真鶫道 (manued.el) の実装について
— 第 248 【会/回】 PTT —

山内斉

1999 年 4 月 22 日 (木)

概 要

第 39 回のプログラミングシンポジウムにおいて竹内郁雄によって提案された Manued の処理系を実装したのでそれについて報告する．Manued とは電子メイル，つまり text 画面上で原稿を校正する方法を指す.

遠隔地において共同研究を行なう際，論文の校正は一つの試練である．郵送によるやりとりはその速度が問題となり，Fax では電子的な情報ではなくなるために編集時にミスが入り易いという欠点がある．そのため，遠隔地での論文のやりとりには通常電子メイルが用いられる．しかし，電子メイルによる原稿のやりとりでは各人の利用する環境・ソフトウェアの違いや利用可能な文字種の制限により，紙に朱を入れるような自由度を達成することは難しい.

このような問題に対して，第 39 回のプログラミングシンポジウムにおいて竹内郁雄が「電子メールで原稿を修正する方法」と題し，いかにして plain text を校正するかの方法を提案した．この校正方法が Manuscript Editing (Manued)，真鶫道である．Manued では入力の容易さや，校正記号の解釈の曖昧さの排除，人間の可読性，数分で理解できる簡易性，機械的な処理などが考慮されている.

今回筆者は Manued を Emacs 上に実装した．実装においては竹内の提案をベースとし，いくつかの機能を拡張した．ベースとなる機能は校正箇所の強調，校正前の文書と校正後の文書の自動抽出，Manued における校正コマンドのカスタマイズ等である．拡張した機能としては，入力補助機能，校正箇所のサーチ機能である．さらに TEX との親和性を考慮し，LaTeX-mode などと共存可能な minor-mode も実装した．それまで筆者は Lisp という言語でプログラムを書いたことがなかったので，これが筆者の書いた初の Lisp プログラムである.

1 はじめに

第 39 回のプログラミングシンポジウムにおいて竹内郁雄によって提案された Manued の処理系を実装したのでそれについて報告する．Manued とは電子メイル，つまり text 画面上で原稿を校正する方法を指す.

真鶫道は文献 [3] によって提案されたものである．まずはじめに，真鶫道について，文献 [3] を引用しつつ概説する．この文献を参照できる方は，本節を飛ばして 2 節から読んでも良い.

1.1 望まれる校正手法

電子メイルで校正を行う方法について文献 [3] では，次のように考えている.

> 電子メールによって原稿を修正する際には,
>
> - 紙の上のような訂正記号が使えない．利用できる文字集合の種類が限られる.
> - 色が利用できない.

- mail reader によっては文書の二次元情報を保存しないものがあり，編集者や著者が異なる mail reader を用いている場合に二次元情報が伝わらない (例えば Eudora など)．もともと電子メイルでは二次元情報についての規定がないのでそのような状況は常に発生しうる．
- 改行位置に依存したくない．エディタやワープロなどの行の折り返し機能などによって訂正情報が失われることを避けたい．

という問題がある．これまでは郵送するという手が一般に行なわれてきたが，それでは間にあわないことが多い．Fax という手もあるが，受信した紙から電子情報への変換時には人間が入力する部分が存在し，手間がかかる上に誤りが混入する可能性が高くなる．手書き文字の OCR 技術もあるがまだ成熟したとは言えない．さらに朱を入れた原稿中の訂正記号を解釈して文書を編集することは現在のところ困難であると思われる．結局，もともとの原稿は多くの場合電子情報であり，それを利用できなくする手はあまり賢いとは言えまい．
そこで,

- 修正前と修正後の文書を機械的に抽出できること (必須)
- 人間にとって読みやすいこと
- 訂正箇所の視認性が良いこと
- 修正に対するコメントが入れられること
- 修正情報の書き込みが容易なこと
- 繰り返される修正を抽象化すること

の可能な校正手法が望まれている．

と述べている

1.2 真鵺道

前節で述べたような用件を満たす方法として文献 [3] では表 1 に示す記法を提案している．まずは簡単な例題を示してその雰囲気をつかんでもらおう．次の文 1 を文 2 のように修正したいとする．

1. **ハリセ・ルダン**のプランの実行には**一千世紀**を要する．
2. **ハリ・セルダン**のプランの実行には**一千年紀**を要する．

これまでの手法としては,

```
ハリセ・ルダンのプランの実行には一千世紀を要する．
^^^^^^^^^^^^^^ ハリ・セルダン   ^^^^^^^^ 一千年紀
銀河帝国 12000 年の中で最も偉大な          原文は one millennium のはず
数学者の名前を間違うとは
```

などのように ^ などを用いて下線を示し，訂正を行う手法が広く用いられているようである．この手法は私もどこで見たのか真鵺道を知るまでは利用していた．文献 [3] でも起源は謎とされており，

調査中とのことである．考えてみると最初に利用した人は確かになかなか賢い人である気がする．筆者はこの手法を conventional な手法と呼んでいる．校正においてはこの場合のようにコメントなどを含めることもある．このコメントについても，どこまでが訂正で，どこまでがコメントなのか不明なことが多く，また記述の個人差も大きい．しかも二次元情報が保持されないような場合には次の囲みのように乱れてしまう．上の囲みは LaTeX の `verbatim` 環境で表示し，下のものはそれを外している．(ただし，^は LaTeX では特殊な記号なので，これだけは `verb` で囲っている．)

ハリセ・ルダンのプランの実行には一千世紀を要する．^^^^^^^^^^^^^ ハリ・セルダン ^^^^^^^^ 一千年紀銀河帝国 12000 年の中で最も偉大な 原文は one millennium のはず数学者の名前を間違うとは

ここに次のようなルールを導入する．

- $[\alpha\backslash\beta\bigstar\gamma]$：$\alpha$ を β に変更する．γ はコメントである．

これは sed の s コマンドのようなものを想像してもらえば良い．このコマンドを利用することによって，これまでの例は次のように書ける．

[ハリセ・ルダン \ ハリ・セルダン] のプランの実行には [一千世紀 \ 一千年紀] を要する．

この記法のバリエーションとして，

- $[\backslash\alpha]$：挿入
- $[\alpha\backslash]$：削除

が定義できる．表 1 の記法を用いてコメントまで加えてみると，

[ハリセ・ルダン \ ハリ・セルダン★銀河帝国 12000 年の中で最も偉大な数学者の名前を間違うとは] のプランの実行には [一千世紀 \ 一千年紀★原文は one millennium のはず] を要する．

となる．

また，文献 [3] では使える文字の制限があることから，これらの校正記号を定義するメタな言語を定義している．たとえば，表 1 の記号は，

defparentheses [ ]
defdelete \
defswap |
defescape ~
defcomment ★

のように定義されている．これは [ ] 部分が校正記号の適用範囲であり，置き換え記号は \ によることなどを規定している．実際，筆者が TeX の原稿を修正する場合にデフォルトの記号を使用することはめったにない．なぜなら，[ ] は TeX の option 引数を示すコマンドであり，また，\

表 1: 文献 [3] で提案されている真鶫道の記法

記法	意味
[$\alpha\backslash\beta$]	α を β に変更する
[$\backslash\alpha$]	α を挿入する (変更の特殊形)
[$\alpha\backslash$]	α を削除する (変更の特殊形)
[$\alpha\|\gamma\|\beta$]	$\alpha\gamma\beta$ を $\beta\gamma\alpha$ に並べ換える
[$\alpha\|\|\beta$]	$\alpha\beta$ を $\beta\alpha$ に並べ換える (上の一般並べ換えの特殊形)
[$\alpha\backslash\backslash$]	前に出てきた [$\alpha\backslash\beta$] を繰り返す (マクロ適用)
[$\alpha\backslash\backslash$ ラベル]	ラベルのところへ α を移動する (長距離移動)
[$\backslash\backslash$ ラベル]	ラベルの定義 (移動コマンドとの後先は問わない)
[…★δ]	δ は…という修正に対するコメント
[★δ]	修正はない, コメントのみ
~c	修正記号 c をエスケープする

は TEX におけるコマンドの始まりを示す文字であるなど，TEX の環境ではエスケープが必要な文字ばかりで繁雑になるからである．筆者が通常使用している各コマンドをこの方法で書けば,

defparentheses 【 】
defdelete /
defescape ~
defcomment ;

となる．

この記号で書いた真鶫道方式の (いささかしつこくはあるが) 修正入り原稿を最後につけよう．私が書いた論文の書き出しを，文章は違えど，内容はそのままに校正したように捏造してみた．

> レイトレーシング法を【高速化/並列化】する際には画面を分割し，それぞれの【/分割された】部分画面を【 PE /プロセッサ】に割り【会てる/当てる】方式が良く知られている．しかし【この方法/レイトレーシング法】では物体の存在する空間【で/内において】，レイ【は/が】反射【/屈折】を繰り返すために空間内のどこにレイが飛んでいくか【計算するまでわからない/をあらかじめ知ることはできない】．したがって，各プロセッサは全ての物体にアクセス可能でなくてはならない．さらに，物体へのアクセスパターンは複雑なシーンでは【非常に/】不規則であることが知られており，【/ある分散共有メモリ型計算機において 2 次】キャッシュのヒット率が 3 割【/ 以下であった】という報告もある．
>
> そこで本研究では空間を分割し，個々の空間に物体を割り当て【える/ることで並列化を行う】．この【方法/手法】は【簡単に思いつくもの/ straightforward な方法】なために【試みは多かったが/これまでに多くの試みがなされたが】，負荷の偏りがあまりに【多きい/大きい】という問題のためにほとんど利用され【なかった/ていない】.【/これは】通常，空間中に物体【が均一に存在することはない/が偏って存在することが原因である】.【/たとえば】部屋であれば床や壁付近に物体が存在し，部屋の中央には空間がある．そのため，空間を多数に分割してそれをプロセッサに割り当てても【何

もない/空虚な】空間を担当するプロセッサが多数を占め，効率が上がらない．本研究の貢献はこの問題を解決したことにある．それは，

修正前と後の原稿をそれぞれ示す.

- ［修正前］ レイトレーシング法を高速化する際には画面を分割し，それぞれの部分画面を PE に割り会てる方式が良く知られている．しかしこの方法では物体の存在する空間で，レイは反射を繰り返すために空間内のどこにレイが飛んでいくか計算するまでわからない．したがって，各プロセッサは全ての物体にアクセス可能でなくてはならない．さらに，物体へのアクセスパターンは複雑なシーンでは非常に不規則であることが知られており，キャッシュのヒット率が 3 割という報告もある．

  そこで本研究では空間を分割し，個々の空間に物体を割り当てえる．この方法は簡単に思いつくものなために試みは多かったが，負荷の偏りがあまりに多きいという問題のためにほとんど利用されなかった．通常，空間中に物体が均一に存在することはない．部屋であれば床や壁付近に物体が存在し，部屋の中央には空間がある．そのため，空間を多数に分割してそれをプロセッサに割り当てても何もない空間を担当するプロセッサが多数を占め，効率が上がらない．本研究の貢献はこの問題を解決したことにある．それは，

- ［修正後］ レイトレーシング法を並列化する際には画面を分割し，それぞれの分割された部分画面をプロセッサに割り当てる方式が良く知られている．しかしレイトレーシング法では物体の存在する空間内において，レイが反射屈折を繰り返すために空間内のどこにレイが飛んでいくかをあらかじめ知ることはできない．したがって，各プロセッサは全ての物体にアクセス可能でなくてはならない．さらに，物体へのアクセスパターンは複雑なシーンでは不規則であることが知られており，ある分散共有メモリ型計算機において 2 次キャッシュのヒット率が 3 割以下であったという報告もある．

  そこで本研究では空間を分割し，個々の空間に物体を割り当てることで並列化を行う．この手法は straightforward な方法なためにこれまでに多くの試みがなされたが，負荷の偏りがあまりに大きいという問題のためにほとんど利用されていない．これは通常，空間中に物体が偏って存在することが原因である．たとえば部屋であれば床や壁付近に物体が存在し，部屋の中央には空間がある．そのため，空間を多数に分割してそれをプロセッサに割り当てても空虚な空間を担当するプロセッサが多数を占め，効率が上がらない．本研究の貢献はこの問題を解決したことにある．それは，

ここまで細かな修正となると書き直した方が早い位であるが，このような修正をもし conventional な手法で行うのは困難であることが想像できることと思う.

2 実装までの歴史的経緯

冒頭に書いたように筆者が真鵺道に出会ったのは，第 39 回のプログラミングシンポジウムであった．それまで遠隔地での校正を行う際に筆者はいくつかの手法を工夫していた．筆者が遠隔地における校正を行なった最初の経験は，博士課程の 2 年の時である．当時，筆者の担当教官がアメリカに 1 年間出張するということになり，やむを得ずアメリカと日本で論文を互いに校正することになった.

先生が出国されるまでは電子メイルがあるからやりとりは一瞬で簡単だろうなどと多寡をくくっていたのであるが，この校正は困難を極めた．国際会議への投稿時には，論文の締切が近づく上に締切近くには次々と疑問や修正すべき点，あるいは新しい実験などを繰り返すというお決まりのコースをとるようになり，なかなか内容について収束しない．一つのアクションごとに毎回タイムラグがあるのだ．経験してみればわかるがこれはかなりきついことである．その上，インタラクティブ性をさらに低下させる要素として，時差による互いの生活周期のずれがある．Text に ^ で下線を書いて修正箇所を示す conventional な校正方法は修正箇所が伝わりにくい上に，その修正すべき箇所を原稿に手動で反映させていくために誤りなどが入り込んでしまい，修正が進まないという事態も起こった．完全に修正した原稿を送れば良いと考えたが，その場合にはどこを修正したかの指示が難しい．first author が全ての責任を負えば良いという考えもあろうが，筆者は学生であり指導を必要としていた．これまでに述べた当時試みた校正方法とその欠点をまとめてみると，次のようになる．

- **電子メイルで Text に ^ で下線を書いて修正箇所を示す conventional な校正方法**
  修正箇所の入力や，修正する際の指示にあいまいさが入るための誤りの問題
- **修正した原稿を国際電話で Fax**
  手動で戻す際の誤り．Fax 用紙がつまってしまったが時差があったために同期が遅れてしまうこと．解像度の悪さなどから文字が判読できないなどの問題．電子的情報でないのでサーチなどが困難
- **phone による指示**
  場所の指定などの問題，完全に同期する必要があること．回線の状態の悪さが如実に反映されることなど
- **電話による指示**
  画像情報の欠如．修正箇所の指示の難しさや，完全な同期が必要であること
- **diff : 完全な原稿だけを送信し，以前のものと diff をとる**
  diff はエディタの整形の影響を受けるために擬似的な差 (タブやスペースの変更や行末の単語が次の行に移ったなど) に弱い．コメントなどを付加することが難しい．互いに修正箇所の確認を行わないような盲目的な修正ならば問題ないが，校正では通常そのようなことはないだろう

などがある．郵送は速度の問題から試みなかった．結局，電子メイルでの conventional な校正方法を繰り返した．バージョン管理も問題であり，二箇所で同時に修正した場合の conflict の解消などは難しい問題であった．(当時は CVS の存在を知らなかった．) 最近，当時一緒に論文を書いていた後輩と会う機会があったが，二人で当時をふりかえってみると，すごいストレスだったと意見が一致した．

個人的な問題であるが，筆者は FAX をほとんど使用しないために FAX は苦手である．ある時には，論文の締切近くに友人の結婚式を梯子するという事態が発生した．友人の家に宿泊しながら，友人の家の FAX を借りて大学にいる先生と論文の校正を行なったのである．

筆者は，その時に友人に FAX を送るから原稿をコピーしてくると言って，近くのコンビニエンスストアで修正原稿のコピーをとった経験がある．FAX を使って送ると原稿が消え，それが相手先に再構成されると筆者は考えていたのだ．結局友人には StarTrek の見すぎだと言われた．どうも FAX は苦手である．

プログラミングシンポジウムに行く直前には，以前研究を共同で行なっていた後輩が修士論文を見て欲しいと連絡をくれた．電子的な原稿は ftp で入手した．conventional な校正手法で校正を行なったが，紙の上で読んで修正するのに 3 時間，それを入力してメイルで送信するのにさらに 4 時間かかり，これはたまらないと思ったものである．特に修正箇所を間違えないようにする指示が難しいことを感じた．

その直後に筆者は真鶫道に出会ったのである．筆者が普段使用しているエディタが Emacs であったことから，Emacs 上への実装を考えることはごく自然であった．しかし，大きな問題が一つあった．私は Lisp でプログラムを組んだ経験が一度もなかったのである．一度東北大で関数型言語の授業を受けたことがあり，その時に佐藤雅彦先生から紹介された (が読むのを挫折していた) 本 [6] だけが手元にあった．まずは Lisp ではどうやってプログラムを書くのかから学ぶこととなった．ある範囲に色をつけるにはどうするのか? 文字列を探すにはどうするのか? 筆者は Lisp でループすら書けない状態であったが，慶應の中西研出身の前田氏が同僚というだけの根拠で実装は始まり，現在に至っている．

3 真鶫道の emacs lisp 上の実装

3.1 機能概要

真鶫道の規則は基本的に 2 つしかない．今回筆者が実装したのはそのうちの一つである．したがって，現在の実装においては真鶫道の規則は 1 つしかない．それは

- `[A/B;C]`

のパターンである．この意味は，

- A を B に置き換える．
- C はコメントである．

である．もう一つのパターンは，`[A|B|C;D]` である．こちらは実装していないため，詳細は文献 [3] に譲る．

このような記法の文字を固定することは当初の目的に反する．固定してしまえば文書によってはエスケープが多量になり，入力の省力化，人間に対する表現力の低下があるからである．たとえば，文献 [3] において提案されている記号は TEX のコマンドと重なるために TEX を用いる場合には好ましくない．

文献 [3] ではこの問題に既に気がついており，カスタマイズの方法についての提案がある．今回はほぼそれに従った実装を行なった．まずは実装の説明をするための言葉をいくつか定義し，実際の実装を見ていくことにする．

3.2 真鶫道修正記号定義コマンドと真鶫道コマンド

まず以下の言葉を定義する．

- 平文：通常の文
- 真鶫道コマンド：校正すべき場所と校正方法 (置換など) を示す文字列

- 真鶴道修正記号定義コマンド (def コマンド) : 真鶴道コマンドを定義する文字列

ここでの名前は adhoc なものであるためにもう少し考える必要があると思うが，実装にあたってはこのような名前をつけていた．

たとえば，`[A/B;C]` における，‘[’, ‘]’, ‘/’, ‘;’ のような文書を実際に操作する記号を真鶴道コマンドと呼ぶ．またこれらの真鶴道コマンドを定義するための文字列を真鶴道修正記号定義コマンドと呼ぶ．真鶴道修正記号定義コマンドは ‘def’ で始まる文字列から成るために簡単のために ‘def コマンド’ と呼ぶ場合もある．今後は ‘def コマンド’ と呼ぶことにする．この def コマンドによって真鶴道コマンドが決定される．

‘[’‘]’ という真鶴道コマンドに囲まれた範囲が真鶴道コマンド中の文であり，そうでない部分が平文である．真鶴道コマンド中の文のことを真鶴道文とも呼ぶことにする．

def コマンドは

- 電子メイルで利用可能な記号が少なく，文書により利用する真鶴道コマンドを変化可能にすることで可搬性を増す．
- 固定した記号を用いた場合にはエスケープを多用せざるを得ない文書が存在し可読性が下がることを防ぐ．たとえば TEX の文書中では‘[’‘\’‘]’ のような記号を真鶴道コマンドとすると可読性が低下する．これを防ぐ．

ためにある．

def コマンドは真鶴道で校正する対象の文書中のどこに現れても良いが行頭に位置する必要がある．これは文中に def コマンドと同じ文字列が存在した場合でも区別を可能とするためである．通常は def コマンドは文書の冒頭に置いて真鶴道文書であることを示すと共に，文書とそこで用いる記号の対応を示すべきであろう．

各真鶴道コマンドは，基本的に真鶴道修正記号定義コマンドから先頭の def を除いた名前を持っている．たとえば，真鶴道における修正前と修正後の部分を交換するコマンドは swap コマンドである．この swap コマンドの記号を★にするには，

```
defswap ★
```

という行を文書の行頭に記す．この定義のみが行なわれた場合には他の記号はデフォルトの記号となり，

```
[もとの文書★修正後の文書]
```

により「もとの文書」が「修正後の文書」に置換される．

def コマンドが文書中に存在しない場合には，真鶴道コマンドとして文献 [3] で竹内と秘密結社大日本清談会が提案している default の記号 (表 1) が利用される．ただし,「★」だけは突然出てきた感のする 2 バイト文字なので「;」を用いた．

今回実装した def コマンドの一覧を表 2 に示す．ここで一つ告白せねばならないが，筆者の勘違いにより，defdelete コマンドが defswap コマンドとして間違った名前で実装されている．これはやがて訂正するつもりである．

表 2 の `deforder` について補足する．真鶴道のコマンド内部の並びは表 2 の記法で `[first/last]` として順序づけられている．ここで `first` の部分を修正後の文書と新しいみなすか，あるいは `first` 部分を訂正前の文書とみなすかのカスタマイズのために `deforder` がある．この指定は，(newer, older) と (first, last) の組み合わせによる．したがって，その組み合わせは

表 2: 真鶸道の def コマンド

def コマンド	default 値	定義するコマンドの内容
defLparentheses	[	校正範囲の始まりを示す
defRparentheses	]	校正範囲の終了を示す
defswap	/	校正範囲中でこの記号の前を消去し後の部分に変換する
defescape	~	校正範囲中で真鶸道コマンドをエスケープする
defcomment	;	校正範囲中でこの記号から後をコメントみなす.
deforder	newer-last	swap コマンドによって交換される文のどちらが新しい文かを示す

- newer-first
- newer-last
- older-first
- older-last

となる．default 値の newer-last というのは，たとえば `[A/B]` という真鶸道コマンドを含む文において修正後の新しい (newer) 文は B (last) 部であることを意味している．もし，新しい方を A 部として定義したい場合には，newer-first (あるいは older-last) と書く．

3.3 変更部分

3.3.1 defparentheses の分離

文献 [3] においては，defparentheses によって真鶸道文の範囲を指定するコマンドを指定する．しかしこの手法では真鶸道コマンドの指定においてその「始まり」と「終わり」の記号を分離しなくてはならない．実際にはそれは難しくないのであるが，最初にコーディングをした時の Lisp に対する私の skill 不足もあり，全ての def コマンドの引数の数を統一するのが良いのではないかと考え，defparentheses を defLparentheses と defRparentheses の 2 つに分離した．

3.3.2 manued 文書範囲指定

`manued.el` では，真鶸道の文書として扱われる範囲を指定できる．`manued:doc-begin-pat` がバインドされている文字列と `manued:doc-end-pat` がバインドされている文字列の間しか真鶸道コマンドは有効ではない．ただし，これらが nil の場合には文書の最初から最後までが真鶸道コマンドの有効範囲となる．

3.3.3 未実装部分

文献で提案されているもので実装していないものは，

- 前後の入れ替えのコマンド `[A|B|C]`

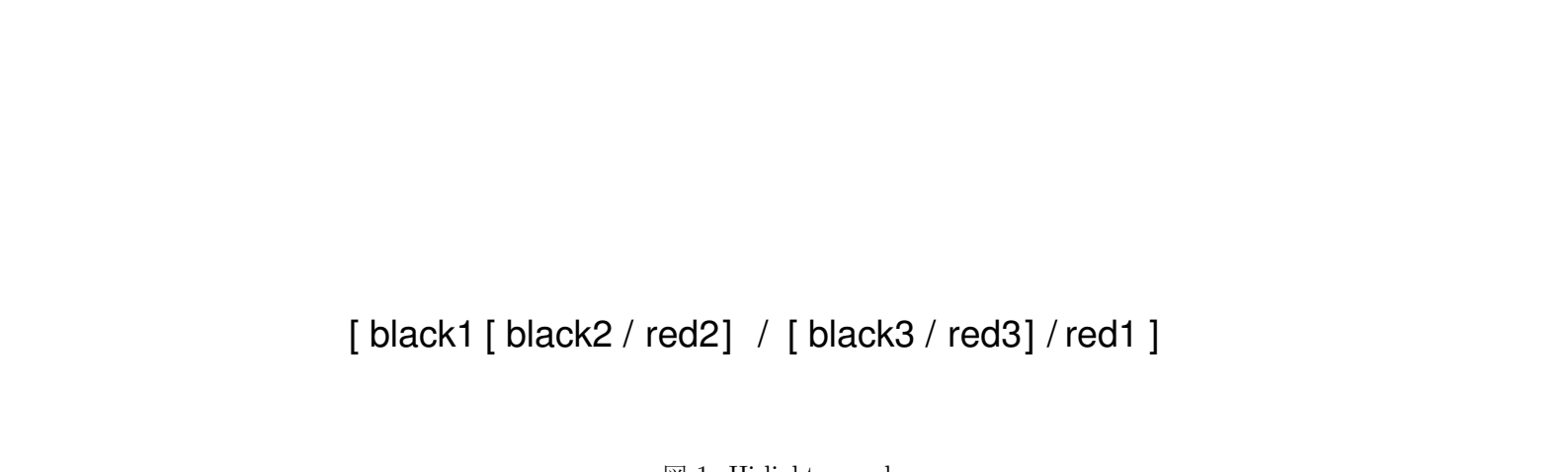

図 1: Hi-light sample

- ラベル
- 大域的移動

である．前後の入れ替えのコマンドが存在すると実装上の解釈に問題がでるのではないかと考えて実装していない．たとえば，前後入れ替えのコマンドと削除のコマンドが同時に現われた場合にどちらが先に適用されるべきかなどの解釈を決められなかった．また繁雑であると思った．結局どちらが基本的な操作かを考えた時に，消去と挿入の可能な `[A/B]` を採用した．

3.4 文書の取り出し部分の実装

この節からは実際の例題を使用して説明を行う．例として図 1 の真鶴道文を利用する．おおまかな処理は次の通りである．

1. 初期化として，真鶴道コマンドの適用可能な範囲を探し，次に def コマンドを探して真鶴道コマンドを決定する．
2. 文書の最初に戻る．
3. 真鶴道コマンドを順に探し，削除と挿入動作を行う．途中に真鶴道コマンドがでてきたら再帰的に削除と挿入動作を行う．
4. 文書の終わりに来たら終了．

実際にはエラー処理が含まれている．たとえば defLparentheses に対応する defRparentheses が存在しない (括弧の数が合わない) 場合などには大域的な脱出を行い，エラーの発生した場所で停止する．また，def コマンドの定義に矛盾がある場合，たとえば，右括弧と左括弧を同じ文字列で定義するなどを検知して停止する．また，サーチ時にはエスケープ処理を行う．

また，文書の取り出し部分は基本的な部分であり，素直な再帰を利用して書くことができる．その様子を図 2 に示す．ここでの level とは再帰の深さのレベルを示している．この例では，各レベルの処理の詳細は

1. level 2 の処理
   `black2` → `red2` の書き換え及び，`black3` → `red3` の書き換えの処理が行なわれる．結果，

   ```
   [ black1 red2 / red3 red1 ]
   ```

   という真鶴道文が生成される．
2. level 1 の処理
   `black1 red2` → `red3 red1` の変換が行なわれ，結果，

```
red3 red1
```

と変換される.

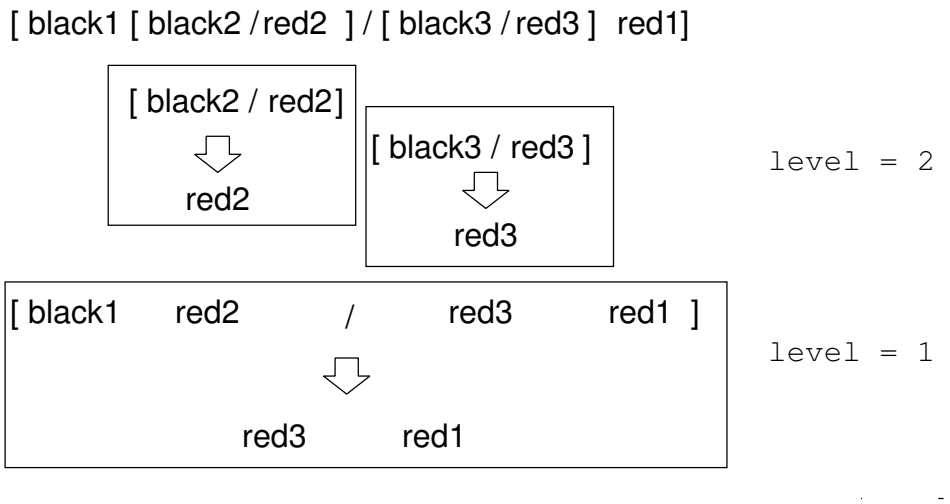

図 2: Algorithm : Show newer document

3.5 色つけ部分 (manued:hilit) の実装

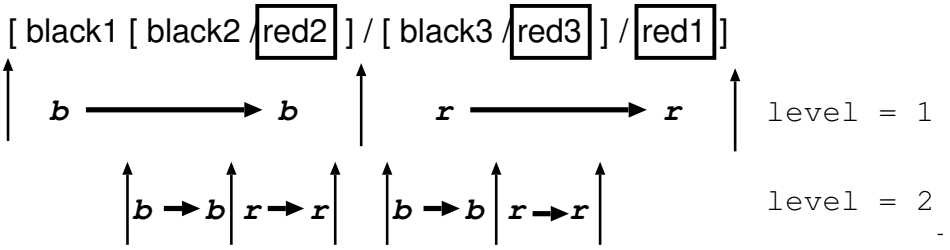

図 3: Algorithm : highlight-newer

色を付加するルーチンは，文書の取り出しルーチンと共有するように設計を行いたかった．これは，これらの機能がほとんど同じことと，保守の容易さからである．しかし，残念ながらこの 2 つのルーチンはコードの共有を行っていない．なぜなら，色つけ作業は副作用のある作業であり，単純な再帰では思ったような動作が難しいからである.「思ったような動作」とは，図 1 に示すような真鵺道文の場合，black の部分を黒に，red の部分を赤に塗るという動作である.

真鵺道ではコマンドの入れ子を許しているので，色を付ける時と文書を取り出す場合，その処理が問題となる．入れ子の処理の順番の規定が無いとプログラミングシンポジウムでは質問があり，筆者はその時の答えをはっきりとは覚えていないが，内側を優先に処理するような話になったと思う．(チャーチ・ロッサー性の話があったことは覚えているのであるが，) 筆者も内側優先の方が自然であると考えたので本実装では内側から処理をしている.

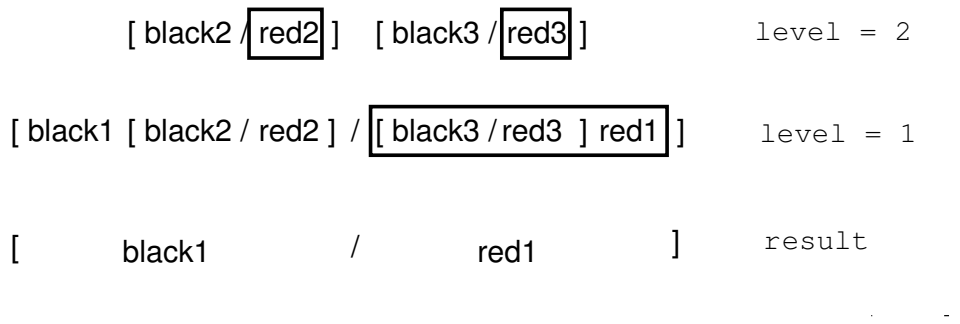

図 4: Fail sample : hilight-newer

Emacs lisp で色をつける方法は前田氏の助言を得て，mule に付属していた `hilit19.el` 中の `hilit-region-set-face` を使うことにした．この選択自身に問題があるのかは不明である [1]．この `hilit-region-set-face` は，

```
hilit-region-set-face: a compiled Lisp function.
(hilit-region-set-face START END FACE-NAME &optional PRIO PROP)

Highlight region from START to END using FACE and, optionally, PRIO.
The optional 5th arg, PROP is a property to set instead of 'hilit.
```

と説明される関数であり，`START` から `END` まで `FACE` の属性を付加する．つまり，ある連続領域に色を塗ることができる．また，適用する順番があり，上書きを許している．つまり後から書いた方が有効である．これ自身は妥当な設計であると思うが，真鵜道の文を素直に処理しようとすると問題が発生する．

処理結果としては図 1 に記された通りに色が付加されて欲しい．新しい文を取り出す (つまり，`red3`, `red1` の部分を取り出す) 場合には，前節で説明したように 2 レベルの再帰処理を行う．図 3 には，各レベルの処理を図示した．図中の 'b' は `black` を示し，'r' は `red` を示す．b → b の間は黒い色がテキストに付加される．r → r は赤の色をつける範囲である．この処理が再帰的に行われる．

しかし，そのまま内部処理優先で色を付加していくと，図 4 のように，level 1 の処理が最後に行なわれるため，最上位のレベル (level = 1) の処理しか反映されず，望みの結果を得ることができない．

処理する範囲の決定にはやはり再帰的な処理が素直であるため，範囲の決定を再帰的行い，その範囲を stack に積んでおいて処理する方法などを考えたが，この時点で文書の取得ルーチンと色の付加ルーチンを共有することは困難であると判断し，色付けルーチンと文書取得のルーチンを分離した．

本実装では，図 5 に示すように，色をつける部分は，大域的なポインタ (GPM : Global Point Marker) を一つ用意し，状態の変化ごとにそのポインタから現在の位置まで色を付加するよう実装した．現在の色は何かという部分を求める方法は上述の再帰的な手法を用いた．具体的には以下のようにする．

[1] 例えば，和田が作成した `manued` 関数 [1] では，`put-text-property` を利用している．

1. 真鶴道コマンドの始まりに来たらまずは付加する色を black とする．この時，最初の位置に GPM をセットしておく．

2. 次に出現する真鶴道コマンドをサーチする．

   - defswap 真鶴道コマンドであれば，現在の色で defswap 真鶴道コマンドまでを着色し，GPM を defswap 真鶴道コマンドまで移動する．そして現在の色を red とする．
   - もし新たな真鶴道文の開始 (defLparentheses) であれば，現在の色で defLparentheses の真鶴道コマンド位置までを塗り，GPM をセットする．そして，再帰する．
   - もし真鶴道コマンドの終了 (defRparentheses) であれば，GPM と終了位置までを現在の色で塗り，再帰から帰る．

このように，再帰的処理によって色の状態を保ちつつ，色を塗る範囲は順にスキャンしていくアルゴリズムで実装した．

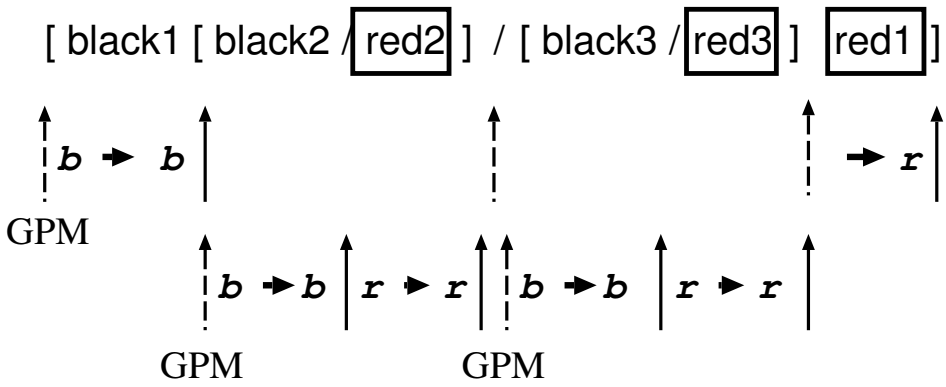

図 5: Implementation of hilight-newer

3.6 真鶴道 Tips

ここでは，より利用しやすくするためにちょっとしたコマンドをいくつか実装しているので，それについて概説する．

- 真鶴道コマンド部分のサーチ
  どの位置を修正したかを素早く知ることができれば校正には有益である．現在の defLparentheses の示す真鶴道コマンドの位置を M-n, M-p で前後に移動することが可能である．

- 入力補助
  マークした部分からのリージョンに対して真鶴道のコマンドを適用する，あるいは，真鶴道形式のコメントとするコマンドを追加した．例えば真鶴道コマンドの文字列が複雑な場合なども真鶴道コマンドの挿入が容易となる．

- TEX との親和性
  defLparentheses などの def コマンドは文章中に出てくることはないと考えたが，行頭に書

かねばならないという制約をつけてより安全にした．そのため，TEX ファイルを真鶴道文書とすると，コンパイルのためにいちいち def コマンド群を消去するか，コメントアウトしなくてはならない．そのため，def コマンドの前にいくつか文字を置くことが可能なように実装した．この文字を TEX のコメントを示す % などとすれば，真鶴道文書の TEX ファイルをそのままコンパイルできる．

- minor-mode
  `manued.el` には minor mode が存在する．manued-minor-mode は LATEXmode などと共用することが可能である．これによって TEX のコマンドの入力の容易さを保ちつつ，真鶴道の編集が可能になった．

- Menu
  現在の `manued.el` では真鶴道の規則が一つしかないとは言え，新しい文書の抽出や，古い文書部分を hilit させたいなどのコマンドを覚えておけない場合のために menu を付加した．

- 差分を知る
  前回 (第 40 回) のプログラミングシンポジウム会場で幾人かの方に真鶴道を御覧に入れた所，変更後の文書と現在の真鶴道文書とを比較したいという要望があった．つまり，校正後の文書を取り出した時点で直った部分だけを知る方法はないかということである．これについては，消去した部分の表示などが難しいなと考えていたが，よく考えると差分をとれば良いのであり，ediff を利用できる．ediff 自体よくできているので，私は差分を見たい場合には ediff の利用をおすすめする．

- バージョン管理
  バージョン管理も真鶴道の問題である．しかし真鶴道自身はある snap shot の変更部分を記録する以上のことをすると繁雑になると思う．しかし，やはりバージョン管理はしたい．ということで筆者は現在 RCS を利用している．真鶴道文書を作成し，check in する．そして新しいバージョンを取り出しで再び check in するということをやっている．また，校正の情報をわざわざ書き込んで E-mail で送るということは複数人間でバージョン管理をするということにほかならないので，現在は，もし次の論文を遠隔地で共同執筆する場合があればCVS を導入することを考えている．

4 例

ここではいくつかの例題を示す．これらの例題はレジメからはなかなか伝わらないと思うため，PTT ではデモンストレーションを行う予定である．また，これまでの説明で通常のものはおおよそ了解して頂けたと思うので，ちょっと変化に富んだ例を示す．実際にこのような例を使うことはないだろう．

4.1 defLparentheses を文字列にした場合

```
%%defLparentheses こ
%%defRparentheses .
%%defswap /
```

```
%%defcomment      ;
%%defescape       ~
%%deforder        newer-first
```

「defLparentheses」として,「真鵺道」「ど」「こ」などを割り当てる．また,「defRparentheses」としては「.」を用いる．

真鵺道の話を今日の PTT でやるって本当ですか.
どうも最近そういうのをしていて自分の才能の無さを感じるんだ.
この前は多田先生が来てなかった．やはり飲み会が他にあったんでしょうか.
真鵺道のユーザは 3 人位しかいないらしいって飲み会の話題にでてたよ.
この次の飲み会，つまり今日の飲み会に行けばきっとその話になると思うよ.
どうせならば一回何かで本当に試してみたいね.
真鵺道を実装した本人は 6 台のマシンに install したって言っていたよ．だからもう少しユーザはいるんじゃないの.
どっかへ留学するというのはいかがかな.
こんかいの PTT は友人の奥さんにばったり出会ったりしたなかなか驚きだったな.

4.2 defLparentheses を ‘defLparentheses’ としてみよう

```
defLparentheses defLparentheses
defRparentheses defRparentheses
defswap /
defcomment      ;
defescape       ~
deforder        older-first
```

左近「岡部君,『defLparentheses 名のるほどのものではございません/匿名希望 defRparentheses』さんからよ 」
真美「あ，フクちゃんから? 」
岡部「フクちゃん? 」
真美「覆面のフクちゃんよ.」
岡部「defLparentheses 千春/千秋 defRparentheses さんでしょ... はい，電話かわりました」

金田一春彦と defLparentheses イギンズ/ヒギンズ defRparentheses 教授の関連について

「駅前に I-node の広告があったんで驚いて近付くと，
defLparenthesesI-model/I-modedefRparentheses だった.」

4.3 適用の副作用を使う例

真鶫道文書の範囲を指定しない場合には def コマンドも真鶫道コマンドの影響を受けるという副作用を使い，次々に真鶫道コマンドの定義を変更することができる．

```
【
%%defLparentheses        【
%%defRparentheses        】
%%deforder       older-first
%%defswap        ,
】

「
%defLparentheses          「
%defRparentheses          」
」

『
defLparentheses           『
defRparentheses           』
』
```

「コマンドにはサーチの順番がある

登場人物　　織部（課長）
　　治虫（社員）
　　佳子さん（秘書）
　　村人（通りすがり）」

村『今日のコンサートのある文化なんとか会館を御存知ですか』
治「さあ、文化なんとかって聞いたことある」
佳【文化振興会館じゃないでしょうか】
織【確かギター… だったかな，のコンサートです】

5 おわりに

今回は真鶫道についての説明とその実装例の manued.el について概説した．真鶫道は遠隔地にて共同執筆を行う場合に有益である．筆者は，論文の校正や，あるいは MPI の仕様の翻訳などにおいて使用し，通常の方法よりも効率良く校正が可能になったと考えている．

筆者は現在の manued.el に特に不満はないため，今後の真鶫道についてのビジョンは特に持っていない．いくつか判明している Bug fix を行う必要があると考えている位である．また，この manued.el は私が生まれて始めて作った Lisp のプログラムであるため，何か大きく変更する場合には書き直してみたいという考えはある．まだ私は ‘ と , を使えない Level 0 の Lisp 初心者であ

るので，次に書き直す際にはマクロを使えるようになっていれば嬉しい．

また，共同研究を行っている方からは，「確かに真鶴道は校正時に間違いがなく，しかもサーチなどがあるので校正の効率が向上する．しかし，校正をする時に真鶴道コマンドを意識するのはなかなか面倒である．」という意見を聞いている．そこで，2 つの文書から真鶴道文書を生成することはできないかと考えている．コメントなどは `yamauchi@is.uec.ac.jp` までお願いする．また，これから Lisp をやってみようという方のために，今回参考にした本を参考文献に挙げておく．この他にも中西正和著の Lisp 入門や，Abelson と Sussman の Structure and Interpretation of Computer Programs などを紹介されたが，それらはこれからゆっくり読むつもりである．

6 Appendix

ここでは真鶴道のインストール，真鶴道コマンドとカスタマイズ手法について概観する．

6.1 `manued.el` の入手方法

`http://www.archi.is.tohoku.ac.jp/~yamauchi/otherprojects/manued/index.shtml` から入手可能．配布は条件は GPL に基づくフリーソフトウェアとする．

6.2 Install 方法

`manued.el` は Emacs Version 19 以降に依存している．Emacs Version 19 以降のものあるいは，それに基づいた Mule を利用すること．作者が `manued.el` を開発した環境は GNU Emacs 19.28.1, Mule Version 2.3 (SUETSUMUHANA) である．

1. `manued.el` を load-path の通っている directory に置く
2. .emacs に次の行を追加
   - `(autoload 'manued-mode "manued" "manuscript editing mode" t)`
   - `(autoload 'manued-minor-mode "manued" "manuscript editing minor mode" t)`

6.3 `manued-mode` におけるコマンド一覧

真鶴道文書をバッファに読み込み，`M-x manued-mode` で真鶴道モードに入る．以下のコマンドが利用可能となっている．キーのバインドなどは今後の検討課題である．

- `M-x manued-mode`
  真鶴道 major mode に入る
- `M-x manued-minor-mode`
  真鶴道 minor mode に入る．例えば LaTeX mode などで major mode を変更すると LaTeX mode のコマンドが使用できなくなるが，manued minor mode に入れば LaTeX mode のコマンドを使用可能なまま真鶴道が使える．ただし，キーバインドが他のモードとぶつからないよう多少繁雑になる．

- `M-x manued:version`
  真鶴道のバージョンを知らせる
- `M-x manued:hilit-older`
  元文書部分を hi-light する
- `M-x manued:hilit-newer`
  修正後の文書部分を hi-light する
- `M-x manued:show-older-in-manued-buffer`
  整形し，元文書を抽出する
- `M-x manued:show-newer-in-manued-buffer`
  整形し，修正後の文書を抽出する
- `M-x manued:insert-swap-command`
  マークした所から現在のポイントに対する swap コマンドを挿入する
- `M-x manued:next-l-parentheses`
  直前の真鶴道コマンドの始まりをサーチし移動する
- `M-x manued:previous-l-parentheses`
  次の真鶴道コマンドの始まりをサーチし移動する
- `M-x manued:eval-last-manuexp`
  直前の真鶴道コマンドを整形する

6.4 カスタマイズ

基本的なカスタマイズは変数の default 値を変更することにより行われる．変更できる変数は以下の通りである．また，関数 `‘manued-mode’` は最後に `(run-hooks ’manued: mode-hook)` を実行するため，この mode に依存したカスタマイズも可能である．(が，現在は可能であるというだけかな．)

6.4.1 修正記号定義コマンドが無い場合のデフォルト値

この真鶴道コマンド記号のデフォルト値は文章中に修正記号定義コマンド (def*) が出現すると上書きされる．それらの指定が無い場合に採用される記号である．

- `manued:l-parentheses-str`
  真鶴道コマンドのはじまりの記号の default 値 (defalut ”`[`”)
- `manued:r-parentheses-str`
  真鶴道コマンドの終わりの記号の default 値 (defalut ”`]`”)
- `manued:swap-str`
  スワップコマンドの記号の default 値 (defalut ”/”)

- `manued:comment-str`
  コメントのはじまりの記号の default 値 (defalut ";")

- `manued:escape-str`
  エスケープ記号の default 値 (defalut "~")

- `manued:order-str`
  スワップコマンドの適用順 "older-first", "newer-last" あるいは "older -last", "newer-first". 真鶴道のコマンドは `[first/last]` の形をしており，どちらを元の文書 (older) とみなすかを指定する．

ただし，もしこれらに同一の記号がわりあてられた場合には多分正しく動かないだろう．

6.4.2 真鶴道文書の範囲指定

manued.el は真鶴道の文書の範囲を指定することができる．ただし，これらの文字列が存在しない場合には，真鶴道文書の初まりは文書の最初に，終わりは最後とみなされる．

- `manued:doc-begin-pat`
  真鶴道文書の初まりを示す文字列を入れておく `(defalut "-*-*-BEGINMANUEDOU -*-*-")`

- `manued:doc-end-pat`
  真鶴道文書の終わりを示す文字列を入れておく `(defalut "-*-*-ENDMANUEDOU -*-*-")`

6.4.3 Hi-Light の色

色は swap コマンドの first 部分を hi-light する場合と last 部分を hi-light する場合でどの色にするかを color-list の値によって決めている．color-list は 4 つの色を指定したリストであり，`'(swap-first-color swap-last-color comment-color command-color)` となっている．それぞれ,

- `swap-first-color`
  `[first/last]` の first 部分の色

- `swap-last-color`
  `[first/last]` の last 部分の色

- `comment-color`
  `[first/last;comment]` の comment 部分の色

- `command-color`
  コマンドの記号 (default では `'['`,`'/'`,`';'`,`']'`) の色

である．たとえば,

```
'(red blue firebrick-italic gray60)
```

このようなリストとなっている．また，デフォルト値を設定している変数は次の通りである．

- `manued:first-hilit-color-list`
  first 部分を hi-light する場合の色リスト
- `manued:last-hilit-color-list`
  last 部分を hi-light する場合の色リスト

6.4.4 その他のカスタマイズ

- `manued:mode-syntax-table`
  syntax-table の default 値 (default text-mode-syntax-table)
- `manued:mode-abbrev-table`
  abbrev-table の default 値 (default text-mode-syntax-table)
- `manued:mode-map, manued:minor-mode-map`
  キーマップの default 値
- `manued:is-swap-command-with-comment-on`
  swap command 内部でコメント文字を挿入するかしないかを制御する
- `manued:is-auto-insert-headder`
  値は '(auto-insert query-when-insert) というリスト．バッファに defcommand が存在しない場合に defcommand を挿入するか．
  - `auto-insert` が `t` の時，manued-mode に入った時にそのバッファに defcommand が存在しない場合に defcommand を挿入する．
  - `auto-insert` が `t` の時に `query-when-insert` が `t` ならば挿入するかどうか尋ねてくる．
- `manued:ask-if-formatted-buffer-is`
  既に整形済みの buffer が存在した場合に尋ねるかどうか．`t` で尋ねてくる
- `manued:defcommand-head-str-list`
  真鶴道定義コマンドの先頭につけることが可能な文字列のリスト．TEX などのコメントにしておく．Tips を参照．

6.5 Bugs

現在 (1999 年 4 月 17 日 (土)) エスケープ関係にバグがあることが判明している．時間ができたら直す予定．

謝辞

プログラミングシンポジウムで利用した電子的原稿を Web に掲載する許可を下さった真鶴道の考案者，電気通信大学の竹内郁雄氏に感謝します．電気通信大学の前田敦司氏には Lisp をまったく知らなかった私の質問に辛抱強く答えて下さり，感謝いたします．

参考文献

[1] Personal communication with WADA Eiichi, January 25 1999.

[2] Robert Krawitz, Bil Lewis, Dan LaLiberte, Richard M. Stallman, and Chris Welty. *Emacs Lisp Info : edition 2.4*.

[3] 竹内 郁雄. 電子メールで原稿を修正する方法 — manuscript editing (manued, 真鶴道) を目指して —. **第 *39* 回プログラミングシンポジウム**, pages 61–68, 1 1998.

[4] 佐藤雅彦ら. Skk version 9.6, Feb 1996.

[5] 青柳龍也. *Unix* **短編シリーズ** *Emacs Lisp*. 工学図書, 1997.

[6] 竹内郁雄. **初めての人のための** *LISP*. サイエンス社, 1986.