

L T B 文生成部の構成

轟野浩司¹ 福島秀雄¹ 小島量² 池田光生³ 重永信一³

¹新世代コンピュータ技術開発機構 ²管理工学研究所 ³松下電器東京研究所

1. はじめに

我々は、汎用日本語処理系（L T B）^①に組み込まれる日本語文生成部の設計開発を行なっている。文生成部は、日本語文を生成する生成エンジン、形態素処理機構のほかに、編集機能、デバッグ機能など、文生成に必要なデータをユーザが作成するためのツールを提供するものである。本稿では、S I M P O S 上に設計した文生成部の全体構成についての説明をするとともに、開発用ツール群を使用して文法規則および生成用辞書を作成した例について述べる。開発した文生成部は次のような特徴を持つ。

- ・日本語の構造的な情報を用いて生成を行なう。
- ・形態素生成に用いる活用表は、L A X^②の形態素文法に準拠する。
- ・生成する語彙や語順を必要に応じて指定できる。
- ・文法と辞書が完全に分離されている。

2. 文生成部の構成

文生成部の全体構成を図1に示す。文生成部は、次のふたつの処理系からなる。

実行系：入力した中間表現をもとに表層文を生成するための処理機構（生成エンジン、形態素生成部）と、生成過程を追跡、デバッグするためのツール（デバッガ）からなる。

編集系：文生成に必要な各種データを作成、変更するためのツール（構造エディタ）と、データ変換のためのツール（トランスレーター）

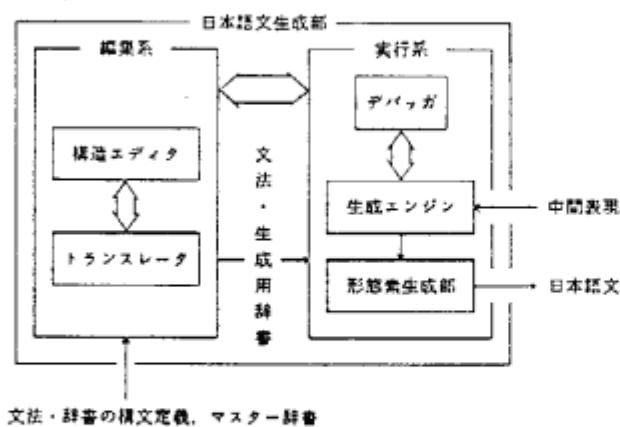


図1 文生成部の全体構成

からなる。

ここでいう中間表現とは、生成すべき日本語文の構文構造を記述したものであり、C I L^③の部分項で表現されている。また、辞書については現在、解釈／生成共用辞書（マスター辞書）^④を生成用辞書に変換したものを利用している。しかし、マスター辞書がない項目を一時的に追加したり、直接生成用辞書に変更を加えることも可能な構成になっている。実行系／編集系の利用は、メニューによるコマンド選択方式によって統合的に行なうことができる。つぎに、実行系および編集系について説明する。

2.1 実行系

表層文の生成は次のようにして行なう。

- (1) あらかじめ用意した文法規則および生成用辞書を用いることにより、入力した中間表現から表層構造を生成する。規則の適用および辞書の参照は生成エンジンが制御する。
- (2) (1) で生成した表層構造に形態素処理を施すことにより、表層文を生成する。この処理は、形態素生成部が行なう。

上記の(1)および(2)の過程をデバッガで追跡、デバッガするわけである。生成方式の詳細の説明については他稿^⑤にゆすることとし、本稿ではデバッガのもう機能について説明する。

デバッガ

文生成用デバッガは、C I L デバッガ^⑥の機能を用いたものである。C I L デバッガの種々の機能のうち、文生成部のデバッガには特に次の機能が有効である。

- (a) 規則適用過程を追跡する機能
- (b) 適用中の規則をソースファイル上で反転表示する機能
- (c) 実行中に各種データの内容を表示、変更する機能（インスペクタ）
- (d) 実行中の規則に一時的な変更を加える機能。文生成の際、文法規則および辞書は内部的にC I L述語に変換して用いているが、C I L デバッガ上に文法規則、辞書それぞれの構文定義を行なうことによってユーザが内部表現を意識せずにデバッガすることが可能になっている。また、上記(a)～(d)の機能の他に、生成用の独自の機能として、文生成部に入力した中間表現をウインドウにプリティプリントする機能などがある。デバッガ

^① Configuration of the Sentence Generator in LTB

HATANO, Kozi¹ FUKUSHIMA, Hideaki¹ OJIMA, Ryo² IKEDA, Mitsuo³ SHIGENAGA, Shin-ichi³

¹Institute for New Generation Computer Technology ²Management Engineering Research Institute, Ltd.

³Tokyo Research Lab., Matsushita Electric Industrial Co., Ltd.

を用いて生成過程の追跡を行なっている例を図2に示す。図では文法規則の適用過程を追跡している。

2.2 編集系

生成に必要なデータは、次のようにして作成する。

- (1) 文法規則：ユーザがエディタを用いて作成する。

- (2) 生成用辞書：マスター辞書を変換して作成するか、生成用辞書を直接編集する。

上記(1), (2)の作業を編集系が支援するのである。編集系は、文法の内容をそれらの構文定義にしたがって構造的に編集する構造エディタ、マスター辞書を生成用辞書に変換するトランスレーターからなる。以下に構造エディタ、トランスレーターについて説明する。

構造エディタ

文法規則など、定まった構文を持つデータの編集のために、構造エディタをツールとして用意した。本エディタの操作は、基本的にSIMPPOS上のPMACSの操作体系に基づいているが、ユーザがBNF記法による編集対象データの構文定義ファイルを作成することにより、次のような構造的編集を行なうことを可能にしている。

- (a) データの構文チェック
- (b) データのプリティプリント
- (c) 節単位の移動、複写、削除、
- (d) データのエントリ名検索
- (e) データ中の各種パラメタ記述の変更

本エディタによれば、データをテキストとして編集したり、構造的に編集したりという切り換えをコマンドにより編集中に行なうことができる。図2に本エディタを用いて文法規則を編集している例をあげた。図では、文法規則の修正を行なっている。

トランスレーター

トランスレーターは、マスター辞書から、生成に必要な情報をとりだして生成用辞書を自動作成する。変換は、マスター辞書が格納されているデータベースをアクセスしながら行なう。本トランスレーターは、前述の構造エディタからコマンドによって呼び出すことができる。

3. 開発環境の使用例

以上に述べた開発環境を用いて、我々は4000単語規模の辞書および文法を作成した。現在、これらのデータは文生成部に組み込まれ、全体として汎用的な日本語文生成部として稼働している。

4. おわりに

中間表現から日本語文を生成する文生成機構、および文生成用文法規則開発者用の開発環境をSIMPPOS上に構築し、汎用的な日本語文生成部を開発した。現在我々は、この文生成部を利用しながら談話理解システムDUALS第3版¹¹の文生成ブランディング部の設計を行なっている。

文字式 <pre> (語彙 / 生き ロール / 行為者 / 行為 / 語彙 / 語彙 / 適用性 <CONTROL> rewriter fail_if exists <SOURCE> up_to down_to edit_in edit_end acc_end <GOAL> Insect<dict> pale<dict> モード / [既成、既 モード / 必然] </pre>	<pre> CIL DEBUG AIDEDECAMP (test_grammar.cil.3) Command (LEASIO child:SPC parent:spc var:spc no_trace:n <CONTROL> rewriter fail_if exists <SOURCE> up_to down_to edit_in edit_end acc_end <GOAL> Insect<dict> pale<dict> (H1) 1 NCALL sentence(A,B) with:pattern , (C,D)) => &(bunkaku(A,C),symbol(D)) child (H2) 2 NCALL bunkaku(A,B) with:pattern , (C,D)) => &(jutsaku(A,C),symbol(D)) child (H3) 3 NCALL bunkaku2(A,B) with:AI名詞化+! 形式::+(pattern,(C,...))=> &(jutsaku(A,C), &Aspect(A,B),&(model(A,B),mishi(Cと,D))))) T no_trace X->["品詞"/N,"+eters"/D] => mishi_kv(X,P); bunkaku2(X,P) with AI名詞化+形式 pattern ::= (A,A,B) => jutsaku(X,B) & symbol((B,C,D)); aspect(X,A) & model(X,B) & mishi(Cと,D); bunkaku(X,P) with "+sp" (get_dict,[X,H,B]), X->["品詞"/N,"+eters"/D] => mishi_kv(X,P); mishi_kv(X,P) with "SS" (get_dict,[X,H,B]), X->["品詞"/N,"+eters"/D] => mishi_kv(X,P); mishi_kv(X,P) with "X->[get_dict,[X,H,B]]", X->["品詞"/N,"+eters"/D] => mishi_kv(X,P); </pre>
この選択の上で人間が生き続けていくためには、自然の進むどうしてもやらなければならない。	

左上：生成エンジンに入力した中間表現 左下：出力した日本語文
右上：文法規則の適用経過 右下：文法規則のソースファイル（現在適用されている規則部が反転表示されている。）

図2 デバッガの使用例

Original editor	Focus window
<pre> sentence(X,P) with pattern ::= (B,Sy) => bunkaku(X,B) & symbol((B,C,D)); </pre>	<pre> bunkaku(X,P) with "+sp" (get_dict,[X,H,B]), X->["品詞"/N,"+eters"/D] => bunkaku2(X,P); </pre>
<pre> bunkaku(X,P) with "+sp" (get_dict,[X,H,B]), X->["品詞"/N,"+eters"/D] => mishi_kv(X,P); </pre>	<pre> mishi_kv(X,P) with "SS" (get_dict,[X,H,B]), X->["品詞"/N,"+eters"/D] => mishi_kv(X,P); </pre>
<pre> bunkaku2(X,P) with AI名詞化+形式 pattern ::= (A,A,B) => jutsaku(X,B) & symbol((B,C,D)); aspect(X,A) & model(X,B) & mishi(Cと,D); </pre>	<pre> mishi_kv(X,P) with "X->[get_dict,[X,H,B]]", X->["品詞"/N,"+eters"/D] => mishi_kv(X,P); </pre>
<pre> mishi_kv(X,P) with "X->[get_dict,[X,H,B]]", X->["品詞"/N,"+eters"/D] => mishi_kv(X,P); </pre>	<pre> mishi_kv(X,P) with "X->[get_dict,[X,H,B]]", X->["品詞"/N,"+eters"/D] => mishi_kv(X,P); </pre>

左：編集中のデータ（文法規則）のソースファイル内容 左上：
現在編集中の箇 右下：プリティプリントされたデータ

図3 構造エディタの使用例

【参考文献】

- [1] 杉村：汎用日本語処理系LTBの構成、情報処理学会第37回全国大会予稿集、1988年9月
- [2] 杉村：論理型形態素解析LAX、Proceedings of the Logic Programming Conference、1988年4月
- [3] CIL言語マニュアル（第3版第2刷）、ICDT-TM-242、1988年4月
- [4] 渡辺：LTBマスター辞書の構成、ソフトウェア学会、論理と自然言語研究会、1987年12月
- [5] 池田：LTB文生成部の生成方式、情報処理学会第37回全国大会予稿集、1988年9月
- [6] 楠田：DUALSにおける談話処理、情報処理学会第37回全国大会予稿集、1988年9月