デモ－１９

# 統一的かつ宣言的知識記述に基づく言い換えエンジン*

岩倉友哉† 高橋哲朗† 飯田龍‡ 乾健太郎‡

九州工業大学大学院情報工学研究科情報科学専攻† 九州工業大学情報工学部知能情報工学科‡

## 1 はじめに

言い換えは，ある言語表現を意味を保持したまま同一言語内の別の表現に変換する作業である．言い換えを自動化する技術は，自然言語処理の様々なアプリケーションに応用できる可能性がある重要な要素技術であり，近年研究者の関心も高まっている [1]．言い換えはテキストの翻訳の一種と見なせるので，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るのが自然であろう．ただし，同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) したがって，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る．また，(c) 態の変換や文の分割のように原文の統語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，構造変換において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．これらの性質から言い換えは，表層に近い統語レベルの構造変換で実現することが望ましいと考えられる．ただし，その場合，構造変換で表層情報を扱っても変換規則が過度に複雑にならないようにする工夫が必要である．

このような背景から我々は，言い換えの実現方法について考察し，文献 [8, 3] で以下の特徴を持つアーキテクチャ（図 1 参照）を提案した．

- 構造変換規則の複雑さを抑えるために，変換規則に非決定性を持たせるとともに，適用条件や変換処理の記述が不完全 (underspecified) な規則を許す．
- 不完全な変換規則の適用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」の知識に基づいて修正，棄却，あるいはランクづけすることによって吸収する．
- 入力の意味・文脈情報は，必要になった時点でオンデマンドに解析・収集する．

このアーキテクチャの利点は次のように要約できる．

- 構造変換時に表層・統語情報を捨象しないので，上述のように言い換えの性質に適合している．
- 従来の構造変換方式による機械翻訳と対比させると，我々の方式は目的言語側の生成処理あるいは後編集処理の役割を大幅に拡大し，構造変換の負荷を極小化することに相当する．変換規則に不完全さを許すので，規則開発のコストを削減できるだけでなく，言い換え事例から帰納的に獲得した不完全な変換知識も直接的に利用することができる．
- 変換後の修正・棄却のための知識の多くは，言語が静的に持つ言語的適格性の制約であり，個々の構造変換には依存しない．これを変換規則とは独立に開発・保守することにより，知識のモジュール性が増し，同じ知識が様々な言い換えで利用可能になると期待できる．このことにより，システム全体の知識開発のコストを抑えることができる．
- 意味・文脈解析処理をデータに依存して駆動することにより，「いつどの部分の何に関する意味・文脈解析をするか」という制御を自然に最適化することができる．

本稿では，上述のアーキテクチャの実装方法に焦点を当て，我々が開発した言い換えエンジン KURA の概要を報告する．KURA は主として以下の特長を持つ．

(a) 変換モデルおよび言語モデルからなる言い換え知識を統一的かつ宣言的に記述できる．
(b) 大規模な言い換え実験に耐えるように高速化の工夫が施されている．
(c) 言い換え知識の開発を効率化する環境を提供する．

以下，(a)，(b) についてそれぞれ 2 節，3 節で述べる．

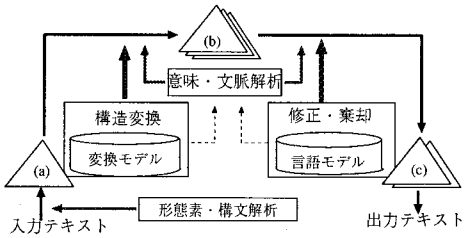


図 1: 処理の流れ (revision-based paraphrasing)

## 2 言い換え知識の記述方法

本節では，依存構造木のデータ構造について述べた後，規則記述の統一性（2.2 項）と規則表現階層（2.3 項）について述べる．

### 2.1 データ構造

図 1 に示唆したように，KURA では，(a) 形態素・構文解析の結果，(b) 変換規則適用結果，(c) 言語モデル適用結果，を共通のデータ構造を用いて表現する．現在用いているデータ構造は，形態素を節点とする依存構造木であり，これを型つき素性構造で表現する．このデータ構造では，統語情報の他，必要に応じて種々の意味・文脈情報を素性として注釈づけすることができる．

機械翻訳における多くの研究 [7, 5] が示すように，依存構造は句構造に比べて文の構成素の移動や置換を行いやすいという利点がある．依存構造の単位については，文節ベースでは書き換えの単位として粒度が粗すぎ，表層上の細かい変換・修正の扱いが困難になると考えたため形態素レベルを採用した．

### 2.2 構造変換/修正・棄却規則の統一的記述

KURA の開発当初は，言語モデルによる修正・棄却は必要な知識をいくつかの専門家モジュールに分割して手続き的に実装し，黒板モデルで実現することを考えていた．しかし，文献 [8, 3] で報告したアプローチで言い換え実験を進めた結果，修正・棄却に必要な知識の大部分は，依存構造の書き換え規則として宣言的に記述できることが明らかになってきた．この知見に基づき，構造変換規則と修正・棄却規則をともに依存構造書き換え規則として統一的に記述する環境を構築した．

構造変換過程では，一つひとつの規則が言い換えの選択肢に対応し，それぞれ別々の言い換え候補を生成する．一方，修正・棄却過程では，言語モデルを構成する規則集合全体が言語的適格性の制約に相当し，構造変換で生成された個々の言い換え候補に対し，これらの規則集合全体を適用することによってそれぞれ最終解を得る．

### 2.3 知識表現の階層化

個々の書き替え規則の表現として，図 2 のような抽象度の異なる 3 つの階層（知識表現階層）を用意した．ユーザは任意のレベルで，規則を記述することができる．①から

---

*KURA: A Paraphrasing Engine Based on Uniform and Declarative Knowledge Representation

Iwakura Tomoya†, Takahashi Tetsuro†, Iida Ryu† and Inui Kentaro†
† ‡Kyushu Institute of Technology
{t_iwa,t_taka,r_iida,inui}@pluto.ai.kyutech.ac.jp

②，②から③ への変換は，トランスレータ，コンパイラが自動的に行う．以下，各記述レベルについて説明する．

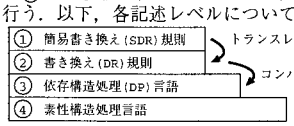


図 2: 規則の変換の階層

**簡易依存木書き換え (SDR) 規則**：図 3 のような表層に近い形で規則の記述ができるように設計されている．

SDR(Simplified Dependency tree Rewriting) 規則では，形態素を**変数**として使用でき，形態素・意味情報も指定できる．図 3 ($R1$) の N は名詞，V は動詞を意味する*1．($R2$) の V と “ない” の間にはアスペクト，ヴォイス等を意味する形態素列が存在する場合がある．これらに対しては，**可変長変数**を用いることができる．ユーザは挿入する可変長変数の種類を指定でき，指定がない場合は，トランスレータが自動的に挿入する．可変長変数は図 4 のような形態素列の正規表現である．依存構造は，($R2$) のように “[”,“]” を使って部分的に指定でき，指定が無い場合は構文解析結果が用いられる．規則 ($R3$) は，否定表現としか呼応しない副詞が “決して V する” という形で使用されている非文を**棄却**する．”!” は情報の否定であり，V の係り先が否定表現” ない” 以外の形態素であることを意味している．

```
N(CLASS:数量) しかない->N 以上はない.          …(R1)
[[N しか]V しない]->[V するのは]N だけだ].  …(R2)
決して-V する A(BASE_FORM!:ない)->reject.  …(R3)
```

図 3: SDR 規則の例

```
v_pos(setubiji,[(付属語活用あり,*)]).
v_pos(joshi, [(助詞, *)]).
```

図 4: 可変長変数の例

**依存木書き換え (DR) 規則**：SDR 規則をトランスレートした結果が DR(Dependency tree Rewriting) 規則となる．DR 規則は規則のクラス，識別子，名前，図 5 のような書き換え前後の依存構造の 5 つ組で表現される．

**依存構造処理 (DP) 言語**：DR の依存木はコンパイラにより，DP(Dependency tree Processing Language) オペレータの列に変換される．依存構造に対する操作はすべて DP オペレータを用いて行い，SDR や DR のレベルで記述できない手続きや意味解析部の実装にも用いる．図 6 は図 5 の DR 規則 ($R2$) をコンパイルした結果である．

## 3 高速化

規則の増加に伴い，規則との照合速度が問題となる．Kura では，高速化のためにコンパイラがヒューリスティックスを用いて規則内部の DP オペレータ列を最適化しており，各規則は情報が厳密に指定されている形態素により索引付けされている．修正・棄却では，構造変換で影響があったノードの情報から適用可能な規則集合を検索するデータ駆動をとっている．

## 4 実装・実験

システム構築には LiLFeS[6] を用いた (図 2④のレベル)．LiLFeS は単一化ベースで処理を宣言的に記述できる論理型言語的特長と，破壊的代入や繰返し，大域変数などを扱える手続き型言語的特長を兼ね備えている．Kura では本の照合に単一化を用いている．

今回の実験では，京大コーパス [4] から抽出した解析済みの 5000 文を入力とし，表現文型，否定に関する言い換え規則 452 個，修正・棄却規則 123 規則 [2]，活用形修正は別にモジュールとして用意して実験を行った．その結果 447 件の言い換え事例が得られた．入力 1 文に対する処理時間は Sparc 400MHz の計算機で約 0.4 秒であった．

*1 N や V などはデフォルトの変数であるが．図 3 ($R3$) の助動詞として用いられている $A$ などのデフォルトで指定されていない変数はユーザが定義すれば使用できる．その他の記号もマクロ定義が可能である．

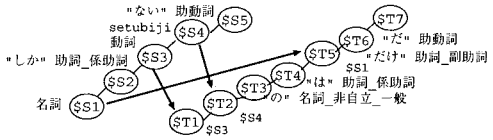


図 5: SDR 規則 ($R2$) に対応する DR 規則

```
dp_rule(M, negation, 108,"N-しか V-ない
        -> V-するのは N-だけ", 助動詞) :-
exist($S5&助動詞 (ない), M),
depend($S5&助動詞 (ない), $S4&setubiji, $S3&動詞),
depend($S3&動詞, empty, $S2&助詞_係助詞 (しか)),
depend($S2&助詞_係助詞 (しか), empty, $S1&名詞),
substitute($T5, $S1&名詞),
disown($S2&助詞_係助詞 (しか)),
substitute($T1, $S3&動詞),
substitute($T2, $S4&setubiji),
replace($S5&助動詞 (ない), $T7&助動詞 (だ)).
```

図 6: 規則 ($R2$) に対応するオペレータ列

## 5 まとめ

本稿では，言い換えエンジン Kura を紹介した．今後は，言語モデル内の規則の競合問題を考える必要がある．1 節 (c) については，文献 [9] を参照願いたい．なお，本システムは，ソースとともに **http://www.pluto.ai.kyutech.ac.jp/plt/inui-lab/** で公開している．

**謝辞**

(株)NTT で作成された日本語語彙大系を利用させていただきました．実装にあたり，東京大学辻井研究室で開発されたプログラミング言語 LiLFeS を使用させていただき，度重なる質問にも丁寧にご回答くださいました．形態素・構文解析に奈良先端大松本研究室で開発された ChaSen,CaboCha を使用させていただきました．また，活用形の修正のための知識として ChaSen の辞書を参考にさせていただきました．ここに深く感謝申し上げます．

**参 考 文 献**

[1] 言語処理学会．言語処理学会第 7 回年次大会ワークショップ論文集，2001.

[2] 飯田龍，徳永泰浩，乾健太郎，衛藤純司．言い換えエンジン KURA を用いた節内構造および機能語相当表現レベルの言い換え．情報処理学会第 63 回全国大会，3H-03，2001.

[3] 岩倉友哉，高橋哲朗，乾健太郎．不完全な内部表現および変換規則を用いた言い換えの実現方法．第 15 回人工知能学会全国大会，1A1-08，2001.

[4] 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.

[5] Lavoie, B., Kittredge, R., Korelsky, T. and Rambow, O. A Framework for MT and Multilingual NLG Systems Based on Uniform Lexico-Structural Processing. *ANLP-NAACL*, 2000.

[6] 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.

[7] Meyers, A., Yangarber, R. and Grishman, R. Alignment of Shared Forests for Bilingual Corpora. *In Proceeding of the 16th International Conference on Computational Linguistics*, pp. 460-465, 1996.

[8] 高橋哲朗，岩倉友哉，乾健太郎．不完全な構造変換規則による言い換えの実現方法．言語処理学会第 7 回年次大会論文集，p335-p338，2001.

[9] 高橋哲朗，岩倉友哉，飯田龍，乾健太郎．Kura：統一的かつ宣言的記述法に基づく言い換え知識の開発環境．電子情報通信学会思考と言語研究会，TL2001-12，2001.