COLING 82, J. Horecký (ed.)
North-Holland Publishing Company
© Academia, 1982

IMPLEMENTATION AND CONVERSATIONAL ENVIRONMENT OF ARIANE 78.4, AN INTEGRATED SYSTEM FOR AUTOMATED TRANSLATION AND HUMAN REVISION

Ch. BOITET, P. GUILLAUME and M. QUEZEL-AMBRUNAZ

Université Scientifique et Médicale de Grenoble
GETA - BP. 53X
38041 Grenoble Cédex
FRANCE

INTRODUCTION

ARIANE-78.4 is a computer system designed to offer an adequate environment for constructing machine translation programs, for running them, and for (humanly) revising the rough translations produced by the computer. ARIANE-78 has been operational at GETA for more than 4 years now. This paper refers to version 4. It has been used for a number of applications (russian and japanese, english to french and malay, portuguese to english) and has constantly been amended to meet the needs of the users. Parts of this system have been presented before [2,3,7,8], but its whole has only been described in internal technical documents.

This paper tries to give such a global presentation. Given the space constraint, it centers on the conversational environment under which users may manipulate the data bases of texts (with their partial transforms, their translations and their revision) and of linguistic programs (e.g. grammars and dictionaries), test and debug their programs, run complete translations and revise them.

Part I gives the necessary introduction to the system and its components. Each component (e.g. morphological analysis, structural analysis, etc.) is written is one of the four languages specialized for linguistic programming (LSLP) supported by the system (ATEF, ROBRA, TRANSF and SYGMOR). Those languages have been presented elsewhere.

In part II, we explain the global queries and actions of ARIANE-78. Some global variables (such as the source language code) may be set. Various informations may be extracted from the data base, such as which texts have partial results for a given step of the translation process, or which are the lexical units for a given source (or target) language code, and in which grammars or dictionaries they appear. Some global actions, such as archiving part of a linguistic application, may also be performed.

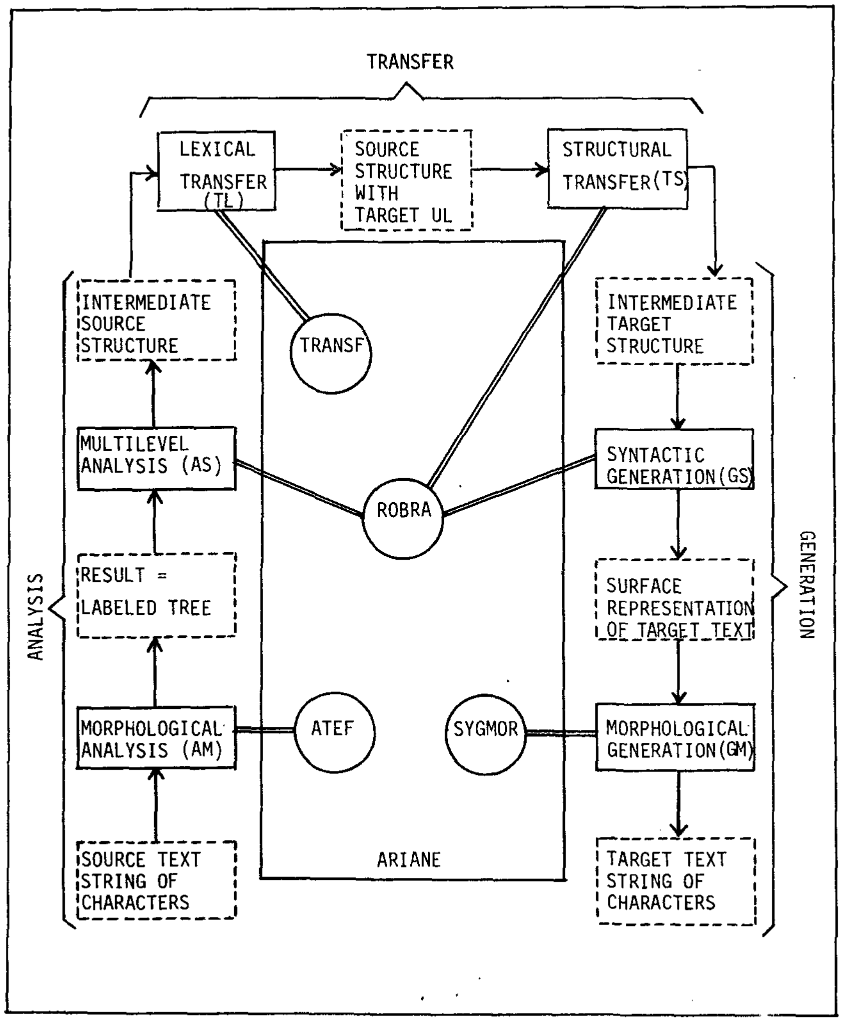
Part III and IV concern the subenvironments under which linguists may prepare the texts in different corpuses, by entering them directly, or reading them from input devices, as well as compose and compile the linguistic files.

Part V describes some of the facilities supported for the interactive debugging of the different steps of an application. Last, but not least, Part VI presents 2 very powerful operators and the subenvironment for human revision. The first operator allows to compile a given application, partially or totally, by issuing one command and its parameters. Should an error occur in some file, the linguist may correct it, and the process will automatically continue the compilation. The second operator allows to use ARIANE-78 as a real operational system. In one command, the user may order all or some texts of a given corpus to be automatically translated by a given source-target language pair.

We conclude by giving some estimates of the static and dynamic costs of the system (disk space, central memory, CPU times).

I - THE SYSTEM AND ITS COMPONENTS

The schema below shows the different steps of the translation process. The components of ARIANE-78 implementing the 4 different algorithmic models appear within circles, they are linked by double lines to the rectangles corresponding to the linguistic data written in the associated metalanguage for the indicated step. Simple arrows show the flow of control. UL means "lexical unit". The usual abbreviation for each step is given between parentheses (e.g. AS for multilevel analysis).


TRANSFER
LEXICAL TRANSFER (TL)
SOURCE STRUCTURE WITH TARGET UL
STRUCTURAL TRANSFER(TS)
INTERMEDIATE SOURCE STRUCTURE
TRANSF
INTERMEDIATE TARGET STRUCTURE
MULTILEVEL ANALYSIS (AS)
ROBRA
SYNTACTIC GENERATION(GS)
RESULT = LABELED TREE
SURFACE REPRESENTATION OF TARGET TEXT
ANALYSIS
GENERATION
MORPHOLOGICAL ANALYSIS (AM)
ATEF
SYGMOR
MORPHOLOGICAL GENERATION(GM)
SOURCE TEXT STRING OF CHARACTERS
ARIANE
TARGET TEXT STRING OF CHARACTERS

In each step, the linguistic data may be of four different kinds :

- grammatical "variables" (like gender, number, semantic type), which define the attributes present on each node of a (linguistic) tree structure ;
- classes, describing useful combinations of values of variables ;
- dictionaries ;
- grammars, containing the rules and the strategy for using them.

They are expressed in a metalanguage. Their syntax and coherency is first checked by the corresponding compiler which then generates a compact intermediate code. At run-time, this code is interpreted.

The conversational monitor ARIANE is a transparent interface with the user. It handles the data-base of linguistic files and of texts, with their intermediate results. The entire system exists in 2 versions (french and english). Any user space is implemented as a virtual machine (under VM/CMS), and may support any number of analyses, transfers and generations. A "source language code" (a "target language code") is associated with each analysis (generation), and a pair "source code-target code" with each transfer.

Once a user has logged in, he (or she) types in "ARIANE" and enters level 0 of the monitor, which is constructed as a hierarchy of subenvironments, with corresponding menus. At any time, entering a null line pops to the next higher environment, if any, and "STOP" allows to exit from any depth. A trace of the session is always constructed and may be printed or discarded (the normal type of terminal is a screen).

ARIANE handles certain global variables, like the dialog mode (brief or detailed), the source and/or target language codes, or the name of the current corpus (homogeneous set of texts with the same method of structuring). Their values are asked by the system, if there is no default, and may be explicity assigned by the user.

All environments provide a help function. Hitting '?' gives a brief explanation, and 'DET' a more complete one (not all environments need it, however).

These features enable users with no particular computer science background to use the system. The only things to know are the standard editor (EDGAR or XEDIT), and the metalanguages of the different components.

II - GLOBAL QUERIES AND ACTIONS

1 - Setting the global variables

CHGMOD	switches between brief and detailed mode.
LG	resets the langage code(s).
INDGTXT	resets the current corpus name.
LGTXT	is the same as LG + INDGTXT.

2 - Information on the data base

The information asked may concern the linguistic data, the corpora, the texts, the (partial) results and the lexical units.

Linguistic data

LIENS	gives all analyses, transfers and generations known to ARIANE in the user space.

PRET visualizes the state of any or all steps of these applications. A given step must be completely compiled in order to execute it.

LISGEN (LISVAR) lists all the linguistic files (only the variables) of a given application, from one step to another, e.g., with "AS TS", from structural analysis to structural transfer.

Corpora and texts

LINDG, LTLG and LTOT list respectively all corpus names, the names of all texts in a given corpus, or the names of all texts in all corpora.

LTXIG an LTXTIG list respectively all texts of a given corpus, or all texts of all corpora. It is not unusual to have more than a dozen corpora and some hundreds of texts in one of them.

Results of different processings

RESULT allows to print a rough translation and/or its revision, with or without the source text. The output is formatted (by using the standard SCRIPT system), and contains the date where the rough translation was obtained (if any), the dates (of compilation) and language codes of the analyses, transfers and generations used, as well as the date of the last (manual) revision, if any. The rough output itself may not be altered manually.

RESGT indicates, for given language code(s), for one (or all) corpus, and for a given step (AS, TS, GS, GM or RV - for revision), the names of the texts having a corresponding result. All intermediate results (not the translations or revisions, however) are erased when the corresponding linguistic data is modified. The rough translation (not the revision) is erased when the source text is modified. If TRAD or REVIS is used instead of GM or RV, RESGT acts like a global RESULT, which allows to print all results of a given type, for one or all corpora, in one command.

Lexical units

LTULS (or LTULC) gives tables indicating where the source (or target) lexical units have been referenced. For instance, it is possible to know, in the case when a given generation is shared by several applications (e.g. russian-french, english-french), for which source languages a given target lexical unit has been an equivalent in one of the transfer dictionaries.

3 - Global actions

DUPLG allows to copy a given application (or part of it) onto another one, which may already exist or not. This is of course very useful during the development of a project.

DESTRUC is the opposite function, and allows (with a lot of warnings !) to erase a given application (or part of it).

ELIMIG allows to erase an entire corpus, with all related results, in much the same way.

III - PREPARATION OF THE TEXTS

This environment is called PRTXT. Its subcommands are either proper or general. The data base of texts is divided into corpora. For each corpus, there is an associated structuring method, defined by an hierarchy of separators.

Usual interpretations are in terms of sections, paragraphs and sentences, but nc interpretation is forced on the user. Hence, a text always appear with an associated tree-like structure of decomposition.

1 - Proper subcommands

CREAT, ELIM and MODIF are respectively used for creating, erasing or modifying a text (under a full screen editor).

REGROUP allows to group several texts in one.

CARTES is used to enter texts via a tape or a card reader.

DESCRT is used to display or to modify the "descriptor" of the current corpus, which contains the ordered list of the separators defining the structuring method of the corpus.

LISTE, with subcommands LITEX, LTDES, LIOCC, allows to print a text, in a formatted or unformatted way, its tree structure (deduced from the structuring method), and the sorted list of its "occurrences" ("words", or "forms").

2 - General subcommands

Under this environment, it is also possible to ask global queries relative to the corpora and the texts : LINDG, LTLG, LTOT, LTXIG, LTXTIG (see above).

IV - PREPARATION OF THE LINGUISTIC DATA

There are 6 subenvironments, denoted by PR<name of phase> : PRAM, PRAS, PRTL, PRTS, PRGS, PRGM. The subcommands are either a global action, or some acronym for a subset of the components of the linguistic data.

1 - Commands relative to the components of the linguistic data

These components vary with the metalanguage of the given step. In general, they comprise :

- the declaration of the "variables", which would be better called "attribute types". The fundamental construct, in any given step, is a decoration type (a hierarchical collection of attributes). Each node of a tree structure bears a decoration of the declared type. The "variables" ("attribute types") are either non-terminal, and correspond to subdecorations, or elementary, and may then be of types akin to PASCAL scalar, set and integer types. See [9] or [11] for examples. We use DV (DVM and DVS in AM, where there are two sets of variables).

- formats, which are constant such record structures. Linguistically speaking, a format corresponds to a "class" (combination of certain values of the variables). We use FTM, FTS and FTSG in AM, and FAF elsewhere.

- boolean procedures, which may express any boolean conditions on the values of the variables of one or several decorations. We use PCP as acronym.

- assignment procedures, which express assignments (of values of variables) from one set of decorations to another. We use the name PAF.

- dictionaries (DIC). All phases (AM, TL, GM) using dictionaries may declare several dictionaries.

- grammars (GR).

We give now the components which are expected for each step of the translation process.

AM	(ATEF) :	DVM, DVS, FTM, FTS, FTSG, GR, DIC (1 to 7)
AS,TS,GS	(ROBRA) :	DV, FAF, GR
TL	(TRANSF):	DV, FAF, PCP, PAF, DIC (1 to 7)
GM	(SYGMOR):	DV, FAF, PCP, GR (0 to 9), DIC (1 to 7).

The division in different dictionaries is used for strategic purposes. At TL, for instance, it is possible to choose, before execution, any ordered subset of the present dictionaries. The induced priorities are used for the choice of equivalents.

The following subcommands apply to any of these components :

M (modify) calls the editor and destructs the invalidated compiled files.
V (visualize) calls the editor, but no modification is allowed.
L (list) prints the data, with a variety of options.
C (compile) compiles the data, with a variety of options.

For the dictionaries, a number of predefined sorting options are defined, and the user may also sort on any key he chooses (TRIZONE command).

2 - Global actions

In any PRXX environment, LISTEF will print all the data, with or without the dictionaries. In PRAM, PRTL and PRGM, the commands CMPDIC and EFDIC are used to compile all the dictionaries, or to erase some of them. In PRGM, the commands CMPGRAM and EFGRAM are used in much the same way for the grammars.

V - INTERACTIVE DEBUGGING

There are 6 subenvironments, denoted by EX<name of phase> : EXAM, EXAS, EXTL, EXTS, EXGS, EXGM. They are used to debug the translation process up to the indicated phase. If some previous intermediate result already exists for the processed text, the system asks the user whether to use it as starting point or not. The possible subcommands are relative to the preparation of a text (typically, a short text for testing a particular phenomenon), or to the execution proper.

1 - Commands relative to the submitted text

These commands are a subset of the PRTXT subcommands, namely CREAT, ELIM, MODIF, LISTE. RESULT is also possible (so that one can visualize the old results before starting a new execution).

2 - Execution proper

Execution is called by the EXECUT command. It may concern the totality of the text, or part of it. This part is expressed in a logical or in a physical way, by giving either the first and last units, or the first and last segments. A unit is any part or subpart of a text, as defined by the structuring method of the corpus. A segment is an integer number of units of the highest possible level such that the number of words of the corresponding fragment of text lies between two specified limits (which are parameters of ARIANE and depend of the size of the virtual memory of the virtual machine). A trace of the segmentation may be provided, as well as CPU times for the different steps. The EXECUT subenvironment asks the user to give, for each step, tracing and output parameters. ATEF and ROBRA provide a variety of possible traces. With some options, execution may be followed step by step. In ROBRA, the trace parameters may be different for each grammar call (a transformational system is a structured collection of such grammars). To give an example, it is quite usual to produce no trace at all for a large part of a process, then a reduced one for the grammar preceding the one being debugged, and then a fairly complete one for some grammar calls, and so on if the behaviour of several grammars must be investigated. This point is quite crucial for the development of applications of reasonable size and scope.

VI - GLOBAL OPERATORS

1 - Global compilation

Due to the variety of components of linguistic data used in a given application (up to 21 dictionaries and usually 5 grammars, with variables, formats, procedures), the global operator COMPGEN is very useful in practice. The user indicates the first and last steps to be compiled (e.g. AM-GM, or AS-TS, etc.), with the compilation options.

Should an error occur in a given component, the system asks the user whether he wants to modify it. If no, compiling may continue (e.g. when a dictionary contains some errors), or not (other cases). If yes, the editor is called, modifications are made, and the global compilation may proceed.

2 - Processing of a quantity of texts

This is the TRAGEN global command, with arguments A, T, GS and GM. These arguments indicate up to which step execution must proceed. In one line, the user gives all general parameters (source and target languages, output units, if any, for the unknown words, the results, the CPU times, and the trace of segmentation ; and form of the result - with or without the source text).

The system then asks which texts are to be processed. It is possible to give them explicitly, by entering their names and their corpus, or to specify an entire corpus, with or without the possibility to amend (under the editor) the list proposed by the system.

Hence, TRAGEN (GM) may be used as a production environment "cranking out" rough translations of texts already in the data base. An operational production environment, with constant input flow of texts (from tapes, disquettes, terminal, etc.) and output flow of translations, is currently being developed. It will also include a more sophisticated revision subenvironment than ARIANE.

3 - Human revision

Being used as a system for automated translation, ARIANE provides an environment for human revision, called REVISION. The rough machine output may or may not exist. In the last case, revision is the same as human translation and revision. In the first case, obviously the most interesting, a file for the revised translation is created by copying the rough translation. The rough translation itself may never be altered by the user.

Revision is done under the standard editor, with 2 or 3 windows on the same screen, on which the source text, the rough translation (optionnally) and the revised text appear. In the future, a real text processing system may be used instead of the editor, with functions specialized to frequent actions done during revision, such as erasing, inserting or swapping words, sentences or paragraphs.

VII - STATIC AND DYNAMIC COSTS

We don't consider here the cost of preparing the grammars and dictionaries for a given application, but rather the static and dynamic costs incurred while developing, maintaining and using an application.

1 - Static costs

We refer to them only in terms of space on secondary storage (disks).
This cost is divided into 2 parts :

- the space for the files containing the basic software in executable form. This space is shared by all user spaces, and amounts to roughly 7 Mbytes for any of the 2 (english and french) versions of ARIANE-78.4.
- the space for the texts and the linguistic data.

The space taken for a source text is just the size of the file containing it, and the same goes for its rough translation and its revision, if any. If an intermediate result is kept, its size is roughly 4 times the size of the source text (it contains some representation of the associated tree structure). For any text and any language pair, up to 3 intermediate results may exist (after analysis, transfer, or syntactic generation). Obviously, the space taken by the texts does not measure the complexity or validity of an application. It is only a measure of the size of the corpus (corpora) used for developping and debugging an application. In the current russian-french application, for example, the source texts occupy roughly 12 Mbytes on disk. This corresponds to roughly 700000 words, because of some redundancies.

The space taken by the linguistic data is first divided in two parts : the source data and the compiled data. Second, one must distinguish between data relative to the linguistic model, and the typology (variables, formats, procedures, grammars), and the dictionaries, the size of which may expand if domains are added, or better covered. The following table gives this partition for a previous russian-french version (1980), using roughly 4000 lexical units, and reasonably large grammars.

Linguistic data	Relative to the model	Dictionaries	Total
source	0,35	1,25	1,6
compiled	0,3	0,5	0,8
Total	0,65	1,75	2,4

The marginal space for adding one lexical unit (in the dictionaries of AM, TL and GM) is roughly 310 bytes/UL in source form, and 120 bytes/UL in compiled form. The compression factor is higher for dictionaries than for grammars.

2 - Dynamic costs

They may be measured in terms of the CPU time and the size of the virtual memory used while executing a translation, as this computation dominates the costs.

A main principle of the implementation has been to use the virtual memory facility in such a way that, should the real memory be big enough, the only I/O incurred for executing a translation are due to loading the executable module generated by ARIANE, and reading/writing the input text, the intermediate results and the translation on disk files. In other terms, everything is done in central memory.

For an application like the one mentioned above, a 2 Mbytes virtual memory is enough to contain the OS (CMS) of the virtual machine, the compiled tables, ARIANE's programs, and the work areas. As we said before, adding 1000 UL causes this size to augment by roughly 120K.

As far as CPU times are concerned, only virtual CPU is significative. As it varies considerably from one type of machine to another, we prefer to give estimates in millions of (virtual) machine instructions necessary to translate one word, or Mi/w. If we consider various applications (russian-french, french-english, english-malay or english-chinese), this complexity lies between 3 and 7 Mi/w. According to the size and load of the computer system, a certain overhead (for simulation and paging) has to be added to obtain the total CPU time. No uniform rule may be given.

REFERENCES

[1] Nêdobejkine N. - "Application du système ATEF à l'analyse automatique des textes russes". COLING-73, Pisa, 1973.

[2] Quézel-Ambrunaz M., Guillaume P. - "Analyse automatique de textes par un système d'états finis". COLING-73, Pisa, 1973.

[3] Chauché J. - "Transducteurs et arborescences. Etude et réalisation de systèmes appliqués aux grammaires transformationnelles". Doct. thesis, Grenoble, 1974.

[4] Boitet Ch. - "Un essai de réponse à quelques questions théoriques et pratiques liées à la traduction automatique. Définition d'un système prototype". Doct. thesis, Grenoble, 1976.

[5] Boitet Ch. - "Problèmes actuels en traduction automatique : un essai de réponse". COLING-76, Ottawa, 1976.

[6] Nêdobejkine N. - "Niveaux d'interprétation dans une traduction multilingue : application à l'analyse du russe". COLING-76, Ottawa, 1976.

[7] Thouin B. - "Système informatique pour la génération morphologique de langues naturelles en états-finis". COLING-76, Ottawa, 1976.

[8] Boitet Ch., Guillaume P., Quêzel-Ambrunaz M. - "Manipulation d'arborescences et parallélisme : le système ROBRA". COLING-78, Bergen, 1978.

[9] Boitet Ch., Nêdobejkine N. - "Russian-French at GETA : outline of the method and detailed example". COLING-80, Tokyo, octobre 1980.

[10] Guilbaud J.Ph. - "Analyse morphologique de l'allemand en vue de la traduction par ordinateur de textes techniques spécialisés". Thesis, Paris, 1981.

[11] Nêdobejkine N., Boitet Ch. - "Russian-French machine translation at Grenoble. A general software used for implementing a particular linguistic strategy". To appear in Linguistics (1982).