Paper 297-2007

Creating Order out of Character Chaos: Collation Capabilities of the SAS System

Scott Mebust and Michael Bridgers, SAS Institute Inc., Cary, NC

ABSTRACT

Traditionally, data is ordered to facilitate further processing or to enable you to quickly find information in a report or other form of data presentation. The SAS System's primary means of achieving an alternative collating sequence has been to specify a *translation table* (TRANTAB), using the PROC SORT SORTSEQ option, with which PROC SORT can reorder individual characters. SAS® 9.2 extends the SORTSEQ option to enable the specification of an arbitrary encoding for *non-native* binary collation. SAS 9.2 also extends the SORTSEQ option to enable the specification of *linguistic collation*, which is useful for presenting data because it produces results that are more intuitive and culturally acceptable. The linguistic collation capability is highly compatible with the Unicode Collation Algorithm and adaptable to user preference using various options.

In this paper, we describe the four collation capabilities offered by PROC SORT in SAS. We further detail the applicability, the advantages, the processing requirements, and the processing implications of each approach. We conclude with information regarding the future directions of collation and sorting within the SAS System.

INTRODUCTION

In data processing, you often order data within a set to facilitate further processing, such as aggregation, for the production of summary statistics within the data set, or merging or joining the data with data from another set. It is also common to order a result set so that it can be presented in an organized form, such as a table or report, which allows information of interest to be quickly located. When using SAS, you might want to order observations within a data set according to the value of a numeric variable, a character variable, or some combination of variables of either type.

Collation, in a general sense, is the process of arranging items into a logical order. A collating sequence is a specific logical order based on one or more properties of the items being considered. There are a number of different techniques for collation of character data. While these techniques are applicable to operations within SAS, the general focus of this paper is the use of these collation techniques within the context of sorting. In SAS 9.2, four different techniques can be used with the SORT procedure and are recognized within the SAS System. These include classic binary collation, TRANTAB collation, transcoded binary collation, and linguistic collation. Each type of collation serves a slightly different purpose, offers its own advantages, and presents its own processing requirements.

We at SAS are especially excited about the addition of linguistic collation because it provides an ordering of character data that is culturally tailored, and therefore intuitive, making it much easier for people to search for and locate information. Linguistic collation is becoming increasingly popular and prevalent in computer applications and we believe that SAS customers will appreciate this new capability. Customer feedback regarding its usefulness will help us focus future development.

BINARY COLLATION

Within SAS, as within many other data processing systems, the primary method for determining the order of values of character variables or character strings is based upon a direct examination of the characters as they are represented in a computer's memory. Traditionally, character data is represented in a computer's memory using an encoding method that maps some or all of the values of a single byte to desired numeric, alphabetic, and typographical symbols. In this method, every character has an associated integer value used to represent the character in one byte of memory. In contemporary terminology, a particular mapping between byte values and characters is called a *single-byte encoding* or *code page*. The integer to which a particular character maps within a single-byte encoding is called a *code point*. The process of establishing the order of the values of a character variable or set of character strings by directly examining the code points associated with the characters, one by one beginning with the first character in the variable or string, is called *binary collation*. SAS primarily performs a byte-by-byte *binary comparison* between two character strings to determine their equivalence or relative order and performs *binary collation* when sorting by a character variable key.

The sequence of character variable values produced by binary collation is strictly dependent upon the code points of the characters within an encoding. That is, a binary collation will produce a sequence that reflects the order of appearance of characters within a code page and this sequence may not match expectations. For instance, within any code page based upon the American Standard Code for Information Interchange (ASCII) encoding, the capital letter 'Z' precedes the lowercase letter 'a'. Similarly, for any code page based upon the Extended Binary Coded Decimal Interchange Code (EBCDIC) encoding, the lowercase letter 'z' precedes the capital letter 'A'. Given these examples, a binary collation of character data encoded under either standard will not meet the expectations for

alphabetic ordering within the English language (or any other Latin-based language, which incorporates these letters). Further, code pages based on either of the ASCII or EBCDIC encodings will not produce a proper alphabetic ordering for languages which incorporate accented characters or characters that are not part of the 26 characters common to the English language. For these reasons, SAS provides the SORTSEQ option for the specification of an alternative character collating sequence.

TRANTAB COLLATION

The SORTSEQ option, either that of the SAS System or a SAS procedure, can specify the name of a system-provided or user-created translation table. Translation tables can be created with the TRANTAB procedure and are used to translate character data from one encoding to another – a process called *transcoding*. A translation table can be used by a procedure that recognizes the SORTSEQ option to effectively reorder characters when performing binary collation. The result is a *weighted binary*, a *translated binary*, or, within SAS, a *TRANTAB collation*. Translation tables within SAS have been the primary means of modifying character collating sequences for achieving compatibility between different platforms and, perhaps more importantly for international users, for providing collating sequences that are more culturally acceptable to speakers of non-English languages.

With TRANTAB collation, the translation table is used to remap the code point of each character of a variable value to another integer within the range of an 8-bit byte (0 to 255). The remapping is performed by using the code point of a character as an index into the translation table to retrieve the replacement integer. If the translation table was created for transcoding from one single-byte encoding to another, then each character is remapped from its code point in the source encoding to its code point in the destination encoding. This remapping enables you to achieve binary collation compatibility between two systems that use different single-byte encodings. Translation tables do not, necessarily, have to be constructed for transcoding. For the purposes of collation, translation tables might be created, for example, to order characters so that the lowercase and uppercase variants alternate, in an attempt to correct the situation in which 'Z' precedes 'a' within an ASCII-based encoding. A translation table might also be created so that accented alphabetic characters immediately follow their unaccented counterparts, such as 'à' ('a' with accent grave) following 'a' in French. Such translation tables can reorder the characters or can be viewed as a method of re-weighting the characters during collation.

TRANTAB collation is intended only for use with single-byte character encodings and is not capable of correctly handling multi-byte character encodings. Multi-byte encodings were created to accommodate languages requiring the use of more than 256 characters, the limitation imposed by a single-byte encoding. It is not possible to create a translation table with the TRANTAB procedure that will handle a multi-byte encoding as either the source encoding or the destination encoding. Translation tables are limited to remapping or reordering up to 256 characters from a single-byte encoding. While TRANTAB collation can be used to achieve an alphabetic ordering of individual characters, it is limited in its ability to create a collating sequence that is completely intuitive and that meets cultural expectations. For these reasons, two new forms of character collation are being introduced in SAS 9.2: *transcoded binary collation* and *linguistic collation*.

TRANSCODED BINARY COLLATION

The binary collation of data encoded in the SAS session encoding is referred to as a *native binary collation*. When a translation table is used to alter the collating sequence by transcoding character data from one to another single-byte character encoding, the result is a *non-native binary collation*. For example, you can specify SORTSEQ=EBCDIC on a system using an ASCII-based session encoding. However, TRANTAB collation is limited to single-byte encodings and, therefore, cannot serve as a general method for achieving any non-native binary collation. Furthermore, the exact collating sequences achieved using the translation tables provided by the SAS System, such as ASCII and EBCDIC, may not be obvious nor provide the desired results. A natural extension to TRANTAB collation is to allow for a binary collation of character data that is transcoded to any non-native encoding, including multi-byte encodings.

In SAS 9.2, the SORT procedure has been enhanced to recognize the names of arbitrary encodings specified using the SORTSEQ option. If the name of an encoding, which does not conflict with an existing translation table name, is specified with the SORTSEQ option, PROC SORT will transcode character data to that encoding before performing a binary collation. This more general ability, called *transcoded binary collation*, enables PROC SORT to generate collating sequences that are compatible with the binary collating sequences of other data processing systems. PROC SORT recognizes the same set of about 100 encodings that is recognized by the ENCODING option used elsewhere in SAS. However, transcoded binary collation presents the same limitations as binary collation in that the collating sequence is dependent upon the order of appearance of characters within the destination encoding and cannot produce collating sequences that are intuitive and culturally correct.

LINGUISTIC COLLATION

Although we have become somewhat accustomed to the collating sequences produced by the binary, TRANTAB, and transcoded binary collation methods, none of these methods produce sequences that are intuitive. That is, none of these methods produce sequences that we associate with printed materials like dictionaries, phone books, and book indexes. The collating sequences used in print were developed and established hundreds of years before the advent of the computer and have become culturally ingrained. We find that locating a word in the dictionary, a name in the phonebook, or an entry in an index is a relatively simple task because we know, or at least have a good idea, of how the items are arranged. Locating items within a computer-generated report, on the other hand, might be difficult

if we do not immediately realize that, for example, by virtue of the collating sequence used for items within the report, words beginning with uppercase letters are listed separately from words beginning with lowercase letters or that words beginning with accented characters are listed after words beginning with unaccented characters. Globally, different cultural expectations have developed with respect to collating sequences in printed materials for the multitude of languages and writing systems that exist. Linguistic collation addresses these cultural expectations and produces sequences that are deemed reasonable and acceptable throughout the various countries and regions of the world. In SAS 9.2, the SORT procedure has been enhanced to provide linguistic collation. Setting the PROC SORT SORTSEQ option to LINGUISTIC will cause PROC SORT to collate linguistically in accordance with the SAS System LOCALE setting. Options for modifying properties of the linguistic collating sequence may be specified within parentheses following the LINGUISTIC key word.

USES, ADVANTAGES, AND PROCESSING REQUIREMENTS

In SAS, binary collation, TRANTAB collation, transcoded binary collation, and linguistic collation are best applied under different circumstances as they not only provide different results, but each necessitates different processing requirements in computational time, memory space, and storage space. Currently, the results of all but binary collation must also be treated in a special manner during processing because support for alternative collating sequences and collation sensitive processing within SAS is limited.

Binary collation produces results that are useful for internal data processing and that are often acceptable by computer users who have, to an extent, come to accept them. It is the default within SAS and presents a baseline for computational time, memory, and storage requirements. Most sorting of character data performed internally, within the SAS System, is processed with binary collation, although some procedures support the SORTSEQ option for TRANTAB collation. It is also the expected form of collation and, therefore, universally supported during further processing of sorted data within the SAS System.

Alternative collating sequences are best used when there is a desire for compatibility with sequences created by other applications, in support of national languages, or for final presentation of data processing results. The alternative sequences produce results that are more desirable, but do so with the cost of additional computing resources. In addition, with few exceptions, the results must be processed in ways that ignore the collating sequence.

TRANTAB collation is a useful alternative when the results are deemed acceptable or when they are necessary for compatibility with SAS or with other data processing systems. It presents minimal additional computational time, compared with binary collation, and no additional memory or storage requirements. The results of TRANTAB collation must be processed in a manner that ignores the collating sequence.

Transcoded binary collation is useful when, for compatibility or user preference reasons, the desired collating sequence is that of a non-native encoding. It presents additional computational time, memory, and possibly storage space. Like TRANTAB collation, the results must also be processed in a way that ignores the collating sequence.

Finally, linguistic collation is useful when a more intuitive or culturally correct sequence is desired for report generation or other data presentation, as well as for compatibility between systems. Linguistic collation is available for the 100+ locales and 50+ languages supported within SAS 9.2. It, too, presents additional computational time, memory, and possibly storage space. The results must also be processed in a way that ignores the collating sequence.

SORTING AND POST-COLLATION PROCESSING

Within SAS, binary collation of character data is the default whenever sorting is performed either explicitly or implicitly by a procedure or some other part of the SAS System. Both the SORT procedure and the SQL procedure, however, will use TRANTAB collation when the SORTSEQ option is set to the name of an existing translation table. Most parts of the SAS System that produce a data set, which is known to be sorted, will indicate the order of the data within the descriptor information in the data set. This descriptor information enables subsequent processing within the SAS System to be optimized by avoiding redundant sorting. That is, if possible, subsequent processing will take advantage of existing order within a data set. The sort indicator, contained in the descriptor information in the data set, contains a field for describing the collating sequence used when ordering observations by character variables.

In SAS 9.2, the SORT procedure is also capable of performing both a transcoded binary and a linguistic collation when ordering a data set. These new forms of collation introduce new metadata into the sort indicator. No new sort indicator fields have been introduced, but new values for the collating sequence field are created by both transcoded binary collation and linguistic collation. These new collating sequence values and their interpretation are explained below, as the new collation forms are explained in detail.

The collating sequence used for sorting can affect both the internal operation of procedures performing the sorting, as well as subsequent processing of the results by other parts of the SAS System. An obvious example of an effect on internal operations is the use of a case-insensitive collating sequence within the SORT procedure and its effect on the observations eliminated through NODUPKEY option sub-setting. A subtler example is the effect that an alternative collating sequence, such as a case-insensitive one, can have on the processing of a query within the SQL procedure. The results of query processing that involves sorting, either explicitly with an ORDER BY option or implicitly through operations such as UNION, INTERSECT, and EXCEPT, are affected by the specified collating sequence.

While some procedures and parts of the SAS System examine the sort indicator of a data set in order to avoid redundant sorting, other parts of the SAS System, which expect or assume data to be sorted, are NOT sensitive to the collating sequence previously established within the data set descriptor information. Data sorted using a binary character collating sequence requires no special treatment during subsequent processing because, within the SAS system, this is the expected collating sequence and it is universally supported. Data sorted according to an alternative collating sequence, however, may need to be treated in a special manner after the sort order has been established.

One example in which a data set sorted with an alternative collating sequence must be treated in a special manner is during BY processing. Normally, BY processing checks whether the observations being processed are in a sorted sequence and also determines the boundaries between groups of observations with equal BY variable values. The sequence check serves as a safety measure to guard against unintentional BY processing when the observations are not grouped. If the observations are in sorted order, then they are inherently grouped. With few exceptions, BY processing is not sensitive to collating sequence, but performs a binary comparison to determine whether the observations being processed are in sequence and to determine observation group boundaries. Observations that are arranged according to an alternative collating sequence of the BY variables will likely not pass a binary comparison sequence check. Therefore, when you are BY processing data that has been sorted using any alternative, non-binary collating sequence, the observation sequence check must be disabled locally using the NOTSORTED option or disabled globally using the NOBYSORTED option. Ignoring the observation sequence in this manner is similar to processing observations for which the BY values are known to be grouped, but the groups do not necessarily appear in sequence. With the NOTSORTED option, group boundaries are still determined by any difference in the raw variable values. For character variables, standard BY processing effectively examines the code point values of the characters when determining group boundaries.

One exception to this insensitivity to collating sequence is that, since SAS 8.0, the DATA step will recognize a TRANTAB collating sequence when performing BY processing. The DATA step uses the associated TRANTAB to translate character BY variables while performing the sequence check and group boundary detection. This ability was added in support of national language processing using the language-oriented TRANTABs provided with the SAS System.

Another example in which the alternative collating sequence must be ignored is when a procedure would normally perform an implicit sort of the data using a binary collating sequence, but processing or presenting the data by an alternative collating sequence is desired. Some procedures provide the ORDER=DATA option to override the implicit sort and process the data according to its input order.

In addition, insensitivity to collating sequence can result in the disabling of certain optimizations during processing. For instance, with a Base SAS engine data set, sub-setting of a WHERE statement inequality can be terminated early, before the entire data set has been read, when the WHERE processing code can determine that, due to the order of data within the data set, the constraints can no longer be met. This WHERE processing optimization works only with a binary collating sequence and will be disabled for any alternative collating sequence.

BINARY COLLATION

DESCRIPTION

The simplest collating sequence for character variable values or character strings in a single-byte encoding is based upon a direct examination of the code points associated with the characters and is commonly referred to as a *binary* or *bitwise* collation. The terms binary and bitwise are used to describe this type of collating sequence because the characters are stored in memory as code point bit patterns and strings of characters are stored as arrays of code points in a contiguous area of memory.

The underlying representation of the character strings, then, can be viewed at its lowest level as an array of bits; the relative order of two character strings can be determined by a bit-by-bit comparison, from the most significant bit to the least significant bit, of the associated bit arrays. The strings are compared one byte (8 bits) at a time using either a single complex machine instruction or a loop containing a sequence of simple machine instructions. Comparison of character strings in this manner to achieve a binary collation is a relatively inexpensive and quick operation for a computer. This internal representation is also useful for methods of collation that do not perform direct comparisons of the character strings, but that distribute the strings based on an examination of the code point values from the most significant byte to the least significant byte.

EXAMPLES

In the example shown in Table 1, the string “`apple`” appears before “`apply`” in an ascending binary collating sequence when both strings are represented in the ISO 8859-1 (ASCII-based Latin-1 character encoding), because the code points for the first four characters of each string are equivalent and, for the fifth character, the code point of ‘e’ (`101`) is less than the code point of ‘y’ (`121`).

Table 1: Binary Collation Example 1 (ASCII)

apple	a	p	p	l	e	(character)
	97	112	112	108	101	(code point)
apply	a	p	p	l	y	(character)
	97	112	112	108	121	(code point)

However, in this same encoding, the string "Zeus" appears before "able" in an ascending binary collating sequence (see Table 2) because the code point for the character 'Z' (90) is less than the code point for character 'a' (97).

Table 2: Binary Collation Example 2 (ASCII)

Zeus	Z	e	u	s	(character)
	90	101	117	115	(code point)
able	a	b	l	e	(character)
	97	98	108	101	(code point)

ADDITIONAL DETAILS

Binary collation is the default within SAS and is the expectation throughout the SAS System. The collating sequence that is produced is dictated by the encoding of the data being manipulated, after the data has been read into system memory, which is normally the session encoding. A binary collation of the data that is encoded in the session encoding is referred to as a *native* binary collation.

Binary collation produces results that, for small samples, appear reasonable on the surface and are deemed acceptable or at least adequate to many computer users. However, for larger samples or certain specific examples, the results may appear quirky upon closer examination. Binary collation results in the separation of lowercase and uppercase, as well as accented, characters into separate groups within the results. This can cause difficulties when dealing with mixed-case text or words of international origin. While binary collation produces results that may not be intuitive, it is the form of collation that uses the fewest computing resources. Extra space, either memory or external storage, is not necessary for binary collation because character strings can be directly examined or compared, character by character. A comparison of two character strings is accomplished quickly using simple machine instructions.

Binary collation was widely used during the 20th century and remains in use today due to its simplicity and efficiency. The results are well-suited for the bulk of data processing and are often acceptable when used to present result sets. However, its disadvantages include a lack of true alphabetic ordering and results that depend upon the encoding used and that, therefore, differ across systems.

Binary collation provides a baseline for the efficiency and performance of the SAS sorting routines used by the SORT procedure, as well as many other procedures in the SAS System. The SAS sort creates a composite collatable-format sort key from the BY variables of an observation by copying, and possibly transforming, the variable values from the memory containing the observation into an area of memory reserved for the composite sort key. For binary collation, the memory required for a character sort key is equivalent to the size of the character variable. Other character collation techniques may require a key space that is significantly larger than the input string. In addition, the SAS single-threaded sort saves both the observations and the keys to a utility file on disk when it cannot complete the sort using available memory. Thus, the storage space required for a character sort key is also equivalent to the size of the character variable. The SAS multi-threaded sort, on the other hand, saves only the observations to the utility file.

No special care is needed with subsequent processing of sorted data when using binary collation. Binary collation is expected and is the standard throughout the SAS System. Data collated in this fashion is compatible with the BY processing sequence check.

Being the default, no SORT procedure options are required to enable binary collation. However, care should be taken when allowing a sort request to be passed through an ACCESS engine to an underlying database management system (DBMS) as an ORDER BY option. If allowing such pass-through, the DBMS should be configured to return data ordered according to a binary collation of the same encoding being used by SAS (the session encoding).

Upon completion of a sort, the SORT procedure stores the data encoding family of the session encoding, known as the character set, within the data set descriptor information. Examples of the character set indication include the

acronyms *ASCII* and *EBCDIC*. To denote a native binary collating sequence, the collating sequence name field within the sort indicator is set to all blanks.

TRANTAB COLLATION

DESCRIPTION

A TRANTAB collation, also referred to as weighted binary or translated binary collation, is a simple method of reordering characters using a *translation* or *lookup* table. While binary collation orders characters according to the magnitude or *weight* of their associated code points, TRANTAB collation offers the ability to map the character code points to different weight values.

If the new weights are equivalent to the code points of the characters within another encoding, then the collation is a simple form of transcoded binary collation. That is, a translation table that has been created to transcode from one single-byte encoding to another can be used, with character data in the source encoding, to achieve a collating sequence equivalent to that obtained when performing a binary collation on the same data in the destination encoding. Such a collation is referred to as a *non-native* binary collation. Translation tables can also be used to reorder characters within a code page in support of national languages and cultural conventions.

Translation tables are practical for single-byte character encodings because the table can be expressed in a total of 256 bytes. TRANTAB collation is efficient in both memory and storage space and computational time because the translation can be done in-place or, at least, in a space no larger than its input and the translation operation itself is implemented with a few simple machine instructions. After the translation, character strings are examined in a binary fashion. Some translation tables are provided with the SAS System, but the TRANTAB procedure enables SAS users to create their own tables. Translation tables within SAS actually consist of two 256-byte tables, with the second table facilitating an inverse transcoding or mapping that enables recovery of the original character data.

EXAMPLES

You may desire to construct a custom translation table for use with specific source and destination encodings. For example, on a UNIX machine running SAS with a session encoding of Latin1 (ISO 8859-1), you might desire to achieve the same collating sequence that you would obtain on an IBM mainframe running SAS with a session encoding of OPEN_ED-1047 (Open Edition EBCDIC cp1047-Latin1).

To perform TRANTAB collation, the translation table is used to transcode all characters of an input string from a source encoding to a destination encoding. This is done by using the source encoding code point value for each character as an index into the translation table to obtain the code point value of the same character in the destination encoding. Although there are 256 entries in a translation table, the index into the table is zero-based and ranges from 0 to 255.

In the example shown in Table 3, you would transcode the character ‘A’, from its code point of 65 in an ASCII-based encoding, to its code point of 193 in an EBCDIC-based encoding by examining the 66th entry in the table (the table entries are numbered starting with zero). Collation is then performed in a binary fashion by examining the transcoded string byte by byte.

Table 3: Example ASCII to EBCDIC Translation Table

		ASCII to EBCDIC Translation Table
Character	ASCII Code Point	EBCDIC Code Point
. . .	. . .	. . .
A	65	193
B	66	194
. . .	. . .	. . .
a	97	129
b	98	130
. . .	. . .	. . .

On a machine with a session encoding of ISO 8859-1 and a TRANTAB collation using SORTSEQ=EBCDIC, the string "Zeus" will appear after "able" in an ascending binary collating sequence (see Table 4) because, within the destination encoding, the code point for the character 'Z' (233) is greater than the code point for character 'a' (129).

Table 4: TRANTAB Collation Example 1

able	a	b	l	e	(character)
	97→129	98→130	108→147	101→133	(ASCII→EBCDIC code point)
Zeus	Z	e	u	s	(character)
	90→233	101→133	117→164	115→162	(ASCII → EBCDIC code point)

In an attempt to correct situations like characters appearing out of alphabetical order, such as 'Z' appearing before 'a' in an ASCII-based encoding, you can construct a custom translation table that reorders the characters so that differences in the case of a character become less important than the alphabetic difference between characters. Alternating lowercase and uppercase character variants ('a' < 'A' < 'b' < 'B' < …) achieves this goal. It helps to think of the values in the translation table as weights assigned to each character. Without a translation table, the weight assigned to each character is the integer value of the associated code point. This is similar to using an identity translation table, which maps the code points of each character to a weight that is equivalent to the code point of the character (0 → 0, 1 → 1, 2 → 2, …).

When creating a translation table, it is beneficial to create the table, if possible, with a one-to-one mapping between characters and weights. The use of a translation table (through PROC SORT) with a host sort requires that the table have an inverse because the translation is performed directly upon variables within an observation and must be undone when the sorted observations are retrieved. Without a valid inverse table, data corruption is possible. PROC TRANTAB can only create an inverse of a table if the table is a one-to-one mapping. To create a one-to-one table and to avoid altering the weights of any other characters, we must reuse the identity weights of the characters we want to rearrange. For example, in an ASCII-based encoding, the code points of the characters 'A' through 'Z' are 65 through 90 and the code points of the characters 'a' through 'z' are 97 through 122. To achieve the desired reordering, we can use weights within these two code point ranges.

If lowercase character variants are to appear first and alternate with the uppercase variants, we assign the odd weights from these code point ranges to the lowercase characters and even weights from these code point ranges to the uppercase characters. That is, we assign weight 65 to 'a', 66 to 'A', 67 to 'b', 68 to 'B', and so on. We will call this translation table LOWFIRST (see Table 5).

Table 5: Example LOWFIRST Translation Table

		Alternating Lower/Upper Case ASCII Translation Table
Character	ASCII Code Point	Weight
.	.	.
.	.	.
.	.	.
A	65	66
B	66	68
.	.	.
.	.	.
.	.	.
Z	90	116
.	.	.
.	.	.
.	.	.
a	97	65
b	98	67
.	.	.
.	.	.
.	.	.
z	122	115
.	.	.
.	.	.
.	.	.

While an ascending binary collation of an ASCII-based encoding yields the string order "Aztec□□□" < "Zeus□□□□" < "aardvark", where the symbol '□' represents a space character, a TRANTAB collation using the LOWFIRST translation table yields the string order "aardvark" < "Aztec□□□" < "Zeus□□□□."

The language-oriented SAS System translation tables, such as the FINNISH table, are designed to order the characters by re-weighting them so that they appear in proper alphabetical order for the various languages. These tables may maintain the separation of lowercase and uppercase letters that is inherent in both ASCII-based and EBCDIC-based encodings or they may alternate lowercase and uppercase characters, like in the LOWFIRST translation table example.

You can also implement case-insensitive collation by creating a translation table that assigns equal weights to both lowercase and uppercase character variants. Such a table would, of course, not be a one-to-one mapping between characters and weights and would not be suitable for use with a host sort.

ADDITIONAL DETAILS

Weighted collation was an early attempt at dealing with the problems associated with binary collation. It was a simple solution to obtain a non-native binary collating sequence or to achieve an improved collating sequence for languages that used accented characters or characters outside of the set used in the English language.

General TRANTAB collation was introduced in SAS 6.06 with the creation of PROC TRANTAB, although SAS shipped with built-in translation tables prior to that release. With PROC TRANTAB, users can create their own translation tables. The SORTSEQ= *option* was introduced at the same time to enable alternative collating sequences through the specification of arbitrary translation tables. Translation tables provided with the SAS System include ones for transcoding from EBCDIC to ASCII, for transcoding from ASCII to EBCDIC, and others that reorder alphabetic characters according to the rules of various languages.

TRANTAB collation provides elementary support for transcoded collation. Using translation tables created for the purpose of transcoding enables you to achieve a non-native binary collating sequence for a single-byte encoding. This is beneficial when compatibility with another binary collating sequence is desired. However, the result is still a binary collating sequence, and therefore presents the same problems, such as a non-alphabetic ordering, as binary collation. Translation tables are limited to transcoding between two single-byte encodings and cannot be used with multi-byte encodings.

TRANTAB collation provides elementary support for foreign languages. Collating sequences that are produced by the language-oriented translation tables might seem reasonable and are sufficient for some needs. However, problems with this type of collation are evident upon closer inspection and are easy to demonstrate. While individual characters can be reordered using a translation table, mixed-case strings or strings containing accented characters do not collate in a reasonable alphabetic order. Due to its simplicity, TRANTAB collation is inherently incapable of producing a collating sequence that is intuitive, such as the order found within a dictionary. Like binary collation, TRANTAB collation produces quirky results that many users have come to accept.

TRANTAB collation is only slightly more complicated than binary collation. In general, it does not require any additional memory or storage space because the translation can be performed in-place, directly on the existing character strings. The translation process does require slightly more computational time but, because a table lookup requires few simple machine instructions, the extra time should be negligible. The SAS sorting routines used by PROC SORT simply require a minor amount of additional computation time and do not require any additional memory or storage space, compared with that used for binary collation.

Generally, the results of TRANTAB collation must be processed as though the data are grouped, but not sorted. Either the NOTSORTED option for the BY processing statement or the NOBYSORTED option should be used to disable the observation sequence check normally performed during BY processing. One exception to this requirement is BY processing from within the DATA step. The DATA step contains its own implementation of BY processing and is sensitive to a TRANTAB collation sequence within a data set. The DATA step uses the associated table to translate character BY variables during BY processing. Thus, the DATA step is able to both verify an alternative collating sequence and define BY groups by checking for a binary difference in the translated character values.

TRANTAB collation is not the default within the SAS System. It must be requested either through a procedure SORTSEQ option or through the SAS System SORTSEQ option. The SORTSEQ option must refer to a valid translation table, either one supplied by the SAS System or one created by a user with PROC TRANTAB.

Upon completion of a sort, in addition to setting the character set field in the sort indicator in the descriptor information of the data set, the SORT procedure also stores the name of the specified translation table within the collating sequence field. An exception to this action is that the collating sequence field is not set if the translation table is a system table that results in a native binary collation (for example, specifying SORTSEQ=ASCII when the session encoding is ASCII-based). The SORT procedure subsequently recognizes that a data set has been sorted using TRANTAB collation, if asked to perform a redundant sort using the same translation table.

TRANSCODED BINARY COLLATION

DESCRIPTION

A transcoded binary collation is obtained by first transcoding the character data under consideration from its native encoding to a non-native encoding and then performing a binary collation on the transcoded data. Transcoded binary collation is most useful for obtaining a collating sequence that is compatible with another data processing system. Other data processing systems may use encodings that are different from the SAS session encoding. If these systems produce data sets or tables that are sorted using binary collation, it may be desirable to match the collating sequence within SAS when importing the data. Likewise, if these systems expect data to be sorted using a binary collating sequence, it may be desirable to export data that is sorted accordingly. Compatibility between two instances of SAS, running on different platforms for which the session encodings differ, can also be achieved using transcoded binary collation.

Transcoded binary collation can be achieved using translation tables when both the source encoding and destination encoding are single-byte encodings. Transcoding when either the source or destination encoding is a multi-byte encoding is not possible using translation tables. General transcoded binary collation for arbitrary encodings, new in SAS 9.2, overcomes this limitation.

Use of transcoded binary collation, within PROC SORT, is done by specifying the name of the destination encoding as the value to the SORTSEQ option. The encoding names understood by PROC SORT are the same as those that are acceptable for use with the ENCODING option found elsewhere in the SAS System. Quotation marks may need to be used around the name if it contains any characters other than the standard alphabetic and underscore characters usable in SAS variable names. If the specified name does not conflict with the name of an existing translation table, then transcoded binary collation is performed; otherwise, a TRANTAB collation is performed.

EXAMPLES

Transcoded binary collation might be performed to sort a data set within SAS, running on a mainframe with an EBCDIC-based OPEN_ED-1047 session encoding, in preparation for transporting the data set to or accessing it from SAS, running on a personal computer with a Microsoft Windows operating system and an ASCII-based Windows code page 1252 session encoding. To do so, you can specify one of the SAS names for the Windows 1252 code page:

```
proc sort data=foo SORTSEQ=”Western (Windows)”;
by x;
run;
```

or

```
proc sort data=foo SORTSEQ=wlatin1;
by x;
run;
```

The 4-byte, 12-byte, and 32-byte SAS encoding names are all recognized by the SORTSEQ option in the SORT procedure.

Another example might be a desire to obtain a Unicode UTF-8 binary collating sequence when the SAS session encoding is set to “`Japanese (SJIS)`”. Again, you do this by setting the SORTSEQ option to the destination encoding:

```
proc sort data=foo SORTSEQ=”Unicode (UTF-8)”;
by x;
run;
```

ADDITIONAL DETAILS

Transcoded binary collation is a natural extension of the use of translation tables for the binary collation of non-native single byte encodings. Unlike TRANTAB collation, the destination encoding is explicit and transcoded binary collation can handle multi-byte source and destination encodings.

Transcoded binary collation requires more processing time, more memory, and potentially more storage space when compared with binary collation. In PROC SORT, character BY variables must be transcoded to form the composite sort key and this requires more time than simply copying the character values. The memory reserved within the key space, for character variables, is larger than that in binary collation. With binary collation, the memory required for character variables is equivalent to their length (in bytes). For transcoded binary collation, the space reserved is large enough to hold the results of any transcoding. The SAS single-threaded sort stores these sort keys, along with the observations, in a utility file and so it requires additional space. The SAS multi-threaded sort does not store sort keys within the utility file and so does not require any additional utility file storage space. Specification of transcoded binary collation causes one of the SAS sorts to be used; there is no support for transcoded binary collation with host sorts.

The results of transcoded binary collation must be processed as though the data are grouped but not sorted. Similar to TRANTAB collation, the NOTSORTED and NOBYSORTED options should be used for general BY processing. At this time, neither the DATA step nor any procedure, apart from PROC SORT, is capable of performing a sequence check for this alternative type of collation.

Transcoded binary collation introduces new values into the collating sequence field of the sort indicator that is in the descriptor information in the data set. The new collating sequence values consist of the short (4-byte) SAS name for the destination encoding. The SAS name is both preceded and followed by an underscore character to indicate that this collating sequence value is a special value reserved for system use only. An example of this is the value *_WLT1_*, which indicates the “`Western (Windows)`” or “`wlatin1`” encoding.

The SORT procedure, as well as some parts of the SAS System that perform optimizations when data is known to be sorted using a binary collating sequence, recognize the case that the encoding recorded in the collating sequence of the sort indicator matches the SAS session encoding. This situation indicates that the collating sequence is equivalent to a native binary collation.

LINGUISTIC COLLATION

DESCRIPTION

The term *linguistic collation* refers to achieving collating sequences that are culturally acceptable and, therefore, intuitive. Being more intuitive, linguistic collation is highly desirable for presenting data in tables or reports but, due to the use of additional computing resources required to implement linguistic collation, it may not be desirable for general data processing. Methods for linguistic collation vary based on language and writing system. For many languages, such as those based on the Roman (or Latin) alphabet, linguistic collation is attained using a multi-level algorithm in which character attributes are considered separately.

For example, you might use a three-level algorithm that considers alphabetic properties of the characters first, accent marks (diacritics) second, and the case of the characters third. At each level, a character is assigned a weight for the attribute under consideration. On the first level, the characters 'A', 'a', and 'â' would be considered equivalent because, alphabetically, they are all considered to be variants of the letter *A*. On the second level, the characters 'A' and 'a' would be considered equivalent and distinct from 'â' because the former characters are unaccented while the latter character is accented with a circumflex. On the third level, the characters 'a' and 'â' would be considered equivalent and distinct from 'A' because the former are lowercase characters while the latter is an uppercase character. Character strings are ordered by considering one level at a time. As an example, the strings "Age," "age," and "âgé" are all equivalent at the first level. Consideration of the alphabetic property on the first level causes these three strings to appear in a collation before a string such as "zoo". On the second level, the strings "Age" and "age" are equivalent and distinct from "âgé". This second level consideration causes strings with unaccented characters to be grouped separately from those with accents. On the third level, the strings "age", and "âgé" are equivalent and distinct from "Age." This third-level case consideration causes strings with lowercase and uppercase to be grouped separately. Taking all three levels into account, one level at a time, these three strings are all distinct. Linguistically, they might be ordered as "age" < "Age" < "âgé" within a collating sequence.

True linguistic collation for languages using a Latin-based alphabet may involve more than three levels and must deal with complications such as ligatures (character contractions such as 'æ'), digraphs (two adjacent characters which should be treated as a single, distinct character), and specific rules for ordering accented characters. Languages using other alphabets or ideographic writing systems, such as Arabic and Chinese, present their own unique ordering requirements that may need to be tailored based on the particular region within a locale.

Recognizing the growing importance of globalized software, we wanted to provide linguistic collation capabilities within SAS. Many vendors use proprietary codes for linguistic collation, but these can cause differences in collating sequences among data processing applications. Instead of creating another proprietary implementation, we decided to investigate the possibility of incorporating the International Components for Unicode (ICU), a liberally licensed open source library maintained by IBM and others, into SAS 9.2. The ICU contains a highly optimized implementation of linguistic collation, based on the Unicode Collation Algorithm (UCA), for use with Unicode character data. The ICU is being widely incorporated into many products worldwide and its UCA implementation is quickly becoming a *de facto* standard. Using the ICU has the obvious benefit of not having to develop a complete proprietary system for linguistic collation but, more importantly, enables us to achieve instant collating sequence compatibility with other data processing systems that also incorporate this code. That we at SAS were able to incorporate the ICU and leverage its linguistic collation capabilities, on 10 of the 11 platforms for which SAS 9.2 support is planned, is a testament to the quality of the ICU and the effort placed on portability by its developers.

The ICU implementation of the UCA is useful for both comparison-based sorting and distribution-based sorting, because it can compare two character strings to determine their relative order or it can produce a collatable format byte array, known as a sort key, for a given string. For use from within SAS, character strings must be transcoded to UTF-16, the 16-bit Unicode encoding that is expected by the ICU, before strings can be compared or sort keys can be generated. The use of the ICU for linguistic collation from within SAS requires more computational time, memory, and storage space than, for example, simple binary collation. In particular, PROC SORT requires more time, more memory, and possibly more storage space to effect a linguistic collation. The concentration of this first phase effort has been on providing the linguistic collation functionality within the framework and constraints of the current PROC SORT implementation. As such, the implementation is not as efficient as it can possibly be, but we expect this to improve in future releases.

Invocation of linguistic collation with PROC SORT is quite simple. The only requirement is the specification of LINGUISTIC as the value to the SORTSEQ procedure option:

```
proc sort data=foo SORTSEQ=LINGUISTIC;
by x;
run;
```

Alternatively, you can specify SORTSEQ=UCA. This will cause the SORT procedure to collate linguistically, in accordance with the current system LOCALE setting. The collating sequence used is the default provided by the ICU for the given locale. Options that modify the collating sequence can be specified in parentheses following the LINGUISTIC or UCA keywords. Generally, it is not necessary to specify option settings as the ICU associates option

defaults with the various languages and locales. PROC SORT currently allows only a subset of the ICU options to be specified. These options include STRENGTH, CASE_FIRST, COLLATION, and NUMERIC_COLLATION. In addition, a LOCALE option is available to instruct PROC SORT to use a collating sequence associated with a locale other than the SAS System locale.

The STRENGH option specifies the collation strength and corresponds to the various levels in the multi-level collating algorithm. STRENGTH can be set to a number between 1 and 5 or to the corresponding values PRIMARY, SECONDARY, TERTIARY, QUATERNARY, or IDENTICAL. For most languages based on the Latin alphabet, the PRIMARY level corresponds to alphabetic differences, the SECONDARY level corresponds to diacritic differences, and the TERTIARY level corresponds to differences in character case (for most locales, the default STRENGTH is TERTIARY). The QUATERNARY level takes punctuation and other special characters into account while the IDENTICAL or fifth level distinguishes between strings that are equivalent on the first four levels using character code points. Specification of a particular STRENGTH includes all levels up to and including the one specified by the strength value. For example, a TERTIARY strength setting includes levels 1 through 3 which, for languages with Latin-based alphabets, includes alphabetic differences, differences in accents, and differences in case.

The CASE_FIRST option controls whether lowercase characters are collated before uppercase characters or vice versa. The COLLATION option allows selection of different collation types, such as TRADITIONAL Spanish collation versus modern Spanish collation, or a PHONEBOOK style collation for the German language. The NUMERIC_COLLATION option allows integers, expressed as text in a character string, to be ordered numerically. Normally, numbers that are aligned toward their most significant digits (left-aligned for languages that are written from left to right) within a character string will not sort numerically. Instead, the numbers will be grouped according to the first digit encountered so that, for example, 1 and 10 appear before 2 and 20. Normally, to obtain a numeric ordering of character strings, the numbers must be aligned toward their least significant digits (right-aligned for left-to-right languages). Using the NUMERIC_COLLATION option allows a numeric ordering regardless of the alignment. The Unicode Consortium's technical report on the Unicode Collation Algorithm and the ICU documentation provide more details on the various option settings and their meanings.

Linguistic collation not only provides a sequence that is culturally correct and intuitive but, unlike binary and transcoded binary collation, provides a sequence that is largely independent of the underlying encoding or platform on which the collation is being performed. That is, while the ordering of character strings has some dependence upon the code point values of the characters at the highest levels, the major ordering is established by the lower levels and this ordering is independent of both the encoding of the strings and the SAS System on which the code is executing. The collating sequences obtained with the ICU are, therefore, consistent across systems.

EXAMPLES

A few examples of linguistic collation, compared to other collating techniques, may help to illustrate its benefits. The first example (Table 6) compares the results of binary, TRANTAB, and linguistic collation of a short list of English words. The second example (Table 7) does the same for a short list of French words. The third example shows how (Table 8) numeric collation differs from traditional collation of numbers within character strings.

Table 6 contains three columns, each demonstrating a different type of collation. The first column shows the sequence of words that is obtained using a binary collating sequence on words that are represented in an ASCII-based encoding. The second column shows the same list of words in a sequence obtained with a TRANTAB collation using the LOWFIRST translation table defined earlier in this paper. This translation table was designed to alternate the ordering of lowercase and uppercase letter such that ‘a’ < ‘A’ < ‘b’ < ‘B’ < … < ‘z’ < ‘Z’. The third column shows the results of linguistic collation of the same list of words.

In the first column, there is an obvious lack of alphabetic ordering due to the separate grouping of words beginning with uppercase and lowercase letters. In this column, the word “Zeus” incorrectly appears before “aardvark”. This ordering is due to the code points assigned to the characters within the ASCII-based encoding.

In the second column, we have tried to correct this problem by using a translation table that assigns the lowercase and uppercase character variants weights that are very similar. Using this table, by specifying SORTSEQ=LOWFIRST, we can cause “aardvark” to appear before “Zeus”. However, this TRANTAB collation causes the string “azimuth” to appear before “Aaron” because the lowercase character ‘a’ is assigned a weight that is less than the weight of the uppercase character ‘A’. Regardless of the assignment of weights in the translation table, it is not possible to achieve a true alphabetic ordering that takes the character case into account. This problem is inherent in the use of a single-level weighting algorithm like TRANTAB collation.

In the third column, we see that a proper alphabetic ordering is achieved, in accordance with the *en_US* locale, using a multi-level linguistic collation algorithm. This particular collating sequence specifies that, after the alphabetic order is established, words beginning with lowercase letters appear before words beginning with uppercase letters.

Table 6: Linguistic Collation Example 1 (English)

Binary Collation of ASCII-based encoding	LOWFIRST TRANTAB Collation	Linguistic Collation LOCALE=en_US
Aaron	aardvark	aardvark
Aztec	azimuth	Aaron
Zeus	Aaron	azimuth
aardvark	Aztec	Aztec
azimuth	zebra	zebra
zebra	Zeus	Zeus

Table 7 shows a similar example of words from the French language. In the first column, not only does binary collation cause a separation of words into separate groups based on the case of the first character, but it also causes words beginning with accented characters appear in another group. Similar to the English language binary collation example, a word beginning with a capital 'Z' incorrectly appears before a word beginning with a lowercase 'a'. In addition, in the first column, a word beginning with a lowercase 'z' incorrectly appears before a word beginning with a lowercase accented 'a'.

To correct the shortcomings of binary collation, evident in the first column, we can construct a translation table that orders both the letters of the French alphabet, along with the accented variants, so that the lowercase and uppercase characters alternate and accented character variants are assigned weights close to the unaccented characters (for example, 'a' < 'A' < 'à' < 'À' < 'â' < 'Â' < …). The drawback of this approach is immediately evident from the examination of words that do not appear in alphabetic order. Again, similar to the English language example, the single-level weighting approach used in TRANTAB collation is insufficient to properly order the list of words.

The third column shows the same list of words sorted linguistically, according to the rules of the French language.

Table 7: Linguistic Collation Example 2 (French)

Binary Collation of ASCII-based encoding	FRENCH TRANTAB Collation	Linguistic Collation LOCALE=fr_FR
Adrien	agent	Adélaïde
Adélaïde	agréable	Adrien
Afrique	Adélaïde	Afrique
Alsace	Adrien	âgé
Arctique	Afrique	agent
Atlantique	Alsace	agréable
Australie	Arctique	Alsace
Vénus	Atlantique	Arctique
Zacharie	Australie	Atlantique
Zoé	âgé	Australie
agent	échange	échange
agréable	écho	écho
végétal	végétal	végétal
zoologiste	Vénus	Vénus
zèbra	zèbre	Zacharie
âgé	zoologiste	zèbre
échange	Zacharie	Zoé
écho	Zoé	zoologiste

In addition to the ICU's ability to linguistically collate according to many locale settings, it also offers the ability to properly collate integers specified within a character string. Users are often confused by the differences in collating sequences between numeric variable values and the same values expressed as left-aligned numbers within a character string. Numeric variables collate algebraically, with smaller values appearing before larger ones. The collating sequence for numbers within a character string is usually dictated both by the code point (or weight) of the numeric characters and by the order of their appearance within a string, which is from the most-significant digit to the least significant digit. Ordering numbers from the most significant digit to the least significant digit does not produce an algebraic ordering. Table 8 demonstrates the effect of character collation of numeric strings in the left column and the numeric collation of the same strings in the right column. The sequence in the second column was achieved using PROC SORT with SORTSEQ=LINGUISTIC(NUMERIC_COLLATION=ON).

Table 8: Linguistic Collation Example 3 (Numeric Collation)

Standard Collation	Numeric Collation
1	1
10	2
100	9
2	10
20	20
200	90
9	100
90	200
900	900

ADDITIONAL DETAILS

Methods for linguistic collation were developed in recognition of the non-intuitive and culturally incorrect results produce by binary collation and various weighted collation schemes. Linguistic collation techniques were first developed for use with single-byte encodings. Attempts at encoding unification have resulted in the Unicode Standard and various Unicode multi-byte encodings. The UCA was developed in recognition of the need to provide a standard for linguistic collation of data expressed in these encodings. The ICU is an open source project that contains an implementation of the UCA.

SAS 9.2 has incorporated version 3.4 of the ICU for its linguistic collation capabilities and, by transcoding from the SAS session encoding to the UTF-16 encoding required by the ICU interface, provides linguistic collation through PROC SORT for any encoding supported by SAS. Although the ICU contains a highly efficient implementation of the UCA, the additional complexity of transcoding to UTF-16 and then forming collatable format sort key values from the character strings requires additional computational time, memory, and possibly storage space, when compared to binary collation. Specification of linguistic collation causes SAS sort (SORTPGM=SAS) to be used; there is no support for linguistic collation with host sorts.

Like TRANTAB and transcoded binary collation, the results of linguistic collation must be processed as though the data are grouped but not sorted. The NOTSORTED and NOBYSORTED options should be used for general BY processing, to disable the observation sequence check. Currently, neither the DATA step nor any procedure, apart from SORT, is capable of performing a sequence check for data sorted using a linguistic collation.

Linguistic collation introduces a new value into the collating sequence field of the sort indicator. This new value consists of the acronym UCA that is then both preceded and followed by an underscore character to indicate that the value is reserved for use by the SAS System. This value, *_UCA_*, is not a complete description of the linguistic collating sequence. Such a description includes both the specified locale and collating sequence options. For SAS 9.2, we made the decision to not expand the descriptor information to include the collating sequence description. For this reason, although the SORT procedure will indicate that a data set has been linguistically sorted (by setting the collating sequence to *_UCA_*,) it will not subsequently avoid a redundant sort request for the same collating sequence. Although the procedure can determine that the data set has been sorted using a linguistic collating sequence, it cannot determine whether the sequence is the same as a subsequently requested sequence because of the lack of a full collating sequence description.

FUTURE DIRECTIONS

With SAS 9.2, we have provided two new character collation capabilities, transcoded binary and linguistic collation, in addition to the existing classic binary collation and TRANTAB collation. These two new capabilities are accessible through the SORT procedure's SORTSEQ option. They provide for enhanced compatibility between systems and for intuitive, culturally correct sorting of character data. We believe SAS users may find linguistic collation especially useful for final presentation of data in tables and reports.

We encourage user feedback not only on these new capabilities, but also on possible future enhancements. We consider this work a first phase in the inclusion of advanced collation capabilities in the SAS System. If these new capabilities are well received and SAS users express sufficient interest in expanding collation capabilities, then we will consider concentrating development effort in this area within PROC SORT as well as within the rest of the Base SAS System and other products that build upon this foundation.

Future efforts will likely begin by ensuring that the DATA step, PROC SQL, and other parts of the SAS System simply recognize that a transcoded binary collating sequence matches the SAS System collating sequence, allowing the avoidance of redundant sorting and enabling standard optimizations. We will also examine supporting the generation of transcoded binary and linguistic collating sequences by those parts of the SAS System that currently support the SAS System SORTSEQ option, but are not yet capable of generating these collating sequences. We may extend the sort indicator descriptor information in the data set to include a definitive description of the linguistic collating sequence, enabling SORT and other procedures to fully recognize the established linguistic collating sequence. We will increase support for linguistic collation options within PROC SORT and we will examine increasing the efficiency and the performance of sorting using the two new forms of collation.

Numerous future possibilities lie beyond rounding out the ability of the SAS System to recognize and generate alternative collating sequences. These include extending collation support both downward, into the SAS System foundation, and upward into other SAS products. One possibility is enabling the use of linguistic collation when performing simple operations, such as string comparisons within the DATA step and PROC SQL. Another possibility is collating sequence sensitive BY processing in which both the sequence check and BY group boundary detection operations are affected. There are possibilities to affect a wide range of processing within the SAS System, such as WHERE clause evaluation, data set indexing, DATA step merging, and various SQL operations such as the ORDER BY, JOIN, UNION, and INTERSECT. Such changes likely cannot be considered in isolation but would have to be considered system-wide and would represent a substantial effort to implement. Customer feedback regarding these possibilities may play a large role in instigating such an effort.

While we evaluate future directions regarding collation within SAS, we will likely continue to use the ICU for linguistic collation and occasionally incorporate new versions of the ICU. As the ICU is growing in popularity and use, continued use of the ICU within SAS will enhance the compatibility of SAS with other data processing systems. We may also take advantage of the further capabilities of the ICU, such as the evaluation of regular expressions. Regular expression evaluation is, of course, already supported within SAS, but ICU offers support for regular expression evaluation of character data encoded in Unicode. The possibilities seem endless – your feedback will help us establish our direction and focus our future efforts.

ACKNOWLEDGEMENTS

We wish to thank the employees here at SAS who have taken time to review and contribute to this paper. We also thank those employees who have contributed to the effort to implement the two new types of collation within the SAS System. The feedback and participation of everyone has been indispensable in the effort to enhance the SAS System and to create this document describing those enhancements. Finally, we want to recognize and thank the creators, developers, and maintainers of the ICU for the work they have done to make the ICU an open, viable, and portable solution for the manipulation of Unicode and, specifically, providing a wonderful implementation of the Unicode Collation Algorithm for linguistic collation.

REFERENCES

Beatrous, Stephen. 2003. “Multilingual Computing with the 9.1 SAS Unicode Server.” *SAS Presents, Twenty-Eighth Annual SAS Users Group International Conference*, Seattle, WA.

Beers, Biff, and Manfred Kiefer. 1995. “Customizing the SORT Procedure for National Language Sorting.” *Observations: The Technical Journal for SAS Software Users* 4(2): 33-36.

Davis, Mark and Ken Whistler. 2006. “Unicode Technical Standard #10: Unicode Collation Algorithm.” <http://www.unicode.org/reports/tr10/> (April 7, 2007).

Garneau, Denis. 1990. *Keys to Sort and Search for Culturally Expected Results* (IBM Document Number GG24-3516). Roanoke, TX: International Business Machines Corporation.

International Business Machines Corporation. 2006. “ICU User Guide: International Components for Unicode Version 3.4.” <ftp://ftp.software.ibm.com/software/globalization/icu/3.4/icu-3.4-userguide.zip> <http://www.icu-project.org/download/3.4.html> (April 7, 2007).

Sköllermo, A., and Helen Wolfson. 1996. “Using National Characters with the SAS® System.” *Observations: The Technical Journal for SAS Software Users* 5(2): 21-25.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at scott.mebust@sas.com and michael.bridgers@sas.com.