基于 Unicode 编码的藏文转写拉丁文本的算法*

康才畯[1]，江 荻[1,2]

[1]上海师范大学 人文与传播学院，上海 200234

[2]中国社会科学院 民族学与人类学研究所，北京 100081

E-mail: encyc_k@tom.com

摘 要：本文以藏文音节字的结构关系和拼写顺序形成的转写规则为基础，结合 Unicode 编码位置特征，讨论并实现了基于 Unicode 标准的藏文转写拉丁文本的算法。文章提出了以 Unicode 编码区域位置为特点的识别思路，提出显示占位符宽度与 Unicode 编码长度以及基字丁组合层次高度关系的算法公式，并根据公式推导基字位置分解出横向（线性）和纵向（叠置）二维组合中的每个编码符号，予以拉丁转换。

关键词：Unicode；藏文国际标准；拉丁转写；藏文

The Algorithm of Transliteration of Tibetan into Latin Based-on Unicode Standard

Kang Caijun[1], Jiang Di[2]

[1]Humanities and Communications College, Shanghai Normal University, Shanghai 200234

[2]Institute of Anthropology and Ethnology, Chinese Academy of Social Sciences, Beijing 100081

E-mail: encyc_k@tom.com

Abstract: In this paper, we discuss and design an algorithm of Unicode encoded Tibetan's Latin transliteration. The algorithm is based on the structural relationship and the spelling order of Tibetan. From the perspective of Tibetan Unicode encoding features, we also put forward the formula on the width of Tibetan syllabic word, the length of its Unicode encoding length and the height of its precomposed character stack.

Keywords: Unicode; ISO/IEC standard for Tibetan; Latin transliteration; Tibetan characters

1 引言

本文讨论基于 Unicode（或国际标准藏文编码字符集）的藏文转写拉丁字母的算法。文献[1]和[2]曾详尽讨论了藏文转写的原理和规则，本文讨论这些原理和规则的计算机实现算法。转写讨论范围限于现代书面藏语，梵文词语转写规则详见文献[2]，其他数字、标点、变音符、篇章装饰符等文本符号的转写都直接利用对照表转换，无需赘言。从拉丁字母转写还原为藏文的过程除个别情况外，基本是藏文转写成拉丁字母的逆过程，本文也不讨论。

2 藏文字符 Unicode 编码的特点

目前最新的 Unicode 6.0 版本中，藏文编码从 0x0F00~0x0FFF，共 256 项编码位置，收入字符 211 项。其中，0x0F90~0x0FBC 属于组合用辅音字符，0x0F71~0x0F81 为元音字符，均不能单独作为字母使用，组合字符中 0x0FAD, 0x0FB1, 0x0FB2 是藏文的组合用变形符号[3]。需要说明的一点是，组合字符与其原型字符对应相同的拉丁字母转写，组合字符与其原型字符对照表见表 1[2]，该表是藏文转写拉丁字母的核心内容。

* 教育部－国家语委“民族语言文字规范标准建设及信息化科研资助项目（编号：MZ115-020）；中国社会科学院重大课题（YZDA2011-18）资助。

表 1　藏文组合字符与原型辅音字符及拉丁字母对照表

藏文	ཀ	ཁ	ག	གྷ	ང	ཅ	ཆ	ཇ	ཉ	ཊ	ཋ	ཌ	ཌྷ	ཎ
UCS	0F40	0F41	0F42	0F43	0F44	0F45	0F46	0F47	0F49	0F4A	0F4B	0F4C	0F4D	0F4E
组合	◌ྐ	◌ྑ	◌ྒ	◌ྒྷ	◌ྔ	◌ྕ	◌ྖ	◌ྗ	◌ྙ	◌ྚ	◌ྛ	◌ྜ	◌ྜྷ	◌ྞ
UCS	0F90	0F91	0F92	0F93	0F94	0F95	0F96	0F97	0F99	0F9A	0F9B	0F9C	0F9D	0F9E
转写	K	kh	g	gh	ng	c	ch	j	ny	tt	tth	dd	ddh	nn
藏文	ཏ	ཐ	ད	དྷ	ན	པ	ཕ	བ	བྷ	མ	ཙ	ཚ	ཛ	ཛྷ
UCS	0F4F	0F50	0F51	0F52	0F53	0F54	0F55	0F56	0F57	0F58	0F59	0F5A	0F5B	0F5C
组合	◌ྟ	◌ྠ	◌ྡ	◌ྡྷ	◌ྣ	◌ྤ	◌ྥ	◌ྦ	◌ྦྷ	◌ྨ	◌ྩ	◌ྪ	◌ྫ	◌ྫྷ
UCS	0F9F	0FA0	0FA1	0FA2	0FA3	0FA4	0FA5	0FA6	0FA7	0FA8	0FA9	0FAA	0FAB	0FAC
转写	t	th	d	dh	n	p	ph	b	bh	m	ts	tsh	dz	dzh
藏文	ཝ	ཞ	ཟ	འ	ཡ	ར	ལ	ཤ	ཥ	ས	ཧ	ཨ	ཀྵ	
UCS	0F5D	0F5E	0F5F	0F60	0F61	0F62	0F63	0F64	0F65	0F66	0F67	0F68	0F69	
组合	◌ྺ	◌ྮ	◌ྯ	◌ྰ	◌ྻ	◌ྼ	◌ླ	◌ྴ	◌ྵ	◌ྶ	◌ྷ	◌ྸ	◌ྐྵ	
UCS	0FBA	0FAE	0FAF	0FB0	0FBB	0FBC	0FB3	0FB4	0FB5	0FB6	0FB7	0FB8	0FB9	
变体	◌ྭ				◌ྱ	◌ྲ								[illegible]
UCS	0FAD				0FB1	0FB2								-
转写	w	zh	z	v	y	r	l	sh	ssh	s	h	a	kssh	hph
藏文	(ཨ)	◌ི	◌ུ	◌ེ	◌ོ	◌ཱ	◌ཱི	◌ཱུ	◌ྀ	◌ཱྀ	◌ཻ	◌ཽ		
UCS	(0F68)	0F72	0F74	0F7A	0F7C	0F71	0F73	0F75	0F80	0F81	0F7B	0F7D		
转写	(a)	i	u	e	o	aa	ii	uu	·i	·ii	ai	au		

3　藏文音节字的切分

基于 Unicode 的藏文转写拉丁文本的计算机算法分为两个步骤：（1）横向切分出藏文音节字并获得藏文音节字的编码长度；（2）根据音节字的编码长度及藏文 Unicode 编码的特点分解藏文音节字各个位置上的字符，并按照藏文转写原理置换为拉丁字母形式。

若令 F 表示藏文辅音字符 Unicode 编码集合，Y 表示藏文元音字符 Unicode 编码集合，C 表示藏文组合用字符（包括组合用变形符号）Unicode 编码集合，L 表示拉丁转写集合，T 表示转写规则函数，则藏文字符 Unicode 编码特征的数学描述如下：

1. $F=\{0x0F40,\ldots,0x0F6C\}$，$Y=\{0x0F71,\ldots,0x0F81\}$，$C=\{0x0F90,\ldots,0x0FBC\}$；

2. L={k, kh, g,…rr, aa…• ii}；

3. $T(f)=l$，且$\forall f_1 \nexists f_2 (f_1 \neq f_2 \cap T(f_1)=T(f_2))$，其中$f, f_1, f_2 \in F, l \in L$；

4. $T(c)=l$，其中$c \in C, l \in L$；

5. $\forall c \exists f (T(c)=T(f))$；

6. $T(y)=l$，且$\forall y_1 \nexists y_2 (y_1 \neq y_2 \cap T(y_1)=T(y_2))$，其中 $y, y_1, y_2 \in Y, l \in L$；

书面上，藏文音节字是一串不等长的字符集合，每个音节字都可以描述为：位于（篇章/段落）起首符与音节点之间，或者位于音节点与音节点之间，或者位于音节点与单（/双）垂符之间的一串字符[1, 2]。切分音节字主要以起首符、音节点与单（/双）垂符等分隔符号为依据，逐个提取藏文文本中的 Unicode 编码，将起首符、音节点及单（/双）垂符都标记出来，则任意两个分隔符号

间的音节串即为一个独立的音节字，该音节串的 Unicode 编码长度即为音节字的长度 l。

4 藏文音节字的识别与转写

按照传统文法描述，藏文音节字的结构由前加字、上加字、基字、下加字、元音、后加字、重后加字构成，除基字外，其他位置的字符可能缺省（元音缺省即包含无字形的元音 a），即最长 7 个字符，最短 1 个字符。例如，ཁ（kha）“嘴”仅 1 个字符，བསྒྲིམས（bsgrims）“捻（线）过去时”是 7 个字符。基字位置上的其他字符都叠置于基字上方或下方，构成所谓基字丁。就计算机显示处理而言，藏文音节字在垂直方向上重叠的多个字符在水平方向上仅占一个占位符，因此，藏文音节字显示的宽度 w 与音节字的 Unicode 编码长度 l 并不一定相等，上例中 བསྒྲིམས（bsgrims）即是由前加字 བ（b）、基字丁 ཀྲི（sgri）、后加字མ（m）、重后加字ས（s）7 个字符构成的宽为 4 位的音节字。

要实现藏文音节字的转写，必须将音节字中各个字符根据书写顺序分析出来。藏文音节字在计算机中是以 Unicode 编码字符串的一维形式存储的，无法直观的表现藏文音节字在书写中的二维信息。因此，要将音节字中以基字为中心的各个字符分析出来，关键的一点是确定基字丁所在的位置。要确定基字的位置，必须要确定音节字显示的宽度 w。

从上文的分析可知，当藏文音节字中的基字没有叠加其他字符时，Unicode 编码的长度 l 与音节字的宽度 w 是相等的；当音节字中的基字上或下方出现叠加情况时，Unicode 编码的长度 l 则与字宽 w 不相等，但 l 与 w 以及字丁高度 h（叠加字符的数量）始终存在以下关系式：

$$w = l - h + 1 \quad \text{其中} \quad l \geqslant h, h \geqslant 1$$

由此可知，在已知藏文音节字长度 l 的前提下，要得到藏文音节字的宽度 w，需要获得基字丁的高度 h。而藏文字符 Unicode 编码标准的特点，使得通过算法得到 h 的值成为了可能：当音节字的 Unicode 编码中的第 n 位字符的编码值位于集合 C 或者集合 Y 内时，说明该音节字中出现了字符组合叠加的情况，若第 n-1 位字符的编码值仍位于集合 C 内，说明该组合字符上还叠加有其他字符，直到出现第 n-m 位字符的编码值位于集合 F 当中，说明该字符为叠加结构的顶端。由此规则即可通过递推方法获得该音节字中基字丁的高度 h=m+1。

通过 l 及 h 计算获得 w 的数值后，可根据以下几种情况分析基字的位置并进行转写：

1. 当 w = 1 时，为单字符宽度音节字。凡是单字符宽度音节字，不管是否有叠加情况存在，均可根据转写规则函数 T 将其辅音与元音字符的 Unicode 编码依序转写为拉丁字母。如果音节字中不存在元音的 Unicode 编码，则在转写字符串的末端补上默认的元音字母 a；

2. 当 w = 2 时，即 2 字符宽度音节字。当音节字的 Unicode 编码中含有集合 C 或 Y 内的编码时，该编码之前位于集合 F 内的编码所代表的辅音即为基字，基字前面若有字符则为前加字，基字后若有字符则为后加字。若音节字的 Unicode 编码中不含集合 C 或 Y 内的编码值时，则第一个辅音字符为基字，后一个为后加字。转写时，依据：前加字→基字丁→后加字的书写顺序依次调用转写规则函数 T 将藏文字符转写为拉丁字母，其中基字丁的转写参照单字符音节字的转写方法进行；

3. 当 w = 3 时，即 3 字符宽度音节字。与 2 字符宽度音节字类似，当音节字中带有叠加结构或元音符号的 Unicode 编码时，叠加结构或元音符号所依附的辅音即为基字，基字前若有字符则前后字符分别为前加字和后加字，基字后若有字符则为后加字及重后加字。若音节字中不含叠加结构或元音符号，且第 2 个占位符位置上的字符为 ག（g）、ང（ng）、བ（b）、མ（m），第 3 个占位符位置上的字符是 ས（s），或者第 2 个占位符位置上的字符是ན（n）、ར（r）、ལ（l），第 3 个占位符位置上的字符是 ད（d），则第一个占位符位置上的字符为基字，否则第二个占位符位置上的字符为基字。3 字符宽度音节字的转写方法与 2 字符宽度音节字转写方法类似，以书写顺序依次调用转写规则函数 T 将藏文字符转写为拉丁字母，其中基字丁的转写参照单字符音节字的转写方法进行；

4. 当 $w=4$ 时，即 4 字符宽度音节字。每个占位符位置上的字符从左至右分别是前加字、基字、后加字、重后加字。其转写方法同样为以书写顺序依次调用转写规则函数 T 将藏文字符转写为拉丁字母。

转写算法的完整流程图如下所示：

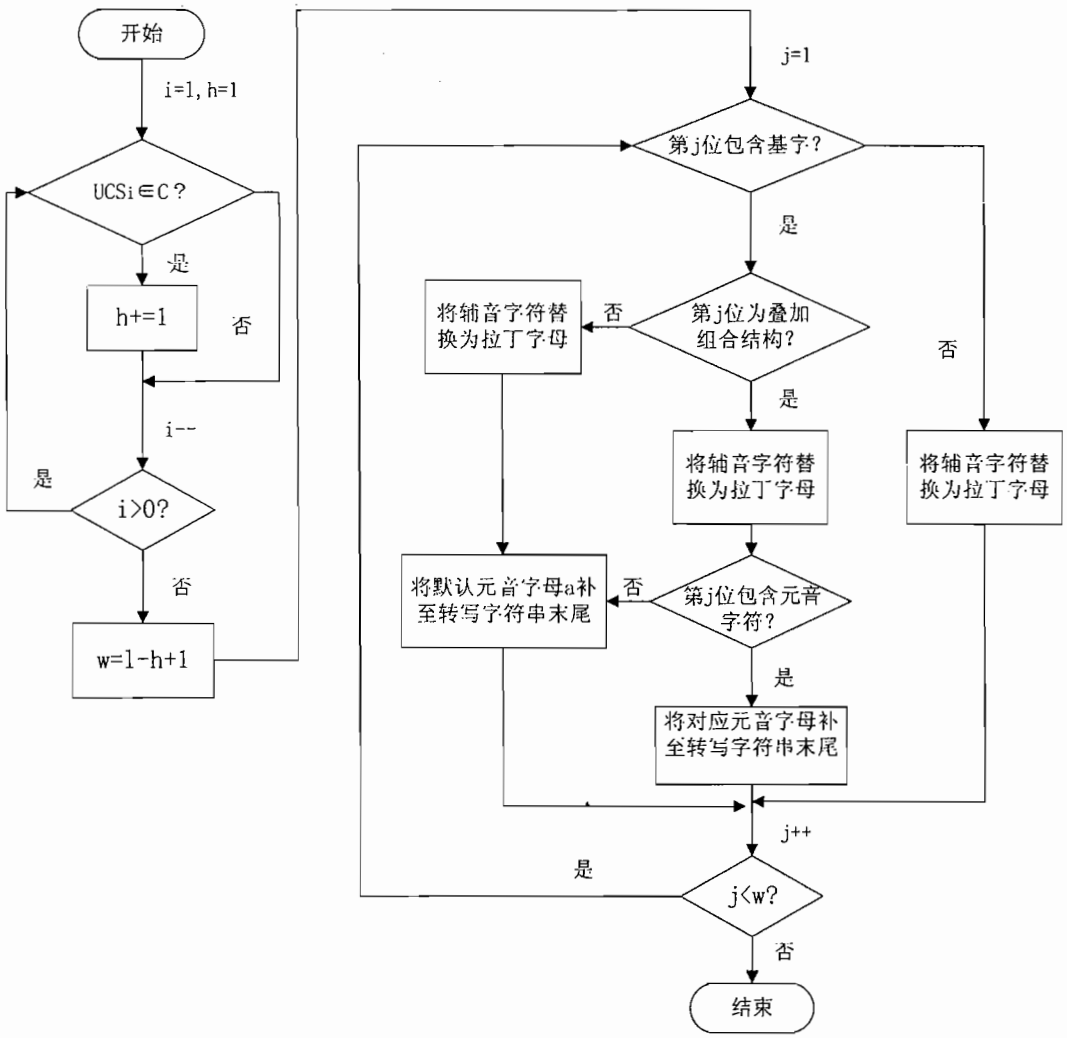

图 1　藏文音节字拉丁转写算法流程图

5　特定结构的转写规则

部分藏文音节字的结构对叠置字符的生成和识别产生影响，需要设置特定的规则[2]。

（1）Unicode 编码中将叠置型梵源藏文字符独立编码[3]，即 ག (gh，0F43)，ད (ddh，0F4D)，ཏ (dh，0F52)，བ (bh，0F57)，ཛ (dzh，0F5C)，对应的组合型字符分别 ◌ྒ (0F93)，◌ྡ (0F9D)，◌ྤ (0FA2)，◌ྦ (0FA7)，◌ྫ (0FAC)。由于叠置引擎技术可以实现单字符的组合，因此ག (gha，0F43)可以用ཧ (0F42+0FB7)表示，ཀྵ (sgha，0F66+0F93)可以用 ཀ (0F66+0F92+0FB7)表示，为此，在纵向高度识别算法上要建立梵源组合藏文字符的 Unicode 编码对应表来实现转写目标。

（2）叠置在基字丁上方或下方的长元音既可以通过长元音字符独立实现，例如◌ི (ii, 0F73)，也

可以通过元音字符结合长音标记 ཱ (0F71) 来实现，转写时元音倍写一次，长元音标记自身不转写，例如，གཱ (gaa)，ཱི (cii)。这两种方法显示出来的音节字完全相同，但后者增长了 Unicode 编码，因此在转写中很容易造成混淆。本文通过判断是否有独立的长音标记和元音符号来避免遗漏长元音的转写。

(3) 基字丁ཡ (ya) 带前加字ག (g-) 转写时在基字 y 之前添加别义符 "-",[5]可以通过 Unicode 编码 0F42+0F61 判定该结构，例如 གྱ (g-yu) 编码串是 0F42+0F61+0F74。

(4) 藏文连词འང (vang)、འམ (vam) 总是与前一无韵尾音节连写构成合体字，破坏了音节规则。文献[2]规定凡这类现象转写应将连词从被附着音节剥离，转写为 vvang 和 vvam，例如 མཐོང་转写成 mtho vvang，མགྲམ 转写成 mgu vvam。算法上，凡出现 0x0F60+0x0F44(vang)和 0x0F60+0x0F58 (vam) 编码串，则独立转写，并重复其中的第一个辅音字母。

(5) 不带叠加字符或元音的 3 字符宽音节字中有极少数可能会出现二义性现象，例如，བགས 可分别转写为 (bags) "逐渐，adv"，(bgas) "砍，刻，v"，这类情况只能通过文本上下文或词频统计来决断。

6 结语

文献[4]主要讨论了藏文预组合字符的拉丁转写算法，不过，随着基于 Unicode 编码的广泛应用，计算机藏文处理的重心逐渐转移到基本编码集的应用上来，因此藏文的拉丁转写也应以此为基础。本项研究开发的藏文转写拉丁字母程序完成了相当数量的文本转写实验，准确率达到 99% 以上，效果令人满意。进一步的工作是完善对藏文译写的梵文词语的转写算法，最终实现一个完全的藏文转写拉丁字母文本系统。

参 考 文 献

[1] 江荻. 2006. 藏文的拉丁字母转写方法——兼论藏文语料的计算机转写处理。《民族语文》第 1 期 45-53 页.

[2] 江荻，龙从军. 2010.《藏文字符研究：字母、读音、编码、字频、排序、图形、拉丁字母转写规则研究》，社会科学文献出版社.

[3] The Unicode Consortium. 2011. The Unicode Standard. Version 6.0, Mountain View, CA.

[4] 陈丽娜，祁坤钰，贾彦民，吴健，康丽. 2006. 藏文拉丁转写的研究与实现.《计算机工程与设计》，27 卷第 1 期.

[5] Wylie, Turrell. 1959. A Standard System of Tibetan Transcription, Harvard Journal of Asiatic Studies, 22 (1959), Pp. 261-267.