自然言語処理 90－2
（1992. 7. 17）

素性論理に基づいた文法記述システム

李 航 †　　落合 尚良 ‡

† 日本電気(株) C&C 情報研究所

‡ 日本電気技術情報システム開発(株)

本稿では素性論理に基づいた文法記述システムについて述べる。素性論理に基づいた文法記述システムには、モジュラー性が高く拡張性が高い等のメリットがある。しかし、素性論理の無矛盾性チェックがNP完全であるため、素性論理をそのまま用いた文法記述システムでは処理の計算量が大きくなってしまう。そこで、我々は、自然言語記述のニーズをできるだけ損なわないように、素性論理の記述機能に幾つかの制限を加え、記述機能の制限された素性論理に基づいた、強力でかつ高速な文法記述システムを開発した。

A Grammar Description System Based On Feature Logic

Hang LI †　　**Takayoshi OCHIAI** ‡

† C&C Information Technology Research Laboratories, NEC

‡ NEC Scienticfic Information System Development,Ltd.

Miyazaki 4-1-1, Miyamae-ku, Kawasaki 216, Japan

This paper describes a grammar description system based on feature logic. Feature logic privides powerful functions for grammar descriptions. But since the consistency checking of a feature logic formula is NP-complete, The efficiency of systems based on feature logic becomes a serious problem. After trading off description functions and efficiency, we restrict some functions of feature logic. As a result, a powerful and efficient grammar description system has been developed.

1 はじめに

自然言語処理システムでは膨大な量の知識を必要とする。どのような知識を利用すれば、自然言語処理の質が向上するかは重要な研究課題である。それに加えて、利用しようとする知識をどのようにシステムで表現するかというのも同様に重要である。様々な知識をうまく表現し、システムでうまく管理することができなければ、優れた自然言語処理システムの実現が不可能である。本稿で述べる内容は、知識をいかに表現し、自然言語処理に役立てるかに関するものである。

近年、素性論理に関する研究が盛んに行われている。素性とは、ラベルとその値のペアのことである。プリミティブな自然言語の知識はほとんど素性によって表現することができる。例えば、英語の人称と数を素性によって表現することができる。素性論理 (Feature Logic) とは、素性に関する論理学の一種である。素性論理によって自然言語処理のための知識を表現することが何人かの人によって提案された。それには以下のメリットがある。

- 知識の追加と変更が容易である。素性論理によって記述される知識とそれを解釈するインタープリタが互いに独立しているため、文法作成者が知識の追加と変更を容易に行える。
- 知識の表現が簡潔である。選言や否定等によって知識を簡潔に記述することができる。
- 知識の管理が容易である。束と素性論理式によって階層的な知識と、制約としての知識を記述することができる。そのため知識のモジュラー性が高い。

しかし、素性論理の欠点は、それに基づいて知識を表現する自然言語処理システムの処理の計算量が大きくなることである (素性論理の無矛盾性チェックは NP 完全である)。この問題はまだ解決されていない。我々の研究目標は素性論理を用いながら、高速な自然言語処理のための文法記述システム (知識表現システム) を実現することである。

我々のとったアプローチは、自然言語処理のための知識表現のニーズを損なわないように素性論理の記述機能を制限することである。しかし、素性論理の記述機能を制限し過ぎると、多くの文法記述機能が失われるので、文法記述機能と処理速度のトレードオフを考慮する必要がある。従って、「どのように素性論理を制限すれば、自然言語処理の応用に役立つか」ということを研究する必要がある。それがまさに本研究の中心課題である。

我々は記述機能の制限された素性論理に基づいて文法記述システムを開発した。その文法記述システムによって自然言語インタフェースと機械翻訳システムを構築し、強力かつ高速な自然言語処理システムが得られた。

本稿の構成は以下のとおりである。第 2 節では、素性論理の形式化について紹介する。第 3 節では、自然言語処理の応用を損なわないように我々はどのように素性論理の記述機能に制限を加えたかについて述べる。第 4 節では、記述機能の制限された素性論理に基づいた文法記述システムの実現について述べる。第 5 節では、その文法記述システムの利用し方について述べる。第 6 節でまとめる。

2 素性論理

素性論理の形式化は様々な形で行われてきた [Johnson88] [Smolka88][Kasper86][Moshier87][Dawar90][李 91]。それらを大きく二つのグループに分類することができる。[Johnson88] と [Smolka88] ではブール論理の立場から素性論理を定義している。一方、[Moshier87] と [Dawar90] と [李 92] では、非ブール論理の立場から素性論理を定義している。この節では統一した観点から二種類の素性論理の形式化を試みる。まず基本的な定義を行う。それからブール束による素性論理の形式化 (ブール論理の立場) について述べる。さらに Brouwerian 束による素性論理の形式化 (非ブール論理の立場) について述べる。なお、この形式化は Smolka の形式化に修正を加えたものである。

2.1 基本的な定義

素性論理を定義する前に、まずシグネチェアを定義する。シグネチェアとは、以下の条件を満足するタップル $\langle T, F, \preceq \rangle$ のことである。

1. タイプの集合 T は $\bot$ と $\top$ を含む。
2. T の上に半順序関係 $\preceq$ が定義される。
   (a) $\top$ が最大要素で、$\bot$ が最小要素である。
   (b) T のすべての要素 A、B は最小上界 $A \sqcup B$ と最大下界 $A \sqcap B$ をもつ。
3. 素性シンボルの集合 F は $F \cap T = \emptyset$ を満足する。

シグネチェアは素性論理を表現するためのプリミティブである。特に、タイプの集合 T が束を形成する。次にシグネチェアによって対象領域を定義する。タイプが集合と対応するとする。さらに、素性が普遍集合の部分集合から普遍集合への部分関数と対応するとする。

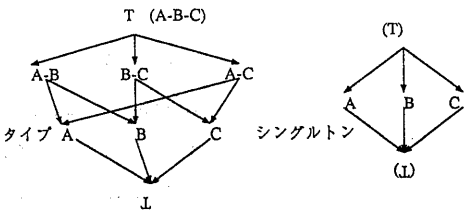

図 1: タイプの束とシングルトンの束

1. $\top^A$ は普遍集合である。
2. $\bot^A$ は空集合である。
3. $A \in T$ ならば、A^A は $\top^A$ の部分集合である。
4. $A \sqcup B = C$ ならば、$C^A = A^A \cup B^A$ である。$A \sqcap B = C$ ならば、$C^A = A^A \cap B^A$ である。
5. D_f^A を $\top^A$ の部分集合とする。しかも、D_f^A と A^A が排他的であるとする。$f \in F$ ならば、f^A は部分関数 $D_f^A \rightarrow \top^A$ である。

次に素性論理式を定義する。

$$\begin{array}{llll} A & \text{タイプ} & s \wedge t & \text{積} \\ f:s & \text{素性} & s \vee t & \text{和} \\ p \downarrow q & \text{一致} & \neg t & \text{補} \\ p \uparrow q & \text{不一致} & & \end{array} \quad (1)$$

素性論理式の解釈は以下のように定義する [Smolka88]。

$$\begin{array}{lcl} [\![A]\!]^{\mathcal{A}} & = & A^{\mathcal{A}} \\ [\![f:s]\!]^{\mathcal{A}} & = & \{a \in D_f^{\mathcal{A}} | f^{\mathcal{A}}(a) \in [\![s]\!]^{\mathcal{A}}\} \\ [\![p \downarrow q]\!]^{\mathcal{A}} & = & \{a \in D_p^{\mathcal{A}} \cap D_q^{\mathcal{A}} | p^{\mathcal{A}}(a) = q^{\mathcal{A}}(a)\} \\ [\![p \uparrow q]\!]^{\mathcal{A}} & = & \{a \in D_p^{\mathcal{A}} \cap D_q^{\mathcal{A}} | p^{\mathcal{A}}(a) \neq q^{\mathcal{A}}(a)\} \\ [\![s \wedge q]\!]^{\mathcal{A}} & = & [\![s]\!]^{\mathcal{A}} \cap [\![t]\!]^{\mathcal{A}} \\ [\![s \vee q]\!]^{\mathcal{A}} & = & [\![s]\!]^{\mathcal{A}} \cup [\![t]\!]^{\mathcal{A}} \\ [\![\neg s]\!]^{\mathcal{A}} & = & \top^{\mathcal{A}} - [\![s]\!]^{\mathcal{A}} \end{array} \quad (2)$$

つまり、素性論理の論理積が対象の積集合を表し、論理和が対象の和集合を表し、また否定が対象の補集合を表す。素性論理式 s は、解釈 $[\![s]\!]^{\mathcal{A}} \neq \emptyset$ が存在するならば、無矛盾であるという。

一方、以下の領域方程式の解 U をタイプ付き素性構造という。

$$\mathbf{U} = (\mathbf{T} - \{\bot\}) \times (\mathbf{F} \rightarrow \mathbf{U}) \quad (3)$$

2.2 ブール的解釈

Smolka の形式化ではタイプが対象領域の集合と一対一に対応すると解釈する。それはタイプがブール束を形成することを意味する。タイプの集合がブール束を形成する時にタイプ付き素性構造の集合 U がブール束を形成する。従って、Smolka の解釈ではタイプ付き素性構造の集合と対象領域が同形である。素性論理式に対して、$l1$ から $l22$ までの推論規則が成り立つ。

Smolka の形式化ではさらに次のようにシングルトンを定義している。シングルトンは以下の条件を満足するタイプ A のことである。

$$A \succ \bot, \text{and } \nexists B, A \succ B \succ \bot \quad (4)$$

Smolka の形式化ではシングルトンがフラットな束を形成する。例えば、図 1ではタイプの集合がブール束を形成し、シングルトンの集合 $\{A, B, C\}$ がフラットな束を形成する。シングルトンからタイプ全体を定義することができるので、実際の応用ではタイプを定義するためにシングルトンだけを定義すればよい。

2.3 非ブール的解釈

Smolka の形式では、タイプがブール束を形成する。また、シングルトンがフラットな束を形成する。しかし、シングルトンがフラットな束を形成すると仮定する必要がない。シングルトンが束を形成すると仮定するのが応用上望ましい。シングルトンが任意の束を形成する時に、タイプが Brouwerian 束を形成する [Aït-Kaci86]。例えば、図 2においてシングルトンが束を形成する時に、タイプが Brouwerian 束を形成する。この仮定ではタイプ付き素性構造の集合 U が Brouwerian 束を形成する。というのは Brouwerian 束の直積が Brouwerian 束だからである。この仮定に基づいて以下のように素性論理を解釈する。タイプが対象領域の集合と対応するとする。しかも、タイプ $A \preceq B$ であれば、それに対応する対象領域の集合は $A^{\mathcal{A}} \subseteq B^{\mathcal{A}}$ を満足

$$\begin{array}{lll} l1) & s \vee t & = t \vee s \\ l2) & s \wedge t & = t \wedge s \\ l3) & s \vee (t \vee u) & = (s \vee t) \vee u \\ l4) & s \wedge (t \wedge u) & = (s \wedge t) \wedge u \\ l5) & s \vee (t \wedge u) & = (s \vee t) \wedge (s \vee u) \\ l6) & s \wedge (t \vee u) & = (s \wedge t) \vee (s \wedge u) \\ l7) & s \vee s & = s \\ l8) & s \wedge s & = s \\ l9) & s \vee (s \wedge t) & = s \\ l10) & s \wedge (s \vee t) & = s \\ l11) & s \vee \neg s & = \top \\ l12) & s \wedge \neg s & = \bot \\ l13) & \neg(s \vee t) & = \neg s \wedge \neg t \\ l14) & \neg(s \wedge t) & = \neg s \vee \neg t \\ l15) & \neg(\neg s) & = s \\ l16) & s \vee \top & = \top \\ l17) & s \vee \bot & = s \\ l18) & s \wedge \top & = s \\ l19) & s \wedge \bot & = \bot \\ l20) & f:(s \vee t) & = (f:s) \vee (f:t) \\ l21) & f:(s \wedge t) & = (f:s) \wedge (f:t) \\ l22) & \neg f:s & = (\neg f:\top) \vee (f:\neg s) \end{array}$$

するとする。この解釈をすれば、素性論理式が Brouwerian 束の演算則を満足する。つまり、この解釈は非ブール的である。具体的には、否定に関する推論規則 $l14$、$l15$ が成り立たなくなる。例えば、図 2において、

$$\neg(\neg female) \neq female \quad (5)$$

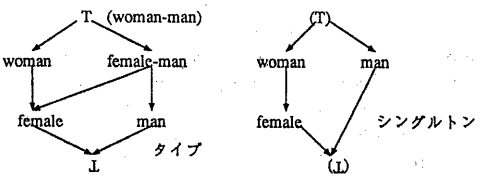

図 2: Brouwerian 束の例

以下では便宜的に対象領域の要素のことをタイプつき素性構造 (Typed Feature Structure, TFS) と呼ぶことにする。素性論理式の論理積の無矛盾性チェックを単一化といい、素性論理式の論理和の無矛盾性チェックを一般化という。これは、論理積と論理和がそれぞれタイプ付き素性構造の最大下界と最小上界を表しているからである。また、説明の都合によって、素性論理式をマトリックス記法によって表現する。マトリックス記法では、[] によって連言を、{} によって選言を、¬ によって否定を、タグによって一致を表現する。

3 素性論理の記述機能に対する制限

素性論理によって文法を強力に記述することができる。しかし、Kasper と Smolka が証明したように否定、選言、一致を含む素性論理式の無矛盾性チェックが NP 完全である [Smolka88]。我々は処理速度を向上させるために、素性論理をそのまま用い

て文法記述システムを実現するのではなく、素性論理の一部の記述機能を犠牲にし、素性論理の記述機能を制限するアプローチをとる。しかし、素性論理の記述機能を制限し過ぎると、多くの記述機能が失うので、記述機能と処理速度をトレードオフする必要がある。

この節では、素性論理の記述機能に対する制限について述べる。なお、素性論理はBrouwerian束によって解釈するものを採用する。従って、予め束を形成するシングルトンが与えられたとする。

3.1 選言

本研究では、Kasperの定義[Kasper86]にしたがって任意の選言は「一般選言」といい、ラベルの下に現れる選言を「値としての選言」という。式6の選言は一般選言であり、式9の選言は値としての選言である。値としての選言は一般選言に含まれる。

$$\left[\begin{array}{ll} cat & s \\ \left\{\begin{array}{l} \left[\begin{array}{ll} voice & active \\ actor & Subj \end{array}\right] \\ \left[\begin{array}{ll} voice & passive \\ goal & Subj \\ actor & Actor \\ adjunct & \left[\begin{array}{ll} cat & pp \\ prep & by \\ obj & Actor \end{array}\right] \end{array}\right] \end{array}\right\} \end{array}\right] \tag{6}$$

Kasperによると、一般選言はKasper標準系とよばれる形に変換するができる[Kasper87]。

$$\begin{array}{l} (uconj_1 \wedge disj_{11} \ldots disj_{1i}) \\ \vee \ldots \\ \vee (uconj_n \wedge disj_{n1} \ldots disj_{nj}) \end{array} \tag{7}$$

選言によってタイプ付き素性構造を表現することはタイプ付き素性構造を部分的に共有することに相当するので、タイプ付き素性構造を簡潔に表現することができる。例えば、語彙youの持つ可能な解釈を展開された選言で表現すると式8のようになる。それは記述の面からみても処理の面からみても効率が悪い。しかし、それを値としての選言によってまとめて表現すると、表現が式9のように簡潔になる。

$$you \rightarrow \left\{\begin{array}{l} pronoun\left[\begin{array}{ll} agr & \left[\begin{array}{ll} person & 2nd \\ number & singular \\ case & subj \end{array}\right] \end{array}\right] \\ pronoun\left[\begin{array}{ll} agr & \left[\begin{array}{ll} person & 2nd \\ number & plural \\ case & subj \end{array}\right] \end{array}\right] \\ pronoun\left[\begin{array}{ll} agr & \left[\begin{array}{ll} person & 2nd \\ number & singular \\ case & obj \end{array}\right] \end{array}\right] \\ pronoun\left[\begin{array}{ll} agr & \left[\begin{array}{ll} person & 2nd \\ number & plural \\ case & obj \end{array}\right] \end{array}\right] \end{array}\right\} \tag{8}$$

$$you \rightarrow pronoun\left[\begin{array}{ll} agr & \left[\begin{array}{ll} person & 2nd \\ number & \{singular, plural\} \\ case & \{subj, obj\} \end{array}\right] \end{array}\right] \tag{9}$$

選言によってタイプ付き素性構造を表現する時、表現が簡潔になるため、単一化の効率が向上する一面がある。しかし、その単一化の後、選言の無矛盾性のチェックを行わなければならないので、効率が逆に悪くなる。Kasperの標準形による一般選言の単一化の後では選言の数がnであれば、無矛盾性チェックの計算量は$O(2^n)$にもなる[Kasper87]。その問題を解決するためにいくつかの提案がなされてきた[Carter90][永田90]。しかし、根本的な改善が難しい。

我々は、

選言を値としての選言に限定する。

そうすれば、単一化の後の無矛盾性のチェックの計算量がかなり減る。この制限によって、式6のような表現ができなくなった。選言による文法記述のメリットが幾分減ることになる。処理速度の向上を考えればやむ得ないことである。

3.2 一致

我々は選言を値としての選言に限定した。しかし、値としての選言内部に一致が存在する時、素性論理式の単一化が実現しにくい[Kasper87]。一方、文法記述の立場からみれば選言が曖昧なタイプ付き素性構造を表現しているため、選言の内部にまたがる一致の記述の需要がそれほど大きくないことも言える。

従って、我々は以下の制限を素性論理式に加える。

選言と否定の内部には一致がない。但し、選言と否定そのものの一致は許される。

例えば、以下の素性論理式は許される。

$$you \rightarrow pronoun\left[\begin{array}{ll} agr & \left[\begin{array}{lll} person & 2nd & \\ number & \{singular, plural\} & X \\ case & \{subj, obj\} & \end{array}\right] \\ sem & you\left[\begin{array}{ll} number & X \end{array}\right] \end{array}\right] \tag{10}$$

素性論理式によって循環するタイプ付き素性構造を表現するニーズがよくある。例えば、「私の買った本」の意味表現が以下のようになる。それには循環構造が存在する。

$$本\left[\begin{array}{ll} rel & 買う\left[\begin{array}{lll} agt & 私 & \\ obj & & X \end{array}\right] \end{array}\right] X \tag{11}$$

我々の文法記述システムでは、循環構造を表現する素性論理式が許される。

3.3 値の否定

否定には値の否定、素性の否定、一致の否定が考えられる。一致の否定はそれほど必要性がないと思われるので、本研究では取り扱っていない。ここでは、値の否定の記述機能に対する制限について述べる。素性の否定の実現し方に関しては後に述べる。

値の否定によってタイプつき素性構造を簡潔に表現することができる。例えば、英語の「第3人称単数以外」ということを否定によって表現することができる。

$$swim \rightarrow \quad verb\left[\begin{array}{ll} agr & \neg\left[\begin{array}{ll} person & 3rd \\ number & singular \end{array}\right] \end{array}\right] \tag{12}$$

本研究では、図3のように否定を推論する。$\neg A \wedge B$ がケース1において真で、ケース2において偽である。それ以外のケース3において推論が失敗するとする。本研究では、このように推論される否定を排他的な否定という。

値の否定としては、排他的な否定だけが許される。

値の否定を排他的な否定に限定するのは、単一化の効率を上げるためである。また、

否定の内部に更に否定が現れることがないとする。

つまり、以下のような素性論理式が定義できない。

$$\neg\left[\begin{array}{ll} l & \neg a \end{array}\right] \tag{13}$$

このような制限をするのは、否定に関する推論規則 $l14$、$l15$ が成り立たないからである。

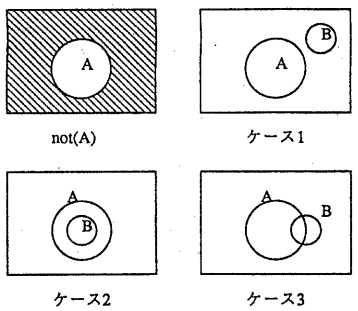

図 3: 排他的な否定

以上のような否定でも、我々の経験によれば、自然言語処理の知識を記述するのに十分強力である。例えば、式12のように *swim* を記述すれば、式14の *he* と式15の *they* との単一化がそれぞれ失敗と成功し、期待とおりの結果が得られる。

$$he \rightarrow \quad pronoun\left[\begin{array}{ll} agr & \left[\begin{array}{ll} person & 3rd \\ number & singular \end{array}\right] \end{array}\right] \tag{14}$$

$$they \rightarrow \quad pronoun\left[\begin{array}{ll} agr & \left[\begin{array}{ll} person & 3rd \\ number & plural \end{array}\right] \end{array}\right] \tag{15}$$

選言の否定を定義することもできる。例えば、日本語の格助詞「は」の語彙項目は以下である。

$$\text{は} \rightarrow \quad postp\left[\begin{array}{ll} pform & ha \\ subcat & \left[pp\left[\begin{array}{ll} pform & \neg\{ga, wo\} \end{array}\right]\right] \end{array}\right] \tag{16}$$

3.4 素性の否定

素性論理式が部分情報記述で、その言及していないことが「未知」になっている。そのため、時には成功して欲しくない全く違うことを記述する素性論理式の単一化が成功してしまう。それを解決するためにある素性論理式がある素性をもっていないことを素性の否定によって記述することが提案された [Smolka88]。例えば、素性論理式がラベル l_2 を持たないことを以下のように表現する。

$$\left[\begin{array}{ll} l_1 & a \\ l_2 & \backslash \end{array}\right] \tag{17}$$

この素性論理式が l_2 をもっていないので、任意の l_2 をもっている素性論理式との単一化が失敗する。しかし、逆のことも考えられる。つまり、ある素性論理式がある素性を必ずもっていることを記述したい場合がある。その素性論理式はその素性をもっていない素性論理式との単一化が失敗する。そこで、我々は以下の提案をする。素性論理式の素性を必須素性と自由素性にわける。必須素性の単一化の場合、素性を一対一に対応し、単一化を行う。自由素性の単一化の場合、普通の素性と同じように単一化を行う。

例えば、リスト $[a, b, c]$ を素性論理式によって表現することができる [Emele90]。しかし厳密にいうと、ここでの *first* と *rest* 素性が必須素性である。つまり、さらに他の素性をもつことやそのいずれの素性が欠けることがリストとして正しくない。

$$cons\left[\begin{array}{ll} first & a \\ rest & cons\left[\begin{array}{ll} first & b \\ rest & cons\left[\begin{array}{ll} first & c \\ rest & nil \end{array}\right] \end{array}\right] \end{array}\right] \tag{18}$$

必須素性と自由素性の区別によって素性の否定より強力な機能を実現することができる。さらに、この拡張には今までの素性論理の形式化に修正を加える必要がない。また、必須素性の単一化を先に行なうことによって単一化失敗の早期発見にも貢献できる。

3.5 制限された素性論理

現在までに述べてきた素性論理の記述機能に対する制限についてまとめる。

1. 選言は値としての選言とする。
2. 値の否定は排他的な否定とする。否定の内部に更に否定を含むことがない。
3. t が選言、或は否定の素性論理の式で、p が t の内部の経路である時、p が任意の経路 q と $p \downarrow q$ になることがない。
4. $p \uparrow q$ が定義されない。
5. 素性には必須素性と自由素性がある。

素性論理式の単一化は以下の式の順で行なう。

$$\neg s \wedge \neg t = \begin{cases} \neg s & t \preceq s \\ \neg t & s \preceq t \\ error & \text{others} \end{cases} \quad (19)$$

$$\neg s \wedge t = \begin{cases} t & s \wedge t = \bot \\ \bot & t \preceq s \\ error & \text{others} \end{cases} \quad (20)$$

$$t \wedge (s_1 \vee s_2) = (t \wedge s_1) \vee (t \wedge s_2) \quad (21)$$

$$f : s \wedge f : t = f : (s \wedge t) \quad (22)$$

$$A \wedge B = C \text{ iff } glb(A, B) = C \quad (23)$$

$$p \downarrow q \quad (24)$$

単一化を行なった後に、選言の論理式に対して以下の推論規則を実行する。

$$\bot \vee s = s \quad (25)$$

$$s \vee t = s, \text{ iff } t \preceq s \quad (26)$$

以上の式は素性論理の推論規則によるもので、それに基づく単一化が健全であることが容易にわかる。単一化を行った結果、得られるのが依然として記述機能の制限された素性論理式であることもわかる。つまり、記述機能の制限された素性論理が閉じている。

我々のシステムでは最後に選言の展開を行なう。[Kasper86]によれば、選言の展開が指数オーダーである。展開は選言の数によるので、最終的に選言の数が十分少なければ、我々の文法記述システムが十分実用に答えられる。幸いなことに、単一化がタイプつき素性構造の具体化の過程であるので、単一化によって選言の数が最終的には相当減る。

また、文法記述システムでは、最初に現れる選言は全部以下式に従ってまとめられる。単一化の途中では、そのまとめをさらにしない。

$$(f : s) \vee (f : t) = f : (s \vee t) \quad (27)$$

文法記述システムでは、推論できないような素性論理式を定義してはいけない。そのことは文法作成者の責任である。

3.6 一般化

以上のような単一化の後で、さらに一般化を行えば、その一般化の健全性が保証されなくなる。具体的には、s と t が排他的である時 $\neg s$ と t の単一化の結果、t が得られる。その後にその結果と u とさらに一般化を行おうとすれば、一般化の結果が正しくない。というのは、$\neg s$ と t の単一化の際、後の一般化に必要な情報 $\neg s$ が欠落したからである。しかし、単一化の後に一般化を行わなければ当然問題が起きない。我々のシステムでは、一般化を行っていない。

3.7 単一化の計算量

命題 1 循環構造を含むタイプつき素性構造の単一化が P のクラスの問題である。

[李 92] で述べたように、循環構造を含む TFS の単一化を併合とタグチェックという二つの過程にわけることができる。併合とタグチェックは P のクラスの問題である。

命題 2 式 19から式 24までの単一化は P のクラスの問題である。

証明:

1). 式 19の単一化について証明する。$s \preceq t$ であるかどうかをチェックするのは s と t が連言或いは選言だけを含み、その内部に一致がない時である。そのような条件では、そのチェックが P のクラスの問題である。従って、式 19が P のクラスの問題である。

2). 式 19と同様に、式 20も P のクラスの問題である。

3). それ以外の式については明らかである。 Q.E.D

4 文法記述システムの実現

この節では、記述機能の制限された素性論理に基づいた文法記述システムの実現の仕方について述べる。

4.1 システムの特徴

我々の文法記述システムでは、文法を拡張された CFG 規則によって記述する。拡張された CFG 規則において文法カテゴリを素性論理式によって記述する。素性論理式の中に選言と否定を定義することができる。また、拡張された CFG 規則の右辺に補強項関数を定義することができる。例えば、拡張された文法規則は次のように定義する。

```
sentence[agr Agr, sem Sem1] ->
noun_phrase[agr Agr, sem Sem1],
verb_phrase[mood or(ind,sub), agr Agr, sem Sem2],
{add(Sem1,agt,Sem2)}
```

シンタクテックには本文法記述システムの文法規則が DCG 規則と類似している。しかし、本システムは基本的な表現形式が述語でなく、素性論理式であるため、記述力が DCG より高い。

文法作成者が文法を定義し、文法規則を解釈するインタープリタが自然言語の解析を行なう。自然言語に対して、インタープリタがボトムアップ横型の解析を行なう。

4.2 データ構造

計算機上では、我々はタグリストというデータ構造によって素性論理式を表現している。タグリストとはタグをキーに素性論理式を表現するデータ構造である。タグリストのデータ構造は図 4のとおりである。タグリストで文法規則を表現すると、図 5-1 のようになる。

タグリストによる素性論理式の単一化アルゴリズムを [李 92] で示した。そのアルゴリズムによって、循環構造を含むタイプつき素性構造の単一化が可能である。

一方、タグを明示的にデータ構造の中で記述せず、ポインタによって一致とされる値を管理する素性論理式のデータ構造が一般的である (図 5-2)。我々のデータ構造には、単一化際のタグ検索のコストが高いという欠点がある。我々は今後データ構造を見なす予定である。

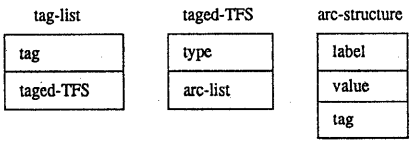

図 4: タグリストのデータ構造

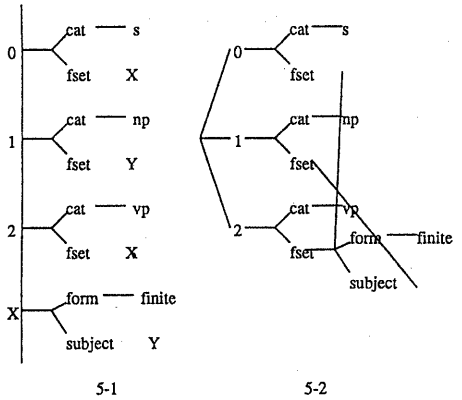

図 5: データ構造の比較

4.3 シングルトンの束

理論的に素性論理式の中のアトムがすべてタイプの束の上に定義される。しかし、実際我々はシングルトンの束だけを定義する。しかも、フラットな部分のシングルトンの束を省略する。そうすることによってメモリの節約ができる。文法記述システムは、定義されたシングルトンの束だけを参照して以上で示した素性論理式の単一化を行う。

4.4 補強項

補強項関数はDCGにならって導入されたものである。文法規則では、素性論理式によって文法カテゴリを表現する。文法カテゴリがタイプつき素性構造であることが仮定されている。補強項関数はタイプつき素性構造に対する操作を定義するものである。補強項関数によって文法記述システムの記述力がさらに強力になる。

補強項関数の実行が失敗したら、その文法規則の適用も失敗する。補強項関数は破壊的である。つまり、あるタイプつき素性構造に対して補強項関数を実行したら、そのタイプつき素性構造の復元ができなくなる。

文法作成者がプログラミング言語で補強項関数を定義することもできる。本文法記述システムでは *add*、*remv*、*bind*、*typep*、*labelp*、*retype* という補強項関数を提供する。*add* はTFSに素性を追加する関数である。*rem* はTFSから素性を削除する関数である。*bind* はあるタグにTFSを束縛する関数である。*labelp* はあるTFSにあるラベルがあるかどうかをチェックする関数である。*typep* はあるTFSのタイプがあるタイプであるかどうかをチェックする関数である。*retype* はTFSのタイプを書き換える関数である。

我々の経験によれば、このわずかの補強項関数でも応用上十分である。例えば、以下の順に補強項関数を適用することによって図6のようにTFSを変換することができる。

$$\begin{array}{ll} step1) & retype(X, b) \\ step2) & remv(X, l2) \\ step3) & add(X, l3, d) \\ step4) & bind(Y, X) \end{array} \tag{28}$$

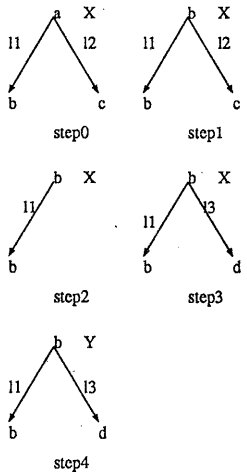

図 6: 補強項関数の適用例

5 文法記述システムの利用

本文法記述システムが強力な文法記述機能かつ高処理速度をもつので、我々はそれを自然言語処理の研究に活用している。本節では、幾つかの文法記述の仕方について紹介する。

5.1 束の利用

束を利用することによって、知識のモジュラ性を高めることができる。例えば、名詞に係る前置詞句と動詞に係る前置詞句をそれぞれ次のように定義するとする。

$$np \rightarrow n, pp \left[\begin{array}{ll} attach & n \end{array} \right] \tag{29}$$

$$vp \rightarrow v, pp \left[\begin{array}{ll} attach & v \end{array} \right] \tag{30}$$

一方、束で $n \preceq nv$ と $v \preceq nv$ という関係を定義する。そうすると、各前置詞がどのような句に係るかということを素直に表現することができる。例えば、

$$with \rightarrow p \left[\begin{array}{ll} attach & nv \end{array} \right] \tag{31}$$

$$in\_spite\_of \rightarrow p \left[\begin{array}{ll} attach & v \end{array} \right] \tag{32}$$

さらに、束と素性論理を利用することによって、増進的に意味の曖昧性を解消することもできる [奥村 89]。具体的には、意味素性間の関係を束で定義し、選択制限を意味素性間の単一化によって実現する。

5.2 補強項関数の利用

補強項関数を利用して、単一化で実現しにくいこと、或は単一化では効率の悪いことを実現することができる。例えば、解析の時、句構造と対応する意味構造を作り上げることを、単一化に頼らずに補強項関数によって実現することができる。

式34と式35が式33右辺の二つの文法カテゴリとそれぞれ単一化成功する。次に、補強項関数 add の適用によって式36が得られる。

$$vp\left[\begin{array}{cc} sem & X \end{array}\right] \rightarrow verb\left[\begin{array}{cc} sem & X \end{array}\right], pp\left[\begin{array}{cc} case & Y \\ sem & Z \end{array}\right] \quad add(X,Y,Z) \tag{33}$$

$$verb\left[\begin{array}{cc} sem & run \end{array}\right] \tag{34}$$

$$pp\left[\begin{array}{cc} case & loc \\ sem & park \end{array}\right] \tag{35}$$

$$vp\left[\begin{array}{ccc} sem & run & \left[\begin{array}{cc} loc & park \end{array}\right] \end{array}\right] \tag{36}$$

ここでは、補強項関数 add によって値をラベルにしている。これは単一化では実現しにくい操作である。

6 おわりに

本稿では記述機能の制限された素性論理に基づいた強力でかつ高速な文法記述システムについて述べた。素性論理に基づいた文法記述システムにはモジュラー性が高く拡張性が高い等のメリットがある。しかし、素性論理の無矛盾性チェックがNP完全であるため、素性論理をそのまま用いた文法記述システムはその処理の計算量が大きくなってしまう。そこで、我々は、自然言語処理のニーズをできるだけ損なわないように素性論理の記述機能に幾つかの制限を加え、記述機能の制限された素性論理に基づいた強力でかつ高速な文法記述システムを開発した。

我々の文法記述システムは、主に以下の点で従来の研究の発展になる。

- 値の否定の取扱いとして、排他的な否定を導入した。
- 必須素性と自由素性を導入した。
- 補強項関数を導入した。

これらは文法記述システムの記述力の強化と高速化の実現に寄与する結果となった。

謝辞

本研究の機会を与えてくださいました日本電気(株)C&C情報研究所の亘理誠夫部長、村木一至課長、及び日本電気(株)関西C&C研究所の市山俊治主任に深く感謝致します。また、研究を進めるにあたって、日本電気C&C情報研究所の山端潔氏からさまざまな助言をいただきました。ここで深く感謝いたします。

参考文献

[Aït-Kaci86] H. Aït-Kaci, *An Algebraic Semantics Approach to the Effective Resolution of Type Equations, Theoretical Computer Science, North-Holland, 1986.*

[Birkhoff40] G. Birkhoff, *Lattice Theory, American Mathematical Society, Providence 1940, Third revised edition, 1979.*

[Carter90] D. Carter, *Efficient Disjunctive Unification for Bottom-Up Parsing, COLING90, 1990.*

[Dawar90] A. Dawar, K.Vijay-shanker, *An Interpretation of Negation in Feature Structure Descriptions Computational Linguistics Vol 16, 1990.*

[Emele90] Martin C. Emele, Rémi Zajac, *Typed Unification Grammars, COLING90, 1990.*

[Johnson88] M.Johnson, *Attribute-value Logic and the Theory of Grammar, CSLI Lectures Notes No16, 1988.*

[Johonson91] M.Johnson, *Attribute and Formula, Computational Linguistics, 1991.*

[Kasper86] R.Kasper,W. Rounds, *A Logical Semantics for Features Structures, ACL86,1986.*

[Kasper87] R.Kasper, *A Unification Method for Disjunctive Feature Descriptions, ACL87, 1987.*

[小暮91] 小暮潔, 型付き素性構造の実装手法, 電子情報通信学会 *NLC*研究会, *1991.*

[李91] 李航, 束によって解釈する素性表現の論理, 電子情報通信学会 *NLC*研究会,*1991.*

[李92] 李航, 落合尚良, 閉路を含む素性構造の単一化, 情報処理学会 *44* 回全国大会,*1992.*

[Moshier87] M.Moshier, W. Rounds, *A Logic for Partially Specified Data Structures. Proc. of the 14th ACM symposium on Principles of Programming Languages, Munich, 1987.*

[永田90] 永田昌明,小暮潔, 文法規則の構造共有による選言的素性構造単一化の効率化, 情報処理学会 *40* 回全国大会,*1990.*

[奥村89] 奥村学, 田中穂積, 自然言語解析における意味的曖昧性を増進的に解消する計算モデル, 人工知能学会誌, *Vol.4, No. 6, 1989.*

[Smolka88] G. Smolka, *A Feature Logic with Subsorts, LILOG Report33 IBM Deutschland, Stuttgart, West Germany, 1988.*