自然言語処理 113－8
（1996．5．29）

LFG 機能スキーマの帰納的獲得

小島丈幸 山口昌也 乾伸雄 小谷善行 西村恕彦
(東京農工大学 工学部 電子情報工学科)

本稿では、**LFG**(語彙機能文法) の機能スキーマを解析例から帰納的に獲得する手法を提案する。本稿で扱う **LFG** は無制限なものではなく、幾つかの制限を加える。その枠組の中で、木構造で表現される表層構造と機能構造で表現される深層構造の対から文法規則の制約条件である機能スキーマを獲得するアルゴリズムについて述べる。本アルゴリズムでは語彙の知識は既知の情報として扱う。本アルゴリズムは、可能な限りきつい機能スキーマを生成する段階と、似た規則を一般化する段階とに分かれる。

定義した **LFG** から生じる例を獲得システムに与え、逆に **LFG** を再構成する実験を行い、獲得した規則を評価した。その結果、正しく機能スキーマが生成され、一般化の段階において生成された規則の約半分の数に一般化されて獲得されることが示された。

Inductive Acquisition of Functional Schemata on LFG

Takeyuki KOJIMA,Masaya YAMAGUCHI, Nobuo INUI,
Yoshiyuki KOTANI, Hirohiko NISIMURA
(Dept. of Computer Science, Tokyo University of Agric. and Tech.)

This paper proposes a system which inductively acquires functional schemata of LFG from examples. An example is a pair of a surface structure as a tree and a deep structure as a functional structure with several constraints. We deal with lexical knowledges as already-known information. The Algorithm consists of the two phases: (1) to generate the schemata strictly and (2) to genaralize rules from similar rules.

Using exapmles generated by a set of LFG, we made the experiment in order to evaluate the schemata acquired. The experimental result showed that functional schemata is generated correctly and that the number of schemata acquired is cut in half.

1 はじめに

自然言語処理に必要とされる文法的知識を例から自動的に獲得する機構をシステムに組み込む研究が多くなされている。このような帰納的な文法的知識の獲得の研究では、文脈自由文法 (**CFG**) が用いられることが多い [2]。これらの研究には、文法的曖昧性を確率文脈自由文法 (**PCFG**) で解消するもの [1] も含む。一方で、主辞句構造文法 (**HPSG**) や語彙機能文法 (**LFG**) を用いて、確率に基づかない制約による曖昧性解消を行うこともできるが、獲得方法はまだ明らかではない。

本稿では、**LFG** を文法的枠組に用いて文法的な知識の獲得を行う。語彙に関する辞書を既知の情報として利用し、木構造で与えられる表層構造と機能構造で与えられる深層構造から帰納的に **LFG** の構成素構造規則を獲得するアルゴリズムを提案する。**CFG** の書換え規則が同じでも機能スキーマの複雑さが変化すると、表現可能な言語が変化する。そこで、本稿で提案する獲得アルゴリズムでは、用いる

規則が可能な限り単純になるような制限を設けた。

2 機能構造と機能スキーマ

2.1 構成素構造規則

LFG は構成素構造規則で構成され、自然言語の形態素列を構成素構造という深層構造に変換する。構成素構造規則は一般的に次のような形をしている。

$$P \rightarrow \underset{\mathsf{sc}_1}{C_1} \quad \underset{\mathsf{sc}_2}{C_2} \quad \cdots \quad \underset{\mathsf{sc}_n}{C_n} \qquad (1)$$

式(1)は n 個の子ノードを持つ構成素構造規則である。この構成素構造規則を本稿では、**CFG** の書換え規則に制約の記述である機能スキーマを加えたものである [3, 4] と考える。式(1)では、sc_i が機能スキーマになる。親ノード P が子ノード $C_1, C_2, \cdots, C_n$ に展開されるという **CFG** の書換え規則に、機能スキーマ $\mathsf{sc}_1, \mathsf{sc}_2, \cdots, \mathsf{sc}_n$ が付加したものである。機能スキーマ sc_i は親ノード P の機能構造と子ノード C_i の機能構造の間の関係を規定するはたらきを持つ。

図1に構成素構造規則の一例を示す。この規則は、連用句と動詞から文が構成されるという情報を **CFG** 書換え規則で表現している。同時に、機能スキーマによって、文に付く機能構造の対象という属性値が連用句に付く機能構造とユニフィケーションされることを示している。

$$\text{文} \rightarrow \underset{\substack{(\text{対象},\uparrow)=\downarrow \\ (\text{格マーカ},\downarrow)=\text{ガ}}}{\text{連用句}} \quad \underset{\substack{\uparrow=\downarrow \\ (\text{ガ格},\downarrow)=\text{対象}}}{\text{動詞}}$$

図 1: 構成素構造規則の例

2.2 構成素構造規則と機能構造

LFG で解析した結果として得られる構成素構造は、**CFG** における木構造の各ノードに機能構造が付加したものになる。機能構造はそのノードの意味的な記述を含む深層的な構造になっている。特に解析結果のトップノードに付加する機能構造は、解析対象である形態素列全体に対する深層構造になる。

図2に構成素構造の一部の例を示す。文・連用句・動詞が非終端記号であり、その下にそれぞれに付加している機能構造が示してある。連用句と動詞の機能構造が既知として、図1に挙げた構成素構造規則を用いると、図2に示した文の機能構造が得られる。

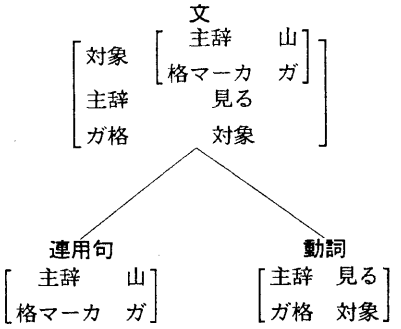

図 2: 構成素構造の一部

機能構造 F は、属性 a とその属性値 v の対 $\langle a, v\rangle$ を要素として持つ集合である。本稿では、機能構造 F が属性名 a に v という値を持つこと、すなわち、$\langle a, v\rangle \in F$ を $(a\ F) = v$ と書くことにする。

また、$(a\ F)$ が機能構造であることを $fst[(a\ F)]$、機能因子 (機能構造以外) であることを $\neg fst[(a\ F)]$ と書くことにする。図2を例にとって考える。文に付加する機能構造を F_s、連用句に付加する機能構造を F_p とすると、(対象 F_s) $= F_\mathrm{p}$、fst[(対象 F_s)] であり、(主辞 F_s) = 見る、$\neg fst$[(主辞 F_s)] となる。

2.3 機能スキーマの分類

CFG 書換え規則が同じであっても、そこに付加する機能スキーマが異なれば、異なる構成素構造規則が構成される。逆に、同じ言語を表現するにも、**CFG** 書換え規則の非終端記号が細かければ簡単な機能スキーマだけでよいし、機能スキーマに複雑な記述を許せば非終端記号の種類を減らすことができる。本稿では、機能スキーマは可能な限り単純なものにして、獲得アルゴリズムを設計する。

そこで **LFG** が文法体系として成立する最も単純な機能スキーマを考察した。その結果、機能スキーマの最も基本的な形式として、次の三つの機能スキーマの分類になることを示すことができた。

- 継承性スキーマ θ_i
- 制限性スキーマ $\lambda_{ij} = \langle a_{ij}, v_{ij}\rangle$

• 決定性スキーマ γ

継承性スキーマ θ_i は、主に親ノードの機能構造と子ノードの機能構造の対応関係を規定する。子ノードの機能構造をそのまま親ノードに持ち上げることを示す "↑=↓" や、子ノードの機能構造を親ノードの属性名 a の値にすることを示す "$(a,\uparrow)=\downarrow$" が継承性スキーマである。

制限性スキーマ λ_{ij} は、そのスキーマが付いている構成素構造規則がどのような子ノードに適用可能なのかを規定する。"$(a_{ij},\downarrow)=v_{ij}$" の形式を持ち、$(a_{ij}\ F_i)=v_{ij}$ を要求する。

決定性スキーマ γ は、親ノードの機能構造の属性を強制的に決定するはたらきをもつ機能スキーマである。決定性スキーマは子ノードの機能構造にはまったく関係ない。親ノードの機能構造の属性 a の属性値が v であることを要求していて、"$(a,\uparrow)=v$" の形式をしている。

2.4 構成素構造規則の種類

本稿では、獲得を簡易化するために、扱う構成素構造規則に対しても制限を加える。構成素構造規則は、語彙の辞書を表現する語彙的規則と統語的な結び付きを表現する統語的規則に二分される。語彙的規則は、子ノードが一つの形態素で、親ノードはその範疇 (ほぼ品詞に対応する) になっている。統語的規則は、親ノード・子ノードとも非終端記号であり、子ノードの数は一つ以上である。図 1 は統語的規則である。

二種類の構成素構造規則に対して、それぞれ 2.3節で定めた機能スキーマを付加し、最も単純な構成素構造規則を決定した。

統語的規則 一つの子ノード $C_i\ (i=1,\cdots,n)$ に対して継承性スキーマ θ_i が一つ、制限性スキーマ $\lambda_{ij}\ (j=1,\cdots,r_i)$ が $r_i\ (\geq 0)$ 個つく。

語彙的規則 決定性スキーマ γ が一つ以上つく。

3 構成素構造規則の獲得

3.1 機能スキーマの獲得

本稿で提案する獲得アルゴリズムでは、入力として、互いに対応する木構造と機能構造を入力を用いる。つまり、ある自然言語文に対する構成素構造を考え、その構成素構造における木構造とトップノードの機能構造が入力になる。このとき、語彙に関する辞書項目である語彙的規則を既知の情報として扱う。したがって獲得の対象になるのは統語的規則だけである。

入力に木構造を与えるということは、式 (1) における **CFG** の書換え規則に相当する情報を与えることになる。本稿では、**LFG** の構成素構造規則を **CFG** の書換え規則と機能スキーマから構成されているものとして扱っているので、実際に獲得するのは機能スキーマ部分だけになる。

機能スキーマの獲得アルゴリズムは、機能スキーマの生成過程と一般化過程の二つの処理過程から構成される。機能スキーマの生成過程では、継承性スキーマを一つつけ、制限性スキーマを最もきつい制約になるようにつける。2.4節に述べたように、統語的規則としては $r_i(\geq 0)$ 個の制限性スキーマのついたものを扱うことにしているので、この r_i ができるだけ大きくなるように制限性スキーマを生成する。

3.2 機能構造の含有因子

機能スキーマの生成では、親ノード P の付加している機能構造 F_0 の属性値 $(a\ F_0)=v$ を順に取り出す。そして、それがどの子ノードから由来しているのかを調べて子ノードにつく機能スキーマを決定する。

機能構造の特定の属性がどの子孫からもたらされたのかを判断するために、任意の機能構造に対する含有因子というものを定義する。ある機能構造 F に対する含有因子 F^* とは、F に含まれているすべての機能因子を再帰的に取り出したものである。式では次のように表現できる。

$$F^*=\bigcup_{fst[(F\ a)]}\{\langle a,(F\ a)\rangle\}\ \cup\ \bigcup_{\neg fst[(F\ a)]}(F\ a)^*. \quad (2)$$

2.3,2.4節で定義した機能スキーマと構成素構造規則の仕様に従った **LFG** であれば、親ノードの機能構造の含有因子は子ノードの機能構造の含有因子から求めることができる。式 (1) の親ノードの機能構造 F_0

の含有因子を F_0^* とすると、子ノード $C_i(i=1\cdots n)$ の機能構造を F_i として、

$$F^* = \bigcup_i F_i^* \tag{3}$$

となる。

本稿の獲得アルゴリズムのように語彙的規則を与えた場合には、入力として与えられた木構造の最も形態素に近いノードの機能構造は辞書びきで完全にわかる。この機能構造の含有因子は、式(3)から容易に算出できる。木構造が与えられていれば、子ノードの機能構造の含有因子から親ノードの機能構造の含有因子は式(2)で求めることができる。機能スキーマの生成は属性が機能構造のときと機能因子のときで異なるので、次にそれを分けて考える。

3.3 機能因子に対する機能スキーマ

$(a\ F_0) = v, \neg fst[v]$ のとき、$\langle a, v\rangle \in F_i^*$ なら $\langle a, v\rangle$ がその子ノード C_i から由来したものとする。$\langle a, v\rangle \in F_i^*$ となる可能性は二つ考えられる。一方は $(a\ F_i) = v$ の場合で、$F_i \subseteq F_0$ である。このとき、継承性スキーマ "↑=↓"、制限性スキーマ"$(a, \downarrow) = v$" をつける。

他方は $(a\ F_i) \neq v$ でも、$\langle a, v\rangle \in (a\ F_i)^*$ が成り立つ場合である。この場合には、必ずしも継承性スキーマ "↑=↓" がつくとは限らない。

両者の違いを判断するためには $(a\ F_i) = v$ が成り立つかどうかを調べなければならない。しかし、F_i^* が既知でも F_i は未知なので、その成立を判断できない。本研究では、後者があり得ないものと仮定して獲得アルゴリズムを考えた。もし、後者が起きた場合には、入力された木構造の一部で機能スキーマの獲得に失敗する。

3.4 機能構造に対する機能スキーマ

$(a\ F_0) = v, fst[v]$ のとき、$v^* \subseteq F_i^*$ なら $\langle a, v\rangle$ が C_i から由来したものとする。$v^* \subseteq F_i^*$ となる可能性も二つに分けて考えられる。一方は $F_i = v$ のときで、継承性スキーマとしては "$(a, \uparrow) = \downarrow$" をつける。制限性スキーマは $\langle a', v'\rangle \in v$かつ$\neg fst[a']$ を満たすすべての a' について、"$(a', \downarrow) = v'$" をつける。

他方は、$(a\ F_i) = v$ のときである。このときは継承性スキーマとして、"↑=↓" をつける必要がある。しかし、ここでも F_i が未知であるために、$(a\ F_i) = v$ が成り立つかどうかを調べることができない。本研究では、機能構造に関する機能スキーマの生成にさきだって機能因子に関する機能スキーマの生成を行い、すでに "↑=↓" がついている場合は $(a\ F_i) = v$ を、ついていない場合は $F_i = v$ を仮定した。

3.5 規則の一般化

少ない例から効率よく規則を獲得するために、機能スキーマの生成によって得られた規則の一般化を行う。ここで一般化の対象にするのは、機能スキーマの生成の際に過剰に付加された制限性スキーマである。

親ノード P とすべての子ノード C_i、そして各子ノードに付加された継承性スキーマ θ_i が等しい二つの構成素構造規則 R, R' の間に距離 $d(R, R')$ を定義する。子ノード C_i の制限性スキーマを$\boldsymbol{\Lambda}_i = (\lambda_{i1}, \lambda_{i2}, \cdots, \lambda_{ir_i})$ として、距離 $d(R, R')$ は、

$$\begin{aligned} d(R, R') &= \sum_{i=1}^{n} d(\boldsymbol{\Lambda}_i, \boldsymbol{\Lambda}_i'), \\ d(\boldsymbol{\Lambda}_i, \boldsymbol{\Lambda}_i') &= d_0(\boldsymbol{\Lambda}_i - X, \boldsymbol{\Lambda}_i' - X), \\ &\quad \text{ただし、} X = \boldsymbol{\Lambda}_i \cap \boldsymbol{\Lambda}_i' \\ d_0(A, B) &= \left|\{a|\langle a, v\rangle \in A\} \cup \{a|\langle a, v\rangle \in B\}\right| \end{aligned}$$

になる。ここで、$d(\boldsymbol{\Lambda}_i, \boldsymbol{\Lambda}_i')$ は i 番目の子ノードと親ノードの間の距離を示している。これは $\boldsymbol{\Lambda}_i$ と $\boldsymbol{\Lambda}_i'$ に共通しない属性の数になる。

一般化は $d(R, R') = 1$ のときに行う。$d(R, R') = 1$ が成り立つときは、R の $\langle a_{ij}, v_{ij}\rangle$ と R' の $\langle a_{ij}, v_{ij}'\rangle (v_{ix} \neq v_{ij}')$ 以外の制限性スキーマがすべて等しい場合と、一方の $\langle a_{ij}, v_{ij}\rangle$ が他方の $\boldsymbol{\Lambda}_i$ に含まれていない場合がある。

前者の場合は、二つの構成素構造規則から属性名 a_{ij} に関する制限性機能スキーマを消去することで構成素構造規則の一般化ができる。後者の場合は、属性名 a_{ij} のない構成素構造規則に吸収される形で一般化がなされる。

4 実験

4.1 実験システムの構成

実験に用いたシステムの構成図を図 3 に示す。まず、24 種類の統語的規則からなる日本語のごく簡単な **LFG**(図 3 中の元規則) を定義した。次にこの **LFG** に基づくパーザを作り、用意したテキストを解析して構成素構造を得た。この構成素構造から木構造と機能構造を抽出し、獲得システムへ入力した。

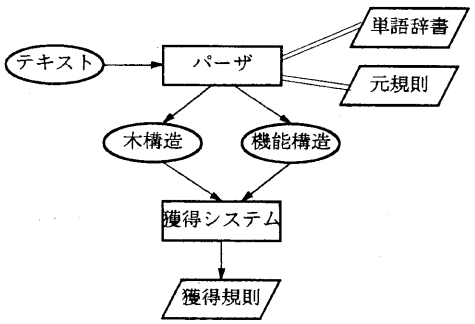

図 3: 実験システム構成図

4.2 入力テキスト

用意したテキストは、**IPAL** 名詞辞書 [5] に用例として載せられているものを抽出・一部改変したもので、分かち書きされた形態素列 20 文である。特徴を表 1 に示す。表中の規則数とは、そのテキストを木構造表現したときに用いられる統語的規則の数である。獲得システムへの入力の木構造は、この統語的規則と語彙的規則 (形態素数と同数) で構成される。

表 1: 入力テキストの特徴

番号	1	2	3	4	5	6	7
形態素数	14	13	6	11	11	14	5
規則数	25	30	12	17	20	28	10
番号	8	9	10	11	12	13	14
形態素数	5	9	5	10	7	5	7
規則数	14	20	10	22	18	10	16
番号	15	16	17	18	19	20	–
形態素数	7	6	6	12	12	11	–
規則数	18	12	12	26	24	28	–

5 結果と考察

5.1 実験結果

実験では統語的規則を 24 種類用意したが、出現頻度の低いものは獲得結果を評価するのに十分でない。そこで、出現頻度が高いものだけを考える。図 4 に、実験で 20 回以上出現した八つの規則の出現数を示す。図中、origin は例を生成するために用いた規則 (元規則) を、generated は例で使われた規則を、acquired は獲得した規則を表す。これらの統語的規則それぞれに対して、生成された規則数と獲得した規則数の関係を図 5に示す。

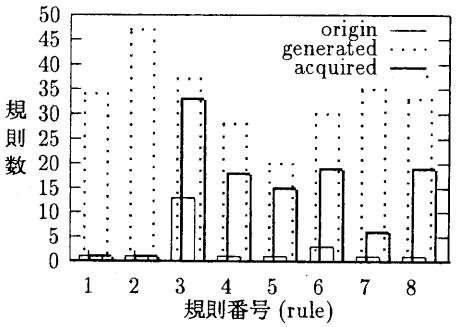

図 4: 構成素構造規則別の出現数

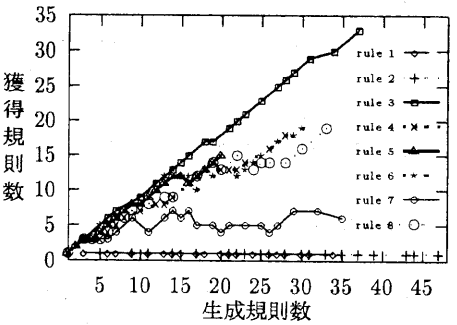

図 5: 構成素構造規則別の一般化の傾向

図 4,5 を見ると、獲得された規則が幾つかのタイプに分けられることがわかる。機能スキーマの生成段階は正しく行われたので、獲得規則の一般化の効果をタイプ別に考察する。

5.2 容易に一般化される規則

ほとんどの入力例に含まれていて、即座に一般化されるタイプの規則がある。これに相当するのは、「連体助詞→助詞 (図 5 の rule 1)」と「名詞句→名詞 (rule 2)」である。図 5 からわかるように、例生成用の規則と同じ一つの規則として獲得される。実際には、どちらも制限性スキーマのない規則になっている ($r_i = 0$)。

これらの構成素構造規則は文を構成する際に、形態素に近い位置で頻繁に使われている規則である。形態素により近いので機能構造にあまり属性-値対がなく、付加される制限性スキーマも比較的少ない。このため、少ない生成規則数で一般化が終了してしまい、早い時期に獲得規則数が一定になる。

5.3 一般化の進まない規則

使用頻度が高いにもかかわらず、生成される規則同士の距離が遠いために一般化が進まないタイプの規則がある。これに相当する規則は「述句→連用句 述句 (rule 3)」、「句→述句 (rule 4)」、「文→句 結び (rule 5)」である。このうち、「述句→連用句 述句」は、例生成用の規則でも多様な制限性スキーマがついていた (図 4)。

これらの構成素構造規則は木構造を形成したときに、形態素から比較的離れた位置で使われる規則である。形態素からより離れているということは、一般に、たくさんの形態素がその構成素構造規則の子孫であるということであり、この構成素構造規則の適用されるノードに付加している機能構造により多くの形態素の機能構造が関連しているということである。このため、機能スキーマの生成段階で、これらの構成素構造規則にはより多くの制限性スキーマがつけられることになる。すると、これらの構成素構造規則は規則ごとに多様な制限性スキーマを付加されることになり、構成素構造規則同士の距離は遠くなる。

5.4 途中で大きく一般化される規則

三つ目のタイプは、一つ目と二つ目のタイプの中間のタイプである。このタイプに当てはまるのは、「名詞句→連体句 名詞句 (rule 6)」、「連用句→名詞句 連用助詞 (rule 7)」、「述句→用述句 (rule 8)」である。特に「連用句→名詞句 連用助詞 (rule 7)」において、途中で大きく一般化される傾向が顕著である。

獲得された規則の中に互いの距離が 2、3 程度のものがあっても、一般化は起こらず別々の規則として扱われる。そこに、双方に対しての距離が 1 になるような新たな規則が生成されると、急激に一般化が進むと考えられる。

6 むすび

本稿では、制限された **LFG** の枠組の中で、木構造と機能構造から帰納的に **LFG** の機能スキーマを獲得するアルゴリズムを提案した。

より表現力の強い機能スキーマや構成素構造規則に対する獲得が今後の課題として挙げられる。他にも、多量の入力データを得られるように、入力データの形式を検討することも必要である。

謝辞

本研究は、東京農工大学工学部電子情報工学科の西村小谷研究ユニットで行われた。ユニット内の方の本研究に対する助言と協力に感謝の意を表する。

参考文献

[1] 白井 清昭, 徳永 健伸, 田中 穂積 : コーパスからの文法の自動抽出, 情報処理学会自然言語処理研究会, Vol. 101. 11, 1994.

[2] 山本 幹雄, 中川 聖一 : 自然言語の構文・意味解析規則の機能的学習システム, 情報処理学会自然言語処理研究会, Vol. 55. 5, 1986.

[3] 淵 一博 監修, 古川 康一, 溝口 文雄 共編 : 自然言語の基礎理論 , 共立出版, 1986.

[4] ピーターセルズ 著, 郡司 隆男 , 田窪 行則, 石川 彰 訳: 現代の言語の基礎理論 , 産業図書, 1991.

[5] **IPAL** 名詞辞書, 情報処理振興協会, 1994