

ネットワーク仮名漢字変換サーバを用いた日本語環境：「たまご」

戸村 哲、石川 裕、二木 厚吉

電子技術総合研究所

「たまご」は日本語文書編集系 nemacs に仮名漢字変換機能を追加し、日本語の入力環境を提供するものである。仮名漢字変換機能は nemacs が直接行なうのではなく、ネットワーク上の仮名漢字変換サーバを呼び出して実現している。現在は仮名漢字変換サーバとして wnn V3 の jserver を利用している。ネットワーク仮名漢字変換サーバを利用することによって容易に仮名漢字変換機能の追加が出来た。また nemacs は自己拡張性を持つために仮名漢字変換機能を用いて日本語入力ユーザーインターフェースを作るのは短期間で出来た。本報告では、「たまご」の概要について述べる。

Egg: Japanese environment communicating with a network *kana kanji* conversion server

Satoru TOMURA, Yutaka ISHIKAWA, Kokichi FUTATSUGI

Electrotechnical Laboratory

1-1-4 Umezono, Tsukuba city, Ibaraki, Japan

“Egg” provides nemacs(extended GnuEmacs for Japanese) with new feature which enables us Japanese character input. To input Japanese character, we need a *kana kanji* conversion facility. The conversion facility is implemented by communicating with a network *kana kanji* conversion server. Currently “egg” communicates with wnn V3’s jserver as a network server. In this report an overview of “egg” is given.

1 はじめに

「たまご」¹はネットワーク上の仮名漢字変換サーバを使用して日本語文書編集系 nemacs での日本語入力機能を提供するものである。現在 1.10 版を公開中であり、1.10 版では仮名漢字変換サーバとして Wnn V3 の jserver を使用している。

漢字端末用の日本語入力方法には、wnn などのフロントエンドプロセッサ方式がある。この方式は、漢字の選択作業をフロントエンドプロセッサが行ない、応用プログラムは漢字コード入力の処理をすれば良いので、応用プログラムに依存しないで日本語入力機能を実現する利点を持つ。こうした利点を持つ一方、フロントエンドプロセッサは漢字変換選択作業のために端末画面制御を行なうので、画面編集系などの応用プログラムが更に端末画面制御を行なっている場合には、フロントエンドプロセッサと応用プログラムの両者が画面を管理制御しながら、お互いに相手の存在を知らないために、それぞれが管理している画面の状態を実際の画面の状態が一致せず、画面表示が乱れる欠点を持つ。

また応用プログラムとフロントエンドプロセッサとが独立であるので、応用プログラムの中から日本語入力に関する制御や新しいサービスを提供するのが難しい欠点もある。

Emacs に代表される自己拡張可能な画面編集系の場合、日本語入力をするのに必要な機能（例えば仮名文字列を漢字文字列に変換する機能など）を画面編集系の中から利用できれば、比較的簡単に日本語入力環境を実現でき、かつ新しいタイプの入力方法などの実験を簡単に行なうことができる。

この2つの観点、従来のフロントエンドプロセッサの欠点をなくし、さらに日本語環境の実験を行なうことを目的として「たまご」を開発した。

本報告ではネットワーク仮名漢字変換サーバを利用して nemacs での日本語入力機能を提供する「たまご」について、その開発動機、経過、および「たまご」の概要について述べる。

2 なぜ「たまご」を作ったか

計算機上の日本語環境も整備され、公開ソフトウェアとして利用できる Unix 上の日本語環境には、仮名漢字変換機能を提供し、漢字端末の日本語フロントエンドプロセッサである Wnn[1] があり、日本語の文書編集系には nemacs[2] がある。これは英語用文書編集系

GnuEmacs[3] を改造し、日本語文書を編集し、漢字端末に表示する機能を追加したものである。さらにワークステーションで X-window system を利用する場合には漢字端末エミュレータ kterm が利用できる。また日本語の文書の清書系には jtex, jlatex があり、この報告もこうした環境の上で書かれたものである。

こうした日本語環境をしばらく使用していると、次の不具合な点に気付いた。問題は主にフロントエンドである wnn と nemacs との相性の問題であった。これを列挙する。

- wnn では、ひらがなだけを入力する場合も含めて、仮名漢字変換の情報が端末画面最下行に表示される。この表示場所は仮名漢字変換結果を実際に挿入する画面上の場所と離れているため、入力・変換・確定のたびに視線が上下に移動して作業しにくい。
- nemacs が画面の再表示を行なうと wnn の表示情報が消えてしまう。
- wnn と nemacs の両方が独立に端末の画面管理と画面制御を行っており、両者ともに実際の画面の状態を調べないので、想定している画面状態と実際の画面状態が一致なくなり、端末画面の表示がおかしくなることがたびたび発生した。
- wnn を起動しないで、nemacs を使い始めた時点で、手紙を書くなどで途中で漢字入力が必要になると、まず nemacs を終了し、wnn を起動して、新しい nemacs を起動する必要がある。
- フロントエンドプロセッサである wnn に異常が発生した場合に、nemacs そのものも強制終了しなければならない。
- 画面編集系を使用しているにも拘らず、例えばバッファにある日本語を編集して辞書登録することができない。つまり仮名漢字変換機能と画面編集系の機能を融合させた使い方ができない。

これらの問題点の原因の一つは2つのプログラムが互いに独立に端末画面を制御していることによっておこる。応用プログラムと独立に動作できることがフロントエンドプロセッサの利点であるが、この場合は逆に問題点となっている。原因のもう一つは日本語入力機能を nemacs の機能の一部として実現せずにフロントエンドプロセッサとして外付けしたことによっている。

これらの問題点を解決するためには、nemacs の中から仮名漢字変換機能を利用できるようにし、その機能を使って日本語入力機能を実現する方法がある。しかもこの方法は、これらの問題点を解決する他に、nemacs が自己拡張可能な画面編集系であるので、より良い日本語

¹「たまご」は、「沢山待たせて御免なさい。」の各文節の先頭1音である「た」、「ま」、「ご」をとったものである。この命名は錦見美貴子氏によるものである。卵は料理の基本材料でいろいろな料理につかえることから、「たまご」もいろいろな日本語環境を作る材料になりたい、というつもりで名付けたものである。

環境に関する実験環境として利用できるという利点もある。

こうした考え方から日本語文書編集系 nemacs に仮名漢字変換機能を付加し、標準的の日本語入力環境を提供する「たまご」の開発を開始した。²

3 どのようにして「たまご」を作ったか

「たまご」の開発の経緯を述べながら、「たまご」の構成を述べる。

3.1 ひらがなのローマ字入力をつくる

仮名漢字変換に基づいて日本語入力を行なうには、まずひらがなを入力する必要がある。ひらがなを入力する方法には JIS 鍵盤を使った仮名直接入力方法と、ASCII 鍵盤でも入力できるローマ字仮名入力方法がある。

ローマ字仮名入力方式の場合は、入力文字列をひらがなに変換する入力文字変換機能が必要である。入力文字変換方式を利用者定義可能な変換表方式を採用すると、変換表を選択することによって仮名直接入力も実現できる。そこで変換表方式の入力文字変換機能を実現することにした。³

この変換表方式の入力文字変換系としては、既に Wnn の romkan が利用可能であった。しかし、romkan を利用するには C でプログラムしなければならない。ところが emacs は自己拡張を行なうための言語として emacs lisp を内蔵しており、emacs 自身の多くの部分がこの emacs lisp で実現されている。そのため、nemacs 上での将来の拡張性を確保し、変更を容易にするためには、emacs lisp を用いて実現するのが望ましい。そこで入力変換系についても emacs lisp で実現する方法を採用することとした。本来 emacs は keymap を持っており、鍵盤から入力された文字に対して keymap で指定した emacs lisp 関数を呼び出すことで、各種のコマンドを実現している。そこでまず入力文字変換を入力文字に従って変換の状態が遷移するとともに keymap を取り替える方法の実験を行ない、実現可能なことを確認した。しかし、この方法は実行速度が速くないと考えらることも、複雑な状況の下で他の emacs プログラムと共存できるかどうか問題がありそうなこと、などの問題点が指摘された。[4]

そこで入力文字変換系の実現方法を emacs lisp で有限状態オートマトンを作る方法に変更した。この方式では、まず変換表を有限状態オートマトンの状態遷移グラフに変換する。そして、グラフに従って有限状態オートマトンの模倣を行なう。

²作業は昭和 62 年 12 月下旬から開始し、実装上の基礎実験を昭和 63 年 1 月頃に終了し、基本実現は昭和 63 年 3 月頃に完成した。

³実際には鍵盤のキーの数が足りないために、仮名直接入力方式を実現できない場合が多い。

入力文字変換表は、入力文字変換規則で定義する。この指定方法は romkan での方法を参考にし、それを簡単にしたサブセットが利用できる。

3.2 どのようにして nemacs に仮名漢字変換機能を追加するか

先に述べたように emacs では、多くのプログラムを emacs lisp で実現する。そのため、将来の拡張性と実現の容易さを考えて nemacs に仮名漢字変換機能を追加する方法として、emacs lisp の関数として仮名漢字変換機能を提供することにした。

仮名漢字変換機能はこれを実現するのは容易ではないため、wnn の仮名漢字変換サーバである jserver を利用することにした。jserver との標準的のインターフェースとしては C 用のインターフェース jljb が提供されている。

emacs lisp と jserver のインターフェースを作るには次の方法がある。

1. emacs の kernel は C で実現されているので、kernel の一部に jljb を利用して追加変更を加え、jserver とのインターフェース関数を emacs lisp の組み込み関数として実現する。
2. emacs kernel を変更せずに、jljb を使って jserver とのインターフェースをするプログラムを emacs の外部プロセスとして並行して実行する。そしてこのプロセスと通信する関数を emacs lisp で定義する。⁴
3. jserver との通信を emacs lisp の network-stream を用いて直接行なう。⁵

1 番目の方法は kernel を変更する方法で、2, 3 番目の方法は kernel を変更しない方法である。2 番目の方法は追加の C プログラムを必要とし、3 番目の方法はすべてが emacs lisp で実現される。

将来の変更や拡張を考えると 3 番目の方法が一番望ましい。次に kernel を変更しない 2 番目の方法が望ましい。

2 番目の方法については、開発当時 emacs lisp から 8 ビットデータをプロセス間通信する方法が良く解らなかったし、一つ余計に外部プロセスを実行させるのは負荷が思いのではないかと考えた。3 番目の方法については、開発当時は network-stream が利用できることを知らなかった。そして 1 番目の方法は比較的簡単にできそうであった。⁶ こうした理由で emacs lisp と jserver

⁴emacs lisp にはパイプを利用したプロセス間通信機能がある。

⁵emacs lisp では tcp/ip の tcp プロトコルを使用するプロセスとネットワーク通信可能である。

⁶現在 3 番目の方式を採用した版を開発中である。

とのインターフェースは、nemacs の kernel 内に wnn の jlib を利用した C プログラムを追加し、emacs lisp の組込関数として実現することにした。

emacs lisp で仮名漢字変換を行なう時に、次の順序で関数を呼び出すものとした。まず任意のひらがなの列に対して「変換始め」によって仮名漢字変換が開始される。そして任意回の「ある文節の次変換」、「ある文節の長さ伸縮」で変換、文節区切りを修正する。各文節についてその候補を選択したかを「文節確定」で決定し、最後に「変換終り」で仮名漢字変換が終了する。

jlib では、「変換始め」が jd_reconv、「ある文節の次変換」が jd_next、「ある文節の長さ伸縮」が jd_tanconv に対応する。ところが、jd_next によってある文節の変換候補は得られるが実際にどれを選んだかを jserver 側に伝える「文節確定」に対応する entry がなかった。

また一回の文字列を変換する間に何回も文節の次候補を必要とする場合に、その度に jserver から変換結果を貰うのは、ネットワークの通信量を不必要に増やし、仮名漢字インターフェースの応答速度が遅くなる原因になると考えた。

「文節確定」を実現し、「ある文節の次候補」の実行速度を速くするために、文節の変換結果すべてを emacs の kernel 内に記憶し、変換が確定して終了する時点でその情報を一度に jserver に伝わることにした。

具体的には、jlib のプログラムを見て jd_open_in と jd_end が使っている構造体の使い方を調べ、この仕様に合った手続きを作成した。jd_next を呼び出す時に使う次候補用のエントリを文節分だけ作り、変換確定時に、次候補用のエントリから jlib が見ている構造体書き込むことで、jd_end を呼んだ時に変換情報が jserver に伝わるようにした。

3.3 「たまご」の使い方を考える

入力文字変換機能で仮名を入力し、仮名漢字変換機能で仮名文字列を漢字かな文字列に変換できるようになったので、これらを組み合わせて標準的な日本語入力インターフェースを作る必要がある。

emacs は文字型端末上の汎用ウィンドウシステムを持ち、バッファに文字を挿入すると画面に表示されるという点で容易に emacs lisp から端末画面制御をすることができる。このため directory editor など文字型端末上の視覚的インターフェースも数多く作られている。

機能的には emacs の機能で十分であるが、視覚的ユーザーインターフェースを作る上で便利な汎用機能として、Notification System と Menu System を作成した。Notification System は「たまご」が minibuffer に表示したメッセージを特別のバッファにメッセージの

表示された時刻とともにとっておくものである。一般にはこうしたメッセージは一時的に表示されるだけで十分であるが、場合によっては後で必要になることもあるので、必要である。

Menu System は minibuffer を用いて選択肢の表示と選択を行なうものである。後述する次候補選択や辞書登録情報選択などはこの機能を使って実現している。

「たまご」は emacs 上に実現してあるので、具体的なユーザーインターフェースを利用者が変更するのは極めて簡単である。しかし、その基となる標準インターフェースの設計をする必要がある。この設計はすでに利用実績がありかなり優れたインターフェースであると考えていた wnn のユーザーインターフェースを基本とした。

4 「たまご」の使い方

本節では「たまご」の使い方の概要を述べる。

4.1 準備：環境を設定する

「たまご」を使うには、まず環境設定を行なう必要がある。「たまご」を使用する環境の指定は、emacs の初期化ファイル ~/.emacs と「たまご」の初期化ファイル ~/.eggrc とで指定する。~/.emacs では仮名漢字変換サーバの指定を行ない、~/.eggrc では使用辞書の指定を行う。

4.1.1 仮名漢字変換サーバの指定

「たまご」が利用する仮名漢字変換サーバが動いているホスト名を変数 jserver-host-name で指定する。jserver が動いているホストが "etlcom" の場合は、~/.emacs に

```
(setq jserver-host-name "etlcom")
```

を追加する。

4.1.2 使用する辞書の指定

~/.eggrc の中では、主に仮名漢字変換サーバの使用辞書の設定を行なう。

```
(set-default-sys-dic-directory  
  "/usr/local/lib/dic/sys/")
```

はシステム辞書ディレクトリのデフォルトディレクトリを指定する。同様に

```
(set-default-usr-dic-directory  
  "/usr/usrdic/$USER/")
```

は利用者辞書ディレクトリのデフォルトディレクトリを指定する。

setsysdic, setusrdic の使い方は Wnn の .wnnrc と同様で、setsysdic はシステム辞書の使用を、setusrdic は利用者辞書の使用をそれぞれ仮名漢字変換サーブに宣言する。

```
(setsysdic "pd-kihon" "pd-kihon" 5)
```

は、1 番目の "pd-kihon" がシステム辞書 /usr/local/lib/dic/sys/pd-kihon.sys を使用することを指定し、2 番目の "pd-kihon" が頻度情報を /usr/usrdic/\$USER/pd-kihon.hindo に格納することを指定する。最後の 5 はこの辞書の優先順位を指定する。拡張子の ".sys" と ".hindo" は「たまご」が自動的に付加する。同様に

```
(setusrdic "my-dict" 5)
```

は利用者辞書 /usr/usrdic/\$USER/my-dict.usr を優先順位 5 で使用することを意味する。拡張子の ".usr" は「たまご」が自動的に付加する。

4.2 どんな端末で使えるか

「たまご」を動かすのに必要な端末に文字型漢字端末である。漢字を表示できれば特別な端末である必要はない。X-window を使用する場合、xterm は漢字が表示できないので、kterm を使う必要がある。

4.3 仮名漢字変換の仕方

「たまご」では漢字を入力する方法として、

1. wnn と同等な対話式漢字入力機能
2. emacs の拡張コマンドによる仮名漢字変換機能

の 2 つを提供している。

4.3.1 対話式漢字入力法

「たまご」が標準的に提供する対話式漢字入力方法は基本的には wnn の方式と同じである。大きく異なる点は入力変換作業が文字の挿入点で行なわれることである。

4.3.1.1 透過モード 「たまご」が実装された nemacs は立ち上がると、モードラインが下記のようになる。この状態を透過モードと呼ぶ。

```
[----] ---- nemacs: *scratch* ...
```

透過モードではほとんどの入力がそのまま nemacs に入力され、通常の nemacs と同様に使用できる。

4.3.1.2 ローマ字仮名モード 透過モードで C-\ を打つと、モードラインが以下ようになる。この状態をローマ字仮名モードと呼ぶ。

```
[ a a] ---- nemacs: *scratch* ...
```

ローマ字仮名モードでもう一度、C-\ を打つと、透過モードに戻る。

ローマ字仮名モードでのコントロールキーやメタキーの入力はそのまま nemacs に入力され、通常の nemacs と同じように使用できる。カーソル移動などは、このモードのままで実行できる。

一方ローマ字仮名モードで、ローマ字を入力するとカーソルの位置に縦棒 (|) 2 つが現れ、その間に入力をローマ字仮名変換したひらがなが表示される。Wnn でおなじみの変換例、「私の名前は中野です」

```
watasinonamaehanakanodesu
```

を打つと、以下ようになる。

```
|わたしのなまえはなかのです|
```

この縦棒をフェンスと呼び、この状態をフェンス・モードという。

4.3.1.3 フェンス・モード フェンス・モードでは文字入力とフェンス・モード編集コマンドしか使用できない。通常の emacs コマンドの内では、C-a, C-b, C-d, C-e, C-f, C-g, C-h, C-k, C-t コマンドが利用できる。

フェンス・モードで半角文字を入力するには、C-\ を叩く。すると、モードラインが

```
[ a a] ---- nemacs: *scratch* ...
```

となる。これを半角入力モードと呼ぶ。フェンス・モードの半角入力モードでは、半角文字が入力できる。

フェンス・モードでは、C-\ を打つとローマ字仮名モードと半角入力モードを行き来する。C-1 または CR を打つと、フェンスが消え、フェンス・モードから抜ける。従ってひらがなの入力をするには、ローマ字入力をして最後に C-1 または CR を打てばよい。

フェンス・モードで C-c または C-g を打つと、フェンス・モードから抜けると同時にフェンスに入力された文字もキャンセルする。

またフェンス・モードで、C-_ と打つと、ミニバッファに JIS code: と表示され、16 進数で JIS code を直接入力できる。

4.3.1.4 漢字変換モード フェンス・モードでは、C-w によって仮名漢字変換できる。C-w のほかに、C-q または SPC でも変換できる。

頻度情報の違いにより、多少変換結果が異なるが、仮名漢字変換を実行した結果が以下のようになっていたとする。

|私のな前は中野です|

各文節は、一個の半角空白で区切られる。ある文節の変換結果が違っている場合には他の変換候補を選択できる。C-0、SPC または C-n によって直後の候補を表示し、C-p によって直前の候補が表示される。

この状態で C-g を叩くとミニバッファに候補一覧が出現し、カーソルがミニバッファに移動する。次候補の数がおおく、ミニバッファですべてを一度には表示できない場合は、C-n で次の候補一覧が、また C-p で前の候補一覧をミニバッファに表示される。ミニバッファに表示されている候補を選択するには、カーソルを候補に移動させる。C-f でカーソルが次の候補に、C-b でカーソルが前の候補に移動する。候補決定には、リターン・キーを打つ。この候補一覧の状態から抜けるには、C-g を打つ。

次の文節への移動は C-f、前の文節への移動は C-b である。先頭の文節への移動は C-a、最後の文節への移動は C-e である。

文節の切り方が違っている場合は、カーソルのいる文節の長さを C-i によって短く、C-o によって長くすることができる。例の場合、先頭文節の長さが長いので C-i で短くすると正しく変換される。

ひらがな入力そのものが違っている場合は、C-c を叩くと、仮名漢字変換を中止してひらがな入力状態に戻る。また先頭のいくつかの文節を確定し、残りの部分をひらがなに戻して編集するには、C-k を叩く。カーソルのいる文節の直前までが確定し、残りがひらがなに戻る。

全ての文節が確定したら C-1 または CR で確定した文字列が入力されフェンスが消える。

4.3.2 Emacs の拡張コマンドによる仮名漢字変換

「たまご」は対話式漢字入力の他に色々な仮名漢字変換機能を Emacs レベルで提供している。これらの機能はもちろん emacs lisp の中から呼び出すことができる。ここで説明する機能は emacs の拡張コマンドとして提供している。emacs で拡張コマンド command を実行するには M-x command を叩くことで実行する。

4.3.2.1 JIS コード入力コマンド JIS コードで直接入力するコマンドに jis-code-input がある。これを実行すると、ミニバッファに JIS code: とプロンプトが表示されるので 16 進コードで JIS コードを入力する。例えば

JIS code: 2170

と入力すると、カーソルの部分に ”\$” が表示される。(16 進数の大文字・小文字の区別はない。)

次に示す一連のコマンドは指定された範囲の文字列を変換するものである。同じ変換をするコマンドには(変換の名前を*で表すと)変換する範囲によって *-region, *-word, *-paragraph, *-sentence がある。以下では *-region を例にとって説明する。

4.3.2.2 全角変換コマンド 英数字や記号などの半角文字を全角文字にするには、zenkaku-region がある。例えば以下の文

These are hankaku characters.

をリージョンとして指定してこれ呼び出すと、

These are hankaku characters.

となる。

4.3.2.3 半角変換コマンド 英数字や記号などの全角文字を半角文字にするには、hankaku-region がある。例えば以下の文

These are zennkaku characters.

をリージョンとして指定してこれ呼び出すと、

These are zennkaku characters.

となる。

4.3.2.4 ひらがな変換コマンド カタカナをひらがなに変換するには、hiragana-region がある。例えば以下の文

コレハカタカナダヨ。

をリージョンとして指定してこれ呼び出すと、

これはかたかなだよ。

となる。

4.3.2.5 カタカナ変換コマンド ひらがなをカタカナに変換するには、katakana-region がある。例えば以下の文

これはひらがなだよ。

をリージョンとして指定してこれ呼び出すと、

コレハヒラガナダヨ。

となる。

4.3.2.6 ローマ字ひらがな変換コマンド ローマ字をひらがなに変換するには、roma-kana-region がある。例えば以下の文

koreharo-majidesu.

をリージョンとして指定してこれと呼び出すと、

これはろーまじです。

となる。

4.3.2.7 ローマ字漢字変換コマンド ローマ字を漢字に変換するには、roma-kanji-region 例えば以下の文

watashino namae ha nakano desu

をリージョンとして指定してこれと呼び出すと、漢字変換モード

！私のな 前は 中野です！

になる。リージョン内の空白文字は削除するのでローマ字のわかし書きをしている場合も扱える。

4.3.2.8 漢字変換コマンド バッファの部分の漢字に変換するには、henkan-region がある。例えば以下の文

わたしのなまえはなかのです

をリージョンに指定してこの機能と呼び出すと、漢字変換モード

！私のな 前は 中野です！

になる。

4.4 どうやって辞書を更新するか

「たまご」で辞書を更新するためには

1. 新しい単語を拡張コマンドで登録する方法
2. ある読みに関するすべての辞書項目を編集する方法

の2つの機能が提供されている。

4.4.1 辞書登録

辞書登録は、wnn 方式のように仮名漢字変換の途中で辞書登録を行なうのではなく、Emacs の拡張コマンドの形式で行なう。

例えば、バッファ上に表示されている「電子技術総合研究所」という単語を登録することにする。登録する語の範囲をリージョンで指定してから toroku-region を実行する。するとミニバッファに

辞書登録「電子技術総合研究所」読み：

と表示されるので、読みをローマ字で入力し（自動的にひらがなに変換する）CRで入力を終了する。すると、語の品詞を選択するためにミニバッファに品詞一覧が表示される。

品詞： 0. 名詞 1. 固有名詞 2. 動詞

品詞選択モードでは C-f でカーソルが右に動き、C-b でカーソルが左に動く。C-n を打つと次の品詞一覧を表示する。

品詞： 0. 副詞 1. 接続詞・感動詞

C-p を打つと前の品詞一覧に戻る。カーソルを移動し、リターンキーを打つことによって品詞を選択する。

動詞にはいろいろな種類があり、例えば「サ変（名詞型）語幹」を選択するには、まず「動詞」を選択し、次の選択リストから、「サ変（名詞型）語幹」を探し、選択する。

今回の場合、「電子技術総合研究所」は固有名詞なので、固有名詞を選ぶ。次に、どの辞書にこの単語を登録するかを選択する。品詞選択の場合と同様に C-f, C-b, C-n, C-p 等を使ってカーソルを登録したい辞書に移動して選択し、リターン・キーを打つことによって登録を行なう。

4.4.2 辞書項目編集

利用者辞書の登録項目を修正するには、edit-dict-item がある。これを実行してからミニバッファで項目の読みを指定する。すると *Nihongo Dictionary Information* というバッファが作られ以下のように表示される。

*話	す	サ行五段語幹	pd-kihon.sys
*離	す	サ行五段語幹	pd-kihon.sys
*放	す	サ行五段語幹	pd-kihon.sys
*花		名詞	pd-kihon.sys
*鼻		名詞	pd-kihon.sys
*華		名詞	pd-kihon.sys
*ハナ		固有名詞	pd-jinmei.sys
*塙		単漢字	pd-tankan.sys

行頭の「*」はその辞書項目がシステム辞書に登録されていて削除できないことを表す。利用者辞書に登録されている辞書項目では*の代わりに空白が表示する。

「話 す」は辞書項目の見出し語で、語幹（「話」）と活用語尾（「す」）とを空白文字で分けて表示する。「サ行五段語幹」は辞書項目の品詞で、最後の項目は登録されている辞書名である。

使い方は、Direx と同様である。通常のカーソル移動コマンドの他に次のコマンドが使用できる。

n: カーソルが下に移動する。
p: カーソルが上に移動する。
a: 辞書項目を追加する。
d: カーソルの辞書項目を削除指定する。削除指定された辞書項目は行の先頭に "D" が表示される。
u: 削除指定された辞書項目の削除指定を解除する。
x: 削除指定された辞書項目を実際に削除する。
q: 編集を終了するが、削除は行なわない。

4.5 ミニバッファで漢字を入力する

ミニバッファでも仮名漢字変換ができる。バッファと同様に C-\ でローマ字仮名モードになる。ただし、モード表示はない。

5 おわりに

フロントエンドプロセッサ方式の日本語入力方式の問題点を解決するために、画面編集系にネットワーク仮名漢字変換サーバを利用して仮名漢字変換機能を追加し、その上に日本語環境「たまご」を作成した。

「たまご」の開発経験は仮名漢字変換機能を画面編集系に追加することで、比較的容易に柔軟な日本語入力システムを実現できることを示している。

また各自が実行する画面編集系が独自の仮名漢字変換システムを内蔵するのではなく、ネットワーク上の仮名漢字変換サーバが提供する仮名漢字変換機能を利用することによって、各自の画面編集系の負担を軽くすることができ、より多くの環境で利用可能なシステムとすることができた。

現在の版の「たまご」では、emacs と仮名漢字変換サーバとの接続は emacs の kernel の一部に C のプログラムを追加して実現している。こうした理由は、開発当時には仮名漢字変換サーバとのインターフェースが C で書かれたものしか知らなかったからである。

emacs kernel に仮名漢字変換機能を組み込む方式は実行速度が速いという利点がある。欠点としては、

1. emacs kernel を変更すると、emacs 全体を make し直す必要があり、デバッグ作業を行なうのに時間が掛かる。デバッグに必要な情報を取り出すにも emacs kernel 自身を変更する必要がある。
2. emacs kernel は C が書かれており、C には動的領域管理機能がない。そのため辞書項目などの領域を静的に確保する必要がある。静的な領域確保は、大きな領域を確保するとディスク領域を不必要に占有し、小さい領域しか確保しないと沢山の辞書項目がある時に処理できないという致命的欠陥がある。

がある。この欠点を解決するには、仮名漢字変換機能を

emacs の自己機能拡張記述言語である emacs lisp で実現する必要がある。emacs lisp で実現すれば変更・検査が容易であり、また lisp の動的領域管理機能を用いて領域確保をすることができる。

このためには emacs lisp にネットワーク通信機能があり、また仮名漢字変換サーバがネットワークプロトコルを採用している必要がある。現在この両方の条件が実現されているので、この方式による仮名漢字変換機能の追加を実験中である。

「たまご」の提供する標準的漢字入力方式の基本設計は wnn の入力方式を採用したものである。今までの多くの利用経験を踏まえてこの入力方式の変更・再検討を行なっていく予定である。また「たまご」ではそうした変更が極めて容易に実現できるので、新しい日本語環境の実験を行なう実験環境としても有効性を示すことができると思われる。

参考文献

- [1] "Wnn 仮名漢字変換システム リファレンス・マニュアル、" アステック、1987。
- [2] 半田剣一、小方一郎、"nemacs" bit 10月号, pp. 22 - 31, 1988。
- [3] R. Stallman, "GNU Emacs Manual," Fourth Edition, Emacs Version 17, Feb. 1986。
- [4] 近山 隆: private communication, Jan. 1988。