VARIOUS REPRESENTATIONS OF TEXT PROPOSED FOR EUROTRA

Christian Boitet(+), Nelson Verastegui(++), Daniel Bachut(++)

(+)Groupe d'Etudes pour la Traduction Automatique
Université Scientifique et Médicale de Grenoble
BP 68 - 38402 Saint Martin d'Hères - France

(++)Institut de Formation et Conseil en Informatique
27, rue Turenne - 38000 Grenoble - France

ABSTRACT

We introduce several general notions concerning the texts and the particularities of text processing on a computer support, in relation to some problems which are specific to M(A)T. And we present the solution we have proposed for the duration of the EUROTRA project.

INTRODUCTION

The input/output modules are very important for a machine (aided) translation system (M(A)T), which must be integrated into some environment (translation office, technical data base, etc.).

From an external point of view, the support of a text is either paper with figures, formulas, tables and typographical conventions, or a magnetic support containing, in addition, formatting and page-setting commands for a special text processing system.

Within all modern M(A)T systems, including EUROTRA (now in the specification phase), a text is viewed, from an *internal* point of view, as a set of decorated nodes, organized according to a particular geometrical distribution (often a tree structure, as in ARIANE-78 (Boitet et al., 1982)).

Our objective in proposing some representations of texts for EUROTRA has been to define an internal structure recognized by the EUROTRA software systems, and carrying all information necessary for the translation model and for the restitution of the preceding information at output time.

TEXT PROCESSING IN GENERAL

Each text (whether or not on computer support) is considered from three points of view, i.e. :

The *Form* is everything related to the particular external aspect of a text on paper. E.g., the fact that it is written in one or several columns, single or double spaced, printed recto or recto/verso, following a special convention for the numbering of chapters and sections, etc.

The *Structure* is the logical division of the text into hierarchically related pieces such as volume, part, chapter, section, sub-section, paragraph, sub-paragraph, sentence, numbered or non-numbered lists, figures, tables, diagrams, etc. This depends on the kind of text : when processing plays, getting rid or their devision into acts and scenes is out of the question. When poetry is processed, the delimitation of each line cannot be left out.

The structure can be externally represented by using various *possible* forms. In the context of M(A)T, the advantages of taking into account the structure of the text are twofold :

- the text can be decomposed if only part of it is to be translated ;
- it is easy to retrieve a piece of text (e.g. when the translation of a long text has failed on one sentence).

The *Content* is the "text" considered as a sequence of "words" carrying some information. Words in different languages may appear, written with special characters, in upper/lower case, diacritics, punctuation marks, stress, etc.

These three notions are interrelated. The content of a text can, for example, refer to a page number, which belongs rather to its form. Often, the length of the original text is not maintained in the translation, and this, therefore, modifies the form.

In text processing systems, a coding (either visible or invisible to the user) enables to express the three above-mentioned characteristics of the text. We will call *formats* the codes related to the form, and *separators* the codes related to the structure. We distinguish four main features of the formattors (some examples can be found in (Furuta et al., 1982 ; Chamberlin et al., 1981 ; Goldfarb, 1981 ; IBM, 1981, 1983 ; Stallman, 1981 ; Thacker et al., 1979).

[1]This work has been carried out as part of a contract with the Commission of the European Communities (in the framework of the EUROTRA Research and Development programme) and the CNRS (Centre National de la Recherche Scientifique). The ideas and proposals in this paper are those of the authors and not necessarily shared or supported by the Commission, nor are they to be interpreted as part of the EUROTRA design. We are grateful to the Commission and the CNRS for agreement to publish this paper.

1. *delayed/immediate* : in the delayed case, there is no interaction with the author and any local modification of the document can only be carried out after a complete reformatting of the text. In the immediate case, the author can immediately see the effect of any modification on the formatting of the document.

2. *text only/figures and text* : systems able to process pictures and text are associated with "addressable dot printers" or with photocomposition machines.

3. *imperative/declarative* : in an imperative system, the user uses formatting commands written in a low-level language (".sp 2;" to skip two blanks,...). In a declarative system, a high-level language enables the "typing" of the different parts of the text, without bothering about the specific result obtained on a specific physical support.

4. *integrated/separated* : depending on the system, several objects can represent a text. When structure and content are "mixed" in each object, the coding is called integrated, otherwise it is called separated.

Let us take the following text as an example :

```
.sp 2
.us on
Avant-dernier exemple:
.us off
<<Où est-il! -- Je ne sais pas. -- Parti,
tout à fait?
-- Non... enfin je ne crois pas... -- Bon,
dit-il. Il a raison. >> (Ch. Rochefort)
```

In that case, the formattor is of delayed, text only, imperative, and integrated type. The form depends on the formats and on their parameters (.sp 2, .us on/off). The structure depends on the punctuation ("!", "...", "--"...), and on some formats.

In the context of M(A)T systems, some decisions must be taken, as to :

- how a text is "decomposed" at input time (into segments, units, words, separators, punctuation, etc.) ;

  To create this structure (and carry out the decomposition of the text) in a system with integrated coding, it suffices to introduce special codes (or to use existing codes, like end-of-text, formats...) to mark the text and to generate the object "structure" automatically from their interpretation.

  In order to do so, the system must know the list of separators as well as their hierarchical ordering ;

- how the formats for page-setting are handled. These formats are almost always linguistically relevant. For example, titles form a particular sublanguage. Hence, a "title" format may be used by the analyzer to use an appropriate subgrammar.

- how alphabetical transcriptions are carried out. No coding standards exist for all languages, although ISO codes and transcriptions (ISO, 1983) have been defined ;

- how the "plates" are handled. Figures, formulas, etc., may be completely left out, or replaced by special "words", or left in the text. This last method implies the use of some formal language for figure description, which must be handled by the linguistic processor.

WHAT COULD BE DONE IN EUROTRA ?

Our proposals are based on our experience with GETA's ARIANE-78 system (Boitet et al., 1982), but also on some others approaches (Morin, 1978 ; Bennett et al., 1984 ; Hawes, 1983 ; Hundt, 1982).

We have proposed that, all along the translation process, a given text is kept together with the attributes defining its three aspects : content, form and structure.

This solution seems more interesting, because all information related to the text is kept. Hence, it is possible to write linguistic processes in such a way that the output text will present the same *form* as the input text. No complex (and often not good enough) restitution program is necessary. Moreover, many codes (formats, separators...) have a linguistic relevance which the linguists might wish to put to profit.

The second idea is to choose a unique and unambiguous internal representation for each character : each symbol of each processed language (including the special symbols such as "/", "%" ...) should be represented by a unique internal code. This obviously has great advantages, for example the ease of transfer of linguistic applications.

One of the basic principles underlying this proposal is, therefore, *its adaptability to the environments*. We wish to work directly on real texts, without being obliged to put them in some form or other prior to process them into the system. Manual pre-editing will be reduced to a minimum.

We wish to access objects in a way which allows to indicate the text processing system used (for the definition of formats and separators), and the input/output device used for entering the text. The proposed solution calls for *three tables*, the content and use of which we will now describe.

These tables (not necessarily disjoint) correspond to the three levels of form, structure and content. The order in which they are described corresponds to the advised order of use.

The tables should be used to drive the so-called input/output module (or conversion module).

Transcription

The transcription table allows the conversion of a text entered on any device whatsoever, into an equivalent text (in the same language). This table, therefore, would depend on the input/output device used.

For reasons of generality and portability, the ISO code seems to be the best choice for the internal code.

Each alphabet would be identified in a unambiguous way by a corresponding escape sequence. In addition, we propose :

- to assign to each alphabet a language code ;
- to define two escape codes for the two possible modes of representing a character : 2 bytes and 1 byte.

We think it would be best to choose for each language a standard which respects its alphabetical order. At the level of the internal code, the transliteration problem does not exist as this code is supposed to contain all the symbols used.

However, we propose to use factorization of the alphabet code only for storage and to keep the 2 bytes code during the whole processing.

This conversion can easily be carried out with the use of an "equivalence" table called *transcription table*. In general, there will be one table for each input/output device and for each language.

The table would function as follows (at input time) : in the first column, recognition of the current symbol of the text, and transformation of this symbol into the corresponding element (in accordance with the storage mode, i.e. adding or not the language code), in the second column.

This table enables us to unify the writing conventions of the text and, in a more general way, would be used for all (input/output) communication between the system and a human partner.

In this table, we also indicate the alphabetical order of each language. Each language has its own characteristics ; in French, for example, dictionaries are sorted according to the letters of the alphabet, and then according to the diacritics. In order to take all these possibilities into account, we propose to add a series of columns to this transcription table : sorting would be carried out in several phases chosen in advance.

Let us assume that French text is entered on an English keyboard : the absence of diacritics oblige to define transcription rules.

The table of transcription would be as follows (the codes are fictitious) :

Human transcription	Internal code	Alphabetic order	Diacritic order
e	e	i	1
e$1	è	i	2
e$2	é	i	3
u$1	ù	j	2
...	...		

Formats

We attempt to define a means of specifying all the characteristics necessary for the recognition of formats on a wide range of formattors and text processing systems. But we may assume that, independently of the formattor chosen, there will be a codification standard for texts which limits the number of possibilities and simplifies entry.

In general, this stage will have three phases (the first phase is strictly computational, the next two are of a linguistic nature), each of which is the object of different information data, stored in the table of formats :

- recognition of the format : features of formats must be coded in some fields of the table ;
- initialization of associated decorations (properties and values), which will characterize it all along the linguistic processing. The linguist should envisage its definition and its use in a way which is coherent with the linguistic models. Freedom of choice of properties and values to be assigned to each format should be left to him.
- transformation of the recognized format in a string. The interest of this string lies in the fact that it can serve to mark different formatting orders which express the same action, in a way which is unique. Similar formats will, then, be unified by one single convention which is defined by the linguist. The model (grammars and dictionaries) would not depend on a particular formatting system. A change of formattor would, therefore, not be felt at the level of the linguistic data.

For the example given above, the table would be as follows :

Prefix	Search Zone		End of format			Param	Occurrence	
	C.Begin	C.End	Leng.	Stop chr	End line		type (format)	string
.sp	1	1	<133	;	YES	YES	PARAGRAPH	
.us on	1	1	<133	;	YES	NO	BEG UNDERLINED	underscore
.us off	1	1	<133	;	YES	NO	END UNDERLINED	
			...				...	

Structural separators

Once the text is in EUROTRA code and decomposed into formats and "non-formats", we identify its structure. To that end, we use a table of structural separators. A *separator* is a string of characters to be found either in the formats or in the other occurrences. It can correspond to a punctuation sign, a word-separator (not necessarily blank or space !), etc. For a format, it is proposed to use its characteristics, as given by the properties and values assigned in the previous table and not the string of characters which enabled its recognition.

In this table, the separators should have a hierarchical order. Therefore, both the *level* of a separator is defined and its place in the hierarchy, the highest possible level being 1. The formats not found in the table will be taken by default as separators of the lowest level.

For the example given in the first part, we can define the below table (the ß represents a blank or a space. The transcriptions are not taken into account).

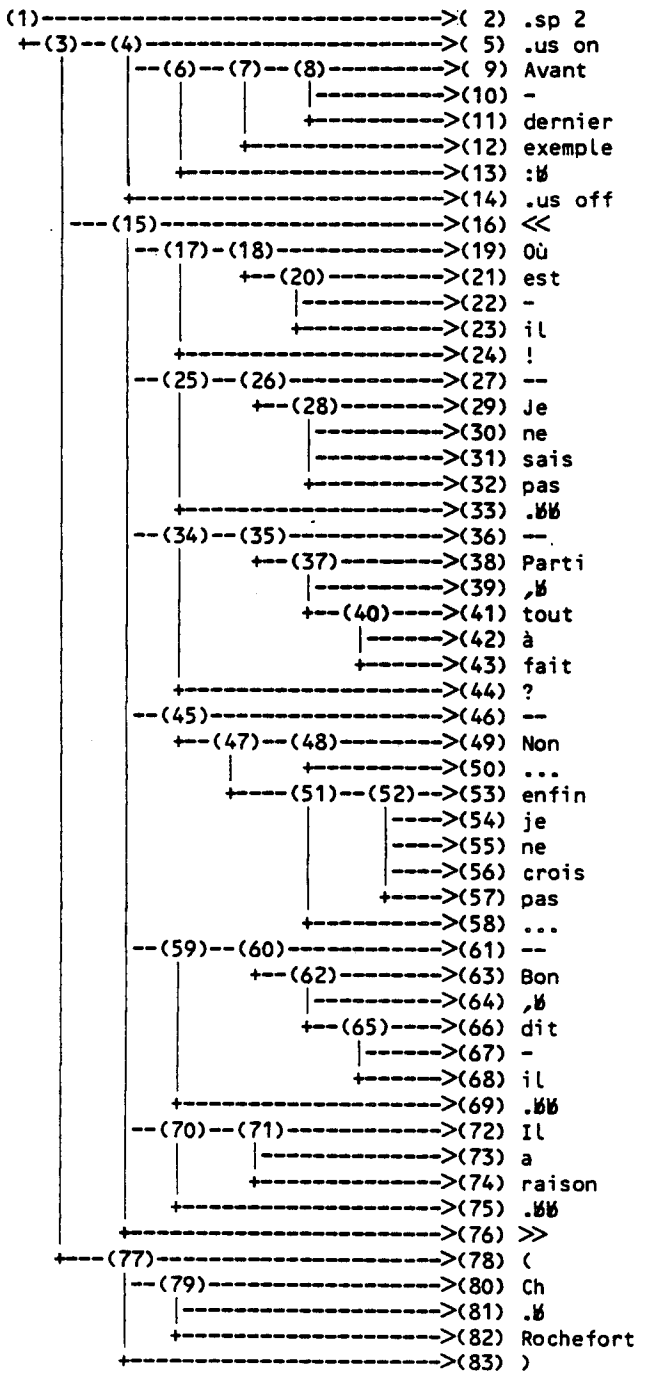
The fact that certain symbols are followed by one or two blanks in order to distinguish their level, could give the impression that this is the result of pre-editing. But this is not the case ! In this example, we have only use a text which follows precise and strict conventions in typography, as is the case for a great number of real texts. Our proposal can also apply to the processing of texts which have no precise conventions. It suffices to define the tables in an appropriate way.

Format separator		Level	Nesting (format)			OCCURRENCE	
yes	no		start	yes	no	DELETE	TYPE(CONTENT)
PARAGRAPH		1	NO			NO	
	!	2	NO			NO	EXCLAMATION
	?	2	NO			NO	QUESTION
	.ßß	2	NO			NO	SENTENCE
	:ß	3	NO			NO	COLON
	—	4	NO			NO	HYPHEN
	...	5	NO			NO	WORD
	,ß	5	NO			NO	WORD
	«	6	YES		»	NO	B_INVERTED COMMAS
	(	6	YES		)	NO	B_PARENTHESES
	»	6	NO			NO	E_INVERTED COMMAS
	)	6	NO			NO	E_PARENTHESES
BEG UNDERLI.		7	YES	END UNDERLI.		NO	
END UNDERLI.		7	NO			NO	
	ß	8	NO			YES	WORD
	-	9	NO			NO	HYPHEN
	.ß	9	NO			NO	FULL STOP
			...		...		

As for the formats, we propose to add to this table properties and values for the recognized separators. We should be able to define the properties and values to be assigned to the simple occurrences not found in the table and to indicate whether the separator, once it is recognized, should be kept or not (blanks, for example).

The next tree is the result of the application of the three tables given above to our example text. Each leaf carries the properties and values given by the tables. The property OCCURRENCE contains the character string indicated. The TYPE of the nodes 2, 5 and 14 is FORMAT. The type of all other leaves is CONTENT.

We have the choice between building up the tree considered, and building up a list of nodes each of which correspond to a leaf of the tree. Maybe the linguist should be able to choose by means of a parameter. In the build-up of a tree, it would be interesting to assign the properties and values of the highest priority separator found amongs its daughters to the internal nodes. Node 1 would thus have the value PARAGRAPH and node 17 the value EXCLAMATION.



CONCLUSION

The creation of the tables will be carried out mainly by a computer scientist, who is supposed to know the hardware, the internal code, the formatting and the structuration conventions of the texts... The linguists should, however, be consulted for the introduction of the conventions they have adopted (names of properties and values, of types of occurrences, of strings...). The information of a linguistic nature is exclusively meant for the unification of data having different sources. The introduction of purely linguistic knowledge is left to a next module in the translation process.

The result of the conversion could be submitted to human revision. This depends on the power of the mechanism using the tables, and on the content of the tables.

The problem of automatic recognition of formulas and plates in general has not been treated. Its solution depends on the text processing system which is chosen and its level of difficulty is highly variables.

The advantages of this solutions are :

- the independ nce with particular peripheral device and text processor ;
- the flexibility of the representation ;
- the general applicability : the EUROTRA machine can be used for processings other than translation.

REFERENCES

BENNETT W., SLOCUM J.
"METAL : The LRC Machine Translation System", Linguistic research center, Austin, Texas, USA, September 1984.

BOITET C., GUILLAUME P., QUEZEL-AMBRUNAZ M.
"Implementation and conversational environment of ARIANE-78. An integrated system for automated translation and human revision", Proceedings COLING-82, North-Holland, Linguistic Series n° 47, pp. 19-27, Prague, July 1982.

CHAMBERLIN D.D., KING J.C., SLUTZ D.R., TODD J.P., WADE B.W.
"JANUS : An interactive system for document composition",
Proceedings of the ACM SIGPLAN SIGOA symposium on text manipulation, Portland, Oregon, June 8-10, 1981, SIGPLAN Notices, V16, N6, pp. 68-73.

FURUTA R., SCOFIELD J., SHAW A.
"Document Formatting Systems : Survey, Concepts, and Issues", Computing Surveys, Vol. 14, n° 3, September 1982, pp. 417-472.

GOLDFARB C.F.
"A generalized approach to document markup", Proceedings of the ACM SIGPLAN SIGOA symposium on text manipulation, Portland, Oregon, June 8-10, 1981, SIGPLAN Notices, V16, N6, pp. 68-73.

HAWES R.
"LOGOS : the intelligent translation system", "Translating and the Computer" Conference, The Press Centre, London, UK, November 1983.

HUNDT M.
"Working with the WEIDNER machine-aided translation system", Department of translation, Mitel Corporation, Kanata, Ontario, Canada, 1982.

IBM
"Document Composition Facility : User's guide", SH20-9161-2, 411 p., September 1981.

IBM
"Office Information Architectures : Concepts", GC23-0765, 38 p., March 1983.

ISO
"International Register of Coded Character Sets to be used with Escape Sequences", Subcommittee ISO/TC 97/SC 2 : Character sets and coding, 326 p., 1983.

MORIN G.
"SISIF : système d'identification, de substitution et d'insertion de formes", Groupe TAUM, Université de Montréal, 1978.

STALLMAN R.M.,
"EMACS : The extensible, customizable self-documenting display editor", Proceedings of the ACM SIGPLAN SIGOA symposium on text manipulation, Portland, Oregon, June 8-10, 1981, SIGPLAN Notices, Vol. 16, N6, pp. 147-156.

TAUM
"TAUM-METEO, Description du Système", Groupe de recherches pour la Traduction Automatique, Université de Montréal, 47 p., Janvier 1978.

THACKER C.P., Mc CREIGHT E.M., LAMPSON B.W., SPROULL R.F., BOGGS D.R.
"Alto : A personal Computer", Technical Report CSL-79-11, Xerox Palo Alto Research Center, August 1979.