1. HRO-自動的にハイフネーションする英文清書システム

京都大学 数理解析研究所　　中島玲二
Reiji Nakajima

一つの論文を完成させるまでには，手書きの原稿から始まって，プレプリント，投稿，査読と修正の繰り返しはつきものである．大切な論文であればあるほどこの期間は長くなりがちである．特に情報学の分野でシステム開発をともなう研究では論文の内容もシステムの進化に合せて発展的に変わる．あるいはタイプしてもらったとたんに大々的に書き変えたくなってタイピストに嫌われたり，投稿直後に誤りを発見していらいらしたり，苦労は絶えない．実際の研究より論文の製作に多くの時間を費すこともめずらしくない．

ところが一昨年，筆者の勤務する研究所の計算機が更新されて事態は一気に好転した．その後の論文作りは，研究室のCRTからEMACSのような便利なエディタで思いつく部分を好きな時に入力し，追加や編集も思うままにできるようになった．手書きの原稿を書く必要は全くなくなったし，スペルを検出するプログラムや清書プログラムRUNOFFを利用して，筆者の研究グループによる論文やマニュアル作りは飛躍的に効率化された．

論文の検討や修正はタイプされ，清書されていない原稿の上ではなかなか思うようにはかどらないものである．かくて査読者の無理な注文もあまり気にならなくなったし，また普段から論文を手もとで暖めて好きなだけいじくれるので，投稿したのちに「こう書けばよかった」とくよくよ後悔することも減った．よい印字品質のタイプライタも購入して万事整ったかに見えたが，昨年になって国際学会のproceeding用の論文を作ることになって問題が生じた．

英文清書プログラム　RUNOFF

RUNOFFのよい解説記事は[1]にあるのでここでは立ち入らないことにする．このプログラムを用いると，ほとんど任意のフォーマットで清書できて大変便利であるが，一番困るのはハイフネーションが不可能なことである．清書例で見てみよう．

```
    The iota system is an integrated man-machine
system for the interactive development,
verification and processing of programs written in
language iota. The language is designed to
support program development in hierarchical
structures with modularity. As a result of
explicitly setting forth the programming
methodology, machine assistance comes to be
possible to a great extent in managing the
development of large scale software systems. On
```

図1．RUNOFFの清書出力例（1）
「右端そろえ」あり

```
    The iota system is an integrated man-machine
system for the interactive development,
verification and processing of programs written in
language iota.  The language is designed to
support program development in hierarchical
structures with modularity.  As a result of
explicitly setting forth the programming
methodology, machine assistance comes to be
possible to a great extent in managing the
development of large scale software systems.  On
```

図2. RUNOFFの出力例（2）
「右端そろえ」なし

いずれの場合も1行の長さは50文字としてある．（1）は行の右端そろえの命令のもとに清書してある．右端は1列になっても単語と単語の間は沢山の空白がとられ，見苦しい．（2）は右端そろえの命令を除いた結果で，右端の見にくさは明らかである．このような現象は単語の長さが平均して大きくなればなるほど起りやすい．プレプリントやマニュアルなら我慢するが，国際学会の proceeding のように投稿論文がそのまま写真印刷される場合は大いに気になる．

HRO

そこで英語の辞典を内蔵して，自動的にハイフネーションしてくれるようなプログラムを考案した．名づけてHRO（Hyphening RunOff），RUNOFFの清書機能でよく用いられ，論文作成に必要と思われるものはほとんど取り入れた．HROによると上掲の英文は次のようになる．

```
    The  iota system is an integrated man-machine
system for the interactive development,  verifica-
tion and processing of programs  written  in  lan-
guage  iota.  The  language is designed to support
program  development  in  hierarchical  structures
with modularity. As a result of explicitly setting
forth the programming methodology, machine - comes
to  be  possible to a great extent in managing the
development of large scale  software  systems.  On
```

図3. HROの清書出力例

当然のこととして，HROの設計では次の点に特に留意した．

(1) 便利であること（なるべくユーザの手間を少くし，使い方を憶えやすいこと．）
(2) 早いこと（「締め切りは来週だから，明日にも速達で発送しなくてはならない」，清書しては検討し，修正してはまた清書という作業の能率を上げたい．）
(3) 筆者がひとりで作成し，保守できること（3週間以内に完成すること——これはHROの作成を思い立った時の筆者の都合である．）
(4) RUNOFFのユーザを奪えること（したがって清書命令など新しく憶えなおす必要のないように設計した．）

辞典の生成

ＨＲＯは論文清書時にハイフニングの必要が起ると，当該の単語の音節情報を辞典で調べるが，もし内蔵の辞典にその単語が見つからない場合はインタラクティブにユーザに尋ねる。辞典が大きいと単語探索に時間を費し，清書は遅くなるが，不完全な辞典はユーザを煩せることになる。そこで次のような方式を考えた。

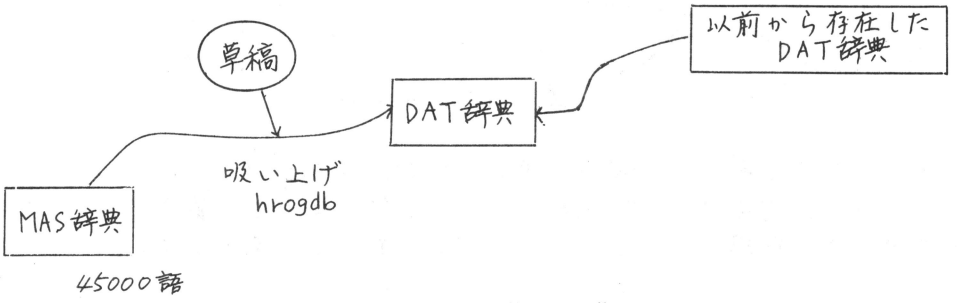

図４．最初の草稿の完成時のDAT辞典の生成

```
@hrogdb

563 WORDS IN DAT FILE
67 WORD IN ADD FILE
ENTER INPUT FILE: IFIP1

1674 WORDS IN NEW DAT FILE
@
```

{@はOS Tops-20のプロンプト信号，この吸い上げで，論文IFIP 1から1674-563 = 111語の単語がDATに登録された。下線の部分がユーザの入力である。}

hrogdbというプログラムは，ディスク・ファイルの草稿論文で使用されている単語を全部，ベースの辞典であるMASで調べ，その音節情報をDATに書き込む。１つの論文の総単語数を4000語として，そのうち相異なる単語の数は1500語程度である。（実際は長さ５以下の短かい単語はハイフニングの対象とはしないから，DATに登録される単語の数はずっと少ない。）すなわちDATは草稿に現われる単語をちょうど全部含むことになる。（同種類の論文を２つ以上書くとき，１度生成されたDATは２度使えるかも知れない。この場合，hrogdbはDATの拡張のために使われることになる。）

DATに加え清書時には（論文の修正のたびに清書は何度も繰り返される。）ADDというwork辞典も用いられる。論文はそれぞれ特殊な専門用語や造語を用いる。また特別の語形変化でMASからの吸い上げだけでは十分にカバーできない単語もありうる。（後述のように通常の語形変化に対しては十分な対応がとられる。）このような単語がたまたま行の末尾にきて音節情報が必要になると，HROはユーザに助けを求める。その入力結果はADD辞典へ入れる。（DATと同様，ADDもすでに存在するものを拡張していける。）AUX辞典は当面使いそうもない特殊語のゴミ溜めのようなところで，HROはADDにもDATにも見つからない単語を（ユーザが指示すると）ここで探すこともできる。

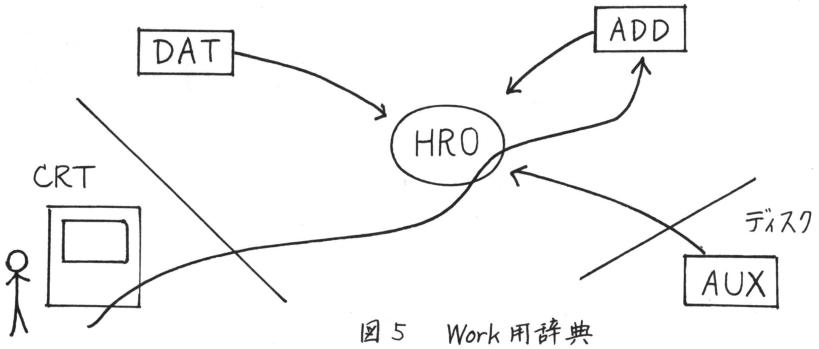

図5　Work用辞典

ＡＤＤは１論文で高々３０程度の単語ですむ。ＡＤＤ中のほとんどの単語は特殊用語でその音節情報は"全く分割なし"（単語全体が１音節）である。ADDへのユーザの入力が負担にならないように努力した。その例を見ておこう。

```
@hro

1632 WORDS IN DAT FILE
36 WORDS IN ADD FILE

HYPHEN RUNOFF SYSTEM
>pp

ENTER INPUT FILE:IFIP1

ENTER PARTITION FOR indicates :2 5 e
in dic ates
OK?:n

ENTER PARTITION FOR indicates :2 4 e
in di cates
OK?:y
REGISTER: indicates?:n
MODIFY indicates TO REGISTER? :y
ENTER A WORD: indicates
MODIFY (2 5) ?:n

ENTER PARTITION FOR enhance56 :s
ENTER A WORD: enhance
en hance56
OK? :y
REGISTER: enhance56 ?:y

24 PAGES
>end
@
```

> はHROのプロンプトである。

eは分割情報の終りを示す。

n = no

y = yes

indicatesの代りにindicateをADDに登録した。

enhance 56の音節情報はenhanceと同じであることをSで指示する。

図6　音節情報の入力

DATとADDは当該論文のテキスト・ファイルとともに保持され，最終的に論文が完成してそのファイルが消されるまで，論文とともに進化拡大する。（もし途中で論文が大巾に修正されることがあれば，hrogdbによるMASからDATへの吸い上げを新たにするとよい。）論文完成後，次の同種の論文清書のために保持するか，少くともADDはAUXにマージしておく。

DATとADDは必要最小限の単語を含むので処理時間とユーザの手間の縮少に成功した。各単語について音節情報は独立に存在するから，この方式が可能だが，

自然言語処理のように，１つの単語の意味を理解するのに（その原稿に出てこない）他の単語の意味が必要になる場合はこの限りでない。

辞典における単語の探索にはハシュ法とバイナリー・サーチを併用して効率化に努めた。またDATとADDを別々に探索すると時間が無駄なので，清書時には同じテーブルにおさめ，ダンプするときに別離することにした。

ハイフネーション

通常の辞引と同様にMASは副詞（−ly），名詞の複数形（−s），規則動詞の過去形（−ed），現在分詞（−ing）は含まない。ユーザがこれらをいちいち尋ねられたらたまらない。そこで例えば modifications の音節情報は modification のそれからというように単語の原形を類推することを自動的にしてほしい。この仕事を清書時に行うと処理を遅くするので，MASからDATに吸い上げるときに行うことにした。

hrogdδ は求める単語でMASに見あたらないものが，−s，−ed，−ing，−ness，−lyで終る場合は，その語尾変化前の原形を類推して，MAS で探す。もし見つかれば，その音節情報から変化形の音節を生成して DAT に登録する。この際 read ⟶ reading，　register ⟶ registeredのように規則的な場合は容易だが，university ⟶ unitersities，　refer ⟶ referredのような特殊な場合をすべてこなすのには苦労があった。

行の右端をそろえる

ハイフネーションが可能だからと毎行がハイフンで終っては見苦しい。そこである種の美的最適化が必要になる。まず任意の単語を任意の音節で区切るわけではない。例えば長さ５以上の単語のみを対象とし，u− とハイフンの前が１文字の場合は避ける。

ハイフネーションなしで右端そろえが可能になる確立を高くするために，制御用文字 % を用いる。%がソース・テキストに現れるとHRO はこれを空白にするかまたは無視するか，全く自由に選択できる。この自由度を利用して，右端そろえが可能な限り，ハイフネーションはなるべく回避される。%は文の終り（.の直後），，の直後，数式の前後など空白が１つでも２つでも見ばえに影響しない場所に入れておく。%%%と続けてもよい。この場合 0≤n≤3の空白に変わる。

ハイフネーションをしても，行の右端に余白が避けられない場合も起りうる。そこで単語と単語の間は最高２つの空白にまで広げてもよいことにする。１行の単語と単語の間が全部２つの空白になるわけではないから，行の左端から広げていくのと行の右端から広げるのを交互にすることにした。さもないと各行のだいたい同じ位置に空白が多く分布して，全体の紙面が割れたような印象を与える危険がある（図９参照）。

もし以上の方策をすべて用いても右端そろえが成功しない場合は，右端の空白が最も少ない状態でがまんする。このような場合は１行 50 語以上の場合，起る確率は非常に低く，大きな数式がある場合に限られる。

ＨＲＯの清書機能

ＨＲＯの清書の命令セットはRUNOFF によく似ている。（少くともサブセットである。）例で紹介しよう。

```
.TL REIJI NAKAJIMA et al
.TR   THE IOTA PROGRAMMING SYSTEM
.PS 35 .LM 10 .SP 1 .RM 60 .PN
```

図７．大域命令の例

これらの命令はソープテキストの初めに定義し，変更されないかぎりテキスト全体を制御する．（RUNOFF と違い，命令は行の先頭に来る必要はなく，テキストと混ってもよい。）　命令はすべて　.— の形である。

.TLは奇数ページのランニング・ヘッド，.TR は偶数ページのランニング・ヘッド，.PS は１ページの行数，.LM（レフト・マージン）.RM（ライト・マージン）.SP（スペーシング），　.RNはページを右肩に印刷する。

```
The relations between  the modules  both completed and unfinished
are highly complex and should  be managed by the   system.%
The Developer of the # system   keeps track of
any change and detects inconsistencies  between modules such as
circularity in module dependencies.%
.I 4
The  Data Base stores  the   changing  information on the  unfinished,%
as well as complete,% modules.%
  .FN 2
The user can be informed by  the Developer
of what remains to be done for each
 module.%  .EF
There are three states in which a module  can take.% In the first,%
 a module itself   is not yet finished.% In the second a module is
  completely  written,% processed  and verified,% but
its status is still conditional  on the state of the
lower  modules  on which  it depends in the object hierarchy.%
 Thus a module in this state will be  transferred to the third state when
all modules on  which   it depends   are transferred  into the third.%
Thus a  module is in the third state  if it is  complete  and  all  modules
 that it depends on   are also  in  the third.%  By a user decision,%
the system will provide that no further  change  is carelessly made to the
   module in the third state.%  Instead of modifying the module itself,%
 it is often more suitable to generate  a virtual module or a variant of
 the original module as we state in the following.%
.SK 1
Modules once completed  are  subject   to   change  at any
period  of the whole program development.%
In particular modules in the lower levels of the hierarchy
tend to be modified by the design conveniences of upper
level modules.%
@
```

図８．ＨＲＯへの入力テキストの例

.I４は改行して，次の行の先頭を４つへこませる（indentation），.SK は指定されただけ空白行を作る。＃は空白に置き換わる。.FN ２は次の文で .EFまでを２行のフット・ノートとする。

次に上のテキストに対する出力を見よう。

```
                should be managed by the system.  The Developer of
                the  system keeps track of any change and detects
                inconsistencies between modules such as circulari-
                ty in module dependencies.
.I 4    ——→      The  Data Base stores the changing information
                on the unfinished,  as well as complete,  modules.
                There are three states in which a module can take.
                In the first, a module itself is not yet finished.
                In the second a module is completely written, pro-
                cessed and verified,  but its status is still con-
                ditional on the state  of  the  lower  modules  on
                which  it depends in the object hierarchy.  Thus a
                module in this state will be  transferred  to  the
                third  state  when all modules on which it depends
                are transferred into the third.  Thus a module  is
                in  the third state if it is complete and all mod-

フット・ノート ——→The user can be informed by the Developer of  what
                remains to be done for each module.

                                                                9
ランニング・ヘッド ——→REIJI NAKAJIMA et al

                ules that it depends on are also in the third.  By
                a user decision,  the system will provide that  no
                further change is carelessly made to the module in
                the third state.  Instead of modifying the  module
                itself,  it  is  often more suitable to generate a
                virtual module or a variant of the original module
                as we state in the following.
.SK 1   ~→
空白を広げ
る方向   ——→Modules␣once␣completed␣are subject to change at
         ←—— any period of the whole␣program␣development.  In
         ——→particular␣modules in the lower levels of the hi-
         ←—— erarchy tend to be modified by the design␣conven-
                iences of upper level modules.
```

図9. HROの出力

HROにあって RUNOFF にないのはよく国際学会の予稿で必要になる1ページ2列（横2段組み）のフォーマットのための清書モード .DOUBLE である。

制御用文字として，$（次にくる単語と単語の間は（行末でないかぎり）丁度1つの空白とする。！（次の文字をそのまま印字する），井（空白とする），&（次の文字にアンダーラインを付ける），%（前述のとうり）がある。

あとがき

HROは DEC SYSTEM 20 の上に LISP {ユタ大のLISPに筆者の研究グループがLISPエディタ PLET[2]，デバック，構造化のための函数を追加したもの} で書いた。オ1 version は筆者1人で3週間（Proc. IFIPに出した筆者らによる論文はこれで清書した。このときはまだ右端そろえの機能はなかった。）で完成した。EMACSなど便利なサポートと使いやすい Os を用いるとプログラミングが効率化されることを再認識した。

その後 HRO の機能は拡張され今日に至っている。日本語の文章作りにも同じ

ような環境がほしいものである。

謝辞

HROのMAS辞典は三省堂のニューコンサイス辞典に基づく。これは編者の佐々木達氏，三省堂株式会社，京都大学工学部の長尾教授，辻井助教授のご好意により可能となった。また同辞典の作成は数理解析研究所技官の鈴木恵子嬢にお世話になった。

文献

[1] 杉本厚吉 "清書プログラム"（RUNOFF の解説）情報処理 Vol. 20, No. 11, '79
[2] 小島哲二 "エディタ論" 本予稿集 '81
[3] 中島玲二 " HRO - user's Manual '80