GETA's MT methodology and its current development towards personal networking communication and speech translation in the context of the UNL and C-STAR projects

Christian BOITET

GETA, CLIPS, IMAG-campus

F-38041 GRENOBLE cedex 9

Christian.Boitet@imag.fr

Abstract

Ariane-G5 is an integrated environment initially designed to facilitate the development of multilingual MT systems for revisors (MT-R), where output quality is obtained by using the heuristic programming facilities of its 5 rule-based languages to specialize the lingware components to the sublanguage at hand. It can support many MT architectures and linguistic methodologies and accepts whole paragraphs or pages as units of translation rather than separate sentences. For MT-R, B.Vauquois' mutltilevel transfer approach has given excellent results on a large number of MT-R mockups and prototypes, as well as two large scale operational systems. As both the computer tools and the linguistic methodology are not embodiments of a particular theory, they are quite easy to adapt to new problems. In the last few years, they have actually been revised and further developed in the framework of new research on high quality MT for monolingual authors (MT-A), relying on a disambiguation dialogue with the author (DBMT), following an all-paths analysis.

The evolution of Ariane-G5/LIDIA and associated linguistic methodologies is now motivated by two projects of different aims and requirements. The UNL project of personal multilingual high-quality communication over the Internet requires the construction of a large lexical database from which coherent dictionaries for MT and for interactive disambiguation will be generated. For the C-STAR Speech Translation (ST) project, it becomes necessary to start from phonetic lattices output by a speech recognizer, and to transmit some linguistically useful memory from one dialogue turn to the next.

Keywords

Machine Translation for Revisors, MT-R, Dialogue-Based Machine Translation, DBMT, Speech Translation, Machine Interpretation, Ariane-G5 MT shell, Multilevel Structures, Specialized Languages for Linguistic Programming, SLLP.

Introduction

From 1970 to 1988, GETA developed a linguistic methodology for building multilingual MT systems for revisors (MT-R), specialized to sublanguages, and relying on heuristic analysis. A comprehensive MT shell, Ariane-G5, has been programmed and used to develop a large number of MT-R mockups and prototypes, as well as two large-scale operational systems. Since 1988, the methodology and the computer tools have been revised and further developed in the framework of new research on high quality MT for monolingual authors, relying on a disambiguation dialogue with the author (DBMT), following an all-paths analysis. The software architecture has become distributed, with the author using a middle-range Macintosh, and Ariane-G5 running as a remote server and accessed through the Internet.

This new basic software is now being used in two projects with different aims and requirements. First, in the UNL project of personal multilingual high-quality communication over the Internet, our challenge is to go from the LIDIA mockup to a real system relying on a huge lexical database. Second, we have started a long term project in Speech Translation (ST), and concentrate first on the French part of C-STAR II demonstrators, to show that useful ST systems can be obtained by integrating admittedly imperfect speech recognition and machine translation components in a communication environment permitting limited user control and multimedia feedback.

In both projects, we reuse the Ariane-G5/LIDIA software, with some new developments: (1) for UNL, we are building a multilingual database to generate Ariane and interactive disambiguation dictionaries, and filters between UNL graphs and Ariane-trees; (2) our first C-STAR prototype uses http access to Ariane-G5, and MT starts from a phonetic word lattice output by the speech recognizer. In both projects, we also reuse the LIDIA multilevel pivot approach, where linguistic structures contain a lexical level of interlingual acceptions, but the analysis strategy is heuristic for C-STAR and all-paths as in LIDIA for UNL, and interlingual acceptions are of a different nature.

I.   Basic software tools and linguistic methodology for multilingual MT-R

Research in MT has been pursued at Grenoble since 1961. At the end of the first period, around 1967-70, a first Russian-French system had been developed, and tested on more than 400,000 running words of real texts (scientific articles). The aim was *MT for the watcher,* or "pure" MT, or MT-W, where a fully automatic process produces unrevised "rough" translations good enough that the reader, assumed to be a specialist of the domain, can access the content of the original without knowing the source language. Such translations are often judged to be very bad by professional translators and very good by users. Almost all PC-based current systems are of this kind.

Due to a change of computer system (IBM 7044 to IBM 360/67), which would have necessitated an important conversion hardly justifiable without the prospect of immediate operational use, this first system was abandoned, despite its remarkable quality and coverage. This was the occasion to start exploring new ideas in the context of *MT for the revisor,* or MT-R, which aims at automatically producing "raw" translations designed to be revised by a professional in order to get final results of professional quality. As the step of human revision must naturally be performed on a computer, *MT for the translator* (machine aids for human translation and revision) was also studied.

In 1978, a new methodology for linguistic programming had been formulated by B. Vauquois (multilevel transfer approach, heuristic programming), and began to be tried on various language pairs, with a large effort on Russian-French, using the components of the first ever (and maybe still unique) comprehensive "MT shell", that is to say, a computer environment for the development and use of multilingual MT systems, which was called "Ariane-78" at the beginning of 1978, when its first complete version became available. Incidentally, this name was chosen several months before it became also used for the European rocket. It was meant as a reference to the Greek goddess and her famous thread, in order to stress the idea that computer science, even if essential, must be put to the service of linguists and lexicographers who are not computer scientists and allow them to work autonomously, thanks to symbolic, rule-based, "specialized languages for linguistic programming" (SLLP) and to a transparent interactive user interface.

Thanks to support from CNRS and DRET (Ministery of Defense), a first full-sized "preoperational" Russian-French system was developed and tested, from the beginning of 1980 to the end of 1986. In parallel, an effort in technological transfer towards industry was undertaken, in particular in the framework of the Machine-Aided-Translation National Project (CAT-NP, or PN-TAO in French, 1983-87 [7]). GETA augmented considerably the power and reliability of Ariane-78.4, and was at the heart of the linguistic specification of the French-English system for aircraft manuals built in cooperation with industrial partners.

Since then, the Ariane system has been considerably extended [15]. The first section describes it in some detail, to show that such an MT shell is rather complex. In the second section, we show how linguistic programming techniques in MT have changed from the stage of workmanship to that of real engineering. This has in turn led to the development of new tools associated with Ariane-G5 and to the evolution of the linguistic methodology (static grammars). In order to give a concrete overview of the linguistic methodology and of what modern MT for the revisor can produce, this section also presents several raw (unrevised) industry-oriented translations produced by B'VITAL's Ariane/aero/F-E system after 3 years of development.

A.   Ariane-G5, an MT shell for building multilingual MT-R systems

1.   General principles

Ariane-G5 is a generator (G) of MT systems based on five (5) specialized languages for linguistic programming (SLLP). Each such language is compiled. The internal structures produced by its compiler are used as parameters by its "engine". The complete documentation, in French, is available at GETA ([6] and later additions). Large parts are available in English and in Chinese (translation by the JTMAT group at Shanghai Jiao tong University).

Although Ariane-G5 is particularly well adapted to the transfer approach and to heuristic analysis and generation, it does not impose them. Apart from some implementation limits, the only strong constraint is that the structures representing the units of translation be decorated trees.

Intrinsic semantics (a term borrowed from J. P. Desclés) is represented in these languages, hence in a linguistic fashion. If one wants to write a system specialized to a restricted sub-language and to a microdomain, it is possible to use the same technique as for METEO [20] and to write "semantic"

grammars and dictionaries. In order to construct a system equipped with *extrinsic semantics* ("ontology" of the universe of reference), it would be necessary to couple Ariane-G5 with an "expert corrector system", as suggested and prototyped in [27].

As opposed to almost all existing systems, Ariane-G5 presents the advantage that the unit of translation is not restricted to the sentence, but may contain several paragraphs (in general, 100 to 200 occurrences, i.e., almost a standard page).

a.   Hardware and software environment

Ariane-G5 runs under VM/ESA/CMS, on IBM computers with 390 architecture (a 9221-130 at GETA, PS-2 with 390 card…). Since 1993, it is accessible through the Internet. VM/ESA is a hypervisor which simulates a set of "virtual machines". Each virtual machine runs under its own operating system. For example, it is possible to let virtual machines run at the same time under MVS/TSO, CMS, and AIX. The RSCS subsystem allows one to organize the virtual machines and the real resources (peripherals) as a network. CMS is a powerful interactive single-user operating system, which supports a large number of programming languages and tools. It is generally used for software development or for interactive applications.

The current version of Ariane-G5 represents about 400,000 source lines, plus 30,000 lines of messages for each dialogue language[1]. The executable part resides on a "minidisk" (virtual disk) of a particular virtual machine, and takes about 10 Mb. To allow another virtual machine to be a "user machine" (of Ariane), one simply performs a logical connection (as "B/A") of Ariane's minidisk to the machine's minidisk ("A"). From then on, Ariane manages on the user minidisk two specialized data bases, one containing the lingware and the other the texts (with a maximum of 3,000 source and target languages, 1,000 "corpora" and 10,000 texts per corpus). The minimal computer background necessary to use Ariane-G5 consists of learning the elementary commands for beginning and ending a VM/ESA session (login, logout), the XEDIT screen editor, and, for the developers of MT systems, the organization of the interactive monitor and the SLLPs.

b.   Logical organisation

i.   Steps, phases and articulations of the translation process

Translation from a "source" language into a "target" language is performed in three successive "steps" : analysis, transfer and generation[2]. Each step is realized in at least two and at most four successive "phases", possibly linked together by "articulations", which may be considered in first approximation as simple "coordinate changes". Each phase is identified by a two-letter mnemonic (e.g. AM for morphological analysis — analyse morphologique in French), and each articulation by a four-letter mnemonic (e.g. AMAS for the AM-AS articulation).

In analysis, the successive phases are :

AM	(morphological analysis)	obligatory,	written in ATEF ;
AX	(expansive analysis X)	optional,	written in EXPANS ;
AY	(expansive analysis Y)	optional,	written in EXPANS ;
AS	(structural analysis)	obligatory,	written in ROBRA.

In transfer, the successive phases are :

TL	(lexical transfer)	obligatory,	written in EXPANS ;
TX	(expansive transfer X)	optional,	written in EXPANS ;
TS	(structural transfer)	obligatory,	written in ROBRA ;
TY	(expansive transfer Y)	optional,	written in EXPANS.

In generation, the successive phases are :

GX	(expansive generation X)	optional,	written in EXPANS ;
GS	(syntactic generation)	obligatory,	written in ROBRA ;
GY	(expansive generation Y)	optional,	written in EXPANS ;
GM	(morphological generation)	obligatory,	written in SYGMOR.

---

[1] Ariane-G5 is programmed in a totally multilingual fashion. Only the French and English versions of the messages have been completed at the time of writing.

[2] This term is used rather than "synthesis", by analogy with that of "generation" in compiler construction.

In the current version, the order of these phases within each step is fixed. Hence, the possible "articulations", all written in TRACOMPL, are AMAX, AMAY, AMAS, AXAY, AXAS, AYAS, ASTL, then TLTX, TLTS, TXTS, TSTY, TSGX, TSGS, TYGX, TYGS, and finally GXGS, GSGY, GSGM and GYGM. As a matter of fact, one needs to write articulations only for composing two phases taken from lingware components using heterogeneous "sets of variables" (see below).

The linguistic operations performed in each phase do not necessarily correspond to their names in a strict manner. For example, morphological analysis may be realized in AM, but it is also possible to distribute it between AM, AX, AY and a fraction of AS (for example, to test for the occurrence of "predicted" possible discontinuous idioms). In general, lexical transfer is also distributed between (at least) TL and TS, for analogous reasons. Similarly, morphological generation of a language such as Arabic [42] may advantageously be distributed between the end of GS and GM.

ii.   Essential data types and basic terminology

At the input and output sides of the translation process, the unit of translation is a simple *string of characters.* The 256 EBCDIC characters may be used in the specialized languages to build strings, and all are considered to be atomic (e.g., "é", "É" and "ê" are not known to be variants of "e"). The blank (X'40') is used as separator of *occurrences.* A translation unit, then, is received as a *sequence of occurrences* by the AM phase.

From the output of AM to the input of GM, a unit of translation is represented by a decorated tree. Each phase contains a part, named "DV", where the linguist declares the decoration type, or *set of variables* in Ariane-G5 terminology.

An *elementary variable* ("variable" for short) is defined by a name and an (anonymous) type. This type is defined by a *kind* (exclusive, non-exclusive, arithmetical, lexical) and a *list of values*. Each variable has a null value, denoted by its name followed by "0".

- An exclusive variable V defined by V == (V1, V2, V3) takes its values in {V0, V1, V2, V3}. There is an analogy with the scalar types of Pascal, but here value identifiers are local to the types.

- A non-exclusive variable V defined by V == (V1, V2, V3) takes its values in the power set of {V1, V2, V3}. V0 denotes the empty set. There is an analogy with the set types of Pascal, the preceding remark being still valid.

- An arithmetical variable V defined by V == (n), where n is a non null natural integer, takes its values in : $\left[-2^{\lceil \log_2(n) \rceil}, +2^{\lceil \log_2(n) \rceil} - 1 \right]$.

- There is always one (and only one) "lexical" variable, named "UL", for *lexical unit* (unité lexicale), which is predefined and takes its values in the set consisting of :

  - the predefined values '' (UL0), 'ULTXT', 'ULFRA', 'ULSOL', 'ULOCC', 'ULMCP' ;

  - the values introduced in the lingware components (essentially in the dictionaries) ;

  - the values constructed dynamically (for example to handle unknown words).

A *decoration,* or *mask of variables* in Ariane-G5 jargon, is a combination of values for all the variables of the considered set, very much analogous to a property list in LISP.

It is possible to group variables in a hierarchical fashion, the top of the hierarchy being predeclared until a level depending on the specialized language. VAR always denotes the set of variables minus the UL variable.

In ATEF, two subsets, VARM and VARS, are distinguished and declared separately, for the abovementioned "morphological" and "syntactic" variables (although, as for the phases, the linguists do not in general respect that division and add a number of variables of semantic nature). VARM and VARS are further subdivided into VAREM and VARNM, VARES and VARNS (exclusive and non-exclusive), as there are no arithmetical variables in ATEF.

In ROBRA and EXPANS, where the three kinds of variables are possible, the top of the hierarchy is VAR (VAREπ, VARNπ, VARAπ), where π is a character (redefinable in DV) characteristic of the phase. By default, π is set to S, R, C, D, G for AS, TL, TS, GS, GM, and to X for the other EXPANS phases. For example, in AS, in order to refine VARE into syntactic and semantic variables, themselves divided into properties and relations, one might write :

-EXC-   ** (key-word for "exclusive").

   VSYNTE ==   (PSYNTE (CAT (N, V, A, R, S…), K (PHVB, PHINF, GV, GN, GA))
                   ,RSYNTE (FS (SUJ, OBJ1, OBJ2, EPIT, CIRC…)).

   VSEME ==   (PSEME (PREDIC (STATE, ACTION, PROC), MATTER (DISC, CONT))
                   ,RSEME (RL (ARG0, ARG1, ARG2, ARG01, ARG02, TRL10…))).

-NEX-   ** non-exclusive.

   …

A *format* (template) is a constant mask of variables to which a name has been given, in order to use it as an abbreviation in dictionaries and grammars. A *decorated tree* is an oriented and ordered tree where each node bears a decoration.

iii.   Components and variants of a phase

As in most NLP systems, a lingware written in a specialized language is organized into physically distinct *components,* for reasons of modularity and size. The components of a phase form an acyclic dependency graph (known by the compiler).

- An ATEF phase contains two components of variables declaration (DVM, DVS), "morphological", "syntactic" and "general" formats (FTM, FTS, FTSG, the last one being optional), 1 to 7 grammars GRi ($1 \leq i \leq 7$), 1 to 6 dictionaries of "morphs" DICi ($1 \leq i \leq 6$), at least one of them being of "bases" (morphs with lexical references), and from 0 to 7 dictionaries of fixed connex idioms, DICi ($7 \leq i \leq 14$). FTM depends on DVM, FTS on DVM and DVS, FTSG on FTS, the DICi on the formats, and the grammars on the dictionaries.
- An EXPANS phase contains a component of variable declaration (DV), one of "condition and assignment procedures" on decorations (PROC), one of "assignment formats" (FAF), optionnally one of "proper condition formats" (FCP), and from 1 to 7 dictionaries (DICi). PROC, FAF and FCP depend on DV, and the DICi on the preceding.
- A ROBRA phase contains DV, FAF, and from 1 to 7 grammars (GR1 to GR7), with the natural dependencies.
- A SYGMOR phase contains DV, FAF, PCP ("proper condition procedures"), GRi ($1 \leq i \leq 7$), and DICi ($1 \leq i \leq 13$, at least one dictionary being accessed by the UL), with the same dependencies as in ATEF, with the exception that grammars do not depend on dictionaries.
- A TRACOMPL articulation contains only one component, DV.

The Ariane-G5 environment ensures at all times the coherency of the compiled components as a function of these dependencies and of the modifications made by the user (linguist).

Each phase may give rise to *variants,* to be defined according to the types of texts to be translated.

- In ATEF and SYGMOR, one chooses one of the grammars.
- In EXPANS, one chooses an ordered subset of the dictionaires, and the mode (deterministic or not) of the engine.
- In ROBRA, one defines a list of at most 14 grammars to be executed sequentially, the same one being allowed to occur more than once.

By combining these choices and the choice of a path in the graph of phases (from AM to GM for a translation), one obtains *execution lines* (for debugging) and *production lines* (for cranking out translations) which are also memorized and managed by Ariane-G5.

c.   Principles of linguistic use

Although Ariane-G5 does not propose or impose any methodology of linguistic programming, most of its users follow a certain number of principles, mainly due to B. Vauquois, which it may be useful to mention briefly at this point.

i.   Intermediate structures

In the context of MT-R of a sublanguage, B. Vauquois recommends that analysis delivers a *UMA-structure,* for "unique, multilevel and abstract" structure [51]. The geometry of the tree reflects one choice for the organization into syntagmatic groups[3], but the structure is an "abstract tree", from

———————————————

[3]The possibility to produce several geometries or bracketings for a sentence exists and has been used later for the

which the text may not necessarily be reconstructed in an immediate way[4]. For example, it is convenient to regroup discontinuous constituents (e.g. "les garçons les ont tous vues" for "the kids$_i$ have all$_i$ seen them$_j$"), to "variabilize" negations, auxiliaries, articles, strongly governed prepositions, certain modals, etc., thus obtaining trees considerably smaller than the "concrete" ("surface") trees provided by direct application of extended context-free grammars (GPSG or others). In principle, each internal node dominates a leaf which is the *governor* of the group (from "gouverneur" in French, usually "head" in English), unless the governor is itself a compound. In order to get a dependency structure analogous to those of the Prague school, it is enough, in a first approximation, to recursively "send up" each governor to replace its mother node. In other words, Vauquois' structures are "lexicalized" intermediates between constituent and dependency structures, at least geometrically.

Properties and relations are coded in the decorations attached to the nodes and constitue the "algebraic" part of the structure. For example, a node having "attribute of object" as value of the syntactic function (SF=ATROBJ) is the attribute of the group dominated by its (unique) sister node having SF=OBJ1. Hence, there are two syntactic levels, that of classes (morphosyntactic and syntagmatic), and that of functions. To translate into languages which are not extremely near to the source language without having to write large structural transfers, it is advisable to add two more levels, logical and semantic.

The logical level (RL variable, for "relation logique") gives the positions of arguments of linguistic predicates. ARG0 denotes the logical subject (most often actor or agent) of the predicate which is the head ("governor") of the same group, ARG1 denotes its logical object (in general the patient, but not in ergative constructs such as "the twig breaks"), and ARG2 denotes its third argument. The numbering is such that ARG1 corresponds to OBJ1 in standard active constructs, but that is purely a convention. For example, "the building of the house" and "to build the house" have identical structures at that level, the group "(of) the house" being ARG1.

TRL10 is used in place of ARG1 if the predicate is attributive ("to be", "to seem", "to appear"…), and TRL21 in place of ARG2 if the predicate attributes ARG2 to ARG1 ("to consider ARG1 as TRL21"). This is a way to indicate that the relation does not link the node with the governor (the predicate), but with another argument. Similarly, one often uses another variable (such as RLI, for "inverse logical relation") to code the link between arguments in control constructs. For example, in "I ask him to come", the group "to come" is ARG1 of "ask", and bears RLI=00 if ARG0 (I) is coming and RLI=02 if ARG2 (he) is coming.

Finally, we use, mainly on circumstancials (modifiers), the semantic relation (RS), which *grosso modo* corresponds to the "deep case" (localization, origin, goal, accompaniment, manner, qualification, measure, cause, concession, etc.). In practice, RL and RS are complementary, because it is extremely difficult (even manually) to assign RS to arguments in a reliable manner, and circumstancials can be correctly translated only if their RS are known.

In this respect, the famous problem of the translation of prepositions is often not well stated. If an argument is concerned, the whole construct (predicate+arguments, e.g. "to talk about sth. with sb.", "to count for sb., sth.") should be translated as a block. If a circumstancial is concerned, the RS, possibly particularized by the preposition (or its absence) should be translated. For instance, in "to come by Lyon" and "to come via Lyon", the circumstancial should bear RS=LOC, SEM=SPACE and SLOC=QUA (localization in space, movement through sth.), thus allowing for exact translation of "by" ("par" and not "près de", "à côté de", "devant", "de", "d'après", "suivant", "à"…). Keeping the preposition also allows one to translate more exactly in a language like French, which also has two prepositions for this sense ("par", "via").

Several levels are similarly used for the actualization variables, such as number (morphological and logical), time vs. tense, etc. The order of the text is reflected as much as possible in the structure. As a matter of fact, order gives important information which is not well formalized, such as thematic articulation and emphasis. This avoids coding it explicitly in a tactical variable.

If one considers the UMA-structure of a sentence only at the "deep" levels, it can be thought to represent a whole family of sentences of equivalent meanings. If it is considered at all levels, it

---

LIDIA project: we will then speak of MMC- or MMA-structures, for "multiple, multilevel, concrete/abstract".

[4]An example of two concrete and abstract multilevel structures for the same sentence is given in section II.

should correspond to only one sentence, notwithstanding spelling variants (such as disc/disk, program/programme, or corpuses/corpora).

Besides these various levels of linguistic description, one also encodes in the UMA-structures produced by analysis unresolved ambiguities and doubts on parts of the construction, in order to avoid a combinatory explosion, and to be able to warn the revisor and at the same time to try to transfer those ambiguities which persist in translation (e.g. "the conquest of the Romans").

The aim of transfer is to perform lexical translation, and some adaptations of the structure aiming at delivering to the generator a structure coherent with the linguistic system of the target language. This structure is called *GMA-structure,* for "generating, multilevel, abstract structure". In principle, the generator considers that the GMA-structure it receives is under-specified with respect to the surface levels, and recomputes them.

Hence, the first logical step of generation consists of selecting a paraphrase of the meaning expressed by the GMA-structure by computing the UMA-structure of the translation to be produced. The second step consists of producing a surface tree ("concrete" tree, or *UMC-structure),* by creating nodes for articles, auxiliaries, negation elements, punctuations, by dividing or merging sentences if necessary, etc. The third step is the morphological generation which, starting from the sequence of the leaves, constructs the occurrences of the final text.

ii.   Organization of dictionaries

The notion of lexical unit is very useful for the generation step. It allows one to represent derivational families in a compact way. Modern dictionaries use a similar notion. In analysis, that notion allows one to reduce the size of the dictionaries, and to handle in a systematic way neologisms obtained by productive derivations.

From the linguistic point of view, one is naturally led to regroup, for example, *to heat, heater, heating, heatable,...* in the UL *heat-V,* the derivations used having the three aspects of semantics, syntax and semantics (e.g. agent or instrument noun ending in -er), in order of importance. From the practical point of view, one often separates the agent or instrument noun from such a family, because the derivation in question cannot be used to produce paraphrases usable in translation ("heater" —> "something which heats" ?), and because that allows one to separate the purely terminological indexing of those terms from the more complex indexing of a whole family of deverbals.

According to the syntactic class of the principal lemma (source of the derivations), one distinguishes verbal, nominal and adjectival ULs. Of course, there are ULs reduced to only one term (function words, non-derived adverbs such as *there, here...).*

In Ariane-78, it was necessary to represent all lexical information in the AM dictionaries. In Ariane-G5, lexical expansion phases have been introduced to give the possibility of distributing the information. A possible organization is to use AM to go from morphs (roots, affixes, endings…) to lemmas, AX to go to the ULs, and AY to handle non fixed or discontinuous idioms (e.g. verbs with separable particles in German). If interlingual wordsenses or "acceptions" are added, they may be introduced as values of a non-exclusive variable defined by SENSE == (1, 2…, MaxNumSense).

iii.   Organisation of grammars

In ATEF and SYGMOR, organizing the grammars is quite simple, although one should resist the temptation to overuse ATEF heuristic functions.

In ROBRA, the programming technique is quite different according to whether analysis, transfer or generation is concerned. Analysis usually begins by working in parallel on the whole unit of translation to normalize the tree (compound words, resolution of immediately solvable ambiguities, grouping of discontinuous idioms, dates, proper nouns, etc.). Then, a sentence-specific transformational sub-system is recursively called on each sentence. Other sub-systems may be called on groups (clauses, phrases). This allows the strategy (traversal of the control graph) to be directed by the data. The end of analysis usually consists of processing again the whole tree, which for example allows one to try to find referents of pronouns for which none has been found inside the sentences where they occur.

In transfer, things are quite simple. One handles the translations of complex groups, tests the context conditions coded in the subtrees representing multiple translations, in order to reduce poly-

semies, and adjusts the g-structure, thereby possibly annotating it with advices or orders to the generator, in order to trigger the production of particular syntactic forms (e.g. to get the reflexive or the passive for stylistic reasons).

Generation uses recursive descent to produce the UMA-structure, and then to begin construction of the UMC-structure. Parallel processing is usually used to assign final values to surface actualization variables (to propagate agreement constraints), which may lead to slight modifications of the geometry (insertion of auxiliaries and clitics). On should be cautious not to use grammars at the same time iteratively and recursively, because that leads to numerous useless grammar calls.

2.   The interactive interface

Under Ariane-G5, it is possible to :

- work on the linguistic components (phases, articulations) in the subenvironments PRAM,… PRGM, PRAMAX,… PRGYGM (creation, modification, compilation, listing…).
- work on the texts (PRTXT), with numerous facilities.
- work on the "execution lines" and on the "production lines".
- execute all or part of the translation process for debugging purposes (each phase receives parameters for tracing and for outputting the result), using one available "execution line".
- crank out translations (no tracing is permitted), using an available "production line".
- revise raw translations (in multiwindow mode under XEDIT, with the possibility of switching to THAM, which offers additional facilities).
- perform actions on several phases at the same time (compilation, listing, erasing, duplication…).
- obtain various informations on the objects managed by Ariane (list of source and target languages, of corpuses, links between source and target languages, compilation status…).
- read or modify the global parameters (dialogue mode, current source and target languages, current corpus…).

An on-line help is available, for the most part at two levels of detail. This interactive monitor is described in [44]. It represents about 50,000 lines of EXEC2 and 100,000 of ASM370.

3.   The specialized languages

a.   ATEF, a language for morphological analysis

ATEF was designed in 1971 by J. Chauché [23], who wrote the engine, while P. Guillaume and M. Quézel-Ambrunaz wrote the compilers of the different components. Since then, ATEF has undergone numerous extensions, but the underlying algorithmic model has not varied. As a matter of fact, it is a very satisfactory tool.

The system successively handles each occurrence of the text, examining *a priori* all possible analyses (non-deterministic total mode with backtracking). The current occurrence is named C. An analysis result is a decoration or a sequence of decorations (in the case of compound words). Each step of a particular analysis consists of choosing one of the open dictionaries, in finding there an item which key, the *morph,* is a prefix (or a suffix, in right-to-left mode) of what remains to be analyzed (noted A), which is reduced accordingly, and in applying one of the rules associated with the morphological format of the considered item.

The rules may contain conditions bearing on the current state (decoration C), on the strings C and A, on the partial results produced by the current analysis (PS1 to PS9) in case of a compound word, and also on the four preceding occurrences (from P1 to P4) and on the results of their analysis. A particular form of condition consists of giving a list of "sub-rules" and in asking that at least one of them applies (as a sub-rule may itself have sub-rules, that happens in non-deterministic unary mode with backtracking). It is finally possible to store a condition on the analysis of the following occurrence.

There exist three types of action : assignment of values to mask C, transformation of what remains to be segmented (string A), and call of special functions. These functions allow to :

- control the built-in backtracking by pruning the choice tree (functions FINAL, ARRET, ARD, ARF, STOP) or by opening and closing dictionaries (through assignment of the obligatory non-exclusive morphological variable DICT) ;

- produce a partial result from the current state C (function SOL) ;
- transform C or A into a UL, thereby reducing A to '' (functions TRANS and TRANSA) ;
- decide that a sentence boundary has been reached (function INIT).

If an occurrence is not recognized ("unknown word"), that is, if no analysis succeeds in reducing A to '' while producing a current state C having a non-null UL, the system starts analysis again, after having attached to the occurrence the obligatory morphological format MODINC, which must in particular call the obligatory rule MOTINC ("rule of the unknown word"). As that rule may call sub-rules, and as that format may call other rules, it is possible to construct a true "grammar of the unknown word", and to program sophisticated strategies for analyzing unknown words.

During processing, the automaton (ATEF "engine") constructs a ("4-colour backward") graph where the nodes are the masks (or lists of masks for the compound words) associated with the solutions found, and the edges indicate compatibility between analyses (at distance 1 to 4). The final graph is finally transformed into the desired form. The Q-graph output is now no longer available, and two other output forms, as 1-colour forward graph and as tree "with homosentences" (presenting all paths in the 4-colour graph separately) are no longer used.

The standard output of ATEF is a tree "without homosentences", which encodes ambiguities. Its root corresponds to the whole text and bears UL='ULTXT'. Its daughters correspond to the sentences (determined by the grammar) and bear UL='ULFRA'. Under each of them are nodes with UL='ULOCC', which correspond to the occurrences (words or fixed connex idioms). Under each 'ULOCC' are the different results of the morphological analysis of the corresponding occurrence. Each result is either a mask of variables (a node) or a sub-tree with root having UL='ULMCP' (compound word) dominating the masks corresponding to the different parts recognized in the word.

b. ROBRA, a language for transforming decorated trees

ROBRA [14] is a language for writing transformational systems working on decorated trees. It is the successor of the CETA language [23] Numerous extensions have been introduced, the semantics has been made more precise, and the engine has been totally respecified and rewritten.

A transformational system (ST) is defined by a control graph (GC), a set of transformational grammars (GT) and a set of rules (RP for "production rules"). A GT is an ordered set of rules. A GC is a graph where each node bears a GT or the exit symbol (&NUL) and the edges bear tree conditions. Note that each "grammar" component GRi of a ROBRA phase actually countains a whole transformational system, possibly consisting of a large GC with dozens of GTs.

To execute an ST on an object tree (AO), ROBRA uses the GC as non-deterministic (unary with backtracking) control structure : starting from an initial node, it looks for the first valid path leading to an exit. On this path, it executes the grammars countained in the nodes, and traverses an edge only if the current AO verifies its condition.

The execution of a GT consists of one elementary application in unitary (U) mode, or of several in iterated mode (E for "exhaustive"). In an elementary application, many rules of the GT are applied in parallel, which necessitates a mechanism for conflict resolution. An elementary application ends only after the recursive calls of sub-grammars (SGT) or sub-systems (SST) possibly triggered by the application of certain rules have been completed.

A system of interdictions (rules are marked, nodes are blocked) allows one to statically test the ST for decidability : the compiler may warn the user of the risks of undecidability (loops in the GC, "free" mode in an iterative GT, constraint on recursive calls not satisfied, etc.).

The schemas which appear in left-hand sides of rules have a very great expressive power. For each node, it is possible to indicate whether its daughters are to be looked for in leftmost or rightmost positions, in order or out of order (free permutations). It is possible to look for nodes at unspecified depths by using "generalized nodes". Finally, the rules may be context-sensitive, the root of the schema (RS) being possibly different from the root of the effective transformation (RT). What is not dominated by the RT constitutes the context, or "hat". The RT may itself be active or contextual. What it dominates belongs to the active part.

The notion of parallel rewriting in ROBRA is quite strong, as parallelism may be "normal" (RT located on distinct nodes of a cut of the AO), "vertical" (a RT may dominate another), and

"horizontal" (several contextual RT and at most one active RT may be instantiated on the same node of the AO).

Finally, it is possible to write extremely complex conditional assignments of variables in the right-hand side of rules, which contributes to making ROBRA an extremely powerful tool.

ROBRA is really a production system of the substitution type, even if, in the current implementation, elementary application of a transformational grammar is done by transduction of an input tree into an output tree (both represented linearly). For that reason, the decoration type is necessarily preserved by a transformation system.

c.   EXPANS, a language for lexical expansion and transfer

EXPANS is based on a model of transduction of decorated trees [30]. The decoration types of the input and output trees may be different. Each node is transformed into a sub-tree in the output tree. This sub-tree is determined by consultation of the dictionaries, in their order of priority, through the UL born by the node. A default action is always foreseen. A dictionary item has a UL value as key, and a list of triplets <condition/image/assignments> as content. The conditions concern the node of the input tree and possibly its immediate neighbours (mother, left and right sisters). The image describes the geometry of the sub-tree to be produced, and the assignments allow to compute the values of the variables on the nodes of the sub-tree from those of the accessible input nodes.

At the level of a dictionary, if the UL of the node is found, EXPANS looks for the first triplet whose condition is verified (the last condition must be empty, that is, identically true), and the corresponding sub-tree is produced. Otherwise, this dictionary fails. In deterministic mode, dictionaries are searched in their order of priority until a success is obtained. In non-deterministic mode, all dictionaries are searched, in order, and the image produced is a sub-tree constructed by rooting the sub-trees produced by the dictonaries under a new node. This mode allows for example not to "hide" the usual translation of a word which has also a different translation in a particular domain whose dictionary has been given higher priority.

d.   SYGMOR, a language for morphological generation

SYGMOR is based on the model of a finite-state deterministic transducer, whose first version was designed by B. Thouin and programmed by D. Jaeger. [29] describes the extensions and amendments he contributed to it in recent years. SYGMOR takes as input a sequence of decorations and produces as output a string of characters. A context reduced to the current (C) and preceding (P) decorations is available, and two strings are used, the "working string" T ("chaîne de Travail"), and the output buffer S ("chaîne en Sortie").

The grammar has quite a simple structure. Each rule is made of a condition, actions and "subsequent rules". Actions consist essentially of writing to the right, to the left or at the middle (last point of concatenation) of T a literal string or the result of accessing one of the dictionaries through the value of one of the variables of C (dictionaries accessed by the UL give the bases, the others the affixes). It is also possible to modify C, and to "recall" S to T (by concatenating it to the left and emptying it).

For each decoration, SYGMOR looks for the first applicable rule and applies it. It then applies the subsequent rules in order, without taking their subsequent rules into account. A subsequent rule may be obligatory or optional. If an obligatory rule fails, SYGMOR goes back to the initial state and applies the rule MOTINC, if present, and otherwise the default error action (empty S, then do S:=T). Processing continues by taking into account the subsequent rules of the last rule applied, until an empty list of subsequent rules is reached.

e.   TRACOMPL, a language for transforming decorations ("articulations")

TRACOMPL [30] is the sub-language used to write all DV components. It has been made into an autonomous language to allow for writing "articulations". The goal is to transform decorations of a Set1 into a Set2. For that, we proceed in two steps :

-   First, one describes Set2 and what should be known of Set1 in order to perform the transformation. The names of the variables present in both sets are prefixed by "$" and those of variables present in Set1 but not carried over to Set2 by "$$". The others (not prefixed) are considered to be new.

- One completes this by writing (CVAR part) a conditional action which can test the variables of the input decoration in Set1, in Set2 after the "reformatting" described by the preceding part, and in Set2 in their current state (during execution of the action). Because of this, it is possible to perform arbitrary transformations of variables.

B. Evolution of tools and methodology due to experiments in MT-R

The organization presented above seems somewhat complex. Why several languages? Why optional phases? As a matter of fact, the initial architecture was a lot simpler, and almost all extensions have been motivated by express demands of linguists.

What we want to illustrate now is that an "MT shell" such as Ariane-G5, with its powerful rule-based languages, is still not enough: developers taking parts in various experiments in building MT-R prototypical or operational systems have expressed the need for more engineering-oriented tools, supporting the development of independent and static lexical and grammatical knowledge.

1. MT-R systems developed with Ariane-G5

a. The preoperational Russian-French prototype (1980—87)

The development of the Russian-French system [16] from the stage of mockup to that of a real prototype, used in a "preoperational" setting (a flow of texts was regularly sent, translations had to be sent back within a certain time, dictionaries had to be improved together with a specialist of technique and translation) has shown us the necessity of developing tools to help linguistic programming in Ariane, such as ATLAS or VISULEX. Since the beginning of the eighties, it has become evident that this type of programming could and should be compared with the programming of large software systems, and attacked accordingly. As an example, the Ariane-78 software has cost about 30 man years of work, and the Russain-French lingware about 80.

This prototype, realized by N. Nédobejkine and his collaborators, is not strongly specialized for a certain type of texts. Its vocabulary, of about 30,000 terms (simple and compound terms, idioms), contains 5 to 6,000 general terms, the rest being distributed between various scientific and technical domains (space and earth sciences, metallurgy, aeronautics, linguistics…). At the beginning, it was not clear whether MT for the watcher or MT for the revisor was aimed at. It appeared that the quality obtained was good enough to allow for revision of a page in about 1/4 of an hour, a time very much comparable to that of the revision of a human raw translation – of course, the errors are not the same.

b. The analyzers of English and the translation mockups (1983—89)

The methodology for constructing MT systems in Ariane owes much to various cooperations through which B. Vauquois was able to try various methods and come up with the above-mentioned principles. This effort began as soon as 1973-74 with studies on the analysis of French, together with SFB/100 at Saarbrücken (J. Weissenborn, E. Stegentritt), and then of Portugese (P. Daun Fraga). The methods developed for French were taken over and improved for Portugese, and again reused and improved on French, in the context of a feasibility study on French-English for the French Telecoms (Vauquois, Guilbaud, Dymetman) in 81-82.

The analysis of English then became the common point of several studies made in cooperation. The most important led to an English-Malay laboratory prototype [50]. Others led to mockups into Chinese [57], Thai, and finally Arabic [42]. This analysis was also used as the starting point for work on a "standard analyzer" of English undertaken during the MAT-NP and pursued for some time afterwards at B'VITAL.

c. B'VITAL's operational French-English system (Ariane/aero/F-E)

The most comprehensive work on French-English using Ariane has been done by B'VITAL, a small firm founded during the MAT-NP in 85-87. The static grammar of French consists of about 150 "construct boards" ("boards" are 2-dimensional representations of correspondence rules) and 450 "disambiguation boards" (preference rules). AS is more than 20,000 lines long (in ROBRA). The dictionaries contain about 20,000 terms, of which more than half are terminological.

2. Commented examples of French-English translations

The following examples have been selected from raw translations produced by B'VITAL's Ariane/aero/F-E system after three years of development.

Après essai, s'assurer du fonctionnement correct de l'ensemble raccord.	*After test, check that the coupling assembly works correctly.*

Note here the passage from a nominal (prepositional) phrase "du fonctionnement correct" to a verbal clause, "that… works correctly", with as corollary the passage from the adjective "correct" to the adverb "correctly". These transformations are not performed during transfer. It is the first step of syntactic generation which, starting from the "g-structure" (generating structure), considered as underspecified with respect to the syntactic functions and the syntagmatic and morphosyntactic classes, recomputes these levels depending on the initial goal (here, to construct a verbal phrase), using the deeper levels (logical relations inside the strict predicative frame, semantic relations for the circumstancial complements).

Using the notion of lexical unit, the generator knows, without having to consult a dictionary, which lemmas are contained in the considered derivational family, which contains the possible paraphrases. Here, "fonctionnement" has been reduced to the UL "fonctionner-V", itself translated as "work-V", which bears the potentiality of deriving an action noun. It is then simply the order of preference between the rules controlling the choice of syntagmatic categories which triggers the construction of a subordinate clause rather than of a nominal phrase ("the correct working of the coupling assembly").

Porter sur celle-ci la date de la dernière réception ou révision.	*Write on this one the date of the last reception or of service.*

"Porter" is a support verb here, and "porter une date" is translated as "to write a date" and not as "to carry a date", thanks to a test performed at lexical transfer on the syntactic and semantic features of the second argument (ARG1) of "porter" (its logical object).

Effectuer la vidange générale et la purge du carburant (voir chapitre 12).	*Drain in a general manner and bleed fuel (see chapter 12).*

"Effectuer la vidange" is translated here by the simple verb "to drain", thanks to the notion of lexical unit, and to the organization of the lexical transfer. "Vidange" is reduced to "vidanger-V", and that UL gives in translation a (sub)tree in which the possibility of the presence of a support verb such as "effectuer", "faire", etc., is coded. The translation of the support verb, if present, will be erased by structural transfer.

Le bouchon a pour but d'assurer la protection d'un raccord auto-obturable lorsque celui-ci n'est pas utilisé au sol ou en vol.	*The trap is used for carrying out the self-sealing coupling protection when this one is not used at the ground or in flight.*

"Avoir pour but" is recognized as a compound predicate, "avoir-but-V(x0,x1)", translated as "use-V(x1,x0)", with a conversion of argumets. This explains the generation of a passive.

Enduire légèrement le joint neuf de liquide d'utilisation.	*Slightly coat the new joint with operating fluid.*

The translation of prepositions is always delicate. It is necessary to know whether they introduce arguments or circumstantials. Here, "enduire-V" is a predicate with 3 arguments (qn enduit qn/qc de qc), the third one being introduced by "de". The analyzer prefers to fill the argument frame, and the introductor of the corresponding argument of "coat-V" is "with".

Ouvrir progressivement le robinet (3), appliquer une pression jusqu'à 1,5 bar jusqu'à l'allumage du voyant lumineux DS2 et l'extinction du voyant DS1.	*Gradually open tap (3), apply a pressure up to 1,5 bar until the light DS2 switching on ((ignition)) and the signal lamp DS1 extinction.*

Here the preposition "jusqu'à" introduces two circumstantials. What is really translated is the semantic relation (here, RS=LOC with SEM=TEMPS and SLOC=QUA), refined by the preposition and by the semantic features of the "governor" (Vauquois' term for "head") of the group, here PROCESS for "allumer-V", and of the predicate ("appliquer-V").

Ouvrir progressivement le robinet (3) jusqu'à obtenir une pression de 9 bars.	*Gradually open tap (3) until a pressure of 9 bars is obtained.No explicit transformation is performed.*

The infinitive clause is rendered by a subordinate clause simply through the normal functioning of the generator, as explained above. As argument 0 (logical subject) is not expressed, a passive is

generated. This is only a matter of stylistic preference. It would be equally possible to generate "until one obtains a pressure of 9 bars", or "until obtaining…", as in the following example.

Procéder à la dépose des panneaux.	*Remove the panels.*
IMPORTANT : avant de déposer ou de reposer le panneau central intrados de voilure, il est nécessaire de procéder à certaines modifications.	*IMPORTANT : before removing or reinstalling the lower central wing panel, it is necessary to proceed with some modifications.*

Here, the construct preferred for the conjunction "before" is the gerundive. On the other hand, the preposition "à" introduces argument 1. In the m-structure produced by analysis, it may well have been suppressed. "With" is contained in the valency frame of "proceed-V" for the same argument position, hence it is introduced by the generator.

3. Evolution of tools and methodology

a. Tools associated with Ariane-G5

A variety of tools have been implemented besides Ariane-G5 to support the development of large dictionaries (ATLAS, VISULEX, BDTAO, TTEDIT) and to handle texts containing markups and non-roman characters (THAM, LT, SCRIBERE).

i. Helps to construct MT systems

ATLAS, a system to help indexing in coded dictionaries

ATLAS is a language for writing "indexing charts" and has been used to produce numerous indexing manuals of the Russian-French system. One describes an acyclic graph where the internal nodes bear questions, the edges possible answers, and the leaves the results attained (usually, names of formats or of procedures).

This graph may be drawn, to produce paper manuals, or be used dynamically, to create menus on-screen in a window and send the results to the appropriate places in a second window showing the dictionary to be constructed or modified.

VISULEX, a tool for the synthetic visualization of lexical informations contained in lingware written in Ariane

VISULEX [2] allows one to visualize all or part of the lexical informations contained in a system written in Ariane-78 or Ariane-G5, at two levels (codes and comments), which frees linguists from having to search many files at the same time.

BDTAO, a lexical data base management system for MT

BDTAO is a lexical data base management system specifically designed by B'VITAL for MT, but not for Ariane only. Analysis and generation formats and dictionaries are compiled from the corresponding monolingual sub-base. For transfer, there is one sub-base for each language pair.

There is a distinction between the "kernel" dictionary, which belongs to the grammatical system of the language and is directly coded, the general dictionary, and the terminological dictionaries. In transfer, the general part is much more complex than the terminological part, and the construction of the corresponding Ariane dictionaries is not yet fully automated (in the meantime, standard indexing manuals are used to index directly in Ariane TL dictionaries).

TTEDIT, a transformational editor of decorated trees

TTEDIT [25] is a tree editor (trees may bear simple labels or complex decorations). Its originality is that its basic operations are sub-tree transformations, as in ROBRA, and not direct manipulations on the nodes and edges. This allows one to work on large trees, in the same manner as one works on texts with editors equipped with highly parametrized search-and-replace facilities. TTEDIT is completely integrated with the standard editor XEDIT, and also allows one to write "macros", which are in fact transformational grammars analogous to those of ROBRA.

TTEDIT has been in particular used in order to carry out, starting from a tree produced by B'VITAL's analyzer of French and transformed by a transfer written for the occasion (in Ariane), additional *ad hoc* transformations leading to an analysis interface structure conforming to the "linguistic legislation" of the Eurotra project and to the personal preferences of people in charge, for special cases not foreseen by that "legislation" [28].

ii.   Tools for handling texts

<u>THAM</u>

THAM stands for "Machine-Aided Human Translation" ("Traduction Humaine Aidée par la Machine"). Programmed in EXEC2/XEDIT, THAM works as an XEDIT macro [2]. Suppose that an Ariane user is revising a raw machine translation. By simply hitting a key, he starts THAM, which allows him to access a "natural" dictionary, dynamically modifiable, together with the revised text, the raw translation and the source text. THAM may also be used in stand-alone mode.

This tool is by far not a full "translator/revisor workstation", like those offered by certain firms (Weidner, ALPS…), but only a useful extension to XEDIT. As a matter of fact, for industrial use, it is far preferable to revise the translations on a PC, a MacIntosh, or, if luxury is permitted, on a Xerox Star™ machine.

<u>LT, a language for writing transcriptors</u>

LT [36] allows one to quickly write transcriptors of texts. For example, the Russian-French system uses a transcription of the Russian texts in a subset of PL/I character set (upper case roman letters, digits and some special signs). Transcriptors written in LT are used to output texts in cyrillic and Chinese scripts while a Roman transcription is preferred within Ariane-G5. A recent implementation in CLOS has been used to transform a large dictionary between various formats [26]

The abstract model here is a finite-state transducer with two tapes. The input tape has two reading heads. The first one can only go forward, while the second one, used as a "look-ahead", may go forward or be "recalled" to the position of the first one. The output tape has one writing head. The states are structured : a full state is the combination of an elementary state and of a set of values of a certain number of variables.

Writing a transcriptor consists of declaring the variables and their types (e.g. font, language, length of the look-ahead…), and then in describing the graph of the transition system, with one node for each elementary state. The edges bear the transitions, which are classical production rules. There are a number of predefined functions, as well as facilities for parametrizing and factorizing, to avoid writing too many rules.

<u>SCRIBERE, an extension to the SCRIPT/DCF formatter</u>

SCRIBERE allows one to describe the textual content and the logical structure of a document by using macros written in IBM's SCRIPT/DCF. This tool offers possibilities inspired by SGML. It has been developed by D. Bachut and N. Verastegui to compensate for the unavailability of GML and SGML at the time. Markup takes into account certain aspects of linguistic nature (current language, current transcription, etc.).

b.   Static <u>grammars</u> <u>and</u> <u>lexical</u> <u>data</u> <u>bases</u>

The need for a "static" specification of "dynamic" analysis and generation grammars appeared around 1980 because of cooperation with linguists having no computational background. [22] and [52] present the formal and practical aspects of the first "static grammar" formalism. A methodology for specifying and implementing analyzers and generators from static grammars was then defined and tested during the ESOPE project of ADI on the development of a pedagogical English-French mockup (BEX-FEX), and put to full use for the development of B'VITAL's Ariane/aero/F-E system. In a nutshell, a static grammar describes the correspondence between texts and associated linguistic structures in a non-directional way. Terminals and non-terminals of static rules stand for subcorrespondences. Although the rewriting mechanism is classical (context-free with attribute and constraints), the structures so defined can be "natural" abstract trees, possibly strongly non-projective [19].

Static grammars were first prepared on paper. A first computer environment to handle them was then implemented on a Macintosh by integrating an array of commercially available tools [56]. In the following years, several studies have been made to refine their formal semantics [19, 37, 58] and some improvements have been proposed. At USM, Y. Zaharin and his team have produced SaGE, a Macintosh application which allows one to compute on a static grammar. This is not yet an executable specification, but SaGE can automatically produce an analyzer and a generator (in ROBRA) from such a grammar, in simple cases, following the general strategic principles explained above. This work is preliminary (in particular, the transformational systems produced are far from being optimized), but quite encouraging.

Finally, the lexical methodology is evolving towards application-independent lexical data bases. We have mentioned B'VITAL's development of BDTAO, which is specific to MT, but system-independent. This need is quite concrete: even now, translators' aids integrated with access to MT systems cannot offer coherent MT and MAT dictionaries. We have undertaken several studies on this topic since 1984 [17], thereby prototyping a trilingual base in telecommunications (French-English-Japanese) in 1986 and participating in the Multilex Esprit project in 91-93. In [3, 46], we have proposed and prototyped solutions to the problems remaining at the level of the definition of the linguistic content of such bases, of their logical structure, and of their implementation.

II.   Evolution towards personal networking communication: LIDIA and UNL

Since 1989, we have turned to the investigation of DBMT (Dialogue-Based MT), a new architecture for MT systems enabling monolingual users to generate high-quality translations into several languages. The quality should be good enough to require no postediting or a minimal postediting in the case of very important documents. Fully automatic MT systems can achieve this goal only for very specific kinds of texts, such as weather bulletins.

In our approach, the document is first interactively "cleaned" and "tagged" (for special terms or idioms, utterance styles, etc.). Each unit of translation is then sent to a state-of-the art all-paths analyzer, which returns a structure factorizing all ambiguities which cannot be reliably solved with the knowledge available. A question mark appears next to each ambiguous unit of translation. The user clicks on it to activate a "disambiguation dialogue" which doesn't assume any particular expertise in linguistics, computer science, or any of the target languages.

This research is now pursued in the context of the UNL project, where input is planned to be either (1) directly constructed in the intermediate UNL pivot language with some specialized editor, or (2) written normally and then interactively disambiguated into UNL structures as in LIDIA.

We first describe where we were at the beginning of the UNL project, after several years of building and experimenting with our LIDIA mockup, both in the design of this new paradigm and in the extension of Ariane-G5 to Ariane-G5/LIDIA. In the last section, we show how this evolution continues in the framework of the UNL project.

A.   The LIDIA project

After explaining the motivations and the main principles of our DBMT approach, we illustrate the concept with an example, and detail the main evolution of the basic software which has been motivated by this new MT paradigm.

1.   Motivations and principles of the DBMT paradigm for "MT-A"

a.   Historical background and motivations

The idea of DBMT has been proposed and tested in various forms during the last 20 years [21, 31-33, 39, 45, 47, 48, 53-55]. However, it has always been taken for granted that the user should be a specialist, linguist or translator, or at least a professional, and that, consequently, the system could and should be specialized. In contrast, we think that DBMT systems should be designed for the general public, and be used on PCs. Consequently, the design of the user interface in general, and of the disambiguation dialogues in particular, becomes extremely important.

The main idea of our concept is that pieces of the text under creation or modification are sent to an analyzer running in the background. If there are ambiguities, be they proper to the source language or relative to the translation into one or several target languages, questions are posed to the author, in the source language. The resulting ambiguity-free structures are sent to transfers and generators into all target languages, producing high quality translations needing no postediting.

During the last few years, we have designed and implemented LIDIA-1 [4, 5, 8-13] as a series of mockups to experiment with this concept of DBMT for (non-specialist) individual authors. In LIDIA-1.2, there is only one source language, French, and three target languages, German, English and Russian, but this is only due to limitations in manpower. The addition of Chinese as a fourth target language is under way, and should be included in LIDIA-1.3, hopefully the last mockup before a full-size prototype.

b.   Main principles of the approach

Here are the essential principles which, if taken together, lead to a new "paradigm", although almost all of them have already been proposed or used in other combinations:

- *distributed processing.* The document is created and interactively disambiguated on a middle-range Macintosh, while the various processes of MT proper are performed by a remote server.

- *application to hypertexts.* The documents are in effect hyperdocuments, in the form of HyperCard stacks. Units of translation are HyperCard textual "fields".

- *asynchronous and non-preemptive processing.* Units of translation are "released" by the author, and then autonomously travel to the MT server, come back after analysis in a "multiple-multilevel-concrete" form (MMC structure), announce the presence of ambiguities by letting a button appear next to them, react to the click by engaging in a disambiguation dialogue, and then, once disambiguated, travel again autonomously to the MT server to be translated, and to finally be inserted in the appropriate field in the target stack.

- *e-mail communication between component processes.* We have switched from a specialized connection to the use of standard e-mail for all communications between the author workstations and the MT server, which can now be located anywhere in the world.

- *deeper multilevel approach.* We have added a level of "interlingual acceptions" (or word senses) to the classical lexical levels of B.Vauquois' multilevel transfer approach ("occurrence" or wordform, "lemma" or citation form, and "lexical unit" or derivational family).

- *guided language approach.* Rather than to force users into a single controlled language, we associate a "document kind" to the whole document, a "texte genre" to large parts (possibly defined as an SGML document structure, with linguistic attributes), and an "utterance style" to each utterance. In an orthogonal way, the lexicon may be specialized by adapting weights ("lexical preferences") on arcs and nodes of the graph representing the lexical data base.

- *disambiguation strategy.* We have developed a generator of disambiguation dialogues which non-specialists can easily understand, and which does not rely on excessively sophisticated linguistic processing, so that the disambiguator can run in real time on the author's personal computer.

- *control by reverse translation.* On demand, the system translates back from the target UMA structures (unambiguous, multilevel, abstract), providing feed-back through a paraphrase in the source language of the translation.

- *homogeneity of knowledge sources.* In the current state of implementation, this concerns only lexical knowledge: both lexical disambiguation messages and MT dictionaries are obtained from the same multilingual lexical data base, PARAX [3], itself implemented in HyperCard.

In real usage, the source texts could be in any format. We are working on tools to transform them into the same kind of HyperCard stacks as the above demo stacks, thereby structuring and segmenting them, identifying the "utterance styles" ("typing") and indirectly preediting text-specific "formulas" or "idioms" in a semi-automatic fashion.

c.   Concrete vs abstract linguistic structures

Let us clarify our notions of "concrete" and "abstract" linguistic structures. A "concrete" representation of a text is such that the corresponding text can be recovered from it by using a standard traversal algorithm and simple morphological and graphemic generation rules. Familiar examples are textbook constituent structures and dependency structures (with left-to-right traversal of the leaves or infix traversal of all nodes). Otherwise, the representation is "abstract".

Note that the information contained in both kinds of structures (on labels and other more or less complex annotations) may be of the same linguistic "depth": there may be "deep" concrete structures and "surface" abstract structures, in this sense, although the opposite is of course more frequent. They may also be "multilevel", which is our preferred strategy.

Take for example the sentence: *"The customers were not given their money back by the cashier, but by the waiter."* A "multilevel" head-driven concrete structure could be:

```
S[type=assertive, time=past, aspect=perfective, tense=c-past, voice=passive…]
  (NP[semrel=dest, logrel=arg2, synfunc=subj, sem=human, num=plur…]
     (Art[lex='the', semrel=deict, synfunc=det, number=plur, deter=definite…]
      Noun[lex='customer', synfunc=head, sem=human, number=plur…])
   aux[lex='be', tense=pret, pers=3, number=plur…]
   neg[lex='not']
   vrb[lex='give', synfunc=head, voice=passive, tense=ppart, vbpart='back'…]
   NP[semrel=patient, logrel=arg1, synfunc=obj1, number=sing…]
      (adjposs[lex='his', semrel=poss, synfunc=det, number=plur, deter=definite…]
       Noun[lex='money', synfunc=head, number=sing…])
   vbpart[lex='back']
   NP[semrel=agent, logrel=arg0, synfunc=agcomp, number=sing, neg=not-but…]
      (prep[lex='by', synfunc=reg]
       art[lex='the', semrel=deict, synfunc=det, number=sing, deter=definite…]
       Noun[lex='cashier', synfunc=head, sem=human, number=sing, neg=not…]
       NP[semrel=id, logrel=arg0, synfunc=coord, number=sing…]
          (conj[lex='but', synfunc=reg]
           prep[lex='by', synfunc=reg]
           art[lex='the', semrel=deict, synfunc=det, number=sing, deter=definite…]
           Noun[lex='waiter', synfunc=head, sem=human, number=sing…]))
   punct[lex='.'])
```

Syntactic categories have been used here as the main labels, with phrases (syntagmas) in capitals and preterminals in small letters. Acronyms should be self-explanatory.

In an abstract structure, some lexical information would be "featurized", and order could be normalized, leading to:

```
S[type=assertive, time=past, aspect=perfective, tense=c-past, voice=passive…]
  (vrb[lex='give'.'back', synfunc=head, voice=passive, tense=c-past…]
   NP[semrel=agent, logrel=arg0, synfunc=agcomp, num=sing, neg=not-but…]
      (neg[lex='not']
       Noun[lex='cashier', synfunc=head, sem=human, number=sing, deter=definite…]
       NP[semrel=id, logrel=arg0, synfunc=coord, num=sing…]
          (conj[lex='but', synfunc=reg]
           Noun[lex='waiter', synfunc=head, sem=human, number=sing, deter=definite…]))
   NP[semrel=patient, logrel=arg1, synfunc=obj1, num=sing…]
      (adjposs[lex='his', semrel=poss, synfunc=det, number=plur, deter=definite…]
       Noun[lex='money', synfunc=head, number=sing…])
   NP[semrel=dest, logrel=arg2, synfunc=subj, sem=human, number=plur…]
      (Noun[lex='customer', synfunc=head, sem=human, number=plur, deter=definite…]))
```

Abstract representations of utterances are far superior to concrete representations as input and output structures of transfers in semantic transfer MT or as "lexical-conceptual structures" [38] in interlingual MT, especially between distant languages. But their relation to the corresponding utterances is not as clear, a natural consequence of abstraction. This "remoteness" is even more apparent with other types of structures, such as conceptual graphs, logical formulae or interlingual representations *à la* KBMT-89 [43].

By contrast, concrete structures are clearly more adequate for interactive disambiguation. They are also superior for a variety of future applications. For example, no text processor today is able to replace "give-back" by "return" in the preceding example, not to speak of changing the modality, the tense, the voice, or the discourse type (say, from direct to indirect or affirmative to negative). Self-explaining documents would make such changes possible.

2. Presentation on an example

The LIDIA-1.2 mockup demonstrates the concept on a HyperCard stack showing typical kinds of ambiguities in French. The target languages are Russian, German and English. The MT server runs on an IBM-9221 minicomputer accessed in the background through a standard e-mail facility.

Ariane-G5 itself has remained stable, but has been extended by a translation server accessible by e-mail and http (Ariane-G5/LIDIA), and by a tool allowing one to develop Ariane-G5 MT systems from a Macintosh running Hypercard and Eudora, a simple e-mail utility.

a. All-paths analysis and interactive disambiguation

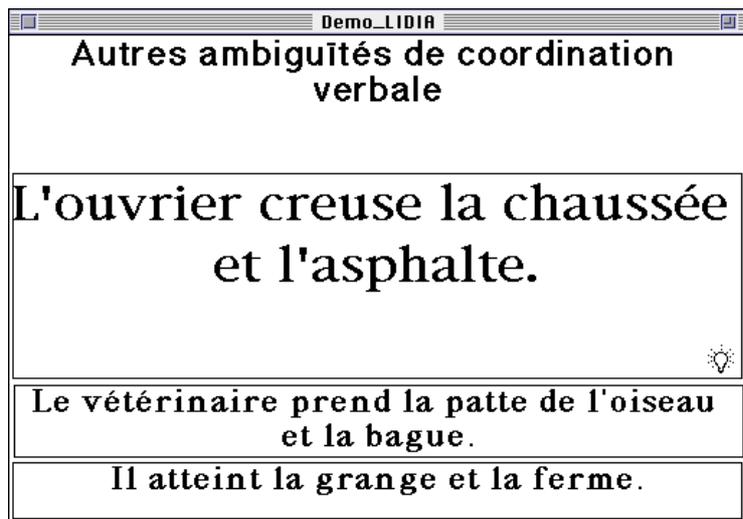Here is a card of a demonstration HyperCard "stack".
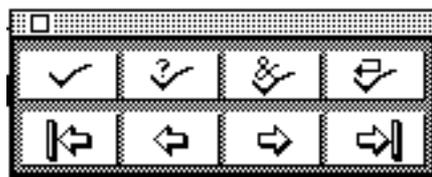

Fig. 1 : A demo LIDIA stack

Fig. 2 : LIDIA Palette

Let us open a card in the demo stack and click on the second field, which contains the sentence :

« L'ouvrier creuse la chaussée et l'asphalte. » [The worker bores a hole into the roadway and asphalts it / The worker bores a hole into the roadway and the asphalt.]

A button (bulb-shaped here) appears next to the field to show that the sentence has been sent to the remote server to be analyzed. When the analysis comes back, another button bearing a question mark appears. Clicking on it with the LIDIA cursor triggers a disambiguation dialogue. Note that the interactive disambiguation step is not meant to replace robust automatic disambiguation. It comes after it to guarantee correct disambiguation of the remaining ambiguities.

The analysis used here produces what we call an "MMC-structure", that is, a multiple, multilevel, concrete structure, actually a large decorated tree containing subtrees, each corresponding to possibly more than one "proper" analysis (disjunctions can appear in the node decorations, which are complex attribute structures).

The analysis dictionaries of LIDIA-1.2 are quite large (about 20,000 lemmas). In particular, they contain all of the verbs in the famous "Bescherelle" dictionary. However, to date only about 2,000 have been indexed to enable transfer into other languages through "interlingual acceptions". The grammatical coverage is not as large, but not at all limited to the small corpus used in the mockup.

The ambiguities produced are of two types, those relative to the source language, and those which appear relative to at least one of the target languages. In our current strategy, we begin with questions on structural ambiguities, continue with functional ambiguities (subject/object, argument-/circumstancial, semantic roles -- none in this example), and finish with the lexical ambiguities.
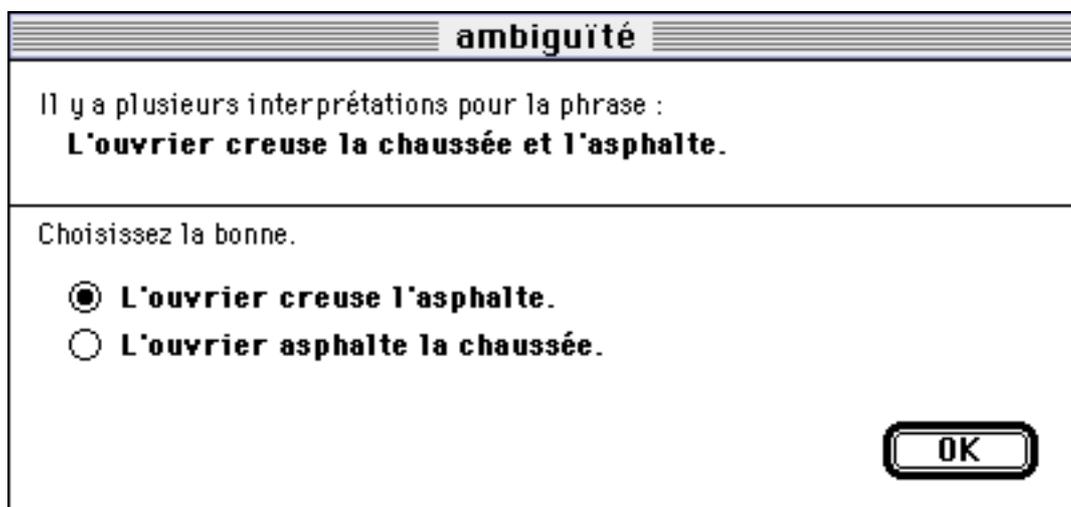

Fig. 3 : A structural disambiguation dialogue

Here, « asphalte » can be a noun or a verb. This lexical ambiguity is also manifested by a structural ambiguity of attachment. According to our strategy, structural ambiguity schemata are activated first, so that we get the following dialogue.

Fig. 4 : A lexical disambiguation dialogue (creuser)

The two meanings here are "to bore a hole through something" and "to make something hollow by boring matter out of it".

Fig. 5 : A lexical disambiguation dialogue (chaussée)

"Chaussée" might be a road, a dam, an underwater reef, or a row of basaltic columns. After disambiguation is finished, the resulting UMC-structure (unique, multilevel, concrete) is automatically sent to the server, where it is submitted to an "abstraction" process, the last phase of analysis, and transformed into a "UMA-structure" (unique, multilevel, abstract), which is better suited to translation into various languages (see below for more details).

Abstraction consists of simplifying and normalizing the geometry of the tree by suppressing nodes corresponding to articles and auxiliaries and coding the corresponding information in the decorations, in regrouping discontinuous constituents, and possibly in changing the linear ordering into a standard one.

b. Transfer and generation

i. German

The lexical transfer of this mockup has only 250 French lexical units (UL), i.e., about 750 lemmas, and 550 German UL (≈ 1,600 lemmas). The structural generation has been developed specifically for this project, to obtain a clear division into two phases.
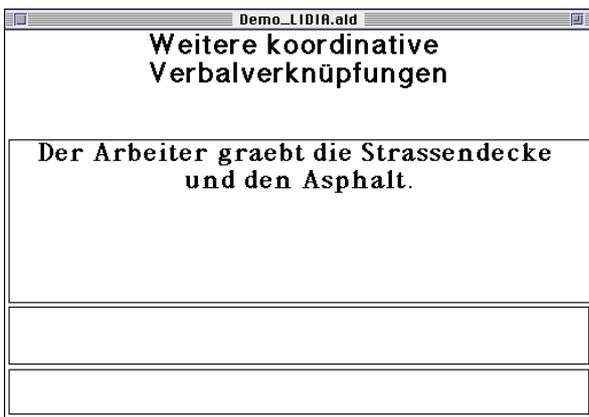
```
┌──────────────── Demo_LIDIA.ald ────────────────┐
│                                                 │
│          Weitere koordinative                   │
│          Verbalverknüpfungen                    │
│                                                 │
│  ─────────────────────────────────────────────  │
│   Der Arbeiter graebt die Strassendecke         │
│            und den Asphalt.                      │
│                                                 │
│                                                 │
│                                                 │
│  ─────────────────────────────────────────────  │
│                                                 │
│  ─────────────────────────────────────────────  │
│                                                 │
└─────────────────────────────────────────────────┘
```

Fig. 6 : Translation into German

ii.   English

```
┌──────────────── Demo_LIDIA.ang ────────────────┐
│ Other Verbal Coordination Ambiguïties           │
│                                                 │
│                                                 │
│ The worker bores a hole into the roadway        │
│          and into the asphalt.                  │
│                                                 │
│                                                 │
│                                                 │
│  ─────────────────────────────────────────────  │
│  ─────────────────────────────────────────────  │
└─────────────────────────────────────────────────┘
```

Fig. 7 : Translation into English

iii.   Russian

```
┌──────────────── Demo_LIDIA.rus ────────────────┐
│   Другие двусмысленности глагольного            │
│              сочинения                          │
│                                                 │
│  ─────────────────────────────────────────────  │
│  Рабочий копает (роет) дорогу и асфальт.        │
│                                                 │
│                                                 │
│                                                 │
│  ─────────────────────────────────────────────  │
│  ─────────────────────────────────────────────  │
└─────────────────────────────────────────────────┘
```

Fig. 8 : Translation into Russian

First, the target "GMA structure" (generating, multilevel, abstract) produced by the transfer is transformed into a UMA structure, by recomputing or confirming the surface-level attributes from the interlingual ones. Second, a UMC structure is obtained via the following syntactic generation phase.

The morphological generation has 1,755 UL (about 5,000 lemmas) and has been obtained by inverting the morphological analysis of a previous project on German-French.

When the result of the translation comes back from the server, it is automatically inserted in the same place in the image German stack (created by LIDIA the first time it is needed).

The transfer is an adaptation of the French-German transfer.

The generation adapts that of a previous feasibility study on French-English.

The figures concerning dictionaries are the same as for German, except that the morphological genaration has less ULs (850 UL, or about 2,500 lemmas).

Here is the result of the translation corresponding to the same answers in the disambiguation dialogue. Again, it is automatically placed in the image English target stack.

Transfer and generation have been obtained by inversing the transfer and analysis phases of the large Russian-French system developed between 1972 and 1987. As for the other languages, transfer dictionaries (using interlingual acceptions, which are difficult to code) have been limited to what is necessary for the mockup.

By contrast, the morphological generation has 12,000 UL (about 30,000 lemmas corresponding to 440,000 inflected and accentuated forms).

As Russian is produced in a special Roman translitteration on the MT server, it is transcribed into cyrillic script when received by LIDIA on the Macintosh.
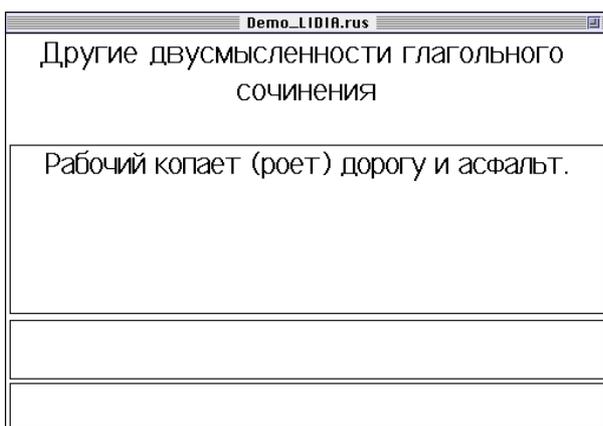
3.   Perspectives for DBMT

a.   Discussion of alternatives

Is DBMT really a better approach than other alternatives? In our opinion, yes, because: (1) very often, these alternatives are not really feasible; (2) the results can be intrinsically better, if presented as self-explaining documents.

First, *machine aids for translators* [1, 40, 41, 49] are usable only if there are available and affordable translators, and if they have enough time to do the job. But, in many situations, there are simply no such translators, especially if translations are required in several languages. For

example, multinational firms, banks, etc., have many unmet translation needs. In Europe, scientists and engineers are engaged in many projects where communication is hampered by the language barrier.

Even if competent translators are available, the delays are often such that translation is impossible. This is for example the case in European institutions, which are theoretically required to issue all important documents in all official languages, but are unable to do so, although they employ more than 1,200 full-time translators, and translate more than 1.2M pages a year. Further, the final versions of documents are too often ready at the last moment. Here, it would make more sense to analyze and disambiguate their parts as soon as they are ready, and to translate them at the last moment.

Another possibility, often advocated, is to write in *controlled languages,* designed to be unambiguous, and use "black box" MT. This can be very successful, in restricted situations, as in the case of the TITUS system [24]. But it is very difficult to force people to write in a controlled language. Controlled languages are also difficult to design, and very task- and domain-specific.

Finally, controlled languages are unambiguous for the analyzer designed to process them, *but not for humans.* While this may be convenient for man-machine communication, it may be counterproductive, or even dangerous, for human communication. If, for example, "to replace (a mechanical part)" is meant to mean only "to replace with a new part" ("remplacer"), and not "to put back in place" ("replacer"), a mechanic may well understand the second, unintended meaning, and put back into place an airplane part which should be replaced by a new one, leading to an accident.

If the concept of self-explaining documents (see below) can be made to work in broad contexts, texts could be written without any restrictions beyond the usual ones, which concern style and terminology, and at the same time actually be less ambiguous than texts written in controlled languages. Even if translation is not an issue, the availability of *"text explainers"* could prove to be a major advance in document processing technology.

b.   A concept for the future: self-explaining documents

i.   Problems with reverse translations

In the course of our experimentation, we have (again) observed that translation introduces ambiguities which are not present in the source text. (*Traduttore, traditore* !) It also happens that all disambiguated analyses of a sentence produce the same translation, which is as ambiguous as the original. One example was the translation from French into Russian of the famous sentence «The man sees the girl in the park with a telescope». It also happens that reverse translations seem as ambiguous as the original. In the case where they are the same as the original, the author is even more confused, because it is not clear whether the system has done anything!

In this case, goes the objection, what is the use of disambiguating the source text if ambiguities reappear in the translation(s), or -- even worse -- if new ones are created, and if reverse translation does not help to detect these problems? Would it not be better to try to produce translations which preserve the ambiguities, and dispense with interactive disambiguation altogether?

Unfortunately, the experience of human translation shows that ambiguities can be *exactly* preserved only in some cases, and that to preserve ambiguities purposefully is quite difficult and often leads to unnatural expressions in the translation. It is also quite clear that the "transferable" ambiguities vary with the target language. Finally, although some texts may be intentionnally ambiguous, especially in poetry and politics, we take it that the vast majority of ambiguities are not intentional, but are rather due to the intrinsic nature of natural languages. Of course, some authors write more clearly than others, yet all authors write unambiguously only in programming languages, unambiguous by construction, but ambiguously in natural languages, ambiguous by nature!

ii.   The concept of self-explaining documents

This observation has led to the idea of *self-explaining documents:* if the target documents are accompanied by their (unambiguous) linguistic structure, with indications of potentially ambiguous parts, so that if the reader in the target language can obtain a clarification of unclear parts in a user-friendly way, the objection disappears. As human users are notably insensitive to ambiguities, however, we should find a way to warn the reader that the target text is ambiguous.

In a multilingual DBMT setting, there is a very simple way to perform this task. The system analyzes the target text using the target language analyzer, and gets the corresponding MMC-

structures. It then runs the disambiguation dialogue on the target side in automatic "mute" mode -- by having the system itself answer each question so that the accompanying structure is contained in the selected subset at each point, and records questions and answers. It is then possible to show the presence of ambiguities by any convenient means, e.g. by creating buttons which the reader may click to obtain the clarification *which would have been given by the author himself, had the text been written in the target language!* To simplify this process, the accompanying structure should then be unambiguous and "concrete".

iii. Overall schema of an architecture for producing parallel self-explaining documents

Here is a functional diagram of the processes we have discussed above. In GMA-structures (generating, multilevel and abstract), non-interlingual linguistic levels are underspecified, and (if present) are used only as reflections of corresponding surface levels in the source language, and are recomputed in the first generation phase, which we call "paraphrase choice".
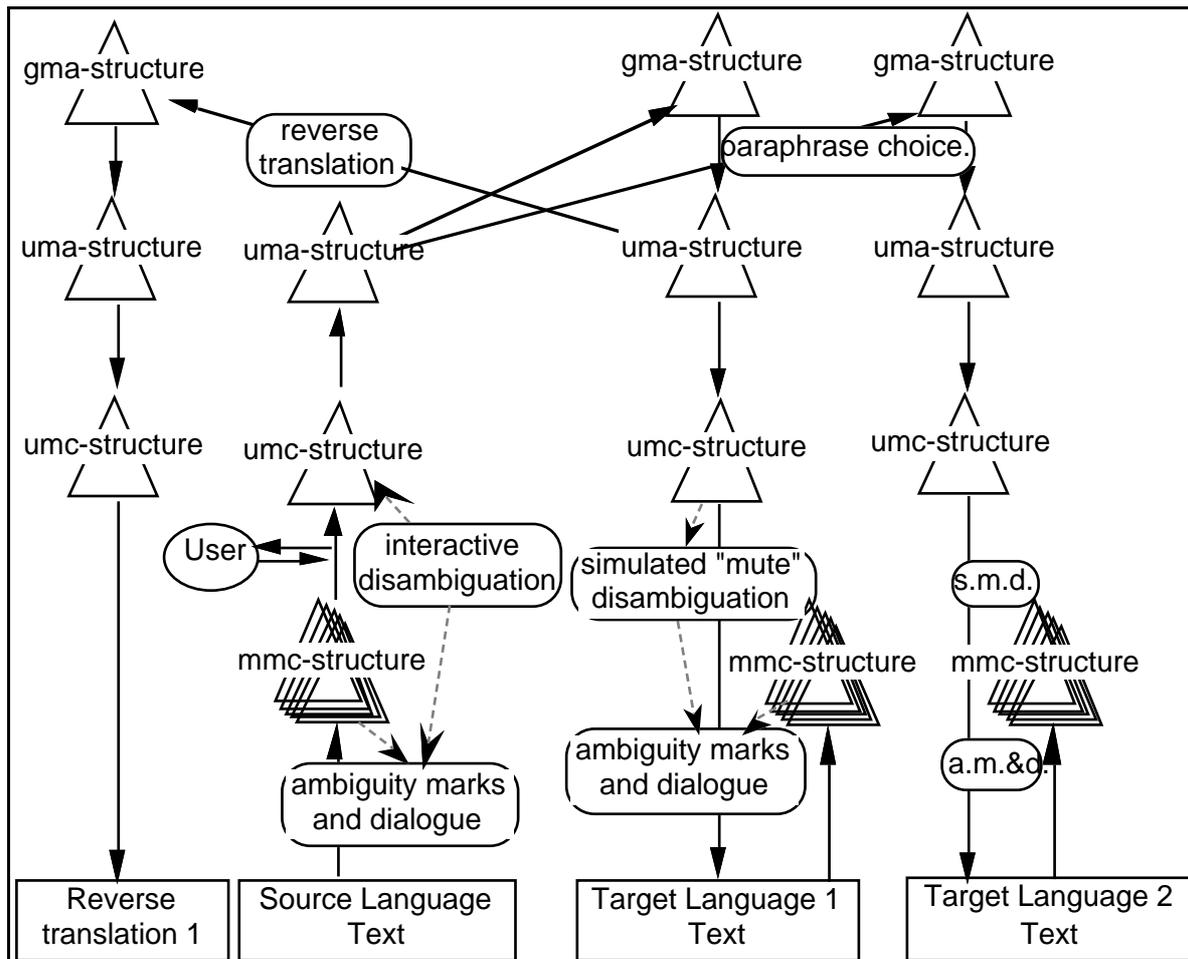
Fig. 9 : Diagram for producing parallel self-explaining documents with DBMT

c. Potential and upscaling problems

Full-scale, general DBMT systems "for everyone" will require extremely large grammatical and lexical knowledge bases. To cover a whole language, a lexical data base should contain on the order of 3 million terms, corresponding to 4 to 5 million monolingual acceptions, perhaps twice as many interlingual acceptions for systems designed to handle 10 to 20 languages.

The development costs are staggering, and probably out of reach if conventional lingware engineering techniques are used. For instance, it has cost EDR (Tokyo) about 1,200 man years to develop 300K terms in 2 languages (200K terminological and 100K general), with the associated 640K interlingual concepts (200K terminological and 2*300K general, minus 60K common). At least 100 times more (1.2M man years!) would be needed for 3 million terms in 20 languages. This is why we advocate a step-by-step approach, along with the development of new groupware techniques for developing very large lexical data bases.

Even if cost were not a problem, developing extremely large lexical data bases and keeping them up to date would be impossible using only professional teams of lexicographers. The quantity is too huge, changes and innovations are too fast, and only specialists can be competent in specific domains. It is a major challenge to design *groupware lexical data base development techniques* based on the *contribution of lexical information by users* of existing text explainers or DBMT systems.

A distributed architecture would be very advantageous for this purpose, because lexical information created by users on their personal computers might transparently and automatically be sent to servers. For example, authors might add new senses to existing terms, add new terms, and propose translations in the languages they know. Professional teams would then process and refine this "raw lexical material" on server sites. This idea is not so far-fetched, and very similar to that used in Eurolang Optimizer™ [1], a distributed environment designed for professional translators.

B.   Evolution: Ariane-G5/LIDIA

Our basic software has been considerably extended for this project. While Ariane-G5 itself has remained stable, an Ariane-G5 server has been developed by P. Guillaume.

A LIDIA user simply sends messages to Ariane-G5 by e-mail. Here is an example, for the sentence "Le vétérinaire prend la patte de l'oiseau et la bague" [the veterinarian takes the leg of the bird and the ring/and rings it].

```
/*--SEPARATEUR--*/
stack_name Demo_LIDIA
card_id 8446
object_id 2
object_type cdfl
visible T
treatment analysis
language français
/*--SEPARATEUR--*/
LGS = FVX
TLS = TY0
LLC = ()
TYTR = A
/*--SEPARATEUR--*/
*LE VE!1TE!1RINAIRE PREND LA PATTE DE L' OISEAU ET LA BAGUE .
/*--SEPARATEUR--*/
```

Here is the message returned with the MMC-structure.

```
/*--SEPARATEUR--*/
======================================================================
Received: from imag.imag.fr by durian.imag.fr (IBM VM SMTP V2R2) with TCP;
   Fri, 31 Mar 95 18:23:49 WDT
…
To: lidia17@durian.imag.fr
From: lidia@warna.imag.fr (lidia)
Subject: ana_C8446_O2_Tcdfl

/*--SEPARATEUR--*/
stack_name Demo_LIDIA
card_id 8446
object_id 2
object_type cdfl
visible T
treatment analysis
language français
/*--SEPARATEUR--*/
LGS = FVX
TLS = TY0
LLC = ()
TYTR = I
/*--SEPARATEUR--*/
*LE VE!1TE!1RINAIRE PREND LA PATTE DE L' OISEAU ET LA BAGUE .

/*--SEPARATEUR--*/
1: 'ULTXT' (2: 'ULFRA' (3: 'ULSOL' (4: 'PHVB' (5: 'DGN' (6: 'LE', 7: 'VE!1TE!1RINAIRE'), 8:
'NV' (9: 'PRENDRE'), 10: 'DGN' (11: 'LE', 12: 'PATTE', 13: 'GP' (14: 'DE', 15: 'LE', 16:
'OISEAU'), 17: 'DGN' (18: 'ET', 19: 'LE', 20: 'BAGUE')), 21: '.')), 22: 'ULSOL' (23: 'PHVB'
(24: 'DGN' (25: 'LE', 26: 'VE!1TE!1RINAIRE'), 27: 'NV' (28: 'PRENDRE'), 29: 'DGN' (30: 'LE',
```

```
31: 'PATTE', 32: 'GP' (33: 'DE', 34: 'LE', 35: 'OISEAU')), 36: 'PHVB' (37: 'ET', 38: 'GPRON'
(39: 'PATTE'), 40: 'NV' (41: 'BAGUER')), 42: '.')))))
```

— end of the geometric part, beginning of algebraic part (content of "decorations") —

```
1 '': UL('ULTXT').
              2 '': UL('ULFRA').
                3 '': UL('ULSOL'),MODIF(NON).
                  4 'PREND':
UL('PHVB'),LINKS(OUI),RECHTS(OUI),ENONCP(DECL),K(PHVB),SUBV(VF),VALET(Q),VOIX(ACT),
                    ARGS(A0,A1),CAT(V),MT(IPR),NBR(SING),PERS(3),PHASE(NONACC).
                  5 'VE!1TE!1RINAIRE':
UL('DGN'),FS(SUJ),K(GN),RL(ARG0),SUBN(NC),VALET(N),CAT(N,D),GNR(MAS),
NBR(SING),PERS(3).
                          6 '*LE': UL('LE'),FS(DES),CAT(D),GNR(MAS),NBR(SING),TYPOG(PCAP).
                          7 'VE!1TE!1RINAIRE':
UL('VE!1TE!1RINAIRE'),FLAG(CPRED)*,FS(GOV),CAT(N),GNR(MAS),NBR(SING),PERS(3),SEMN(ANIME),SENS(
1),SSANIM(HUM).
                  8 'PREND':
UL('NV'),LINKS(OUI),RECHTS(OUI),ENONCP(DECL),K(PHVB),SUBV(VF),VALET(Q),VOIX(ACT),ARGS(A0,A1),C
AT(V),MT(IPR),NBR(SING),PERS(3),PHASE(NONACC).
                    9 'PREND':
UL('PRENDRE'),FLAG(CPRED),AUX(AVOIR),FS(GOV),VAL0(N),VAL1(N),CAT(V),NBR(SING),PERS(3),SEM1(CON
CRET),SEMZ(ANIME),SENS(1,2,4),SSCONC1(MAT).
                    10 'PATTE':
UL('DGN'),RECHTS(OUI),VCOOR(OUI),FS(OBJ),K(GN),RL(ARG1),SUBN(NC),VALET(N),CAT(N,D),GNR(FEM),NB
R(SING),PERS(3).
                          11 'LA': UL('LE'),FS(DES),CAT(D),GNR(FEM),NBR(SING).
                    12 'PATTE':
UL('PATTE'),FLAG(CPRED),FS(GOV),CAT(N),GNR(FEM),NBR(SING),SEMN(ANIME,CONCRET),SENS(1,2),SSANIM
(ANIMAL).
                    13 'OISEAU':
UL('GP'),FS(COMP),K(GN),SUBN(NC),VALET(DEN),CAT(N,D,S),GNR(MAS),NBR(SING),PERS(3).
                          14 'DE': UL('DE'),FS(REG),CAT(S).
                          15 'L''': UL('LE'),FS(DES),CAT(D),GNR(MAS),NBR(SING).
                          16 'OISEAU':
UL('OISEAU'),FLAG(CPRED),FS(GOV),CAT(N),GNR(MAS),NBR(SING),SEMN(ANIME),SENS(1),
SSANIM(ANIMAL).
                    17 'BAGUE':
UL('DGN'),FS(COORD),K(GN),RL(ID),SUBN(NC),VALET(N),CAT(N,D),GNR(FEM),NBR(SING),PERS(3).
                          18 'ET': UL('ET'),FS(REG),CAT(C).
                          19 'LA': UL('LE'),FS(DES),CAT(D),GNR(FEM),NBR(SING).
                          20 'BAGUE':
UL('BAGUE'),FLAG(CPRED),FS(GOV),CAT(N),GNR(FEM),NBR(SING),SEMN(CONCRET),SENS(1,2).
                          21 '.': UL('.'),CAT(P),PONC(DR).
… and so on for the second geometrical structure (dominated by 'ULSOL').
/*--SEPARATEUR--*/
```

On the Ariane-G5 side, however, things are more complex. A whole network of virtual machines takes care of the various parts of the processing (analysis into MMC-structure, abstraction from UMC-structure into UMA-structure, transfers and generations into various target languages).

More recently, P. Guillaume has also developed a WWW front end for Ariane-G5: knowing the "language codes" and using the appropriate character set and transcription, it is possible to run an MT application on a text from any machine connected to the Web.

C.   Using Ariane-G5/LIDIA within UNL

1.   UNL

Our LIDIA-1.2 mockup has shown that Dialogue-Based MT is a technically feasible and promising paradigm for many translation situations. Its distributed architecture has been successfully tested from distant locations. However, it is necessary to build and experiment with full-size prototypes to be sure that upscaling this technology is economically and technically viable. This is one of the reasons why we have welcomed the opportunity to participate in the UNL project. Other motivations are the very international and highly multilingual character of the project, its emphasis on producing a working system, and its linguistic architecture.

UNL is a project of multilingual personal networking communication initiated by the University of United Nations based in Tokyo. The pivot paradigm is used: the representation of an utterance in the UNL interlingua (UNL stands for "Universal Networking Language") is a hypergraph where nodes bear UWs ("Universal Words", or interlingual acceptions) with semantic attributes, and arcs

bear semantic relations (deep cases, such as agt, obj, goal, etc:). Going from a text to the corresponding "UNL text" or interactivley constructing a UNL text is called "enconversion", while producing a text from a sequence of UNL graphs is called "deconversion".

This departure from the standard terms of analysis and generation is used to stress that this is not a classical MT project, but that UNL is planned to be the source format preferred for representing textual information in the envisaged multilingual network environment. The schedule of the project, beginning with deconversion rather than enconversion, also reflects that difference.

13 languages are tackled during the first 3-year phase of the project, while many more are to be added in the second phase. Each group is free to reuse its own software tools and/or lingware resources, or to develop directly with UNL-provided tools.

Emphasis is on a very large lexical coverage, so that all groups will presumably spend most of their time on the UNL-NL lexicons, and develop tools and methods for efficient lexical development.

By contrast, grammars can be initially limited to those necessary for deconversion, and then gradually expanded to allow for more naturalness in formultating text to be enconverted.

2.   Architecture of deconversion and enconversion

Deconversion using Ariane-G5 will occur in three steps:

1) Transformation of UNL graphs into Ariane-G5 UNL trees. A parametrizable tool is being implemented.

2) UNL-French transfer.

3) French generation.

Symmetrically, enconversion using Ariane-G5/LIDIA will occur in 3 steps:

1) French analysis and interactive disambiguation.

2) French-UNL transfer.

3) Transformation of Ariane-G5 UNL trees into UNL graphs.

Ariane-G5 UNL trees are UMA-structures with their usual geometry, and nodes containing UNL information in known variables, such as UNLRS for the UNL deep case, UL for the UW, UNLREF for the references (coindexing), and UNLATTR for the attributes.

3.   Lexical engineering for UNL-FR

The lexical work is determined by the content of the central UNL database. At the moment, it contains about 95765 English "headwords", with nearly 220000 associated UWs. The task is to produce:

1) the corresponding French-UNL data.

2) the corresponding Ariane-G5 dictionaries for French analysis and generation, and transfers between UNL and French.

3) the dictionary for interactive disambiguation (when enconversion is tackled).

The method used to attain this goal is as follows:

1) transform the UNL files into a collection of CLOS objects corresponding to the headwords.

2) enrich these objects by French equivalents and associated information found in available dictionaries such as the USM-MAE-GETA French-English-Malay dictionary, BDTAO, and various MT-oriented dictionaries developed under Ariane-G5 or by other cooperating groups.

3) transform them into a collection of Word files, where each paragraph corresponds to a particular field and has a corresponding style, and each entry contains dummy paragraphs for all obligatory fields.

4) distribute the Word files to lexicographs. The disquettes come with macros which provide online help in the form of menus for all fixed lists (categories, domains, possible next styles, etc.) and of verification utilities.

5) transform the resulting files back into CLOS objects, and store them in a CLOS-based multilingual database organized as PARAX [3], i.e., as a central structured collection of UWs and linked with monolingual bases (at least English and French).

6) enrich the CLOS base by generating the Ariane-G5 related information.

7) mirror this CLOS base using a commercial DBMS system such as 4D, to be used for server functions.

8) produce all necessary dictionaries.

9) iterate the process, ideally in an incremental way: while the CLOS base should be the center of the lexicologic work, the server should be the source of distribution of all lexicographic work, but no lexicographic work should be directly performed on it.

Steps 1-4 are already in progress. Steps 5-8 are under way. No doubt much will be learned in this huge effort.

III. Evolution towards speech translation: integration within C-STAR II

A.   Task and general methodology

In Speech Translation (ST) as in text translation, there are several possible kinds of translation situations which lead to very different requirements. High-quality output might be required for the dissemination of crucial information, but would be obtainable only through severe restrictions on the input and possibly require a high degree of user interaction. On the other hand, on-line aids for human interpreters might be built with simple word-spotting techniques coupled with dictionary access and user-adaptative filtering. We have recently started to investigate ST in the context of the CSTAR project, which lies somewhere between these two extremes.

1.   The task

In the context of C-STAR II, the task is to translate human-to-human tourism-related dialogues (booking a room, buying seats for a show, etc.). As in human interpretation, translation need not be perfect by any means. Because the two interlocutors know about the subject matter and share a common goal, it is enough to produce approximate and possibly incomplete translations, provided that some feedback is given about what has been or will be translated.

The main task of our CLIPS++[5] group within C-STAR II is to integrate French in the architecture and experiments of the consortium. This implies building a translation system from spoken French into at least one target language or into the task-specific interchange format (IF), building or integrating a speech syntheziser for French, and generating French from the IF, because no partner plans to develop translations into French. We intend to produce IF representations as well as full translations into German, Italian, and perhaps English, so that the partners for these languages will also be able to generate them from our IF structure.

The demonstrators under construction will conform to a scenario where a client converses with his travel agent and 2 or 3 foreign travel agents speaking different languages, each dialogue being translated so that all interlocutors can follow all dialogues.

An interesting aspect due to the constraints of the project is that it will be possible to compare different approaches to MT for exactly the same task, namely an interlingua approach (IF) and two transfer approaches, a "GB-based" transfer approach[6] using concrete trees at LATL, and a multilevel transfer approach using abstract trees at GETA.

2.   The approach taken

The language at hand is quite restricted and can be handled as a sublanguage, with heuristic techniques. However, the speech recognition phase may deliver several linguistically plausible utterance candidates. Hence, the approach to analysis is to produce one representation for each such utterance candidate.

---

[5] CLIPS++ consists of GETA and GEOD, two research groups of CLIPS (Grenoble) specializing in Machine Translation, Speech Recognition and Dialogue Processing, LATL (Geneva), also working on MT, and LAIP (Lausanne, speech synthesis).

[6] GB for "Government and Binding"

Although there is almost no room for interactive disambiguation in the Machine Interpretation of Human Dialogues, it seems both necessary and possible to follow the CMU technique and ask the user one question before sending a translation: "Which is the meaning you want me to translate"? Hence, the UMA-structures produced by the analyzer for one utterance should be generated as standard text and proposed to the user in a menu.

In order to better integrate speech recognition and language analysis, we want to start from a lexical lattice (see [18] for a more detailed discussion). If we can develop at some point a speech recognizer incorporating a powerful language model, the lattice will probably contain full or partial trees. But all MT systems, including Ariane-G5, expect a string of characters as input. How can we solve this problem? In the first prototype, the output lattice contains only phonetic words, and we simply transform it into the list of its possible traversals. In the future, we plan to develop a GCFG (Generalized Context-Free) tool for more complex lattices.

The input to Ariane-G5 is thus a normal text, where each sentence corresponds to one possible traversal of the lattice, and is a sequence of phonetic words. After morphological analysis, the utterance is represented by a tree dominated by a 'ULTEXT' node, and subtrees dominated by 'ULFRA' nodes corresponding to each traversal. From then on, analysis is done in parallel on these subtrees, leading to subtrees coding UMA-structures. In the process, some subtrees may disappear because the corresponding sequence of phonetic words cannot be analyzed as a full utterance. It is also possible to erase only the "incorrect" parts and keep the "correct" parts.

B.   A first prototype

We have developed an end-to-end first prototype in a few months. The goals were to:

1) integrate the components developed by the various members,

2) produce a complete prototype showing 2 target languages (German and English),

3) demonstrate French synthesis, without having yet developed generators from the IF,

4) use telecommunication links in at least a part of the demo,

5) use lattices as interfaces between speech recognition and machine translation.

1.   Architecture

The hardware architecture of the demonstrator is shown in the figure below.

ECHO
– speech recognition
ITS2
– French Phon to English Text
DECTALK
– English synthesis

ftp Phon lattice

MCL
– Phon lattice to HTTP requests
– Requests management
– User interface
LAIPTTS, MBROLA, MACSOUND
– French synthesis

http French Phon to French text
http French Phon to German text

ARIANE G5 lingware
– French Phon to French Text
– French Phon to German Text
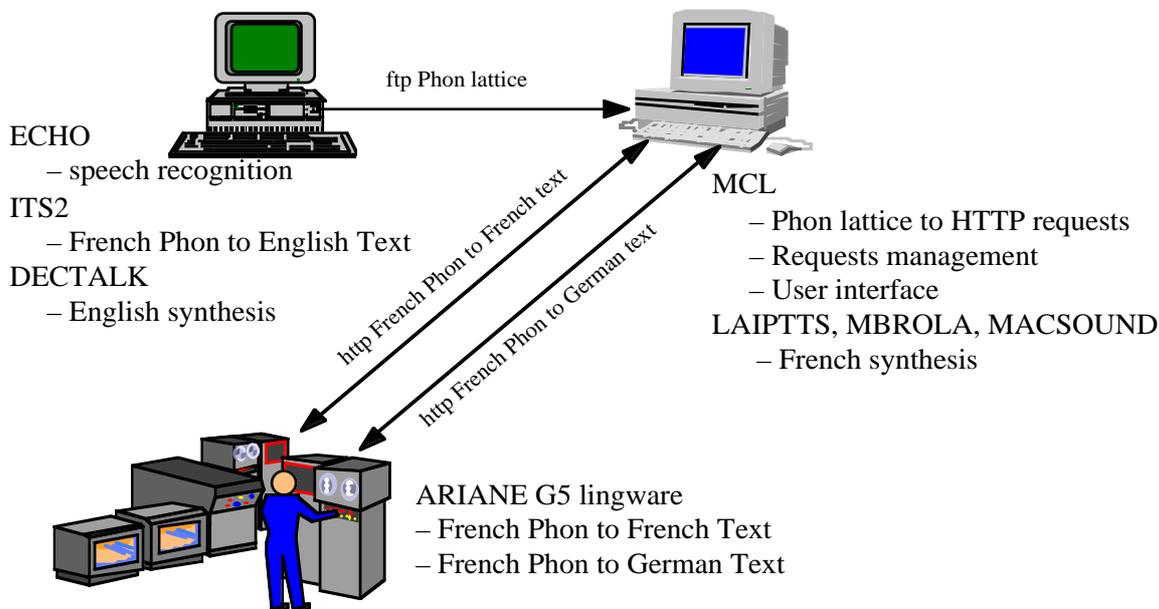
a.   Shared speech recognition

The speech recognizer used in this prototype is an adaptation of the ECHO (Environnement et Communication Homme-Ordinateur) system built by the GEOD group. ECHO is a connected speech recognition system which uses the Hidden Markov Models approach. The system has been orignally implemented on UNIX platforms and then adapted for PC computers. It has a C/C++

language interface (as Dynamic Link Libraries) for developers and an high level interface for end users. From the speech signal, the new ECHO/CSTAR version outputs a phonetic word lattice.

At this point, two paths are followed, the first one to English and the second one to German. The first path is executed on the PC, while the second goes to a Macintosh which calls a remote server.

In each case, the MT lingware produces one or more French orthographic renderings of each possible "trajectory" in the lattice, by a kind of "reverse translation" (phonetic French into written French), and the user can choose "what will be translated".

b.   Path to English

The ITS-II French analyzer has been considerably reconfigured to accept the phonetic output.

It has then been connected to the existing generator of French and to the existing transfer and English generator.

c.   Path to German

The lattice produced by the speech recognition component is sent by ftp (file transfer protocol) to a Macintosh which handles the data and schedules the production of the expected results.

HTTP requests are sent from the Macintosh to the translation server to get:

– the speech to text "translation" of the phonetized input, and

– the German translation of the French Phonetic input.

Translation into German and into French text are adaptations of existing Ariane-G5 lingware. Note that the interface presented below requires that the two resulting texts are strictly parallel.

2.   Work done to adapt the components

a.   Data collection

1) We have created about 400 dialogues, each of about 10 turns, in the domain of hotel reservation. Writers were instructed to follow the general pattern given by 4 example dialogues, and to stick to some negotiation concerning the number and category of the room(s), the dates, the prices, and the location of the hotel.

2) These dialogues have been processed to extract the relevant vocabulary (about 800 lemmas) and the most frequent expressions (such as "Bonjour Madame!").

b.   Speech recognition

1) A restricted sublanguage has then been built, with restricted syntax and vocabulary. For reasons of speed on a PC, only 180 lemmas (900 wordforms) have been retained, and expressions quite natural in speech.

2) This vocabulary has then been learned from several speakers with a Markov model of words.

3) ECHO has then been adapted to this sublanguage. ECHO is a markovian system with word models, controlled by a finite-state automaton language model.

4) In order to prepare for the extension to speaker independence, the 400 dialogues have been read by several speakers, recorded with two microphones, and acquired on machine in stereo.

5) Labelling of this data has begun, in order to automatically learn optimal acoustic units (semi-syllables).

c.   Machine translation with Ariane-G5

1) We have first built a specific morphological analyzer using the phonetic transcription as input.

2) We have then implemented very quickly a first version of the analyzer, geared towards processing of three dialogues, using the technique of "translation memory". This version has been used to integrate the first end-to-end prototype.

3) This analyzer has then been connected to the available French generator and French-German transfer and generator, thereby indexing the necessary vocabulary.

4) We have then built a second version of the analyzer, which uses a memory of short, slightly parametrized fragments ("je voudrais", "N chambres", "pour N nuits", etc.) and a language model expressed as a finite-state automaton on these fragments. This can handle a considerably greater variety of turns.

No German speech synthesizer has been integrated for this demo, but it is possible to send the selected German translation to any machine running a German speech synthesizer via the Internet. Orthographic French is pronounced by the LAIPTTS synthesizer.

d.   Machine translation with ITS-II

The ITS-2 MT system developed at LATL (E. Wehrli) runs on PC machines under Windows 95 or Windows NT. The French-English version is an adaptation of the standard ITS-II F-E lingware.

It receives from the speech recognizer a phonetic representation of the spoken sentence (a lattice), where some phonemes may not have been completely disambiguated ("archiphonemes"). The linguistic engine takes the linearized phonetic representations as input and attempts to parse them, producing abstract syntactic representations for the "correct" ones and discarding the others.

An example of the various levels is given in (1).

(1)   a. j [è-é]m^r[è-é] yn /âbr av[è-é]k du/ pur trwa jur.

                     b. [TP [DP j'] aimerais [VP [DP une [NP chambre [PP avec [NP douche]]]]

                        [PP pour [DP trois [NP jours]]]]]

                     c. [TP [DP I] would [VP like [DP a [NP room [PP with [NP shower]]]]

                        [PP for [DP three [DP days]]]]]

                     d. I would like a room with shower for three days.

(1a) corresponds to the phonetic input with unresolved ambiguities noted with square brackets, and alternatives separated with a dash symbol (e.g. [è-é]).

(1b) is the syntactic skeleton of the GB-structure produced by the parser. It is also the structure used by the transfer component.

(1c) is the corresponding target GB-structure produced by the translator.

Finally, (1d) is the target sentence, which is passed to the DECTalk speech synthesizer.

If the output of the speech recognizer is ambiguous, the possible interpretations are regenerated into orthographic French so that the user can choose which one will be translated. Orthographic French was pronounced by the Lernhout & Houspie synthesizer because the PC version of LAIPTTS was not yet available.

e.   Speech synthesis with LAIPTSS

LAIPTTS [34, 35] is a high-quality text-to-speech system for French built by LAIP at Lausanne. Currently, it uses the Mons (Belgium) MBROLA diphone output system to produce audible output. The system consists of a 350K application (without interface) and uses a 2 Mb dictionary as well as a 4.8 Mb diphone database.

Like most text-to-speech systems, LAIPTTS is structured into four main modules. The first module takes written text and generates an annotated phonetic chain of each sentence on the basis of a dictionary and graphemo-phonetic rules. The chain is parsed into prosodic groups, and various phonological rules (liaison, chaining, schwa-handling, syllabification) are applied.

In the next module, durations and F0 values are generated. These values are combined with diphone segments in the subsequent module, after which the entire signal is reproduced. Calculations for the second and following sentences in a text are performed in parallel with sound output of the preceding sentence. This provides real-time synthesis performance for unrestricted text on most high-entry personal computers (PowerPC, Pentium-level).

3.   Integration

a.   Prolem & approach

From the start, it has been clear that a distributed architecture was necessary. We have in fact integrated seven components on three machines using three different OSs:

- ECHO (speech recognizer) on a PC,

- ITS-II (phonetic French to orthographic French and to orthographic English) on a PC,

- Ariane-G5 (phonetic French to orthographic French and German) on an IBM 9221-130,

- LAIPTTS (French speech synthesizer) on a Macintosh,

- Lernhout & Hauspie (French synthesizer) on a PC,

- User interface and components coordinator (on a Macintosh),

- Web utilities (in MCL on Macintosh and C on IBM-9221).

1) We have first defined the phonetic transcription input to the MT systems, and the detailed format of the data to be exchanged. This comprises the exact syntax of the lattices, including comments.

2) We have then worked on the communication mechanisms, arriving at a solution in which the PC sends lattices to the Mac by ftp, while the Mac communicates with the IBM through http.

3) We have used the "whiteboard architecture" to do the integration at the logical level.

At the logical level, we do not care where the components run, or under which operating system. The only thing we want is fast and reliable communication methods for exchanging data, be they mailboxes, sockets, http or other. Components may be running in distant locations.

The main idea of the "whiteboard architecture" is that a "coordinator" maintains a possibly partial image of the essential data structures manipulated by the components, and acts like an orchestra "conductor", synchronizing the work of the components according to its private strategy. We have been working on that architecture in cooperation with ATR [18], and have tested and improved it in two different forms in the LIDIA project. Its main advantage over other architectures (blackboard, communicating agents, etc.) is its simplicity: components never communicate directly, and never have to adapt to changes in a common data structure.

       b.   Ariane-G5/CSTAR and a demo of the "German path"

Let us now detail the components of the "German" path. The scheduling component has been written in Macintosh Common Lisp (MCL). It is made of several processes.

Permanent processes

- fetch-lattice-process: this process regularly scans a CSTAR-in-mailbox where any lattice produced by the speech regognition component is dropped from the PC (by ftp). When a new lattice file is found, it transforms it into an Ariane-G5 "text", where a paragraph corresponds to one trajectory, and may contain several sentences if the "archiphonemes" (such as E/ for é or è) present in the input give rise to several possible "phonetic sentences", given the current dictionary of phonetic forms prepared from the list of active lemmas.

- process-lattice-process: this process scans regularly the requests queue, and whenever an object is found, treatment requests (Phonetic French to French Text and German translation) are launched as processes. These processes are called http-ariane-requests.

| Processes					
Commands					**Resample**
Name	State	Priority	Idle	% Utilization	
process-lattice-process	waiting	0	2.18s	0.1	
fetch-lattice-process	waiting	0	2.20s	0.4	
HTTP Connection Scavenger	Scavenge Wait	0	3.70h	0.0	
Listener	Input	0	0.68s	0.9	
Initial	Suspended	1	0.00s	96.5	

Temporary processes

     ariane-http-request: there is one such process per pending request. For each lattice, two http requests are sent to the Ariane server, one to get the French orthographic rendering(s)

and one to get the German translation(s). The server answers by sending back a stamp. The process then checks regularly if the result is available, using the stamp to identify its request.

cstar-request: when a result is given to one of the two requests involved in the treatment of one turn, a cstar-request process is created. When both results have been delivered by the Ariane-G5 server, a dialogue is proposed to the speaker. This process is ephemeral.

A dialogue is then produced to ask the user to choose the proper French orthographic rendering, if several have been produced by the recognizer or the analyzer.
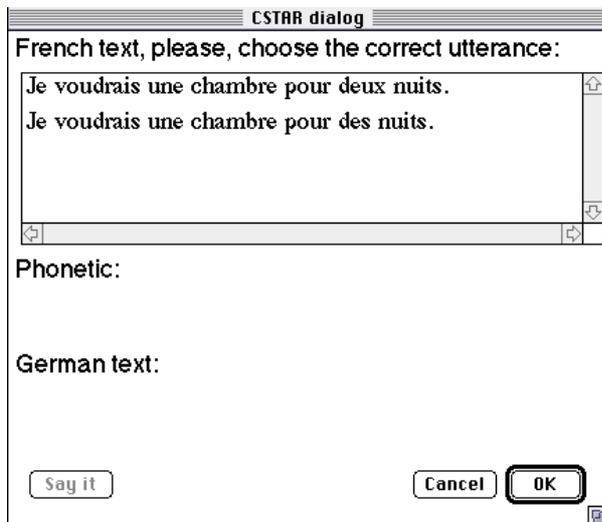
Pushing the "Say it" button invokes LAIPTTS, which pronounces the selected French orthographic rendering.
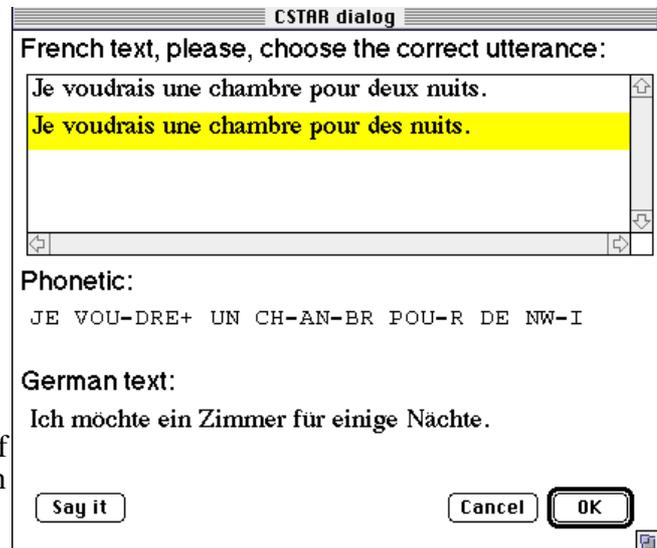
When an item is selected, the phonetic form of the input and the German translation appear in the lower part of the window.

C.   Further evolutions envisaged

1.   Building and passing information between utterances

We have studied how to cope with ellipsis and anaphora across turns, and have come up with the following method:

- store in our "whiteboard" (on the Mac) the memory of possible candidates,
- add special markers at the beginning of the text sent to Ariane for each turn (after morphological analysis, this will give rise to a special "sentence" subtree),
- arrange during analysis that the output French text contains the appropriate record, attached to each possible orthographic rendering of the turn.

- iterate.

2.   More integration between SR and MT

If we have time during the project, or perhaps only afterward, we would also like to develop a GCFG-based tool to handle lattices directly. Substantial work has been described in detail in the literature, notably by P. Quinton (1980, KEAL system) and M. Tomita, so that the implementation effort should be reasonable. It would however be necessary to adapt the analyzer to the new tool, usually a heavy task.

Conclusion

We have described the computer tools and linguistic methods developed at Grenoble during many years for building MT systems for revisors. As they are not embodiments of a particular theory, they are quite easy to adapt to new problems. In the last few years, they have in fact been revised and further developed in the framework of new research on high quality MT for monolingual authors (MT-A), relying on a disambiguation dialogue with the author (DBMT), following an all-paths analysis.

The evolution of Ariane-G5/LIDIA and associated linguistic methodologies is now motivated by two projects of different aims and requirements. The most important points of current developments are the construction of a large lexical database from which coherent dictionaries for MT and for interactive disambiguation will be generated, and the development of robust tools for the integration of heterogeneous speech translation systems using the "whiteboard" approach.

For the future, it will be important to design and implement specialized languages enabling linguists to work directly on complex lattices, with two complementary aims: (1) to better integrate the SR and MT parts of ST systems; and (2) to make it possible to ignore ambiguities most of the time, but to handle them directly at particular points in the overall process.

Acknowledgments

References

[1]   **Bachut D. (1994)** *Le projet EUROLANG : une nouvelle perspective pour les outils d'aide à la traduction.* Proc. TALN-94, journées du PRC-CHM, 7—8 avril 1994, Univ. de Marseille.

[2]   **Bachut D. & Verastegui N. (1984)** *Software tools for the environment of a computer-aided translation system.* Proc. COLING-84, Stanford, ACL, pp. 330—334.

[3]   **Blanc É., Sérasset G. & Tchéou F. (1994)** *Designing an Acception-Based Multilingual Lexical Data Base under HyperCard: PARAX.* Research Report, GETA, IMAG (UJF & CNRS), Aug. 1994, 10 p.

[4]   **Blanchon H. (1992)** *A Solution to the Problem of Interactive Disambiguation.* Proc. COLING-92, Nantes, 23-28 July 1992, vol. 4/4, pp. 1233-1238.

[5]   **Blanchon H. (1994)** *Perspectives of DBMT for monolingual authors on the basis of LIDIA-1, an implemented mockup.* Proc. 15th International Conference on Computational Linguistics, COLING-94, 5-9 Aug. 1994, vol. 1/2, pp. 115—119.

[6]   **Boitet C., Réd. (1982)** *"DSE-1"— Le point sur ARIANE-78 début 1982.* Contrat ADI/CAP-Sogeti/Champollion (3 vol.), GETA, Grenoble, février 1982, 400 p.

[7]   **Boitet C. (1986)** *The French National MT-Project: technical organization and translation results of CALLIOPE-AERO.* Computers and Translation, 1, pp. 281—309.

[8]   **Boitet C. (1989)** *Motivation and Architecture of the LIDIA Project.* Proc. MTS-II (MT Summit), Munich, 16-18 août 1989, pp. 50—54.

[9]   **Boitet C. (1990)** *Towards Personal MT : on some aspects of the LIDIA project.* Proc. COLING-90, Helsinki, 20-25 août 1990, ACL, vol. 3/3, pp. 30-35.

[10] **Boitet C. (1993)** *La TAO comme technologie scientifique : le cas de la TA fondée sur le dialogue.* In "Études et Recherches en Traductique", A. Clas & P. Bouillon, ed., Presses de l'Université de Montréal, Montréal, pp. 109—148.

[11] **Boitet C. (1994)** *Dialogue-Based Machine Translation and Sub-Languages.* Proc. ICLA-94, 26-28 July 1994, USM, 14 p.

[12] **Boitet C. & Blanchon H. (1993)** *Dialogue-Based MT for Monolingual Authors and the LIDIA project.* Proc. NLPRS'93 (Natural Language Processing Rim Symposium, 6-7/12/93, Kyushu Institute of Technology, pp. 208—222.

[13] **Boitet C. & Blanchon H. (1994)** *Multilingual Dialogue-Based MT for monolingual authors: the LIDIA project and a first mockup.* Machine Translation, **9**/2, pp. 99—132.

[14] **Boitet C., Guillaume P. & Quézel-Ambrunaz M. (1978)** *Manipulation d'arborescences et parallélisme: le système ROBRA.* Proc. COLING-78, Bergen, August 1978, 12 p.

[15] **Boitet C., Guillaume P. & Quézel-Ambrunaz M. (1985)** *A case study in software evolution: from ARIANE-78 to ARIANE-85.* Proc. TMI-85 (Conf. on theoretical and methodological issues in Machine Translation of natural languages), Colgate Univ., Hamilton, N.Y., pp. 27-58.

[16] **Boitet C. & Nédobejkine N. (1981)** *Recent developments in Russian-French Machine Translation at Grenoble.* Linguistics, Linguistics, 19, pp. 199—271.

[17] **Boitet C. & Nédobejkine N. (1986)** *Toward integrated dictionaries for M(a)T: motivations and linguistic organization.* Proc. COLING-86, Bonn, IKS, pp. 423—428.

[18] **Boitet C. & Seligman M. (1994)** *The "Whiteboard" Architecture: a way to integrate heterogeneous components of NLP systems.* Proc. COLING-94, 5—9 August 1994, pp. 243—246, 7 p.

[19] **Boitet C. & Zaharin Y. (1988)** *Representation trees and string-tree correspondences.* Proc. COLING-88, Budapest, 22–27 Aug. 1988, ACL, pp. 59—64.

[20] **Chandioux J. & Guérard M.-F. (1981)** *METEO: un système à l'épreuve du temps.* META (Journal des Traducteurs), 1, pp. 17—22.

[21] **Chandler B., Holden N., Horsfall H., Pollard E. & McGee Wood M. (1987)** *N-tran Final Report.* Alvey Project, 87/9, CCL/UMIST, Manchester.

[22] **Chappuy S. (1983)** *Formalisation de la description des niveaux d'interprétation des langues naturelles.* Thèse de 3ème cycle, UJF, Grenoble, juin 1983, 120 p.

[23] **Chauché J. (1975)** *Les langages ATEF et CETA.* AJCL, AJCL (American Journal of Computational Linguistics), microfiche 17, pp. 21-39.

[24] **Ducrot J.-M. (1988)** *Le système TITUS IV.* In "Traduction Assistée par Ordinateur. Actes du séminaire international sur la TAO et dossiers complémentaires", A. Abbou, ed., Observatoire des Industries de la Langue (OFIL), Paris, mars 1988, pp. 55—71.

[25] **Durand J.-C. (1988)** *TTEDIT : Un éditeur transformationnel d'arbres.* Nouvelle Thèse, UJF (Grenoble 1), mars 1988, 120 p.

[26] **Gaschler J. & Lafourcade M. (1994)** *Manipulating human-oriented dictionaries with very simple tools.* Proc. 15th International Conference on Computational Linguistics, COLING-94, 5-9 Aug. 1994, 4 p.

[27] **Gerber R. & Boitet C. (1985)** *On the design of expert systems grafted on MT systems.* Proc. TMI-85 (Conf. on Theoretical and Methodological Issues in Machine Translation of natural language), Hamilton, N.Y., Colgate Univ., pp. 116-134.

[28] **Guilbaud J.-P. (1988)** *Réalisation en ARIANE d'un transfert des structures produites par l'analyseur du français de B'VITAL vers des structures interfaces IS EUROTRA.* Eurotra contract report, GETA, IMAG, juin 1988, 205 p.

[29] **Guillaume P. (1989)** *Ariane-G5 - Extensions apportées au langage SYGMOR.* GETA, IMAG, janvier 1989, 6 p. (Ariane-G5 version 3)

[30] **Guillaume P. (1989)** *Ariane-G5 - Les langages spécialisés TRACOMPL et EXPANS.* GETA, IMAG, septembre 1989, 93 p. (Ariane-G5 version 3)

[31] **Huang X. M. (1990)** *A Machine Translation System for the Target Language Inexpert.* Proc. COLING-90, Helsinki, 20-25 Aug. 1990, ACL, vol. 3/3, pp. 364-367.

[32] **Kay M. (1973)** *The MIND system.* In "Courant Computer Science Symposium 8: Natural Language Processing", R. Rustin, ed., Algorithmics Press, Inc., New York, pp. 155-188.

[33] **Kay M. (1980)** *The Proper Place of Men and Machines in Language Translation.* Research Report, CSL-80-11, Xerox, Palo Alto Research Center, Oct. 1980.

[34] **Keller E. (1997)** *Simplification of TTS architecture vs. operational quality.* Proc. Eurospeech '97. (in press)

[35] **Keller É. & Zellner B. (1996)** *Output requirements for a High-Quality Speech Synthesis System: The Case of Disambiguation.* Proc. MIDDIM-96 Seminar, Le Col de Porte, 12—14 August 1996, pp. 300-308.

[36] **Lepage Y. (1986)** *A language for transcriptions.* Proc. COLING-86, 1986, IKS, vol. 1/1, pp. 402—404.

[37] **Lepage Y. (1989)** *Grammaires Correspondancielles d'Identification.* Thèse, UJF, Grenoble, juin 1989, 184 p.

[38] **Levin L. & Nirenburg S. (1994)** *The Correct Place of Lexical Semantics in Interlingual MT.* Proc. 15th International Conference on Computational Linguistics, COLING-94, 5-9 Aug. 1994, vol. 1/2, pp. 349—355.

[39] **Maruyama H., Watanabe H. & Ogino S. (1990)** *An Interactive Japanese Parser for Machine Translation.* Proc. COLING-90, Helsinki, 20-25 août 1990, ACL, vol. 2/3, pp. 257-262.

[40] **Melby A. K. (1981)** *Translators and Machines - Can they cooperate ?* META, **26**/1, pp. 23-34.

[41] **Melby A. K. (1982)** *Multi-Level Translation Aids in a Distributed System.* Proc. COLING-82, Prague, 5-10 juillet 1982, vol. 1/2, pp. 215-220.

[42] **Moneimne W. (1989)** *TAO vers l'arabe. Spécification d'une génération standard de l'arabe. Réalisation d'un prototype anglais-arabe à partir d'un analyseur existant.* Thèse, UJF, Grenoble, juin 1989, 159 p. (+ annexes)

[43] **Nirenburg S. (1989)** *Knowledge-based Machine Translation.* Machine Translation, 4, pp. 5-24.

[44] **Quézel-Ambrunaz M. (1990)** *Ariane-G5 v.3 - Le moniteur.* GETA, IMAG, juin 1990, 206 p.

[45] **Sadler V. (1989)** *Working with analogical semantics : Disambiguation technics in DLT.* T. Witkam, ed., Distributed Language Translation (BSO/Research), Floris Publications, Dordrecht, Holland, 256 p.

[46] **Sérasset G. (1994)** *Interlingual Lexical Organisation for Multilingual Lexical Databases.* Proc. 15th International Conference on Computational Linguistics, COLING-94, 5-9 Aug. 1994, 6 p.

[47] **Somers H. L., Tsujii J.-I. & Jones D. (1990)** *Machine Translation without a source text.* Proc. COLING-90, 20-25 Aug. 1990, ACL, vol. 3/3, pp. 271-276.

[48] **Tomita M. (1986)** *Sentence Disambiguation by asking.* Computers and Translation, **1**/1, pp. 39-51.

[49] **Tong L. C. (1987)** *The Engineering of a Translator Workstation.* Computers and Translation, **2**/4, pp. 263—273.

[50] **Tong L. T. (1986)** *English-Malay Translation System: A Laboratory Prototype.* Proc. COLING-86, Aug. 1988, IKS, pp. 639—642.

[51] **Vauquois B. & Boitet C. (1985)** *Automated translation at Grenoble University.* Computational Linguistics, **11**/1, January-March 85, pp. 28—36.

[52] **Vauquois B. & Chappuy S. (1985)** *Static grammars: a formalism for the description of linguistic models.* Proc. TMI-85 (Conf. on theoretical and metholodogical issues in the Machine Translation of natural languages), Aug. 1985, pp. 298-322.

[53] **Wehrli E. (1992)** *The IPS System.* Proc. COLING-92, Nantes, 23-28 July 1992, vol. 3/4, pp. 870-874.

[54] **Whitelock P. J., Wood M. M., Chandler B. J., Holden N. & Horsfall H. J. (1986)** *Strategies for Interactive Machine translation : the experience and implications of the UMIST Japanese project.* Proc. COLING-86, Bonn, 25-29 août 1986, IKS, pp. 25-29.

[55] **Wood M. M. G. & Chandler B. (1988)** *Machine Translation For Monolinguals.* Proc. COLING-88, Budapest, 22-27 Aug. 1988, pp. 760—763.

[56] **Yan Y. F. (1987)** *Vers une ingénierie de la production de linguiciels. Spécification et réalisation d'un prototype de poste de travail linguistique pour la spécification de correspondances structurales.* Thèse, UJF, Grenoble, juin 1987, 120 p.

[57] **Yang P. (1981)** *Un essai sur la génération du chinois.* Rapport de stage, GETA, Grenoble, Dec. 1981, 30 p. (+ annexes)

[58] **Zaharin Y. (1986)** *Strategies and heuristics in the analysis of a natural language in Machine Translation.* Proc. COLING-86, Bonn, Aug. 1986, pp. 136—139.

-o-o-o-o-o-o-o-o-o-o-o-

Contents

-o-o-o-o-o-o-o-o-o-o-