[From: *Studies in machine translation*, international workshop…Riyadh, March 1985]

The Approach of GETA to Automatic Translation Comparison with some other methods.

B. Vauquois janvier 1985

Table of Contents

I. Purposes and Limits of the System

In many places, the use of computers for translation purposes is interpreted in different ways. In some cases, the computer's role is considered as a tool to help a human translator for dictionary look-up or, for some standards, in translation of technical texts. In other cases, the computer "translates" using some interactive communication with a human expert. At some extreme cases (TAUM-meteo in Canada) each weather report is either fully translated by the automatic process (if the system did not notice any difficulty) or fully transmitted to a human translator (if the system discovered some failure in the automatic process).

The GETA system is designed for handling multi-lingual translations at various degrees of automation. It can be used at both extremes: either as an aid for human translation using software tools for dictionary look-up and editing, or as a purely automatic translation without any human interference.

In fact, the only experimental production of translations made with this system (Russian-French translations for scientific abstracts in various fields such as aircraft technology, metallurgy, energy, ...) uses a combination for both types.

It is not possible to have human interaction during the automatic translation phase, except (by an optional mode) for introducing and coding new lexical entries in the monolingual dictionary of the source language.

The multilingual aspect of the system comes from the fact that the result of the automatically computed analysis of the input text can be transferred to several models of generation corresponding to the various target languages.

To reach this aim, GETA has developed:

- a basic sofotware called ARIANE-78
- a method for building linguistic models;
- various applications of this method to several prototypes.

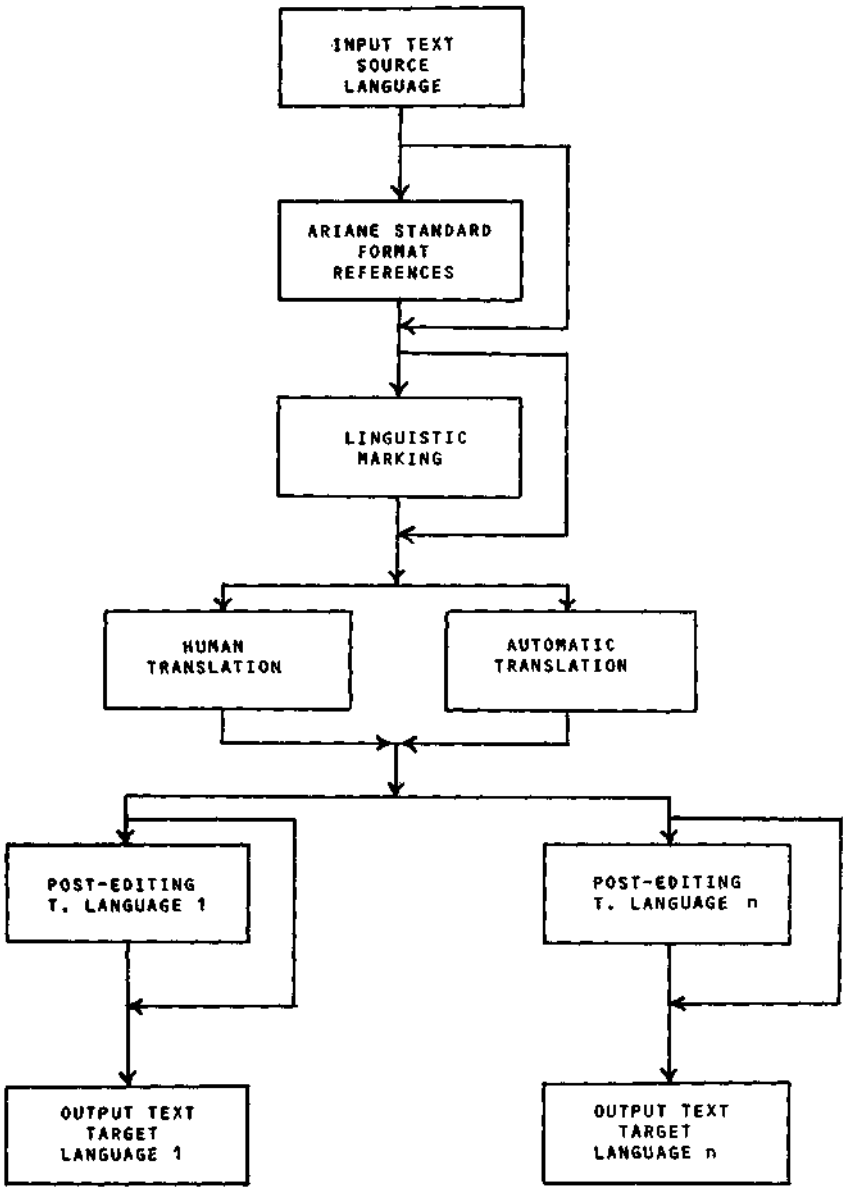

INPUT TEXT
SOURCE
LANGUAGE
ARIANE STANDARD
FORMAT
REFERENCES
LINGUISTIC
MARKING
HUMAN
TRANSLATION
AUTOMATIC
TRANSLATION
POST-EDITING
T. LANGUAGE 1
POST-EDITING
T. LANGUAGE n
OUTPUT TEXT
TARGET
LANGUAGE 1
OUTPUT TEXT
TARGET
LANGUAGE n

Figure 1

l.A The General Architecture of ARIANE-78

As shown in figure 2, ARIANE-78 has various different functions and provides the user with a number of different environments, namely:

a) ***The preparation of texts***

In this environment, the user can declare different corpora, and insert, modify, or delete texts in the selected corpus; other operations enable the user to merge or to segment these texts.

b) ***The preparation (creation or modification) of linguistic models for automatic translation.***

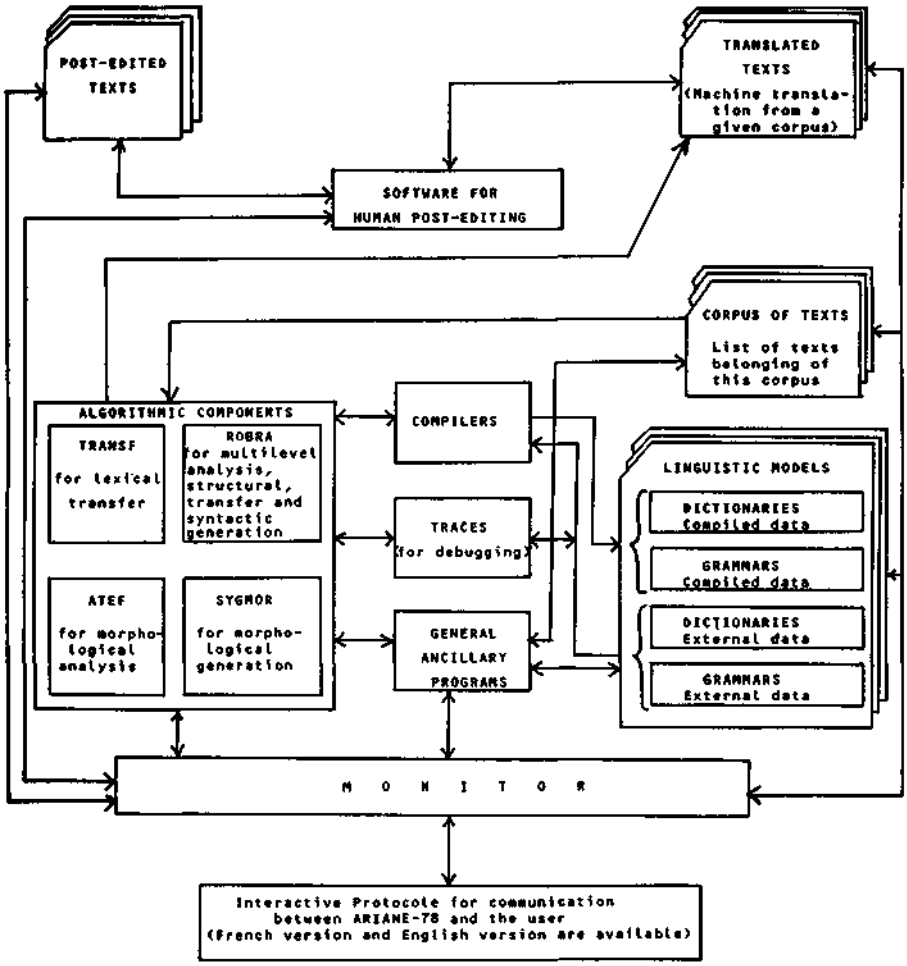
According to the principle of a 3-phase strategy of mono-lingual analysis, bilingual transfer and mono-lingual generation, any source text is processed by a sequence of 6 programs (2 programs for each phase) in order to produce a translated text. As illustrated by figure 3, these programs are executed by the four algorithmic components:

- ATEF for morphological analysis;
- ROBRA for structural analysis, structural transfer and syntactic generation;
- TRANSF for lexical transfer;
- SYGMOR for morphological generation.

In the preparation environment, ARIANE-78 is given new linguistic data for anyone of these programs. These linguistic data have been previously written in the formalism of the external language by the computational linguist. With ARIANE-78, access to the appropriate files for attributes, grammars and dictionaries is handled automatically; when everything is ready for compilation, these external linguistic data are converted into compiled data.

c) ***The execution of linguistic models***

In this environment, for all models which are already compiled, the user can order the processing of a complete translation (i.e. from morphological analysis to morphological generation) on some text (or any sequence of texts) stored in some corpus.



Organization of ARIANE-78.

Figure 2

Partial processing may also be requested, for example only the analysis of the source language for a selected text, or the complete generation in a given target language for some text which has been previously submitted to analysis and transfer.

Parameters are available for independently specifying various levels of trace (or no trace at all) for the execution of each program.

d) ***A special environment for human post-editing***

This environment uses the editor, extended with various annex features to facilitate the task of the post-editor working at the terminal. The screen may be divided into 3 windows showing the source text, the rough translation produced by the automatic process and the revised translation. The post-editor also has access to the dictionaries which have been stored within the ARIANE-78 system.

1 .B. Framework of the Linguistic Models

In order to translate correctly some text from a source language into some target language it is necessary to get some interpretation of it. Any interpretation is referred to a list of concepts and a set of relations between these concepts according to some data structure.

When a text is entered the computer, the concepts are only the set of characters, the relations are limited to the catenation operation on a string data structures. Two sentences which are different by one character did not have the same interpretation at this elementary level. In fact, any sentence is a sequence of words which are related by syntactic rules of the language. A level of interpretation suitable for translation should give the corresponding lexical references in the target language and indicate the relations between these lexical items in terms of logical and semantic relations, free from the syntactic and morphological constraints given by the source language. Such a level of interpretation enables the system to find a common structure for all the sentences which can be considered as synonymous.

If we look at the following sentences:

(1) "This experiment shows the absorption of oxygen by the compound".
(2) "This experiment shows that the compound absorbs oxygen".
(3) "This experiment shows that oxygen is absorbed by the compound".
(4) "The absorption of oxygen by the compound is shown by this experiment".

We want to assign them the same structure of relations between the lexical items, where "absorbe" and "absorption" have the same lexical reference.

The possible translation in Arabic language are the following:

(1') " تبيّن هذه التجربة امتصاص المركب للأوكسجين

(2') " تبيّن هذه التجربة المركب يمتص الأوكسجين

For that purpose it is necessary to have a level of interpretation powerful enough to handle these possibilities of paraphrasing.

In principle, the most ambitious level of interpretation should be the only one to be considered for transfer. Indeed, this level has the greatest possibilities of recognizing or paraphrasing different sentences considered as equivalent. In fact, a complete analysis cannot be expected for each sentence, consequently, for the sake of a fail-soft strategy it is important to preserve the other levels of interpretation which are closer to the surface. In case of failure at the top level, the lower ones can be used as safety nets. In fact, the levels of interpretation which are presented on the same data structure are the following:

a) The level of phrases (K) which reveals bracketted organization of the sentence.

b) The level of syntactic functions (SF) which indicates the distribution of these functions in each phrase.

c) The level of logical relations (LR) which assigns the arguments to their governing predicate and the level of semantic relations (RS) which characterizes the Interpretation of the relation between two nodes which are not connected by any logical relation (as it is the case with circumstantials in clauses and some complements in noun phrases).

Besides these labels dealing with the different levels of interpretation, the nodes of the tree structure associated with each sentence convey other kinds of information by means of other labels.

Included among this information are the following:

-grammatical values (e.g. Number, Tense, Voice, ...)
- syntactic properties (e.g. syntactic valencies,...)
- semantic properties (e.g. semantic valencies, semantic features, derivations, ...)
-tactical information (e.g. the presence of objects, the unsafely of some solution of ambiguity,...)

About the lexical reference, it is obvious that the "usual" word is not the suitable unit. There is some link between "efficient", "efficiency" and "efficiently" and also, between "inefficient", "inefficiency", "inefficiently".

Some other derivations occur between "produce", "production", "producer", "producible", "productive", "productively", "productiveness", ...

Consequently, the lexical organisation of the linguistic models fulfils this expectation. Indeed, the pattern of a "Lexical Unit" (L.U.) is the following:

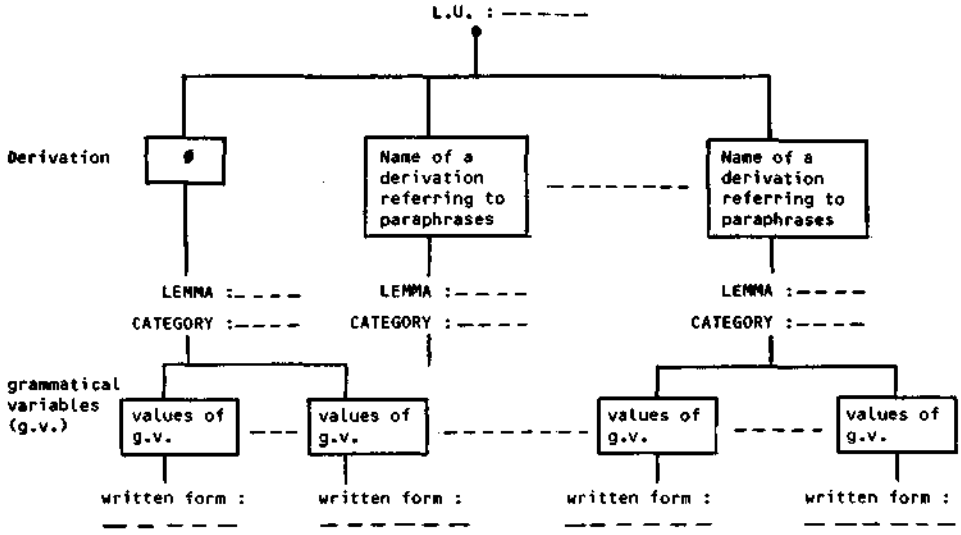


Figure 3

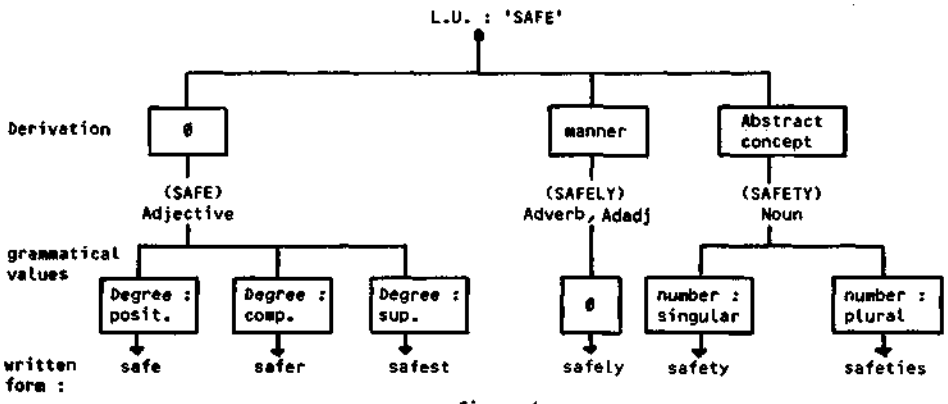


Figure 4

"Unsafe", "unsafely", ... are given by the variable of negation.

The derivation value "manner" is defined by the paraphrases :

(1) : in a [∅] way giving here "in a safe way".

and (2) : with [Abstract concept] giving here "with safety".

l.C. Limits of the System

If the organization of the linguistic models and the flexibility of the software are designed for a multi-lingual automatic translation with a responsible hope of good results, that cannot insure successful results for any kind of texts. Obviously, with the help of coded dictionaries, formalized grammars and sophisticated algorithms it is impossible to represent the whole linguistic knowledge on the one hand and the knowledge about the world referred to by the texts on the other hand. Besides these restrictions coming from scientific considerations it is also important to take care of the economical constraints. Indeed, it is not necessary to increase the cost of dictionary coding by adding more and more information on each lexical unit, if this information is almost never used or if the gain in quality is not significant for the post-editor.

Also, these limits have a consequence on the level of interpretation which can be considered deep enough for the end of the analysis phase. This point is very sensitive when looking for a multilingual automatic translation where the various target languages are either close to or far from the source language.

Finally, these limits impose a strategy for the realization of the linguistic models. In fact, it is impossible to design a general model, for a given language, able to process any kind of texts dealing with any topic. Consequently, the models are aimed for a given typology (style of texts) and for a given technical field. The strategy consists in designing the models in a modular way where a change of typology is performed by the minimum of module substitutions.

The organization of the lexical content by means of general and separate terminological dictionaries is the most efficient way of decreasing lexical ambiguities and changing the field of texts.

2. The Design of a Linguistic Model

Any linguistic model, dealing with one language, should give the mapping between the strings of the language (called level of interpretation 0) and the various levels of interpretation which have been selected. In fact, the description is divided into two parts:

The first one deals with morphology; it is the mapping between the words and the pattern of information representing the associated interpretation. Usually, the mapping is given between one word and one pattern. However, a sequence of words considered as an idiomatic expression is associated with one pattern. Conversely, an ambiguous word may be associated with several patterns.

The second part deals with structural interpretations of the sentences; it is the mapping between the string of patterns coming from the morphological interpretation and the various levels mentioned in part I-B.

The atomic contents of information are given by means of "attributes" (also called "variables"). Any attribute is defined by a name, a type, and values. The type may be either "exclusive" (EXC), or "non-exclusive" (NEX)or integer (ARITH).

For an EXC type only one value can occur in the pattern of information, whereas for a NEX type any subset of the declared values is permitted. For an ARITH type any integer (positive or negative) may occur inside the declared range.

Here are some examples of attribute declaration:

NUM: = :SIGN, DUAL, PLUR). type NEX.
(for number may be any subset of values among singular, dual, plural)
VOICE: = (ACTIVE, PASSIVE), type EXC.
(for voice may be one of the two values either active or passive).
LENGTH: = (150). Type ARITH.
(to indicate the number of words in a sentence, limited to 150).

Many attributes have to be given by the dictionary as properties of the lexical unit and its derived words; whenever a particular form of a word (because of an irregularity, for instance "is") appears as an entry in the dictionary, the properties of the form must be added. Usually, the information conveyed by attributes in the dictionary refer to morphological, syntactic and semantic properties.

Other attributes are defined for the reference to the various levels of interpretation. At the level of morpho-syntactic classes, the selected classes may be declared as:

K: = (ADVP, ADJP, NP, CARDP, VCL, SUBCL, RELCL, PARTCL,...). Type EXC

for Adverbial Phrase, Adjectival Phrase, Noun Phrase, Cardinal Phrase, Verbal Clause, and so on.

The same goes for the other levels of interpretation as:

- syntactic functions	:	SF: = (SUBJ, OBJ1 ,OBJ2, ATSUBJ,....) type EXC
-logical relations	:	LR: = (ARG0, ARG1, ARG2,... type EXC
- semantic relations	:	SR: = (AGENT, INSTRUMENT, AIM, COMPANY, CAUSE,....) type EXC

So, the basic stones with which the model is defined are all these attributes and the lexical units.

The list of logical and semantic relations are supposed to be the same for all languages processed in a multilingual translation.

2. A. An Example of Structural Description of a Sentence at Various levels of Interpretation

The example given by figures 5,6 and 7 shows the associated structures for the English sentence:

"This experiment explains the frequent use of a catalyst to facilitate the reaction".

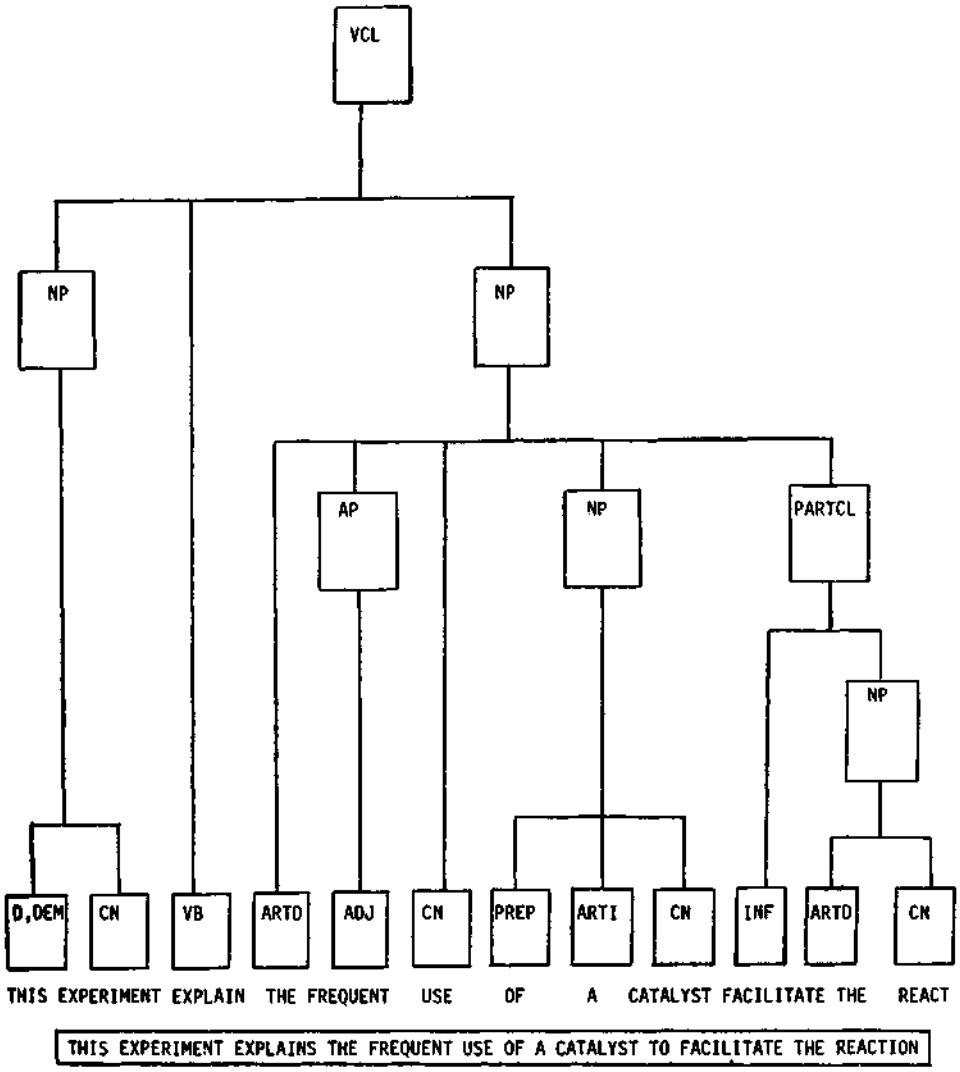

VCL
NP
NP
AP
NP
PARTCL
NP
D,DEM
CN
VB
ARTD
ADJ
CN
PREP
ARTI
CN
INF
ARTD
CN
THIS EXPERIMENT EXPLAIN THE FREQUENT USE OF A CATALYST FACILITATE THE REACT
THIS EXPERIMENT EXPLAINS THE FREQUENT USE OF A CATALYST TO FACILITATE THE REACTION

Figure 5

DISCOVERING THE INTERFACE STRUCTURE LEVEL AFTER LEVEL

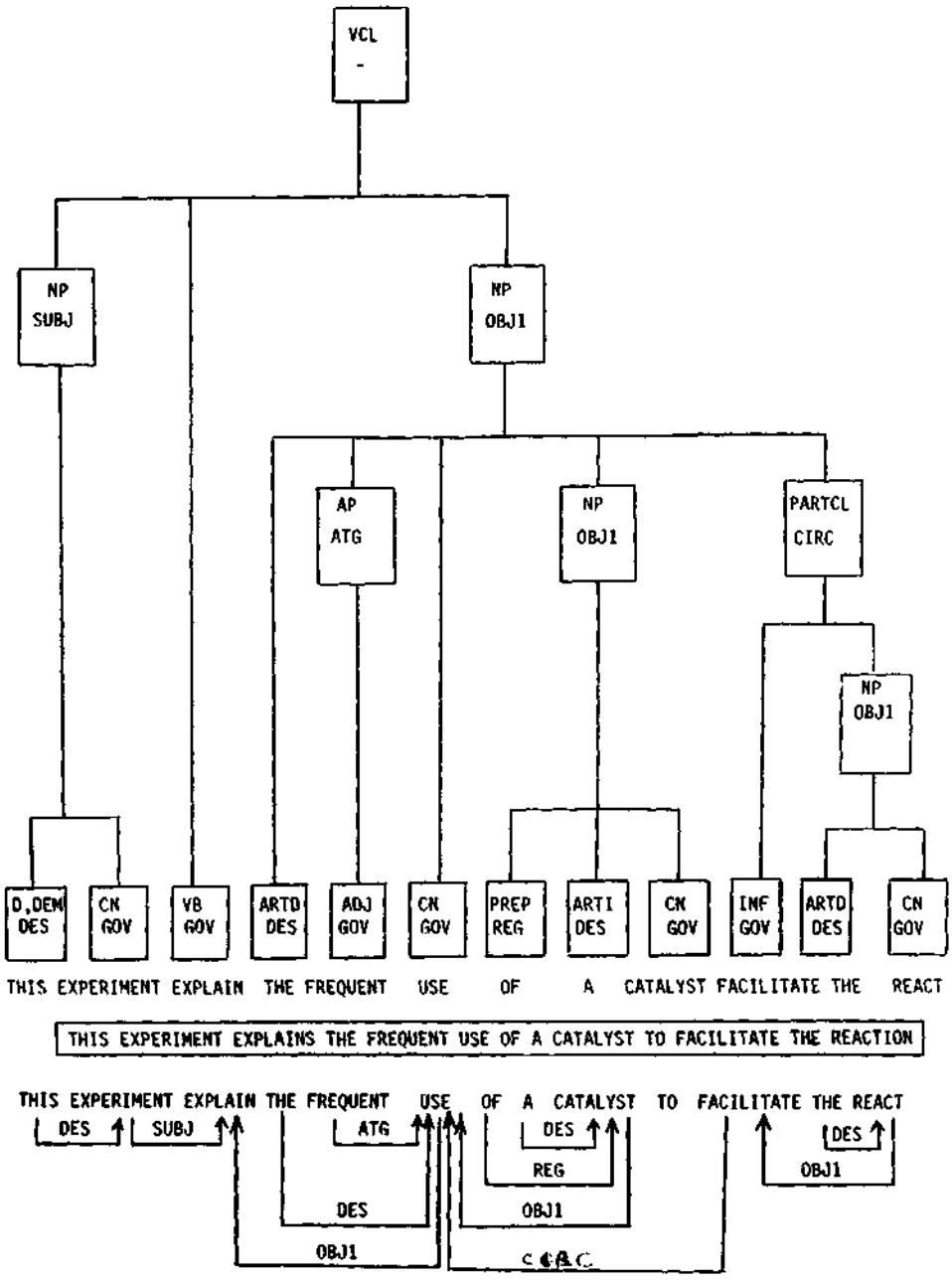


Figure 6

DISCOVERING THE INTERFACE STRUCTURE LEVEL AFTER LEVEL

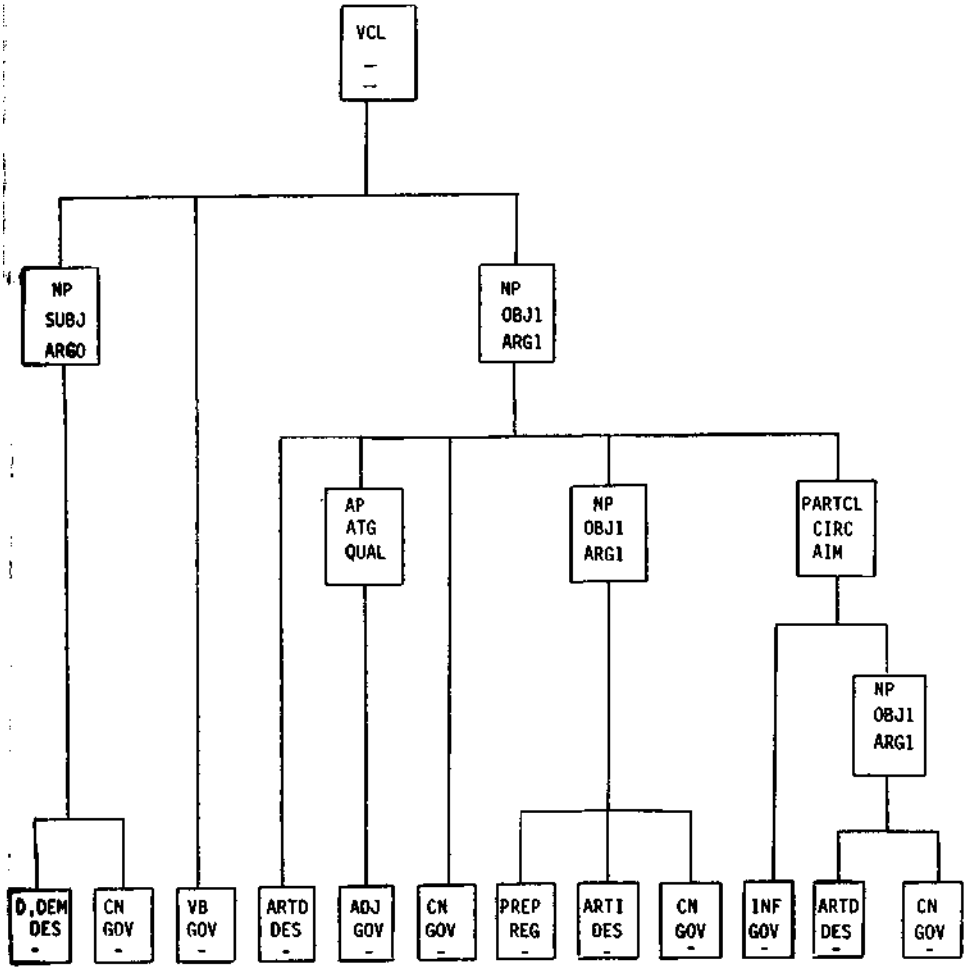
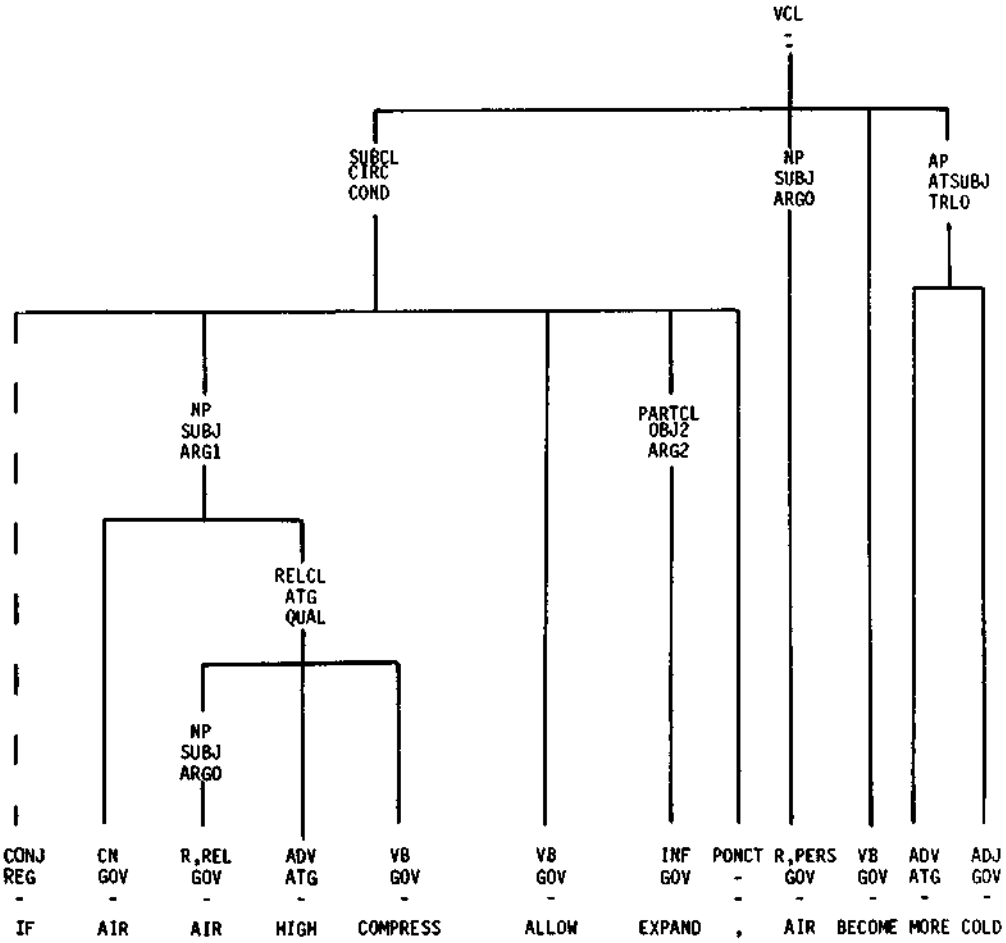


Figure 7

The figure 8 shows directly the multi-structure associated with the sentence : "If air which has been highly compressed is allowed to expand, it becomes colder".

EXAMPLE OF AN INTERFACE STRUCTURE
WITH LABELS FOR THE INTERPRETATION LEVELS



IF AIR WHICH HAS BEEN HIGLY COMPRESSED IS ALLOWED TO EXPAND, IT BECOMES COLDER.

Figure 8

2.B. Framework for Grammars

Considering a sentence as a unit for translation, the automatic translation process has to perform 3 phases:

-1) *Analysis:* this means the transformation of the input text into a structural description as seen just above. This phase is divided into two sequential subphases, namely morphological analysis and structural analysis.

The morphological analysis yields a labelled tree structure as shown by the Figure 9. The structural analysis is expected to yield a labelled tree structure as indicated for the example of 2-A.

- 2) *Transfer:* also divided into two parts, this phase begins with a lexical transfer (a bilingual dictionary look-up) followed by a structural transfer. The latter is mainly an extension of the lexical transfer to solve ambiguities by means of a larger context exploration and computes the contrastive aspects between the source and target languages (for example, contrastive values in tense, mode, aspect, determination, etc.). Finally, if necessary, the structural transfer computes some surface levels of interpretation whenever the analysis phase failed to reach the level of logical and semantic relations.

- 3) *Generation:* also divided into two parts, namely a syntactic generation followed by a morphological generation. The former selects a syntactic structure by computing the new syntactic functions and the new syntagmatic classes of the target language from the only values of logical and semantic relations labelling the tree structure. The complete form of the words built by concatenation (with possible alteration) of the segments (roots, prefixes, suffixes, ...) is performed by the morphological generation.

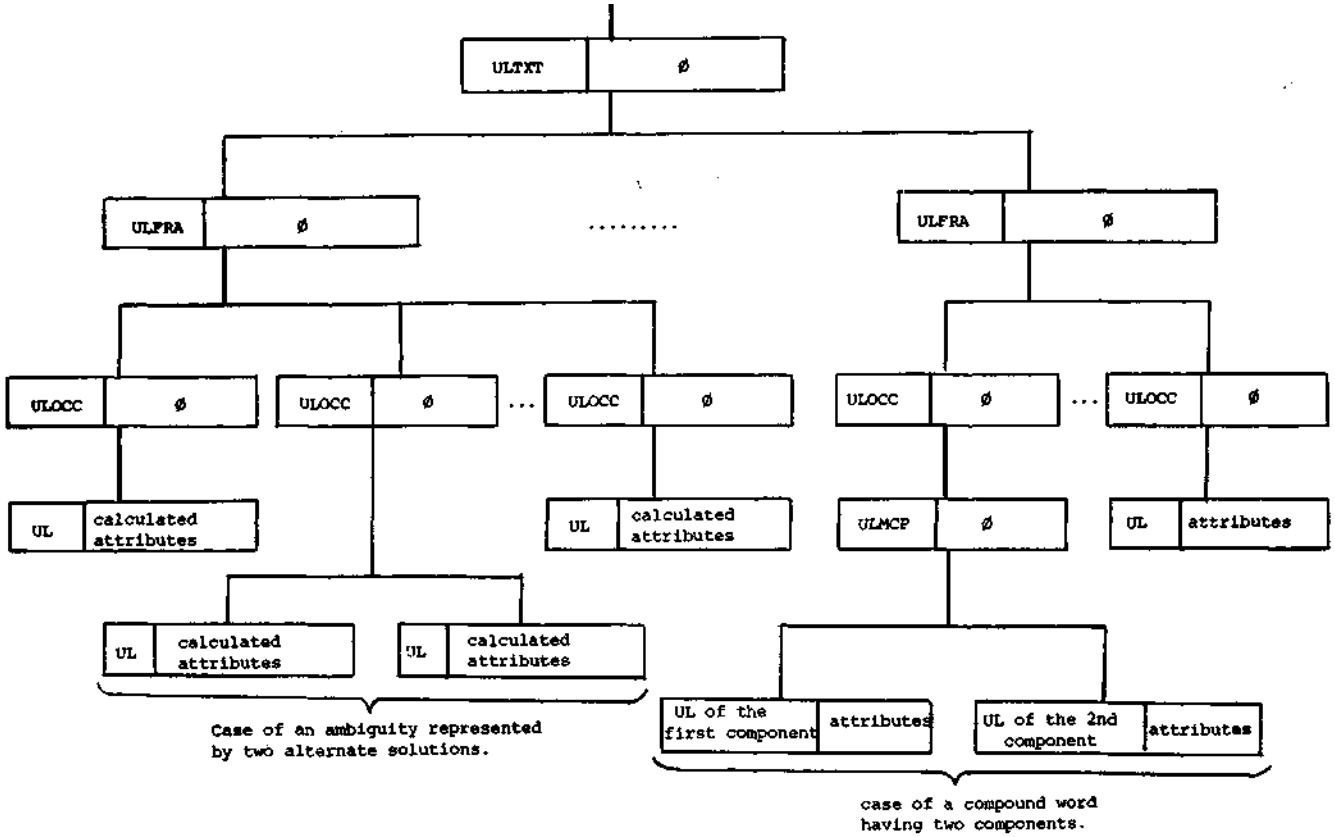


Figure 9

It should be mentionned that ambiguities and compound words are not mutually exclusive under a 'ULOCC' node.

To perform all these tasks, it is necessary to write dictionaries and grammars.

The grammars which are operated by the algorithmic components of ARIANE-78 for those various phases are "dynamic grammars". For a given language, such a grammar which is written for analysis cannot be used for generation and vice-versa. Indeed, the analysis process looks for the computation of the structure and the associated labels to yield the levels of interpretation; during this phase, many ambiguities have to be solved. On the contrary, the generation process has no more ambiguities to solve, but has the guidance of the style chosen for the output text, according to given parameters. Nevertheless, in both cases, the linguistic model considered as a mapping between the strings and the associated structures remains the same.

For this purpose, a formalism has been developed at GETA, called a "static grammar formalism" by means of which the linguist can describe this mapping. The result is a "static grammar" because only the correspondence between strings and structures is given; the dynamic processes for obtaining the structures from the strings or vice-versa are not involved in this description. This formalism is based on the writing of "charts".

Each chart exhibits the mapping between a set of strings and the associated set of structures (the sets may be infinite).

In practical terms, each chart is divided into three zones:

- Zone 1 characterizes the geometry of the tree structures according to the strings. The elements of the string appear as the leaves of the tree.

- Zone 2 imposes restrictions on the strings by means of predicates; it is here that the validity of the substrings is controlled.

- Zone 3 shows the association of labels between the leaves and the other nodes of the tree structure.

An example of such charts is given in the figure 10.

In zone 1, an element of the string written between brackets is optional. The star has the usual meaning of interaction. A node which belongs to the string but is eliminated in the structure is not connected to any other node in the tree.

Conversely, a leaf which belongs to the structure but does not appear in the string is indicated by a symbol "O" instead of symbol "X".

Finally, a leaf may appear at a certain location in the string and at another location in the structure. The two preceding conventions enable the linguist to describe this phenomenon.

Example of a chart for Arabic Language:

The chart given by the figure 10 illustrates the mapping between the strings and their associated structures in the case of a "nominal clause" stated by the sequence of:

- an optional "Kana" or "Inna"
- a mubtada
- and a xabar.

The valid strings have to fulfil the logical conditions stated in zone II. For that purpose, some predicates currently used in these conditions have been separately defined and are given as follows:

Some predicates in use for the static charts

$ AGRNB (X,Y) = NUM (X) ∩ NUM (Y) ≠ NUMO
$ AGRGNR (X,Y) = NUM (X) ∩ GNR (Y) ≠ GNRO
$ AGRPSUJ (X,Y) = NUM (X) = SING (– NUM (Y) PLU SEM(Y) – CONT – HUM ⩓ $ AGRGNR (X,Y) GNR(X) =FEM)
$ AGRSUJP (X,Y) = (NUM(X) ≠ PLU ⩔ SEM(X) – CONT – HUM ⩓ $ AGRGNR (X,Y) ⩓ $ AGRNB (X,Y))
⩔ (GNR(Y) = FEM ⩓ NUM(Y) = SING)
$AGRNN(X,Y) = $ AGRNB(X,Y) ⩓ (DBGNR(Y) ≠ 1 ⩔ $ AGRGR(X,Y)) DBGNR: = (1)

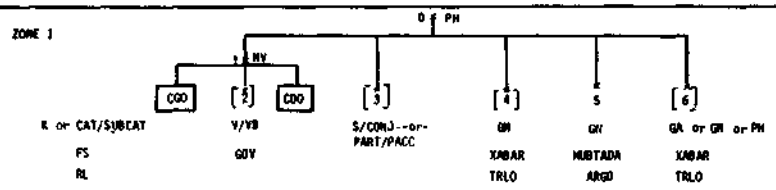
This last variable indicates the possibility of both gender for a common noun.

Described language : ARABIC
Static grammar : S.ARX
Chart n° : 23

TYPE NOMINAL CLAUSE

Considered cases : [KANA ou INNA] + MUBTADA + XABAR
Referred charts : (GR) ALL
(GA) ALL
(NV) 009
(PH) 021, 023

ZONE I



ZONE II

$4 \equiv \neg 6$. VAL1(4) $\neq$ N . K(1) = NV . $2 \equiv \neg 3$.
TYPE(1) = VBEXIST . GNRPH(0) = PHNOM .
(K(6) = GA $\wedge$ $ AGRSUJP(5,6)) $\vee$ (K(6) = E = GN $\wedge$ $ AGRNM(6,5)) .
(2 $\wedge$ CAT(5) = R) $\supset$ (UL(5) = ULO) .
$\neg 2 \Rightarrow$ TENSE(1) = PRES $\wedge$ UL(1) = ULO .
6 $\neq$ DET(5) = 1 .
$ AGRPSUJ(1,5)

ZONE III

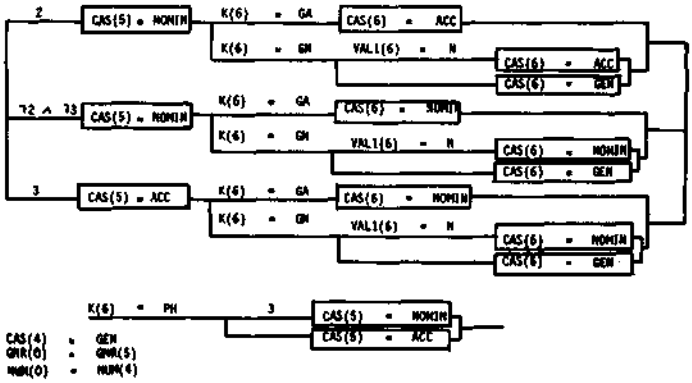


Figure 10

The dynamic grammars are written by the external languages of ARIANE's component, the so-called Specialized Languages for Linguistic Programming (SLLP).

2.C Framework for Dictionaries

Within the GETA's approach, three kinds of dictionaries are necessary for a translation process between two languages.

a) *The source language dictionaries*

For any source language dictionary, the access key is a string of characters corresponding to strings occurring in the text. They may be stems, prefixes, suffixes; they may be also full forms of words (in case of an extremely irregular form); they may be also sequence of words representing an idiomatic expression.

The ARIANE system offers the possibility of defining up to 6 dictionaries for segments and one more for idiomatic expression. The designer of a linguistic model decides what is the vocation of each dictionary: for instance, one dictionary for the stems and full forms of the general vocabulary; another one for prefixes, one more for suffixes; other dictionaries will be specialized for given terminological fields...

The content of information associated with each segment is precisely given to produce the expected values to the attributes of the lexical nodes as mentioned in I-B.

b) *The transfer dictionaries*

These dictionaries are used for the "lexical transfer phase" and they have to deliver a solution (or sometimes alternate solutions) for various cases. For each item, the access key is a source lexical unit.

Then, the following cases may occur:

- simple-to-simple substitution
- simple-to-complex substitution (a single source unit is translated by several target units, e.g. AVEC -» BY MEANS OF)
- complex-to-simple or complex-to-complex substitution (e.g. COMPUTER SCIENCE-* INFORMATIQUE, LET ... KNOW-+ INFORMED).

Moreover, there may be conditions to select an appropriate substitution for a given lexical unit. These conditions may be local, that is they bear on the proper-

ties of the node under consideration (and perhaps some immediate neighbours), or global, in which case a wider context must be examined.

Let us give two examples.

- syntactic valency: the English verb LOOK has at least 4 values for the "object valency" (for argument 1), namely AT, FOR, LIKE, AFTER. According to the syntactic structure which has been built, only one possibility remains after analysis (on the node containing LOOK). We may then express the conditional substitution by an item in an English-Arabic transfer dictionary of the following form:

'LOOK': if VAL1 = AT then	نظر
elsif VALl = FOR then	بحث
elsif VALl = LIKE	شابه
then	اعتنى
else	

- presence or absence of an argument: the usual translation of GIVE is DONNER in French; however, if there is no first argument explicit in the sentence (e.g. "John was given a book"), the translation of GIVE may be RECEVOIR, with the indication that the third argument of GIVE becomes the first argument of RECEVOIR (e.g. "Jean a recu un liver").

Basically, the TRASF SLLP provides the means to write bilingual multi-choice dictionaries. Each node of the input tree is replaced by a subtree in the output tree. This subtree may be selected from several possibilities, according to the evaluation of a predicate on the attributes of the input node and its immediate neighbours.

In simple cases, the selected subtree is reduced to one node. In more complex cases, the selected subtree may:

- give *several possible equivalents*, for further testing in subsequent phases, or production of a multiple equivalent in the final translation (e.g. PROCESS -» PROCESSUS or PROCEDE from English into French).

- express *the prediction* that the considered element may be part of a complex expression in the source language (e.g. LET ... KNOW). In this case, the subtree will contain nodes describing the type of complex predicted, the other elements of the complex, and the translation of the complex (which may again be simple or complex). It will then be one of the tasks of the structural transfer to confirm or infirm this prediction, by using the whole available context, and to take appropriate action: use the translation of the complex if yes, leave the simple translation if no.

This organization has been consciously designed in order to limit the cost of indexing in dictionaries: the lexicographers don't have to write complex tree-transformation rules. Instead, they write (static) subtrees of well-defined (sort of AND/OR) forms, in which some of the information is later used as *indirect call* to transformational procedures written by specialized computational linguists in the structural transfer grammar.

c) *The target language dictionary*

In fact all the semantic and syntactic informations of a target lexical unit have been already given in the transfer dictionary.

The so-called "target language dictionary" is limited to the morphological informations. The main dictionary is accessed by "lexical units". The other dictionaries are accessed by values of attributes. They all give as a result the corresponding string which has to be produced for the output string (the translated text). The suitable concatenation of such strings is guided by the grammar for morphological generation.

2. D. Methodology for a Workshop realizing Linguistic Models

The preceding paragraphs have given a survey of the various tasks to be performed in order to produce computable linguistic models for translation purposes.

The following table, figure 11, shows the linkage between them. The top horizontal zone includes the basic linguistic data (as primary knowledge) which are used for the 3 phases (analysis, transfer, generation). These data appear following one of the 3 possible forms.

a) a rough form as delivered by the corpus. An homogeneous corpus in a given field reveals the typology of the texts (for grammatical consideration) and indicates the usual interpretation for the colloquial words.

b) an elaborate form (sometimes only partially) as given by the traditional dictionaries and the terminological data bases.

c) an intellectual form represented by the general knowledge of linguistics, the particular knowledge of some language and/or by expertise in lexicography and translation.

For example, the syntactic categories, the syntagmatic classes and many attributes (gener, number, tense, aspect, ...) useful for the models can be deduced

I from the general knowledge of linguistics and that of the particular language. |The expertise in lexicography yields the assignment of values for those attributes, as well as the creation of new attributes (for example, semantic features).

The Second zone indicates the processes which aim to obtain more elaborate data as quantitative results about the typology; also, formatted dictionaries (by eliminating irrelevant informations and by adding new informations) from which it can be expected a safe and efficient encoding.

The third zone shows the expected results and the addition of a new knowledge (about the formalism of static grammars).

The fourth zone includes the following processes:

- the design and writing of static grammars. This task is essentially a human process; however, some special text editing routines can be used for up-dating the charts.

- the encoding of the general dictionaries with the help of software tools preparing the "menus".

- the encoding of the terminological dictionaries within the same conditions.

The fifth zone exhibits the results obtained at this level: static grammars, general and terminological dictionaries (both operational dictionaries for automatic translation and "human" dictionaries for post-editing or human translation). The "intellectual" form of knowledge about "informatic" (algorithmic process, data structures, heuristic programming,...) are now introduced in the diagram.

The sixth zone deals with the realization of dynamic grammars for analysis (morphological and structural analysis), transfer (structural transfer) and generation (syntactic and morphological generation).

Finally, the seventh zone exhibits the final products, ready for automatic translation and for human post-editing (or translation) environment.

ORGANIZATION OF THE TASKS FOR LINGUISTIC MODELS

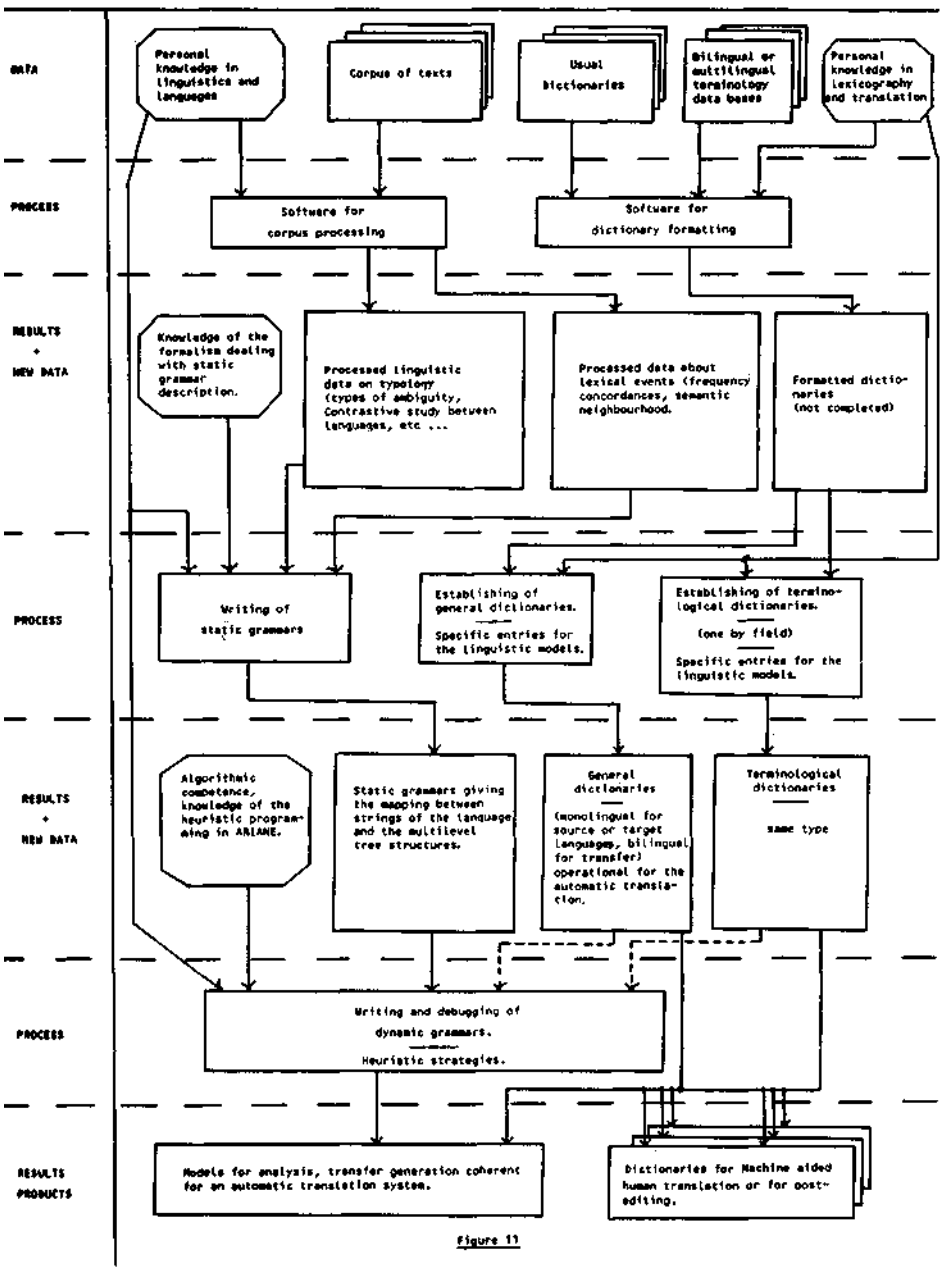


Figure 11

3. - Computational Techniques

The process of automatic translation is a sequence of 6 routines generated by the algorithmic components of ARIANE-78 as shown by the figure 12.

ATEF is the specialized generator of morphological analyzers*.

ROBRA is a tree-to-tree transducer; so, it is the generator of 3 routines, namely: multi-level structural analysis, structural transfer and syntactic generation.

TRANSF appears only once in the process; it is the bi-lingual dictionary processor.

SYGMOR is the specialized generator of morphological generation routines.

All these components are instructed by "Specialized Languages for Linguistic Programming" (SLLPs), available for writing dictionaries, grammars and strategies.

3. A. Dictionaries

Dictionaries are written in the SLLPs, and then compiled into some internal representation which includes a fast access method. At execution time, the dictionaries reside in virtual memory.

ATEF dictionaries use a two-step hash-coding scheme, followed by an ordered-table representation (for morphs of the same length sharing the same initial or final character). Access to a given item involves less than 100 machine operations.

TRANSF dictionaries use a quasi-perfect hash-coding scheme. SYGMOR dictionary access relies on dichotomy for the lexical units and on sequential search for other variables. But, then, the latter dictionaries are always small and of bounded size anyway: they contain the prefixes, affixes and endings.

As a matter of fact, due to the compactness of the internal coding and to the speed of the access methods, dictionaries don't raise any computational problem.

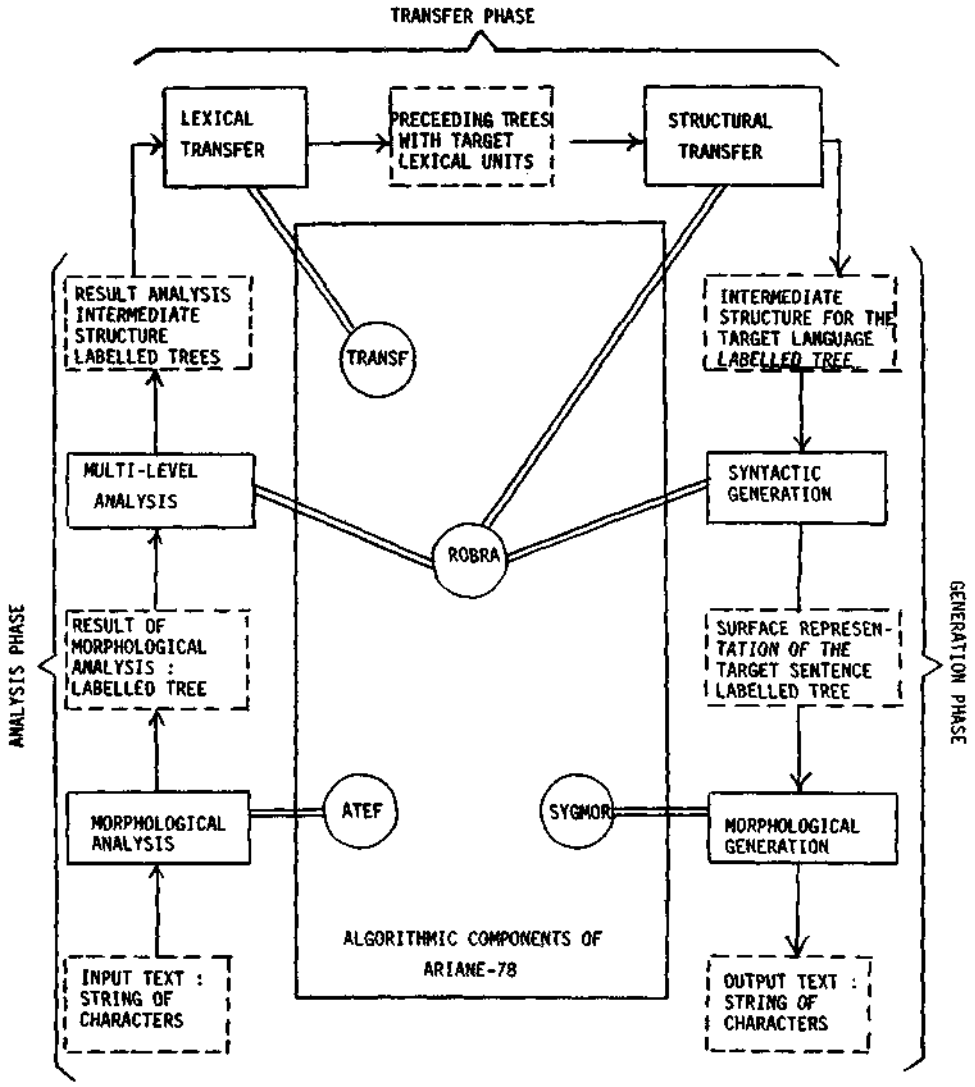

TRANSFER PHASE
LEXICAL TRANSFER
PRECEEDING TREES WITH TARGET LEXICAL UNITS
STRUCTURAL TRANSFER
RESULT ANALYSIS INTERMEDIATE STRUCTURE LABELLED TREES
TRANSF
INTERMEDIATE STRUCTURE FOR THE TARGET LANGUAGE LABELLED TREE
MULTI-LEVEL ANALYSIS
SYNTACTIC GENERATION
ROBRA
RESULT OF MORPHOLOGICAL ANALYSIS : LABELLED TREE
SURFACE REPRESENTATION OF THE TARGET SENTENCE LABELLED TREE
ANALYSIS PHASE
GENERATION PHASE
MORPHOLOGICAL ANALYSIS
ATEF
SYGMOR
MORPHOLOGICAL GENERATION
ALGORITHMIC COMPONENTS OF ARIANE-78
INPUT TEXT : STRING OF CHARACTERS
OUTPUT TEXT : STRING OF CHARACTERS

Figure 12

3. B. Grammars

In ATEF, the grammar describes a finite-state non-deterministic transducer. It is implicitly divided into as many subgrammars as "morphological formats" (classes). Each item in a dictionary has such a format, which contains some static grammatical information, and is also referenced in the l.h.s. of one or several rules, which will be called in a non-deterministic way.

The underlying mechanism for handling non-determinism is simple backtrack. However, heuristic functions may be called by the rules. Their effect is to "prune" the search space in several predetermined ways. For instance, one of these functions says something like: if the application of the current rule leads to some solution, then don't compute the solutions which might be obtained from segments of strictly shorter length than the current one beginning at the same character. Another heuristic function is used to simply state that, if the rule leads to a solution, this solution should be the only one: previously computed solutions are discarded, and further possibilities are not examined.

In ROBRA, there are several levels of control. First, a given transformational system (TS) has a given store of transformational rules (TR), which operate by substitution. Then there is a collection of grammars (TG), each made of an ordered subset of the TRs. The order is local to the grammar, and is interpreted as a priority order. Each TR may contain a recursive call to a (sub) TG or to a transformational subsystem (sub-TS). The top level of control corresponds to the TS (and its sub-TSs), and is described by means of a "control graph" (CG). The nodes of the CG are the TGs, and a special "exit grammar" (&NUL). Arcs bear conditions of the same type as l.h.s. of rules.

ROBRA's interpreter submits the input tree to the initial grammar of the considered (sub) TS, and uses a built-in back-tracking mechanism to find the first path leading from this initial node to the exit node, thereby applying the TGs found in the nodes, and traversing the arcs only if the attached conditions are verified by the current tree. In case of success, the result is the tree which reaches & NUL. In case of failure, the output is set equal to the input.

A "simple" execution of a given TG is carried out in two steps. First, a *parallel* application of the TRs of the TG is performed, by selecting the maximal (according to some parameters) family of non-overlapping occurrences of rule schemas (l.h.s.) and applying the corresponding rules. Then, the recursive calls, if any, are executed, by submitting the appropriate subtrees to the called sub-TG or sub-TS.

The execution of a TG in "exhaustive" mode consists in iterating simple executions of this TG until no rule is applicable any more. In "controlled" mode,

a marking algorithm is used in order to strictly diminish the number of possible occurrences of rules at each iteration, ensuring termination of the process. Hence, it is possible for the compiler to *statically detect possible sources of undecidability*, just by checking the modes of the TGs, testing for loops in the CG, and verifying a simple condition on the form of the recursive calls.

This kind of organization makes it possible to use text-driven strategies, which will operate differently on different parts of the (tree corresponding to the) unit of translation.

The case of SYGMOR is more simple, because the underlying model is a finite-state deterministic transducer. For each new node (leaf of the input tree), the interpreter selects the first rule whose l.h.s. is verified and executes it. Then, it uses the control part of the rule, which consists in an ordered sequence of rules to be applied, some of which may be optional. Usually, SYGMOR grammars are fairly small.

3. C. Efficiency - Comparison with other Softwares

The basic software is programmed at various levels. The compilers and interpreters of the SLLPs are written in assembler or PL 360, the monitor and the macros for the editor (XEDIT) in EXEC/XEDIT (IBM's VM/CMS's shell language). We can say something about efficiency on two levels.

First, the efficiency of the programming itself. Applications such as Russian-French use roughly 1 to 1.5 Mipw (million instructions per word translated) of VCPU (virtual CPU), measured on a 4331, a 370, a 3081 or an Amdahl. More than 85% of this is used by ROBRA's pattern-matching mechanism, and less than 10% by the "dictionary phases" (morphological analysis and generation, lexical transfer). Translation is performed using 2.5 Mbytes of virtual memory, without any access to secondary storage during processing itself.

Recently, we made a comparison with Kyoto University's system, whose design is similar to ARIANE-78's. In particular, the bulk of the computing time is used by GRADE, SLLP of the ROBRA family. The system is programmed in UT-LISP and runs on a FACOM computer (Fujitsu), which is IBM-compatible and very fast (20 Mips).

It turns out that this MT system uses roughly 100 times more Mipw than ARIANE-78, and 40 times more space. Taking into account the fact that there is only 4 Mbytes of virtual memory (divided by 2 for the purpose of garbage collection), that the garbage collector gobble up 40% of the VCPU, and that the run-time access to the 30 to 40 Mbytes of secondary storage (holding the

lingware) takes 20% more, we end up with the net result that a *LISP implementation* (of this type of system) *is 40 times more voracious in computer time and space than a low-level implementation.*

Of course, the amount of programming and maintenance effort is higher for the latter type of implementation. At this point, it is worth reminding that in France, contrary to many countries, research labs usually have access to severely limited computer resources, and must pay for it. Natural Language Processing is very much an experimental science, and the designers of ARIANE-78 have felt they couldn't provide the linguists with a system which they might use for experimentations just about one or two weeks a year because of financial constraints.

One possible reward of this painful kind of implementation is that is seems possible to run the complete system on the PC/XT370, according to the specifications and descriptions which have been published in Europe.

A second level of comparison is by the *computational methods* used. For that, we may use old data from the former CETA system (before 1971), or data from current systems such as METAL (University of Texas at Austin), or KATE (KDD, Tokyo), based on augmented context-free formalisms. From some demonstrations and private communications, we got again the figure of 1 to 1.5 Mipw, for systems written in LISP, and about 40 times less for assembler-level implementation of the basic software.

Our (perhaps to hasty) conclusion is that *pattern-matching based techniques are roughly 40 times costlier in VCPU than classical combinatorial methods* (if programmed in a smart way, of course).

However, we have tried the latter kind of approach in the past, and ended up finding it quite difficult to maintain a large scale system and to raise its overall quality.

3. D. Discussion about the Methodology

1 - *Interlingua vs transfer; multilevel structure*

First of all, we have tried an approximation of the interlingua ("pivot") approach, and found it wanting. In the former CETA system, the pivot representation was of a hybrid sort, using as vocabulary the lexical units of a given natural language, and as relations so-called "universals" corresponding to our current logical and semantic relations, plus abstract features such as semantic markers, abstract time and aspect, and so on. The problem here is threefold.

- it is very difficult to design such a pivot in the first place, and even more so if the vocabulary must also be independent of any particular natural language.

- the absence of surface-level information makes it impossible to use contrastive knowledge of two languages to guide the choice between several possible paraphrases at generation time.

- if the high-level representation cannot be computed on part of the unit of translation, the whole unit gets no representation, and hence no translation, or a word-by-word translation. This is already bad enough at the sentence level, and quite insufferable if the units are larger, in order to access a bigger context.

If we are some day to attain the level of performance of an average or good translator, it is unavoidable to sometimes rely on "rules of thumb", which use surface-level information, and embody some contrastive knowledge of the languages at hand.

Moreover, if the units of translation grow larger, the probability that some part cannot be completely analyzed at the most abstract levels of interpretation approaches certainty.

Hence, we feel that the use of *one* (and not several) so called *'"multilevel structure"* for representing each unit is appropriate. As a matter of fact, we consider such a structure as a *generator* of the various structures which have implicitly been computed at each level of interpretation.

This technique may be compared with the "blackboard" technique of some AI systems. During analysis, the different levels of linguistic knowledge are used in a cooperative way, and not sequentially, as in previous systems.

2 - Phrase-structure grammars and parsers vs transducers

Phrase structure (PS) grammars don't seem adequate for our purpose, even with all (not so recent) additions and niceties such as attributes, validation/ invalidation between rules, attached transformations, etc.This is mainly so because the structures which are associated to the strings are grammar-dependent, although they should be language-dependent invariants.

Another problem comes from the monolithic aspect of such grammars, which make them very difficult and ultimately impossible to understand and modify, although everything seems right at the beginning, with a few hundred rules. Stratification of the grammar in the METAL sense is just a device allowing to conserve the results obtained in simple cases while rules are added to take care

of more complex situations. For the latter, the grammar is just the collection of all rules, with no modularity.

Although procedural programming is notoriously more difficult than writing a collection of static rules used by a standard algorithm, leading to a "yes or no" answer, we consider it a lot better in the situation of incomplete and fuzzy knowledge encountered in MT. It happens that the same position has recently gained ground in AI, with the construction of "expert systems" which embody a lot of knowledge in their control and domain-specific heuristics.

In our case, this amounts to use our SLLPs for "language engineering", in much the same way as usual programming languages are used for software engineering. Starting from a kind of functional specification (expressed by means of static grammars), the computational linguistic constructs a corresponding transducer in the time-honoured way of top-down decomposition and stepwise refinement.

3 - Combinatorial methods vs text-driven heuristic programming

Ambiguity is a fundamental problem in Natural Language Processing. Combinatorial methods tend to compute all possibilities, perhaps weighting them, and to filter out the first one, the first N ones, or all of them. No heuristic programming is possible.

If more than one result of analysis (or transfer) is produced, *the source of the ambiguity is lost*, so that the system must produce several distinct translations for the unit. Again, this is difficult to accept at the sentence level, and certainly unacceptable at the paragraph level.

Heuristic programming and text-driven strategies seem more adequate than the use of a very complex grammar, whose rules are all tried, even in simple cases. Experiments have shown that the flow of control (from TG to TG) is significantly different on different parts (subtrees) of the translation units.

4 - Fail-soft mechanisms

In the setting of second-generation systems, based on implicit rather than explicit understanding, a parallel can be made with compilers for programming languages. We don't want our "supercompilers" (for natural language, that is) to stop and produce nothing if they encounter an ill-formed clause somewhere in the unit of translation.

Rather, we want them to produce the best translation they can, under all circumstances, annotating them with special marks, analogous to error and warning messages, to be used later during the postedition and technical revision of the document.

The "safety net" made of the multiplicity of levels of interpretation available on the same structure makes it possible to use a broad spectrum between high-level and word-by-word translation.

* * *

IV - An Example of Translation

English - Arabic

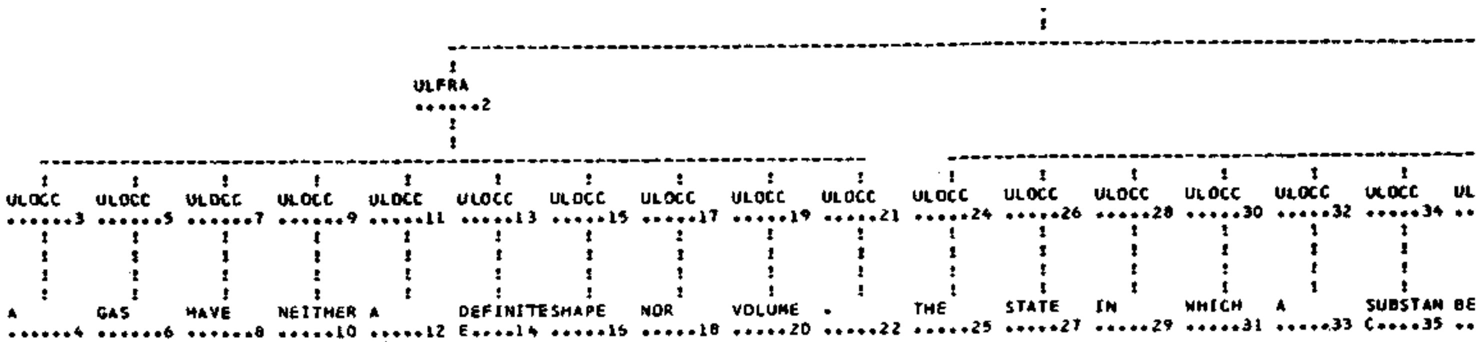
An example of translation of two sentences is presented above with the 6 steps from morphological analysis of english to morphological generation of arabic.

The output of each step is given as a data structure (a complex labelled tree structure) representing the linguistic phenomena. Each node is provided with a decoration describing the properties of this node and its syntactical, logical and semantic relations with other nodes.

*** GROUPE D'ETUDE POUR LA TRADUCTION AUTOMATIQUE ***
****

ANALYSE MORPHOLOGIQUE

RESULTAT DE L'EXECUTION, TEXTE : CHIMI RYADH CODE LANGUE : BXA


ULFRA
......2
ULOCC 3
ULOCC 5
ULOCC 7
ULOCC 9
ULOCC 11
ULOCC 13
ULOCC 15
ULOCC 17
ULOCC 19
ULOCC 21
ULOCC 24
ULOCC 26
ULOCC 28
ULOCC 30
ULOCC 32
ULOCC 34
UL ..
A 4
GAS 6
HAVE 8
NEITHER 10
A 12
DEFINITE....14
SHAPE 16
NOR 18
VOLUME 20
. 22
THE 25
STATE 27
IN 29
WHICH 31
A 33
SUBSTANC....35
BE ..

```
--------------
              !
          ULFRA
          .....23
              !
              !
------------------------------------------------------------------------------
 !      !       !       !       !       !       !       !       !       !
OCC   ULOCC   ULOCC   ULOCC   ULOCC   ULOCC   ULOCC   ULOCC   ULOCC   ULOCC
...36 .....38 .....40 .....42 .....44 .....46 .....48 .....50 .....52 .....54
 !      !       !       !       !       !       !       !       !       !
 !      !       !       !       !       !       !       !       !       !
 !      !       !       !       !       !       !       !       !       !
 !      !       !       !       !       !       !       !       !       !
      DEPEND  ON      THE     ENERGY  THAT    THE     SUBSTAN CONTAIN .
...37 .....39 .....41 .....43 .....45 .....47 .....49 C....51 .....53 .....55
```

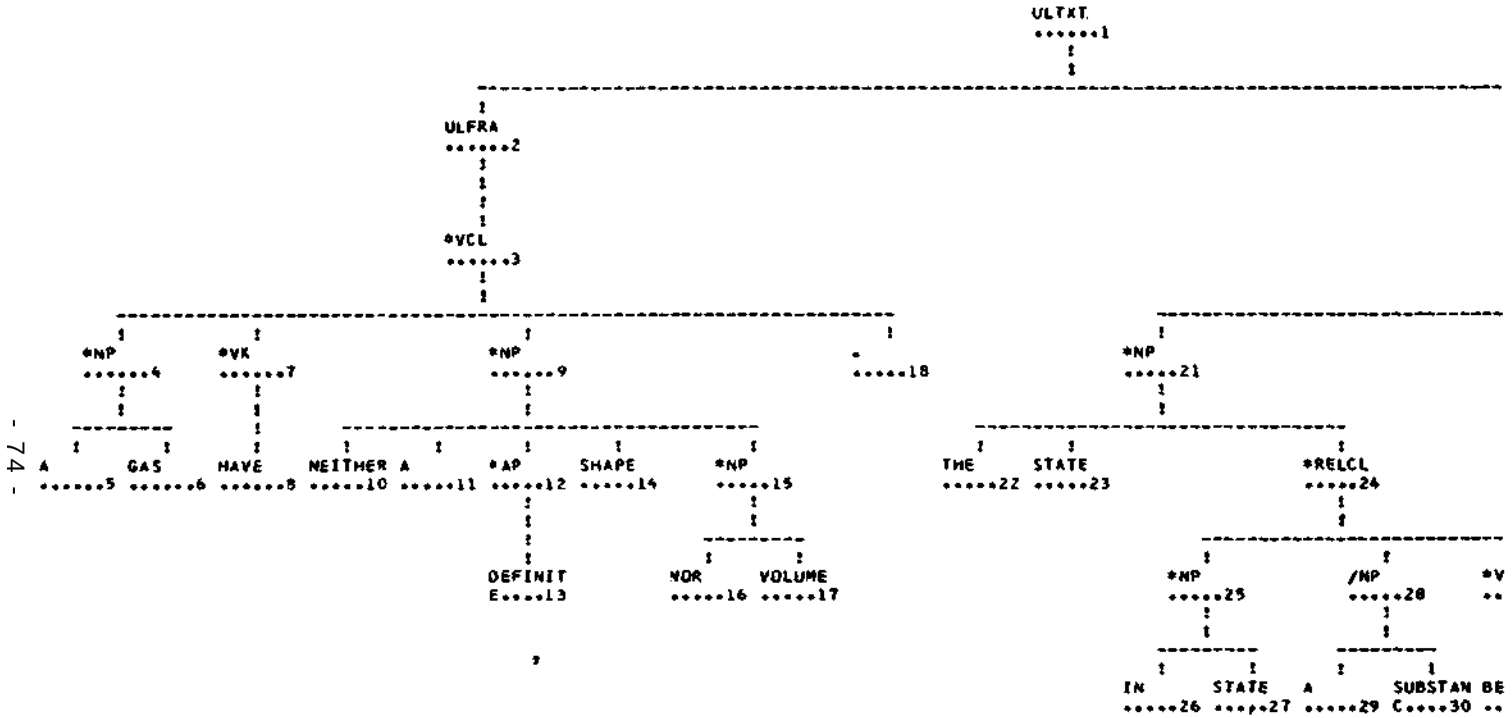
```
SOMMET 1  * *: UL('ULTXT').
SOMMET 2  * *: UL('ULFRA').
SOMMET 3  * *: UL('ULOCC').
SOMMET 4  *A*: UL('A'),STATE(1),CAT(D),SUBD(ART),NUM(SIN).
SOMMET 5  * *: UL('ULOCC').
SOMMET 6  *GAS*: UL('GAS'),STATE(3),CAT(N),SUBN(CN),NUM(SIN),NEND(2),SEM(CONC),SEMCO(SUBST).
SOMMET 7  * *: UL('ULOCC').
SOMMET 8  *HAS*: UL('HAVE'),STATE(3),CAT(V),SUBV(VB),NUM(SIN),TENSE(PRES),VL1(N,I).
SOMMET 9  * *: UL('ULOCC').
SOMMET 10  *NEITHER*: UL('NEITHER'),STATE(1),CAT(A,R),SUBA(ADV,NPMD),SUBR(UTF),RELN(HALF1).
SOMMET 11  * *: UL('ULOCC').
SOMMET 12  *A*: UL('A'),STATE(1),CAT(D),SUBD(ART),NUM(SIN).
SOMMET 13  * *: UL('ULOCC').
SOMMET 14  *DEFINITE*: UL('DEFINITE'),STATE(3),CAT(A),SUBA(ADJ),IMPERS(INPED),SUBJR(S).
SOMMET 15  * *: UL('ULOCC').
SOMMET 16  *SHAPE*: UL('SHAPE'),STATE(3),CAT(N),SUBN(CN),NUM(SIN),NEND(1),SEM(ABST).
SOMMET 17  * *: UL('ULOCC').
SOMMET 18  *NOR*: UL('NOR'),STATE(1),CAT(C),RELN(HALF1,HALF2).
SOMMET 19  * *: UL('ULOCC').
SOMMET 20  *VOLUME*: UL('VOLUME'),STATE(3),CAT(N),SUBN(CN),NUM(SIN),NEND(1),SEM(PHY).
SOMMET 21  * *: UL('ULOCC').
SOMMET 22  *.*: UL('.'),STATE(1),CAT(P).
SOMMET 23  * *: UL('ULFRA').
SOMMET 24  * *: UL('ULOCC').
SOMMET 25  *THE*: UL('THE'),STATE(1),CAT(D),SUBD(ART),NUM(SIN,PLU).
SOMMET 26  * *: UL('ULOCC').
SOMMET 27  *STATE*: UL('STATE'),STATE(3),CAT(N),SUBN(CN),NUM(SIN),NEND(1),SEM(ABST).
SOMMET 28  * *: UL('ULOCC').
SOMMET 29  *IN*: UL('IN'),STATE(1),CAT(S),SUBS(PREP),JPCL(IN),RS(LOCAL),UNSAFE(RS),VL1(IN).
SOMMET 30  * *: UL('ULOCC').
SOMMET 31  *WHICH*: UL('WHICH'),STATE(1),CAT(D,R),SUBD(INT),SUBR(REL,INT),NUM(SIN,PLU),AMB(1).
SOMMET 32  * *: UL('ULOCC').
SOMMET 33  *A*: UL('A'),STATE(1),CAT(D),SUBD(ART),NUM(SIN).
SOMMET 34  * *: UL('ULOCC').
SOMMET 35  *SUBSTANCE*: UL('SUBSTANCE'),STATE(3),CAT(N),SUBN(CN),NUM(SIN),NEND(1),SEM(CONC),SEMCO(SUBST).
SOMMET 36  * *: UL('ULOCC').
SOMMET 37  *IS*: UL('BE'),STATE(3),CAT(V),SUBV(VB),NUM(SIN),TENSE(PRES),TYPE(COP).
SOMMET 38  * *: UL('ULOCC').
SOMMET 39  *DEPENDS*: UL('DEPEND'),STATE(3),CAT(V),SUBV(VB),NUM(SIN),TENSE(PRES),VEND(1),SEMV(STATE),VL1(ON).
SOMMET 40  * *: UL('ULOCC').
SOMMET 41  *ON*: UL('ON'),STATE(1),CAT(S),SUBS(PREP),JPCL(ON),RS(LOCAL),VL1(ON).
SOMMET 42  * *: UL('ULOCC').
SOMMET 43  *THE*: UL('THE'),STATE(1),CAT(D),SUBD(ART),NUM(SIN,PLU).
SOMMET 44  * *: UL('ULOCC').
SOMMET 45  *ENERGY*: UL('ENERGY'),STATE(3),CAT(N),SUBN(CN),NUM(SIN),NEND(3),SEM(PHY).
SOMMET 46  * *: UL('ULOCC').
SOMMET 47  *THAT*: UL('THAT'),STATE(1),CAT(D,R,S),SUBD(DEM),SUBR(DEM,REL),SUBS(CONJ),NUM(SIN),AMB(1).
```

```
SOMMET 48  ' ': UL('ULOCC').
SOMMET 49  'THE': UL('THE'),STATE(1),CAT(D),SUBD(ART),NUM(SIN,PLU).
SOMMET 50  ' ': UL('ULOCC').
SOMMET 51  'SUBSTANCE': UL('SUBSTANCE'),STATE(3),CAT(N),SUBN(CN),NUM(SIN),NEND(1),SEM(CONC),SEMCO(SUBST).
SOMMET 52  ' ': UL('ULOCC').
SOMMET 53  'CONTAINS': UL('CONTAIN'),STATE(3),CAT(V),SUBV(VB),NUM(SIN),TENSE(PRES),VEND(1),SEMV(STATE),VL1(N).
SOMMET 54  ' ': UL('ULOCC').
SOMMET 55  '.': UL('.'),STATE(1),CAT(P).
```

*** GROUPE D'ETUDE POUR LA TRADUCTION AUTOMATIQUE ***
****

ANALYSE STRUCTURALE

RESULTAT DE L'EXECUTION, TEXTE : CHIMI RYADH CODE LANGUE : BXA


ULTXT
1
ULFRA
2
*VCL
3
*NP
4
*VK
7
*NP
9
18
*NP
21
A
5
GAS
6
HAVE
8
NEITHER A
10
A
11
*AP
12
SHAPE
14
*NP
15
THE
22
STATE
23
*RELCL
24
DEFINIT
E
13
NOR
16
VOLUME
17
*NP
25
/NP
28
*V
IN
26
STATE
27
A
29
SUBSTAN
C
30
BE

ULFRA
*VCL
*VK
DEPEND THE
*NP
ENERGY
*RELCL
*NP
*NP
*VK
ENERGY THE SUBSTAN CONTAIN

```
SOMMET 1   * *: UL('ULTXT').
SOMMET 2   * *: UL('ULFRA').
SOMMET 3   * *: UL('*VCL').SLOCK(1).LOCKZ(1).LOCK1(1).K(VCL).CAT(V).SUBV(VB).NUM(SIN).TENSE(PRES).
SOMMET 4   * *: UL('*NP').RL(ARGO).K(NP).SF(SUBJ).CAT(N).SUBN(CN).NUM(SIN).SEM(CONC).SEMCO(SUBST).VL1(N).
SOMMET 5   'A': UL('A').SF(DES).CAT(D).SUBD(ART).NUM(SIN).
SOMMET 6   'GAS': UL('GAS').SF(GOV).CAT(N).SUBN(CN).NUM(SIN).SEM(CONC).SEMCO(SUBST).
SOMMET 7   * *: UL('*VK').SLOCK(1).LOCKZ(1).LOCK1(1).K(VK).SF(GOV).CAT(V).SUBV(VB).NUM(SIN).TENSE(PRES).VL1(N).
SOMMET 8   'HAS': UL('HAVE').SLOCK(1).LOCKZ(1).LOCK1(1).SF(GOV).CAT(V).SUBV(VB).NUM(SIN).TENSE(PRES).VL1(N).
SOMMET 9   * *: UL('*NP').RL(ARG1).K(NP).SF(OBJ1).CAT(N).SUBN(CN).NUM(SIN).RELW(HALF1).SEM(ABST).VL1(N).
SOMMET 10   'NEITHER': UL('NEITHER').SF(REG).CAT(C).RELW(HALF1).
SOMMET 11   'A': UL('A').SF(DES).CAT(D).SUBD(ART).NUM(SIN).
SOMMET 12   * *: UL('*AP').RS(QUAL).K(AP).SF(ATG).CAT(A).SUBA(ADJ).IMPERS(IMPEO).SUBJR(S).
SOMMET 13   'DEFINITE': UL('DEFINITE').SF(GOV).CAT(A).SUBA(ADJ).SUBJR(S).
SOMMET 14   'SHAPE': UL('SHAPE').SF(GOV).CAT(N).SUBN(CN).NUM(SIN).SEM(ABST).
SOMMET 15   * *: UL('*NP').K(NP).SF(COORD).CAT(N).SUBN(CN).NUM(SIN).RELW(HALF2).SEM(PHY).VL1(N).
SOMMET 16   'NOR': UL('NOR').SF(REG).CAT(C).RELW(HALF2).
SOMMET 17   'VOLUME': UL('VOLUME').SF(GOV).CAT(N).SUBN(CN).NUM(SIN).SEM(PHY).
SOMMET 18   '.': UL('.').CAT(P).
SOMMET 19   * *: UL('ULFRA').
SOMMET 20   * *: UL('*VCL').SLOCK(1).LOCKZ(1).LOCK1(1).K(VCL).CAT(V).SUBV(VB).NUM(SIN).TENSE(PRES).SEMV(STATE).
SOMMET 21   * *: UL('*NP').RL(ARGO).K(NP).SF(SUBJ).CAT(N).SUBN(CN).NUM(SIN).SEM(ABST).VL1(N).
SOMMET 22   'THE': UL('THE').SF(DES).CAT(D).SUBD(ART).NUM(SIN).
SOMMET 23   'STATE': UL('STATE').SF(GOV).CAT(N).SUBN(CN).NUM(SIN).SEM(ABST).
SOMMET 24   * *: UL('*RELCL').RS(QUAL).UNSAFE(STR).REL(1).SLOCK(1).LOCKZ(1).K(RELCL).SF(ATG).CAT(V).SUBV(VB).
    NUM(SIN).TENSE(PRES).
SOMMET 25   * *: UL('*NP').RS(LOCAL).UNSAFE(RS).REL(1).K(NP).SF(CIRC).CAT(R).SUBR(REL).NUM(SIN).SEM(ABST).VL1(IN).
SOMMET 26   'IN': UL('IN').RS(LOCAL).UNSAFE(RS).SF(REG).CAT(S).SUBS(PREP).VL1(IN).JPCL(IN).
SOMMET 27   'WHICH': UL('STATE').REL(1).MCUL(SUBS).K(NP).SF(GOV).CAT(R).SUBR(REL).NUM(SIN).SEM(ABST).
SOMMET 28   * *: UL('/NP').RL(ARGO).K(NP).SF(SUBJ).CAT(N).SUBN(CN).NUM(SIN).SEM(CONC).SEMCO(SUBST).VL1(N).
SOMMET 29   'A': UL('A').SF(DES).CAT(D).SUBD(ART).NUM(SIN).
SOMMET 30   'SUBSTANCE': UL('SUBSTANCE').SF(GOV).CAT(N).SUBN(CN).NUM(SIN).SEM(CONC).SEMCO(SUBST).
SOMMET 31   * *: UL('*VK').RS(QUAL).SLOCK(1).LOCKZ(1).K(VK).SF(GOV).CAT(V).SUBV(VB).NUM(SIN).TYPE(COP).TENSE(PRES).
SOMMET 32   'IS': UL('BE').SLOCK(1).LOCKZ(1).SF(GOV).CAT(V).SUBV(VB).NUM(SIN).TYPE(COP).TENSE(PRES).
SOMMET 33   * *: UL('*VK').SLOCK(1).LOCKZ(1).LOCK1(1).K(VK).SF(GOV).CAT(V).SUBV(VB).NUM(SIN).TENSE(PRES).
    SEMV(STATE).VL1(ON).
SOMMET 34   'DEPENDS': UL('DEPEND').SLOCK(1).LOCK2(1).LOCK1(1).SF(GOV).CAT(V).SUBV(VB).NUM(SIN).TENSE(PRES).
    SEMV(STATE).VL1(ON).
SOMMET 35   * *: UL('*NP').RL(ARG1).K(NP).SF(OBJ1).CAT(N).SUBN(CN).NUM(SIN).SEM(PHY).VL1(ON).
SOMMET 36   'THE': UL('THE').SF(DES).CAT(D).SUBD(ART).NUM(SIN).
SOMMET 37   'ENERGY': UL('ENERGY').SF(GOV).CAT(N).SUBN(CN).NUM(SIN).SEM(PHY).
SOMMET 38   * *: UL('*RELCL').RS(QUAL).UNSAFE(STR).REL(1).SLOCK(1).LOCKZ(1).LOCK1(1).K(RELCL).SF(ATG).CAT(V).
    SUBV(VB).NUM(SIN).TENSE(PRES).SEMV(STATE).
SOMMET 39   * *: UL('*NP').REL(1).RL(ARG1).K(NP).SF(OBJ1).CAT(R).SUBR(REL).NUM(SIN).SEM(PHY).VL1(N).
SOMMET 40   'THAT': UL('ENERGY').REL(1).MCUL(SUBS).SF(GOV).CAT(R).SUBR(REL).NUM(SIN).SEM(PHY).
SOMMET 41   * *: UL('*NP').RL(ARGO).K(NP).SF(SUBJ).CAT(N).SUBN(CN).NUM(SIN).SEM(CONC).SEMCO(SUBST).VL1(N).
SOMMET 42   'THE': UL('THE').SF(DES).CAT(D).SUBD(ART).NUM(SIN).
SOMMET 43   'SUBSTANCE': UL('SUBSTANCE').SF(GOV).CAT(N).SUBN(CN).NUM(SIN).SEM(CONC).SEMCO(SUBST).
```

```
SOMMET 44  ' ': UL('*VK'),RS(QUAL),SLOCK(1),LOCK2(1),LOCK1(1),K(VK),SF(GOV),CAT(V),SUBV(VB),NUM(SIN),TENSE(PRES),
    SEMV(STATE),VL1(N).
SOMMET 45  'CONTAINS': UL('CONTAIN'),SLOCK(1),LOCK2(1),LOCK1(1),SF(GOV),CAT(V),SUBV(VB),NUM(SIN),TENSE(PRES),
    SEMV(STATE),VL1(N).
SOMMET 46  '.': UL('.'),CAT(P).
```

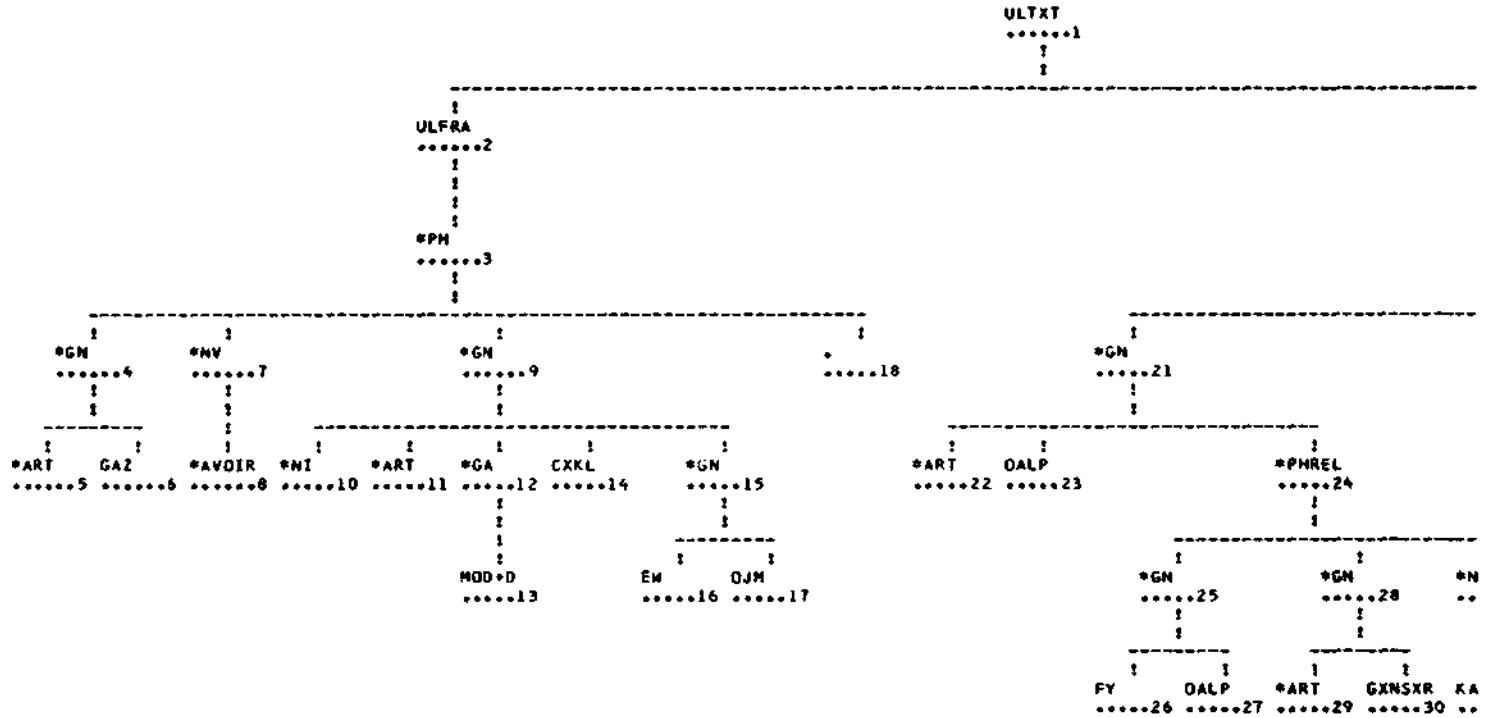
11 FEVRIER 1985 10H 41MN 29S

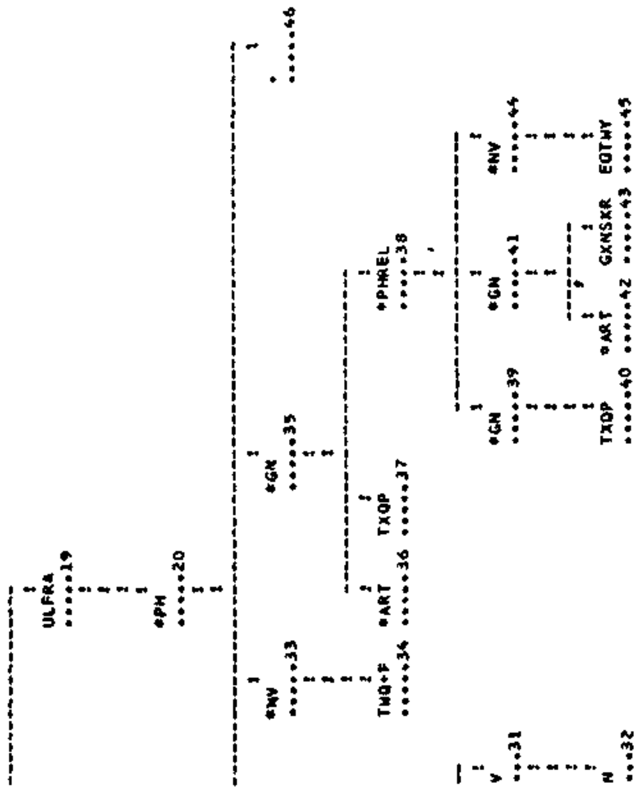
*** GROUPE D'ETUDE POUR LA TRADUCTION AUTOMATIQUE ***
****

TRANSFERT LEXICAL

RESULTAT DE L'EXECUTION, TEXTE : CHIMI RYADH

CODE LANGUE : BXA - ARX


ULTXT
ULFRA
*PH
*GN
*NV
*GN
*GN
*ART
GAZ
*AVOIR
*NI
*ART
*GA
CXKL
*GN
MOD*D
EW
OJM
*ART
OALP
*PHREL
*GN
*GN
FY
OALP
*ART
GXNSXR


ULFRA
*PH
*NV
*GN
TWQ+F
*ART
TXQP
*PHREL
*GN
*NV
TXQP
*ART
GXNSKR
EQTWY
V
N

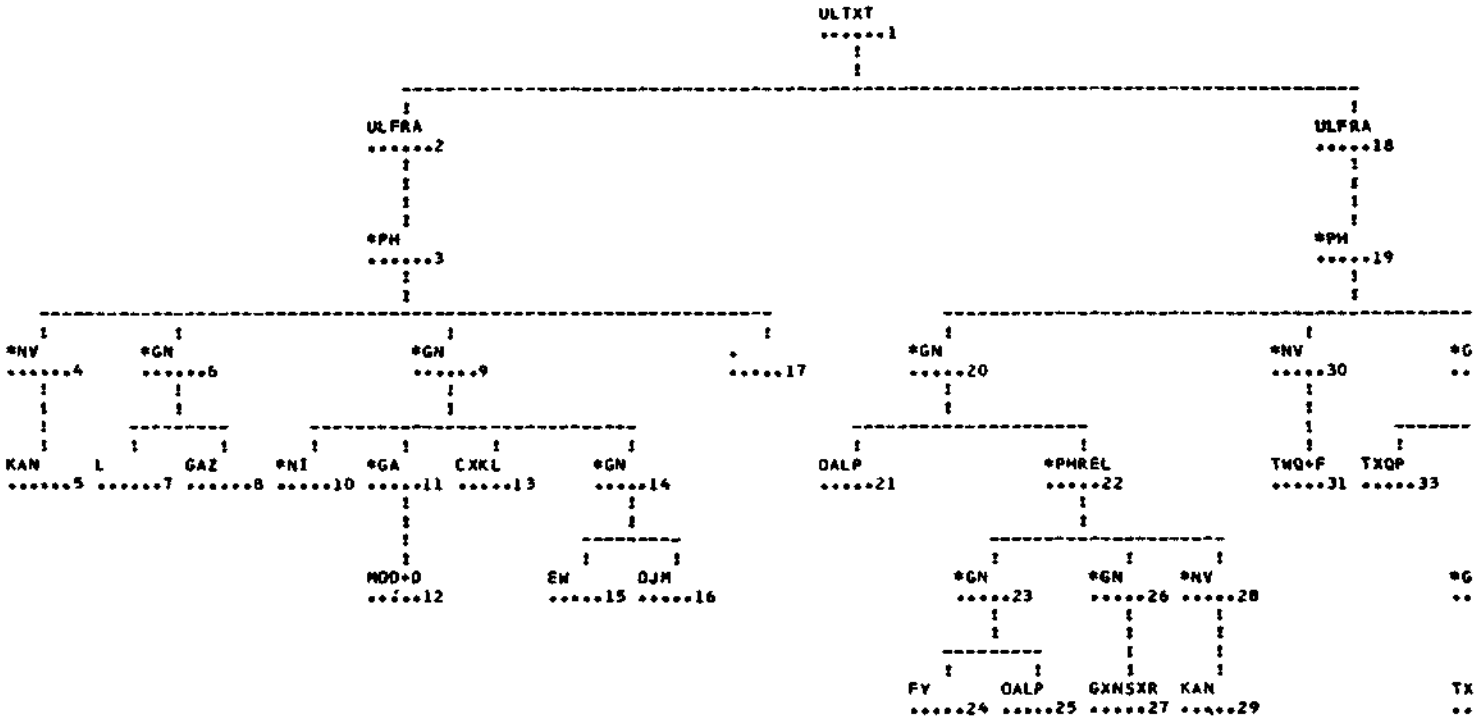
```
SOMMET 1  ' ': UL('ULTXT').
SOMMET 2  ' ': UL('ULFRA').
SOMMET 3  ' ': UL('*PH').SLOCK(1).LOCK1(1).LOCKZ(1).K(PH).CAT(V).SUBV(VB).TENSE(PRES).
SOMMET 4  ' ': UL('*GN').K(GN).SF(SJBJ).RL(ARGO).CAT(N).SEM(CONC).SEMCO(SUBST).
SOMMET 5  'A': UL('*ART').SF(DES).CAT(D).NUM(SIN).
SOMMET 6  'GAS': UL('GAZ').SF(GOV).CAT(N).GNR(MAS).NUM(SIN).SEM(CONC).SEMCO(SUBST).
SOMMET 7  ' ': UL('*NV').SLOCK(1).LOCK1(1).LOCKZ(1).K(NV).SF(GOV).TENSE(PRES).
SOMMET 8  'HAS': UL('*AVOIR').SLOCK(1).LOCK1(1).LOCKZ(1).SF(GOV).CAT(V).NUM(SIN).TYP(EXIST).
SOMMET 9  ' ': UL('*GN').K(GN).SF(OBJ1).RL(ARG1).CAT(N).SEM(ABST).
SOMMET 10  'NEITHER': UL('*NI').SF(REG).CAT(A).NI(1).SUBA(ADDEICT).
SOMMET 11  'A': UL('*ART').SF(DES).CAT(D).NUM(SIN).
SOMMET 12  ' ': UL('*GA').K(GA).SF(ATG).RS(QUAL).CAT(A).SUBA(ADJ).
SOMMET 13  'DEFINITE': UL('*ROD+D').SF(GOV).CAT(A).SUBA(ADJ).
SOMMET 14  'SHAPE': UL('CXKL').SF(GOV).CAT(N).GNR(MAS).NUM(SIN).SEM(ABST).
SOMMET 15  ' ': UL('*GN').K(GN).SF(COORD).CAT(N).SEM(PHY).
SOMMET 16  'NOR': UL('EW').SF(REG).CAT(A).NI(1).SUBA(ADDEICT).
SOMMET 17  'VOLUME': UL('OJM').SF(GOV).CAT(N).GNR(MAS).NUM(SIN).SEM(PHY).
SOMMET 18  '.': UL('.').CAT(P).
SOMMET 19  ' ': UL('ULFRA').
SOMMET 20  ' ': UL('*PH').SLOCK(1).LOCK1(1).LOCKZ(1).K(PH).CAT(V).SUBV(VB).SEMV(STATE).TENSE(PRES).
SOMMET 21  ' ': UL('*GN').K(GN).SF(SUBJ).RL(ARGO).CAT(N).SEM(ABST).
SOMMET 22  'THE': UL('*ART').SF(DES).CAT(D).DET(1).NUM(SIN).
SOMMET 23  'STATE': UL('QALP').SF(GOV).CAT(N).GNR(FEM).NUM(SIN).SEM(ABST).
SOMMET 24  ' ': UL('*PHREL').SLOCK(1).LOCKZ(1).REL(1).K(PHREL).SF(ATG).RS(QUAL).TENSE(PRES).
SOMMET 25  ' ': UL('*GN').REL(1).K(GN).SF(CIRC).RS(LOCAL).CAT(R).SUBR(REL).SEM(ABST).
SOMMET 26  'IN': UL('FY').SF(REG).CAT(S).SUBS(PREP).
SOMMET 27  'WHICH': UL('QALP').REL(1).SF(GOV).CAT(N).GNR(FEM).NUM(SIN).SEM(ABST).
SOMMET 28  ' ': UL('*GN').K(GN).SF(SUBJ).RL(ARGO).CAT(N).SEM(CONC).SEMCO(SUBST).
SOMMET 29  'A': UL('*ART').SF(DES).CAT(D).NUM(SIN).
SOMMET 30  'SUBSTANCE': UL('GXNSXR').SF(GOV).CAT(N).GNR(MAS).NUM(SIN).SEM(CONC).SEMCO(SUBST).
SOMMET 31  ' ': UL('*NV').SLOCK(1).LOCKZ(1).K(NV).SF(GOV).RS(QUAL).TENSE(PRES).
SOMMET 32  'IS': UL('KAN').SLOCK(1).LOCKZ(1).SF(GOV).CAT(V).NATM(KAL).NUM(SIN).TYP(EXIST).
SOMMET 33  ' ': UL('*NV').SLOCK(1).LOCK1(1).LOCKZ(1).K(NV).SF(GOV).SEMV(STATE).TENSE(PRES).
SOMMET 34  'DEPENDS': UL('TWQ+F').SLOCK(1).LOCK1(1).LOCKZ(1).SF(GOV).CAT(V).NUM(SIN).SEMV(STATE).VAL1(ALA).
SOMMET 35  ' ': UL('*GN').K(GN).SF(OBJ1).RL(ARG1).CAT(N).SEM(PHY).
SOMMET 36  'THE': UL('*ART').SF(DES).CAT(D).DET(1).NUM(SIN).
SOMMET 37  'ENERGY': UL('TXQP').SF(GOV).CAT(N).GNR(FEM).NUM(SIN).SEM(PHY).
SOMMET 38  ' ': UL('*PHREL').SLOCK(1).LOCK1(1).LOCKZ(1).REL(1).K(PHREL).SF(ATG).RS(QUAL).SEMV(STATE).TENSE(PRES).
SOMMET 39  ' ': UL('*GN').REL(1).K(GN).SF(OBJ1).RL(ARG1).CAT(R).SUBR(REL).SEM(PHY).
SOMMET 40  'THAT': UL('TXQP').REL(1).SF(GOV).CAT(N).GNR(FEM).NUM(SIN).SEM(PHY).
SOMMET 41  ' ': UL('*GN').K(GN).SF(SUBJ).RL(ARGO).CAT(N).SEM(CONC).SEMCO(SUBST).
SOMMET 42  'THE': UL('*ART').SF(DES).CAT(D).DET(1).NUM(SIN).
SOMMET 43  'SUBSTANCE': UL('GXNSXR').SF(GOV).CAT(N).GNR(MAS).NUM(SIN).SEM(CONC).SEMCO(SUBST).
SOMMET 44  ' ': UL('*NV').SLOCK(1).LOCK1(1).LOCKZ(1).K(NV).SF(GOV).RS(QUAL).SEMV(STATE).TENSE(PRES).
SOMMET 45  'CONTAINS': UL('EQTNY').SLOCK(1).LOCK1(1).LOCKZ(1).SF(GOV).CAT(V).NATM(RNY).HAMZ(DM).NUM(SIN).
    SEMV(STATE).VAL1(ALA).
SOMMET 46  '.': UL('.').CAT(P).
```

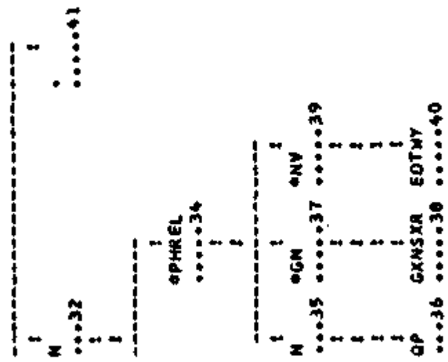
*** GROUPE D'ETUDE POUR LA TRADUCTION AUTOMATIQUE ***
****

TRANSFERT DE STRUCTURE

RESULTAT DE L'EXECUTION, TEXTE : CHIMI RYADH

CODE LANGUE : BXA - ARX


ULTXT
1
ULFRA
2
*PH
3
*NV
4
KAN
5
L
7
*GN
6
GAZ
8
*GN
9
*NI
10
*GA
11
MOD+0
12
CXKL
13
*GN
14
EH
15
OJH
16
*
17
ULFRA
18
*PH
19
*GN
20
OALP
21
*PHREL
22
*GN
23
FY
24
OALP
25
*GN
26
GXNSXR
27
*NV
28
KAN
29
*NV
30
TWQ+F
31
TXQP
33
*G
TX



```
SOMMET 1  ' ': UL('ULTXT').
SOMMET 2  ' ': UL('ULFRA').
SOMMET 3  ' ': UL('*PH'),SLOCK(1),LOCK1(1),LOCKZ(1),K(PH),GNRPH(PHNOM),CAT(V),SUBV(VB),TENSE(PRES),TEMPS(PRES),
    FORME(F2).
SOMMET 4  ' ': UL('*NV'),SLOCK(1),LOCK1(1),LOCKZ(1),K(NV),SF(GOV),SUBV(VB),TYP(EXIST),TENSE(PRES),TEMPS(PRES),
    FORME(F2).
SOMMET 5  'HAS': UL('KAN'),SLOCK(1),LOCK1(1),LOCKZ(1),SF(GOV),CAT(V),NB(SIN),TYP(EXIST).
SOMMET 6  ' ': UL('*GN'),K(GN),SF(ATSUBJ),RL(TRLO),CAT(N),DET(1),SEM(CONC),SEMCO(SUBST).
SOMMET 7  ' ': UL('L'),SF(REG),CAT(S),SUBS(PREP).
SOMMET 8  'GAS': UL('GAZ'),SF(GOV),CAT(N),GNR(MAS),NB(SIN),SEM(CONC),SEMCO(SUBST).
SOMMET 9  ' ': UL('*GN'),K(GN),SF(SUBJ),RL(ARGO),CAT(N),SEM(ABST).
SOMMET 10  'NEITHER': UL('*NI'),SF(REG),CAT(A),NI(1),SUBA(ADDEICT).
SOMMET 11  ' ': UL('*GA'),K(GA),SF(ATG),RS(QUAL),CAT(A),SUBA(ADJ).
SOMMET 12  'DEFINITE': UL('MOD+D'),SF(GOV),CAT(A),SUBA(ADJ).
SOMMET 13  'SHAPE': UL('CXKL'),SF(GOV),CAT(N),GNR(MAS),NB(SIN),SEM(ABST).
SOMMET 14  ' ': UL('*GN'),K(GN),SF(COORD),CAT(N),SEM(PHY).
SOMMET 15  'NOR': UL('EW'),SF(REG),CAT(A),NI(1),SUBA(ADDEICT).
SOMMET 16  'VOLUME': UL('OJM'),SF(GOV),CAT(N),GNR(MAS),NB(SIN),SEM(PHY).
SOMMET 17  '.': UL('-'),CAT(P).
SOMMET 18  ' ': UL('ULFRA').
SOMMET 19  ' ': UL('*PH'),SLOCK(1),LOCK1(1),LOCKZ(1),K(PH),CAT(V),SUBV(VB),SEMV(STATE),TENSE(PRES),TEMPS(PRES),
    FORME(F2).
SOMMET 20  ' ': UL('*GN'),K(GN),SF(SUBJ),RL(ARGO),CAT(N),DET(1),SEM(ABST).
SOMMET 21  'STATE': UL('QALP'),SF(GOV),CAT(N),GNR(FEM),NB(SIN),SEM(ABST).
SOMMET 22  ' ': UL('*PHREL'),REL(1),SLOCK(1),LOCKZ(1),K(PHREL),SF(ATG),RS(QUAL),SUBV(VB),TENSE(PRES),TEMPS(PRES),
    FORME(F2).
SOMMET 23  ' ': UL('*GN'),REL(1),K(GN),SF(CIRC),RS(LOCAL),CAT(R),SUBR(REL),DET(1),SEM(ABST).
SOMMET 24  'IN': UL('FY'),SF(REG),CAT(S),SUBS(PREP).
SOMMET 25  'WHICH': UL('QALP'),REL(1),SF(GOV),CAT(N),GNR(FEM),NB(SIN),SEM(ABST).
SOMMET 26  ' ': UL('*GN'),K(GN),SF(SUBJ),RL(ARGO),CAT(N),SEM(CONC),SEMCO(SUBST).
SOMMET 27  'SUBSTANCE': UL('GXNSXR'),SF(GOV),CAT(N),GNR(MAS),NB(SIN),SEM(CONC),SEMCO(SUBST).
SOMMET 28  ' ': UL('*NV'),SLOCK(1),LOCKZ(1),K(NV),SF(GOV),RS(QUAL),SUBV(VB),TENSE(PRES),TEMPS(PRES),FORME(F2).
SOMMET 29  'IS': UL('KAN'),SLOCK(1),LOCKZ(1),SF(GOV),CAT(V),NATM(KAL),NB(SIN),TYP(EXIST).
SOMMET 30  ' ': UL('*NV'),SLOCK(1),LOCK1(1),LOCKZ(1),K(NV),SF(GOV),SUBV(VB),SEMV(STATE),TENSE(PRES),TEMPS(PRES),
    FORME(F2).
SOMMET 31  'DEPENDS': UL('TWQ+F'),SLOCK(1),LOCK1(1),LOCKZ(1),SF(GOV),CAT(V),NB(SIN),SEMV(STATE),VAL1(ALA).
SOMMET 32  ' ': UL('*GN'),K(GN),SF(OBJ1),RL(ARG1),CAT(N),DET(1),SEM(PHY).
SOMMET 33  'ENERGY': UL('TXQP'),SF(GOV),CAT(N),GNR(FEM),NB(SIN),SEM(PHY).
SOMMET 34  ' ': UL('*PHREL'),REL(1),SLOCK(1),LOCK1(1),LOCKZ(1),K(PHREL),SF(ATG),RS(QUAL),SUBV(VB),SEMV(STATE),
    TENSE(PRES),TEMPS(PRES),FORME(F2).
SOMMET 35  ' ': UL('*GN'),REL(1),K(GN),SF(OBJ1),RL(ARG1),CAT(R),SUBR(REL),DET(1),SEM(PHY).
SOMMET 36  'THAT': UL('TXQP'),REL(1),SF(GOV),CAT(N),GNR(FEM),NB(SIN),SEM(PHY).
SOMMET 37  ' ': UL('*GN'),K(GN),SF(SUBJ),RL(ARGO),CAT(N),DET(1),SEMCO(SUBST).
SOMMET 38  'SUBSTANCE': UL('GXNSXR'),SF(GOV),CAT(N),GNR(MAS),NB(SIN),SEM(CONC),SEMCO(SUBST).
SOMMET 39  ' ': UL('*NV'),SLOCK(1),LOCK1(1),LOCKZ(1),K(NV),SF(GOV),RS(QUAL),SUBV(VB),SEMV(STATE),TENSE(PRES),
    TEMPS(PRES),FORME(F2).
SOMMET 40  'CONTAINS': UL('EOTWY'),SLOCK(1),LOCK1(1),LOCKZ(1),SF(GOV),CAT(V),NATM(RMY),HAMZ(OM),NB(SIN),
```

*** GROUPE D'ETUDE POUR LA TRADUCTION AUTOMATIQUE ***

TRANSFERT DE STRUCTURE

RESULTAT DE L'EXECUTION, TEXTE : CHIMI RYADH

CODE LANGUE : BXA - ARX

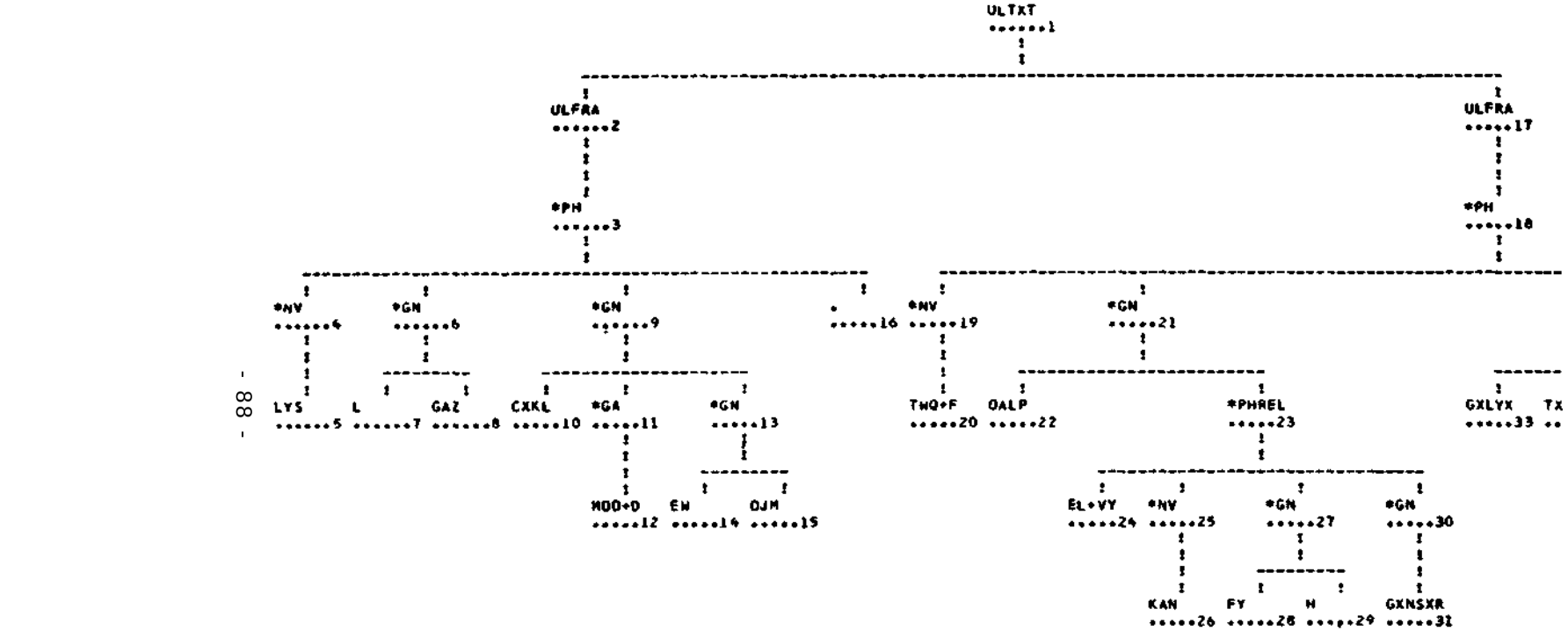
```
      SENV(STATE),VAL1(ALA).
SOMMET 1  *.*: UL('.'),CAT(P).
```

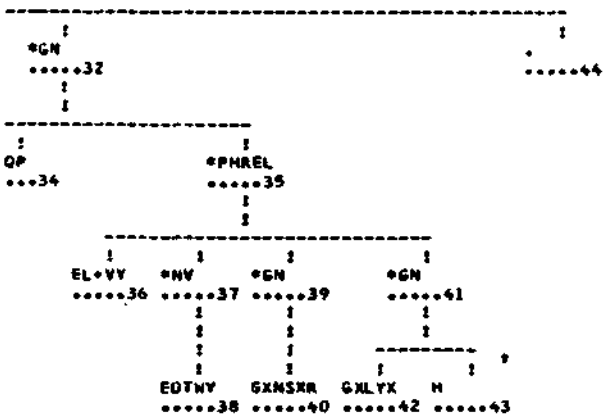
11 FEVRIER 1985 10H 41MN 29S

*** GROUPE D'ETUDE POUR LA TRADUCTION AUTOMATIQUE ***
****

GENERATION SYNTAXIQUE

RESULTAT DE L'EXECUTION• TEXTE : CHIMI RYADH CODE LANGUE : BXA - ARX PAGE 1


ULTXT
ULFRA
*PH
ULFRA
*PH
*NV
*GN
*GN
*NV
*GN
LYS
L
GAZ
CXKL
*GA
*GN
TWQ*F
OALP
*PHREL
GXLYX
TX
MOO*D
EW
OJM
EL*VY
*NV
*GN
*GN
KAN
FY
H
GXNSXR


*GN
32
QP
34
*PMREL
35
EL*VV
36
*NV
37
*GN
39
*GN
41
EQTWV
38
GXNSXR
40
GXLYX
42
H
43
44

```
SOMMET 1  * *: UL('ULTXT'),NOEUD(*4).
SOMMET 2  * *: UL('ULFRA').
SOMMET 3  * *: UL('*PH'),KLOCK(1),SFLOCK(2),K(PH),CAT(V),GNRPH(PHNOM),SUBV(VB),TEMPS(PRES),VOIX(ACT),FORME(F2).
SOMMET 4  * *: UL('*NV'),KLOCK(1),SFLOCK(1),K(NV),SF(GOV),SUBV(VB),TEMPS(PRES),VOIX(ACT),TYP(EXIST),FORME(F2).
SOMMET 5  'HAS': UL('LYS'),SFLOCK(1),PERS(3),SF(GOV),CAT(V),GNR(MAS),PREF(PY),TEMPS(PRES),VOIX(ACT),NB(SIN),
    TYP(EXIST),FORME(F2).
SOMMET 6  * *: UL('*GN'),CAS(GEN),KLOCK(1),SFLOCK(1),K(GN),SF(ATSUB),RL(TRLO),CAT(N),GNR(MAS),DET(DE),NB(SIN),
    SEM(CONC),SEMCO(SUBST).
SOMMET 7  * *: UL('L'),SFLOCK(1),SF(REG),CAT(S),SUBS(PREP).
SOMMET 8  'GAS': UL('GAZ'),CAS(GEN),KLOCK(1),SFLOCK(1),SF(GOV),CAT(N),GNR(MAS),DEF(DF),DET(DE),NB(SIN),SEM(CONC),
    SEMCO(SUBST).
SOMMET 9  * *: UL('*GN'),CAS(NOMIN),KLOCK(1),SFLOCK(1),K(GN),SF(SUBJ),RL(ARGO),CAT(N),GNR(MAS),NB(SIN),SEM(ABST).
SOMMET 10  'SHAPE': UL('GXKL'),CAS(NOMIN),KLOCK(1),SFLOCK(1),SF(GOV),CAT(N),GNR(MAS),NB(SIN),SEM(ABST).
SOMMET 11  * *: UL('*GA'),CAS(NOMIN),KLOCK(1),SFLOCK(1),K(GA),SF(ATG),RS(QUAL),CAT(A),GNR(MAS),NB(SIN),SUBA(ADJ).
SOMMET 12  'DEFINITE': UL('MOD=D'),CAS(NOMIN),SFLOCK(1),SF(GOV),CAT(A),GNR(MAS),NB(SIN),SUBA(ADJ).
SOMMET 13  * *: UL('*GN'),CAS(NOMIN),KLOCK(1),SFLOCK(1),K(GN),SF(COORD),CAT(N),GNR(MAS),NB(SIN),SEM(PHY).
SOMMET 14  'NOR': UL('EM'),SFLOCK(1),SF(REG),CAT(A),N1(1),SUBA(ADDICT).
SOMMET 15  'VOLUME': UL('OJM'),CAS(NOMIN),KLOCK(1),SFLOCK(1),SF(GOV),CAT(N),GNR(MAS),NB(SIN),SEM(PHY).
SOMMET 16  '.': UL('.'),SFLOCK(1),CAT(P).
SOMMET 17  * *: UL('ULFRA').
SOMMET 18  * *: UL('*PH'),KLOCK(1),SFLOCK(1),K(PH),CAT(V),GNRPH(PHVB),SUBV(VB),TEMPS(PRES),VOIX(ACT),SEMV(STATE),
    FORME(F2).
SOMMET 19  * *: UL('*NV'),KLOCK(1),SFLOCK(1),K(NV),SF(GOV),SUBV(VB),TEMPS(PRES),VOIX(ACT),SEMV(STATE),VAL1(ALA),
    FORME(F2).
SOMMET 20  'DEPENDS': UL('TNQ+F'),SFLOCK(1),PERS(3),SF(GOV),CAT(V),GNR(FEM),PREF(PT),TEMPS(PRES),VOIX(ACT),NB(SIN),
    SEMV(STATE),VAL1(ALA),FORME(F2).
SOMMET 21  * *: UL('*GN'),CAS(NOMIN),KLOCK(1),SFLOCK(1),K(GN),SF(SUBJ),RL(ARGO),CAT(N),GNR(FEM),DET(DE),NB(SIN),
    SEM(ABST).
SOMMET 22  'STATE': UL('OALP'),CAS(NOMIN),KLOCK(1),SFLOCK(1),SF(GOV),CAT(N),GNR(FEM),DEF(DF),DET(DE),NB(SIN),
    SEM(ABST).
SOMMET 23  * *: UL('*PHREL'),REL(1),KLOCK(1),SFLOCK(1),K(PHREL),SF(ATG),RS(QUAL),GNRPH(PHNOM),SUBV(VB),DET(DE),
    TEMPS(PRES),VOIX(ACT),FORME(F2).
SOMMET 24  * *: UL('EL+VY'),CAT(R),SUBR(REL),GNR(FEM),NB(SIN).
SOMMET 25  * *: UL('*NV'),SFLOCK(1),K(NV),SF(GOV),RS(QUAL),SUBV(VB),TEMPS(PRES),VOIX(ACT),TYP(EXIST),FORME(F2).
SOMMET 26  'IS': UL('KAN'),PERS(3),SF(GOV),CAT(V),GNR(MAS),NATM(KAL),PREF(PY),TRCAR(AW),TEMPS(PRES),VOIX(ACT),
    NB(SIN),TYP(EXIST),FORME(F2).
SOMMET 27  * *: UL('*GN'),CAS(GEN),REL(1),KLOCK(1),SFLOCK(1),K(GN),SF(ATSUB),RS(LOCAL),CAT(R),SUBR(REL),GNR(FEM),
    DET(DE),NB(SIN),SEM(ABST).
SOMMET 28  'IN': UL('FY'),SFLOCK(1),SF(REG),CAT(S),SUBS(PREP).
SOMMET 29  'WHICH': UL('H'),CAS(GEN),REL(1),KLOCK(1),SFLOCK(1),SF(GOV),CAT(R),SUBR(PERS),GNR(FEM),DEF(DF),DET(DE),
    NB(SIN),SEM(ABST).
SOMMET 30  * *: UL('*GN'),CAS(NOMIN),KLOCK(1),SFLOCK(1),K(GN),SF(SUBJ),RL(ARGO),CAT(N),GNR(MAS),NB(SIN),SEM(CONC),
    SEMCO(SUBST).
SOMMET 31  'SUBSTANCE': UL('GXNSXR'),CAS(NOMIN),KLOCK(1),SFLOCK(1),SF(GOV),CAT(N),GNR(MAS),NB(SIN),SEM(CONC),
    SEMCO(SUBST).
SOMMET 32  * *: UL('*GN'),CAS(GEN),KLOCK(1),SFLOCK(1),K(GN),SF(OBJ1),RL(ARG1),CAT(N),GNR(FEM),DET(DE),NB(SIN),
    SEM(PHY),VAL1(ALA).
```

```
SOMMET 33  ' ': UL('GXLYX'),CAT(S),SUBS(PREP).
SOMMET 34  'ENERGY': UL('TXQP'),CAS(GEN),KLOCK(1),SFLOCK(1),SF(GOV),CAT(N),GNR(FEM),DEF(DF),DET(DE),NB(SIN),
    SEM(PHY).
SOMMET 35  ' ': UL('*PHREL'),REL(1),KLOCK(1),SFLOCK(1),K(PHREL),SF(ATG),RS(QUAL),GNRPH(PHV8),SUBV(VB),DET(DE),
    TEMPS(PRES),VOIX(ACT),SEMV(STATE),FORME(F2).
SOMMET 36  ' ': UL('EL+VY'),CAT(R),SUBR(REL),GNR(FEM),NB(SIN).
SOMMET 37  ' ': UL('*NV'),SFLOCK(1),K(NV),SF(GOV),RS(QUAL),SUBV(VB),TEMPS(PRES),VOIX(ACT),SEMV(STATE),VAL1(ALA),
    FORME(F2).
SOMMET 38  'CONTAINS': UL('EQTWY'),PERS(3),SF(GOV),CAT(V),GNR(MAS),NATN(RNY),HAMZ(OM),PREF(PY),TEMPS(PRES),
    VOIX(ACT),NB(SIN),SEMV(STATE),VAL1(ALA),FORME(F2).
SOMMET 39  ' ': UL('*GN'),CAS(NOMIN),KLOCK(1),SFLOCK(1),K(GN),SF(SUBJ),RL(ARGO),CAT(N),GNR(MAS),DET(DE),NB(SIN),
    SEM(CONC),SEMCO(SUBST).
SOMMET 40  'SUBSTANCE': UL('GXNSXR'),CAS(NOMIN),KLOCK(1),SFLOCK(1),SF(GOV),CAT(N),GNR(MAS),DEF(DF),DET(DE),NB(SIN),
    SEM(CONC),SEMCO(SUBST).
SOMMET 41  ' ': UL('*GN'),CAS(GEN),REL(1),KLOCK(1),SFLOCK(1),K(GN),SF(OBJ1),RL(ARG1),CAT(R),SUBR(REL),GNR(FEM),
    DET(DE),NB(SIN),SEM(PHY),VAL1(ALA).
SOMMET 42  ' ': UL('GXLYX'),CAT(S),SUBS(PREP).
SOMMET 43  'THAT': UL('H'),CAS(GEN),REL(1),KLOCK(1),SFLOCK(1),SF(GOV),CAT(R),SUBR(PERS),GNR(FEM),DEF(DF),DET(DE),
    NB(SIN),SEM(PHY).
SOMMET 44  '.': UL('.'),SFLOCK(1),CAT(P).
```

CHIMI RYADH 11 FEVRIER 1985 13H 9MN 19S

LANGUES DE TRAITEMENT: BXA-ARX

-- TEXTE ORIGINE --

a gas has neither a definite shape nor volume. the state in which a substance is depends on the energy that the substance contains.

-- TEXTE TRADUIT --

----- (TRADUCTION DU 11 FEVRIER 1985 13H 09MN 08S) -----
VERSIONS : (A : 31/12/84 ; T : 2/02/85 ; G : 31/01/85)

lys llgaz cxkiux mod+dux e3w ojmux. ttwq+f aloalp al+ty ykwn fyha gxnsxrux gxiy altxaqp al+ty yotwy algxnsxr gxiyha.

ليس للغاز مشكل محدد أو حجم

تتوقّف الحالة التي يكون فيها عنصر على الطاقة التي يحتوي العنصر عليها.

References

Chauche (1974) "Transducteures et arborescences. Etude et réalisation de systèmes appliqués aux grammaires transformationnelles".
Thèse d'Etat, Grenoble, décembre 1974.

Vauquois (1975) "La Traduction Automatique à Grenoble".
Document de Linguistique Quantitative 29, Dunod, Paris, 1975.

Boitet (1976)"Un essai de réponse à quelques questions théoriques at pratiques liées à la Traduction Automatique. Définition d'un système prototype".
Thèse d'Etat, Grenoble, avril 1976.

Boitet, Guillaume, Quezel-Ambrunaz (1978) "Manipulation d'arborescences et parallélisme: le système ROBRA".
Proc. of COLING-80, Bergen.

Vauqois (1979) "Aspects of automatic translation in 1979".
IBM-Japan, Scientific Program.

Boitet, Nedobejkine (1981) "Recent developments in Russian-French Machine Translation at Grenoble".
Linguistics 19, 199-271, 1981.

Verastgegui 1982 "Etude du parallélisme appliqué à la traduction automatisée par ordinateur. STAR-PALE: un système parallèle". Thése de Docteur Ingenieur, USMG & INPG, Grenoble, mai 1982.
See also Proc. of COLING-82, Prague.

Clemente-Salazar 1982 "Etudes et algorithems liés à une nouvelle structure des données en TA: Les E-Graphs". Thèse de Docteur Ingenieur, USMG & INPG, Grenoble, mai 1982.
See also Proc. of COLING-82, Prague.

Boitet, Guillaume, Quezel-Ambrunaz 1982 "ARIANE-78: an integrated environment for automated translation and human revision"
Proc. of COLING-82.

Chappuy 1983 "Formalisation de la description des niveaux d'interpretation des langages naturelles".
Thèse de 3ème cycle, Grenoble, juin 1983.

Vauquois 1983 "Automatic Translation". Proc. of the summer school. "The Computer and the Arabic Language". Ch. 9, Rabat, October, 1983.

Gerber 1984 "Etude des possibilités de cooperation entre un système fondé sur des techniques de compréhension implicite (système logico-syntaxique) et un système fondé sur des techniques de compréhension explicite (système expert)". Thèse de 3ème cycle, Grenoble, janvier 1984.

Boitet 1984 "Research and development on MT and related techniques at Grenoble University (GETA)".
Lugano Tutorial on Machine Translation, April 1984.

Mozota 1984 "Un formalisme d'expression sur la spécification du contrôle dans les systèmes de production".
Thèse de 3ème cycle, Grenoble, juin 1984.

Boitet, Gerber "Expert systems and other new techniques in MT".
Proc. of Coling-84, Stanford, 1984.

Bachut, Verastegui 1984 "Software tools for the environment of a computer-aided translation system."
Proc. of Coling-84, Stanford, 1984.

Vauquois, Boitet 1984 "Automated Translation at GETA".
Journal of ACL; special issue for M.T.

* * *