不完全な構造変換規則による言い換えの実現方法

高橋哲朗[*1] 岩倉友哉[*2] 乾健太郎[*2*3]
[*1] 九州工業大学院情報工学研究科情報科学専攻
[*2] 九州工業大学情報工学部知能情報工学科
[*3] 科学技術振興事業団さきがけ研究21
{t_taka, t_iwa, inui}@pluto.ai.kyutech.ac.jp

1 はじめに

テキストの言い換えは，様々な自然言語処理の応用分野に貢献する重要な技術であり[15]，近年これを応用からある程度独立した要素技術として捉え，問題の性質や実現方法を解明する試みも見られるようになってきた[13, 15, 6, 7].

佐藤の指摘[15]にもあるように，言い換えは，表層のテキストから表層のテキストへ意味をできるだけ変えないように変換するという点で，テキストの翻訳の一種と見なすことができる．したがって，その実現方法は，従来から広く研究されてきた異言語間機械翻訳，とくに構造変換（トランスファ）方式の翻訳に範を求めればよいように見える．しかし，単言語内翻訳である言い換えには，(a) 一つの単語の言い換えから文の分割・併合まで，一つひとつの構造変換がそれ単体で一つの言い換えを構成し得る，(b) 多くの場合，原文の大部分が言い換え後も保存される，など異言語間翻訳にはない性質もあり，それらを踏まえた実現方法の議論が必要であると考えられる．

本稿では，言い換えと異言語間翻訳を対比させながら，言い換えの実現に適したアーキテクチャについて論じ，プロトタイプシステムの開発の現状を報告する．

2 構造変換方式の翻訳と言い換え

構造変換方式については，図1に示すように，構文レベル（あるいはより表層に近いレベル）で変換するものから意味レベル（あるいはより抽象的なレベル）で変換するものまで幅広い選択肢があり，それぞれの長所・短所が議論されてきた（e.g. [1, 10]）．それらは次のように要約できる:

- **構造変換の負荷：** 変換レベルの抽象度が（構文レベルから意味レベルの方に）上がるほど個別の言語に固有な構造情報が捨象されるので，変換の負荷は軽くなると期待できる．すなわち，必要な変換規則の数が減り，その開発保守のコストも軽くなると考えられる．ただし，変換レベルの抽象化が進むと，中間表現のオントロジの抽象度も上がるため，逆にオントロジの定義や理解が困難になり，多人数で規則を開発することが難しくなる恐れもある．
- **解析の負荷：** 変換レベルの抽象度が上がると，解析の負荷は重くなる．意味解析は構文解析に比べて未解明の問題が山積しており，意味的な曖昧性を無理に解消しようとすると，解析誤りのリスクが高くなる．また，意味解析の計算コストも無視できない可能性がある．意味解析部の開発コストも考慮すべきである．

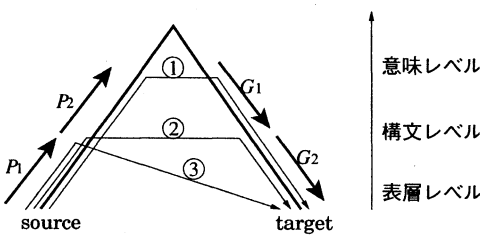

図 1: 構造変換方式の翻訳

これらの議論を単言語内翻訳である言い換えに置き換えると，まず意味構造変換については，以下の理由から言い換えの実現方式として適当でないと考えられる．

- 冒頭で述べたように，言い換えでは，原文の一部を変換するだけでそれ自体が一つの言い換えを構成し得るため，原文の大部分が言い換え後も保存される．このため，個々の言い換えでは，文全体の意味情報のごく一部しか参照せず，その他の大部分の意味情報は利用されない．したがって，構造変換の前に文全体の意味解析を行う意味構造変換方式では，意味解析のオーバヘッドの方が高くなると考えられる．
- 変換レベルの抽象化は，もともと個別言語に特有な構造情報を捨象することが目的であった．しかし，テキストの簡単化や要約，推敲支援といった言い換えのアプリケーション（e.g. [2, 4, 5]）では，態の変換や文の分割のように，原文の構文構造を特定の構造に変換すること自体が言い換えの動機になる場合も多く，表層や構文の情報を過度に捨象する抽象化は望ましくない．

以上より，構造変換のレベルとしては，(必要に応じて意味情報が注釈づけされた）構文構造のレベル，あるいはそれより抽象度の低いレベルを言い換えごとに選択できることが望ましいと考えられる．

一方，構文構造変換のように変換レベルの抽象度を下げると，前述のように，必要な変換規則（言い換えパタン）の数が組み合わせ的に増える恐れがある．Mu システム[11]などに代表される構文構造変換方式のシステムでは，様々な前処理や後処理を導入することによって

構造変換の負荷を軽減する工夫を試みているが，意味構造変換の場合と比較すると構造変換の負荷が大きいことに変わりはない．変換レベルを過度に抽象化することなく，必要な規則数をさらにドラスティックに抑える何らかの工夫が必要である．

3 生成に基づく言い換え

前節の考察を踏まえ，次のような特徴を持つ言い換えのアーキテクチャを提案する（図2参照）．

- 構造変換のレベルは，意味情報注釈つき構文構造のレベルよりも抽象化しない．
- 構造変換規則の数を抑えるために，規則に非決定性を持たせるとともに，不完全 (underspecified) な規則の記述を許す．
- 変換規則の非決定性と不完全性によって生じる不適格な変換結果は，変換後に言語モデルに基づいて変換候補を選択・修正することによって吸収する．
- 構造変換あるいは選択・修正に必要な構文・意味情報は，必要になった時点でオンデマンドに収集し，中間表現に追加する．

既存の言い換え方式の多く (e.g. [13, 15, 16]) は，構文構造変換に基づいており，言い換えに伴う形態素レベルの修正や語の共起の制約などを変換規則に組み込んでいる．このため，精密な言い換えを行おうとすると，変換規則が組み合わせ的に増大する危険性がある．とくに言い換えでは，変換前と変換後の中間構造のオントロジが共通であり，図1の③のようにソース側の中間構造を直接ターゲット側の文字列に変換する規則も自然に書けてしまうため，異言語間翻訳に比べてその傾向は強い．これに対し，我々のアーキテクチャでは，後述するように②のレベルで構造変換を行いながらも，①と②の抽象度の違いを G_2 の生成過程（変換候補の選択と修正）で吸収することが可能である．

以下本節では，3.1で変換規則と言語モデルの役割分担について論じ，3.2で言語モデルで扱うべき言語知識の例を示す．

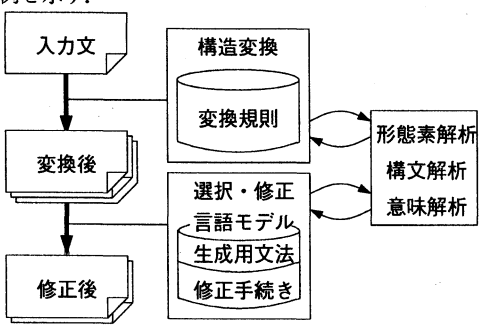

図 2: 言語モデルに基づく言い換えのアーキテクチャ

3.1 言い換え知識＝変換規則＋言語モデル

我々のアーキテクチャの最も重要な特徴は，言い換えのための知識を次の二つに分割して記述する点である:

変換規則: 個々の構造変換に固有の知識

言語モデル: 構造変換とは独立な静的な言語的制約（生成用文法），および非文の修復に用いる修正手続き

例として，「～しか～ない」という否定表現の言い換えに必要な知識を考えよう．

「～しか～ない」については，(1) や (2) のような言い換えパタンが考えられる．

(1) A しか B しない ⟶ B するのは A だけだ

(2) A しか B しない ⟶ B するのはせいぜい A だ

しかし，実際に (3) や (4) のような言い換えを実現するには，少なくとも規則 (5) や (6) のように，規則適用の条件づけや形態的な変形といった様々な知識を上の言い換えパタンに追加する必要がある．

(3) a. 彼はベジタリアンだったので，パーティーでも野菜しか食べなかった．
b. 彼はベジタリアンだったので，パーティーでも食べたのは野菜だけだった．

(4) a. ジーンズと T シャツくらいの簡単な服装しか持ってこなかった．
b. 持ってきたのはせいぜいジーンズと T シャツくらいの簡単な服装だ．

(5)
a. A しか B ない ⟶ B' のは A だけだ
b. B': B を連体形に変換
c. B': 言い換え前の「ない」の時制が過去なら，B を過去にする

(6)
a. A しか B ない ⟶ B のはせいぜい A だ
b. B': B を連体形に変換
c. B': 言い換え前の「ない」の時制が過去なら，B を過去にする

実際には，これらの規則はまだ記述が十分でないかもしれない．たとえば，規則 (6) を (3a) に適用すると，(7) のように談話レベルで不適格な文が出力されるので，規則 (6) はさらに適用条件の詳細化が必要である．

(7) *彼はベジタリアンだったので，パーティーで食べたのはせいぜい野菜だ．

このように，精密な言い換えを実現しようとすると，変換規則の中に細かい条件づけや制約を組み込むことになり，規則が組み合わせ的に複雑になる恐れがある．

ここで注意したいのは，規則 (5) と (6) で b と c の記述が重複している点である．これらの記述はいずれも，構造変換に依存しない静的な言語的制約，あるいはそのような制約が満たされない場合に実行すべき修正方法を記述したものと見なせる．このような知識を変換規則に組み込むことは，変換規則の組み合わせ的増大を招く恐れがあり，望ましくない．

この問題は，たとえば規則 (5), (6) の知識を二つに分け，a の知識を変換規則として記述し，b や c の知識

を言語モデルとして記述することによって大幅に解消できると考えられる．変換規則は条件づけや変換手続きの記述が不完全 (underspecified) なため，必ずしも規則適用後に適格な構造が得られるとは限らない．そこで，規則適用後に残る問題（不適格性）は言語モデルで修正することによって吸収する．このとき，言語モデルによる修正が不可能な場合は，その変換候補を棄却する．構造変換後に候補の棄却を許すためには，変換規則に非決定性を導入し，構造変換で複数の変換候補を生成できるようにすればよい．

ここで，変換規則と言語モデルの棲み分けをこれ以上厳密に議論するのはあまり有益でない．むしろ，両者の間にある程度の記述の重複を許す方が，規則開発の負担を効果的に削減できると考えられる．

3.2 言語モデル

言語モデルが扱うべき問題の例を以下に挙げる．

構文レベルの問題

- **品詞連接・活用形:**
  最も基本的な言語的制約の一つに品詞連接や活用形に関する制約がある．言語モデルは，これらの制約について変換後の構造を調査し，活用形を修正したり，修正不可能な候補を棄却したりする．たとえば，前述の例 (3a) に変換規則 (1) を適用すると，次の構造が出力される．
  (8) 彼はベジタリアンだったので，パーティーでも食べの は野菜だけだ．
  下線部では，動詞未然形「食べ」が形式名詞「の」に連接しているが，品詞連接制約に修正手続きを埋め込んでおけば，正しい活用形に修正することができる．
- **主題化の問題:**
  変換規則 (1) を用いて (9a) を変換する場合，変換結果の (9b) は (9c) に修正しなければならない．一方，(10a) を変換する場合は修正は不要である．この種の修正を実現するには，主題化に関する様々な制約を言語モデル上に実装しておくことが必要である．
  (9) **a** 彼はりんごしか食べない．
  b *彼 は 食べるのはりんごだけだ．
  c 彼 が 食べるのはりんごだけだ．
  (10) **a** 彼はりんごしか食べないと言った．
  b 彼 は 食べるのはりんごだけだと言った．

意味レベルの問題

- **副詞と述語の呼応:**
  (11) の規則は，(12) には適用できるが，(13) には適用できない．
  (11) A しか V ない ⟶ A だけ V
  (12) **a** りんごしか食べていない．
  b りんごだけ食べている．
  (13) **a** まだりんごしか食べていない．
  b まだりんごだけ食べている．
  これは，副詞「まだ」が否定文に現れる場合と肯定文に現れる場合で意味が異なるからである．この種の副詞が変換後の文に出現した場合，変換前後で肯定／否定の転換が起っていれば，変換後の文を棄却する必要がある．このような知識は言語モデルも言語モデルで扱うべきである．
  また,「あまり，さほど，たいして，全然，全く，さっぱり，少しも，ちっとも」などの副詞は否定文としか呼応しない．よってこれらを言語モデルに加えることで，これらの副詞が肯定文中に現れた場合に非文として棄却することができるようになる．
- **意味クラスに基づく語選択:**
  副助詞「も」に関する言い換えを行う場合,「も」に係る名詞の種類により変換する候補が (14) と (15) のように異なる．名詞の意味クラスと，それに対応する語の知識が必要である．
  (14) 一つも食べない．⟶ 何も食べない．
  (15) 一人も食べない．⟶ 誰も食べない．
- **副助詞の格の兼務:**
  変換規則 (16) を用いて (17a) を変換した場合，(17b) は非文ではないが，(17c) のように副助詞「しか」の兼務する格助詞を補完した方がより自然な言い換えができる．
  (16) A しか V したことがない ⟶ A だけ V したことがある
  (17) **a** 風邪ぐらいの軽い病気しかかかったことがない．
  b ?かぜぐらいの軽い病気だけかかったことがある．
  c かぜぐらいの軽い病気だけ に かかったことがある．
  どの格助詞を補完すべきかは，(16) の A と V の格関係に依存することに注意されたい．
- **文内の意味のつながり:**
  (18) を用いて (19) を (20) に変換することはできる．
  (18) A しかいない ⟶ A こそふさわしい
  (19) こんな場面では，彼しかいなかった．
  (20) こんな場面では，彼こそふさわしかった．
  しかし，(21) を (22) に変換することはできない．
  (21) 教室に入ると，彼しかいなかった．
  (22) *教室に入ると，彼こそふさわしかった．

談話レベルの問題

- **修辞的関係:**
  (23) を (24) に言い換えた場合，修辞的な問題から (24) は不自然な文となってしまう．
  (23) 朝から晩まで働いたのに，3000 円しかくれなかった．
  (24) *朝から晩まで働いたのに，3000 円だけはくれた．

4 実装

前節で提案したアーキテクチャに基づき言い換えを行うシステムを実装した．実装には，型付き素性構造を扱える論理型プログラミング言語 LiLFeS[9] を用いた．これまでに試験的に実装した知識を表 1 に示す．

表 1: 今回実装した知識

変換規則：	「～しか～ない」に関する規則 14 個
言語モデル：(知識)	活用形辞書，連接テーブル，シソーラス，格フレーム辞書，
言語モデル：(手続き)	連接・係り受けによる活用チェック，格補完，文末の修正，時制修正，連接共起チェック

このシステムの有用性を確認するために，まず「～しか～ない」を含む否定文について 14 個の変換規則を作成した．これらはそれぞれ異なる表現に言い換える規則であり，条件づけなどの詳細化によって規則数が増えている訳ではない．次に，京大コーパス [8] の一部と日本語表現文型例文集 [14] から「～しか～ない」を含む文を 63 文収集し，これらに対して上述の変換規則を適用した．

変換規則と入力文の組み合わせ 882 件（63 文× 14 規則）のうち，260 件が変換規則にマッチした．その結果を表 2 に示す．表 2 から分かるように，変換後の文のうち約 68%を適格文として出力できた．またそのうち約 63%は言語モデルによって正しく修正された文であり，言語モデルの有効性を示している．不適格文についても，表 2 中で不適格の原因となっている処理の精度を上げることによって正しく修正できるようになると期待できる．今回の予備実験では，修正に必要な意味解析をヒューリスティックに基づく簡単な処理だけで行ったが，本システムでは個々の意味解析をモジュール化して組み込めるので，現在のモジュールをより高精度のモジュールに容易に組み変えることができる．

表 2: 実装結果

修正	適格/不適格	不適格の要因	260
あり	適格	—	110
	不適格	格補完	3
		修正規則不備	14
なし	適格	—	66
	不適格	修辞的問題	38
		意味による語選択	26
		副詞と述語の呼応	3

5 おわりに

言い換え知識の設計方法に重点を置き，不完全な変換規則とそれを補う言語モデルを用いた言い換えのアーキテクチャを提案した．このアーキテクチャを単一化システム上に試験的に実装し，例題として否定に関する変換規則を載せたところ，比較的少ない数の不完全な変換規則で多様な言い換えを行うことができることを確認できた．

言語モデルを洗練・拡張するに当たっては，既存の言い換え知識 [6, 7, 15, 12, 3] を要素技術としてこれに組み込むことができると考えられる．今後はまずこれらの知識を整理し実装する予定である．また，今回の実験では問題を限定するために，係り受け解析の結果のうち，一位の候補だけを用いた．構文解析や意味解析における曖昧性を並列に保存したまま効率的に変換・修正を行う手法についても今後検討する必要がある．

参考文献

[1] Buschbeck-Wolf B. and C. Tschernitschek. What you always wanted to know about semantic transfer. *B.Buschbeck-Wolf and C.Tschernitschek. What you always wanted to know about semantic transfer. Verbmobil-report 114, IBM Deutschland Informationssysteme GmbH, Institut fr Logik und Linguistik*, 1996.

[2] Dras, M. Reluctant paraphrase: Textual Restructuring under optimization model. *Proc. of PACLING*, pp. 98–104, 1997.

[3] 蓮井洋志，川口湊，小倉久和．科学技術系論文における付属連鎖の統語的，意味的な誤りの検出方法．情報処理学会論文誌, Vol.37, No.11 1928-1940, 1996.

[4] 乾健太郎．テキスト簡単化における聾者向け読解支援-現状と展望-．電子情報通信学会福祉情報工学研究会，WIT00-34，2000.

[5] 片岡明，増山繁，山本和英．要約のための連体修飾節の“A の B”への言い換え．自然言語処理研究会報告書, NL-133-7, pp. 37–44, 1999.

[6] 近藤恵子，佐藤理史，奥村学．「サ変動詞＋する」から動詞相当句への言い換え．情報処理学会論文誌 Vol.40,No.11,pp.4064-4074，1999.

[7] 近藤恵子，佐藤理史，奥村学．格変換による単文の言い換え．情報処理学会自然言語処理研究会，NL-135-16，pp.119-126，2000.

[8] 黒橋禎夫，長尾 眞．京都大学テキストコーパス・プロジェクト．言語処理学会第 3 回年次大会発表論文集, pp.115–118, 1997.

[9] MAKINO T., K.TORISAWA and J.TSUJII LiLFeS - Practical Programming Language For Typed Feature Structures. *In the Proceedings of Natural Language Pacific Rim Symposium '97*, 1997.

[10] Dorna M., A.Frank, J.Genabith and M.Emele. Syntactic and Semantic Transfer with F-Structures. *COLING-ACL'98.*, 1998.

[11] 長尾真．画像と言語の認識工学．コロナ社，1989.

[12] 西田 元樹，松本 裕治．テンス・アスペクトを考慮した現代日本語の複文の生成．情報処理学会研究報告, 96-NL-116, pp.151-156, 1996.

[13] 野上優，藤田篤，乾健太郎．文分割による連体修飾節の言い換え．自然言語処理学会第 6 回年次大会 pp.215-218，1999.

[14] 坂本正．学習者の発想による日本語表現文型例文集．凡人社，1996.

[15] 佐藤理史．論文表題を言い換える．情報処理学会論文誌，Vol.40,No.7,pp.2937-2945,1999.

[16] 山崎邦子，三上真，増山繁，中川聖一．聴覚障害用字幕生成のための言い替えによるニュース文要約．言語処理学会第 4 回年次大会論文集, pp646-649，1998.