同期文法による文生成

武田 浩一
日本アイ・ビー・エム株式会社 東京基礎研究所
神奈川県大和市下鶴間 1623-14
takeda@trl.ibm.co.jp

1 まえがき

文法を利用した文生成は [19, 15, 10]、機械翻訳や文書要約といった応用のための宣言的な自然言語処理システムを構築する上で非常に重要な技術である。特に、双方向文法と呼ばれる、文の解析と生成に同一の文法を利用するという研究 [14, 4, 8] では、単一化文法を中心とした文の解析・生成のアルゴリズムや諸性質がよく知られている。単一化文法は、解析の高速化手法 [18] も提案されており、実用化への期待が高まっている。

しかし最近の自然言語処理技術は、インターネットにおける情報の収集・検索、加工、発信といった様々な側面で積極的に利用される傾向にあり、ホームページ検索エンジンや電子図書館のような巨大な応用では、何百万件というデータを効率よく処理するという要求が最優先される。このため有限状態トランスデューサ [9, 2] のように、線形時間程度の非常に高速なアルゴリズムで実現可能な処理が本質的な役割を果たすと考えられる。この点で文法による文生成は、実用レベルに達しているとはいいがたい。本論文で示すように、単一化文法を用いた場合には、文生成は一般にNP-完全問題になってしまう。これは、生成の補助情報として、例えば句構造の導出木を指定した場合でも同様の結果となる。

この計算の複雑さの問題に対する解決策としては、文法を効率よく処理できるようなサブクラスに限定したり、厳密な生成を行なうという制限を緩めた近似アルゴリズムを用いる、といったものが考えられる。本論文では、このような方向から、特に文生成を、原言語から目的言語への翻訳に利用する目的に限定して、単一化文法ではなく、文脈自由文法の対から構成される同期文法 (synchronized grammar) という文法の表現手段を用いた文生成について考察する。このような文法を選んだ理由は、計算の複雑さと密接な関係のある単一化と素性構造を排除するとともに、これに代わるものとして、(原言語と目的言語の) 文法規則対からなる文法を用いて、文生成の計算的な実質上の下限を明らかにすることにある。同期文法による原言語の文を解析・翻訳するための計算量は、文法のサイズG、1つの規則に現れる最大の非終端記号の数K、および入力となる文のサイズnに対して$O(G^2n^{2K+1})$となることが示せるため、これを手がかりとして、種々の高速化を考えることが可能である。例えば、解析結果を固定した場合には、文生成は$O(K)$程度になる。

同期文法のファミリーには、STAG(Synchronized Tree-Adjoining Grammar) のような文脈自由文法を真に包含するような文法の対が含まれるだけでなく、原理的には単一化文法の対をも考えることができるため、目的に応じた文法の最適な記述能力を選択することができる。また、原言語の文法の代わりに意味表現や概念表現を記述する文法を用いることで、機械翻訳以外の文生成にも適応できる。

2 文法による文生成

文字列である文と、その言語学的表現として定められた素性構造との関係を記述できるような文法Gを考えると、文法Gによる文生成とは、次のように定義できる：

> 与えられた文法Gのもとで、素性構造f(統語、意味表現といったいかなるものでもよい) が指定されたときに、fと同じ素性構造を導出できる文sを生成すること。これをgen(G,f) = sと書く。

ここで、もしそのようなsが複数個存在するときには、文法は**曖昧**であり、そのようなsが存在しないときには、次の3通りが考えられる。

1. Gは、fの部分的な情報を表現するような文sを生成する。(過小生成)
2. Gは、fを真に含む情報を表現するような文sを生成する。(過大生成)
3. Gは、fと比較不能な情報を表現するような文sを生成する。

文生成の目的は、与えられた情報を正確に表現するような文を求めることであるため、上記の3通りのいずれの場合も望ましくないが、直観的には過小生成は「嘘は言わない」生成、過大生成は「与えられたことは全部

言う」生成であり、意味処理や文脈処理のもとでは、情報が正確に表現できている場合もある。このような現象は、従来の機械翻訳では「トランスファー」処理で暗黙的に処理されていたが[3]、素性構造の差分によって定量的に扱うことも可能である[16]。本論文では、素性構造のような内部表現と、文のような外部表現とのミスマッチの問題は考えず、情報の過不足のない生成のみを考慮する。

いま、文法として、PATR-II[13]のような句構造規則と素性構造の単一化式からなる単一化文法を考える。

定理 1 Gを文法、fを与えられた素性構造とする。このとき、$gen(G,f) = s$を満足する、高々K個の規則からなる文sの生成木が存在するかどうかの決定問題はNP完全である。

証明:高々K個の規則列を推定し、これが定められた生成過程(例えば、トップダウン最左導出)のもとで終端記号列のみを葉に持つ生成木を構成し、かつ与えられた素性構造fを再構成するかどうかの決定は、fに関する多項式時間で実行できる。次に、NP完全問題として知られている「ハミルトン閉路」問題(HC)[7]が、この文生成問題に多項式時間で変換できることが以下のように示せる。

- HCのインスタンスを節点集合Vと枝集合Eからなるグラフ H = (V,E) とし、節点の数をNとする。Hから、次のような文法Gを構成する。
- 各節点$v_i \in V$に対し、Gは

  $$X_{i,i,1} \rightarrow v_i$$
  $$\langle X_{i,i,1}\ v_i \rangle = \text{通過}$$

  という規則を含む。
- 節点$v_i, v_j \in V$を結ぶ枝$e_{i,j} = \langle v_i, v_j \rangle \in E$に対し、Gは、

  $$X_{m,j,k+1} \rightarrow X_{m,i,k}$$
  $$\langle X_{m,j,k+1} \rangle = \langle X_{m,i,k} \rangle$$
  $$\langle X_{m,j,k+1}\ v_j \rangle = \text{通過}$$

  および

  $$S \rightarrow X_{j,i,N}$$
  $$\langle S \rangle = \langle X_{j,i,N} \rangle$$

  という規則を含む。

上記の文法から直ちに、Gの任意の文を生成する導出規則はN個の句構造規則と、1つの字句規則からなる鎖になることが示せる。今、素性構造としてv_i = 通過 $(i = 1, \ldots, N)$を与えると、Hがハミルトン閉路を持つ時、またその時のみGはある終端記号$v_i 1$文字からなる文を生成する。[証明終わり]

従って、単一化文法のもとでは、一般的に素性構造から効率よく文を生成するアルゴリズムが得られそうにないと考えられる。この単一化文法に本質的な計算の複雑さは、たとえ素性構造以外の補助的な情報を用いても改善しない。

定理 2 文法Gを、その句構造規則が字句規則か、すべてが非終端記号からなる規則からなるものとする。いま、Gの生成木から、具体的な単一化の等式と字句規則に関する情報を取り除いた部分生成木Dが与えられたとき、GがDの各句構造規則に実際の規則を割り当て、さらに字句規則を加えて妥当な生成木を構成できるかどうかを判定する問題はNP完全である。

略証:Dに字句規則を含むGの実際の規則を推定し、それが妥当な生成木かどうかを判定するのはNPであることがわかる。次に、やはり既知のNP完全問題である「3充足問題」(3SAT)[7]が多項式時間で単一化文法Gに変換でき、Gが、ある自明な部分生成木Dに対し、妥当な生成木を持つことと、もとの3SATが充足可能なことが等価であることが示せる。

定理 3 文法Gに部分生成木Dと素性構造fが与えられたとき、GがDの各句構造規則に実際の規則と字句規則を割り当て、fを再構成するような妥当な生成木を構成できるかどうかを判定する問題はNP完全である。

略証:この問題がNPであることは先ほどと同様の議論から示すことができる。この問題も、ハミルトン閉路問題から多項式時間であるGに変換でき、Gが、ある自明な部分生成木Dと素性構造fに対し妥当な生成木を持つことと、ハミルトン閉路が存在することが等価であることが示せる。

このような結果から、単一化文法を用いた文生成は最悪の場合には極めて効率の悪い処理となる。計算時間を改善するためには、単一化文法に何らかの制約を加えたクラスを考えるか、前述したような過小生成や過大生成を許した、近似生成アルゴリズムを用いる必要がある。

3 同期文法による文生成

単一化文法よりも表現能力の低い文法を利用するとしても、文脈自由文法は素性構造を持たないために不適切である。前節の部分生成木のような構造も、文脈自由文法に対しては、生成の問題が自明になるか、または非常に抽象的な指定になるため、実用的ではない。このた

め、単一化に変わる表現能力を実現する手段として、原言語と目的言語との対応を、文脈自由文法の規則の対で表現する同期文法 [17] を考える。同期文法では、句構造規則が素性構造を記述するかわりに、直接それに対応する目的言語の句構造規則を記述する。例えば、

$S \leftarrow NP_1$ amaze NP_2,
$S \leftarrow NP_2$ は NP_1 を 驚かす

という規則対は、英語の「X amazes Y」という文と「Y は X を 驚かす」という文が対応することを示す。NP_1 や NP_2 は非終端記号で、その添字が原言語と目的言語の文法規則で対応する非終端記号であることを示す。(規則対で必ずしも同じ非終端記号を用いる必要はない。)

同期文法による文の生成は、もともと Aho,Ullman らの「トランスデューサ」による文の翻訳の概念に遡ることができる [2]。最近では、STAG(Synchronized Tree-Adjoining Grammar)[15] による、単一化文法を用いない意味構造の記述や、機械翻訳への応用 [1] が知られている。同期文法では、原理的には任意の表現能力を持つ文法を対にすることが可能であるが、その諸性質は文脈自由文法や LTAG(Lexicalized Tree-Adjoining Grammar) と呼ばれる文法の規則対をもつ同期文法に関してよく知られている [2, 11]。

同期文法では、原言語の文を解析すると同時に、対応する規則対を用いて目的言語の文を生成することができる。この処理に要する計算時間は、与えられた同期文法 G の規則対の数 $|G|$, G の 1 つの規則に含まれる最大の非終端記号の数 K と、入力となる文の長さ n に関する多項式でおさえられる。

定理 4 $G = \{\langle r_L, r_R \rangle\}$ を文脈自由文法の規則対からなる同期文法とし、K を、*1* つの規則 r_L または r_R に含まれる最大の非終端記号数とする。このとき G に対する *Earley* 方式 *[5]* の解析／生成アルゴリズムが存在し、その計算時間は $O(|G|^2 n^{2K+1})$ となる。

略証: 文脈自由文法に対する Earley パーザを拡張して、同期文法に対する解析／生成を実行するアルゴリズムを得ることができる。与えられた原言語の文に対する可能なすべての規則の具体化 (instantiation) は高々 n^{K+1} 個しか存在しえないため、Earley パーザ部分の隣接する 2 つの具体化された規則の completion 処理は、高々 $O(|G|^2 n^{2K+1})$ 回しか実行されない。1 つの原言語の規則が具体化されたときに、その規則対で対応する目的言語の規則を具体化するための計算時間は $O(K)$ なので、全体として上記のオーダの計算時間でおさえられる。

Shabes らによって、TIG(Tree Insertion Grammar) と呼ばれる文脈自由文法に強等価な文法に対する Earley パーザがが存在することが示されている [12] ので、上記の定理は、限定された Tree-Adjoining Grammar についても成立することがわかる。

上記のアルゴリズムは、解析と生成を同時に処理するナイーブなものであるため、

- $O(|G|^2 n^{2K+1})$ という非常に効率の悪い計算になる。
- 通常生成したい文の候補数よりもはるかに多い数の文を生成する可能性がある。
- 解析に曖昧性があったり、優先的な解釈や生成をしたいときに、これを処理する機構が含まれない。

といった問題がある。

この問題を解決する方法の 1 つは、解析と生成の同時実行をやめ、既存の文脈自由文法による文解析の手法を採用し、解析結果が得られた場合に、この解析結果に規則対の目的言語側の規則を適用し、文を生成するという方法をとることである。解析木 T が与えられた時に、目的言語の規則を具体化する計算は $O(KT)$ 時間で行なえる。文脈自由文法の解析は $O(|G|n^3)$ 時間で実行できることが知られているので、この方法によって文の解析と生成を非常に効率よく実行できる。ただし、与えられた解析木から必ず文を生成できるという保証がないため、この方法は「健全」ではあるが「完全」なアルゴリズムとはいえない。解析で上位 m 個の解析木を用いるとしても同様である。

実用性の観点からすると $O(|G|n^3)$ 程度の計算時間は限界であり、例えば、インターネットの検索エンジンから得られた外国語のホームページのリストから、その最初のページまたは概要を翻訳して表示する、といった処理には完全性を犠牲にしても、上記の高速な方法を選ばざるをえない。このため、ヒューリスティックな生成失敗回復機構を組み合わせて、実用的な処理を実現する必要がある。この他に、上記の方法を改良するものとして、

- 文法を事前にコンパイルして左辺が同じ規則対をまとめて処理する
- 文法規則に、半順序を定義し、例えば字句化 (lexicalized) された規則を純粋に非終端記号のみからなる規則よりも優先的に適用する
- 各規則に確率やコストを対応させ、優先処理を実現する

といったことが、解析と生成の両方の規則について考えることができる。このような方法は、古瀬らのパターンによる翻訳 [6] や、Earley パーザが利用できる文法なら、同様に適用できる。また翻訳対の原言語文法だけを、よく知られている実用的な文法にして、目的言語の生成処理を、素性構造を通してではなく、規則対による記述で表現することにより実現することができる。

4 あとがき

前節の方法を、性数一致の記述や、若干の意味素性を加えて拡張した文脈自由文法を用いて実験した。規則対数が約 2000 程度の同期文法で、1 文平均 12 語の英文を日本語に翻訳するための計算時間は大体 0.8 秒程度で、良好な結果を得た。翻訳結果も 60%程度は理解できるものであった。

インターネットにおける情報検索のように、今後の巨大なテキスト処理を含む応用には、単純な有限状態トランスデューサでは満足できない自然言語処理を必要とすることが増えると考えられ、これを解決するために、高速のアルゴリズムの研究がさらに重視されると思われる。同期文法のような枠組は、有限状態トランスデューサと、単一化文法理論の中間にあたり、効率と処理能力の実用的に最適な組合せを提供する役割を果たすものとして期待できる。

謝辞

日頃御討論いただく東京基礎研究所の研究員諸氏に感謝いたします。

参考文献

[1] A. Abeillé, Y. Schabes, and A. K. Joshi. "Using Lexicalized Tags for Machine Translation". In *Proc. of the 13th International Conference on Computational Linguistics*, volume 3, pages 1–6, Aug 1990.

[2] A. V. Aho and J. D. Ullman. *"The Theory of Parsing, Translation, and Compiling"*, volume I: Parsing. Prentice-Hall, Englewood Cliffs, New Jersey, 1972.

[3] B. J. Dorr. *"Machine Translation: A View from the Lexicon"*. The MIT Press, Cambridge, Mass., 1993.

[4] M. Dymetman. "Inherently Reversible Grammars, Logic Programming and Computability". In *Proc. of ACL Workshop on Reversible Grammar in Natural Language Processing*, pages 20–30, Berkeley, California, June 1991.

[5] J. Earley. "An Efficient Context-free Parsing Algorithm". *Communications of the ACM*, 6(8):94–102, February 1970.

[6] O. Furuse and H. Iida. "Cooperation between Transfer and Analysis in Example-Based Framework". In *Proc. of the 14th International Conference on Computational Linguistics*, pages 645–651, Aug. 1992.

[7] M. Garey and D. Johnson. *"Computers and Intractability"*. W.H.Freeman and Co., San Francisco, 1979.

[8] K. Hasida. "Common Heuristics for Parsing, Generation, and Whatever, ...". In *Proc. of a Workshop on Reversible Grammar in Natural Language Processing*, pages 81–90, June 1991.

[9] L. Karttunen. "Constructing Lexical Transducers". In *15th International Conference on Computational Linguistics*, pages 406–411, Kyoto, Aug. 1994.

[10] M. Martinovic and T. Strzalkowski. "Comparing Two Grammar-Based Generation Algorithms: A Case Study". In *Proc. of the 30th Annual Meeting of ACL*, pages 81–88, June 1992.

[11] O. Rambow and G. Satta. "Synchronous Models of Language". In *Proc. of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 116–123, San Diego, Jun. 1996.

[12] Y. Schabes and R. C. Waters. "Tree Insertion Grammar: A Cubic-Time, Parsable Formalism that Lexicalizes Context-Free Grammar without Changing the Trees Produced". *Computational Linguistics*, 21(4):479–513, Dec. 1995.

[13] S. M. Shieber. *"An Introduction to Unification-Based Approaches to Grammar"*. CSLI Lecture Notes, Number 4, Stanford, CA, 1986.

[14] S. M. Shieber. "A Uniform Architecture for Parsing and Generation". In *Proc. of the 12th International Conference on Computational Linguistics*, pages 614–619, August 1988.

[15] S. M. Shieber and Y. Schabes. "Synchronous Tree-Adjoining Grammars". In *Proc. of the 13th International Conference on Computational Linguistics*, pages 253–258, August 1990.

[16] K. Takeda. "Tricolor DAGs for Machine Translation". In *Proc. of the 32nd Annual Meeting of ACL*, pages 226–233, Las Cruces, New Mexico, June 1994.

[17] K. Takeda. "Pattern-Based Context-Free Grammars for Machine Translation". In *Proc. of the 34th Annual Meeting of ACL*, Santa Cruz, Calif., June 1996.

[18] K. Torisawa and J. Tsujii. "Computing Phrasal-Signs in HPSG Prior to Parsing". In *Proc. of the 16th International Conference on Computational Linguistics*, pages 949–955, Aug. 1996.

[19] J. Wedekind. "Generation as Structure Driven Derivation". In *Proc. of the 12th International Conference on Computational Linguistics*, pages 732–737, August 1988.