ＪＳＩ　ＡＩ　ワークステーション（４）

― 日本語ユーザモニター ―

1Q-4

穂積元一　諸橋正幸　吉永秀志

日本アイ・ビー・エム株式会社　サイエンス・インスティチュート

１．はじめに

ＪＳＩ　ＡＩワークステーション上にbilingual function と multi font service 機能を持ったユーザモニターを作成した。このモニターの中から『維摩』がサポートする各種のbilingual functionが使用可能となったので、このモニターの設計作成について述べる。

２．設計上の特徴

『維摩』システムそのものについては，文献[1]，[2]に解説されているので，重複を避けるために，ここでは述べない。

上記の日本語ユーザモニターの設計にあたっては次の２つの特徴を持っている。

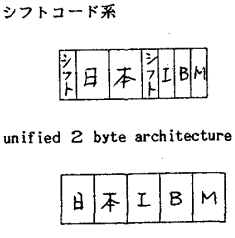
1）unified 2 byte architecture の採用。
2）カナ漢字変換機能の日本語トークナイザーとしてのサブルーチン化。

以下にこれらについて詳述する。

２．１　unified 2 byte architecture の採用

unified 2 byte architecture とは基本的には英数字及び日本語を含めてすべて２ Byte で表現するやりかたのことである。ただしシフトコードは存在しない。この方法が bilingual function を持つ日本語ユーザモニターにとっては本質的である。

例



以下に unified 2 byte architecture の利点をシフトコード付の１ Byte ２ Byte 混在系と比較して述べる。

ａ．文字列データの扱い易さ

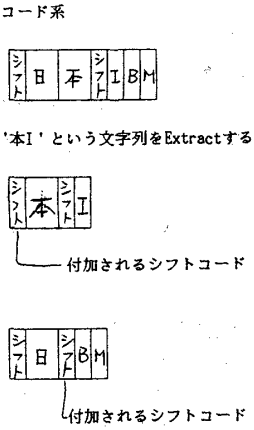
以下のｂ．～ｅ．に述べる事柄はすべてこの問題に帰着すると言っても過言ではない。基本的な文字列データの操作には次のものが存在する。

・Ｆｉｎｄ　（探索）
・Ｅｘｔｒａｃｔ　（引抜）
・Ｉｎｓｅｒｔ　（挿入）
・Ｃｏｎｃａｔｅｎａｔｅ（連結）

何れの場合もシフトコードが存在する方が明らかに複雑になる。

ここではＥｘｔｒａｃｔの場合についてのみ例を示す。

例



unified 2 byte architecture では、このような余分な処理が一切ない。

ｂ．コマンドパーザの作りやすさ

コマンドパーザの基本的な動作はＦｉｎｄとＥｘｔｒａｃｔの組合せであるから、上記ａ．の文字列操作の容易さに帰着できる。また、コマンド名およびパス名の扱いについても、unified 2 byte architecture のほうがはるかに簡単である。ＣＰＲでは、ロードモジュール名に２byteコードを許しているため、この点の問題はない。更に、パス名のスキャンに関しても、シフトコードの処理がないだけ楽である。

注）英数字も２byteとなっているために、漢字コードの２byte目が／コードかといった１byte ２byte 混在系の問題は存在しない。

ＪＳＩ　ＡＩ　Ｗｏｒｋｓｔａｔｉｏｎ　Japanese User Monitor

M.Hozumi M.Morohashi H.Yoshinaga
IBM Japan Science Institute

c．Ｉ／Ｏレコードの作りやすさ

シフトコード有りの世界では、１レコード中にシフトインコードとシフトアウトコードの対応がとれている必要がある場合が多い。つまり出力レコードにわざわざシフトコードを付加したりしなければならない。又、データベースレコードでは、シフトコードを除く場合が多い。これは、レコードの内部形式が予め決定されているためである。

（逆に入力レコードではシフトコードを付加する手間がいる。）

このような繁雑さはunified 2 byte architectureには一切存在しない。

d．Ｆｉｌｅ名の管理のし易さ

Ｆｉｌｅ名に２ｂｙｔｅコードを許した場合に、派生する問題には次のようなものがある。

・ディレクトリ管理の問題。
・コマンド管理の問題。
・その他のＦｉｌｅ名を持つテーブル管理の問題。

ディレクトリ自身をファイルと考えれば、ｃ．の問題になる。後はパス名管理のルーチンが２ｂｙｔｅコードを知っていればよい。

コマンド管理の問題については、ｂ．に述べた。

テーブル内にシフトコードを持ち込むと、ワイルドカードの処理等で困難が生じることが多い。

これに対して、リソース名管理を２ Byteで統一していれば、unified 2 byte architecture をテーブル内に持ち込むことができる。

e．Ｅｄｉｔｏｒの作りやすさ

Bilingual Editor の作りやすさとはａ．～ｄ．各項の容易さを総合したものといえる。ａ．項はEditorとして、当然文字列操作が要求される。更に、Editorのコマンド処理および中間ファイルのＩ／Ｏを考えれば、ｂ，ｃ項に関係してくる。そして、編集対象ファイル名に２Byteコードのものを許せば、ｄ．項となる。

２．２　カナ漢字変換機能の日本語トークナイザーとしてのサブルーチン化。

従来一般のアプリケーションからカナ漢字変換機能をサブルーチンとして使用する場合に、次のような問題点があった。

・変換の単位を誰が管理するのか
・ユーザインタラクションをどうするのか
・カナ漢字変換機能の初期化を誰がやるのか

これらの問題に加えて、今後のＡＩアプリケーションでは、自然言語処理との密接な関連から、文字列としての日本語より、トークナイズされた日本語を扱いたいという要求がある。これは自然言語の意味解釈をＡＩアプリケーションが必要とするためである。

ａ．変換単位の管理

従来アプリケーションからカナ漢字変換を使用する場合の変換単位は動的には変更できないものが多かった。つまり、ユーザインタラクションの許されない場合には、特別のコードまたは１６進表示によって指定するか、カナコードを使用しても次候補等の選択はできなかった。

これに対して、今回は次候補、全候補機能を『維摩』の方法でユーザインタラクションを含めてサブルーチン化できるので、アプリケーションの中でこれらを呼ぶことでユーザに変換単位の管理をさせることができる。又、ユーザインタラクションのできない場合には全候補全体をリスト形式で返還するサブルーチンを使用することで、アプリケーションの責任で後に最適なものを選択すればよい。

更に連文節変換機能を使用すれば、正解の範囲で一連のカナコード列が自動的に変換される。

ｂ．カナ漢字変換の初期化

今回の日本語ユーザモニターでは、モニター自身がカナ漢字変換機能の初期化を行うため、各アプリケーションがそれぞれ初期化を行う必要がなくなった。ただしユーザ辞書はユーザＩＤ対応に決定されてしまうので、アプリケーションが独自のユーザ辞書を指定することはできない。

ｃ．トークナイズ機能

日本語ユーザモニター上のＡＩアプリケーションが、自然言語として日本語を処理する場合、単に日本語文字列を扱うのではなく、意味解釈を伴うことが多いために、トークナイズされている方が有利である。

今回のカナ漢字変換機能は、文節変換を基本としているために、副産物としてカナ文字列のトークナイズを行うことができる。しかし、カナ漢字変換がサブルーチン化されることで、ＡＩアプリケーションから積極的にトークナイズ機能を使用することができるようになった。これによって、自然言語処理ＡＩアプリケーションの前処理としての、カナ漢字変換機能の位置付けが明確になっていくものと思われる。

３．おわりに

ＪＳＩ　ＡＩワークステーション上でＡＩアプリケーションがbilingualな自然言語処理を行う上で、よりよい環境を提供しようとするものが、日本語ユーザモニターであった。

コード系の処理のためには、unified 2 byte architectureを用い、日本語トークナイズのために、カナ漢字変換機能のサブルーチン化を行った。

今後のＡＩアプリケーションにとって自然言語処理が必須のものである、例えば音声認識されたカナ文字列をトークナイズした後、意味処理するといったとき、この日本語ユーザモニターの果たす役割は大きいといえる。

４．参考文献

[1] 諸構他「ワークステーション制御システム『維摩』とテキスト・エディタ」日本ソフトウェア科学会第１回大会3C-3,1984

[2] 諸構他「オフィスシステムのためのワークステーション制御システム『維摩』」情処全国大会59秋3E-4,1984