単一化文法を用いた実用的な日本語生成

荻野 紫穂　　武田 浩一
日本アイ・ビー・エム株式会社 東京基礎研究所
shiho@trl.ibm.co.jp　　takeda@trl.ibm.co.jp

1 はじめに

単一化文法を使った日本語生成では、文の概念表現と表層構造との体系的な写像が、詳細部分ではあまり定まっていないため、きめ細かな日本語表現を出力しようとする際に、いろいろな問題が起こることが多い。例えば、並列句や語順、共起関係の扱いなどは、大まかな枠組だけでは扱い切れないことも多く、その制御に困難さが伴い易い。また、こういった困難の解決には、表層構造だけでなく、概念表現と表層構造との写像やレキシコン情報の表現などからの、統一的な扱いが必要とされることも多い。

我々が現在開発中の英日機械翻訳システム Shalt2 は、日本語生成部に、単一化文法による解析／生成双方向文法を用いている。本稿では、この文法の生成用文法としての面を取り上げ、並列句処理などにおけるセットの処理のための記述拡張、生成順序制御に使用する情報とその処理、規則や好ましい表現の優先順位の制御、文法・レキシコン・概念表現とのインタフェース、更に、日本語生成における効率化における工夫などについて述べる。

2 単一化文法の拡張

2.1 文法の枠組

まず、本稿で説明する単一化文法の等式と、生成の際の素性構造の処理について、簡単に触れる。

疑似単一化文法はの句構造規則は、

$r1$: $X_0 \rightarrow X_1 X_2 \ldots X_m (X_i \in V_N)$

$r2$: $X_0 \rightarrow a_1 a_2 \ldots a_n (X_i \in V_N, a_i \in V_T)$

という句構造の記述と、単一化に関する0個以上の等式 $eq_1, \ldots, eq_k$ の集合からなる（V_N は非終端記号の集合、V_T は終端記号の集合）。各 eq は、

$e1$: $\langle x_i L_1 \rangle = \langle x_j L_2 \rangle$

$e2$: $\langle x_i L_1 \rangle = a$

$e3$: $\langle x_i L_1 \rangle = c\ a$

$e4$: $\langle x_i L_1 \rangle = *defined*$

$e5$: $\langle x_i L_1 \rangle = *undefined*$

という形をしている（a は定数）。ここで L_n をパスといい、素性名の列（空でも可）を示す。x_n は、上記の句構造記述の n 番目の非終端記号 X_n に対応する素性構造を表現している（左辺の非終端記号を0番目とする）。

$e1$ は、$x_i L_1$ と $x_j L_2$ との単一化の結果で双方を置き換える。$e2$、$e3$ は、素性が定数か、空でない素性構造の場合に成功するが、単一化の後、$e3$ で参照された素性は定数となる。$e5$ は、素性が空または存在しない場合に成功する。

生成においては、素性構造を句構造に分解し、単語導出に関する各素性を等式で参照し単一化することによって、空（または存在しない値）にする、つまり《外す》作業を繰り返す。その際には原則的に、分解された各構文要素について、構文的・形態素的に遠い構成要素に対応する素性から順番に外していき、最後にヘッドとなる素性構造が残るようにする。これは、いわゆる修飾ー被修飾の関係を表す規則でも、形態素の接続によって表される規則でも原則的には同じである。全ての単一化が成功して終端記号、或いは前終端記号に至った時は、ヘッドの基幹情報が定数として残るが、その基幹情報は、原則として辞書のレキシコン情報として書き込まれているものと単一化が可能である[1]。

2.2 集合制約のための拡張

名詞句や動詞句などの並列構造は、並列句を構成する要素のどれかをヘッドに定めるのではなく、セットが1つの構文要素としてセットの外側から扱われるものとする[2][4]。セットと、同じ修飾要素などをまとめるリストとは扱いが異なる。ここではまずセットの扱いについて述べ、次にセットの扱いと対比してリストの扱いについて述べる。

セットは以下のように記述される。

```
(*set*
 （セット共通の素性構造）
 （セット・メンバの素性構造）
```

[1] 規則に定数が記されている場合には、辞書のレキシコンに基幹情報が記されている必要はない。

[2] セットはヘッドを持たないとすることもある [7]。

.....

（セット・メンバの素性構造） ）

最初の*set*は、この構文要素がセットであることを示す。セット共通の素性構造には、セット自体が持つ素性が記述され、セットのメンバには配分されない（cf. [6]）。現在は、実際には接続詞などセットのメンバ同士の関係を示す素性がセット共通の素性構造に含まれており、品詞や活用型などの素性は含まれない[3]。

セット共通の素性構造に対する等式は、

$e1'$: $\langle x_i L_1 \rangle = s \langle x_j L_2 \rangle$

$e2'$: $\langle x_i L_1 \rangle = s\ a$

という形をしている。これらは $e1$、$e2$ に対応する等式で、x_i をセットと見做し、そのセット共通の素性構造のパス L_1 と右辺の値を単一化する。セットの各メンバの素性構造は変わらない。

付属語の接続などにより、セットの品詞や活用型・接続型などの素性を単一化する必要がある場合は、セットの最後のメンバの品詞や活用型を参照したいことが多い。日本語の場合、セットヘッドとする構文要素の活用型・接続型は、ほぼセットの最後のメンバのそれが受け継がれるからである。この際、活用型などは、セットのメンバ全体に共通する性質ではないので、セット共通の素性に単一化することはできない。また、同じ理由により、セットのメンバに配分されるような単一化をすることもできない。このような単一化を扱うため、セットの最後のメンバである素性構造に関する等式を導入する。

セットの最後のメンバに対応する素性構造に関する等式は、

$e1''$: $\langle x_i L_1 \rangle = l \langle x_j L_2 \rangle$

$e2''$: $\langle x_i L_1 \rangle = l\ a$

$e3''$: $\langle x_i L_1 \rangle = lc\ a$

$e4''$: $\langle x_i L_1 \rangle = l\ *defined*$

$e5''$: $\langle x_i L_1 \rangle = l\ *undefined*$

という形をしている。セットの最後のメンバーの素性を参照すること以外、働きは同じである。これらの等式は、セット以外の素性構造に対しては、2.1の等式と同じ働きをする。逆に 2.1の等式の結果は、通常通りセットのメンバ全てに配分される（図 1）。

これらに加え、

$e6$: $x_i = *set*$

$e7$: $x_i = *plain*$

という等式を便宜的に導入する。$e6$ は x_iがセットである時に成功し、$e7$ は x_iがセットでない時に成功する。各素性構造の素性の値自体は変化しない。

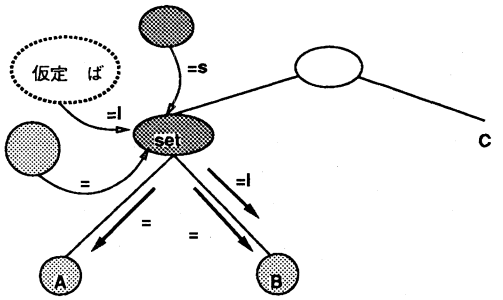

図 1: セットの規則

2.3 リスト

連体修飾などでは、1つのヘッドに対して複数の修飾句が存在することがある。この場合、複数の修飾要素は1つにまとめられ、リストとして扱われる。リストはセットとは違い、共通の素性構造を持たない。また、リストは全てが似たような構文的性質を持つとは限らない順序リストであるため、最後の要素の素性にリストのメンバ全ての素性を代表することはできない。つまり、リストのメンバ全体を個別に処理する必要がある。具体的には、リストは以下のように表される。

```
(*multiple*
 (修飾要素の素性構造)
    ......
 (修飾要素の素性構造) )
```

リストは、各要素を個別に扱う必要があるため、これに対応する等式は、

$e1'''$: $\langle x_i L_1 \rangle > \langle x_j L_2 \rangle$

となる。この等式は、$x_i L_1$をリストの可能性がある素性構造と見做し、その最初の要素と $x_j L_2$とを単一化する。このリストは順序リストなので、単一化は、必ずリストの最初の要素と行なわれる。この等式が成功すると、$\langle x_i L_1 \rangle$ は、最初の要素が取り除かれたリストと置き換えられる（図 2）。リストでない素性構造に対しては、この等式は $e1$ と同じ働きをする。

[3]セット共通の素性構造は、例えばセットの各メンバを修飾する修飾要素と、セット全体を修飾する修飾要素のスコープの区別などに使用することもできるが、現在は構文的には、全てがセット全体を修飾するように、スコープを《ぼかして》いる。

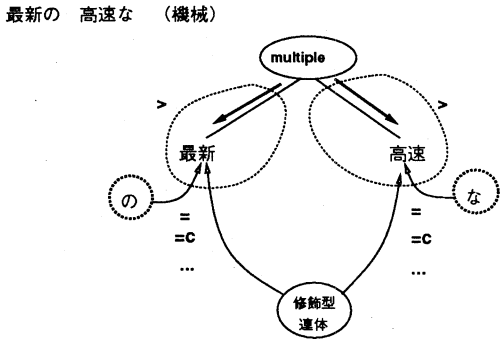

図 2: リストの規則

3　レキシコン、概念表現とのインタフェース

3.1　レキシコンによる表現の優先と導出

“know:知る・知っている”のように、英語側に特別な情報がなくても、日本語側がテイルをとる動詞がある。こういった日本語側の各語に依存した情報は、日本語側のレキシコンの情報として辞書に記述する。

このテイルは動詞そのものの直後に接していなければならず、また、このテイルが生成される場合は、例えば現在進行を表すテイルは生成されてはならない。テイルの生成に限らず、付属語の生成順序などは、《外側の》形態素から順番に外されていく。よって、ある形態素生成に対応する規則が適応される際には、それよりも遠い位置ににあるべき形態素に対応する素性が残されていてはならない。こういった生成は、それぞれの規則で、対応する情報をチェックすることにより制御する。

“once:一旦～すると”（cf. “一回～する”）などの呼応表現を導く可能性がある副詞などは、英語における位置的条件などを満たすかなどによって、呼応表現を導くものと導かないものとに写像が分かれる。呼応表現を導くものについては、レキシコンに導かれる呼応表現とその条件とを記述しておく。文法ではそのレキシコンを参照し、呼応表現を導く副詞などが生成される際に、それに対応する呼応表現が現れるべき句に導かれる呼応表現の条件を単一化する。

“～になる”など、補語相当で動詞に密着して現れる句と、その制限を持たない句との語順などは、密着して現れる句に関する条件をレキシコンに記述することにより制御する。文法はそのレキシコンを参照し、密着して現れる句に対応する素性が存在する時は、それ以外の句を生成しないようにして語順を制御する。

これらに対し、トピックや修飾句のスコープなど、英語構造から渡される情報は、主に写像規則によって制限される。

3.2　量詞となる付属語の生成

英語においては、“only”や“also”などは副詞であるが、日本語においては“だけ”や“も”などの助詞として訳出したほうが、自然であることが多い。これを実現するためには、英語の解析において“also”などのスコープを決定し[5]、そのスコープのヘッドの量詞を示す素性に“also”などの情報を写像する。

文法においては、量詞素性を持つ句について、その助詞列を生成する際に、助詞列生成順序に違反しないように量詞の助詞付け加えればよい。ただしその際、“誰も～ない”など、他の素性から導かれる（日本語において同一の）量詞助詞の生成は抑制される。またこういった語は、助詞でなく副詞に対する写像もあるので、原則的に曖昧さを持った素性構造が作成されるが、どちらの素性構造が優先されるかは、基本的に規則の順序に依存する。

3.3　写像規則による表現の制御

英語での修飾句は、例えば連体修飾なら、接頭辞、接尾辞、連体詞、名詞＋連体修飾助詞など、日本語側でさまざまな構文構造を取る。接頭辞、接尾辞は被修飾名詞句に密着して現れ、連体詞など他の連体修飾句は多少離れてもよい。文法は規則が現れる順番に適応されるため、大まかに言って優先されるべき規則が先に書かれている。リストである修飾句は個々が独立して処理されるため、文法規則で他の修飾句を見ながら生成順序を制御することができない。このため、写像規則は、英語での語順から、日本語での語順への修飾句などの必要な語順変換を行なう。この語順変換により、構造的に正しい生成をより早く行なうことが可能である。日本語で必要な語順変更は、英語側の語順よりも優先されるが、必要な変更がない限り、英語側の語順は保存される。

4　効率化と失敗回復

4.1　等式の効率化

本稿の文法は双方向性を持つため、原則的には生成／解析の双方で等式の評価が行なわれる。そのため、生成

時には生成停止性の保証のために、単一化が成功した値を等式 $e5$ などにより空で置き換えることが多いが、解析時にはその値が存在していないことは自明、といった冗長な等式評価が多くなり易い。これを回避するのに、$=g$ による単一化が役立つ。$=g$ は、生成の際には使用されるが、解析の際には自明なため等式を評価する必要はないことを示す。

4.2 効率化のための素性伝搬

"誰が～するか" など、不定詞とカなどの呼応については、通常は、カに対応する素性が処理される時には、不定詞が同じ節の中の、その部分木に現れるか分からない。このため、カが外された時点で素性構造に特別な素性 wh-plus を与えておき、それ以後に処理される、呼応不定詞が現れ得るどの部分木にも、それに対応する素性 wh を与え、呼応不定詞が見つかると wh を単一化し空に置き換える。wh が存在しない値に置き換えることに成功した部分木を 1 以上もつ素性構造において、wh-plus も単一化され空に置き換えられる。こうして wh-plus が空に置き換わったものだけを生成成功とすると、不定詞とカとの呼応生成が成功する。

この方法でもこうした呼応の扱いは可能だが、ほとんど全ての部分木について、wh の単一化→不成功を繰り返すため、非常に処理に負担がかかりやすい。そのため、本稿の処理では、素性構造が作成された時点で、不定詞のレキシコンから得られる上記の wh のような素性は、あらかじめその素性が影響を及ぼす範囲のノードまで上に伝搬させておいている（図3)。これによって、単一化不成功の繰り返しを防ぎ、処理の効率化を図っている。

4.3 未知語と失敗回復モジュール

日本語における未知語は、その概念をストリングにしたものが、変換モジュールによって素性構造中に語幹表記として表される。ただし、英語からの情報によって、「動詞らしい」「形容詞類らしい」などの判別も同時に行なわれるため、「"概念表記" する」「"概念表記" だ」などが、実際に素性構造に渡される表記となる。

効率化のため、実際の処理では、最大ステップ数を定め、そのステップ数を越えたものは単一化失敗と同等と見なし、失敗回復モジュールに素性構造を渡して処理をまかせる。現在、最大ステップ数は、500 に設定されている。

失敗回復モジュールは、大まかな情報を補いながら素性構造の各表記部分を組み立てていく。単一化失敗がこのモジュールに処理が渡される原則的な前提になっているので、単一化や細かい素性の一貫性はチェックせず、単なる日本語ストリングを多少分かりやすい順番で出すための後処理部分として起動される。

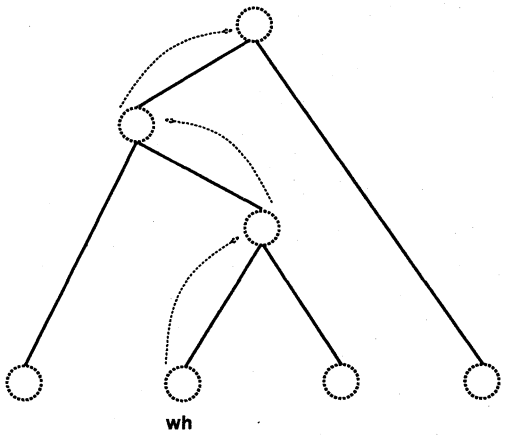

図 3: wh の事前伝搬

5 おわりに

現在開発中の英日機械翻訳システム Shalt2 の日本語生成部について述べた。今後の課題としては、生成文全体を見渡しながらの句読点の制御や、全文処理などを使ったより好ましい表現の選択生成などが残されている。

文献

[1] 南不二男 (1986) 質問文の構造,『朝倉日本語新講座 4 文法と意味 II』朝倉書店, pp. 39 - 74.

[2] 徳永健伸, 乾健太郎 (1991) 1980 年代の自然言語生成,『人工知能学会誌』, vol. 6, no. 3 - 5.

[3] 武田浩一 (1991) 疑似単一化文法の双方向性,『情報処理学会第 42 会全国大会講演論文集』(3), pp. 116 - 117.

[4] 浦本直彦 (1993) 英日機械翻訳システム Shalt2 における並列句の取り扱い,『情報処理学会第 47 会全国大会講演論文集』(3), pp. 177 - 178.

[5] 那須川哲哉 (1995) 頑健な文脈処理を取り入れた機械翻訳,『言語処理学会第 1 会大会講演論文集』.

[6] Peter Sells (1985), "Lectures on Contemporary Syntactic Theories," The University of Chicago Press.

[7] Carl Pollard and Ivan A. Sag (1994), "Head-Driven Phrase Structure Grammar," The University of Chicago Press.