解　説

素性構造の単一化†

今　村　　誠††

1. はじめに

素性構造 (feature structure) とは，自然言語のためのいくつかの文法理論[8),9),15),19)]において，名詞句や動詞句などの文の構成要素がもつ情報を表現するために用いられるデータ構造である．そして，素性構造の**単一化** (unification) は，文の構成要素同士を集めてさらに大きな構成要素を作る際に用いられる主な演算である[10),31)]．

当初，素性構造はいろいろなノーテーションで表記され，また直感的に用いられてきたが，ここ5，6年の間に，素性構造の形式化を行う研究がいくつかなされ，素性構造の意味や単一化アルゴリズムなどが次第に整理されるようになった．

また，素性構造は，従来，知識表現の分野で用いられてきたフレーム[36)]やプログラミング言語の分野で用いられてきたデータ型であるレコード[3)]などと類似の構造でもある．そのため，素性構造の定式化に関する研究は，これら類似のデータ構造の表現能力やアルゴリズムを比較検討する土台の一つを与えることになり，現在，知識表現言語の研究と関連をもつようになっている．

本稿では，自然言語の文法理論のために用いられてきた素性構造が一階論理の部分言語を用いて定式化できることについて，以下の順序で説明する．**2.**では，素性構造の単一化が自然言語の文法規則の記述と利用の際に，どのように用いられているかについて述べる．**3.**では，素性構造の一階論理を用いて形式化できることについて，**4.**では素性構造の単一化アルゴリズムについて説明する．**5.**では，素性構造の記述能力向上に関する最近の話題について述べている．

† Unification of Feature Structures by IMAMURA Makoto (COMPUTER AND INFORMATION SYSTEMS Lab., MITSUBISHI ELECTRIC Corp.).

†† 三菱電機(株)情報電子研究所

2. 自然言語の文法理論における素性構造の利用

単一化を主な演算としてもつ文法理論（単一化に基づく文法理論[31)]）では，言語に関する知識は，語彙項目と文法規則によって記述される．語彙項目は単語を定義しており，文法規則は文の構成要素同士を結合してより大きな構成要素を構成する方法を定義している．その文法規則には文の構成要素がもつ素性構造同士の制約が記述されており，この制約によって文の構成要素同士の依存関係（たとえば，数，格，性，人称の呼応関係など）が表現される．また，このような語彙項目と文法規則を用いて文を解析する際には，文法規則を次々に適用することによって，文を構成する単語の語彙項目がもつ素性構造からその文がもつ素性構造を構成する操作がなされ，この操作の際に単一化が中心的な働きをしている．

この章では，論理を用いた文法記述の立場にたって，文法規則が素性構造同士の制約を用いて宣言的に記述され，また文法規則を操作的に利用することで文の解析がなされることについて説明する．

2.1 素性構造

文法記述への利用について説明する前に，まず素性構造について説明する．素性構造とは，素性-値対の集合からなる構造であり，通常，下にあるような行列形式で表記される[31)]．

$$\begin{bmatrix} number: & \text{singular} \\ person: & \text{third} \end{bmatrix} \qquad (1)$$

この例は，英語の3人称単数の名詞がもつ情報を表現しており，行列の左の列にあるものが**素性** (feature) であり，その右にあるものが対応する値である．この行列の1行目は，素性 *number*（数）の値が singular（単数）であることを示している．同様に，2行目は，*person*（人称）という素性の

値が third であること，すなわち，3人称であることを表現している．本稿では，素性はイタリックで，定数はローマンで示すことにする．

また，次の例にあるように，素性の値はアトミックなシンボルだけでなく，素性構造であってもよい．

$$\begin{bmatrix} tense: & \text{present} \\ subj: & \begin{bmatrix} num: & \text{sg} \\ person: & \text{3rd} \end{bmatrix} \end{bmatrix} \quad (2)$$

この素性構造は，素性 *subj* の値が(1)の素性構造であることを示している．ただし，この例では，素性 *number* は *num*，素性 *subject* は *subj* と略記している．また，以後にあげる例では，素性 *agreement* は *agr*，素性 *category* は *cat*，定数 singular は sg，定数 plural は pl，singular は sg と略す．

さらに，素性の値を共有することを示す相互参照 (coreference) を記述することができる．下の例にあるように，相互参照は素性の値を線で結ぶことによって示される．

$$\begin{bmatrix} cat: & \text{np} \\ agr: & \begin{bmatrix} num: & \text{sg} \\ person: & \text{3rd} \end{bmatrix} \\ subj: & \text{(agr の値を共有)} \end{bmatrix} \quad (3)$$

この素性構造は素性 *agr* と *subj* の値が共有されていることを示している．

素性構造の集合を考えると，素性構造同士がもつ情報の大小を比較するために用いられる半順序関係 (partial order)* である**包摂関係** (subsumption relation) と素性構造がもつ情報を結合する演算である単一化が定義される．以下，例をあげながら説明する．

二つの素性構造 d1, d2 に関して，両者のもつ情報内容が互いに矛盾せず，d2 がもつ情報内容が d1 のもつ情報内容よりも多いという関係が成り立つとき，d1 は d2 を**包摂する** (subsume) と呼び，d1⊑d2 と書く[31]．

例1 素性構造の包摂関係

$$[\ ] \quad (4)$$

$$\begin{bmatrix} cat: & \text{np} \end{bmatrix} \quad (5)$$

$$\begin{bmatrix} cat: & \text{np} \\ agr: & \begin{bmatrix} num: & \text{sg} \\ person: & \text{3rd} \end{bmatrix} \\ subj: & \begin{bmatrix} num: & \text{sg} \\ person: & \text{3rd} \end{bmatrix} \end{bmatrix} \quad (6)$$

上のような素性構造に関して，以下のような包摂関係が成立する．

(4)⊑(5)⊑(6)⊑(3) □

素性構造(6)と(3)は，一見同じ情報内容をもっているようにみえるが，相互参照を用いているほうが情報内容が多い．これは，(3)のほうは，*agr* か *subj* のどちらかの値になんらかの情報内容が付与された場合には，もう一方の素性の値にも同じ情報内容を付与しなければならないからである．

素性構造 d1 と d2 の単一化とは，d1⊑d3, d2⊑d3 をみたす最小の素性構造 d3 を求めることである．すなわち，d1 のもつ情報と d2 のもつ情報を矛盾のないかぎり結合した情報を求める演算である（矛盾がある場合は失敗する）．以下に例をあげる．

例2 素性構造の単一化

$$\begin{bmatrix} agr: & \begin{bmatrix} num: & \text{sg} \end{bmatrix} \end{bmatrix}$$

と

$$\begin{bmatrix} subj: & \begin{bmatrix} person: & \text{3rd} \end{bmatrix} \end{bmatrix}$$

を単一化すると

$$\begin{bmatrix} agr: & \begin{bmatrix} num: & \text{sg} \end{bmatrix} \\ subj: & \begin{bmatrix} person: & \text{3rd} \end{bmatrix} \end{bmatrix}$$

を得る。

$$\begin{bmatrix} agr: & \begin{bmatrix} num: & \text{sg} \end{bmatrix} \\ subj: & \text{(agr の値を共有)} \end{bmatrix} \quad (7)$$

と

$$\begin{bmatrix} subj: & \begin{bmatrix} person: & \text{3rd} \end{bmatrix} \end{bmatrix} \quad (8)$$

を単一化すると

$$\begin{bmatrix} agr: & \begin{bmatrix} num: & \text{sg} \\ person: & \text{3rd} \end{bmatrix} \\ subj: & \text{(agr の値を共有)} \end{bmatrix} \quad (9)$$

を得る．

2.2 文法規則の宣言的な記述

単一化に基づく文法理論における文法規則は，次の二つの部分からなっている．

(a)「文字列がどのように連結される (concatenate) か」を規定する部分．

(b)「文字列が連結される際に，連結される

* 反射律，反対称律，推移律を満たす関係を半順序関係という．素性の値に集合を許す場合には，反対称律が成立しないので半順序関係にならない[28]．

文字列のもつ素性構造は互いにどのような関係にあるか」を規定する部分.

文法によって受理される文は，規則を用いて構成することができる文字列として特徴づけられる．また，規則の集合は，文とその文に割り当てられる素性構造の関係を宣言的に記述しているとみなすことができる．例として，主語と動詞間の数と人称の一致の規則を記述した例をあげる．文法規則に記述される内容や形式はおのおのの文法理論ごとに異なっているが[8),9),15),19)]，ここでは文献34)を参考にした単純な文法例を紹介する．「連結に関する規則」は文脈自由規則を用いて記述され，「連結される文字列に割り当てられる素性構造同士の制約」は素性に関する制約（素性制約）を用いて { } の中に記述される．文法規則中の f(S) は，Sのもつ素性構造の素性 f の値を示し，記号 "≒" は両辺がさす素性構造を単一化することを示している．また，制約中には連言を表す記号 "∧" や否定を表す記号 "¬" や選言を表す記号 "∨" を用いることができ，通常の論理式の場合と同様の意味をもっている．

[文法規則]

S ⟶ NP VP
{ S≒VP ∧ *subj*(VP)≒NP }.

[語彙項目]

NP ⟶ jack
{ *person*(NP)≒3rd ∧ *num*(NP)≒sg }.

VP ⟶ sleeps
{ *tense*(V)≒present
∧ *person* (*subj*(VP))≒3rd
∧ *num*(*subj*(VP))≒sg }.

VP ⟶ sleep
{ *tense*(V)≒present
∧ ((*person*(*subj*(VP))≒ -3rd
∧ *num*(*subj*(V))≒(sg∨pl))
∨(*person*(*subj*(VP))≒ 3rd
∧ *num*(*subj*(V))≒ pl)) }.

以下，これらの規則について簡単に説明する．

はじめの規則は，名詞句 NP と動詞句 VP から文Sが構成されることを示している．制約 S≒VP はSの素性構造が VP の素性構造と同じであることを示し，制約 *subj* (VP)≒NP☆ は，VP の素性構造の素性 *subj* の値は，素性構造 NP と同じであることを示している．

2番目の規則は，"jack" が名詞句 NP となることを示している．素性制約ははじめの規則と同様に読むことができ，NP の属性構造が3人称単数であるという情報を表している．

3番目の規則は，"sleeps" が動詞句 VP となることを示しており，1番目の規則の制約と合わせることで，「主語となる名詞句が3人称単数である」という，数と人称に関する一致 (agreement) の制約を表している．この制約中の *person* (*subj* (VP)) は，VP の素性構造の素性 *subj* の値である素性構造の素性 *person* の値をさす表現である．

4番目の規則は，"sleep" が動詞句 VP となることを示している．この制約は，主語の人称が3人称であれば，主語の数は複数でなければならないこと，また，主語の人称が3人称以外であれば，主語の数は単数でも複数でもよいことを表現している．

このような文法規則と語彙項目の集合が与えられると，これらの規則を用いて構成することができる文字列の集合が決まり，このような文字列は文法的に正しい文字列と呼ばれる．

2.3 文法規則の操作的な利用

2.2 のように記述された文法規則を用いて構成できる文字列を受理するパーザを，論理型言語の推論を用いて実現することができる．DCG で書かれた文法規則を Prolog のプログラムに変換できる[22),27)]のと同じように，**2.2** にあげた文法規則は，素性に関する制約解消機構を組み込んだ制約論理型言語[7),12)]のプログラムに変換することができるからである．この場合，変換されたプログラムは，「文法規則によって受理される文字列」と「その文字列のもつ素性構造」の関係を規定しているとみなせる．この性質を利用すると，制約論理型言語の推論機構を用いて，構文解析を行うことができる．そして，適用された文脈自由規則の履歴から句構造木を得ることができ，適用された規則から得られる素性制約の標準形を求めることにより句構造木のノードがもつ素性構造を求めることができる．素性制約の標準形を求める方法の詳細は，**4.** で説明することにし，ここでは **2.2** の文法例での文の解析例を示す．

2.2 にあげた文法例を用いて，"jack sleeps" という文を解析する場合，変換された制約論理型言

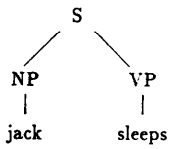
☆ 制約の表記は文法理論によっておのおの異なる．たとえば，文献15) の LFG では (VP *subj*)=NP と記述されるが，素性が素性構造から素性構造への関数としてモデル化されることを強調するために，*subj*(VP) という表記を用いた[14)]．

語のプログラムに対して，「"jack sleeps" が文かどうか」という質問を行うと，文法規則中の "S ⟶ NP VP", "NP⟶ jack", "VP⟶ sleeps" に対応する部分が呼ばれ，呼ばれた規則の制約を集めることにより制約(10)が得られる．

$$
\begin{aligned}
&(\ S \doteq \mathrm{VP} \\
&\wedge\ \mathit{subj}(\mathrm{VP}) \doteq \mathrm{NP} \\
&\wedge\ \mathit{person}(\mathrm{NP}) \doteq 3\mathrm{rd} \\
&\wedge\ \mathit{num}(\mathrm{NP}) \doteq \mathrm{sg} \\
&\wedge\ \mathit{tense}(\mathrm{VP}) \doteq \mathrm{present} \\
&\wedge\ \mathit{person}(\mathit{subj}(\mathrm{VP})) \doteq 3\mathrm{rd} \\
&\wedge\ \mathit{num}(\mathit{subj}(\mathrm{VP})) \doteq \mathrm{sg}\ )
\end{aligned}
\tag{10}
$$

そして，どの規則がどの規則を呼んだかという関係から句構造木が得られ，制約の標準形を求めることから句構造木のノードがもつ素性構造が得られる．以下，得られる句構造木とノードがもつ素性構造を示す．

[句構造木]



[対応する素性構造]

• NP のもつ素性構造

$$
\begin{bmatrix} \mathit{person}: & 3\mathrm{rd} \\ \mathit{num}: & \mathrm{sg} \end{bmatrix}
$$

• VP のもつ素性構造

$$
\begin{bmatrix} \mathit{tense}: & \mathrm{present} \\ \mathit{subj}: & \begin{bmatrix} \mathit{person}: & 3\mathrm{rd} \\ \mathit{num}: & \mathrm{sg} \end{bmatrix} \end{bmatrix}
$$

• S のもつ素性構造

$$
\begin{bmatrix} \mathit{tense}: & \mathrm{present} \\ \mathit{subj}: & \begin{bmatrix} \mathit{person}: & 3\mathrm{rd} \\ \mathit{num}: & \mathrm{sg} \end{bmatrix} \end{bmatrix} \tag{11}
$$

また，素性構造の単一化を用いた文法記述は，発話に関する構文的・意味的・語用論的な制約を統一的に扱うことができるという利点をもっている[10]．そのため，素性構造の単一化を用いたパーザが検討されるようになっている[11),20),35)]．

3. 素性構造の定式化

ここ5，6年の間に素性構造の形式化を行ういくつかの研究[1),13),16),23),24),28),32)]がなされた．おのおのの理論で用いられる数学的な道具立てが異なるので容易に比較することはできないが，素性に関する制約の表現能力の向上を目指して発展してきたといえる．中でも，Johnson と Smolka は，連言，選言，否定，相互参照を含んだ素性に関する制約が，等号付き一階論理の部分言語を用いて定式化できることを示しており，素性構造の意味論がよく知られている一階論理の概念を通じて理解されるようになった．

この章では，文献34)を参考にして，通常の一階論理[21)]との差異に注意しながら，素性構造の意味について説明する．素性構造は，素性構造が満たす素性制約によって特徴づけられるので，素性制約の構文と意味について順に述べる．

3.1 素性制約の構文

素性制約の構文は，一階言語の場合と同様に定義される．以下，互いに交わらないシンボルの集合L（素性の集合），V（変数の集合），C（定数の集合）を仮定する．まず素性制約における項を定義する．関数として1引数の部分関数のみを扱う点が特徴である．

定義1（項）

項は，次のように帰納的に定義される．

（a） 変数は項である．

（b） 定数は項である．

（c） f が素性，s が項であるならば，f(s) は項である． □

次に素性制約を定義する．通常の等号付き一階論理の式との違いは，「関数 f は項 s の上では定義されていない」ということを表現する制約 f(s)↑ が許される点である．

定義2（素性制約）

素性制約は，次のように帰納的に定義される．

（a） s, t が項ならば，s≒t は素性制約である．

（b） f が素性，s が項ならば，f(s)↑ は素性制約である．

（c） ϕ, ψ が素性制約ならば，$\phi \wedge \psi$, $\phi \vee \psi$, $\neg\phi$, $\phi \rightarrow \psi$ は素性制約である． □

（d） ϕ が素性制約，x が変数ならば，$\exists x.\ \phi$, $\forall x.\ \phi$ は素性制約である．

ここでは，一階言語の場合と同様に，与えられたシンボルの集合L，V，Cから構成されるすべての素性制約の集合を素性制約言語と呼ぶことにする．

3.2 素性制約の意味

続いて素性制約の意味について説明する．一階論理の場合と同様，素性制約はその制約のモデル

となる解釈によって特徴づけられる．素性は定数上では定義されないような1引数部分関数として形式化されるので，素性制約言語の解釈はそれを反映した特殊なものになっている．

定義3（素性制約言語の解釈）
素性制約言語の解釈Aは，下のような条件を満たす「空でない集合 D^A」と「素性と定数への割り当てを与える関数 $\bullet^A$」からなる．D^A は，解釈の領域と呼ばれる．

（a） 素性fへの割り当て f^A は，D^A から D^A への1引数部分関数である．

（b） 定数aへの割り当て a^A は，D^A の要素である．

（c） aとbが異なる定数ならば，$a^A \neq b^A$ である．

（d） aが定数，fが素性ならば，f^A は a^A 上では定義されない． □

一階論理の場合と同様に，解釈に関する変数の割り当てを利用して，素性制約の解や充足可能性を定義することができる．

定義4（変数割り当て）
Aを素性制約言語の解釈とする．（Aに関する）変数割り当てとは，素性制約言語の変数の集合VからAの領域 D^A への写像である．このような割り当ての集合を ASS[A] と表記する． □

定義5（項割り当て）
Aを素性制約言語の解釈，α を解釈Aに関する変数割り当てとする．素性制約言語中の項への（Aと α に関する）項割り当てとは，次のように定義される．

（a） 各変数には，α に従って項割り当てが行われる．変数xへの項割り当てを，$x_\alpha{}^A$ と表記する．

（b） 各定数には，$\bullet^A$ に従って割り当てが行われる．定数aへの項割り当てを，$a_\alpha{}^A$ と表記する．

（c） $s_\alpha{}^A$ をsへの項割り当てとし，f^A をfへの割り当てとするとき，$f^A(s_\alpha{}^A)$ は，項f(s)への項割り当てである．ただし，fは部分関数なので，$f^A(s_\alpha{}^A)$ は必ずしも定義されるとは限らない． □

以上のような準備のもとで，素性制約の解を定義することができる．素性制約の解は，素性制約が正しくなるような変数の割り当ての集合によって定義される．

定義6（素性制約の解）
素性制約の解は，以下のように定義される．ただし，s, t は項，ϕ, ψ は素性制約，x は変数であるとする．

（a） $(s \doteq t)^A \overset{\text{def}}{=} \{\alpha \in \mathrm{ASS[A]} \mid s_\alpha{}^A = t_\alpha{}^A\}$
（$s_\alpha{}^A$ や $t_\alpha{}^A$ が定義されないときには，$s_\alpha{}^A = t_\alpha{}^A$ は成立しないとみなす）

（b） $(f(s)\uparrow)^A \overset{\text{def}}{=} \{\alpha \in \mathrm{ASS[A]} \mid \forall d \in D^A f^A(s_\alpha{}^A) \neq d\}$

（c） $(\phi \wedge \psi)^A \overset{\text{def}}{=} \phi^A \cap \psi^A$

（d） $(\neg\phi)^A \overset{\text{def}}{=} \mathrm{ASS[A]} - \phi^A$

（e） $(\phi \vee \psi)^A \overset{\text{def}}{=} \phi^A \cup \psi^A$

（f） $(\exists x(\phi))^A \overset{\text{def}}{=} \{\alpha \in \mathrm{ASS[A]} \mid \exists d \in D^A \alpha(x/d) \in \phi^A\}$
（$\alpha(x/d)$ は，x に d が割り当てられている以外は，α と同じ変数割り当てである）

（g） $(\forall x(\phi))^A \overset{\text{def}}{=} \{\alpha \in \mathrm{ASS[A]} \mid \forall d \in D^A \alpha(x/d) \in \phi^A\}$ □

一階論理の場合と同様に，素性制約 ϕ が解をもつような解釈が少なくとも一つ存在するときに，ϕ は充足可能であるという．

4. 素性構造の単一化アルゴリズム

素性構造の単一化は，素性構造を特徴づける素性制約の充足可能性の判定問題に帰着される．この章では，素性制約の充足可能性の判定アルゴリズムについて説明し，素性構造の単一化への利用例を示す．

素性制約の意味を一階論理の部分言語として定式化することができたため，一階論理における論理式の標準形への変換の方法を利用することができる．前章で定義した素性制約の充足可能性の判定問題は一般的には決定可能でないが，変数として限量子で束縛されていない変数（自由変数）のみをもつ素性制約の場合には，充足可能性を判定する問題が NP-完全であることが知られている[13),34)]．このクラスの素性制約の充足可能性を判定手続きを，素性制約を標準形へ変換するアルゴリズムによって与えることができ，その手続きを素性構造の単一化に利用することができる．

変数として自由変数のみをもつ素性制約の充足可能性の判定手続きは次の4つのステップからな

る．ステップ1，2は一階論理の場合と同様のもので，ステップ3，4は素性制約特有のものであり，素性制約の解釈の与え方の特殊性を反映している部分である．

①ステップ1

下にあげる4つの制約をプリミティブ制約と呼ぶ．

$f(s)\doteq t$, $f(s)\uparrow$, $s\doteq t$, $s\neq t$

(f は素性，s, t は変数または定数を示す)

ステップ1では**図-1** と**図-2** の変換規則を用いて，素性制約をプリミティブ制約の連言と選言のみを用いて構成される制約へ変換する．図-1 の変換規則によって，含意記号と否定記号が除去される．図-2 の変換規則によって，$f1(f2(s))\doteq t$ のように素性の連続した適用を含む制約や，$f1(s)\doteq f2(t)$ のように制約の両辺に素性の適用を含む制約が除去される．図中の変換規則は，〜の左側の制約を右側の制約へと変換することを意味し，ϕ, ψ は素性制約，p, q は素性の連続した適用，f は素性，x, y は変数，a は定数，s, t は変数または定数を示している．

②ステップ2

ステップ2では，命題論理の場合と同様の方法を用いて，ステップ1で得られた制約の選言標準形を求める．プリミティブな制約の連言のみを用いて構成される制約は，一階論理でリテラルの連言からなる論理式を節と呼ぶのにならって，素性節 (feature clause) と呼ばれる．

③ステップ3

ステップ3では，素性節を標準形へ変換する．**図-3** にあげる変換規則が適用されなくなるまで

$\phi\rightarrow\psi \sim \neg\phi\vee\psi$
$\neg(\phi\wedge\psi) \sim \neg\phi\vee\neg\psi$
$\neg(\phi\vee\psi) \sim \neg\phi\wedge\neg\psi$
$\neg\neg\phi \sim \phi$
$\neg(p(s)\doteq q(t)) \sim p(s)\uparrow\vee q(t)\uparrow \vee (p(s)\doteq x \wedge q(t)\doteq y \wedge x\neq y)$
(但し，x と y は s，t とは異なる変数である。)
$\neg(p(s)\uparrow) \sim p(s)\doteq x$　(但し，x と s は異なる変数である。)

図-1 プリミティブ制約への変換規則1

$p(s)\doteq q(t) \sim (p(s)\doteq x \wedge q(t)\doteq x)$
(但し，x は s とも t とも異なる変数である。)
$f(p(s))\doteq t \sim (p(s)\doteq x \wedge f(x)\doteq t)$
(但し，x は s とも t とも異なる変数である。)
$p(f(s))\uparrow \sim f(s)\uparrow \vee (f(s)\doteq x \wedge p(x)\uparrow)$
(但し，x は s と異なる変数である。)

図-2 プリミティブ制約への変換規則2

$x\doteq s \ \&\ C \sim x\doteq s \ \&\ [x/s]C$
(但し，x は C に出現し，$x\neq s$ である。)
$a\doteq x \ \&\ C \sim x\doteq a \ \&\ C$
$f(x)\doteq s \ \&\ f(x)\doteq t \ \&\ C \sim f(x)\doteq s \ \&\ s\doteq t \ \&\ C$
$s\doteq s \ \&\ C \sim C$
$f(a)\uparrow \ \&\ C \sim C$
$a\neq x \ \&\ C \sim x\neq a \ \&\ C$
$a\neq b \ \&\ C \sim C$　(但し，$a\neq b$ である。)

図-3 素性節の標準形への変換規則

変換を行う．図中の [x/s]C は，変数 x の出現を s で置き換えることでCから得られる素性節を示している．

④ステップ4

ステップ4では，ステップ3によって得られた制約が充足可能であるかどうかのチェックを行う．与えられた制約が充足可能である必要十分条件は，選言記号で結ばれたおのおのの素性節が下の条件をすべて満たすことである．

- a が定数ならば，$f(a)\neq s$ を含まない．
- $s\neq s$ を含まない．
- a と b が異なる定数ならば，$a\doteq b$ を含まない．
- $f(x)\uparrow$ と $f(x)\doteq s$ を同時に含まない．

以下に，素性制約の充足可能性の判定例をあげる．

例3

素性制約 ($person(X)\doteq 3rd\rightarrow Y\doteq pl)\wedge person(X)\doteq 3rd\wedge Y\doteq sg$) が充足可能でないことを示す．上にあげたアルゴリズムに従って，以下のような変換を行えばよい．

$(\ person(X)\doteq 3rd \rightarrow Y\doteq pl)$
$\wedge\ person(X)\doteq 3rd \ \wedge Y\doteq sg$
⇓ステップ1
$(person(X)\uparrow$
$\vee\ (person(X)\doteq X1 \ \wedge X1\neq 3rd)$
$\vee\ Y\doteq pl)$
$\wedge\ person(X)\doteq X2 \wedge X2\doteq 3rd \wedge Y\doteq sg$
⇓ステップ2, 3
$(\underline{person(X)\uparrow} \wedge \underline{person(X)\doteq X2} \ \wedge X2\doteq 3rd$
$\wedge\ Y\doteq sg)$
$\vee\ (person(X)\doteq X1 \ \wedge \underline{3rd\neq 3rd} \ \wedge Y\doteq sg)$
$\vee\ (\underline{pl\doteq sg} \ \wedge person(X)\doteq X2 \ \wedge X2\doteq 3rd)$
⇓ステップ4

おのおのの素性節の下線の部分がステップ4の条件を満たしていないため，全体の制約が充足可能でないことが示された．□

次に，素性制約の充足可能性の判定アルゴリズムを利用した素性構造の単一化の例をあげる．

例4

2.の単一化の例にある素性構造(7)と素性構造(8)の単一化を行う.

素性構造(7)に対応する素性制約は,

$agr(X) \doteq subj(X) \wedge num(agr(X)) \doteq sg$ であり,

素性構造(8)に対応する素性制約は,

$person(subj(Y)) \doteq 3rd$ である.

単一化を行うには,両者の制約に $X \doteq Y$ を加えた制約の標準形を求めればよい.すなわち,

$$agr(X) \doteq subj(X) \wedge num(agr(X)) \doteq sg \wedge person(subj(Y)) \doteq 3rd \wedge X \doteq Y$$
$$\Downarrow$$
$$agr(X) \doteq Z \wedge subj(X) \doteq Z \wedge num(Z) \doteq sg \wedge person(Z) \doteq 3rd$$

この制約は素性構造(9)に対応している. □

実は,上にあげたアルゴリズムは必ずしも効率がよくない.それは,命題論理での変換アルゴリズムと同様,ステップ2における選言標準形への変換が指数オーダであるためである.選言は,自然言語処理では曖昧性(ambiguity)を表現するために用いられるので,実用のためにはアルゴリズムの高速化が必要になる.素性制約の充足可能性の判定はNP完全であるので,本質的なアルゴリズムの改善は難しいが,おのおのの制約の変換が等価であることを利用して,変換規則の適用の↗↘順序を適切に選択したり,変換規則の適用を遅延させたりすることで計算の効率をあげることができる.Kasper[17),18)] や Eisele[6)] らは選言標準形へ展開するのを遅延させることによって平均的に計算の効率をあげる方法を示している.

5. 素性制約の拡張

自然言語の文法を記述する際に,素性構造に関する制約は **3.** であげたもの以外のものが用いられる場合がある.ここでは,集合値をとる制約と包摂関係制約を含んだ素性制約について例をあげながら述べる.

5.1 集合値をとる制約の導入

自然言語の分野においては,"jack bought and ate apples" のような文を扱う場合に,文法規則中で素性構造の集合を扱う制約が用いられる.集合値をとる制約を用いることによって,動詞 "bought" と動詞 "ate" が主語と目的語を共有していることを簡潔に表現することができる.以下に文法規則例と文に対応する素性構造をあげる.

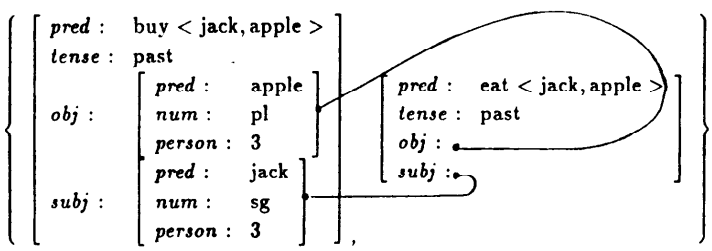
[文法規則の一部]

S→NP VP {subj(S)≒NP ∧S≒VP}

VP→V NP {VP≒V ∧obj(VP)≒NP}

V→V1 V2{V1∈V∧V2∈V}

[文のもつ素性構造]



また,集合値をとる素性制約の意味論も検討されはじめている[25),30)].

5.2 包摂関係制約の導入

集合値をとる制約と同様に,包摂関係制約を扱う必要が生じる場合がある.たとえば,英語の動詞の多くは,動詞の目的語や補語として等位接続詞で結ばれた句をとる場合には,等位接続詞で結ばれたもの同士は同じカテゴリ,すなわち共に名詞句であったり共に動詞句であったりする.しかし,"is" や "become" のような例外となる動詞もある.この現象について以下に例をあげて説明する.ただし,例文中の *NP*, *AP*, *PP* はおのおの名詞句,形容詞句,前置詞句を示している.

(1) Pat hired [$_{NP}$ a Republican] and [$_{NP}$ a banker].

(2) *Pat hired [$_{NP}$ a Republican] and [$_{AP}$ proud of it].

(3) Pat is [$_{AP}$ healthy] and [$_{PP}$ of sound mind].

例文(1)は正しい文であるが,例文(2)は,動詞 "hired" は名詞句と形容詞句を等位接続詞で結んだものを目的語としてとることができないので非

文となる．一方，“is” は，例文(3)のように名詞句と前置詞句を等位接続詞で結んだものを補語としてとることが許される．

そこで，“is” や “become” の補語にくるものをどのように特徴づければよいかという問題がおこる．この問題を名詞句と形容詞句が共通にもつなんらかの性質があるという考え方で説明する立場があり，このことを表現する際に包摂関係制約が用いられる[5]．

$C \rightarrow AP$ “and” PP $\{C \sqsubseteq AP \wedge C \sqsubseteq PP\}$.

また，包摂関係制約は知識表現の継承と関連づけられるため，知識表現言語において，素性制約の包摂関係が扱われる[37]．包摂関係制約をもつ素性制約の充足可能性の判定問題に関してもいくつか結果が知られている[5),25]．

6. おわりに

はじめに述べたように，素性構造の定式化は類似の構造を扱う分野と関連をもつようになっている．たとえば，Smolka らは，演繹データベースの分野で用いられる Aït-Kaci の ψ 項[1]との関連を扱っており[33]，Nebel らは，知識表現言語 KL-ONE[2] との関連について考察している[26]．素性-値対（属性-値対）からなるデータ構造は多くの分野で用いられており，いずれの場合にもなんらかの情報をこのデータ構造によって表現し，そのデータ構造を操作することによって情報処理を行っている．それぞれの方法でこのデータ構造の構文と意味が規定され，この構造のもつ基本的な演算が検討されているが，素性構造の定式化に関する研究は，これらのデータ構造が背景にもつ共通の数学的基盤を明らかにしようという試みとみなすことができる．素性構造の研究は，同種の構造を扱う分野との相互交流を通じて，知識表現の基盤の一つとなることが期待されている．

謝辞 草稿に対する貴重なコメントをいただきました査読者の方々，大阪大学の郡司隆男助教授，東京工業大学の徳永健伸博士，ICOT の向井国昭主任研究員，ICOT の安川秀樹第3研究室長代理に深謝いたします．

参考文献

1) Aït-Kaci, H.: An Algebraic Semantics Approach to the Effective Resolution of Type Equations, *Theor. Comput. Sci.*, Vol. 45, pp. 293-351 (1986)).
2) Brachman, R. and Schmolze, G.: An Overview of the KL-ONE Knowledge Representation System, Cognitive Science, Vol. 9, No. 2, pp. 171-216 (1985).
3) Cardelli, L.: A Semantics of Multiple Inheritance, *Inf. Control*, Vol. 176, No. 2/3, pp. 138-164 (1988).
4) Dawar, A. and Vijay-Shanker, K.: An Interpretation of Negation in Feature Structure Descriptions, Computational Linguistics, Vol. 16, No. 1, pp. 18-24 (1989).
5) Dörre, J. and Rounds, W. C.: On Subsumption and Semiunification in Feature Algebra, In Proc. 5th Annual Symposium on Logic in Computer Science, pp. 300-310 (1990).
6) Eisele, A. and Jochen, D.: Unification of Disjunctive Feature Descriptions, In Proc. 26th Meeting of the Association for Computational Linguistics, pp. 286-294 (1988).
7) 淵 一博監修，溝口文雄，古川康一，Lassez, J. L. 編：制約論理プログラミング，共立出版 (1989).
8) Gazdar, G., Klein, E., Pullum, G. K. and Sag, I. A.: Generalized Phrase Structure Grammar, Cambridge, Mass., Harvard University Press (1985).
9) Gunji, T.: Japanese Phrase Structure Grammar: A Unification-based Approach, Dordrecht, D. Reidel (1987).
10) 郡司隆男：言語構造と単一化，情報処理 (in this volume).
11) 橋田浩一，白井英俊：条件付き単一化，コンピュータソフトウェア，Vol. 3, No. 4, pp. 28-38 (1986).
12) Jaffar, J. and Lassez, J.: Constraint Logic Programming, In Proc. 14th ACM Symposium on Principles of Programming Languages, ACM, pp. 111-119 (1987).
13) Johnson, M.: A Logic of Attribute-Value Logic and the Theory of Grammar, CSLI Lecture Notes, No. 16 (1988).
14) Johnson, M.: Expressing Disjunctive and Negative Feature Constraints with Classical First-Order Logic, In Proc. 28th Meeting of the Association for Computational Linguistics (1990).
15) Kaplan, R. M. and Bresnan, J.: Lexical-Functional Grammar: A Formal System for Grammatical Representation, Mental Representation of Grammatical Representations, Cambridge, Mass., MIT Press, pp. 173-381 (1982).
16) Kasper, R. and Rounds, W. C.: A Logical Semantics for Feature Strucutres, In Proc. 24th Meeting of the Association for Computational Linguistics, pp. 257-266 (1986).

17) Kasper, R.: A Unification Method for Disjunctive Feature Discription, In Proc. 25th Meeting of the Association for Computational Linguistics, pp. 235-242 (1987).
18) Kasper, R.: Conditional Description in Functional Unification Grammar, In Proc. 26th Meeting of the Association for Computational Linguistics, pp. 233-240 (1988).
19) Kay, M.: Parsing in Functional Unification Grammars, Natural Language Parsing, Cambrideg, Cambridge University Press (1985).
20) Kogure, K.: Parsing Japanese Spoken Sentences Based on HPSG, In Proc. of IWPT 89, pp. 132-141 (1989).
21) Lloyd, J. W.: Foundations of Logic Programming, Springer-Verlag (1984).
22) 松本裕治：論理型言語による自然言語へのアプローチ，自然言語の基礎理論，共立出版，pp. 51-86 (1986).
23) Moshier, M. D. and Rounds, W. C.: A Logic for Partially Specified Data Structures, In Proc. 14th ACM Symposium on Principles of Programming Languages, pp. 156-167 (1987).
24) Mukai, K.: Anadic Tuples in Prolog, ICOT TR-239 (1987).
25) Mukai, K.: Co-inductive Semantics of Horn Clauses with Compact Constraint, ICOT TR-562 (1990).
26) Nebel, B. and Smolka, G.: Representation and Reasoning with Attributive Description, Lecture Notes in Artificial Intelligence 418, Springer-Verlag, pp. 112-139 (1989).
27) Pereira, F. C. N. and Warren, D. H. D.: Definite Clause Grammars for Language Analysis—A Survey of the Formalism and a Comparison with Augumented Transition Network, *Artif. Intell.*, Vol. 13, pp. 231-278 (1980).
28) Rounds, W. C. and Manaster-Ramer, A.: A Logical Version of Functional Grammar, In Proc. 25th Meeting of the Association for Computational Linguistics, pp. 89-96 (1987).
29) Rounds, W. C.: Set Values for Unification-Based Grammar Formalisms and Logic Programming, CSLI Technical Report 129 (1988).
30) Rounds, W. C.: Complex Objects and Morphisms, A Set-Theoretic Semantics, to appear.
31) Shieber, S. M.: An Introduction to Unification-Based Approaches to Grammar, CSLI Lecture Notes No. 4 (1986).
32) Smolka, G.: A Feature Logic with Subsorts, LILOG Report 33, IBM Deutschland, Stuttgart, F. R. G. (1988).
33) Smolka, G. and Aït-Kaci, H.: Inheritance Hierarchies: Semantics and Unification, J. Symbolic Computation, Vol. 7, pp. 343-370 (1989).
34) Smolka, G.: Feature Constraint Logics for Unification Grammars, IWBS Report 93, IBM Deutschland, Stuttgart, W. Germany (1989).
35) Tuda, H., Hasida, K. and Shirai, H.: JPSG Parser on Constraint Logic Programming, In Proc. 4th Conference of the European Chapter of the Association for Computational Linguistics, pp. 95-102 (1989).
36) Winston, P. H.: Artifical Intelligence, Addison-Wesley Publishing Company (1977).
37) Yasukawa, H. and Yokota, K.: Labbeled Graphs as Semantics of Objects, 第 80 回データベースシステム研究会資料 DB-80 (1990).

(平成 3 年 3 月 25 日受付)

今村 誠 (正会員)

1961 年生. 1984 年京都大学工学部数理工学科卒業. 1986 年同大学院修士課程修了. 同年三菱電機(株)入社. 現在，同社情報電子研究所に勤務. 自然言語インタフェース，知識表現ツールなどの研究開発に従事.