漢字情報学序説「入力、出力、そして検索」

安岡孝一

はじめに

文字に対する情報処理の基本は、入力・出力・検索である。漢字に対しても、この点は変わりない。しかし、たかだか 26 種類の文字か、あるいはその倍も処理すればよい欧米諸語とは異なり、漢字情報処理は 10000 種類を超える文字を相手にしなければならない。そこにはおのずから、漢字情報処理ならではの手法、工夫、面白味あるいは限界がある。本稿では、漢字情報学の入門編として、その手法や面白味を概説する。

1　漢字の入力

アメリカでコンピュータに文字を入力する際には、タイプライタ風のキーボードを用いれば、さほど不便は感じない。何せ、コンピュータで扱える文字が基本的には 95 種類しかない (図 1) のだから、50 キーかそこらのキーボードにシフト機構があれば (図 2)、そのまま入力できてしまうのである。

日本でも、アルファベットとカタカナくらい (図 3) なら、50 キーかそこらのキーボードに「英数/仮名」切り換えとシフト機構があれば (図 4)、まあ何とかそのまま入力できる。しかし、漢字となるとそうはいかない。3000 字程度なら、ダイレクトに入力するためのキーボードを作ることもできる (図 5) が、それ以上の字種となると、キーボードを作るのが物理的に難しくなってくる。やはり、キーボードで入力した文字列を、一度 (あるいは二度三度と) 変換することで、目的の漢字を入力するという手法を、とらざるを得なくなるのである。

1.1　コードで入力

コンピュータで扱う漢字全てに、数字かアルファベットによる一意な「コード」を割り振っておく方法である。「コード」はキーボードで入力し、それを変換することで直接、所定の漢字に行き着くことができるので、漢字の字種が増えても入力上の問題は起こらない。ただし、字種が増えると「コード表」が電話帳のような分厚さになってしまい、捜すのが大変になる、という問題はある。最近よく使われているのは、Unicode[11] という「コード表」のようである。

ROW	b_4	b_3	b_2	b_1	COLUMN → 0	1	2	3	4	5	6	7
					b_7 → 0	0	0	0	1	1	1	1
					b_6 → 0	0	1	1	0	0	1	1
					b_5 → 0	1	0	1	0	1	0	1
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
1	0	0	0	1	SOH	DC1	!	1	A	Q	a	q
2	0	0	1	0	STX	DC2	"	2	B	R	b	r
3	0	0	1	1	ETX	DC3	#	3	C	S	c	s
4	0	1	0	0	EOT	DC4	$	4	D	T	d	t
5	0	1	0	1	ENQ	NAK	%	5	E	U	e	u
6	0	1	1	0	ACK	SYN	&	6	F	V	f	v
7	0	1	1	1	BEL	ETB	'	7	G	W	g	w
8	1	0	0	0	BS	CAN	(	8	H	X	h	x
9	1	0	0	1	HT	EM	)	9	I	Y	i	y
10	1	0	1	0	LF	SUB	*	:	J	Z	j	z
11	1	0	1	1	VT	ESC	+	;	K	[	k	{
12	1	1	0	0	FF	FS	,	<	L	\	l	|
13	1	1	0	1	CR	GS	-	=	M	]	m	}
14	1	1	1	0	SO	RS	.	>	N	^	n	~
15	1	1	1	1	SI	US	/	?	O	_	o	DEL

図 1: ASCII [1]p.184

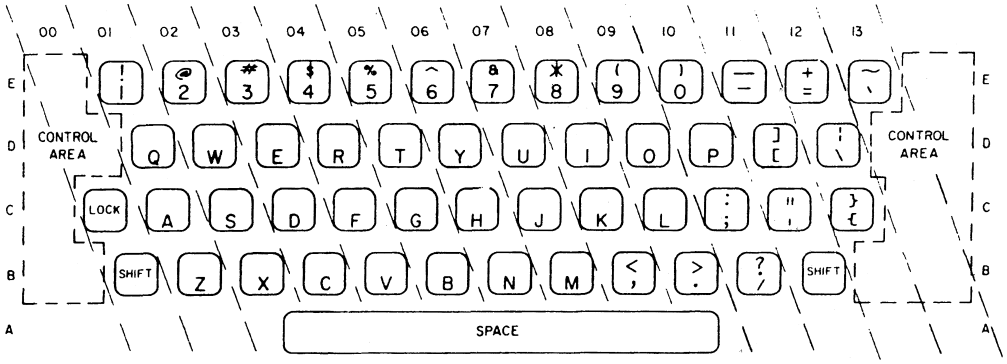
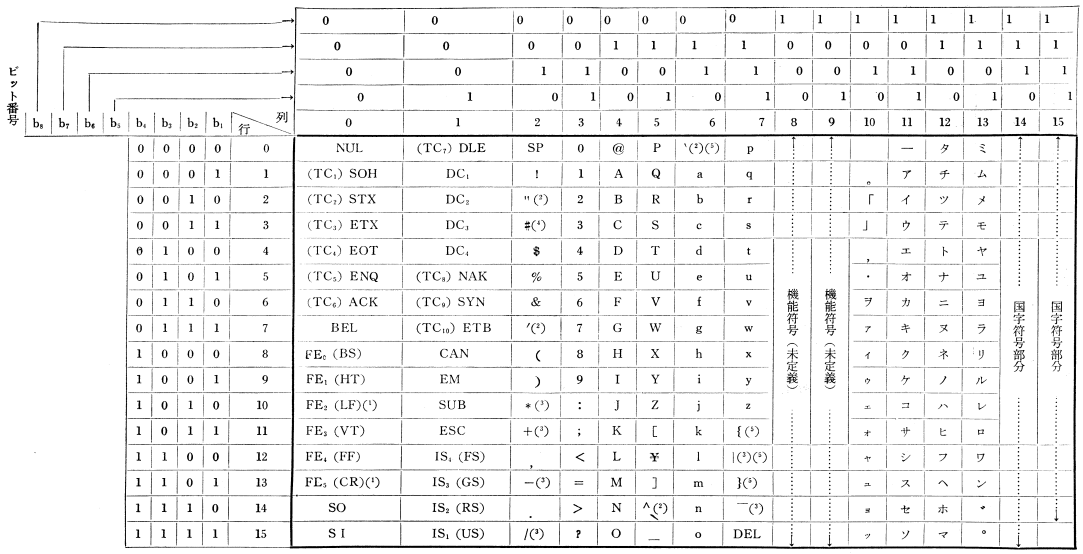


図 2: ANSI X4.14 Typewriter Pairing [4]p.9



ビット番号 b_8	b_7	b_6	b_5	行 / 列	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	NUL	(TC_7) DLE	SP	0	@	P	`(2)(5)	p				—	タ	ミ		
0	0	0	1	1	(TC_1) SOH	DC_1	!	1	A	Q	a	q			。	ア	チ	ム		
0	0	1	0	2	(TC_2) STX	DC_2	"(2)	2	B	R	b	r			「	イ	ツ	メ		
0	0	1	1	3	(TC_3) ETX	DC_3	#(4)	3	C	S	c	s			」	ウ	テ	モ		
0	1	0	0	4	(TC_4) EOT	DC_4	$	4	D	T	d	t			、	エ	ト	ヤ		
0	1	0	1	5	(TC_5) ENQ	(TC_8) NAK	%	5	E	U	e	u			・	オ	ナ	ユ		
0	1	1	0	6	(TC_6) ACK	(TC_9) SYN	&	6	F	V	f	v			ヲ	カ	ニ	ヨ		
0	1	1	1	7	BEL	(TC_{10}) ETB	'(2)	7	G	W	g	w			ァ	キ	ヌ	ラ		
1	0	0	0	8	FE_0 (BS)	CAN	(	8	H	X	h	x			ィ	ク	ネ	リ		
1	0	0	1	9	FE_1 (HT)	EM	)	9	I	Y	i	y			ゥ	ケ	ノ	ル		
1	0	1	0	10	FE_2 (LF)(1)	SUB	*(3)	:	J	Z	j	z			ェ	コ	ハ	レ		
1	0	1	1	11	FE_3 (VT)	ESC	+(3)	;	K	[	k	{(5)			ォ	サ	ヒ	ロ		
1	1	0	0	12	FE_4 (FF)	IS_4 (FS)	,	<	L	¥	l	|(2)(5)			ャ	シ	フ	ワ		
1	1	0	1	13	FE_5 (CR)(1)	IS_3 (GS)	−(2)	=	M	]	m	}(5)			ュ	ス	ヘ	ン		
1	1	1	0	14	SO	IS_2 (RS)	.	>	N	^(2)	n	¯(3)			ョ	セ	ホ	゛		
1	1	1	1	15	SI	IS_1 (US)	/(3)	?	O	_	o	DEL			ッ	ソ	マ	°		

機能符号（未定義）　機能符号（未定義）　国字符号部分　国字符号部分

図 3: JIS X 0201 [2]p.4

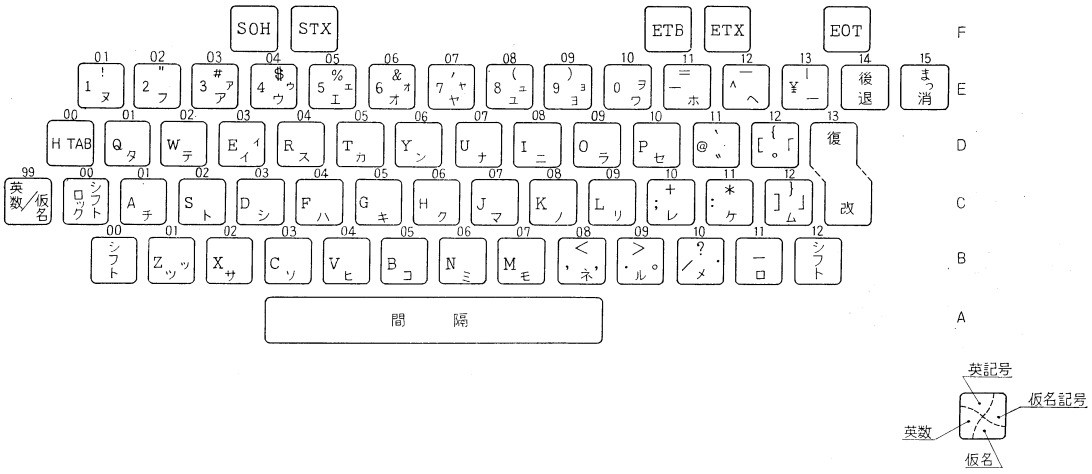


図 4: JIS X 6002 けん盤配列 2 [5]p.3

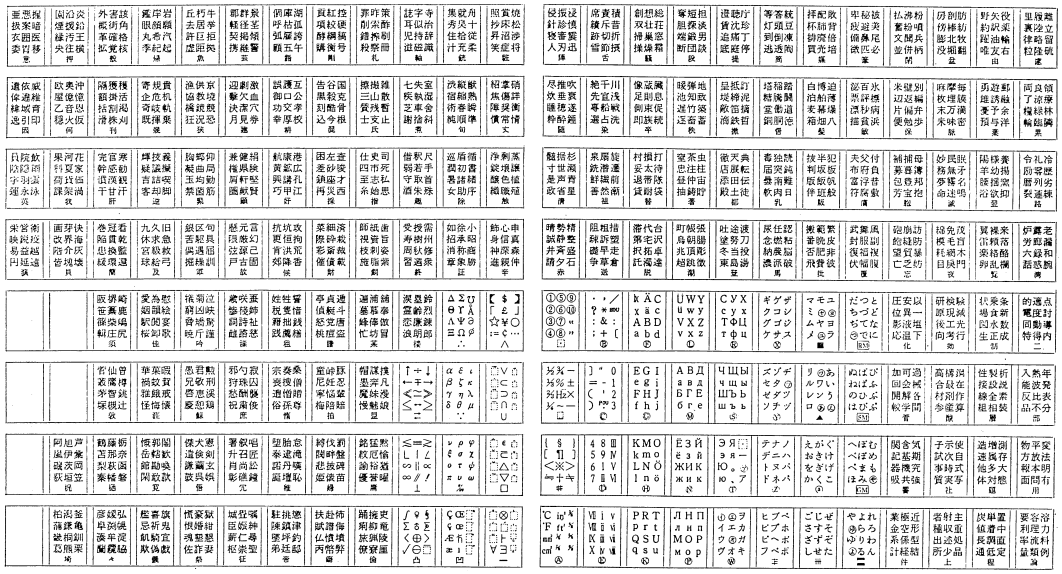


図 5: 日本科学技術情報センターの漢字けん盤配列 [3]p.312

1.2 読みで入力

コンピュータに漢字を入力する際に、漢字の「読み」を用いる方法は、日本では非常にポピュラーなものである。「ローマ字かな漢字変換」と呼ばれるこの方法は、キーボードでローマ字を入力し、それを一旦ひらがなに変換した上で、ひらがなをさらに漢字に変換するという、とてつもなくまどろっこしいものである。キーボード上にカナが書いてある (図 4) にもかかわらず、あえてそれを使わないのは、カナの配列を覚えるのが面倒くさいからしい。これに対し、中国では拼音を入力し、それを漢字に変換するので、まどろっこしさはない。一方、台湾では、注音字母の書かれたキーボード (図 6) で注音を入力し、それを漢字に変換するので、やはりまどろっこしさはない。

「読み」を用いて漢字を入力する方法の問題点は、人間が知っている「読み」と、コンピュータが知っている「読み」とが一致しないと、所定の漢字が入力できないところにある。つまり、ある漢字を入力したい場合に、その「読み」を人間とコンピュータの両方が知っていなければならない。しかしながら、人間が覚えられる「読み」の数には限界があるので、結局、使える漢字の字種を増やせないということになる。

1.3 形で入力

漢字の「形」を使って入力するという手法は、漢字の「読み」がわからなくてもよいという点から、漢字の有効な入力方法であり、非常に多くの研究がなされ

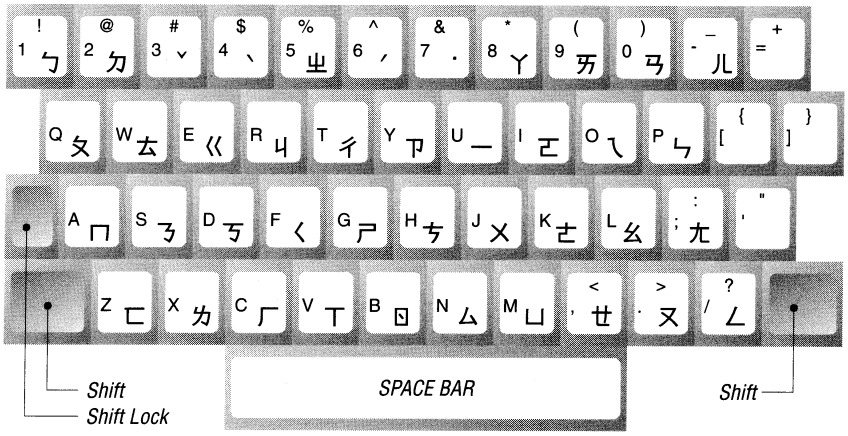


図 6: 注音字母けん盤配列 [10]p.246

てきた。しかし、漢字の「形」とキーボードとをどう対応づけるかという点において、それぞれの研究者がそれぞれの方法を取ったために、現在は百花繚乱の状態にある。

本稿では、これらの「形」による入力手法それぞれについては言及しない。というのも、筆者の考えでは、漢字の部品を直接キーボードと対応させるような現在の方法は、いずれ主流ではなくなっていくと思われるからである。すなわち、「読み」のわかっている漢字の「部品」を組み合わせて別の漢字を作る、というようなスタイルが、今後の入力方法のあるべき姿だと考えるからである。

2　漢字の出力

出力の仕掛けそれ自体に関しては、欧米諸語だろうと漢字だろうとあまり変わりはない。画面であれ紙であれ、文字を大きくしてみたり、字体を変えてみたり、読みやすく配置してみたり、ということができれば、出力としては上出来である。しかし、このようなことをする場合でも、欧米諸語に対して漢字は多少、不利な点がある。

2.1　文字の大きさを変える

文字を大きく表示/印刷することを考えると、文字の形はアウトライン情報 (図 7) として持つ方が良い。では、小さく表示/印刷する場合はどうだろう。たとえば 12 ポイントの文字を 600DPI のプリンタに印刷する場合は、1 文字の大きさは 100×100 ドットなのでアウトライン情報で OK だが、同じ 12 ポイントの文字を 72DPI のディスプレイに表示する場合は、1 文字の大きさは 12×12 ドットになっ

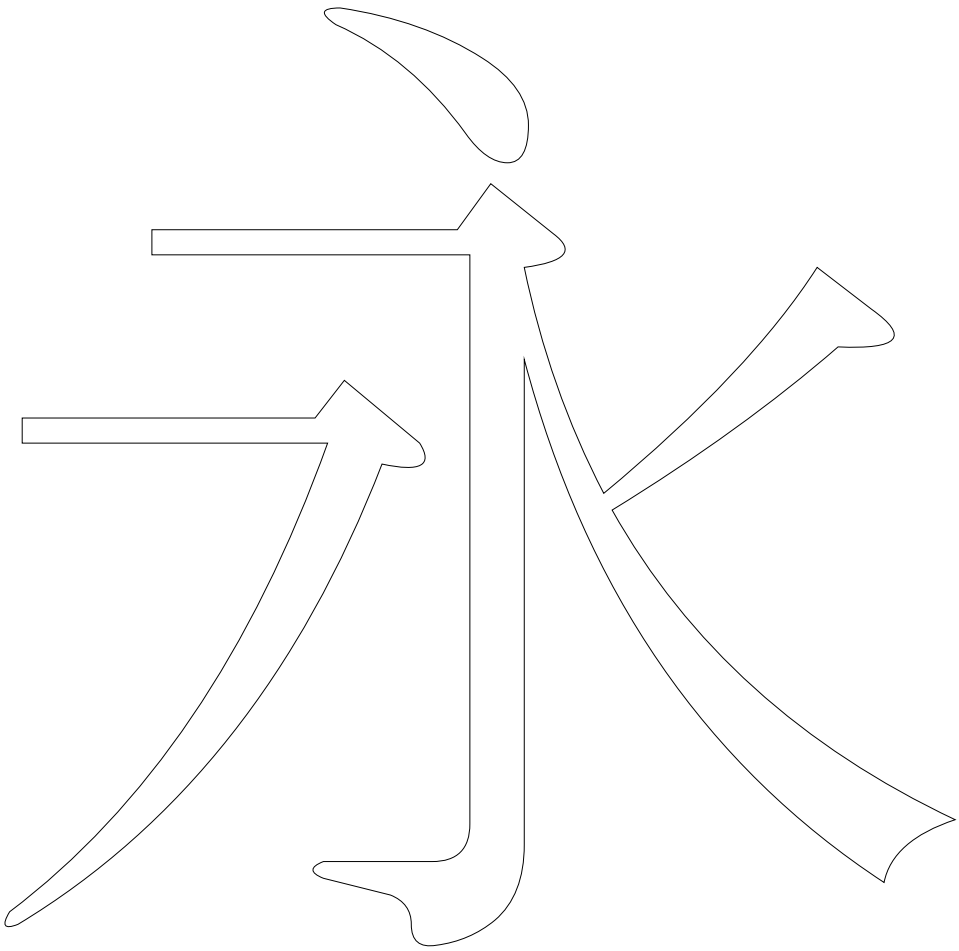


図 7: 「永」のアウトライン情報

てしまう。12×12 ドットとなると、アウトライン情報で実現するのは難しく、いきおい 12×12 のドットパターンを準備することになるが、漢字に対してはそれは非常な困難を伴う。

欧米諸語では、多くても 100 字かそこらを見分けることができればよいので、12×12 のドットパターンでも、そこそこ視認性の高いものを作ることができる。しかし、少なく見積もっても 2000 字以上の文字を弁別しなければいけない漢字では、12×12 のドットパターンはほぼ限界の小ささである。字種の多い漢字は、少ないドット数での視認性という点では明らかに不利であり、独特とも言えるドットパターン (図 8) を準備する必要があるのである。

図 8: 東雲 12 ドットフォント [http://openlab.jp/efont/shinonome/]

2.2　字体を変える

欧文諸語はフォントが豊富である。1フォント当たり100字程度デザインすれば十分なので、ひとまとまりのフォントを作りやすく、その結果、様々なフォントがデザインされてきたのである。これに対し、漢字はフォントが少ない。コンピュータでそれなりに使おうとすると7000字程度、学術目的にまで対応しようとすると30000字程度は準備しなきゃならないので、1フォント当たりのデザインに欧文諸語の70〜300倍の労力が必要となってしまうのである。つまり、漢字においては字体を変えるのが非常に難しい、というのが現状である。これを打破するためには、全ての文字デザインを手工業に頼っている現在のやり方をやめて、自動的に文字デザインをおこなうような仕掛けを開発する必要がある、と筆者は考えている。

2.3　文字を読みやすく配置する

文字組版という点において、漢字が欧米諸語と決定的に異なっているのは、縦書きが必要な点である。しかも、漢字を単に縦に並べればよい、というものではなく、縦書きと横書きが混在する場合(たとえば24や20,000,000のように)があり、やっかいである。

さらに日本には、ルビと呼ばれる独自の組版形式が存在する。ありていに言えば、読み仮名を振るだけのことだ。しかし、読みやすさを考えた場合、いくつかの組版上の制約(たとえば、ルビを振られた単語は、行末と次の行頭に泣き分かれになってはいけない)があり、これもやっかいである。

また、中国の古典を扱う場合には、割注はかかせない。割注は、文章の中に入った注(たとえばこんな形式)だが、割注が元の文章の複数行にまたがってしまう場合を考えると、非常に面倒な組版をおこなうことになる。

3　漢字の検索

テキストの中から、自分の知りたい事項が書かれている部分を探し出したい、という要求は、何もコンピュータに限ったことではない。書物に対しては、索引という形で実現されてきたことである。ただ、単語を単位とする欧米諸語と違い、漢字に対しては一字索引のような形で実現されてきたのもまた事実である。

コンピュータにおける検索には、大きく2つのやり方がある。1つは検索時に全データを順次チェックしていく方法、もう1つは事前にインデックスを作っておいて検索時にはインデックスを調べる方法である。データの更新が頻繁におこなわれる場合は前者の方がよいし、そうでない場合は後者の方が効率が上がる。ただ、前者の場合で、しかも検索に漢字がカラんでいる時には、以下に述べるような問題が表面化することになる。

3.1　文字コードの問題

コンピュータ上で文字を検索する場合、どうしても考えなければならないのが文字コードの問題である。コンピュータでの検索は、いわば 0/1 のパターンマッチングであり、何らかの方法で 0/1 に文字を変換してやらなければ、検索はおぼつかないからである。この「0/1 に文字を変換する方法」というのが、いわば文字コードである。

欧米諸語では、文字コードといえば、まずは ASCII(図 1) である。しかし、漢字を扱う場合には、シフト JIS[6, 7]、BIG5[8]、UTF-8[9, 11, 12] など、文字コードに関して様々な選択肢が存在し、下手をするとデータそのものも、これら様々な文字コードで書かれたものを、混在して扱わねばならない場合がある。

これら様々な文字コードを混在して扱わなければならない場合、現時点での有効な解は、検索時にデータを一旦 UTF-8 に変換することである。UTF-8 は、Unicode の変形の一種であり、シフト JIS や BIG5 の漢字をとりあえず全て収録していると同時に、高速な検索に適するように文字コードが設計されているからである。ただし、シフト JIS や BIG5 から UTF-8 への変換をおこなう場合には、いくつか注意を要する点がある。シフト JIS と BIG5 で良く似た形の漢字どうしが、同じ UTF-8 に変換されてしまう場合と、そうでない場合とがあるという点である。たとえば、シフト JIS の「憎」と BIG5 の「憎」は、UTF-8 に変換すると、どちらも E6868E となる。しかし、シフト JIS の「増」と BIG5 の「增」は、UTF-8 に変換すると、それぞれ E5A297 と E5A29E となってしまうのである。また、シフト JIS から UTF-8 への変換に際して、文字化けが起こることもある。最も有名な例は、シフト JIS の「¥」が、UTF-8 の「\」に化けてしまうものである。このような文字化けが起こっている場合は、逆変換に注意する必要があるだろう。

3.2　異体字の問題

欧米諸語に較べ、漢字の検索における異体字の問題は、より深刻である。たとえば、「図書」という文字列を検索したい場合、「図」に関しては「圖」「图」「圗」「𡈓」「𡆸」「圇」「𡈝」「𡈺」という漢字が、「書」に関しては「书」「𦘕」という漢字が、それぞれ異体字だと考えられるので、これらの組み合わせ、すなわち「圖書」「图书」を含むあらゆる組み合わせに関して、網羅的に検索したいという要求があるのである。

異体字の問題は、文字コードに収録されている漢字が増えれば増えるほど、大きくなる。たとえば、現時点での UTF-8 は漢字を 70195 字も収録しているが、その異体字関係がどのようになっているかは、ごく一部を除いてほとんど明らかになっていない。しかし、これを明らかにしなければ、漢字における網羅的な検索の精度を上げることはできないのである。頭の痛い問題だ。

おわりに

漢字の入力・出力・検索に関して、いくつかの手法や工夫あるいは限界を解説した。果たして面白味が伝わったかどうか、多少こころもとないのだが、これを機に漢字情報処理の道を邁進してほしい。

参考文献

[1] Fred W. Smith: "Revised U.S.A. Standard Code for Information Interchange", Western Union Technical Review, Vol.21, No.4 (November 1967), pp.184-190.

[2] JIS C 6220-1969 情報交換用符号, 日本規格協会, 東京 (1969 年 6 月 1 日).

[3] 高橋達郎, 森田朗, 広田広三郎: 漢字入出力機器について, 情報管理, Vol.12, No.6 (1969 年 9 月), pp.309-319.

[4] ANSI X4.14-1971 American National Standard Alphanumeric Keyboard Arrangements Accommodating the Character Sets of ASCII and ASCSOCR, American National Standards Institute, New York (March 31, 1971).

[5] JIS C 6233-1980 情報処理系けん盤配列, 日本規格協会, 東京 (1980 年 2 月 1 日).

[6] 長谷川均, 岡崎健: 漢字CP/M®のコード体系, 情報処理学会マイクロコンピュータ研究会資料, 26-2 (1983 年 3 月).

[7] 阿部雅人: CP/M 漢字標準化 漢字処理の現状, information, Vol.2, No.4 (1983 年 7 月), pp.81-87.

[8] 電脳用中文字型与字碼対照表 (技術通報 C-26), 資訊工業策進会, 台北 (1984 年 5 月).

[9] File System Safe UCS Transformation Format (FSS-UCS), X/Open, Reading (May 1993).

[10] Ken Lunde: CJKV Information Processing, O'Reilly & Associates, Sebastopol (January 1999).

[11] The Unicode Standard, Version 4.0, Addison-Wesley, Reading (August 2003).

[12] ISO/IEC 10646:2003 Information Technology — Universal Multiple-Octet Coded Character Set (UCS), ISO, Geneva (December 15, 2003).