自然言語処理 76－5
（1990．3．9）

タイプ付き素性構造を用いた生成過程の宣言的制御

上田 良寛　　小暮 潔
ATR自動翻訳電話研究所　　NTT基礎研究所

素性構造を入力とする生成システムにおいて、タイプ付き素性構造を導入することにより、句構造派生の制御を宣言的な記述で行う方法について述べる。この生成システムで用いる文法は、そのまま解析でも用いることができる双方向文法となっており、解析と生成の文法の一貫性を保つのに効果がある。また、この生成システムでは、句構造の複製を減らすために、選言を含む素性構造を導入している。このシステムは対話文翻訳システムの生成部として開発したものであり、文法は電話会話における話者の意図を適切に解析･生成するよう設計されている。

Declarative Control of the Generation Process using Typed Feature Structures

Yoshihiro UEDA
ATR Interpreting Telephony Research Laboratories
Sampeidani, Inuidani, Seika-cho, Kyoto, 619-02 JAPAN

and

Kiyoshi KOGURE
NTT Basic Research Laboratories
9-11, Midori-Cho, Musahino-Shi, Tokyo, 180 JAPAN

This article describes a bi-directional grammar generation system which utilizes typed feature structures to control the top-down derivation in a declarative way. This generation system also uses disjunctive feature structures to reduce the number of copies of the derivation tree. This generator will be the generation component of the spoken dialogue translation system and the grammar is designed to properly generate the speaker's intention in a telephone dialogue.

1. はじめに

対話においては、発話に含まれる話者の意図(発語内的力)は、その命題内容と同様に重要な働きをもっている。発語内的力を適切に翻訳するために、ATRでは意図伝達方式(Intention Translation Method)と呼ばれる方法を開発した(Kogure et al., 1988)。

対話文翻訳システムの生成部は、この発語内的力を適切に生成する必要がある。この文法は、発語内的力と表層の文の関係が宣言的に記述されていることが望ましい。特に、解析にも用いることのできる双方向文法ならば、文法の一貫性を保つのに効果がある。このような考えから、双方向文法を用いた生成システムを開発した(上田·小暮、1989)。このシステムは、単一化を基本にした文法(CFG+素性構造の制約)を用い、解析と逆方向に文法を適用し句構造を組み立てていくものである。

生成システムでは、同じ規則が何度も適用され停止に至らない場合がある(Shieber et al., 1989)。また、最終的に書換えの失敗に至る無駄な規則が適用される場合がある。これを避けるため、先のシステムでは、Assertionsという制約条件により宣言的記述で文法適用の制御を行う方法を提案した。

本稿では、このような宣言的な制御を、Assertions制約条件のような特別な機構を付加する代わりに、タイプ付き素性構造(Aït-Kaci, 1986)の導入による統一的な枠組みで行う方法について述べる。

また、規則適用の際に複数の規則の候補が生じた場合、その数だけ派生木(素性構造で表現された句構造)の複製を作る必要があった。素性構造に選言を含めることにより、派生木全体を複製することを避ける方法を示す。

ここでは、先ず、生成におけるルール適用の制御の必要性を考察する。次に、タイプ付き素性構造および選言つき素性構造にってこの問題を解決する方法について示す。また、発話意図を生成するためにHPSG (Pollard & Sag, 1987) に基づいて作成された文法を、生成例とともに示す。

2. ルール適用制御の必要性

2.1 停止性の問題

上田·小暮 (1989)が示した生成システムは、基本的には、CFG規則をトップダウンに適用し、親ノードの素性構造を子ノードに分配することにより、派生木を組み立てていくものである。このための文法は、D-PATRのもの(Karttunen, 1986)から作成した。

Shieber et al. (1989) は、このようなトップダウン生成システムには停止性の問題があることを指摘している。例えば、

```
S -> NP VP                                        (1)
    <S SEM CONT> == <VP SEM CONT>
    <VP SUBCAT FIRST> == <NP>
```

という規則においては、NPには素性構造による制約が何も与えらていないので、NPというカテゴリに属するどのようなもの(句、節、語)にも書き換えられることになる。

この問題は、ある要素に、意味素性(ここでは、<SEM CONT>)が与えられるまで、それ以上の派生を行わず待ち状態にすることで解決される(上田·小暮、1989)。Wedekind (1988)は、この条件を"connectedness"と呼んでいる。

左再帰文法規則も停止性の問題がある。次に示す規則(2)は動詞VP2のSUBCATリスト(下位範疇化フレーム)を、ある要素XPで埋める規則である。

```
VP1 -> VP2 XP                                     (2)
    <VP1 SEM CONT> == <VP2 SEM CONT>
    <VP2 SUBCAT FIRST> == <XP>
    <VP1 SUBCAT> == <VP2 SUBCAT REST>
```

解析の場合、この規則の適用は、SUBCATリストの要素を1個減すことになるので、SUBCATリストが尽きた時点で、それ以上この規則が適用されることはない。また、入力文字列が有限長であることも、規則の無限回の適用を制限している。一方、生成においては、規則(2)をVPノードのSUBCATリストに

要素を1個増やす方向に適用するため、無制限に適用可能になる。

SUBCATリストに長さの制限を加える(英語の場合主語を別にして最大2となる)ことで、この問題を処理することは可能である。この方法はオランダ語などでは適用できない(Shieber et al., 1989)が、現在我々は対象を日本語と英語に限定しているので、このような制限は無視できる。なお、この問題に関しては、再度検討する。

2.2 適切な文法規則の選択

上田·小暮(1989)が提示した問題は、複数の文法規則の左辺の要素につけられる素性構造が同じ形式をしている場合に、このような素性構造に対して生成時に適用すべき文法規則を選択できないということであった。

例えば、次の文法規則(3)を考える。文法規則(2)および(3)から得られる素性構造は図1のようになり、同じ形式をしている。

```
VP1 -> VP2 PP                                    (3)
    <VP1 SEM CONT> == <PP SEM CONT>
    <PP SUBCAT FIRST> == <VP2>
    <PP SUBCAT REST> == NONE
```

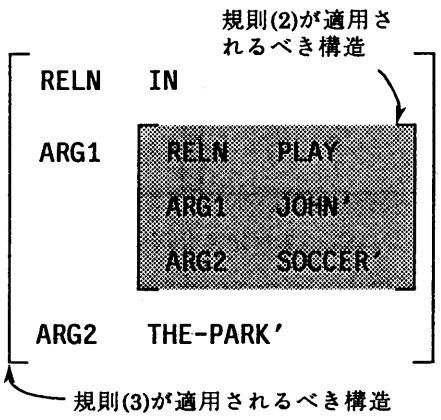

図1 素性構造の例

それぞれに対して適切な規則が選択できなければ、間違った方向に派生を続けていき、結果的に途中で行き詰まることになる。このような間違った方向への派生は効率に大きな影響を及ぼす。

これを避けるために導入したのが親の素性構造に対して制約条件を与えるAssertionsという記述である。Assertions制約条件は、文法規則につけられ、生成時にその文法規則が適用可能か否かをチェックするために用いられる。(2)、(3)のそれぞれの規則に対して、次のような記述を与える。

```
(2) (:ASSERTIONS
        (:EQ (:TYPE <SEM CONT RELN>) V))
(3) (:ASSERTIONS
        (:EQ (:TYPE <SEM CONT RELN>) P))
```

ここでTYPEはキーとなる素性(この文法では<SEM CONT RELN>素性)に対して、辞書項目中で与えられている。:and, :orなどのオペレータを用いて、Assertions制約条件を組み合せることができる。

ここまでの解決方法は、上田·小暮(1989)で示したものである。3節ではこれにタイプ付き素性構造を導入することを、4節では選言を含む素性構造を導入することを検討する。

3. タイプ付き素性構造の導入

Assertionsを導入した動機は、意味素性中のキーとなる素性のタイプを制約として扱うことにあった。通常の素性構造では、特別な解釈機構を導入することなしにこの目的を達成することができない。しかし、Aït-Kaci (1986) によるタイプ付き素性構造を導入すれば、特別な解釈機構なしに統一的に扱うことができる。

このタイプは、単純な分類でなく、階層を形成する。例えば動詞は、その下位範疇化フレームの要素数によって下位分類することができる(3.2で例を示す)。タイプ階層において、上位タイプ(supertype)は下位タイプ(subtype)の和(OR結合)を表し、下位タイプは上位タイプの積(AND結合)を表す。Assertions制約条件のうち、タイプのand, or組合せで表現された部分は、タイプ階層を用

いた表現に置き換えることができる。タイプ階層の利用を3.2および3.3で考察する。

3.1 文法規則の選択

タイプ付き素性構造を用いると、規則(2)のAssertions制約条件は、(2')の下線部のように記述することができる(ここ以降では現在のシステムにおける記述方法を用いる[1])。

```
VP =HC*=> (VP XP)                                    (2')
    (<!m sem cont>
        == <!head-dtr sem cont>)
    (<!head-dtr !subcat first>
        == <!comp-dtr-1>)
    (<!m !subcat>
        == <!head-dtr !subcat rest>)
    (<!head-dtr sem cont reln> == [V])
```

ここで下線部は、<SEM CONT RELN>素性がタイプVと単一化されることを示す。この単一化が成功する、すなわち、<SEM CONT RELN>素性の値がタイプ階層の中でタイプVの下にある場合に、この文法規則は適用可能である。

Assertions制約条件を用いる場合には、この制約条件の保持機構および解釈機構、タイプ解釈機構が独立したものとして付加される必要があった。しかし、タイプ付き素性構造の単一化を導入することで、これらの制約を統一的に取り込むことができる。

3.2 カテゴリとの関連づけ

タイプ階層の利用方法のひとつに、カテゴリ(ノンターミナルシンボル)との関連づけがある。カテゴリはタイプを用いて表現され、階層化がなされている。例えば、規則(2')で現れる下位範疇化フレームに入る要素XPは、図2に示されるようにNP, VP, PP, Sを一般化したものとなっている。

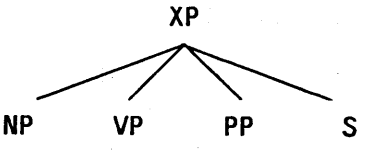

図2 カテゴリシンボルのタイプ階層

実際に下位範疇化フレームに入る要素は、個々の動詞で規定される。"must"の辞書項目中では、下位範疇化フレームの要素が1個だけで、そこには動詞句が入るということが、次のように記述されている。

```
(deflex "must" dyadic
    (<!subcat first> == [VP])
    (<!subcat rest> == [LIST-END])
    ...)
```

"must"が規則(2')と単一化されるときに、規則中のXPは、VP(must)のSUBCATフレームの記述により、[VP]と単一化され[VP]となる。その後の派生はVPからなされる(図3)。

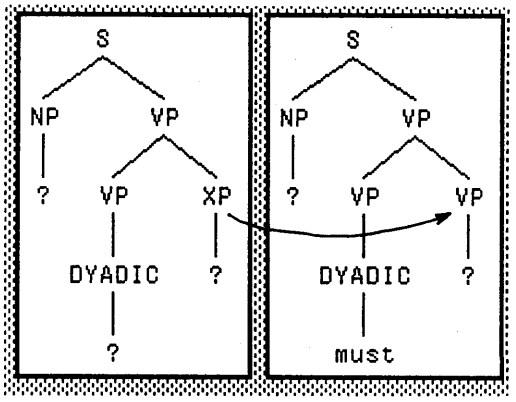

図3 未指定カテゴリの決定

解析の場合には、次のような規則を用意することで対処する。

```
XP == (VP)                                           (4)
XP == (NP)
...
```

ここで==リンクは、左辺と右辺が同一のレベルで単一化されること(右辺が左辺のdaughterに入らないこと)を指定している。

---

1. この規則で、=HC*=>リンクは、右辺の最初の要素がhead daughter、その他がcomplementになるということを表している。S=CH=>NP VPのようなcomplementが先にくる構成のために、=CH=>リンクも用意されている。
!マークがつけられたシンボルは別に定義されるテンプレートを意味している。

このような未指定カテゴリに対して、例えばD-PATR (Karttunen, 1986)では、特別なシンボルX, Yなどを CFGに導入し、CFG規則中でこのようなシンボルにあたったときに特別な処理を行うようにしている。このような特別な機構を用いなくとも、未指定カテゴリをタイプ階層の中に組み入れることにより、無理なく対処することができる。

3.3 停止性問題への対処

規則(2)の停止性の問題は、SUBCATリスト(下位範疇化フレーム)の長さに上限を与えることによって解決可能であることは触れた。英語の場合、主語を除けばSUBCATリストの長さはせいぜい2であるので、次のような制限を規則(2)に与えることができる。

```
(:or ((<!head-dtr !subcat rest>
       == [list-end]))
     ((<!head-dtr !subcat rest rest>
       == [list-end])))
```

この制限では全ての動詞に対して規則(2)を2回適用させることになる。DEFLEXで規定されるターミナル(辞書エントリ)への書き換えは、SUBCATリストの長さが0のもの、1のもの、2のものの3つの場合で試みられる。動詞が自動詞の場合の書き換えは、そのうちSUBCATリストの長さが0のものだけが成功することになる。全ての動詞でこのような失敗が2回起こることになり、無駄が大きい。

これを避けるために、動詞をその長さに応じて3つのサブタイプ(Monadic, Dyadic, Triadic)に分類し、次の制約を規則(2)に加える。

```
(:or ((<sem cont reln> == [dyadic])
      (<!head-dtr !subcat rest>
       == [list-end]))
     ((<sem cont reln> == [triadic])
      (:or (<!head-dtr !subcat rest>
            == [list-end])
           (<!head-dtr !subcat rest rest>
            == [list-end]))))
```

このように、タイプ階層の利用により、文法規則の適用を制御し、生成の効率化を図ることができる。

なお、このような左再帰規則(2)をやめ、

```
VP =HC*=> (Monadic)
VP =HC*=> (Dyadic XP)
VP =HC*=> (Triadic XP XP)
```

などのような規則に変更すると、さらに効率を向上させることができる。しかし、このようにすることは、HPSGのモジュラリティを損ねることになってしまう。例えば、ギャップがある場合の規則は、最初の方法の場合、

```
VP =HC*=> (VP)                                    (5)
    ----
    (<!head-dtr !subcat first>
         == <!gapin>)
    (<!m !subcat>
         == <!head-dtr !subcat rest>)
```

を追加すればよい。しかし、左再帰規則をやめた場合では、XPがギャップになる全ての可能性を規則で表現する必要がある。

4. 選言の導入

HPSGでは、主語、目的語などは、主動詞の下位範疇化フレームで指定されるため、主動詞が決定されるまでは決定されない。一方、主動詞は、主語-動詞の一致により、主語が決定されるまでは完全には決定されない。このため、主動詞を派生する際にその候補の数だけ派生木全体を複製しなければならなかった。主語が決定されればこのうちの一つしか生き残らないので、無駄である。

素性構造に選言を導入することにより、動詞の表層形の違いを一つの辞書項目にまとめることができる。このようにしてまとめられた辞書項目を、lexical unitと呼ぶ。"be"動詞の場合の例を示す。

```
(deflex-unit  |be-Unit|  dyadic
  (:or
    (!finite-form !present-tense
     (:or ((<word> == "am")
           !1sg-subj-agr)
          ((<word> == "are")
           (:or  (!2sg-subj-agr)
                 (!pl-subj-agr)))
          ((<word> == "is")
```

```
        !3sg-subj-agr))
    ......)   ;過去形、過去分詞など
```

主語が派生され、単一化されると同時に、この3つの表層形の候補のうちから1つが決定される(図4)。

この問題に対して、Shieber et al. (1989) は、"postponing lexical choice" という方法を提案している。選言を含む素性構造によってパッキングするこの方法では、語彙の選択が後に行われるのではなく、主語の決定と同時に行われている。

選言を含む素性構造の単一化には、Kasper (1987)の方法を用いている。

5. 文法と生成例

この生成システムのために開発した英語文法は 、HPSG (Pollard & Sag, 1987) とBorsley (1987) によるHPSGの修正版を基礎にしている。HPSGでは、ほとんどの言語情報は辞書項目中に置かれる。この生成用文法では、発語内的力関係(illocutionary force)も辞書項目中に記述されることが多い。例えば、発語内的力タイプPROMISE (行為拘束型)は、"will" の辞書項目中に次のように記述される。

```
(deflex-unit |will-Unit| dyadic
  (:or ((<sem cont reln> == [*promise*])
        (<sem cont obje>
           == <!subcat-1 sem cont>)
        (<!subcat-1 sem cont reln>
           == [*action*])
           ;主動詞は行為型
        !1st-subj-agr)
           ;主語は1人称(肯定文の場合)
    ...  ;;「未来」の意味の記述
```

発語内的力のうちのいくつかは、辞書項目でなく、文法規則中に記載されている。例えば、発語内的力REQUEST (行為指導型)は、命令文で表現されるので、命令文の規則に記述されている。また、"Would you give ... ?"のような疑問文でもREQUESTを表現できる。

生成例として、発語内的力タイプREQUESTを含む図5のような素性構造表現を入力とする。この素性構造(fs-1)から、次のような生成結果が得られる。

```
> (gen3 fs-1)
("would you send me a registration form"
"could you send me a registration form"
"send me a registration form")[2]
```

これらの文はいずれも同じ発語内的力を表現するが、それぞれの表現に込められた丁寧さのレベルが異なっている。丁寧さのレベルを表す素性(現在は<prag params politeness>)を導入し、対応する文法規則に丁寧さのレベルとの対応づけを記述する。先の素性構造に丁寧さのレベルを最高に設定したもの(fs-1-3)を入力すると、次のような結果が得られる。

2. NP to NPの形式("send a registration form to me")は、ここでは省略する。

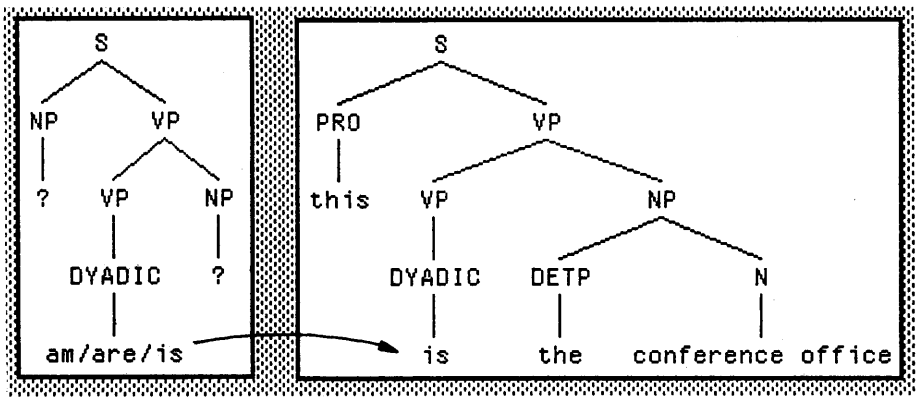

図4 主語による動詞表層表現の決定

```
> (gen3 fs-1-3)
("could you send me a registration form")
```

このように、双方向文法生成システムは、過生成の傾向があるため、表層発話に含まれる意図の違いを、文法作成者に気がつかせるという働きがある。このため、発話意図の研究の道具としても有用である。

なお、現在のところ表層の違いによる発話意図の違いをとらえきれていない場合も多い。例えば、副詞/副詞相当語句の位置の発話意図に対する関係も未整理である。この関係についてさらに研究する必要がある。

図6に生成結果を示す。

6. 他の研究との関連

Shieber et al. (1989)は、生成システムをトップダウンとボトムアップの2つのカテゴリに分け、トップダウン生成の欠点を指摘している。その主要な論点は、HPSGのようにhead部に情報が集中するような文法では、head部を先に決定することが望ましいということである。本稿で述べた生成システムはトップダウン生成に属するが、派生はhead部から順になされるように制御されており、トップダウン生成の欠点を克服している。

タイプ付き素性構造を生成に応用したものとしては、Emele (1989) の研究がある。Emeleは、Aït-Kaci (1986)のタイプ付き素性構造を表現方法として採用しているだけでなく、その評価機構(タイプ書き換え機構)が生成メカニズムとしても利用できることを示した。しかし、この方法では、生成の効率は必ずしも良くない。

7. おわりに

タイプ付き素性構造と選言を含む素性構造を用いて、生成過程を宣言的な記述で制御する方法について述べた。また、この生成システムが発話意図を扱う文法を研究するのにも適切であることを示した。

現在の文法は、発話意図と表層発話の対応づけを含め、まだ十分なものとなっていない。今後の課題としては、この文法をさらに拡充することがある。

最後に、常に有益な助言をくださる飯田仁主任研究員を初めとする言語処理研究室の諸氏に感謝の意を表明して、この稿を閉じます。

参考文献

上田、小暮 (1989):「素性構造を入力とする英文生成」、情報処理学会第38回全国大会、2D-4.

Aït-Kaci, H. (1986): "An Algebraic Semantics Approach to the Effective Resolution of Type Equations", Theoretical Computer Science 45, pp. 293 - 351, North-Holland.

Borsley, R. D. (1987): "Subjects and Complements in HPSG", Report No. CSLI-87-107, CSLI.

Emele, M. C. (1989): "A Typed Feature Structure Unification-based Approach to Generation", 電子情報通信学会NLC-88-32.

Karttunen, L. (1986): "D-PATR: A Development Environment for Unification-Based Grammars", Report No. CSLI-86-61, CSLI.

```
[[SEM [[CONT [CIRC[RELN [*REQUEST*]]
                 [AGEN ?X03[IND-OBJ[LABEL *SPEAKER*]]]
                 [RECP ?X02[IND-OBJ[LABEL *HEARER*]]]
                 [OBJE [CIRC[RELN [*SEND-1*]]
                             [AGEN ?X02]
                             [RECP ?X03]
                             [OBJE !A-REG-FORM']]]]]]]
                             ;ここでは!A-REG-FORM'という形で省略した。
  [PRAG [[HEARER ?X02]
         [SPEAKER ?X03]]]]    ;?で始まるシンボルにより同一の構造を指すことを示す。
```

図5 入力素性構造

Kasper, R. T. (1987): "A Unification Method for Disjunctive Feature Descriptions", in 25th ACL, pp. 235 - 242.

Kogure, K. et al. (1988): "A Method of Analyzing Japanese Speech Act Types", in 2nd MT Conference.

Pollard, C. and I. A. Sag (1987): "An Information-Based Syntax and Semantics, Volume 1, Fundamentals", CSLI Lecture Notes Number 13, CSLI.

Shieber, S. M. et al. (1989): "A Semantic-Head-Driven Generation Algorithm for Unification-Based Formalisms", in 27th ACL, pp. 7 - 17.

Wedekind, J. (1988): "Generation as Structure Driven Derivation", in Coling 88, pp. 732 - 737.

```
> (pprint-fs fs-2)
[SENTENCE[SEM [[CONT [CIRC[RELN [*PROMISE*]]
                          [AGEN ?X03[IND-OBJ[LABEL  *SPEAKER*]]]
                          [RECP ?X02[IND-OBJ[LABEL  *HEARER*]]]
                          [OBJE [CIRC[RELN [*SEND-1*]]
                                     [AGEN ?X03]
                                     [RECP ?X02]
                                     [OBJE !a-registration-form']]]
                          [TENSE [*PRESENT*]]]]]]
        [PRAG [[HEARER ?X02]
               [SPEAKER ?X03]]]]
> (gen3 fs-2)
("i will send you a registration form" "i will send a registration form to you")

> (pprint-fs fs-3)
[SENTENCE[SEM [[CONT [CIRC[RELN [*IDENTICAL*]]
                          [IDEN !the-conference-office']
                          [OBJE ?X01[IND-OBJ[LABEL  *SPEAKER*]]]
                          [TENSE [*PRESENT*]]]]]]
        [PRAG [[HEARER [[LABEL  *HEARER*]]]
               [SPEAKER ?X01]]]]
> (gen3 fs-3)
("this is the conference office")

> (pprint-fs fs-4)
[SENTENCE[SEM [[CONT [CIRC[RELN [*DESIRE-1*]]
                          [AGEN ?X02[IND-OBJ[LABEL  *SPEAKER*]]]
                          [OBJE [CIRC[RELN [*ATTEND-2*]]
                                     [AGEN ?X02]
                                     [SLOC !the-conference']]]
                          [TENSE [*PRESENT*]]]]]]
        [PRAG [[HEARER [[LABEL  *HEARER*]]]
               [SPEAKER ?X02]]]]
> (gen3 fs-4)
("i would like to attend the conference" "i want to attend the conference")

> (gen3 fs-4-1)       ;fs-4に(<prag params moderate> == -)を与えたもの
("i want to attend the conference")
> (gen3 fs-4-2)       ;fs-4に(<prag params moderate> == +)を与えたもの
("i would like to attend the conference")
```

図6 生成例