選言情報を含む単一化文法記述から論理的制約に基づく内部表現への変換法

中野幹生† 島津 明†

本論文では，選言情報を含む単一化文法記述を論理的制約に基づく内部表現に変換する方法を提案する．単一化文法の記述法では，素性名と素性値とのペアのリストおよび経路方程式を用いて素性構造を記述する方法が，一階論理の項を用いる方法よりも，記述，変更，理解が容易である点で優れている．一方，単一化文法の内部表現は，素性構造をラベルつき有向グラフで表す方法と，一階論理の項を用いる方法とに分類できるが，後者を用いた方が構文解析の効率が良い．したがって，素性名と素性値とのペアのリストおよび経路方程式を用いた文法記述を，一階論理の項に基づく内部表現に変換する方法が望まれる．しかしながら，現在までに提案されている変換法は，選言情報を含む単一化文法を扱うことができない．選言情報を含む素性構造の記述は文法中の冗長性を減らすのに有効であり，選言情報を含む内部表現を用いることにより構文解析の効率が向上するため，選言情報を含む記述を選言情報を保ったまま一階論理の項に基づく内部表現に変換する方法が望まれる．本論文では，素性名と素性値とのペアのリストおよび経路方程式を用いた文法記述で選言情報を含むものを，選言を展開することなく，一階論理の項をベースにした内部表現に変換する方法を提案し，実験によりその有効性を示す．

Generating a Logical-constraint-based Internal Representation from a Unification Grammar Formalism with Disjunctive Information

MIKIO NAKANO† and AKIRA SHIMAZU†

This paper proposes a method for generating a logical-constraint-based internal representation from a unification grammar formalism with disjunctive information. Unification grammar formalisms based on path equations and lists of pairs of feature name and feature value are better than those based on first-order terms in that the former is easier to describe, to amend, and to understand. Internal representations of unification grammars can be classified into those using labeled, directed graphs and those using first-order terms. Parsing is more efficient when the latter representations are used. Therefore, it is effective to translate a formalism based on path equations and lists of pairs of feature name and feature value into a term-based internal representation. Previous translation methods are problematic in that they cannot deal with disjunctive feature descriptions, which reduce redundancies in the grammar. Moreover, incorporating disjunctive information into internal representation makes parsing efficient. This paper presents a method for translating a grammar formalism with disjunctive information based on path equations and lists of pairs of feature name and feature value into an internal representation using first-order terms, without expanding disjunctions. It also describes the experiment results that show advantages of the proposed method.

1. はじめに

近年，単一化に基づく自然言語解析の研究が多く行われてきている．単一化に基づくアプローチは，文法を宣言的に記述することが可能であり，したがって，文法の開発や修正が容易であるといった多くの利点を持っている[1)]．

単一化文法では，文法情報は，素性の集まりである素性構造で表される[28)]．各素性は，素性名と値のペアによって規定される．単一化文法に基づく解析システムは，素性構造をどのようなデータ構造（内部表現）を用いて表すかによって，(a) ラベルつき有向グラフを用いる方法[6),15),27)]と，(b) 一階論理の項を用いる方法[18),24),25),32)]に分類できる．

内部表現とは別に，文法記述者がどのような記述法

† NTT 基礎研究所
NTT Basic Research Laboratories

を用いるかという視点で分類することもできる．文法記述法は，(i) 素性構造を素性名と素性値のペアのリストや経路方程式を用いて記述するもの[15),19),27),30)]，(ii) 素性構造を一階論理の項で記述するもの[18),24),25),32)]の2つに分類できる．(i) の記述法は，構文解析システム PATR[27)]の文法記述で用いられているので，以後 PATR 形式と呼ぶ．

従来のシステムの多くは，(i) の文法記述から (a) の内部表現を生成して解析を行うか[15),27),30)]，(ii) から (b) を生成して解析を行うものである[18),24),25),32)]．しかし，後述するように，内部表現に関しては (b) の一階論理の項を用いるのが単一化の効率が良く，記述法としては (i) の方法が記述のしやすさ，可読性に優れている．そこで，(i) の記述形式から (b) の内部表現に変換する方法☆1がいくつか提案されているが[3),9),26)]，それらは次に述べる選言情報を含む文法を変換することができない．

選言情報を含む素性構造の記述（選言的素性記述）は，文法中の冗長性☆2を減らすのに有効であり[10),13),14)]，さらに，選言情報を含む内部表現を用いることにより構文解析の効率が向上する[4),5),8),11),20),31)]．したがって，選言情報を含んだ PATR 形式の文法記述を (b) の内部表現に変換する方法が望まれる．ただし，変換時に選言を展開し，選言を含まない素性構造を列挙した形（選言標準形）に展開してしまうと，処理効率が悪くなる[11),20)]．

本論文では，選言情報を含む PATR 形式の文法記述を，選言情報を保ったまま，項に基づく内部表現に変換する方法を提案し，実験結果によりその有効性を示す．この方法で用いる文法記述は，選言情報の記述を許すように PATR の記述法を拡張したもので，PATR-DDM（PATR with Disjunctively Defined Macros）記法と呼ぶ．文法の内部表現は，論理的制約[22)]に基づく表現（Logical-Constraint-Based Grammar Representation: LCGR）を用いる．内部表現が LCGR に基づく場合，制約単一化[8),31)]や制約射影[20)]などの論理的制約の処理による単一化法を用いることによって，あらかじめ選言を展開してから単一化する方法[13)]よりも効率良く構文解析を行うことができる[20),31)]．

本論文の構成は次のとおりである．2 章では，単一化文法の記述法と内部表現について考察し，PATR 形式の文法記述から項に基づく内部表現に変換することの意義を説明する．3 章では PATR-DDM を説明し，4 章では LCGR を説明する．5 章では PATR-DDM を LCGR に変換する方法を提案し，6 章では実験結果によって有効性を示す．

2. PATR 形式の文法記述から項に基づく内部表現への変換

2.1 単一化文法の記述法

単一化文法の記述法は，素性構造をどのようにして記述するかという観点から，(i) 素性構造を素性名と素性値のペアのリストや経路方程式を用いて記述するもの[15),19),27),30)]（PATR 形式），(ii) 素性構造を一階論理の項で記述するもの[18),24),25),32)] の2つに分類できる．

図 1 に記述の例を示す．(b1) は，言語学において (a) のように書かれる素性構造☆3を PATR 形式，すなわち素性名–素性値ペアのリストと経路方程式を用いて記述した例である．(b2) は一階の項による記述の例である．`f` は素性構造を表す関数記号で，その引数は順に，pos, agr, subj, num, per 素性に相当するとする．大文字で始まる記号 `X1`, `X2`, `X3` は変数を表し，"_" は無名変数を表す．無名変数は，1 回しか現れない変数を表すときに用いる．変数 `X1`, `X2`, `X3` は共有を表す☆4．

一階論理の項を用いる方法では，すべての素性構造のすべての素性の値を指定できるが，値の指定されていない素性，すなわち，具体化されていない変数が多いと，それらをすべて記述しなくてはならず，労力が必要となる．この労力を減らすため，素性構造に型を導入する．どのような素性を持ち得るかは，型によって決まる．そして，同じ型の素性構造のみが単一化できるとする．具体的には，型 t の素性構造を，関数記号 `t` を持つ項で表し，t と `t` は一対一に対応しているとする．図 1 (a) の素性構造の場合，全体の素性構造と subj 素性値を *sign* 型，agr 素性値を *agr* 型とし，これらの型を関数記号 `sign`, `agr` で表して，関数記号の各引数がどの素性値に対応するかを (1) のように与えると，図 1 (b2) の表す内容は (2) のように表すことができる．

☆1 いったん (a) の内部表現に変換してから (b) の内部表現に変換するものを含む．

☆2 本論文では文法は構文・意味規則と辞書項目を含んだものを指す．

☆3 図 1 で，pos は part of speech（品詞）素性，agr は agreement（一致）素性，subj は subject（主語）素性，num は number（数）素性，per は person（人称）素性を表す．sing は singular（単数）を，third は 3 人称を表す．

☆4 この記法ではタグや経路方程式によって陽に共有を表現していないが，2 箇所に現れる `f(X1, X2, X3, sing, third)` は同じ関数記号と引数を持っており，変数 `X1`, `X2`, `X3` が具体化されても同じ構造になるので，共有を表していることになる．

(a)
$$\begin{bmatrix} \text{pos} & \text{verb} \\ \text{agr} & \boxed{1}\begin{bmatrix} \text{num} & \text{sing} \\ \text{per} & \text{third} \end{bmatrix} \\ \text{subj} & \begin{bmatrix} \text{agr} & \boxed{1} \end{bmatrix} \end{bmatrix}$$

(b1)
```
<pos> = verb
<agr> = [num: sing
         per: third]
<subj agr> = <agr>
```

(b2)
```
f(verb, f(X1,X2,X3,sing,third),
  f(_,f(X1,X2,X3,sing,third),_,_,_),
  _, _)
```

図1 素性構造の記述例
Fig. 1 Descriptions of a feature structure.

(1) 関数記号 **sign**:
第1引数 pos, 第2引数 agr,
第3引数 subj
関数記号 **agr**:
第1引数 num, 第2引数 per

(2)
```
sign(verb, agr(sing, third),
     sign(_, agr(sing, third), _))
```

2.2 単一化文法の記述法の比較

単一化文法の記述の仕方は，理解，記述，変更が容易であるものが望ましい．これらの点に関して，上記の2種の記述法の比較を示す．

- 理解の容易さ
  項による記述は，どの引数がどの素性値に対応するのかがすぐには分からないという欠点がある．PATR形式では，各素性構造がどのような素性を持っているかがすぐ分かり，理解しやすい[6),28)]．
- 記述の容易さ
  素性値の記述と，値の共有の記述が単純であることが望まれる．PATR形式では，素性値の記述は素性名-素性値ペアのリストで，値の共有はタグや経路方程式で簡単に記述できる．項による記述では，どの素性がどの引数に対応するのかがすぐに分からないので，素性値の記述に労力がいる．また，共有を表すには，値の指定されていない素性をすべて同じ変数で表す必要がある．
- 変更の容易さ
  素性の種類を増減する場合を考えると，項では，その素性を持つ型のすべての項の引数を増減しなければならない．これに対し，PATR形式では，その素性が現れるところにだけ変更を加えればよい[28)]．

以上の3点より，PATR形式の方が項による記述より優れている．

2.3 単一化文法の内部表現

単一化文法の内部表現には，一般に2種類の方法が用いられる．1つは，素性構造をラベルつき有向グラフで表現する方法（グラフ表現）[6),15),27)]で，もう1つは，一階論理の項を用いる方法（項表現）[18),24),25),32)]である．

グラフ表現は，素性構造および素性値を節点で表し，素性名をラベルに持つ弧でこれらの節点を結んだものである[6)]．項表現は，DCG[25)]などのように，(2)のような項による記述を木構造を用いた内部表現に直接的に変換したものである[23)]．

2.4 単一化文法の内部表現法の比較

項表現では引数位置が素性を表すので，単一化は各引数どうしを順にマッチすることによって行うことができる．したがって単一化の計算時間は，引数の個数，すなわちその素性構造が持ち得る素性の数を n とすると，$O(n)$ になる．これに対しグラフ表現では，一方の素性構造が持つ素性の各々に対して，相手の素性構造が持つ素性のリストの中に同じ素性があるかどうかを探す必要がある．したがって，最悪の場合の計算時間は $O(n^2)$ になる[26)]．これより，項表現を用いた解析がグラフ表現を用いた解析より効率が良いと考えられる．

Schöter[26)]と Hirsh[9)]は，グラフ表現を用いた文法を項表現に実際に変換して，項表現を用いた方が効率が良いことを示している．Schöter[26)]の実験によると，項表現を用いた解析が，グラフ表現を用いた解析よりも3.8～5.4倍速い．

2.5 変換の必要性

以上のことから，記述法としてはPATR形式が良く，内部表現としては，項表現が良い．したがって，PATR形式で書かれた文法を項による内部表現に変換する方法が望まれる．

今までに提案された変換法[3),9),26)]は，選言情報を含まない単一化文法を対象にしている．選言情報を文法に導入することは，記述，処理の双方で利点がある．一部分だけ異なる素性構造を複数記述する代わりに，異なる部分のみを選言の形で記述することによって，全体の記述量を減少させることができる[10),13),14)]．また，処理に関しても，一部分だけ異なる素性構造の単一化を複数回試みる代わりに，選言情報を持った1つの素性構造の単一化を行い，その中で選言の処理を行うことによって，全体の処理をより効率的にできる[4),5),8),11),20),31)]．したがって，PATR形式で書かれた文法で選言情報を含むものを，選言情報を保ったまま，項を用いた内部表現に変換する方法が望まれる．

⟨文法記述⟩ ::= {⟨型定義⟩ | ⟨句構造規則⟩
| ⟨辞書項目⟩ | ⟨マクロ定義⟩}

⟨型定義⟩
::= (**deftype** ⟨型⟩ ⟨素性名⟩$_1$ ... ⟨素性名⟩$_n$)
[意味]⟨型⟩は，素性構造の型の名前で，この型の素性構造が⟨素性名⟩$_1$, ..., ⟨素性名⟩$_n$の素性を持ちうる．

⟨句構造規則⟩ ::=
(**defrule** ⟨規則名⟩
(⟨変数⟩$_0$ -> ⟨変数⟩$_1$ ... ⟨変数⟩$_n$)
(⟨経路方程式⟩$_1$ ... ⟨経路方程式⟩$_m$)
⟨選言定義マクロ⟩$_1$ ... ⟨選言定義マクロ⟩$_l$)
[意味]⟨変数⟩$_0$は親ノードの素性構造，⟨変数⟩$_1$, ..., ⟨変数⟩$_n$は子ノードの素性構造を表す．これらの素性構造が⟨経路方程式⟩$_1$, ..., ⟨経路方程式⟩$_m$および⟨選言定義マクロ⟩$_1$, ..., ⟨選言定義マクロ⟩$_l$の制約を満たすとき，素性構造⟨変数⟩$_1$, ..., ⟨変数⟩$_n$を持つ構成素の並びが素性構造⟨変数⟩$_0$の構成素になる．

⟨辞書項目⟩ ::=
(**defword** ⟨単語⟩ (⟨変数⟩)
(⟨経路方程式⟩$_1$ ... ⟨経路方程式⟩$_m$)
⟨選言定義マクロ⟩$_1$ ... ⟨選言定義マクロ⟩$_l$)
[意味]⟨変数⟩が表す素性構造が⟨経路方程式⟩$_1$, ..., ⟨経路方程式⟩$_m$および⟨選言定義マクロ⟩$_1$, ..., ⟨選言定義マクロ⟩$_l$の制約を満たすとき，素性構造⟨変数⟩は⟨単語⟩の語彙素性構造である．

⟨マクロ定義⟩ ::=
(**defddmacro** ⟨マクロ名⟩ (⟨変数⟩$_1$ ... ⟨変数⟩$_n$)
(⟨経路方程式⟩$_1$ ... ⟨経路方程式⟩$_m$)
⟨選言定義マクロ⟩$_1$ ... ⟨選言定義マクロ⟩$_l$)
[意味]素性構造⟨変数⟩$_1$, ..., ⟨変数⟩$_n$が⟨経路方程式⟩$_1$, ..., ⟨経路方程式⟩$_m$および⟨選言定義マクロ⟩$_1$, ..., ⟨選言定義マクロ⟩$_l$の制約を満たすとき，⟨マクロ名⟩で表されるマクロが満たされる．

図 2 PATR-DDM の構文と意味 (I)
Fig. 2 The syntax and semantics of PATR-DDM (I).

⟨選言定義マクロ⟩
::= (⟨マクロ名⟩ ⟨経路⟩$_1$ ... ⟨経路⟩$_n$)
[意味]⟨経路⟩$_1$, ..., ⟨経路⟩$_n$が表す素性構造が⟨マクロ名⟩のマクロ定義のうちのどれかの制約を満たす時，この選言定義マクロの制約が満たされる．

⟨経路方程式⟩ ::= ⟨経路⟩$_1$ = ⟨経路⟩$_2$
[意味]⟨経路⟩$_1$と⟨経路⟩$_2$が同一の素性構造であるという制約を表す．

⟨経路⟩ ::=
⟨値⟩ | <⟨変数⟩ ⟨素性名⟩$_1$ ... ⟨素性名⟩$_m$>
| [⟨素性名⟩$_1$: ⟨経路⟩$_1$ ... ⟨素性名⟩$_l$: ⟨経路⟩$_l$]
[意味]⟨値⟩の場合は，素性を持たない atomic な素性構造を表す．<⟨変数⟩ ⟨素性名⟩$_1$ ... ⟨素性名⟩$_m$>の場合は，⟨変数⟩が表す素性構造の⟨素性名⟩$_1$ ... ⟨素性名⟩$_m$で表される素性経路の値である素性構造を表す．[⟨素性名⟩$_1$: ⟨経路⟩$_1$ ... ⟨素性名⟩$_l$: ⟨経路⟩$_l$] の場合は，$1 \leq i \leq l$の各 i に対して，⟨素性名⟩$_i$で表される素性の値が⟨経路⟩$_i$が表す素性構造と同一である素性構造を表す．

⟨変数⟩ ::= ⟨シンボル⟩
⟨素性名⟩ ::= ⟨シンボル⟩
⟨型⟩ ::= ⟨シンボル⟩
⟨マクロ名⟩ ::= ⟨シンボル⟩
⟨規則名⟩ ::= ⟨シンボル⟩
⟨単語⟩ ::= ⟨シンボル⟩
⟨値⟩ ::= ⟨シンボル⟩

図 3 PATR-DDM の構文と意味 (II)
Fig. 3 The syntax and semantics of PATR-DDM (II).

3. 選言情報を含む単一化文法記述

本論文では，選言情報を含む文法記述の方法として，選言を扱えるように PATR[27]を拡張したものを用いる．これを PATR-DDM（PATR with Disjunctively Defined Macros）と呼ぶ．

PATR と PATR-DDM の主な違いは，PATR では1つのマクロの定義が1つしか記述できないのに対し，PATR-DDM では複数個記述できることである．複数の定義は選言の関係にあり，そのうちのどれかの条件を満たせば，そのマクロが満たされているとする．PATR-DDM では，型定義，句構造規則，辞書項目，マクロ定義によって文法記述を行う．PATR-DDM の構文と意味を図 **2**，図 **3** に示す．

以下に具体例を示す．まず，型定義の例を示す．

```
(deftype sign
  pos agr subj)
```

これは，sign という型が存在し，sign 型の素性構造が pos，agr，subj 素性を持ち得ることを意味する．

次に，句構造規則の例を示す．

(3)
```
(defrule psr1 (s -> np vp)
  (<s pos> = sentence
   <np pos> = noun
   <vp pos> = verb
   <vp subj> = <np>
   <np agr> = <vp agr>
   <s agr> = <vp agr>))
```

psr1 はこの規則の名前である．s は親カテゴリの素性構造を表す変数で，np と vp は娘の素性構造に相当する変数である．規則 psr1 は 3 つの素性構造 s，np，vp の関係を表す．第 4 項は経路方程式の集合である．経路方程式 <s pos> = sentence は，変数 s で表される素性構造の pos 素性の値が sentence であることを示す．経路方程式 <vp subj> = <np> は vp の subj 素性値が np の素性構造に等しいことを示す．経路は，⟨素性名⟩:⟨経路⟩ のリストの形でもよい．これを用いて (3) の代わりに，(4) のように記述することができる．

(4)
```
(defrule psr1 (s -> np vp)
  (<s> = [pos: sentence
          agr: <vp agr>]
```

```
<np> = [pos: noun
        agr: <vp agr>]
<vp> = [pos: verb
        subj: <np>]))
```

次に辞書項目の例を示す.

(5)
```
(defword walk (sign)
  (<sign pos> = verb
   <sign agr> = <sign subj agr>)
  (not3s <sign agr>))
```

sign が walk の語彙素性構造を表す変数である. 最後の (not3s <sign agr>) は選言定義マクロであり, sign の agr 素性値がマクロ not3s の定義のどれかを満たさなくてはならないことを示している.

次に, マクロ定義の例として not3s の定義を示す.

(6)
```
(defddmacro not3s (agr)
  (<agr num> = sing)
  (first-or-second <agr per>))
```

(7)
```
(defddmacro not3s (agr)
  (<agr num> = plural))
```

これらのどちらかが満たされれば, マクロ not3s の制約が満たされる. すなわち, (6) と (7) は選言の関係にある.

4. 選言情報を含む単一化文法の内部表現

選言情報を含む文法の項に基づく内部表現である, 論理的制約に基づく内部表現[22]を説明する.

まず, 論理的制約を説明する. 論理的制約 (以後制約と呼ぶ) は, 一階の確定節論理[17]の正リテラルの集合である. 本論文では, 制約を **B**, **C** などで表し, リテラルを T, H などで表す. リテラルは DEC-10 Prolog[2]の記法を用いて書く. X, Y など大文字で始まるものは変数である. 一度しか現れない変数は, 無名変数 '_' を用いて表す. (8) は制約の一例である.

(8) {p(X), q(X, f(Y))}

ある述語 *pred* の定義節は, 頭部の述語が *pred* である確定節のことである. たとえば, 次は p の定義節である.

(9) p(f(X, Y)) ← {r(X), s(Y)}

定義節の本体は, 頭部にある変数への制約である. たとえば, 定義節 (9) は X と Y のあるインスタンスに関し, それらのインスタンスが, {r(X), s(Y)} を満たせば, p(f(X, Y)) が真であることを意味する. 本体が空の場合は ← の右辺には何も書かない. システムに登録されている定義節の集合をデータベースと呼ぶ.

定義節と制約の意味は, 最小エルブランモデル[17]により規定される. このため, ある基礎原始式 (ground atomic formula)[17]が真であるということは, その基礎原始式がデータベースに存在する節から推論できるということに等しい. たとえば, 述語 p の定義節が次の 2 つだけであったとする.

p(a) ←
p(b) ←

この場合, p(a) と p(b) は真であるが, p(c) は真ではない. したがって, 制約 {p(X)} は, 変数 X が a か b であると規定していることになる.

選言を含む素性構造は, 述語とその定義節によって表現することができる. たとえば, (10) は, 述語 p と確定節の集合 (11) で表すことができる. ここで, sign, agr の引数に相当する素性は, (1) で規定されているとする.

(10)
$$\left[\begin{array}{l} \mathrm{pos}\ \ \mathrm{verb} \\ \mathrm{agr}\ \ \boxed{1}\ \left(\left[\begin{array}{l}\mathrm{num}\ \ \mathrm{sing}\\ \mathrm{per}\ \ \begin{pmatrix}\mathrm{first}\\ \vee\\ \mathrm{second}\end{pmatrix}\end{array}\right] \vee \left[\begin{array}{l}\mathrm{num}\ \ \mathrm{plural}\end{array}\right]\right) \\ \mathrm{subj}\ \left[\begin{array}{l}\mathrm{agr}\ \ \boxed{1}\end{array}\right] \end{array}\right]$$

(11)
```
p(sign(verb, Agr, sign(_, Agr, _)))
      ← {not3s(Agr)}
not3s(agr(sing, Per))
      ← {first_or_second(Per)}
not3s(agr(plural, _)) ←
first_or_second(first) ←
first_or_second(second) ←
```

このとき, {p(X)} を充足する X のインスタンス, たとえば, sign(verb, agr(sing, first), sign(_, agr(sing, first), _)) は, (10) が表す素性構造の 1 つである.

述語 p1 で表される選言的素性構造と述語 p2 で表される選言的素性構造の単一化は, {p1(X), p2(X)} を充足する X のインスタンスの集合を求めることによって行うことができる[7]. 単一化の結果を別の単一化で用いることを考えると, 結果を制約の形で出力することが望まれる. そのような充足方法を制約変換[7]という. 制約変換は, 制約が充足可能な場合は, 同等で簡単化された制約を求め, 充足不可能な場合は失敗する演算である. どのように簡単化されるかが, 新たな制約変換で用いられるときの効率を左右する. この演算の効率的な方法として, 制約単一化[8]や制約射影[20]などの制約の変換法が提案されている. これらの方法を用いると, 不必要な選言の展開は行われず, 変換され

てできた制約を再度用いるときに効率が良い.

次に論理的制約を用いた文法の内部表現法（LCGR: Logical-Constraint-Based Grammar Representation）[22]を示す. LCGRは，句構造規則の集合，辞書項目の集合，データベースからなる.

各句構造規則は，3つ組 $\langle V, \xi, \mathbf{C}\rangle$（Vは変数，$\xi$は変数のリスト，$\mathbf{C}$はVと$\xi$の変数への制約）である. これは，各変数のインスタンスが$\mathbf{C}$を満たせば，この規則によって許される構文構造をなすことを表す. たとえば，⟨X, [Y, Z], {psr1(X, Y, Z)}⟩ は，{psr1(X, Y, Z)} を満たすX, Y, Zのインスタンスの組x, y, zがあるとき，素性構造yを持つ句と素性構造zを持つ句の並びが，素性構造xを持つ句になることを示している.

各辞書項目は，2つ組 ⟨w, p⟩（wは単語，pは述語）である. これは，{p(X)} を満たすXのインスタンスが，単語wの語彙素性構造になることを示す. たとえば，⟨walk, lex_walk⟩ は，{lex_walk(X)} を満たすXのインスタンスが，walkの語彙素性構造であることを示す.

データベースは確定節の集合である. 制約で用いられている述語，および，データベース中の確定節の本体にある述語は，すべてデータベースにその定義節があるとする.

5. PATR-DDMからLCGRへの変換

5.1 変換のアルゴリズム

PATR-DDMで記述された文法からLCGRによる内部表現への変換は，次のような手順で行う.

(i) 型定義から素性値を表す述語を作る.

(ii) 句構造規則，辞書項目，マクロ定義をLCGRの要素に変換する.

(iii) 確定節から冗長性を取り除く.

5.1.1 素性値を表す述語の作成

型定義の一般形は，

```
(deftype t
    f1 f2 ... fn)
```

である. これから，変数Xのf_i素性がY_iであることを示すリテラル $f'_i(X, Y_i)$ の述語，f'_i の定義節を作る. これは，以下の定義節をデータベースに加えることによって行う.（以下の説明では，PATR-DDMでのシンボルxをLCGRの表現に変換したものをx′で表す.）

$f'_1(t'(X, \_, \ldots, \_), X) \leftarrow$

$f'_2(t'(\_, X, \ldots, \_), X) \leftarrow$

...

$f'_n(t'(\_, \_, \ldots, X), X) \leftarrow$

これにより，$t'(\_, \_, \ldots, \_)$ の形の n 引数の項が，t型の素性構造を表し，各引数が順に，f_1 素性，... f_n 素性を表す.

たとえば次の型定義があったとする.

```
(deftype sign
    pos agr subj)
```

すると，sign型の素性構造は，sign(_, _, _) の3引数の項で表され，各引数がpos素性値，agr素性値，subj素性値を表す. これを用いて，次の3つの確定節を作り，データベースに加える.

(12) $pos(sign(X, \_, \_), X) \leftarrow$
$agr(sign(\_, X, \_), X) \leftarrow$
$subj(sign(\_, \_, X), X) \leftarrow$

5.1.2 句構造規則，辞書項目，マクロ定義の確定節への変換

句構造規則，辞書項目，マクロ定義は，それぞれ1つの確定節に変換し，データベースに加える. 以下に変換のアルゴリズムを示す☆. この過程で，PATR-DDMの変数は同じ名前のLCGRの変数に，素性名，型，値は，同じ名前のLCGRの定数に，マクロ名は同じ名前のLCGRの述語に変換される.

(I) 頭部となるリテラルを作る. 句構造規則，辞書項目の場合は，今まで使われていないシンボルを述語とし，第3要素に現れる変数（句構造規則の場合は，親ノードと子ノードを表す変数，辞書項目の場合は，語彙素性構造を表す変数）を引数とする. マクロ定義の場合は，マクロ名を述語とし，同じく第3要素に現れる変数（マクロの引数である変数）を引数とする.

(II) 経路方程式と選言定義マクロから，頭部に対する制約を求め，それを本体として確定節を作る. 具体的には次のように行う.

(IIa) 本体を空とする.

(IIb) 経路方程式と選言定義マクロに現れる各経路について次を行う. (i) 経路がシンボルならLCGRの定数に置き換える. (ii) シンボルでない場合は，以下のようにして変数に置き換える. 経路を<v_1 f_1 ... f_n>とし，v_1 に相当するLCGRの変数 v'_1 の f_1 素性の値を変数 v'_2 で表すとすると，型定義の変換で作られた述語 f'_1 を用いて v'_1, v'_2 の関係を $f'_1(v'_1, v'_2)$ というリテラルで表すことができる. 同様に f_n 素性の値を変数 v' で表すと，経路は

☆ 素性名–素性値ペアのリストを表す[...]の扱いは同様に行えるので省略する.

v' で置き換えられる．そして $\{f'_1(v'_1, v'_2), \ldots, f'_n(v'_n, v')\}$ を本体に加える．

(IIc) 各経路方程式は (IIb) で $v' = w'$ の形になっているので，これからリテラル $eq(v', w')$ を作り本体に加える．ここで，eq は同一性を表す述語で，次のように定義される．

$eq(X, X) \leftarrow$

(IId) 各選言定義マクロは，(IIb) で $(p\ v'_1 \ldots v'_n)$ の形になっている．これから，リテラル $p'(v'_1, \ldots, v'_n)$ を作り，本体に加える．

(III) (I) で作った述語を用いて，LCGR における句構造規則と辞書項目を作る．

次の walk の辞書項目を例にして説明する．

```
(defword walk (sign)
  (<sign pos> = verb
   <sign agr> = <sign subj agr>)
  (not3s <sign agr>))
```

まず，(I) で新しい述語 c0 と sign に対応する LCGR の変数 Sign が作られ，頭部は c0(Sign) となる．次に (IIb) で，2 行目の<sign pos>は，変数 X1 に置き換えられ，pos(Sign, X1) が本体に加えられる．verb は定数 verb に置き換えられる．3 行目の左辺の<sign agr>は X2 に，右辺の<sign subj agr>は X4 に置き換えられ，agr(Sign, X2)，subj(Sign, X3)，agr(X3, X4) が，本体に加えられる．そして，経路方程式は，

(13) X1 = verb
X2 = X4

となる．また，4 行目の<sign agr>は X5 に置き換えられ，agr(Sign, X5) が本体に加えられる．マクロ (not3s <sign agr>) は，(not3s X5) になる．(IIc) で (13) の経路方程式から eq(X1, verb) と eq(X2, X4) が作られて本体に加えられる．(IId) で not3s(X5) が本体に加えられる．

したがって，

(14) c0(Sign) ← {pos(Sign, X1), agr(Sign, X2), subj(Sign, X3), agr(X3, X4), agr(Sign, X5), eq(X1, verb), eq(X2, X4), not3s(X5)}

が，データベースに登録される．最後に ⟨walk, c0⟩ が辞書項目になる．

句構造規則やマクロ定義も同様にして変換する．句構造規則 (15) からは，(16) の確定節が作られ，⟨S, [NP, VP], {c1(S, NP, VP)}⟩ が句構造規則として登録される．

(15)
```
(defrule psr1 (s -> np vp)
  (<s pos> = sentence
   <np pos> = noun
   <vp pos> = verb
   <vp subj> = <np>
   <np agr> = <vp agr>
   <s agr> = <vp agr>))
```

(16) c1(S, NP, VP) ← {pos(S, X1), pos(NP, X2), pos(VP, X3), subj(VP, X4), agr(NP, X5), agr(VP, X6), agr(S, X7), agr(VP, X8), eq(X1, sentence), eq(X2, noun), eq(X3, verb), eq(X4, NP), eq(X5, X6), eq(X7, X8)}

また，マクロ定義 (17) からは，(18) のような確定節が作られる．

(17)
```
(defddmacro not3s (agr)
  (<agr num> = sing)
  (first-or-second <agr per>))
```

(18) not3s(Agr) ← {num(Agr, X1), per(Agr, X2), eq(X1, sing), first-or-second(X2)}

以上の変換で，あるマクロ m に複数の定義があると，述語 m' に複数の定義節ができるので，選言が展開されずに変換が行われる．

5.1.3 確定節の縮約による冗長性の除去

前項の方法で作られた確定節では，同一性を表す述語 eq，素性値を表す述語，および単一の定義しか持たないマクロを表す述語など，定義節を 1 つしか持たない述語が多く使われている．もし，このままの形で解析に用いると，その 1 つしかない定義節を同じ句構造規則や辞書項目が使われる度に調べなくてはならず，解析効率が悪い．

このため，**図 4** にアルゴリズムを示す手続き reduce-dc を用いて，定義節を 1 つしか持たない述語を本体と置き換える．この手続きを縮約と呼ぶ．

例として，上記の (14) を変換する．A3 で，

(19) reduce ({pos(Sign, X1), agr(Sign, X2), subj(Sign, X3), agr(X3, X4), agr(Sign, X5), eq(X1, verb), eq(X2, X4), not3s(X5)})

が呼ばれる．B3 で，L = pos(Sign, X1) とすると，pos の定義節は，(12) で作られた，次の定義節である（後のステップの説明のために，無名変数を通常の変数で書いておく）．

pos(sign(X, Y, Z), X) ←

単一化を行うと，θ ={Sign/sign(X, Y, Z), X1/X} となる．次に，B4 で，

reduce ({agr(sign(X, Y, Z), X2), subj(sign(X, Y, Z), X3), agr(X3, X4), agr(sign(X, Y, Z), X5),

表1 実験結果
Table 1 Experiment results.

入力文（わかち書きしたもの）	特徴	解析時間A（秒）	解析時間B（秒）
誰 が 京都 に 行っ た の	疑問詞疑問文	6.05	1.42
鈴木さん は 岡山 で 生まれ 東京 で 育っ た	並列構造	167.49	13.11
太郎 が 花子 に 数学 を 喫茶店 で 教え た	単文	5.52	1.79
山田君 に マージャン の ルール を 教え た の は 課長 だ	AはBだの型の文	21.87	4.42
京都 で 学会 が あっ た 時 に 僕 も 行き ま し た	形式名詞を含む文	22.94	5.03
太郎 が 花子 に 教え た 道 を 健 が 奈緒美 に 教え た	連体修飾節を含む文	12.75	3.44
鈴木さん は 大学 で 物理 を 勉強 し た が 今では 言語 の 研究 を し ている	接続助詞を含む長文	78.19	10.24

解析時間A：選言を展開してから変換した文法を用いたときの解析時間
解析時間B：選言を展開せずに変換した文法を用いたときの解析時間

Procedure reduce-dc (DC：確定節)
A1. DCをデータベースから取り除く．
A2. HをDCの頭部，**B**をDCの本体とする．
A3. reduce(**B**)を行い，"fail"が返れば終了し，$(\theta, \mathbf{B}')$が返れば，$H\theta \leftarrow \mathbf{B}'$をデータベースに登録する．
Function reduce (**C**：制約) は，代入と制約の2つ組か，もしくは，"fail"を返す．
B1. 定義節のない述語を持つリテラルが**C**にあれば，"fail"を返す．
B2. $\mathbf{C} = \{\}$，または，**C**のすべてのリテラルの述語が定義節を2つ以上持てば，$(\epsilon, \mathbf{C})$を返す（ϵは空代入[17]）．
B3. 述語が定義節を1つだけ持つあるリテラル$L \in \mathbf{C}$に対して，その定義節を$H \leftarrow \mathbf{B}$とするとき，HとLの単一化を試み，失敗すると"fail"を返す．成功すれば，θをHとLの最汎単一化子（most general unifier[17]）とする．
B4. reduce($(\mathbf{B} \cup (\mathbf{C} - \{L\}))\theta$)を行い，その値が"fail"なら"fail"を返し，$(\sigma, \mathbf{C}')$なら$(\theta\sigma, \mathbf{C}')$を返す．

図4 縮約のアルゴリズム
Fig. 4 Algorithm for reduction.

eq(X, verb), eq(X2, X4), not3s(X5)})
が呼ばれる．同様の操作を繰り返すと，このreduceは，
({X/verb, Y/X6, Z/sign(X7, X6, X8)},
{not3s(X6)})
を返す．ただし，以後使われない変数の束縛は省いた．ここで，not3s(X6)は述語が定義節を2つ持つのでこれ以上縮約されない．こうして，(19)のreduceは，
({Sign/sign(verb, X6, sign(X7, X6, X8))},
{not3s(X6)})
を返す．reduce-dcに戻り，A3で，
c0(sign(verb, X6, sign(X7, X6, X8)))
← {not3s(X6)}
がデータベースに加えられる．上述したように，以上の変換操作で述語not3sで表現されている選言は展開されないので，冗長性が取り除かれる一方で選言情報は保たれている．

縮約の操作では，定義節を2つ以上持つ述語を用いた項を定義節の本体で置き換えないため，置換え操作を行ったときには失敗する場合でも変換が成功してしまう場合がある．本論文の変換法は，矛盾のない文法記述，すなわち，上記の置換え操作を行ったとしても失敗しないような文法記述を前提としている．このことと，構文解析の効率が良い内部表現への変換の観点から，選言の展開に相当する上記の置換え操作は行わない．

6. 実　　験

過去の変換法と比較した場合の，本論文で提案した変換法の利点は，選言情報を展開することなく変換できる点にある．したがって，選言情報を展開してから変換した文法と，本論文の方法により選言情報を保ったまま変換した文法の2種類を用いて構文解析実験を行い，解析にかかった時間を比較した．

実験は，メインメモリ384メガバイトを持つSun SparcStation 20，SunOS 4.1.3上のLucid Common Lisp 4.0.0を用いて行った．実験で用いた文法は，下位範疇化，受動文，使役文，話題化，疑問文，等位接続，否定文，「AはBだ」の形の文，連体修飾，接続助詞を含む文などの，日本語の基本的な文型の文を扱うことができる．構文解析は上昇型のチャート解析[12]を用い，単一化は制約射影[22]を用いた．文法は，(1)選言を展開してから変換し，選言を含まない素性構造を列挙した形の内部表現（解析時間A），(2)本論文の方法で，選言情報を保ったまま確定節に変換してできた内部表現（解析時間B），の2種類を用い，全解探索を行う時間を計測した（**表1**）．その結果，すべての入力文を(1)よりも(2)の方が短い時間で解析することができた．これにより，選言情報を保ったまま変換する本方法の利点が確認できた．

7. おわりに

本論文では，素性名-素性値ペアと経路方程式を用いた，選言情報を含む単一化文法記述を，選言情報を保ったまま論理的制約に基づく内部表現に変換する手法を示した．この手法により，記述，理解，変更の容易な形式の単一化文法記述を，処理効率の良い内部表現に変換することができる．本論文で示した変換法は，対話文解析実験システム[16),21),29)]で用いている．

謝辞 日頃ご指導いただく石井健一郎情報科学研究部長，討論していただいた小暮潔氏，堂坂浩二氏，川森雅仁氏，論理プログラミングと制約処理に関して御教示をたまわった山崎憲一氏，橋田浩一氏，津田宏氏，実験システムの作成を手伝っていただいた井上みづほ氏，静洋一郎氏に感謝いたします．

参考文献

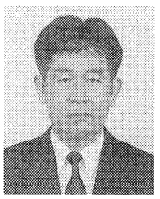
1) Allen, J.: *Natural Language Understanding* (second ed.), Benjamin/Cummings (1995).
2) Clocksin, W.F. and Mellish, C.S.: *Programming in Prolog*, Springer-Verlag (1984).
3) Covington, M.: GULP 2.0: An Extension of Prolog for Unification-Based Grammar, Technical Report AI-1989-01, The University of Georgia (1989).
4) Dörre, J. and Eisele, A.: Feature Logic with Disjunctive Unification, *Proc. 13th COLING*, Vol.2, pp.100–105 (1990).
5) Eisele, A. and Dörre, J.: Unification of Disjunctive Feature Descriptions, *Proc. 26th ACL*, pp.286–294 (1988).
6) Gazdar, G. and Mellish, C.: *Natural Language Processing in Lisp: An Introduction to Computational Linguistics*, Addison-Wesley (1989).
7) 橋田浩一：情報の部分性と制約プログラミング，制約論理プログラミング，古川康一，溝口文雄，Lassez, J.-L.（編），共立出版，pp.123–142 (1989).
8) Hasida, K.: Conditioned Unification for Natural Language Processing, *Proc. 11th COLING*, pp. 85–87 (1986).
9) Hirsh, S.: P-PATR: A Compiler for Unification-based Grammars, *Natural Language and Logic Programming*, II, Dahl, V. and Saint-Dizier, P. (Eds.), pp.63–78, Elsevier Science Publishers (1988).
10) Karttunen, L.: Features and Values, *Proc. 10th COLING*, pp.28 33 (1984).
11) Kasper, R.T.: A Unification Method for Disjunctive Feature Descriptions, *Proc. 25th ACL*, pp.235–242 (1987).
12) Kay, M.: Algorithm Schemata and Data Structures in Syntactic Processing, Technical Report CSL-80-12, Xerox PARC (1980). Reprinted in Grosz, B.J., Johns, K.S. and Webber, B.L. (Eds.): *Readings in Natural Language Processing*, Morgan Kaufmann (1986).
13) Kay, M.: Parsing in Functional Unification Grammar, *Natural Language Parsing: Psychological, Computational and Theoretical Perspectives*, Dowty, D.R., Karttunen, L. and Zwicky, A.M. (Eds.), pp.251–278, Cambridge University Press (1985).
14) 小暮 潔：増進的複製による型付き素性構造汎化手法，情報処理学会論文誌，Vol.34, No.9, pp.1993–1930 (1993).
15) Kogure, K., Iida, H., Hasegawa, T. and Ogura, K.: NADINE–An Experimental Dialogue Translation System from Japanese to English, *Proc. InfoJapan90*, Vol.2, pp.57–64, Tokyo, Japan (1990).
16) Kogure, K., Shimazu, A. and Nakano, M.: Recognizing Plans in More Natural Dialogue Utterances, *Proc. ICSLP-94*, pp.935–938 (1994).
17) Lloyd, J.W.: *Foundations of Logic Programming*, Springer-Verlag (1984).
18) Matsumoto, Y., Tanaka, H., Hirakawa, H., Miyoshi, H. and Yasukawa, H.: BUP: A Bottom-up Parser Embedded in Prolog, *New Generation Computing*, Vol.1, pp.145–158 (1983).
19) Mukai, K. and Yasukawa, H.: Complex Indeterminates in Prolog and Its Application to Discourse Models, *New Generation Computing*, Vol.3, No.4, pp.145–158 (1985).
20) Nakano, M.: Constraint Projection: An Efficient Treatment of Disjunctive Feature Descriptions, *Proc. 29th ACL*, pp.307–314 (1991).
21) Nakano, M., Shimazu, A. and Kogure, K.: A Grammar and a Parser for Spontaneous Speech, *Proc. 15th COLING*, pp.1014–1020 (1994).
22) 中野幹生，島津 明：論理的制約の射影演算を用いた単一化に基づく構文解析，情報処理学会論文誌，Vol.36, No.1, pp.22–31 (1995).
23) Norvig, P.: *Paradigms of Artificial Intelligence Programming*, Morgan Kaufmann (1992).
24) Pereira, F.C.N.: Extraposition Grammars, *American Journal of Computational Linguistics*, Vol.7, No.4, pp.243–256 (1981).
25) Pereira, F.C.N. and Warren, D.H.D.: Definite Clause Grammars for Language Analysis–A Survey of the Formalism and a Comparison with Augmented Transition Networks, *Artif.*

Intel., Vol.13, pp.231–278 (1980).
26) Schöter, A.: Compiling Feature Structures into Terms: An Empirical Study in Prolog, Technical Report EUCCS/RP-55, Centre for Cognitive Science, University of Edinburgh (1993).
27) Shieber, S.M.: The Design of a Computer Language for Linguistic Information, *Proc. 10th COLING*, pp.362–366 (1984).
28) Shieber, S.M.: *An Introduction to Unification-based Approaches to Grammar*, CSLI (1986).
29) Shimazu, A., Kogure, K. and Nakano, M.: Cooperative Distributed Processing for Understanding Dialogue Utterances, *Proc.ICSLP-94*, pp.99–102 (1994).
30) Tsuda, H.: cu-Prolog for Constraint-based Natural Language Processing, *IEICE Transactions on Information and Systems*, Vol.E77-D, No.2, pp.171–180 (1994).
31) Tuda, H., Hasida, K. and Sirai, H.: JPSG Parser on Constraint Logic Programming, *Proc. 4th European Chapter of ACL*, pp.95–102 (1989).
32) 徳永健伸，岩山 真，田中穂積：論理文法におけるギャップの扱い，情報処理学会論文誌，Vol.32, No.11, pp.1355–1365 (1991).

(平成8年5月31日受付)
(平成9年1月10日採録)

中野 幹生（正会員）

昭和40年生．昭和63年東京大学教養学部基礎科学科第一卒業．平成2年同大学院理学系研究科修士課程修了．同年日本電信電話（株）入社．以来，同社基礎研究所において，自然言語処理，対話理解の研究に従事．人工知能学会，言語処理学会，日本ソフトウェア科学会，ACL各会員．



島津 明（正会員）

昭和23年生．昭和46年九州大学理学部数学科卒業．昭和48年同大学院修士課程修了．同年日本電信電話公社武蔵野電気通信研究所入所．現在，NTT基礎研究所情報科学研究部対話理解研究グループリーダ．言語コミュニケーションの研究に従事．工学博士．言語処理学会，計量国語学会，電子情報通信学会，人工知能学会，ACL，ACM各会員．