A Gurmukhi Collation Algorithm

G S Lehal
Department of Computer Science & Engineering,
Punjabi University, Patiala.
gslehal@yahoo.com

Abstract. Sorting and Indexing is one of the basic necessities of the database management system. But unfortunately there does not exist any software for automatic sorting of Punjabi words. The collating sequence provided by ISCII or UNICODE is not adequate. Separate rules have to be written for sorting. In this paper we discuss the various issues involved in sorting of Gurmukhi text and present a Gurmukhi collation algorithm. Gurmukhi, like other Indian languages has a unique sorting mechanism. Unlike English, consonants and vowels have different priorities in sorting. Words are sorted by taking the consonant's order as the first consideration and then the associated vowel's order as the second consideration. In addition there is another complication. The properly sorting "characters" in Gurmukhi often requires treating multiple (two or three) code points as a single sorting element. Thus we cannot depend on character encoding order to get correct sorting instead we have to develop using sorting rules of Gurmukhi linguistic collation function which convert the word into some intermediate form for sorting.

Keywords : Collation, multi-level sorting, Indexing, Gurmukhi

1. INTRODUCTION

Sorting and Indexing is one of the basic necessities of the database management system. But unfortunately there does not exist any software for automatic sorting of Gurmukhi words. The collating sequence provided by UNICODE or ISCII is not adequate, as it is not compatible with the traditional sorting for Gurmukhi words. In fact, the first character itself in ISCII for Gurmukhi is ਅ while traditionally ੳ is the first character in the alphabet set of Gurmukhi.

Sort orders vary from culture to culture, and many specific applications require variations. Languages vary not only regarding which types of sorts to use, but also in what constitutes a fundamental element for sorting. For example, Swedish treats Latin capital A with Diaeresis as an individual letter, sorting it after z in the alphabet; Germen, however sorts it either like *ae* or like other accented forms of a.. following a.Combinations of letters can be treated as if they were one letter. For example, in traditional Spanish "ch" is treated as a single letter, and sorted between "c" and "d". As a result, it is neither possible to arrange characters in an encoding in an order so that simple binary string comparison produces the desired collation order, nor is it possible to provide single-level-sort-weight tables. To address the complexities of culturally expectant sorting, a multilevel comparison algorithm is typically employed. Each character in string is given several categories of *sort weights*. Categories can include alphabetic, case and diacritic weights, among others. In a first pass, these weights are accumulated into a *sort key* for the string. At the end of the first pass, the sort key contains a string of alphabetic weights, followed by a string of case weights, and so on.

Gurmukhi, like other Indian languages has a unique sorting mechanism. Unlike English, consonants and vowels have different priorities in sorting. Words are sorted by taking the consonant's order as the first consideration and then the associated vowel's order as the second consideration. In addition there is another complication. The properly sorting "characters" in Gurmukhi often requires treating multiple (two or three) code points as a single sorting element. Thus we cannot depend on character encoding order to get correct sorting instead we have to develop using sorting rules of Gurmukhi linguistic collation function which convert the word into some intermediate form for sorting.

In this paper we discuss the various issues involved in sorting of Gurmukhi text and present an algorithm for Gurmukhi collation.

2. GURMUKHI SCRIPT

Punjabi is the world's twelfth most widely spoken language. The populace speaking Punjabi is not only confined to North Indian states such as Punjab, Haryana and Delhi but is spread over all parts of the world. In fact there are more people using Punjabi on computers abroad than in India. Punjabi is written in Gurmukhi script. It had been introduced by Guru Angad Dev of Sikhs in sixteenth century. Some of the major properties of the Gurmukhi alphabet are:

- Gurmukhi script alphabet consists of three (ੳ ਅ ੲ) vowel-carrier letters and nine vowel signs. By using the vowel signs with the three vowel-carrier letters ten vowels are obtained. The vowel-carriers ੳ and ੲ are never used without a vowel sign.
- There are 38 consonants and all the ten vowel-signs are used with all the other consonants (Fig. 1).
- In addition there are two nasal signs ◌ਂ (*bindi*) and ◌ੰ (*tippi*) used for sounds produced through nasal cavity. The symbol *adhak* ◌ੱ , is used to produce the sound of a double consonant.
- There are three half characters in Gurmukhi alphabet. The complete character set of Gurmukhi is depicted in Fig. 1.

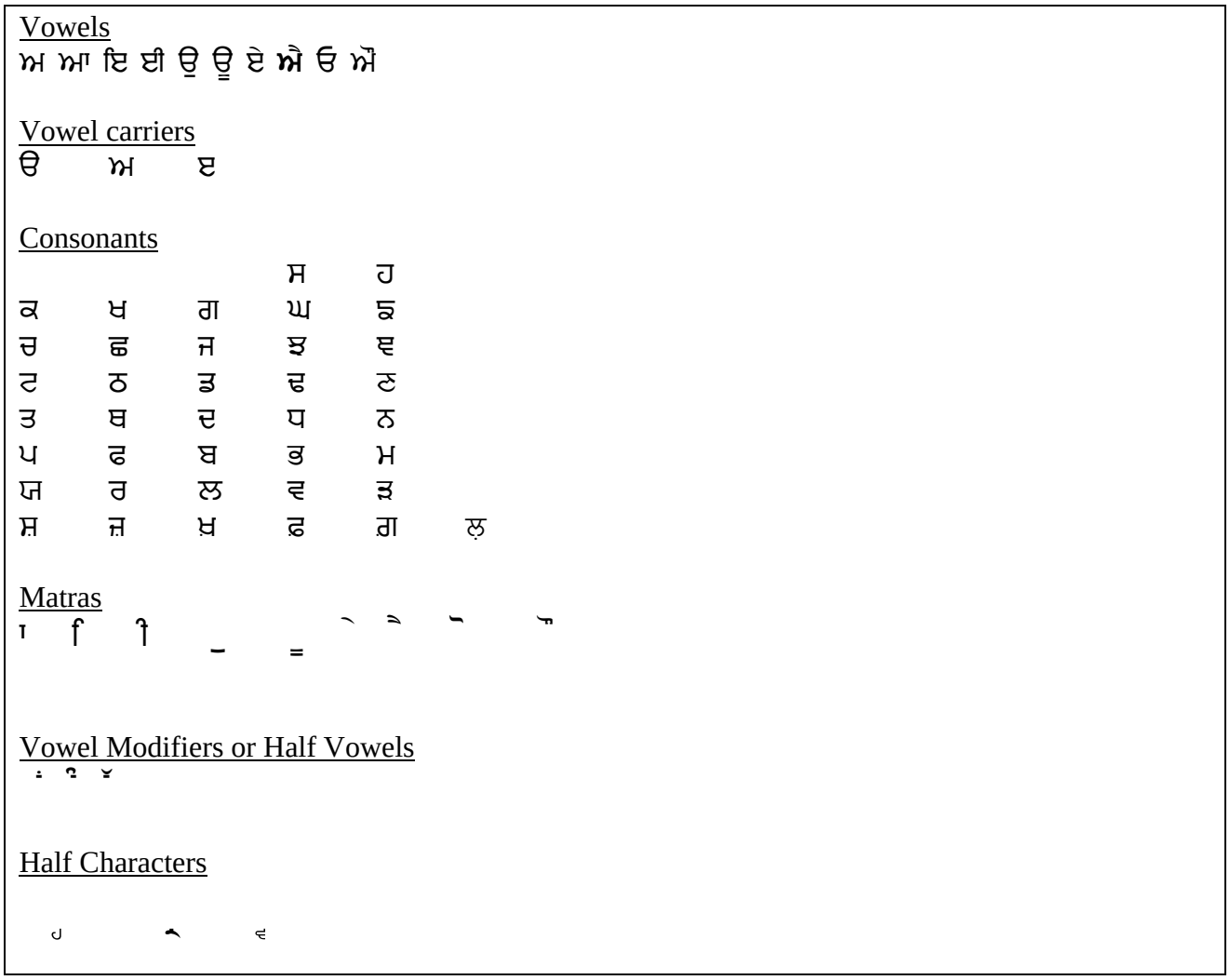

Fig 1 : Gurmukhi character set

3. ISSUES IN GURMUKHI SORTING

Gurmukhi text sorting two main factors have to be taken into consideration:

1. Deciding the collating sequence.
2. Taking care of primary, secondary and tertiary weight levels to be assigned to some characters.

After discussions with linguists and studying in detail the alphabetic order in Punjabi dictionaries[Singh 1991], the collating sequence and the collating code to be assigned to various Gurmukhi characters as displayed in Table 1 is developed. For convenience of non-Punjabi readers, the equivalent Roman transliteration is also provided in Table 1. For some Punjabi characters such as ੳ, ੲ and ੵ there are no equivalent Roman symbols.

Table 1 : Gurmukhi collating sequence and collating code

Code	Punjabi Symbol	Equivalent Roman
128	ਃ	
129	ੵ	
130	ਁ	
131	ੳ	
132	ੁ	u
133	ੂ	oo
134	ੋ	o
135	ਅ	a
136	ਆ	aa
137	ਐ	ay
138	ਔ	au
139	ੲ	
140	ਇ	i
141	ਈ	ee
142	ਏ	e
143	ਸ	s
144	ਸ਼	sh
145	ਹ	h
146	ਕ	k
147	ਖ	kh
148	ਖ਼	Kh
149	ਗ	g
150	ਗ਼	G
151	ਘ	gh
152	ਙ	Mg
153	ਚ	ch
154	ਛ	chh

155	ਜ	j
156	ਜ਼	Z
157	ਝ	jh
158	ਞ	Mj
159	ਟ	T
160	ਠ	Th
161	ਡ	D
162	ਢ	Dh
163	ਣ	n
164	ਤ	t
165	ਥ	th
166	ਦ	d
167	ਧ	dh
168	ਨ	n
169	ਪ	p
170	ਫ	ph
171	ਫ਼	f
172	ਬ	b
173	ਭ	bh
174	ਮ	m
175	ਯ	y
176	ਰ	r
177	ਲ	l
178	ਲ਼	L
179	ਵ	w
180	ੜ	rh
181	ੰ	
182	ੱ	
183	ੴ	
184	ਾ	aa
185	ਿ	i
186	ੀ	ee
187	ੁ	u

188	ੂ	oo
189	ੇ	e
190	ੈ	ay
191	ੋ	o
192	ੌ	au

The above collating sequence is different than the one provided in ISCII or Unicode.

Based on the collating sequence provided in Table 1, ਭਜਨ < ਭਜਾਉਣਾ < ਭਜਾਏ < ਭਾ

Since ਨ < ਾ so ਭਜਨ < ਭਜਾਉਣਾ
as ਉ < ਏ thus ਭਜਾਉਣਾ < ਭਜਾਏ
and as ਜ < ਾ thus ਭਜਾਏ < ਭਾ

3.1 Semi-Vowels or Vowel Modifiers

The semi-vowels do not participate in primary sorting but they play their role in secondary sorting. The words are first sorted in lexicographic order based on Table 1 after ignoring the semi-vowels and next, equivalent strings, i.e. words differing by their semi-vowels only, are sorted among each other using the collating order of the semi-vowels and their position in the word.
Among the semi vowels the collating sequence is ੰ < ੱ < ੍
Thus if have to sort the words ਜਤ ਜੱਤ ਜੰਤ ਜਤਨ ਜੱਗ ਜੰਗ
Then in the first pass the sorted order will be
(ਜੱਗ = ਜੰਗ) < (ਜਤ = ਜੱਤ = ਜੰਤ) < ਜਤਨ . To resolve the ties, we look at the collating sequences of the semi-vowels and thus get the following order:
ਜੱਗ < ਜੰਗ < ਜਤ < ਜੱਤ < ਜੰਤ < ਜਤਨ

3.2 Ignorable characters

Another class of cases involves ignorable characters such as spaces, hyphens, and some other punctuation marks. In this case, the character itself is ignored unless there is no other difference between the strings. Thus for example blackbird < black-bird < blackbirds. In Gurmukhi, the hyphen has a very high frequency of occurrence. Hyphen is part of many Punjabi words such as ਭੂਤ-ਪ੍ਰੇਤ ਟ੍ਰੇਡ-ਮਾਰਕ ਦੂਰ-ਦ੍ਰਿਸ਼ਟੀ ਕੜਾਹ-ਪ੍ਰਸ਼ਾਦ ਕਿਰਪਾ-ਦ੍ਰਿਸ਼ਟੀ ਵਿੱਦਿਆ-ਪ੍ਰਨਾਲੀ etc. In fact from a statistical analysis of all the Punjabi words and their inflections it was found that the hyphen has 0.75% of occurrence[Singh 1991], which is higher than many other Gurmukhi characters such as ਙ, ਞ, ਧ etc. Usually both the spellings, with and without hyphen are commonly used for many Gurmukhi words.
At the third level, we have taken care of hyphens present in Gurmukhi words. The words are sorted after ignoring the hyphens present. The equivalent words are then checked for presence of hyphen. Using this scheme ਖੇਤੀਬਾੜੀ < ਖੇਤੀ-ਬਾੜੀ < ਖੇਤੀਬਾੜੀਆਂ

3.3 Conjunct Consonants

Combination of two consonants is called conjunct consonants. In the combination, the first consonant is written fully, while the latter is written as a symbol below the first consonant but

pronounced fully. In Gurmukhi only three letters ਹ,ਰ,ਵ are used as conjunct symbols. The symbols of these conjunct consonants are ੍ਹ, ੍ਰ and ੍ਵ . Thus for example, the word ਪ੍ਰੇਮ is pronounced as half ਪ + ਰੇ + ਮ. It is also important to note that the vowel sign attached with the consonant is associated with the half character rather than the consonant under which the half character is lying. Thus ਪ੍ਰੇਮ is not pronounced as half ਪੇ + ਰ + ਮ but is is pronounced as half ਪ + ਰੇ + ਮ. So internally the order of storage is consonant followed by half character and then the vowel sign associated with the half character. Also the vowel sign ਿ is written before a consonant but as its sound is produced after the consonant so internally it is stored after the consonant.
As it can be observed from Table 1, in the collating sequence, the vowels come in the first place, followed by the consonants and half characters in that order. Thus ਕ੍ਿਆ < ਕ੍ੇਨ < ਕਾਰ as internally ਕ੍ਿਆ is stored as ਕ + ੍ + ਿ + ਆ , ਕ੍ੇਨ is stored as ਕ + ੍ + ੇ + ਨ while ਕਾਰ is internally stored as ਕ + ਾ + ਰ and since ੍ < ਾ and ਿ < ੇ so ਕ੍ਿਆ < ਕ੍ੇਨ < ਕਾਰ
By the same logic ਸਰਨਾ < ਸੈ < ਸਾਰ

4. ALGORITHM

We have developed a general sorting algorithm, which works on text encoded in any popular Gurmukhi font or ISCII. Since we have multiple-level comparisons to be carried out, so first the text to be sorted is transformed into a collation element table and then into equivalent sort keys[Davis and Whistler]. These sort keys might consist of a string of base weights followed by strings for weights used for secondary and tertiary differences. These keys are then sorted based on some sorting algorithm. In databases, these sort keys may be separately permanently in stored a field, obviating recomputation when a list needs to be re-sorted.
The algorithm makes use of following three levels as follows:

1. alphabetic ordering
2. semi-vowel ordering
3. hyphen based ordering.

A Collation Element Table contains a mapping from one (or more) characters to one *collation element*, where a collation element is an ordered list of two 8-bit weights (w_1 and w_2) and a third one bit weight w_3. Each collation element corresponds to a consonant, vowel or half character optionally followed by a semi-vowel and/or hyphen. The weight w_1 represents the collating code for the consonant, vowel or half character. If the character combinations contains a semi-vowel, then the weight w_2 is assigned the collating code of semi-vowel else w_2 is assigned value 0. Similarly if hyphen is a part of the character combination, the weight w_3 is assigned value 1 else it is 0.

The following are sample collation elements that are used in the examples illustrating the algorithm.

Character(s)	Collation Element
ੳ	[131.0.0]
ੳ-	[132.0.1]
ੳਂ	[132.128.0]
ੳੱ	[132.129.0]
ਅ	[135.0.0]
ਅਂ-	[135.130.1]

In the above example, for the first character combination, there is only a single character ੳ and no semi-vowel or hyphen and so in the equivalent collation element, w_1 = collating code of ੳ , while w_2 and w_3 are equal to 0. Similarly in the character combination, ਅੰ-, w_1 = collating of ਅ, w_2 = collating code of ◌ੁ and w_3=1, since hyphen is present in the character combination.

The main algorithm has four steps. First is to normalize each input string, second is to produce an array of collation elements for each string, and third is to produce a sort key for each string from the collation elements. Two sort keys can then be compared with a binary comparison; the result is the ordering for the original strings.

4.1 Normalize each input string

Step 1. Produce a normalized form of each input string, applying S1.1 to S1.3. This step is needed for font encoded strings only, where the vowel modifier ◌ਿ is stored before the consonant, but for sorting purpose its position has to be changed. Additionally the user can type consecutive occurring semi-vowels/upper vowels and half character/lower matra (◌੍ or◌ੁ) in any order and they all look visually similar. For example, the word ਖੁਲ੍ਹੇ can be typed by the typist as ਖ + ◌੍ + ਲ +◌ੁ +◌ੇ or as ਖ + ◌੍ + ਲ +◌ੇ +◌ੁ and both will be displayed as ਖੁਲ੍ਹੇ but the first word ਖੁਲ੍ਹੇ < ਖੁੱਲੇ while the second ਖੁਲ੍ਹੇ > ਖੁੱਲੇ, if we compare the two words character by character according the collating sequence of Table 1.

S1.1 Swap the positions of the vowel modifier ◌ਿ and its associated consonant. As already discussed, this step will not be needed for strings encoded in ISCII or Unicode as it has already been taken care of.

- For example, “ਸਿਰ” is rearranged as “ਸਰਿ”.

S1.2 If a half character or a lower matra (◌੍ or◌ੁ) and a semi-vowel occur together , then in case the semi-vowel comes before the half character or lower matra, interchange the two.

- For example, if “ਹੁੰਦੇ” is entered as ਹ + ◌ੁ +◌੍ + ਦ +◌ੇ then after S1.2 it will be internally rearranged as ਹ +◌੍ + ◌ੁ+ ਦ +◌ੇ.

S1.3 If a half character occurs in the string, then if there is a vowel sign immediately before it then interchange the two characters.

- For example, “ਕ੍ਿਆ” after S1.1 will be internally rearranged as ਕ + ◌ਿ +◌੍ + ਅ + ◌ਾ and after S1.3 will be internally stored ਕ +◌੍ + ◌ਿ + ਅ + ◌ਾ.
- Similarily, if the word “ਪ੍ਿੰਗਾ” is typed as ਪ +◌ੈ + ◌ੰ +◌੍+ ◌ਿ+ ਸ + ◌ੁ+ ਗਾ , after steps S1.1-S1.3 it will be internally stored as :

  i. After S1.1 : ਪ +◌ੈ + ◌ੰ +◌੍+ ਸ + ◌ਿ+ ◌ੁ+ ਗਾ
  ii. After S1.2 : ਪ +◌ੈ +◌੍ + ◌ੰ + ਸ + ◌ਿ+ ◌ੁ+ ਗਾ
  iii. After S1.3 : ਪ +◌੍ +◌ੈ + ◌ੰ + ਸ + ◌ਿ + ◌ੁ+ ਗਾ

4.2 Produce an array of collation elements for each string

Step 2. The collation element array is built by sequencing through the normalized form as follows:

S2.1 i=0; j=0;
S2.2 c=string[j]
If c not equals one of the semi-vowel or punctuation mark, then
w_1[i]=collation code of (c) (Table 1)
w_2[i]=0
w_3[i]=0
increment i
else if c equals semi-vowel
w_2[i-1] = collation code of(c) (Table 1)
else if c equals hyphen
w_3[i-1] = 1

S2.3 Increment j.

S2.4 GoTo S2.2 until the end of the string is reached.

Example:

normalized string:	collation element array: [w_1.w_2.w_3]
ਜੱਗ	[155.129.0], [149.0.0]
ਜੰਗ	[155.130.0], [149.0.0]
ਜਗ	[155.0.0], [149.0.0]
ਜਗਤ	[155.0.0], [149.0.0], [164.0.0]
ਜੈਕ	[155.0.0], [190.0.0], [146.0.0]
ਜਗ-ਮਗ	[155.0.0], [149.0.1], [174.0.0], [149.0.0]

4.3 Form a sort key for each string

Step 3. The sort key is formed by successively appending weights from the collation element array. It is implemented in the following steps:

S3.1. For each collation element in the collation element array append w_1 to the sort key.

S3.2. For each collation element in the collation element array append w_2 to the sort key.

S3.3. For each collation element in the collation element array append w_3 to the sort key.

Example:

collation element array:	sort key:
[155.129.0], [149.0.0]	155 149 129 0 0 0
[155.130.0], [149.0.0]	155 149 130 0 0 0
[155.0.0], [149.0.1], [174.0.0], [149.0.0]	155 149 174 149 0 0 0 0 0 1 0 0
[155.0.0], [190.0.0], [146.0.0]	155 190 146 0 0 0 0 0 0
[155.0.0], [149.0.0], [164.0.0]	155 149 164 0 0 0 0 0 0

4.4 Compare the sort keys

Step 4. Compare the sort keys for each of the input strings, using a binary comparison. This means that:

- Level 3 differences are ignored if there are any Level 1 or 2 differences

- Level 2 differences are ignored if there are any Level 1 differences
- Level 1 differences are never ignored.

Example:

String Sort Key

ਸ੍ਹੈ	143 183 190 000 000 000 000 000 000
ਸਾਰ	143 **184** 176 000 000 000 000 000 000
ਜਗ	**155** 149 000 000 000 000
ਜੱਗ	155 149 **129** 000 000 000
ਜੰਗ	155 149 **130** 000 000 000
ਜਗਤ	155 149 **164** 000 000 000 000 000 000
ਜਗਮਗ	155 149 **174** 149 000 000 000 000 000 000 000 000
ਜਗ-ਮਗ	155 149 174 149 000 000 000 000 000 **001** 000 000
ਜਗਮਗਾ	155 149 174 149 **184** 000 000 000 000 000 000 000 000 000 000
ਜੈਕ	155 **190** 146 000 000 000 000 000 000

The differences that produce the ordering are shown by the **bold underlined** items:

Formally, the Gurmukhi Collation algorithm is described in Fig. 1.

Gurmukhi Collation Algorithm

```
type
  CollitionElement = Record
                       w1, w2, w3 : byte;
                     end;
  SortKey :Array of Byte;
var
  Words : Array of String;
  NormalizedWord : String;
  CE : Array of CollitionElement;
  SK : Array of SortKey;
  i, count : integer;
begin
  i := 0;
  While i < count do
  begin
    Normalize(Words[i], NormalizedWord);
    GenerateCollationElement(NormalizedWord, CE);
    GenerateSortKey(CE, SK[i]);
    i++;
  end;
  Sort(SK, Words);
end;
```

Fig. 1 : Gurmukhi Collation algorithm

Count stores the number of elements in array Words.
The procedure Normalize(S1, S2) normalizes the string S1 and stores the normalized form in string S2 as discussed in 4.1
The procedure GenerateCollationElement(S1, CE) generates the CollitionElement array, CE, from normalized string S1 as discussed in 4.2.

The procedure GenerateSortKey(CE, SK) converts the CollitionElement array, CE, to SortKey SK as described in 4.3.
The procedure Sort(SK, Words) sorts the array SK using any of the standard sorting algorithm and simultaneously rearranges Words to produce the sorted output.

5. CONCLUSION

This is the first time that sorting for Gurmukhi has been implemented according to the Gurmukhi dictionary layout. Table 2 displays a set of Gurmukhi words automatically sorted based on the collating sequence and Gurmukhi sorting rules discussed in above sections.

Table 2 : Sorted list of Gurmukhi words

Sequence	Word
1.	ਉਪਾਊ
2.	ਉਪਾਅ
3.	ਉਪਾਇਆ
4.	ਉਪਾਇਆਂ
5.	ਉੱਭਰੇਂਗਾ
6.	ਉਭਰੇਗੀ
7.	ਉੱਭਰੇਂਗੀ
8.	ਸੜਾ
9.	ਸ੍ਵਸਥਤਾ
10.	ਸ੍ਵੰਬਰਾਂ
11.	ਸ੍ਵੈ
12.	ਸੁਲ੍ਹਾ
13.	ਸੁਲੇਖ
14.	ਸੌਲ੍ਹਾਂ
15.	ਸੌਲਾਂ
16.	ਸ਼ਮ੍ਹਾਂ

A Gurmukhi sorting utility, based on above algorithm, has been provided in **Akhar**, a Punjabi word processor. An electronic Punjabi-English dictionary has also been developed in MSAcess, which has been indexed on Punjabi words using the above collation algorithm.

References:

1. Harkirat Singh, " *Punjabi Shabad-Roop te Shabad Jorh Kosh"*, Punjabi University, Patiala. (In Punjabi)
2. Mark Davis, Ken Whistler: "Unicode Technical Standard #10, Unicode Collation Algorithm" (http://www.unicode.org/unicode/reports/tr10/)