4S-01 複数の文字コード体系を統一的に扱う固定長文字コードの一表現形式

橋本裕† 早川栄一‡ 高橋延匡‡

†東京農工大学 ‡拓殖大学

1. はじめに

電子計算機の能力の高まりと個人への普及により、その応用範囲が広がってきている。その結果、従来以上の文字種使用要求が高まり、複数の新しい文字コード体系([1][2][3])が作成された。また、インターネットの普及により、さまざまな言語の文字を同時に扱いたいという要求が生まれている。その問題の解決ために Unicode[4] の使用が期待されている。これに加え、例えば研究や遊びのため、あるいは新しい国ができたなどの理由から将来新たな文字コード体系が出現する可能性ある。このような事から、プログラム上からさまざまな文字コード体系を統一的に扱えるような符号化方式が必要となる。特に Unicode はさまざまな規格に採用されつつあるので、ある程度 Unicode との親和性がよい方式が望まれる。

2. 既存の符号化方式

既存の文字コードの符号化方式には、UTF8 (UCS Transformation Format) や EUC (Extended Unix Code) のような可変長符号化方式と、UTF32[5] や ASCII のような固定長符号化方式が存在する。可変長符号化方式では使用する文字コードの分布によっては無駄なメモリを使わずにすむことがある反面、文字コードの判別に状態を持ち込むことになるため、プログラムが複雑になるという問題がある。それに対して、固定長符号化方式ではそのような状態を含まないため、プログラムが単純になるという利点がある。しかし、文字コードに偏りがあったときも、使用される領域は常に一定なので可変長方式よりメモリを多く必要とすることがある。ただし現在では計算機で利用できる資源が拡大してきているので、これはあまり大きな問題にはならないと思われる。また固定長符号化方式では一つのコード体系だけを対象に考えていることが多く、複数のコード体系を扱う方法はあまり考えられていない。

3. 本符号化方式 Multicode

本稿では前述の問題に対応する一つの方式として Multicode という符号化方式を設計した。

3.1 設計目標

Multicode を設計するにあたっての目標は次の通りである。

- 複数のコード体系を統一的に扱える
- Unicode との親和性を考える
- 容易にプログラムから扱える

3.2 Multicode の設計

Multicode は 32bit の固定長符号化方式であり、各文字コードはそれぞれ 16bit の BankID とコードポイントから合成される。Unicode の UTF32 との親和性を持っている。

(1)32bit 固定長符号化方式

プログラムからは、可変長符号化方式より固定長符号化方式の方が容易に扱うことができる。そのようなことから固定長符号化方式を採用する。ここで固定長として割り当てる bit 数の問題が

A Fixed Width Character Encoding Scheme to Handle Various Coded Character Sets Uniformly

Yutaka Hashimoto†, Eiichi Hayakawa‡ and Nobumasa Takahashi‡

†Tokyo University of Agriculture and Technology

‡Takushoku University

ある。8bit では明らかに不足している。16bit では Unicode の例を見ればわかるとおり不足している。そのようなことから Multicode では 32bit の固定長符号化方式を用いる。

(2)各文字コード体系の統一的な扱いを実現するための 16bit BankID とコード範囲

32bit コード空間中で複数の文字コード体系の文字コードを統一的に扱うには、コード体系とその体系内での各コードのコードポイントを合成して得られる文字コードを使用する方法が考えられる。本方式ではこの方式を用いる。コード体系をあらわす ID を BankID、そのコード体系の最大文字数をコード範囲と定義する。コードポイントはこのコード範囲の中に含まれる値となる。

次に、BankID とコード範囲、それぞれに振り分ける bit 数の問題がある。例えば、BankID に 8bit、コード範囲に 24bit を無条件に割り当てる方式だと無駄が多い。Multicode ではそれぞれ 16bit を割り当てる。なぜならば、文字数が多いと言われている漢字でも、高々数万といわれているからであり、実際 JIS の第 1,2,3,4 水準で定義されている文字数をすべて加算しても 65536 文字には及ばない。BankID については計算機で処理しやすいという立場から 16bit とする。

(3)コード空間の無駄のない利用と Private Use Area を使った親和性の高い符号化方式

コード体系の中には、高々 8 bit のコード空間ですんでしまうものも少なくない。また Unicode の一表現形式である UTF32 との親和性を考慮すると、利用者が自由に使える Private Use Area を使用することが望ましい。これらを考慮した結果、図 1 に示す生成方式を用いて符号化することにした。

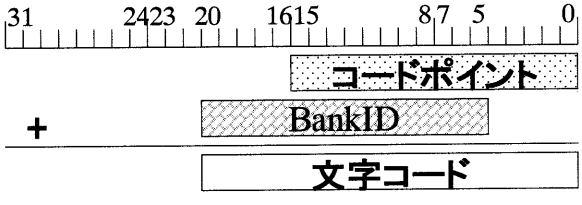

図 1　文字コード生成方式

つまり、BankID を 5bit 左にシフトさせ、その値にコードポイントを加算するのである。この結果、複数のコード体系を使用しても、体系毎に最大 31 文字が無駄になるだけで、効率的にコード空間を使用できる。同時に利用できる最大文字数は 200 万文字程度となる。

UTF32 からは、BankID 毎にきれいに領域が分断されるように見える(表 1)。Unicode と親和性を持ちつつ複数のコード体系を扱うために、他の文字コード体系には Private Use Are の範囲になる BankID を実行時に動的に割り当てる。この場合それらの文字コード体系で利用できる最大文字数は 100 万文字程度となる。BankID に 8800 以上の値を使うと、UTF32 で定められた範囲をはみ出し、UTF32 との互換性を失う。

表 1　Unicode との関係

Unicode(UTF32)	BankID	コード範囲	名前
0000- 007F	0000	0000-007F	Basic Latin
0080- 00FF	0004	0000-007F	Latin-1 Supplement
0100- 017F	0008	0000-007F	Latin Extended-A
0180- 024F	000C	0000-00CF	Latin Extended-B
		｜	
4E00- 9FFF	0270	0000-51FF	CJK Unified Ideographs
		｜	
E000- F8FF	0700	0000-18FF	Private Use Area
		｜	
F0000- FFFFF	7800	0000-FFFF	Private Surrogate Area
100000-10FFFF	8000	0000-FFFF	Private Surrogate Area
110000-11FFFF	8800	0000-FFFF	Extended Private Use Area
		｜	
1F0000-1FFFFF	F800	0000-FFFF	Extended Private Use Area

4. おわりに

複数の文字コード体系を統一的に扱う固定長符号化方式 Multicode について述べた。今後の課題としては、BankID の管理方法や、この符号化形式を使用するアプリケーションの設計、実現、及びその評価などが挙げられる。

参考文献

[1]JIS X0213:2000「7 ビット及び 8 ビットの 2 バイト情報交換用符号化拡張漢字集合」, 日本規格協会
[2]文字鏡研究会, http://www.mojikyo.gr.jp/
[3]ゆたかな文字文化を創りあげるために,
http://www.l.u-tokyo.ac.jp/KanjiWEB/00_cover.html
[4]Unicode, http://www.unicode.org/
[5]UTF32, http://www.unicode.org/unicode/reports/tr19/