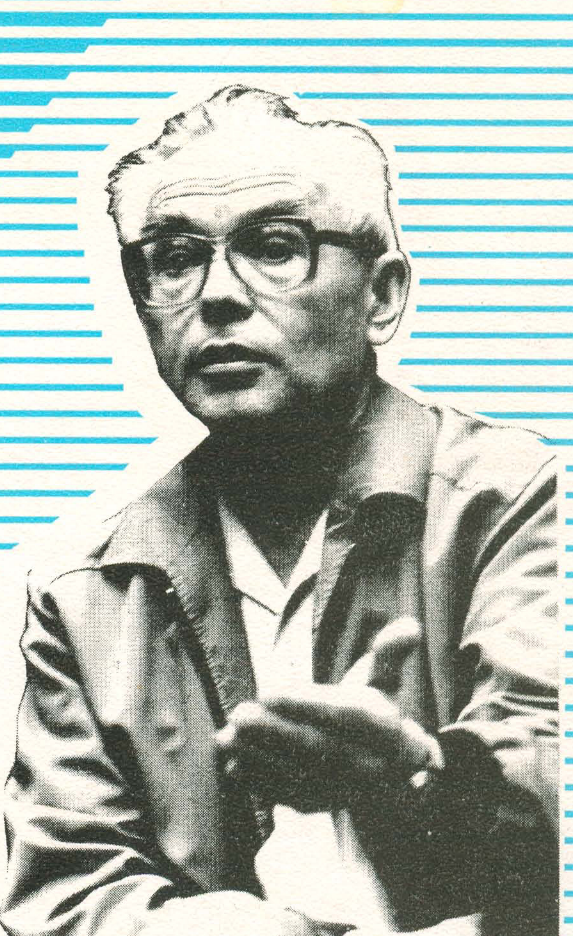
Preliminary Proceedings PSI '96

Andrei Ershov Second International Memorial Conference

Perspectives of System Informatics

Novosibirsk
Akademgorodok
June 25-28,
1996



Fast and Efficient Multilingual Alphanumeric Ordering Systems

Nikos Gaitanis, Maria Katzouraki and Manolis Gergatsoulis

Institute of Informatics & Telecommunications,
NCSR 'Demokritos',
153 10 A. Paraskevi, Greece,
e_mail: {nikos,maria,manolis}@iit.nrcps.ariadne-t.gr

Abstract. Ordering words and names of human languages is a very interesting and complex problem that depends on the applied ordering processes of various cultures and the ways the computers can deal with them. A formal mathematical method for the specification of multilingual alphanumeric ordering systems has been developed[2] incorporating the traditional multistep comparisons of word components. Specifications defined according to this method consist of the data part that depends on the language characters and the behavior part that can be determined by the specified system. In this paper we present a new fast and efficient ordering process of Natural Language words and names using only one word comparison step always from the left to the right that covers all the traditional ordering methods of European Languages. This ordering process can be applied as a European Multilingual Ordering standard instead of the traditional multistep process since it has a general use and it is very simple minimizing the software cost the computation time and the specification complexity of conforming Alphanumeric ordering systems.

1 Introduction

Traditional ordering processes of world languages depend on how people order words and names, and can be derived from Dictionaries, Lexica, Encyclopaedias and other official documents. In general there is a long list of different types of ordering parameters as well as practical methods which in most cases are described using ordering rules and directives. This kind of informal description cannot be generalized since there exist no general ordering rules and the directives cannot be transformed directly into computer program specifications. Also it cannot be easily proved that all the implementations that conform with a particular informal description produce a "complete ordering of words" (there are not two different words with the same order). Rolf Gavare [3] was among the first to publish a paper on Alphanumeric Ordering using multistep comparisons of world components and Alain LaBonte [6, 5] was the first to describe a systematic multistep ordering method and he also implemented it as a Canadian Alphanumeric Ordering Standard [1]. René Haentjens [4] describes a multilevel or multistep ordering method of character strings and Nikos Gaitanis[2] presents a formal specification method of multilingual alphanumeric ordering systems that can be used for every European language of Latin, Greek or Cyrillic collections as well as for mixing script ordering processes. In general a multistep ordering method is based on a decomposition of characters into their components with the corresponding significance in the ordering of characters that can be described in posix notation [8] or as a formal parameter set in [2]. Drawing inspiration from the above as well as the ISO bibliographic, Canadian, Swedish, German, Danish and Austrian standards, a European prestandard (ENV)[7] for the European Multilingual Ordering, has been written by CEN/TC304/WG1 that adopts the multistep ordering procedure. Although all the above references agree with the multistep ordering process, there is an argument now about the minimum number of comparison steps that have to be accepted as a European standard even though some restrictions will be imposed to the traditional ordering processes in some European Languages such as Greek and French. In this paper, we solve this serious problem looking the ordering process from another point of view applying a more sophisticated approach. The result is a new general ordering method covering all the traditional ordering processes of European Languages. This method applies only one comparison step, starting from the left and comparing one by one all the corresponding characters of two words through the last non_blank character. For this we use the new concept of *character distance* which identifies all the component differences between two different characters and we introduce systematic computation of successive comparison results using the formal description technique of Finite State Machines.

2 Definitions

Definitions relevant to the multilingual alphanumeric ordering can be distinguished into the following groups:

2.1 Characters

An *Alphabetic Character* ia a member of a set of elements used for the organization, control or representation of data. An *Arithmetic Character* is a Graphic symbol used for the representation of numerical values. An *Alpharithmetic Character (Cr)* is an Arithmetic or an Alphabetic Character. A *Language Alpharithmetic Repertoire (Re)* is the set of Arithmetic and Alphabetic Characters used for writing words and numbers of a given Natural Language.

2.2 Character Components

Definition 1. *Character components:* Are the elements of Alphanumeric Characters which contribute to the ordering of characters or the corresponding character strings.

An alpharithmetic character (Cr) consists of:

1) A *Basic Alphanumeric Symbol (As)* that can be one of the *Basic Numerical Symbols* $\{0,1,2,3,4,5,6,7,8,9\}$ or one of the *Basic Letter Symbols* with small or Capital size $\{a, A, b, B, c, C, ...\}$. The absence of an As is indicated by a special member of the set As called an *Empty Letter Space (ELS)*.

2) A *NonBreak Symbol (Bs)* that can appear at the right of an As symbol. The nonbreak symbols belong to an international set of special symbols (Space, Underscore Minus, Comma,....). A Bs is used for the connection of words into the same term. There also exists a blank special symbol in the set Bs which indicates the absence of NonBreak Symbols and it is called *Empty NonBreak Space (EBS)*.

3) Above and below the Letter symbol appear various types of *Diacritic marks*. Diacritic marks are used with the basic letter symbols in order to define syntactic, phonetic or semantic attributes.

Definition 2. *Character Component Types (T) :* They are disjoint blocks of character components with an ordering relation. Character components are distinguished into character types so that at most one element of each type appears in a character.

There are four different component types for the Latin characters:

1) **Tn**: Is the type of letter names ((Blank) < (a=A) < (b=B) < (c=C) < (d=D) < (e=E) < (f=F) < ......

2) **Td**: Is the type of Diacritic marks (None < acute < grave < circumflex < caron < ring < diaeresis < double_acute < tilde < dot < stroke < cedolla < ogonek < macron < line_below < preceded_by_apostrophe).

3) **Tz**: Is the type of letter size (small < Capital).

4) **Tb**: Is the type of NonBreak Symbols (Blank < NonBreakSpace < Underscore < Macron < Soft Hyphen < Minus < Comma < Semicolon < Colon < Exclamation Mark < .....).

For the Greek politonic characters there are six different component types:

1) **Tn**: letter names (Blank < alpha < beta < gamma < delta <....).

2) **Ta**: Aspiration diacritic marks (none < psili < dasia).

3) **Tc**: Accent diacritic marks (none < varia < oxia < tonos < perispomeni).

4) **Ts**: Special diacritic marks (none < ypogegrammeni < prosgegrammeni < dialytika).

5) **Tz**: Letter size (small < Capital).

6) **Tb**: Is the type of NonBreak Symbols (Blank < NonBreakSpace < Underscore < Macron < Soft Hyphen < Minus < Comma < Semicolon <...).

Definition 3. *Type ordering parameter:* is a function $v : NatNum \rightarrow T$ which defines the significance ordering of character component types for a particular Language. Then, j-Type $\equiv T_j \equiv T_{v(j)}$ where $j = 1, ..., k$ and $k = number\ of\ \ character\ \ types.$

Taking into account the traditional ordering processes of words with Latin and Greek script we conclude that the following Type ordering parameters are commonly used for sorting European words:

a) Latin script: $v(1) = n$, $v(2) = d$, $v(3) = z$ and $v(4) = b$.

b) Greek script: $v(1) = n$, $v(2) = a$, $v(3) = c$, $v(4) = s$, $v(5) = z$ and $v(6) = b$.

In both cases, a difference in the name of characters of two character strings (words) is of highest significance and decides for the ordering of words no matter what other type differences exist.

Definition 4. *Character Projections*: $f_j : Re \rightarrow T_j$ are used to represent the components of a given character $C \in Re$.

Example 1. Let us consider as an example the Greek character C = GREEK SMALL LETTER IOTA WITH TONOS AND DIALYTIKA Then $f_1(C) = IOTA$, $f_2(C) = NONE$, $f_3(C) = TONOS$, $f_4(C) = DIALYTIKA$, $f_5(C) = SMALL$, $f_6(C) = BLANK$.

Definition 5. *Character representation* is a function $Hg : T_1 \times T_2 \times .... \times T_k \rightarrow Re$. The domain of Hg is a sequence of k character components (where k=4 at least for the Latin script and k=6 at least for the Greek script) with normal ordering from left to right, starting with the first significance component Type (1-Type).

Example 2. $C = Hg(IOTA, NONE, TONOS, DIALYTIKA, SMALL, BLANK)$.

Definition 6. Let C and Z be two alphanumeric characters. Then the *Character distance from C to Z* is defined as follows: $d(C, Z) = (y_1, y_2, y_3, y_4, y_5, y_6)$ is a k-tuple of three state variables y_j with values defined so that $y_j = 0$ if $f_j(C) = f_j(Z)$ or $y_j = 1$ if $f_j(C) > f_j(Z)$ and $y_j = 2$ if $f_j(C) < fj(Z)$.

Example 3. Let us consider as an example two Greek characters:
C = Hg(IOTA, NONE, TONOS, DIALYTIKA, SMALL, BLANK)
Z = Hg(ALPHA, PSILI, OXIA, NONE, CAPITAL, BLANK)
then : $d(C, Z) = (1, 2, 1, 1, 2, 0)$.

2.3 Words

Definition 7. A *Normal Word* $W0 = Qq(C_1, C_2, ...C_t)$ of length t is a sequence of characters with normal ordering from left to right where the first character C_1 at the left side is a NON_BLANK Character.

Definition 8. An *Inverse Word* $W1 = Qq(C_t, C_{t-1}, ..., C_1)$ of length t is obtained from the Normal Word W0 with inverse ordering of Characters. The first character C_t at the left side of W1, may be a BLANK or a NON_BLANK Character.

Definition 9. The *Word Projections* $F_{0j}(W0)$ or $F_{1j}(W1)$ are words of j-Type elements that can be obtained from the normal word W0 or from the inverse word W1 by substitution of word characters $C_1, C_2, ...$ by their j-Type components $f_j(C_1), f_j(C_2), ...$ so that $F_{0j}(W0) = Qq(f_j(C_1), f_j(C_2), ..., f_j(C_t))$ or $F_{1j}(W1) = Qq(f_j(C_t), f_j(C_{t-1}), ..., f_j(C_1))$.

Definition 10. *Position Ordering Parameter* x(j)=0,1 is a numerical value assigned to the j-Type word projections of a given word W0 and defines forward $x(j) = 0$ or backward $x(j) = 1$ position ordering of j-Type character components.

This parameter expresses the traditional ordering processes obtained from the Dictionaries, Lexica or other official documents for the particular Language. In general, we use the following position ordering parameters:

	Name	*Diacritic Marks*	*Size*	*NonBreak*
Greek Script :	$x(1) = 0$	$x(2) = 1, x(3) = 1, x(4) = 1$	$x(5) = 0$	$x(6) = 1$
Latin script :	$x(1) = 0$	$x(2) = 1$	$x(3) = 0$	$x(4) = 1$

2.4 Formal Word Representations

Definition 11. The *Formal Representation* $F(F_{x(1)1}(Wx(1)), F_{x(2)2}(Wx(2)), ...)$ of a given normal word $W0$, is a sequence of word projections defined according to the position ordering parameter $x(j)$, and starting from the left side with the first significance type (1-Type) according to the type ordering parameter $v(j)$ with $j = 1, 2, 3, ...$

Example 4. For the Greek word in example 3, $W0 = Qq(C_1, C_2)$ we have the following Formal Representation defined according to the given Type ordering parameter $v(j)$ and Position ordering parameter $x(j)$: $W0 = F((ALPHA, IOTA), (NONE, PSILI), (NONE, OXIA), (NONE, NONE), (SMALL, SMALL), (BLANK, BLANK))$.

Definition 12. *Word Formal Difference* $DF(W0, U0)$ of normal words $W0 = Qq(C_1, C_2, ...)$ and $U0 = Qq(Z_1, Z_2....)$ is the first different pair of word projections $F_{x(j)j}(Wx(j)), F_{x(j)j}(Ux(j))$ in the corresponding Formal Representations $F(W0)$ and $F(U0)$ of words starting from the first significance component Type.

Example 5. Let us consider the word $W0$ of example 4 and the word $U0 = Hg(ALPHA, PSILI, OXIA, YPOGEGRAMMENI, SMALL, BLANK)$.

Then, we have :
$U0 = F((ALPHA, BLANK), (NONE, PSILI), (NONE, OXIA), (NONE, YPOGEGRAMMENI), (SMALL, SMALL), (BLANK, BLANK))$ and $W0 = F((ALPHA, IOTA), (NONE, PSILI), (NONE, OXIA), (NONE, NONE), (SMALL, SMALL), (BLANK, BLANK))$ and the word formal difference is $DF(W0, U0) = [(ALPHA, IOTA), (ALPHA, BLANK)]$.

Definition 13. The *Word Projection Difference* $Df(F_{x(j)j}(Wx(j)), F_{x(j)j}(Ux(j))$ of a given pair of word projections of j-Type is the first different pair of character projections $f_j(C_i)$, and $f_j(Z_i)$ detected at the i position following the defined position ordering $x(j)$ for the particular component type.

Example 6. According to the above we have the following Word projection difference Df((ALPHA,IOTA), (ALPHA, BLANK))= [IOTA, BLANK].

3 Ordering Rules

1. Ordering of two words $W0 = Qq(C_1, C_2, ..., C_t)$ and $U0 = Qq(Z_1, Z_2, ..., Z_t)$ of a given Language is defined according to their word Formal difference:
   $DF(W0, U0) = [F_{x(j)j}(Wx(j)), F_{x(j)j}(Ux(j))]$.
2. Ordering of two word projections of the same j-Type $F_{x(j)j}(Wx(j))$, and $F_{x(j)j}(Ux(j))$ is defined according to their word projection difference:
   $Df(F_{x(j)j}(Wx(j)), F_{x(j)j}(Ux(j))) = [f_j(C_i), f_j(Z_i)]$.
3. Ordering of two character components $f_j(C_i)$, and $f_j(Z_i)$ is defined according to the ordering relation in the corresponding character type.

Example 7. For the words $W0$ and $U0$ in example 5 we have: $IOTA > BLANK \rightarrow (ALPHA, IOTA) > (ALPHA, BLANK) \rightarrow W0 > U0$.

4 Number of ordering steps

From the above it follows that ordering of words for a particular Language is a sequential process performed in k steps where each step compares the corresponding character components of two words starting from the left to the right or from the right to the left.

In this paper, another simple and efficient ordering process is described based on character by character comparison always going from left to right of the words. This process is applied with the successive character distance states as inputs and can be implemented by a sequential system specified by a Finite State Machine.

Definition 14. A *Finite State Machine* $M(I, S, O, f, g)$ consists of: 1) an input set I, 2) a set of internal states S, 3) a set of output states O, 4) the next state function $f : IXS \rightarrow S$ and 5) the output function $g : S \rightarrow O$.

The operation of a Finite State Machine is defined by the next state function $f : IXS \rightarrow S$ represented by the well known State Table where an input state collumn cross a current internal state row at the next state cross point.

The ordering process of French words is performed according to: (where x is a dont_care state)

Tb	0	x	x	x	x	x	x	1	2	
Tz	0	x	x	x	x	1	2	0	0	
Td	0	x	x	1	2	0	0	0	0	
Tn	0	1	2	0	0	0	0	0	0	*output*
a	a	$b1$	$b2$	$c1$	$c2$	$d1$	$d2$	$e1$	$e2$	0
$b1$	$b1$	$b1$	$b1$	$b1$	$b1$	$b1$	$b1$	$b1$	$b1$	1
$b2$	$b2$	$b2$	$b2$	$b2$	$b2$	$b2$	$b2$	$b2$	$b2$	2
$c1$	$c1$	$b1$	$b2$	$c1$	$c2$	$c1$	$c1$	$c1$	$c1$	1
$c2$	$c2$	$b1$	$b2$	$c1$	$c2$	$c2$	$c2$	$c2$	$c2$	2
$d1$	$d1$	$b1$	$b2$	$c1$	$c2$	$d1$	$d1$	$d1$	$d1$	1
$d2$	$d2$	$b1$	$b2$	$c1$	$c2$	$d2$	$d2$	$d2$	$d2$	2
$e1$	$e1$	$b1$	$b2$	$c1$	$c2$	$d1$	$d2$	$e1$	$e2$	1
$e2$	$e2$	$b1$	$b2$	$c1$	$c2$	$d1$	$d2$	$e1$	$e2$	2

based on the Canadian Alphanumeric ordering standard [1] with Type ordering parameter $v(1) = n$, $v(2) = d$, $v(3) = z$ and $v(4) = b$ and position ordering parameter $x(1) = 0$, $x(2) = 1$, $x(3) = 0$ and $x(4) = 1$ defined in [2].

Example 8. Let us consider two different French words $R =$ *côte* and $G =$ *coté* where $d(R1, G1) = (0, 0, 0, 0)$, $d(R2, C2) = (0, 1, 0, 0)$, $d(R3, G3) = (0, 0, 0, 0)$, $d(R4, G4) = (0, 2, 0, 0)$. Then:
1) with current state (a) and input $d(R1, G1)$ we go to the next state $f(a, (0, 0, 0, 0)) = (a)$ and output state $g(a) = 0$ indicating $R = G$.
2) with current state (a) and input $d(R2, G2)$ we go to the next state $f(a, (0, 1, 0, 0)) = (c1)$ and output state $g(c1) = 1$ indicating $R > G$.
3) with current state (c1) and input d(R3,G3) we go to the next state $f(c1, (0, 0, 0, 0)) = (c1)$ and output state $g(c1) = 1$ indicating $R > G$.
4) with current state (c1) and input d(R4,G4) we go to the next state $f(c1, (0, 2, 0, 0)) = (c2)$ and output state $g(c2) = 2$ indicating $R < G$.

Also the ordering process of Greek words is performed according to:

Tb	0	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*	1	2	
Tz	0	*x*	*x*	*x*	*x*	*x*	*x*	*x*	*x*	1	2	0	0	
Ts	0	*x*	*x*	*x*	*x*	*x*	*x*	1	2	0	0	0	0	
Tc	0	*x*	*x*	*x*	*x*	1	2	0	0	0	0	0	0	
Ta	0	*x*	*x*	1	2	0	0	0	0	0	0	0	0	
Tn	0	1	2	0	0	0	0	0	0	0	0	0	0	*output*
a	*a*	*b*1	*b*2	*c*1	*c*2	*d*1	*d*2	*e*1	*e*2	*h*1	*h*2	*k*1	*k*2	0
*b*1	*b*1	*b*1	*b*1	*b*1	*b*1	*b*1	*b*1	*b*1	*b*1	*b*1	*b*1	*b*1	*b*1	1
*b*2	*b*2	*b*2	*b*2	*b*2	*b*2	*b*2	*b*2	*b*2	*b*2	*b*2	*b*2	*b*2	*b*2	2
*c*1	*c*1	*b*1	*b*2	*c*1	*c*2	*c*1	*c*1	*c*1	*c*1	*c*1	*c*1	*c*1	*c*1	1
*c*2	*c*2	*b*1	*b*2	*c*1	*c*2	*c*2	*c*2	*c*2	*c*2	*c*2	*c*2	*c*2	*c*2	2
*d*1	*d*1	*b*1	*b*2	*c*1	*c*2	*d*1	*d*2	*d*1	*d*1	*d*1	*d*1	*d*1	*d*1	1
*d*2	*d*2	*b*1	*b*2	*c*1	*c*2	*d*1	*d*2	*d*2	*d*2	*d*2	*d*2	*d*2	*d*2	2
*e*1	*e*1	*b*1	*b*2	*c*1	*c*2	*d*1	*d*2	*e*1	*e*2	*e*1	*e*1	*e*1	*e*1	1
*e*2	*e*2	*b*1	*b*2	*c*1	*c*2	*d*1	*d*2	*e*1	*e*2	*e*2	*e*2	*e*2	*e*2	2
*h*1	*h*1	*b*1	*b*2	*c*1	*c*2	*d*1	*d*2	*e*1	*e*2	*h*1	*h*1	*h*1	*h*1	1
*h*2	*h*2	*b*1	*b*2	*c*1	*c*2	*d*1	*d*2	*e*1	*e*2	*h*2	*h*2	*h*2	*h*2	2
*k*1	*k*1	*b*1	*b*2	*c*1	*c*2	*d*1	*d*2	*e*1	*e*2	*h*1	*h*2	*k*1	*k*2	1
*k*2	*k*2	*b*1	*b*2	*c*1	*c*2	*d*1	*d*2	*e*1	*e*2	*h*1	*h*2	*k*1	*k*2	2

This ordering process takes into acount the following Type ordering parameters: $v(1) = n$, $v(2) = a$, $v(3) = c$, $v(4) = s$, $v(5) = z$, $v(6) = b$ and position ordering parameter $x(1) = 0$, $x(2) = 1$, $x(3) = 1$, $x(4) = 1$, $x(5) = 0$, $x(6) = 1$.

Example 9. Let us consider the Greek words W0 = Qq(C1,C2) and U0=Qq(Z1) of example 5 ordered according to the multistep traditional process. where $d(C1, Z1) = (0, 0, 0, 2, 0, 0)$ and $d(C2, Z2) = (1, 0, 0, 0, 0, 0)$. Then:
1) with current state (a) and input d(C1,Z1) we have $f(a, (0, 0, 0, 2, 0, 0)) = (e2)$ and output state g(e2) = 2 indicating $U0 > W0$.
2) with current state $(e2)$ and input $d(C2, Z2)$ we have $f(e2, (1, 0, 0, 0, 0, 0)) = (b1)$ and output state $g(b1) = 1$ indicating the correct $W0 > U0$.

From the above state tables it follows that for a given state mi (with $m = a, b, c, d, ...$ and $i = 1, 2$) that corresponds to a j-Type character component difference $d(C_r, Z_r) = (0, 0, ., 0, y_j, ., 0)$ then:
A) Any subsequent difference $d(C_{r+1}, Z_{r+1}) = (0, ., 0, y_p, ., 0)$ at the p-Type character component with: 1) $p < j$ changes the internal state as defined by the character distance $d(C_{r+1}, Z_{r+1})$, 2) $p > j$ then the state mi remains the same.
B) Any subsequent difference $d(C_{r+1}, Z_{r+1}) = (0, ., 0, z_j, ., 0)$ of the same j-Type character component with $0 < y_j$, z_j and $y_j > z_j$ or $y_j < z_j$: 1) Changes the internal state mi into another ml with $i < l$ if $y_j < z_j$ and $i > l$ if $y_j > z_j$ when the position ordering parameter $x(j) = 1$, and 2) The internal state mi remains the same when the position ordering parameter $x(j) = 0$.

5 Conclusions

A new fast and efficient single step ordering process which can be applied for every national or international ordering standard that has been formally specified according to the general specification method established in [2], is presented in this paper. This single step implementation of various cultural dependent alphanumeric ordering standards of European Languages, satisfies all the necessary requirements to become a standard ordering process as it does not impose any restrictions to the national standardization commettees to apply their traditional ordering processes and normalizes the required operations into the minimum number of comparisons of word characters.

References

1. *CSA Z243.4.1/ 1992 Canadian Alphanumeric Ordering Standard.* 1992.
2. N. Gaitanis and S. Kokkotos. Formal specification of multilingual alphanumeric ordering systems. *Computer Standards & Interfaces*, 17:535–552, 1995.
3. R. Gavare. Alphanumeric ordering in a lexicoligical perspective - a computational model for extracting alphabetization of dictionary and encyclopaedic material. *Studies in Computer-Aided Lexicology*, 63–102, 1988.
4. R. Haentjens. The ordering of universal character strings. *Dijital Technical J.*, 5(2):43–52, 1993.
5. A. LaBonte. Multiscript ordering for unicode. In *Proc. 4th Unicode Implementors Workshop*, 1992.
6. A. LaBonte. *Regles du Classement Alphabetique en Langue Francaise et Procedure Informatisee pour le Tri Ministere des Communications du Quebec.* 1988.
7. K. Simonsen. *European Multilingual ordering rules.* Working draft CEN/TC 304/WG1 N438, 1995-04-18.
8. K. Simonsen. *Nordic and Baltic POSIX Locales and Character Sets.* CEN/TC 304 N080.