THE ORDERING OF UNIVERSAL CHARACTER STRINGS

By René Haentjens

ABSTRACT

In the countries of the world, people have developed various methods to order words and names based on their cultures. Many challenges and problems are associated with developing ways for computers to emulate human ordering methods. An efficient computer method for obtaining a quality ordering has been devised as an extension to the single-step compare. It solves many but not all of the problems. A universal code now exists to store words and names written in many languages and scripts, but there is no universal way to order words and names. Hence, formal specification methods are needed for computer users to describe culture-specific ordering rules. This area is still open to research. Meanwhile, international standardization committees endeavor to formulate sensible proposals for multicultural contexts.

INTRODUCTION

Today, when we access information stored in computers, we often ask the computer to present us lists of items arranged in an order that is meaningful to us and easy to use. In the future, will the computer render obsolete the lists of words and names ordered for human reference? Will the computer look up all information in our place? Will we no longer need the skills to find our way around in dictionaries, telephone directories, and the like? These things are not impossible, but we ourselves might not live to see them happen.

If ordering for human consumption is to stay around for a while, then the next question that we might ask is whether or not it would be possible to harmonize the ways in which lists are ordered around the world. Most people are aware that alphabetic order may differ from one country to another. The same is true for scripts that are not based on an alphabet: although the Chinese Han characters are used to write Japanese and Korean, lists with Han characters are not in the same order in the People's Republic of China, Japan, Korea, and Taiwan, Republic of China.

Can we change to a universal ordering system or at least make ordering the same where the same script is being used? If the order of words were the same, life would surely be easier for the traveler! Unfortunately (if the reader permits that expression), the way in which we work with ordered lists is a cultural aspect and is related to the languages that we use. A proposal to change ordering habits is a bit like proposing a spelling reform.

Everyone is in favor of simplification as long as it applies to other groups of people, but we see no reason to change things for ourselves. In fact, looking back to the roots of our own culture, we find many good reasons why things are as they are today, so a change is seldom perceived as an improvement.

The conclusion is, for the time being, that we may as well use the computer to help us organize lists and to take into account that the task of ordering lists is not universally the same.

This paper explores the issues involved with ordering and the ways the computer can deal with them. It describes how people order words and names, and consequently, how they expect words and names to be ordered if a computer does the ordering. It presents examples of ordering in various cultures. This paper concentrates on the ordering of words and names; it does not include a discussion of numerical ordering.

WORDS, NAMES, AND CHARACTER STRINGS

Computers store words and names as character strings. The symbols that we use for writing are mapped to bit patterns in computers, and these patterns are chained together. For pragmatic reasons, the bit patterns do not correspond to graphic symbols in a simple one-to-one fashion. Attributes such as the font in which the symbol is presented and the size of the symbol are usually stored in separate areas, and the bit pattern for the specific character that represents the symbol remains the same. Also, several characters or bit patterns can sometimes be represented by the same graphic symbol. For example, the characters LATIN CAPITAL LETTER A and GREEK CAPITAL LETTER ALPHA can be rendered with the same graphic symbol A. Finally, the chaining of characters to strings may not completely agree with the visual arrangement of corresponding graphic symbols.

In other words, there are differences between how people order words and names and how computers order the corresponding character strings. People combine knowledge about words and names (for example, how to read and pronounce them) with visual aspects of the written or printed words and names. Computers must work with the bit patterns.

With regard to character coding, the International Standard ISO/IEC 10646-1:1993, Universal Multiple-Octet Coded Character Set, and the de facto standard, Unicode version 1.1, are considered state of the art. These two coding methods can conveniently be considered as identical, and the same abbreviation, UCS, refers to both of them. With UCS coding, words and names can be stored in many of the scripts of the world, and Chinese Han characters can be chained together with Latin, Greek, Cyrillic, Hebrew, and Arabic letters and many more.

Before discussing the complexities of UCS coding, this paper

explores some important aspects of ordering of character strings in the next section.

LEXICAL ORDERING

With lexical ordering, the computer takes into account only the kinds of characters that appear in the strings and the arrangement of these characters. Apart from the ordering algorithm and the associated data, the computer uses no other knowledge that it might have about the words in the character strings. For example, it does not use an electronic dictionary or rules about natural language syntax, phonetics, and semantics. The idea is to see how computers can work with reasonably efficient techniques, while staying close to how people work. Meaning-based ordering and searching with the computer is an interesting subject in itself, but is too broad a scope for this paper.

When people order words or names or when they are looking for them in an ordered list, they often use (unconsciously sometimes) the meanings of these words or some other knowledge about the words or names. For example, when looking for the name McMillan in a telephone directory, they might try to find it between MacLeod and MacNeville, knowing that Mc is the same as Mac. They might even look between Melbourne and Murphy, ignoring the Mc of McMillan altogether. If the computer has only a character string that represents the letters of the name McMillan, then it lacks the knowledge to look up the name any other way. Lexical ordering cannot incorporate expanding or ignoring prefixes and abbreviations; there is no lexical rule to determine what part of the character string might be a prefix or an abbreviation.

As another example, in Japanese many Han characters (called kanji by the Japanese) are pronounced in a different way depending on the context. Japanese dictionaries for general use are ordered by pronunciation; therefore, if the computer has only the kanji character in the character string, it cannot order or look up in the same way as people do in Japan. The character for rice, for example, is pronounced mai in a form such as gai mai (imported rice), but as bei in a form such as bei koku (America). The difference is due to the historical background of the character or when, in its specific context, it was borrowed from the Chinese. When kanji are used in proper names, such as names of persons and geographical names, there may be no context information, and human intervention might be needed to know the correct pronunciation.

In these cases, since the computer must mimic how people order and is limited to lexical techniques, more than codes for the letters or for the kanji must be stored in the character strings. For example, the computer might have a character string that contains a kanji character plus its pronunciation represented with kana characters. Or the computer might have strings such as

(Mc)Millan with the convention that the parentheses indicate parts to be ignored for ordering and searching.

Modern dictionaries and telephone directories use lexical techniques as much as possible, which is better in a multicultural environment. It is much easier to understand and apply lexical rules for searching than to acquire intuitive knowledge of an unfamiliar culture.

Words, Not Individual Letters

It is important to understand that people order words and names, not just the individual letters and symbols. Consequently, good quality lexical ordering that comes close to how people work cannot be achieved by looking at all the characters in a string only once, from the first one through the last one. This concept can best be illustrated with alphabetic scripts, and some English examples are given below.

When one looks for SOS in a modern English dictionary, one expects to see it between sort and soul. Now, to find SOS between sort and soul, one must ignore that SOS is in uppercase letters and sort and soul are in lowercase. This type of lookup is achievable by looking at all the letters once.

Now consider the abbreviation CAT, meaning clear air turbulence. CAT is listed between casual and catalyst. In this case, we cannot ignore the difference between CAT and cat. The dictionary lists both words, and some dictionaries consistently list lowercase words before uppercase words (or vice versa), so the order using lowercase first would be casual, cat, CAT, catalyst. It is not possible to devise an algorithm or method that would arrange these four words in the correct order by looking at all the letters once. To guarantee the correct order in all cases, a first step is needed in which uppercase is considered equal to lowercase; the two words cat and CAT must be placed in the correct order in a second step, in which uppercase and lowercase make a difference.

Dealing with uppercase and lowercase is not the only issue for alphabetic ordering. Many languages use letters with diacritical marks such as accents. Words and names may also contain spaces or special symbols, such as hyphens, apostrophes, and points. Examples are big bang, best-seller, rock 'n' roll, and P.S. When ordering is strictly alphabetic, as is the case in many dictionaries, then accents on letters, spacing, and special symbols are ignored in the first step, but they are taken into account to resolve a tie. For example, the correct order in French might be denier, dénier, dernier; or Nb, NB, N.B., Nd, n.d., N.D. in English.

TABLE-DRIVEN MULTILEVEL ORDERING

The heart of ordering methods is the comparison of two character strings. If we have an algorithm to determine whether one string should precede, follow, or be considered equal to a second string, then arranging a list of strings in the correct order is straightforward.

Single-step or One-level Compare

The single-step compare or one-level ordering algorithm is known by most readers:

Compare the first characters of the two strings; if equal, then compare the second characters; continue until a difference is found or until at least one string is exhausted. If a difference is found, then the character-collating sequence determines which string precedes the other. (Example: words precedes working because d precedes k.) If one of the two strings is exhausted, then the shorter string precedes. (Example: word precedes words.) If both strings are exhausted, then they are considered equal.

Multiple-step or Multilevel Compare

The state-of-the-art computer method for comparing character strings is a generalization of the single-step compare. If, after using the above algorithm with the first collating sequence, both strings are found to be equal, then in the second step the algorithm is repeated. Both strings are compared again, starting from their first characters, now using the second collating sequence. The second step may be followed by a third step and so on, one step for each collating sequence.

To be precise, the one collating sequence of all characters is replaced by a matrix of collating weights and collating weight sequences for each weight (W) column. Consider the following example:

	W1	W2	W3
LATIN CAPITAL LETTER D	<D>	<NONE>	<UC>
LATIN SMALL LETTER E	<E>	<NONE>	<LC>
LATIN SMALL LETTER E WITH ACUTE	<E>	<ACUTE>	<LC>
LATIN SMALL LETTER E WITH GRAVE	<E>	<GRAVE>	<LC>
LATIN CAPITAL LETTER E	<E>	<NONE>	<UC>
LATIN CAPITAL LETTER E WITH ACUTE	<E>	<ACUTE>	<UC>
LATIN CAPITAL LETTER E WITH GRAVE	<E>	<GRAVE>	<UC>
LATIN SMALL LETTER F	<F>	<NONE>	<LC>

The collating sequence for W1 is <A>, <B>, <C>, etc. This means that, with the example matrix, all variants of Latin letter E are equal in the first comparison step. The collating sequence for W2 is <NONE>, <ACUTE>, <GRAVE>, which means that in the second step,

the accents make a difference, but there is no distinction between lowercase and uppercase variants. That distinction is made in the third step: the collating sequence for W3 is <LC>, <UC>.

The weight matrix and the collating sequences can be placed in tables that are used by the ordering algorithm, hence the name table-driven multilevel ordering.

If this example matrix is extended in a similar way, then the multilevel algorithm would place the following words (most of which are real French words) in this correct order:
dénie, DÉNIE, denier, DENIER, dénier, DÉNIER, dènier, dernier.

The method that is described here is also used in POSIX (ISO/IEC 9945-2.2 Shell and Utilities, LC_COLLATE Definition).[1] Rolf Gavare was among the first to publish a paper on multiple-step comparisons.[2] Alain LaBonté was the first to describe it as explained in this paper, and he also implemented it as a Canadian Standard (CSA Z243.4.1-1992). LaBonté devised a complete and predictable ordering method that corresponds to very fine detail with the best examples of French and English dictionary ordering.[3]

Generate Comparison Key

With the multilevel method, it is also possible to have the algorithm generate a comparison key for a specific character string rather than always compare two strings. These comparison keys can be stored with the character strings; a one-level comparison of keys then gives the same result as a multilevel comparison of the original character strings. For example, and again extending the example matrix given above, the comparison key for dénie could be a convenient numerical representation of <D><E><N><I><E><nil><NONE><ACUTE><NONE><NONE><NONE><nil> <LC><LC><LC><LC><LC>.

The <nil> precedes all other weights. Its presence at the end of the comparison key subfields guarantees that shorter strings precede longer strings. Efficient compression techniques exist for such comparison keys.

VARIATIONS OF THE MULTILEVEL METHOD

The following section expands upon the multilevel method and gives examples of changes necessary to accommodate cultural differences in word order.

Special Symbols

With a small extension, the multilevel method can also handle

special characters such as the hyphen and the apostrophe, to mimic traditional human alphabetic ordering. Another weight column must be added to the matrix given above to distinguish letters from special characters:

LATIN SMALL LETTER E	<E>	<NONE>	<LC>	<LTR>
...				
HYPHEN-MINUS	IGNORE	IGNORE	IGNORE	<HPH>

The IGNORE indicates that the character is skipped in the comparison algorithm in the first three steps. A collating sequence for W4, in which <LTR> precedes all symbols for special characters such as <HPH>, guarantees that words and names without special characters precede the ones with exactly the same letters, but with special characters.

A four-level ordering such as the one suggested here is sufficient for a good quality, complete, and predictable alphabetic ordering with the Latin alphabet.

Additional Letters

For most languages written in Latin characters, the correct order of words would be senior, señorita, sentimental, separable. To achieve this order, W1 would be ..., <M>, <N>, <O>, ..., and the matrix would include LATIN SMALL LETTER N WITH TILDE, where W1 is <N>, W2 is <TILDE>, and W3 is <LC>.

In Spanish, the N WITH TILDE is considered a letter to be ordered between N and O and the correct order is senior, sentimental, señorita, separable. To achieve this type of ordering, W1 would be ..., <M>, <N>, <NTILDE>, <O>, ..., and the matrix would add LATIN SMALL LETTER N WITH TILDE, where W1 is <NTILDE>, W2 is <NONE>, and W3 is <LC>.

Ligatures

The multilevel method can also handle ligatures by allowing each matrix element to be a sequence of weights, rather than one weight. For Æ in French and Swedish, the matrix would include LATIN SMALL LIGATURE AE, where W1 is <A><E>, W2 is <LG><LG>, and W3 is <LC><LC>. In these languages, LIGATURE AE is equivalent to two letters when ordering words. In Norwegian, the Æ is a letter on its own. W1 is ..., <Y>, <Z>, <AE>, <OSTROKE>, <ARING>. For the matrix element, LATIN SMALL LIGATURE AE, W1 is <AE>, W2 is <NONE>, and W3 is <LC>.

Logograms

Some special symbols, sometimes called logograms, can be seen as short notations for words: & + %. A culture-specific ordering may replace such symbols by the corresponding words. If the language is English, for example, then Research & Development can be ordered as Research and Development. As long as a fixed rule exists for replacing symbols by equivalent words, the extension that was introduced for Æ can be applied in a similar way to obtain the desired ordering. On the other hand, if the replacement word depends on the language used in the rest of the string, then lexical ordering cannot do the job properly without more information coded in the character strings.

Fine Tuning for the Accents

The table-driven multilevel method, as explained so far, would place French words in this order: cote, coté, côte, côté, maçon, mâcon. In a traditional, correct ordering, they should be in the following order: cote, côte, coté, côté, mâcon, maçon. (In general, accents at the end of a French word are more important for understanding than other accents.)

To obtain the desired ordering, another extension of the multiple-step method is needed: for the second step, the one that discriminates between quasi-homographs (words that differ only in their diacritical marks), the comparison algorithm should start from the end of the strings rather than from the beginning. For the other Western languages that use the Latin alphabet, this reverse processing for the accents is not needed. On the other hand, it does not hinder either, so the French method is acceptable as well.

French is not the only language with such quasi-homographs. In new-Greek, with the modern monotoniko spelling, all multisyllabic words have one accent that indicates the stressed syllable. New-Greek has many quasi-homographs, including the following examples, which use a simple transcription of Greek letters to Latin letters: árguros, argurós, diakonía, diakoniá, métro, metró, pára, pará. The French method of reverse processing produces acceptable results for new-Greek as well.

Fine Tuning for the Special Symbols

With the tables extended as explained in the section Special Symbols, the multiple-step algorithm would order words as follows: unionized, union-ized, un-ionized. For the exceptional cases such as this one, in which two words are identical except for the placement of a special symbol, the order unionized, un-ionized, union-ized may seem more appropriate. Usually, the hyphen is perceived as a word break, not on the first level, but on a subsequent level, and with word breaks, shorter words always come first.

To obtain the latter ordering, one could use the same technique as for the diacritical marks: have the algorithm start from the end of the strings for the level that deals with the special symbols. POSIX has a small extension to the multilevel method that gives similar results while still moving forward. This extension adds the position of the symbol to its table weight during comparison.

Special Symbols in Combination with Uppercase and Lowercase Characters

This section does not introduce a new extension but reconsiders the extension for the special symbols. This method adds a fourth weight column:

```
LATIN SMALL LETTER E            <E>     <NONE>  <LC>    <LTR>
...
HYPHEN-MINUS                    IGNORE  IGNORE  IGNORE  <HPH>
```

With W3 for uppercase and lowercase and W4 for the special characters, the distinctions between uppercase and lowercase are considered more important than the presence or absence of spacing and special symbols. In many cultures, this is indeed the case with proper names of people. The following order is desired with names that differ in use of uppercase or lowercase letters: deGroot, de Groot, Degroot, De groot, DeGroot, De Groot.

For some geographical names, it could be argued that special symbols are more significant than the difference between lowercase and uppercase. For example, the desired order is Sanssouci, SANSSOUCI, Sans Souci, SANS SOUCI, Sans-Souci, SANS-SOUCI. (Sanssouci is a castle near Potsdam in Germany; Sans Souci is a city in South Carolina, U.S.A., and a suburb of Sydney, Australia; and Sans-Souci is a historical place on Haiti.) To obtain this order, W3 and W4 must be switched.

SOME PROBLEMS WITH THE MULTILEVEL METHOD

To obtain the correct order, changes are sometimes necessary to the multilevel method. This section discusses cases in which it is less easy to adapt the table-driven multilevel method.

Digraphs and Collating Elements

CH and LL have special placement in the Spanish alphabet. Spanish is not unique in this respect; combinations of letters also have special placement in the Albanian, Hungarian, Vietnamese, and Welsh alphabets. The Welsh ordering alphabet, for example, is A B C CH D DD E F FF G NG H I J L LL M N O P PH R RH S T TH U W Y, and the following list of words is correctly ordered in Welsh: acw, achos, adwy, addas, agwedd, angau, almon, allan, anfynych,

anffodus, antur, anthem.

Before the multilevel method can be applied, it is necessary to replace the multiple-character combinations by pseudo-characters. In POSIX LC_COLLATE, such a mechanism is foreseen. One can declare combinations such as LATIN SMALL LETTER C followed by LATIN SMALL LETTER H to be collating elements and give them a name that can be used in the matrix.

At first it would seem that this solves the problem. One complication, however, is that the two letters together do not always represent the special alphabet letter. In Welsh, for example, the N and G are separate letters in the Welsh words melyngoch, dangos, gwyngalchu, and mwynglawdd. The word melyngoch then is among words starting with melyn, not after the words with melyg. More information must be coded in the character strings that represent Welsh words to define a correct lexical ordering.

A similar problem exists with Danish. In most Danish words, aa is semantically and phonetically equivalent to å. Danes expect aa and å to be ordered together, after Z, Æ, and Ø. But in words of foreign origin, aa is just A + A.

The reader with a knowledge of programming complexity will probably also see that the collating-element extension makes the table-driven multilevel method less straightforward to implement. If there are only a few collating-element extensions, then simple workarounds might help, but what if there are thousands of them? (Improbable? Wait to form your opinion until you read the section Added Complexity with UCS Coding.)

Sequences, However Long

Other ordering requirements are difficult to accommodate with the matrix method. For example, the British standard on ordering, BS 1749:1985, requires that (in the first step) spaces, dashes, hyphens, and diagonal slashes and sequences of them be treated as a single space (which is significant), except at the beginning of an entry, where they should be ignored. Making a space significant for ordering is easy, but the collating-element extension unfortunately does not allow recursive definitions, so it cannot incorporate the sequences of spaces, etc.

Other Problems

Context dependencies illustrate another problem for collating-element extensions. The Japanese language has several DUP characters, the weights for which depend on the context. For first-level ordering, a DUP character in a Japanese word or name can be considered equivalent to the character that precedes it. Hence, if X represents a Japanese character, then X followed by DUP is equivalent to X followed by X in the first comparison

step. Tie breaking is done in a subsequent step: X DUP then precedes X X. If collating-element definitions are used, definitions for all possible combinations are required.

ADDED COMPLEXITY WITH UCS CODING

The concepts discussed in this section have existed in other coded character sets for some time. For example, ISO 6937 has combining characters, and ISO/IEC 8859-7 contains Latin and Greek letters. With UCS, script mixing and combining characters will for the first time be implemented on a wide scale, not only geographically speaking, but also when counting the number and the importance of the computer platforms on which UCS coding will exist.

UCS has room for some 65,000 characters in the currently defined basic multilingual plane. The first and most obvious implication is that the tables for the multilevel method will be huge with UCS.

Mixing Scripts

With UCS coding, many scripts can be used in a single character string. Although all languages with a non-Latin script have some tradition of incorporating words and names written in Latin letters, there are not many rules about ordering in such a context. For example, where should the Latin-letter abbreviation SOS be placed in a Greek, Russian, or Chinese dictionary? The problem with computers, of course, is that everything must be specified, including the unusual situations.

Ordering Han Characters

As previously stated, UCS also codes Han characters. The people who use them for writing characterize a Han character with attributes such as its main radical, the number of pen strokes to draw the character, and its Chinese or Japanese pronunciation. (A radical is a constituent part of the character.)

For example, the Han characters with Japanese pronunciation tera (temple), kata (type), and shiro (capital) all have the same main radical. Tera has six strokes; kata and shiro have nine. The Chinese pronunciations are ji, kei, and jyou.

A popular ordering is by radical first, then by number of keystrokes, and finally by Chinese pronunciation. With this ordering, tera comes first (it has only six strokes), and kata precedes shiro because of the Chinese pronunciation. If this were the one and only way of ordering Han characters, then the computer would not need to know about the radicals, pen strokes, etc. Each Han character has a different code (bit pattern), so a single (but long) collation order for the corresponding codes would be sufficient.

Significantly, each dictionary of Han characters has developed its own tradition for ordering. Depending on the application, audience, school, or political considerations, the preferred ordering may be different. For example, the onyomi ordering is also in popular use in Japan. It is by Chinese pronunciation first, then by stroke count. With onyomi ordering, kata comes first, then tera, and shiro is the last one.

Han characters are always ordered character by character, so the multilevel method that applies multiple weights in multiple steps involving complete strings is not required. Han characters require multiple weights with a specific combination that is dynamically selected for a single-step ordering.

It is not evident how this dynamic single step can be combined with the standard multiple-step method, which is needed for UCS strings containing Han characters mixed with other ones.

Combining Characters

UCS also contains the concept of combining character. In the example matrices given above, it was assumed that letters with accents such as LATIN SMALL LETTER E WITH ACUTE are coded as one character. UCS indeed has such one-character codings, but it allows a letter with an accent to be coded as two characters as well. The sequence of two characters LATIN SMALL LETTER E followed by COMBINING ACUTE is also valid in UCS.

UCS does not state that LATIN SMALL LETTER E WITH ACUTE is the same as LATIN SMALL LETTER E followed by COMBINING ACUTE; it leaves it to applications to consider them equivalent or not. Needless to say, many application developers will want users to have the possibility of considering both forms equivalent, at least for ordering.

The notion of equivalence becomes quite intricate with two or more diacritical marks. See the paper on Unicode in this issue for a discussion on transformations between equivalent spellings.[4]

For our extended matrix method, not only thousands, but an unlimited number of collating elements would have to be defined. UCS allows any number of combining characters to follow a noncombining character.

Logical Order and Coding Order

With UCS coding, the order of the characters in a string is the logical or reading order, not the order in which the symbols have to be printed or displayed. Hence, UCS encoded text is difficult to display and print, but relatively easy to be processed, e.g., for ordering.

In Thai, unfortunately, this approach was not implemented totally. The vowels and diacritics that appear above or under a consonant are coded in logical (reading) order, but Thai has five so-called pre-positioned vowels that are written and coded before the consonant after which they have to be pronounced. This corresponds to current computing practices in Thailand and was incorporated in UCS coding as a sort of backward compatibility. For example, the word written and encoded as E + CH + N (ignoring vowel shortener and tone mark) is pronounced chên and ordered accordingly. To allow correct ordering for UCS-encoded Thai, some preprocessing is necessary to arrange the Thai vowels in the correct position for the ordering step.

Formatting Characters

Many coded character sets contain characters that do not correspond to some written symbol but have some control function, often for output formatting. For ordering, these formatting characters can usually be handled in the same ways as special characters.

The characters ZERO WIDTH JOINER and ZERO WIDTH NON-JOINER are among the UCS formatting characters. Their primary purpose is to influence the display of characters of a cursive script such as Arabic. Before UCS was finalized, some people suggested that ZERO WIDTH NON-JOINER might be used to indicate the absence of special digraphs such as in the Welsh word melyngoch. It has also been proposed that ZERO WIDTH JOINER might be used to create new letters such as unusual or newly invented ligatures. Today, this is no longer considered a valid use of these formatting characters.

TOWARD A FORMAL DESCRIPTION OF ORDERING

Excellence for computer applications means not only that the application incorporate a different way of ordering for each culture, but also that it give freedom to its users to define variations and use different approaches to ordering. This is important for some cultures. Not so long ago, the use of multiple letter fonts was considered specialized work for professional printers; today every word processor must allow it. Flexibility with regard to ordering may also become commonplace a few years from now. But how can such flexibility be provided in a computer-digestible yet user-friendly way?

Many documents describe ordering in an informal way. National standards on ordering are seldom formal definitions. They contain directives such as each unbroken sequence of digits, disregarding commas, spaces, and stops, is considered as one character; or multiple hyphens collate as one; or ij is ordered as i + j; or ß = ss. Such directives are vague for computers. They are imprecise: Is the hyphen to be understood as the character

HYPHEN-MINUS only, or also as related, but distinct characters in UCS coding such as HYPHEN, MINUS SIGN, and others? They are also incomplete: ij is ordered, but not IJ, Ij, and iJ. They use graphic symbols, where the computer wants to know things about characters: Does ß stand for LATIN SMALL LETTER SHARP S or for GREEK SMALL LETTER BETA?

On the other hand, the descriptions for POSIX LC_COLLATE are quite formal. They are more or less bound to a specific implementation, in this case the table-driven multilevel method described above. A more simple formal description is sometimes sufficient. For example, if the data to be ordered is filtered and contains only uppercase Latin letters, then the POSIX syntax may seem an overkill. In other cases, the LC_COLLATE formalism lacks expressive power, as we have seen.

Is it possible to design a formal specification method that falls between the descriptive texts in country standards and the almost algorithmic parameters such as POSIX LOCALEs?

ISO/IEC 10646-1:1993 may provide a first step to build formal definitions. It is the most comprehensive repertoire of characters to date and a strict superset of many earlier repertoires and coded character sets. Moreover, it establishes a unique and authorative naming for characters. This paper uses character names such as LATIN CAPITAL LETTER E WITH ACUTE. ISO has decided that the 10646 names will be used in all future character set standards and standard updates. In a certain sense, ISO/IEC 10646-1:1993 is a character reference manual, and formal definitions about ordering can be built upon its content.

PREPROCESSING

Preprocessing a character string, transforming it into text elements or linguistic units in a logical sequence, is a second concept that deserves elaboration. It was mentioned in relation to Thai with its pre-positioned vowels in a preceding section.

Breaking down a string into the smallest units to be processed by an ordering algorithm and arranging these units in the desired processing order is a powerful mechanism. It could also be used to detect collating elements, to replace Japanese DUP characters, or to transform character sequences that contain combining characters. This mechanism would then allow the table-driven multilevel method to be used to its full extent on preprocessed strings.

Preprocessing might change the character string: units are rearranged, characters are replaced by other ones, etc. It is possible that two originally different character strings could be preprocessed to an identical intermediate form. If ordering is to be complete and predictable, preprocessing must generate additional tags that are taken into account by the multilevel

method.

Consequently, the output of the preprocessing phase might be more than pieces of character strings. The lines used in the matrices for the multilevel method have (names of) characters as labels. If preprocessing were designed to generate an output that is easier to consume by the multilevel method, the labels could be anything that seems suitable.

The problem, again, is how to allow for the specification of preprocessing in a formal yet user-friendly way. Transformations based on regular expressions and finite state machines are a possible path. These techniques allow an efficient implementation. P. J. Plauger has published material about using them for ordering with the C language.[5,6]

CONCLUSIONS

The evolution of computer systems is progressing toward a better quality interaction with people. An aspect of that interaction is the ordering of words and names. Efficient methods exist today for obtaining a quality ordering. Although some software uses these methods, many applications perform computer-friendly ordering rather than human-friendly ordering. There is no technical limitation to improve on that aspect; for example, a multilevel algorithm with user-specified tables can replace a single-step bit-code ordering.

For some cultures and in multicultural environments, not all ordering problems are solved. Research is needed, as well as formal rules to allow users to specify ordering preferences.

Some useful ordering techniques are in place. The table-driven multilevel method is an important one. Preprocessing can solve some problems, but a convenient formalism is needed to specify it. UCS coding provides many new challenges; but at the same time it offers a new fixed point, from which it may be possible to derive user-friendly formal definitions.

APPENDIX:
INTERNATIONAL STANDARDIZATION EFFORTS

Many countries have developed a standard on ordering. These standards are not listed in this section.

ISO/IEC JTC1/SC22/WG15 (Programming Languages) is the committee and work group that is discussing the POSIX work (ISO 9945).

ISO/IEC JTC1/SC22/WG20 (Internationalization) is working on a Technical Report that will provide a framework for internationalization. The work group is also preparing documents on the registry of cultural elements, specification methods for defining string comparison, and a default-tailorable ordering for

10646.

CEN (European Standardization Committee) BTS7 (Technical Bureau on IT)/TC304 (Character Set Technology) has a project on European character string ordering rules. The scope is to establish procedures for the registration of national and regional ordering rules and to prepare multilingual character ordering rules for European scripts (Latin, Greek, and Cyrillic).

ISO TC37/SC2/WG2 is currently working on multilingual ordering for terminological and lexicographical purposes. ISO TC46/SC9 has similar work but for bibliographical purposes. The approach is application oriented, whereas the other ISO and CEN efforts mentioned above are computer-oriented approaches.

To allow for some level of synchronization of these efforts and to avoid overlaps, liaisons have been established between all these committees.

ACKNOWLEDGMENTS

Alain LaBonté of the Gouvernement du Quebec, Direction Générale des Technologies de l'Information, has been the inspiration for many things written in this paper. He has on many occasions encouraged me to continue with my explorations of ordering. I also owe thanks to Johan van Wingen, independent consultant in Leiden, the Netherlands, who has gathered and made available much background information on coded character sets and ordering practices. A special word of thanks goes to Kevin P. Donnelly, to Denis Garneau, and to P. J. Plauger for reviewing this paper and for providing many useful comments and suggestions.

Of the many colleagues in Digital who have helped me, I want to especially mention Masahiro Morozumi of International Systems Engineering in Japan, with whom I could exchange many mails about ordering in Japanese and about Digital's implementation of XPG4. I also want to mention Tim Greenwood of International Systems Engineering in the U.S., who has done a lot of coordination work for this issue of the Digital Technical Journal, and for my contribution to it in particular. And I know that I'm doing an injustice by not naming the many other colleagues who contributed.

REFERENCES

1. X/Open CAE Specification, System Interface Definitions, Issue 4, X/Open Doc N C204 (London: X/Open Company Limited, 1992).

2. R. Gavare, "Alphabetical Ordering in a Lexicological Perspective," Data Linguistica 18 (Almquist & Wiksell, 1988).

3. A. LaBonté, Regles du classement alphabetique en langue francaise et procedure informatisee pour le tri (Ministere

des Communications du Quebec, 1988).

4. J. Bettels and F. Bishop, "Unicode: A Universal Character Code," Digital Technical Journal, vol. 5, no. 3 (Summer 1993) 21-31.

5. P. Plauger, "Translating Multibyte Characters," The Journal of C Language Translation (June 1991).

6. P. Plauger, The Standard C Library (Englewood Cliffs, NJ: Prentice Hall, 1992).

GENERAL REFERENCES

G. Adams, "Introduction to Unicode," Proceedings of the Fourth Unicode Implementors Workshop (Mountain View, CA: Unicode Consortium, 1992).

B. Comrie, ed., The World's Major Languages (Oxford: Oxford University Press, 1990).

F. Coulmas, The Writing Systems of the World (Oxford: Basil Blackwell, 1989).

J. DeFrancis, The Chinese Language (Honolulu: University of Hawaii Press, 1984).

D. Garneau, Keys to Sort and Search for Culturally Expected Results (Ontario: IBM National Language Technical Center, 1990).

S. Jones et al., Developing International User Information (Burlington, MA: Digital Press, 1992).

K. Katzner, The Languages of the World (London: Routledge, 1989).

C. Kennelly, Digital Guide to Developing International Software (Burlington, MA: Digital Press, 1991).

E. Kohl, The Art of Arranging Files, ISO Bulletin (December 1986).

A. LaBonté, "Multiscript Ordering for Unicode," Proceedings of the Fourth Unicode Implementors Workshop (Mountain View, CA: Unicode Consortium, 1992).

A. Nakanishi, Writing Systems of the World (Rutland, VT and Tokyo: Charles E. Tuttle, Co., 1980).

G. Sampson, Writing Systems (Hutchinson, 1985).

STRI TS73, Nordic Cultural Requirements on Information Technology (Reykjavik: Idntaeknistofnum Islands, 1992).

U. Warotamasikkhadit and D. Londe, Computerized Alphabetization of Thai, Technical Memo TM-BA-1000/000/01 (Santa Monica, CA: System Development Corp., 1969).

Unicode 1.0.1, Report from the Unicode Consortium (Mountain View, CA: Unicode, Inc., 1992).

TRADEMARKS

Digital is a trademark of Digital Equipment Corporation.

Unicode is a trademark of Unicode Inc.

BIOGRAPHY

René Haentjens René Haentjens is a software consultant working for both Digital Consulting Belgium and Corporate Standards and Consortia. He was the Belgian local engineering manager for two years. Today, René is a member of the Belgian, the European (CEN), and the ISO committees on character sets and internationalization. He contributed significantly to the ISO/IEC 10646-1:1993 standard. He has a civil engineering degree (chemistry) from the University of Ghent and has contributed to publications on compiler portability, on software engineering, and on developing international software and user information.