．前節では基本名詞の取る値を指定するのに用いていた．

代入を用いない名詞句 N_i を含むリスト述語は，N_i を $N_i:=t_i$ で置き換えたリスト述語が成り立つとき，真となる（**図 2-(1)**）．ただし，t_i はシステムが与えるユニークな変数である．代入を二重に用いた名詞句 $(N_i:=t):=t_i$ を含むリスト述語は，$t=t_i$ かつ，$(N_i:=t):=t_i$ を $N_i:=t_i$ で置き換えたリスト述語が成り立つとき真となる（図 2-(2)）．

論理演算

和を用いた名詞句 $(N_{i_1}+N_{i_2}):=t_i$ を含むリスト述語は，$(N_{i_1}+N_{i_2}):=t_i$ を $N_{i_1}:=t_i$ または $N_{i_2}:=t_i$ で置き換えたリスト述語が成り立つとき真となる（図2-(3, 4)）．

積を用いた名詞句 $(N_{i_1}\ \&\ N_{i_2}):=t_i$ を含むリスト述語は，$(N_{i_1}\ \&\ N_{i_2}):=t_i$ を $N_{i_1}:=t_i$ に置き換えても $N_{i_2}:=t_i$ に置き換えてもそのリスト述語が成り立つとき真となる（図 2-(5)）．

否定を用いた名詞句 $(-N_i):=t_i$ を含むリスト述語は，$(-N_i):=t_i$ を取り除いたリスト述語が成り立ち，かつ，$(-N_i):=t_i$ を $N_i:=t_i$ で置き換えたリスト述語が成り立たないとき真となる（図 2-(6)）．

グループ・バイ演算

グループ・バイ演算を用いた名詞句 $(N_{i_1}/N_{i_2}):=t_i$ を含むリスト述語は，$(N_{i_1}/N_{i_2}):=t_i$ を $N_{i_2}:=t_{i_2}$ で置き換えたリスト述語が成り立ち，かつ，$t_i=\{t_{i_1}|[N_{i_1}:=t_{i_1}, N_{i_2}:=t_{i_2}]\}$ のとき真となる（図 2-(7)）．ただし，t_{i_1}，t_{i_2} はシステムが与えるユニークな変数である．実際には，集合はリストを用いて表現されている．図 2-(7) において，$\mathrm{setof}(t_{i_1}, [N_{i_1}:=t_{i_1}, N_{i_2}:=t_{i_2}], t_i)$ は，$t_i=\{t_{i_1}|[N_{i_1}:=t_{i_1}, N_{i_2}:=t_{i_2}]\}$ なる t_i を求める述語である．例えば，child/person という名詞句は，person の値が同じである人の child の値の集合を値に取る．すなわち，ある人の子供の集合を値に取る．

射影演算

射影演算を用いた名詞句 $(N_{i_1}, \ll N_{i_2}):=t_i$ を含むリスト述語は，$(N_{i_1}\ll N_{i_2}):=t_i$ を取り除き，$N_{i_1}:=t_i$ と N_{i_2} を加えたリスト述語が成り立つとき真となる（図 2-(8)）．例えば，first_name≪(last_name:='Yamada') という名詞句は，last_name が 'Yamada' である人の first_name の値だけを取ることができる．

関数

名詞句 $f(N_{i_1}, \cdots, N_{i_m}):=t_i$ を含むリスト述語は，$f(N_{i_1}, \cdots, N_{i_m}):=t_i$ を取り除き，$N_{i_1}:=t_{i_1}, \cdots, N_{i_m}:=t_{i_m}$ を加えたリスト述語が成り立ち，かつ，$t_{i_1}, \cdots, t_{i_m}$ を関数 f に適用した値 $f(t_{i_1}, \cdots, t_{i_m})$ が t_i のとき真となる（図 2-(9)）．ただし，$t_{i_1}, \cdots, t_{i_m}$ はシステムが与えるユニークな変数である．また，図 2-(9) において evaluate_function$(f(t_{i_1}, \cdots, t_{i_m}), t_i)$ は，$f(t_{i_1}, \cdots, t_{i_m})$ を評価してその値を t_i に求める述語である．例えば，count(X) という関数が，リスト X の要素数を返す関数だとすると，count(child/person) の値は子供の数になる．

リスト化

名詞句 $[N_{i_1}, \cdots, N_{i_m}]:=t_i$ を含むリスト述語は，$[N_{i_1}, \cdots, N_{i_m}]:=t_i$ を取り除いて，$N_{i_1}:=t_{i_1}, \cdots, N_{i_m}:=t_{i_m}$ を加えたリスト述語が成り立ち，かつ，$t_i=[t_{i_1}, \cdots, t_{i_m}]$ のとき，真となる（図 2-(10)）．ただし，$t_{i_1}, \cdots, t_{i_m}$ はシステムが与えるユニークな変数である．

(1) $[N_1,\ldots,N_i,\ldots,N_k] \leftarrow [N_1,\ldots,N_i:=t_i,\ldots,N_k]$.
(2) $[N_1,\ldots,(N_i:=t):=t_i,\ldots,N_k] \leftarrow t=t_i, [N_1,\ldots,N_i:=t_i,\ldots,N_k]$.
(3) $[N_1,\ldots,(N_{i_1}+N_{i_2}):=t_i,\ldots,N_k] \leftarrow [N_1,\ldots,N_{i_1}:=t_i,\ldots,N_k]$.
(4) $[N_1,\ldots,(N_{i_1}+N_{i_2}):=t_i,\ldots,N_k] \leftarrow [N_1,\ldots,N_{i_2}:=t_i,\ldots,N_k]$.
(5) $[N_1,\ldots,(N_{i_1}\&N_{i_2}):=t_i,\ldots,N_k] \leftarrow [N_1,\ldots,N_{i_1}:=t_i,\ldots,N_k], [N_1,\ldots,N_{i_2}:=t_i,\ldots,N_k]$.
(6) $[N_1,\ldots,(-N_i):=t_i,\ldots,N_k] \leftarrow [N_1,\ldots,N_{i-1},N_{i+1},\ldots,N_k]$, not$([N_1,\ldots,N_{i-1},N_i:=t_i,N_{i+1},\ldots,N_k])$.
(7) $[N_1,\ldots,(N_{i_1}/N_{i_2}):=t_i,\ldots,N_k] \leftarrow [N_1,\ldots,N_{i_2}:=t_{i_2},\ldots,N_k]$, setof$(t_{i_1}, [N_{i_1}:=t_{i_1}, N_{i_2}:=t_{i_2}], t_i)$.
(8) $[N_1,\ldots,(N_{i_1}\ll N_{i_2}):=t_i,\ldots,N_k] \leftarrow [N_1,\ldots,N_{i_1}:=t_i,N_{i_2},\ldots,N_k]$.
(9) $[N_1,\ldots,f(N_{i_1},\ldots,N_{i_m}):=t_i,\ldots,N_k] \leftarrow [N_1,\ldots,N_{i_1}:=t_{i_1},\ldots,N_{i_m}:=t_{i_m},\ldots,N_k]$, evaluate_function$(f(t_{i_1},\ldots,t_{i_m}),t_i)$.
(10) $[N_1,\ldots,[N_{i_1},\ldots,N_{i_m}]:=t_i,\ldots,N_k] \leftarrow [N_1,\ldots,N_{i_1}:=t_{i_1},\ldots,N_{i_m}:=t_{i_m},\ldots,N_k], t_i=[t_{i_1},\ldots,t_{i_m}]$.
(11) $[N_1,\ldots,[]:=t_i,\ldots,N_k] \leftarrow [N_1,\ldots,N_{i-1}:=t_{i-1},N_{i+1}:=t_{i+1},\ldots,N_k], t_i=[]$.

図 2 名詞句に対するリスト述語の定義
Fig. 2 The definition of list predicates for noun phrases.

名詞句 $[]:=t_i$ を含むリスト述語は，$[]:=t_i$ を取り除いたリスト述語が成り立ち，かつ，$t_i=[]$ のとき，真となる（図 2-(11)）．

例えば，[first_name, last_name] という名詞句の値は，first_name の値と last_name の値からなるリストである．

3.4 基本形容詞

W を定義済みの名詞と形容詞から作られるすべての名詞句の集合とする．基本形容詞 A は

$$\begin{aligned}&\text{basic\_adjective}(A) ::= \\ &\quad [\text{source}@N_s:=X_s, \\ &\quad\ \text{destination}@N_d:=Y_d, \\ &\quad\ p(X_s, Y_d)]. \qquad (2)\end{aligned}$$

のような形の式で定義する．W_1 と W_2 を W のコピーとする．**図 3** に示すように，基本形容詞 A は，W_1 と W_2 との間に，W_1 の名詞句 N_s の値 X_s と W_2 の名詞句 N_d の値 Y_d との間に述語で表される関係 $p(X_s, Y_d)$ が成り立っているという制約を表している．W_2 の名詞句 N に対して，$A@N$ のようにラベル A をつけて表すことにより，W_1 と W_2 の語彙を区別でき，W から $W'(=W\cup A@W$, ただし $A@W=\{A@N|N\in W\})$ へと使用可能な語彙が増加する．ラベル A は名詞句 N を修飾することで制約を加えており，自然言語における形容詞に相当する．'@' は基本形容詞とそれが修飾している名詞句とを Vocalog の構文において分離するための区切り子である．

Vocalogシステムは，(2)式のような基本形容詞の定義を次のような事実に変換してシステムに格納する．

$$\text{adj\_sdc}(A, N_s:=X_s, N_d:=Y_d, p(X_s, Y_d))\leftarrow.$$

このように定義された基本形容詞を用いた名詞句 $(A@N_i):=t_i$ を含むリスト述語は，$(A@N_i):=t_i$ を $N_s:=X$ で置き換えたリスト述語と，述語 $p(X, Y)$ とリスト述語 $[N_d:=Y, N_i:=t_i]$ の三つがすべて成り立つとき真となる．

$$\begin{aligned}&[N_1, \cdots, (A@N_i):=t_i, \cdots, N_k]\leftarrow \\ &\quad \text{adj\_sdc}(A, N_s:=X, N_d:=Y, p(X, Y)), \\ &\quad [N_1, \cdots, N_s:=X, \cdots, N_k], \\ &\quad p(X, Y), \\ &\quad [N_d:=Y, N_i:=t_i].\end{aligned}$$

3.5 形容詞句

この節では，形容詞句を作るための語構成子について説明し，それらの語構成子を用いた形容詞句が使われたリスト述語がどのように評価されるかを示す．

逆演算

逆演算を用いることで，形容詞が表す関係の逆の関係を表すことができる．基本形容詞 A に逆演算を適用した形容詞句 A^- が使われたときは，**図 4**-(1)に示す評価規則を用いて評価される．逆演算を用いると，図 3 における W_1 と W_2 を入れ換えたことになる．

語構成子を用いた形容詞句への逆演算の適用は次のように定義する．

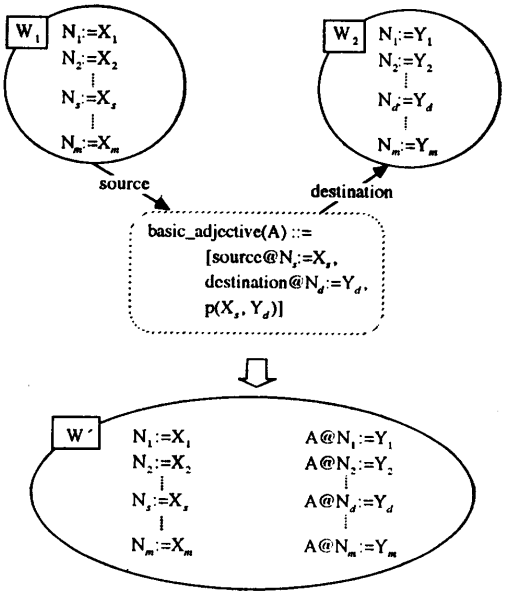


図 3 基本形容詞による制約の付加
Fig. 3 A constraint defined by a basic adjective A.

(1) $[N_1,\ldots,(A^-@N_i):=t_i,\ldots,N_k]\leftarrow \text{adj\_sdc}(A, N_s:=X, N_d:=Y, p(X,Y)), [N_1,\ldots,N_s:=Y,\ldots,N_k], p(X,Y), [N_s:=X, N_i:=t_i].$

(2) $[N_1,\ldots,(A^-@N_i):=t_i,\ldots,N_k]\leftarrow \text{inverse}(A, B), [N_1,\ldots,(B@N_i):=t_i,\ldots,N_k].$

(3) $[N_1,\ldots,((A+B)@N_i):=t_i,\ldots,N_k]\leftarrow[N_1,\ldots,(A@N_i):=t_i,\ldots,N_k].$

(4) $[N_1,\ldots,((A+B)@N_i):=t_i,\ldots,N_k]\leftarrow[N_1,\ldots,(B@N_i):=t_i,\ldots,N_k].$

(5) $[N_1,\ldots,((A\&B)@N_i):=t_i,\ldots,N_k]\leftarrow[N_1,\ldots,(A@N_i):=t_i,\ldots,N_k], [N_1,\ldots,(B@N_i):=t_i,\ldots,N_k].$

(6) $[N_1,\ldots,((-A)@N_i):=t_i,\ldots,N_k]\leftarrow[N_1,\ldots,(-(A@N_i)):=t_i,\ldots,N_k].$

(7) $[N_1,\ldots,((A:B)@N_i):=t_i,\ldots,N_k]\leftarrow[N_1,\ldots,(B@(A@N_i)):=t_i,\ldots,N_k].$

(8) $[N_1,\ldots,((A^*)@N_i):=t_i,\ldots,N_k]\leftarrow[N_1,\ldots,(A@N_i):=t_i,\ldots,N_k].$

(9) $[N_1,\ldots,((A^*)@N_i):=t_i,\ldots,N_k]\leftarrow[N_1,\ldots,((A^*:A)@N_i):=t_i,\ldots,N_k].$

図 4 形容詞句に対するリスト述語の評価規則
Fig. 4 Evaluation rules of list predicates including adjective phrases.

$(A^{\sim})^{\sim}=A$
$(A+B)^{\sim}=(A^{\sim})+(B^{\sim})$
$(A\&B)^{\sim}=(A^{\sim})\&(B^{\sim})$
$(-A)^{\sim}=-(A^{\sim})$
$(A:B)^{\sim}=(B^{\sim}):(A^{\sim})$
$(A^{*})^{\sim}=(A^{\sim})^{*}$

ここで，A，B は任意の形容詞句であり，'+'，'&'，'−'，':'，'*' は後述する語構成子である．語構成子を用いた形容詞句が使われたリスト述語は，図 4-(2) に示す評価規則を用いて評価される．ただし，述語 inverse(X, Y) は，形容詞句 X に逆演算を適用した結果の形容詞句 Y を求める述語である．

論理演算

A と B が形容詞句のとき，形容詞句 $A+B$，$A\&B$，$-A$ は，それぞれ A か B によって表される関係（図 4-(3, 4)），A と B の両方で表される関係（図 4-(5)），A によって表される関係がないという制約（図 4-(6)）を表す．

合成演算

A と B が形容詞句のとき，形容詞句 $A:B$ は，関係 B と A をこの順番で合成した関係を表す．すなわち，N を任意の名詞句とするとき，$(A:B)@N$ は，$B@(A@N)$ のように定義される（図 4-(7)）．

推移閉包演算

推移閉包演算により，形容詞が表す関係の推移閉包を表すことができる（図 4-(8, 9)）．すなわち，X^{*} という形容詞句は，$X+X:X+X:X:X+\cdots$ という形容詞句に等しい．

3.6 派生名詞と派生形容詞

名詞句 N に対して新たに N' という名詞としての名前付けを行い，派生名詞 N' を定義するには，

noun(N') ::=N.

という形の式を用いる．Vocalog システムは，これを次の形の事実に変換して格納する．

dicNL(N', N)←.

このように定義された派生名詞を用いた名詞句 $N':=t_i$ を含むリスト述語は，

$[N_1,\cdots,N':=t_i,\cdots,N_k]$←
　　dicNL(N', N), $[N_1,\cdots,N:=t_i,\cdots,N_k]$.

のように N' を N で置き換えたリスト述語が成り立つとき真となる．

形容詞句 A に対して新たに A' という形容詞としての名前付けを行い，派生形容詞 A' を定義するには，

adjective(A') ::=A.

という形の式を用いる．Vocalog システムは，これを次の形の事実に変換して格納する．

dicAL(A', A)←.

このように定義された派生形容詞を用いた名詞句 $(A'@N_i):=t_i$ を含むリスト述語は，

$[N_1,\cdots,(A'@N_i):=t_i,\cdots,N_k]$←
　　dicAL(A', A),
　　$[N_1,\cdots,(A@N_i):=t_i,\cdots,N_k]$.

のように，A' を A で置き換えたリスト述語が成り立つとき真となる．

4. プログラム例

Vocalog におけるプログラミング例として，家系に関する語彙の構築例と文法規則の表現例を示す．

4.1 家系に関する語彙の構築例

個人情報を管理している 5 引数の述語 person と，親子関係を表している 2 引数の述語 is_parent_of が定義されているとする．各人の名前，姓，性別，誕生日，血液型がそれぞれ述語 person の第 1 引数から第 5 引数に格納されている．各親子の組に対して，述語 is_parent_of の第 1 引数には，親の名前と姓のリストが，第 2 引数には，子供の名前と姓のリストが格納されている．

最初に，述語 person の第 1 引数から第 5 引数の役割を表すラベルとして，基本名詞 first_name, last_name, sex, birthday, blood_type を導入する（図 5-(1)）．

次に，基本名詞と述語を用いて，基本形容詞を定義する．親子関係を表す基本形容詞 parent（図 5-(2)），修飾された名詞が男の人の属性を表す基本形容詞 male（図 5-(3)），修飾された名詞が女の人の属性を表す基本形容詞 female（図 5-(4)），修飾された名詞が自分自身の属性を表す基本形容詞 self（図 5-(5)）を定義する．

以上のような基本概念を表す基本語彙をもとに，図 5-(6) から (18) に示すような家系に関する複合概念をジェネリックに定義することができる．語彙という形で家系に関する概念を定義しておくことにより，語彙中の語を組み合わせ，複雑な文を容易に表現することができる．以下に質問例を示す．

(1) 'Ken' という名前の人の兄弟の名前 X を求めよ．

```
?-[first_name:='Ken',
    sibling@first_name:=X].
```

```
(1) basic_predicate(person) ::= [first_name,last_name,sex,birthday,blood_type].
(2) basic_adjective(parent)::= [source@[first_name,last_name]:=X, destination@[first_name,last_name]:=Y,is_parent_of(Y,X) ].
(3) basic_adjective(male)::=[destination@sex:=X, X=male].
(4) basic_adjective(female)::= [destination@sex:=X,X=female].
(5) basic_adjective(self) ::= [source@[first_name,last_name]:=X, destination@[first_name,last_name]:=Y,X=Y].
(6) adjective(child)::=parent~.
(7) adjective(father)::=parent&male.
(8) adjective(mother)::=parent&female.
(9) adjective(son)::=child&male.
(10) adjective(daughter)::=child&female.
(11) adjective(sibling) ::= (child:parent)&(-self).
(12) adjective(brother)::= sibling&male.
(13) adjective(sister)::= sibling&female.
(14) adjective(cousin)::= (child:sibling):parent
(15) adjective(niece) ::= (child:sibling)&female.
(16) adjective(nephew) ::= (child:sibling)&male.
(17) noun(person) ::= [first_name,last_name].
(18) noun(number_of_children) ::= count((child@person)/person).
```

図 5 家系に関する語集の構築例
Fig. 5 Example word definitions about family lines.

(2) 'Taro' という名前の人の兄弟の数 X を求めよ.

```
?-[first_name:='Taro',
   count((sibling@first_name)/first_name):=X].
```

(3) 'Ken' という名前の人の兄弟で, 'Ken' と同じ血液型の人の名前 X を求めよ.

```
?-[first_name:='Ken', blood_type:=Y,
   sibling@[first_name:=X, blood_type:=Y]].
```

4.2 文法規則の表現例

DCG で, **図 6** のように表されている文法規則を考える. この文法規則を Vocalog で表すと**図 7** のようになる.

最初に, 差分リストを表す述語 d_list を定義する (図 7-(1)). さらに図 7-(2)から(6)に示すような述語を定義する.

次に, 述語 d_list と対応させて, 基本名詞 d_top, d_end を定義する (図 7-(7)).

基本形容詞として, proper_noun, posses, noun, intrans_verb, trans_verb を定義する (図 7-(8-12)).

以上のように文法規則を表現する上での基本概念を基本語集として定義することにより, 図 7-(13) から (18) に示すように文法に関する制約を派生語集として定義できる. 文法に関する制約を語集という形で記述することにより, 文に対する制約を語の組み合わせにより表現することができる. 以下に質問例を示す.

```
sentence --> noun_phrase,verb_phrase.
noun_phrase --> proper_noun.
noun_phrase --> posses, noun.
verb_phrase --> intrans_verb.
verb_phrase --> trans_verb, object.
object --> noun_phrase.
proper_noun --> [john].
posses --> [my].
noun --> [dog].
intrans_verb --> [runs].
trans_verb --> [likes].
```

図 6 DCG による文法規則の表現例
Fig. 6 Example grammar rules in DCG.

```
(1) d_list(X,X).
    d_list([E|X],Y) :- d_list(X,Y).
(2) proper_noun([john|X],X).
(3) posses([my|X],X).
(4) noun([dog|X],X).
(5) intrans_verb([runs|X],X).
(6) trans_verb([likes|X],X).
(7) basic_predicate(d_list) ::= [d_top, d_end].
(8) basic_adjective(proper_noun)::=[source@d_top:=X, destination@[d_top:=Y,d_end:=Z], proper_noun(Y,Z),X=Z].
(9) basic_adjective(posses)::=[source@d_top:=X, destination@[d_top:=Y,d_end:=Z], posses(Y,Z),X=Z].
(10) basic_adjective(noun)::=[source@d_top:=X, destination@[d_top:=Y,d_end:=Z], noun(Y,Z),X=Z].
(11) basic_adjective(intrans_verb)::=[source@d_top:=X, destination@[d_top:=Y,d_end:=Z], intrans_verb(Y,Z),X=Z].
(12) basic_adjective(trans_verb)::=[source@d_top:=X, destination@[d_top:=Y,d_end:=Z], trans_verb(Y,Z),X=Z].
(13) adjective(noun_phrase)::=proper_noun+posses:noun.
(14) adjective(object)::=noun_phrase.
(15) adjective(verb_phrase)::=intrans_verb+trans_verb:object.
(16) adjective(sentence) ::= noun_phrase:verb_phrase.
(17) noun(period)::= d_end:=[].
(18) noun(sentence)::= sentence@d_top.
```

図 7 Vocalog を用いた文法規則の表現例
Fig. 7 Example grammar rules in Vocalog.

(1) 'john likes my dog' という文は，この文法規則から作ることができますか．

```
?-[sentence :=[john, likes, my, dog], period].
```

(2) この文法規則から作られる文で，固有名詞から始まり，その後に他動詞が続く文 X を求めよ．

```
?-[(sentence&
    (proper_noun : trans_verb))@d_top :=X,
   period].
```

5. おわりに

自然言語における語彙は，人と人との間のコミュニケーションや，概念構築において重要な役割を担っている．本論文では，自然言語における語彙の持つ特徴を Prolog に導入したシステム Vocalog を提案した．

Vocalog では，述語の引数の役割を表すラベルとして基本名詞を導入した．基本名詞を用いることで，述語の引数を直接的に指定することができるようになった．さらに，名詞に対する語構成子を導入し，語構成子を用いて名詞を組み合わせた名詞句で，複雑な概念を表すことができた．名詞に対する語構成子として，代入 (':='), 論理演算 ('+', '&', '−'), グループ・バイ演算 ('/'), 射影演算 ('≪'), 関数，リスト化，形容詞句を導入した．

名詞句と述語との間の関係として基本形容詞を導入した．基本形容詞により，名詞間に新しい関係を定義でき，扱える概念の幅を増やすことができた．形容詞に対する語構成子を導入し，これを用いて形容詞を組み合わせた形容詞句で，複雑な関係概念をジェネリックに表現できた．形容詞に対する語構成子としては，逆演算 ('~'), 論理演算 ('+', '&', '−'), 合成演算 (':'), 推移閉包演算 ('*') を導入した．

定義された語やそれらを組み合わせた句に対して名前付けを行い，この名前を新たに語として用いることを許す機能として，語彙構築機能を導入した．語彙構築機能により，基本語彙から，使える語彙を増やしていくことが可能となった．さらに，基本語彙は，プログラムと対応させて導入したが，語彙構築により導入された派生語彙は，基本語彙をもとにレキシカルに定義することでプログラムとは独立に導入することができた．

最後の章では，Vocalog におけるプログラミング例をあげ，Vocalog におけるプログラミングが語彙構築を行う過程に外ならないことを示した．

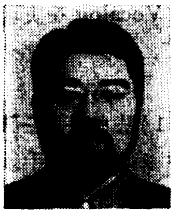
なお，Vocalog は現在 SICStus Prolog 上にメタ・インタプリタにより実現されている．Vocalog を実現する上での機構それ自体は単純なので，メタ・インタプリタでもかなり高速に実行できる．

語彙利用の今後の展望としては，(1)基本語彙を計算機間で共通に持たせて語彙をプログラム流通の手段として用いることや，(2)対象の概念や対象間の関係概念を表す語彙における，概念の推論体系などの研究が考えられる．

参考文献

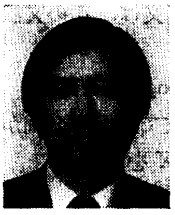
1) Tanaka, Y.: Information Space Model, *Proc. 2nd Workshop on Formal Bases for Databases*, Toulouse (1979).
2) Tanaka, Y.: Vocabulary Building for Database Queries, *Lecture Notes in Computer Science, Vol. 147: RIMS Symposia on Software Science and Engineering*, pp. 215-232, Springer-Verlag (1983).
3) Tanaka, Y.: Roles of a Vocabulary in Knowledge-Based Systems, *IFIP WG 10.1 Int'l Workshop on Knowledge Architecture*, Gotenba (1987).
4) Tanaka, Y.: Vocabulary-Based Logic Programming, *Proc. InfoJapan '90*, Tokyo, Vol. 2, pp. 25-32 (1990).
5) Goto, F. and Tanaka, Y.: Introducing a Large Vocabulary into Prolog, *Proc. Pacific Rim International Conference on Artificial Intelligence '90*, pp. 816-821 (1990).
6) Ait-Kaci, H.: An Algebraic Semantics Approach to the Effective Resolution of Type Equations, *Theor. Comput. Sci.*, Vol. 45, pp. 293-351 (1986).
7) Ait-Kaci, H. and Nasr, R.: LOGIN : A Logic Programming Language with Built-in Inheritance, *Journal of Logic Programming*, Vol. 3, pp. 185-215 (1986).
8) Mukai, K.: Anadic Tuples in Prolog, ICOT-TR 239 (1987).
9) Yokota, K.: Deductive Approach for Nested Relations, ICOT-TR 334 (1988).
10) Maier, D.: A Logic for Objects, *Proc. Workshop of Deductive Database and Logic Programming*, pp. 6-26 (1986).
11) Kifer, M. and Lausen, G.: F-Logic : A 'Higher-Order' Logic for Reasoning about Objects, Inheritance, and Scheme, *SIGMOD '89*, pp. 134-146 (1989).
12) Chen, W. and Warren, D.: C-Logic of Complex Objects, *PODS '89*, pp. 369-378 (1989).

(平成3年5月17日受付)
(平成4年2月14日採録)



後藤文太朗（正会員）

昭和40年生．昭和63年北海道大学工学部電気工学科卒業．平成2年同大学院工学研究科修士課程修了．現在，北海道大学大学院工学研究科博士後期課程電気工学専攻在学中．論理型プログラミング，学習理論に関する研究に従事．ソフトウェア科学会会員．



田中　譲（正会員）

昭和25年生．昭和47年京都大学電気工学科卒業．昭和49年京都大学電子工学専攻修士課程修了．工学博士．現在，北海道大学電気工学科教授．データベースマシン，データベース理論，メディア・ベース，論理型プログラミング等の研究に従事．主たる著書，「コンピュータ・アーキテクチャ」（オーム社，共著）．IEEE，ソフトウェア科学会，人工知能学会各会員．