13
Ariane (GETA)

13.1 Historical background

Perhaps the most influential individual person in the history of post-ALPAC MT research was Bernard Vauquois, who led a team of researchers at the Universiy of Grenoble until his untimely death at the age of 56, in 1985. The Grenoble group started work on MT as early as 1960, and, even before the ALPAC report in 1966, was developing a system incorporating characteristics of what are now generally called 'second generation' systems. In particular, the Grenoble group, at first known as CETA (*Centre d'Etudes de la Traduction Automatique* 'Centre for the Study of Machine Translation'), concentrated on the development of formalisms for expressing linguistic information and the algorithms which implemented them, and on the idea of doing translation by computing a succession of 'linguistic representations'. The original CETA system, developed between 1960 and 1970, for three language pairs (into French from Russian, German and Japanese, though with most work being done on the Russian–French system) was an interlingua system. A change in computer facilities in 1971, among other reasons, led the Grenoble team (now renamed GETA: *Groupe d'Etudes pour la Traduction Automatique*) to rethink the design of their MT system and to design a 'transfer' system, officially known as Ariane, though often referred to simply as 'the GETA system'.

The principal research work continued to be done on Russian to French translation, although there was also substantial research on a German–French system which used the same generation programs as the Russian–French version. From time to time there were other languages investigated by researchers who came

to Grenoble: Portuguese, Malay, Japanese, and Chinese. GETA has consistently encouraged the training and establishment of MT projects throughout the world, notably in Southeast Asia, Japan and China. The most important practical application of Ariane came through involvement in the French national project launched in 1983. The aim of the Calliope project, as it was called, was the development of a French–English system for aeronautics and an English–French system for computer science and data processing. However, after a bright start the project ended in early 1987.

The GETA system is important not only because it is a good example of a 'second generation' system, but also because the research at Grenoble was very influential. Many of the Japanese MT systems are quite similar in design to the GETA system, especially Kyoto University's Mu system, which itself influenced several of the Japanese commercial systems. Early work on Eurotra was much influenced by Grenoble (see Chapter 14), as was later work at Saarbrücken; the TAUM group at Montreal collaborated closely with GETA, and many researchers starting work in the field after 1975 have taken GETA's work as a reference point.

The system has gone through a number of improvements and modifications: Ariane-78, Ariane-85 and the most recent version Ariane-G5. The long-term goal has been to develop an MT 'engine' as the foundation for multilingual translation. This chapter describes basically Ariane-78.

13.2 General description

Ariane is a transfer system with analysis and generation both split into morphological and syntactic modules. Transfer also has two phases: lexical and structural transfer. Computationally, Ariane is interesting in its use of special-purpose rule-writing formalisms for each of the modules, and a (theoretically) strict separation of linguistic and algorithmic knowledge at each stage. The system is truly multilingual in the sense that the formalisms and the algorithms which implement them are quite independent of the languages to be treated; also from a linguistic point of view the analysis and generation programs for a given language-pair can be re-used for different target or source languages respectively. The representation levels used in Ariane are also of interest from a linguistic point of view, in that they are multi-level structures which combine dependency relations and constituent structures, and contain both deep and superficial linguistic information.

Figure 13.1 shows the general architecture of the system. The figure shows source language analysis up the left-hand side, transfer across the top, and target language generation down the right. The lozenges represent intermediate data structures, while the solid square boxes characterise the linguistic processes. The inner shaded box represents the software system itself, with the separate software tools implementing rule-writing formalisms shown in oval boxes.

The flow of the translation process follows a standard linguistic stratification: the source text string of characters undergoes morphological analysis resulting in a flat labelled tree. Multilevel analysis gives an intermediate source structure, which then undergoes a two-stage transfer: in lexical transfer the source-language lexical

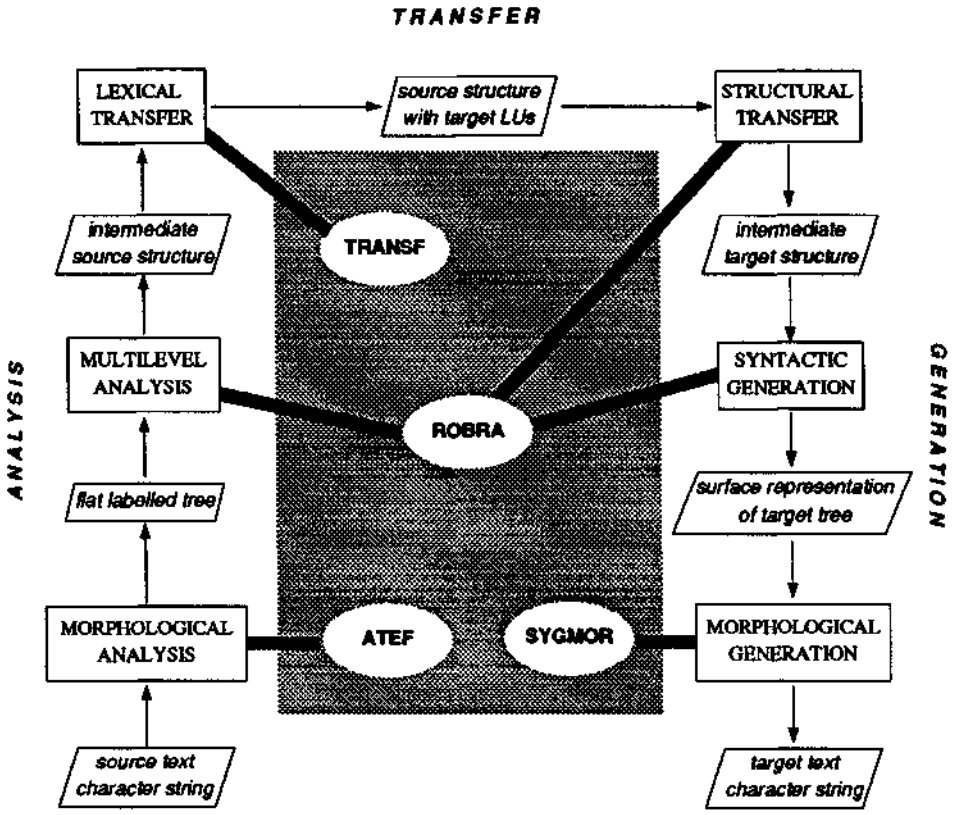


Figure 13.1 Configuration of the Ariane system

items, or 'lexical units' (LUs), are replaced with the corresponding target-language LUs giving a source-language structure with target-language LUs. Structural transfer produces an intermediate target structure which is input to syntactic generation and morphological generation.

Ariane has four rule-writing formalisms, effectively task-specific very-high-level programming languages, each with its associated implementation, the details of which are hidden from, and in theory of no interest to, the linguist writing the rules. The four software packages correspond to four different types of linguistic data-processing, in accordance with four different types of data structure and associated manipulation. This is an important feature of Ariane (distinguishing it, for example, from Météo) since each of the formalisms is specifically designed to facilitate a certain type of computation, with just enough (computational) power and flexibility for the allotted task.

ATEF is designed for morphological analysis, and maps strings of characters onto bundles of features arranged on a flat labelled tree. ROBRA (used for three of the six stages, in particular for the two most significant stages) is a very powerful software tool, which permits the description of **tree transductions**, i.e. it allows rules to take as input arbitrarily complex tree structures, to manipulate these

structures, and to output new tree structures. TRANSF is a slightly less powerful formalism (used for lexical transfer), which takes trees as input but does not have the power to rearrange them, only altering the labels on the branches. SYGMOR (used in morphological generation) has the function of converting labelled trees into character strings; it realises a finite state deterministic automaton, reflecting the lesser complexity of this phase.

We shall describe the Ariane formalisms in some detail (section 13.5) First however, we look more closely at the linguistic aspects of the GETA system, and in particular, the linguistic theory embodied in the analysis (and generation) procedures, and in the intermediate representation(s) used.

13.3 Multi-level representation

One of the most positive characteristics of the GETA approach is its incorporation of linguistic theory in both procedures and representations. Within the overall modular system design, each of the modules is motivated linguistically. At the heart of the linguistic approach taken by the GETA group is the representation which is aimed at as the output of analysis and, after lexical and structural transfer, the input to generation. This is the **interface structure**, where the influence of linguistic theories of the time (early 1970s) is particularly evident.

The interface structure is the linguistic representation resulting from a stratified linguistic analysis which we will describe below (section 13.4). As might be expected from the modular approach, there are various 'levels' of analysis. These are reflected in the linguistic representation, which combines simultaneously information at different levels: morphological, syntactic, and 'logico-semantic'. It is this 'multi-level' representational structure which is particularly noteworthy and which has been highly influential.

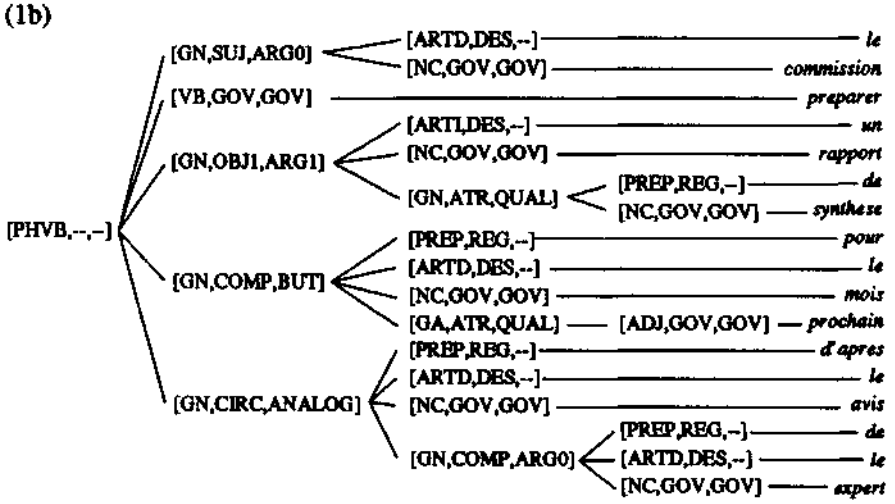
At the morphological level, the representation captures the difference between textual strings such as *dogs* and their morpho-lexical interpretation, e.g. lexical unit *dog*, noun plural. The syntactic level is a representation in terms of phrase structure constituents, such as noun phrase and verb group, together with (surface) syntactic relations, such as subject and object, complement and modifier. The so-called logico-semantic level is a deep syntactic representation showing dependency relations (heads and modifiers) of a logical nature, e.g. predicates and arguments or circumstantial elements with their semantic roles (goal, cause, location, etc.).

The representation for the sentence (1a) is shown in (1b).

(1a) *La Commission prépare un rapport de synthèse pour le mois prochain d'après les avis des experts.*
'The Commission is preparing a synthesis report for next month according to expert opinion'

It should be noticed first that at each terminal node in the tree is shown the lexical value of the node in a canonical form (i.e. *le* for all articles, singular form for nouns, infinitive for verb and so on). At both terminal and non-terminal nodes there appears (in square brackets) a three-valued label. This label gives the values at the three principal levels of morpho-syntax, surface syntax, and deep

(1b)



Key:

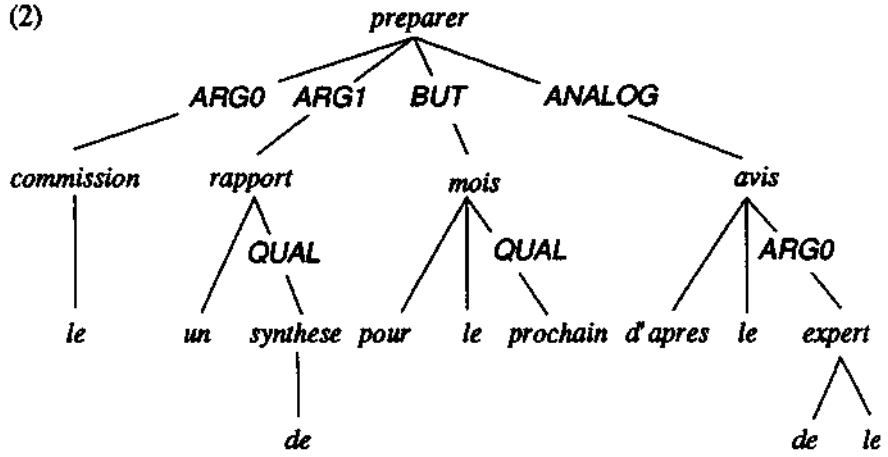
morphosyntactic labels: ADJ - adjective; ARTD - definite article; ARTI - indefinite article; GN -noun group; NC - common noun; PHVB - verbal sentence; PREP - preposition; VB - verb

syntactic labels: ATR - atribute; CIRC - circumstantial; COMP - complement; DES - designator; GOV - governor; OBJ1 - direct object; REG - 'regisseur' (head); SUJ - subject

logico-semantic labels: ANALOG - analogic; ARG0 - deep subject; ARG1 - deep object; BUT - goal; GOV - governor; QUAL - qualifier

syntax, i.e. the values of three principal features, MS (morpho-syntax), SF (syntactic function) and LS (logico-semantics). At each node there is also a large number of other feature–value pairs giving further pieces of information (not illustrated here), including lexically derived information such as grammatical gender, and morphologically derived information such as number, tense and so on.

The structure itself is rather flat from a typical phrase structure point of view, in that there are few intermediate constituents. For example, there are no separate prepositional phrases since prepositions are treated as parts of nominal groups. An essential feature to be noticed is that, looking from the top of the tree (the left-hand side in this diagram) each non-terminal node has exactly one daughter whose syntactic and deep syntactic label is GOV. In this way, the representation is not just a phrase structure tree but also, simultaneously, a dependency tree with 'lowered' governors. This can be demonstrated by reconstructing the tree, taking each governor as the root and its sisters as dependents, to give the dependency representation (2), with logico-semantic functions also shown where appropriate. Notice that the logical function of *experts* is ARG0, i.e. the 'deep subject' of *avis* ('opinion') treated as if it were a nominalized verb.



13.4 Linguistic processes

13.4.1 Morphological analysis

The linguistic processes involved in arriving at an interface representation such as (1b) are roughly as follows. First, the text is subjected to a morphological analysis phase, using rules written in the ATEF formalism (described below). This phase gives the labellings for each terminal node, based on the combination of dictionary look-up and string segmentation. Often, the results of morphological analysis are ambiguous. The labellings are attribute–value pairs, as in (3) for our example sentence (1a).

The output from morphological analysis is a (set of) flat labelled trees, where each minimal tree consists of a single dummy parent node dominating as many daughter nodes as there are words in the sentence, but where there are (as yet) no intermediate constituents. Each such flat tree will contain one reading of any ambiguous items, so there will be as many outputs as there are combinations of possible morphological analyses of the individual words.

13.4.2 Multi-level analysis

The next phase is the multi-level analysis, the most complex part of the system and employing the powerful ROBRA formalism (described in more detail in section 13.5.2 below). It involves a combination of 'parsing' to build an initial syntactic tree structure, and a kind of 'transformational' stage where surface syntactic tree structures are mapped onto deeper representations, where, for example, discontinuous lexical units are 'joined up' and surface syntactic functions like subject and object are replaced by logico-semantic relations.

The 'parsing' phase involves the application of rules which take sequences of LUs and build intermediate constituents. So, for example, a rule to build a GN

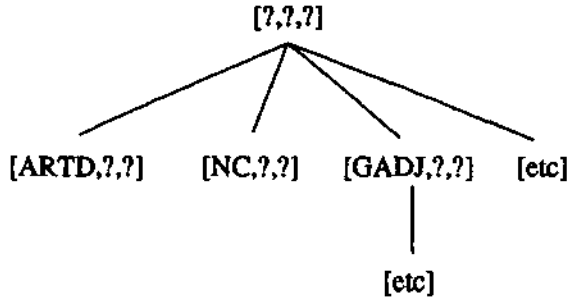
(3)

la	LU="LE", CAT=ARTD, GNR=FEM, NBR=SG
la	LU="LUI", CAT=PRON, GNR=FEM, NBR=SG, CASE=OBJ
commission	LU="COMMISSION", CAT=NC, GNR=FEM, NBR=SG, SEMFEAT=etc
prépare	LU="PREPARER", CAT=VB, NBR=SG, PERS=1/3, TNS=PRES
un	LU="UN", CAT=ARTI, GNR=MASC, NBR=SG
un	LU="UN", CAT=CARD
rapport	LU="RAPPORT", CAT=NC, GNR=MASC, NBR=SG, SEMFEAT=etc
de	LU="DE", CAT=PREP
synthèse	LU="SYNTHESE", CAT=NC, GNR=MASC, NBR=SG, SEMFEAT=etc
pour	LU="POUR", CAT=PREP
le	LU="LE", CAT=ARTD, GNR=MASC, NBR=SG
le	LU="LUI", CAT=PRON, GNR=MASC, NBR=SG, CASE=OBJ
mois	LU="MOIS", CAT=NC, GNR=MASC, NBR=SG, SEMFEAT=etc
prochain	LU="PROCHAIN", CAT=ADJ, GNR=MASC, NBR=SG
d'après	LU="D'APRES", CAT=PREP
les	LU="LE", CAT=ARTD, GNR=M/F, NBR=PL
les	LU="LUI", CAT=PRON, GNR=M/F, NBR=PL, CASE=OBJ
avis	LU="AVIS", CAT=NC, GNR=MASC, NBR=S/P, SEMFEAT=etc
des	LU="DE", CAT=PREP LU="LE", CAT=ARTD, GNR=M/F, NBR=PL
experts	LU="EXPERT", CAT=NC, GNR=MASC, NBR=PL, SEMFEAT=etc

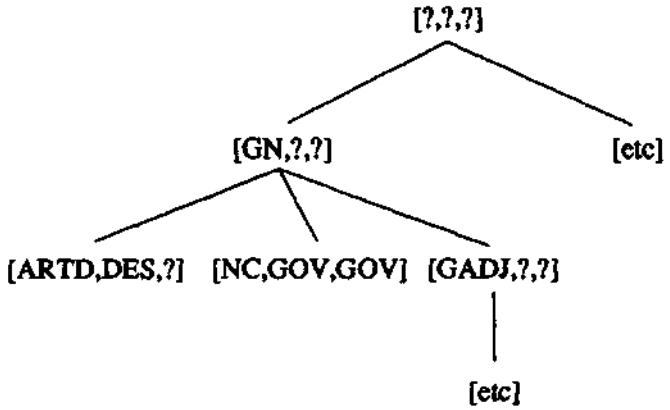
Key: CASE case: OBJ - object
CAT category: ADJ - adjective; ARTD - definite article; ARTI - indefinite article; CARD - cardinal number; NC - common noun; PREP - preposition; PRON - pronoun; VB - verb
GNR gender: FEM - feminine; MASC - masculine; M/F - either masculine or feminine
LU lexical unit
NBR number: PL - plural; SG - singular; S/P - either singular or plural
PERS person: 1/3 - either 1st or 3rd person
SEMFEAT semantic features

(noun group) node might look for a sequence of article, noun, adjective. Roughly speaking, such a rule might look for a tree structure as in (4a) and produce a structure as in (4b), having introduced an intermediate GN node and filled in some syntactic functions. The '?' sign indicates a value not yet specified.

(4a)



(4b)

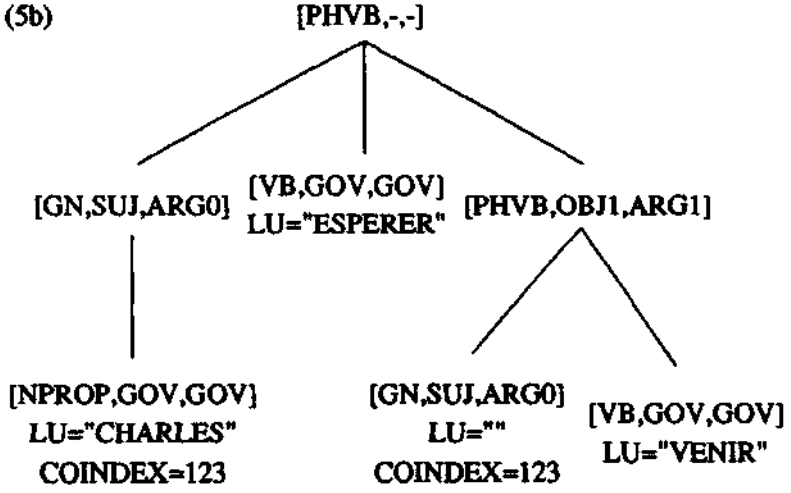


Because ROBRA is a tree transduction formalism, the rules do not have to be simple re-write rules as found in a standard phrase structure grammar: constraints such as number and gender agreement can be easily built into the rules. The rule sketched above, for example, would be conditioned by the compatibility of the values for features like NBR and GNR on the apprropriate nodes.

Alongside the analysis of phrase structure, rules which refer to the internal structures of already constructed constituents operate, reassigning the values of features or rearranging the shapes of tree structures. This activity is part of the analysis of syntactic and logico-semantic functions, and may, for example, depend on the application of subcategorization information relating to GOV nodes. Sometimes, nodes are deleted or inserted, as with the representation of a sentence like (5a), where a dummy deep subject is introduced into the structure for the embedded sentence (5b), or with sentence (6a) where the surface subject of the verb has no true logical role (6b).

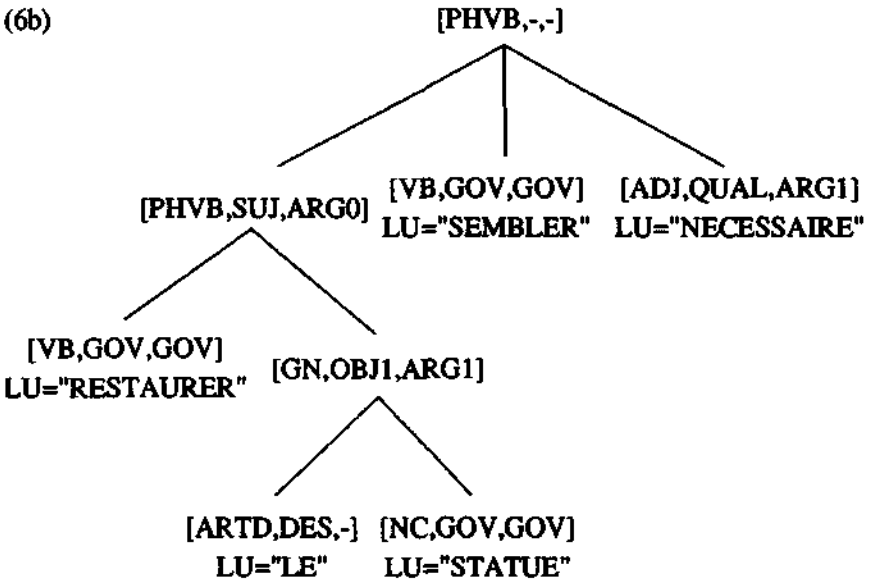
(5a) *Charles espère venir.* 'Charles hopes to come'

(5b)



(6a) *Il semble nécessaire de restaurer le statue.*
'It seems necessary to repair the statue'

(6b)



One of the major justifications for the multi-level type of representation is that it allows a 'safety net' approach to translation. If something goes wrong in the later stages of analysis, there will probably be a well-formed syntactic tree structure which can be passed to generation and from which some kind of translation can be produced, even if of a lesser quality than desirable.

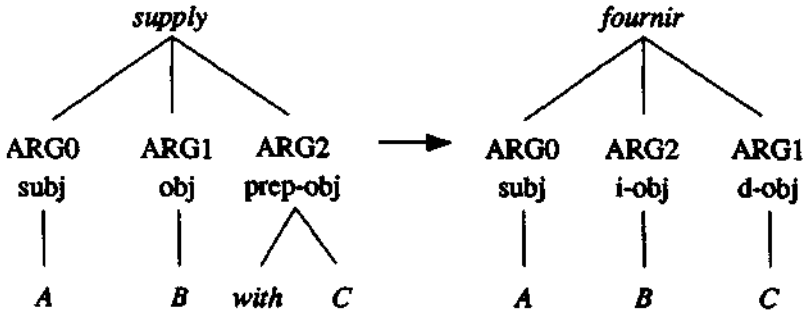
13.4.3 Transfer and generation

The output of multi-level analysis is a tree structure showing dependency and phrase structure relations, and indicating logico-semantic as well as surface syntactic functions. The next stage of the translation process is lexical transfer, in which source language LUs are replaced by corresponding target language items. Often, the choice of target LU depends on the surrounding context. In the case of example (6), there could be a straightforward lexical transfer rule for *statue* to give the corresponding English word; on the other hand, there might be several different translations for *restaurer*, including *restore*, *repair* and *refresh*, depending on the nature of its object, so in this case the rule might stipulate the subtree showing the verb with its ARG1, and specify the semantic features attached to the GOV of the ARG1.

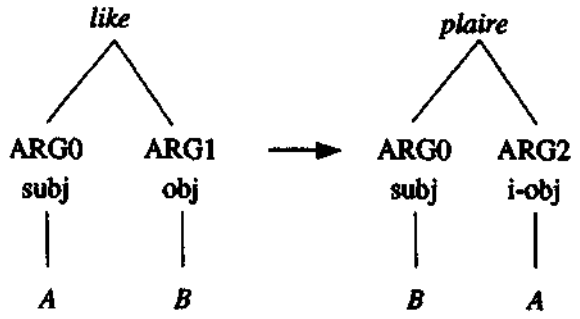
The structure is passed to structural transfer which makes the necessary structural alterations, including renumbering ARGs, introducing or deleting case-marking prepositions, and, quite often, restructuring the tree. Input is source-language structures (with target-language LUs already inserted), while the

output should be the corresponding target-language (deep) structures. The tree transduction formalism ROBRA is again involved, for manipulations such as changes in argument notation (7) and (8), and the more complex tree structure changes in (9).

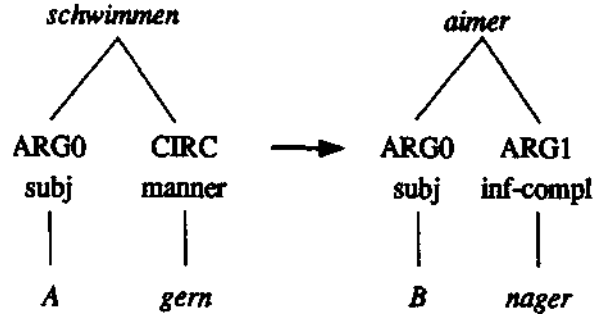
(7) A supplies B with C → *A fournit C à B*



(8) A likes B → *B plaît à A*



(9) *A schwimmt gern* → *A aime nager* ('A likes swimming')



In generation, the next stage of 'syntactic generation' involves the assignment of surface structure labels, that is, the choice of surface subjects and objects, the selection of appropriate verbal auxiliaries and so on, the rearrangement of word order, and the setting of values for morphological variables (e.g. number and gender agreements). It involves again the restructuring of trees and hence the use of the ROBRA formalism.

Morphological generation is the final stage of the translation process. The associated rule-writing formalism, SYGMOR, has the function of converting the labelled trees output by the syntactic generation phase into character strings, including punctuation. First trees are transformed into strings of pairs of values and attributes, and then these are output as surface strings.

13.5 Rule-writing formalisms

As already mentioned, the four software packages of Ariane correspond to four different types of linguistic data-processing: strings to trees (ATEF), trees to trees (ROBRA and TRANSF), and trees to strings (SYGMOR).

The underlying computational structure of Ariane is a 'production system' (cf section 3.8.6). The declarative aspect is strictly separated from the procedural aspect, i.e. the linguistic knowledge to be expressed is separated from the knowledge of how that lingustic knowledge is to be applied. It will be recalled that a production system consists of a data structure, a rule set and an interpreter. In this case, the data structure is the linguistic representation, implemented as a chart with labelled arcs indicating tree structures. The rule set is, of course, the set of linguistic rules written in the formalisms. And the interpreter is the underlying computer program which implements the rule-writing formalism.

The system as a whole works like a production system in that (at least in theory) the linguist writes the rules as productions. The linguist does not describe explicitly the translation processes, but rather writes linguistic rules which describe the configurations to be searched for in the data structure (i.e. tree structures, or sequences of arcs) and what changes are to be made (i.e. what new structures are to be added to the chart) if the configurations are found. It is left to the software to determine which rules are applicable to the data structure at any time, to apply the rules, and to repeat the cycle. Likewise, the software will incorporate methods of resolving rule conflicts, of knowing when all the applicable rules have been applied and when the process has terminated. In theory, none of this needs to be stated explicitly by the linguist.

In order to improve the efficiency of the production system, the rules can be divided by linguists into smaller 'subgrammars', which are effectively self-contained production systems. To a certain extent, the linguist can indicate which subgrammars should come into play at which point in the processing, thus permitting some procedural control over the flow of processing. In this respect, the separation of algorithms and linguistic information is compromised, although, it should be stressed, within the subgrammars the pure production system architecture is respected.

However, MT processes are often too complicated for a pure production system architecture, and in practice linguists at GETA have been forced to find ways of introducing procedural controls into the declarative mechanism for the sake of efficiency and practicality. In addition, although the chart is an excellent data structure for representing ambiguity, rules can refer only to single arcs and not to bundles of arcs spanning the same set of nodes. This means that it is not possible to refer explicitly to ambiguous structures, and linguists must rely on the rule set and the way the interpreter implements the rules to disambiguate ambiguous sequences.

13.5.1 ATEF

ATEF stands for *Analyse de textes en état fini* ('finite state text analysis') and its task is the mapping of strings of characters onto bundles of features arranged on a flat labelled tree, as in (3) above. The rules written in the ATEF formalism operate roughly by stipulating string segmentations and corresponding feature assignments. The application of a rule is restricted by conditions relating to information found in the lexical entries for the substrings. The feature assignments are given either in absolute terms or with reference to the lexicon. For example, the rule applying to *condensed* and the lexical entry in (10) is approximately as in (11) (the exact formalism has been simplified for the purposes of illustration):

(10)
```
LU=CONDENSE, CAT=V, CONJ=NO, TNS=PAST, VOX=PAS
```

(11)
```
$X + "D" ==
        [LU=$X, CAT=V, CONJ=NO, TNS=PAST, VOX=PAS] /
        CAT($X)=V and end($X)="E"
```

where $X is a string variable ending in *e* and is in the lexicon with category (CAT) verb (V).

ATEF was innovative as a tool for computational morphology in many ways, especially in listing both stems and endings in the lexicon. So for example, a rule something like (12) might deal simultaneously with both present and past participles, assuming that the respective lexical entries for the strings *d* and *ing* as verbal suffixes would give the appropriate values for TNS and VOX.

(12)
```
$X + $Y ==
   [LU=$X, CAT=V, CONJ=NO, TNS=TNS($Y), VOX=VOX($Y)]
    if CAT($X)=V and end($X)="E" and CAT($Y)=VBSUFX
```

Another feature is the possibility in ATEF of 'string transformations' as part of rules, allowing for substrings to be replaced by other substrings. This is important when morphological rules are applied sequentially, as in the treatment of English derivational morphology. An illustration is the word *unhappiness*. First the suffix *-ness* is identified as one for deriving nouns from adjectives; then the *-i* ending is changed to give *unhappy*, and lastly a rule deals with the prefix *un-*. By having a rule to change *i* to *y* (reflecting a regular English morphophonemic pattern), there is no need to include the stem *happi-* in the lexicon, which would duplicate information associated with *happy*.

13.5.2 ROBRA

The heart of the GETA system is the ROBRA rule-writing formalism. ROBRA is used for three of the six stages that make up the translation process, and in particular for the two most significant stages, namely multi-level analysis and structural transfer. As we have seen, these stages involve complex 'tree transductions'.

In this section we look in detail at an actual example of a rule written in the ROBRA formalism: a rule for compound nouns. In this way, it should be possible to get a clearer idea of the kinds of linguistic generalizations that the rule formalism allows or encourages, as well as some indication of the nature of the formalism. In discussing the example rule, we will also indicate some of the other features available in the formalism.

Figure 13.2 shows the rule (slightly simplified) as it would appear to the linguist. Line numbers have been added to facilitate discussion.

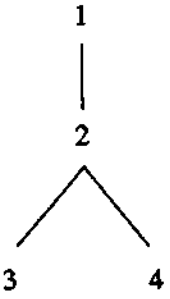
```
 1      COMPN: 1(2(3,4)) /
 2
 3      CAT(3) -E- N -ET-
 4      (CAT(4) -E- N -OU- SUBD(4) -E- CARD)
 5      -OU-
 6      CAT(3) -E- PREF -ET- CAT(4) -E- N
 7
 8      ==
 9
10      1(3,4) /*<--2/
11      4:4,
12      -SI- SUBD(4) -NE- CARD
13      -ALORS- SF(4) := GOV,
14      SF(3) := JUXT,
15      RS(3) := QUAL,
16      VAR(1) := VAR(4)
17      -SINON- SF(3) := GOV,
18      SF(4) := JUXT,
19      RS(4) := QUAL,
20      VAR(1) := VAR(3)
21      -FSI-
22      1:1, K:=NP, UL:='*NP', VLI:=N
```

Figure 13.2 Example ROBRA rule

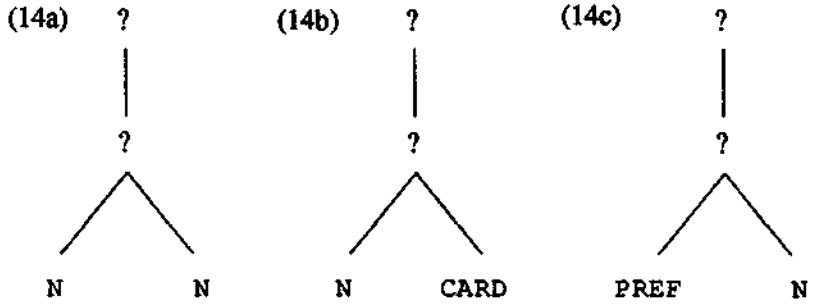
The rule has a name, COMPN 'compound noun' (line 1), which is used mainly as a useful mnemonic for linguists when they are trying to debug the system: a trace of the rules which have been applied will refer to this rule by this name, though otherwise it has no significance. The body of the rule is a left-hand side (lines 1–6) and a right-hand side (lines 10–22). The left-hand side gives the pattern match to be sought in the data structure, while the right-hand side details the new data to be created. In this sense, the rule is like a production rule, or pattern–action pairing.

The pattern-match part of the rule consists of a tree structure, or 'schema' in GETA terminology, (line 1) and a set of 'conditions'. The bracket notation is a fairly standard way of representing the tree structure in (13):

(13)



The conditions (lines 3–6) stipulate that either (i) the CAT[egory] at node 3 should be N[oun] and the category at node 4 should be N or the value of SUBD should be CARD[inal], or (ii) that the category at node 3 should be PREF[ix] and the category at node 4 should be N. The operator –E– means 'equal', –ET– means 'and', and –OU– means 'or', with bracketing giving the operator precedence. In other words, this rule will apply to any of the three trees shown in (14).



As it stands, the rule would match trees where node 2 has further unspecified daughters (or indeed sisters). Sometimes it is desirable to stipulate in the tree schema that the nodes identified should be contiguous, or that there should be no other sister nodes. This can be done using the 'anti-node' formalism, so that the tree with nodes 3 and 4 as contiguous sisters would be expressed as `1(2(3,*,4))`; the asterisk indicates that there must not be other nodes in this position. In the general case, however, the tree formalism (line 1) will in fact match a potentially infinite range of structures with arbitrary numbers of sister nodes at each level.

The condition part of the pattern-match can be quite complex. Often the same or similar conditions are repeated in many different rules. For this reason the ROBRA formalism allows the definition of macros or 'formats', defined separately

which stipulate that two nodes must agree in number and gender, where the statement of 'agreement' may actually be quite complex. For example, a format for agreement might be defined as in (15), which says that CONCORD(A,B) is true if A and B have the same value for NBR and either they agree in GNR or the value for NBR is PL[ural]:

```
(15) CONCORD(A,B) ==
        NBR(A) -E- NBR(B) -ET-
        (GNR(A) -E- GNR(B) -OU- NBR(A) -E- PL)
```

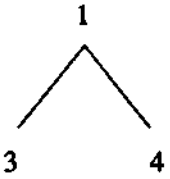
This format would then be called in appropriate places, with actual node numbers in place of A and B, for example CONCORD(3,4). Such a formalism has the advantage of enabling the linguist to state declaratively that there is a relationship CONCORD which may exist between two nodes, and to define this independently of any rules which use it. A second advantage is that, if the name chosen for the format is relatively transparent, complex conditions can be expressed in more meaningful ways and individual rules are easier to understand.

Returning to our example rule, we note that it covers three cases of noun compounds, namely noun+noun (*ville dortoir* 'dormitory town', *homme grenouille* 'frogman'), noun+cardinal number (e.g. *salle 15* 'room 15', *figure 6*), and prefix+noun (e.g. *contre-pas* 'half pace').

After the == sign (line 8) comes the right-hand side of the rule, which details the target tree structure to be built. Like the left-hand side, this has two parts, a structural part (line 10) and a set of assignments ('affectations' in GETA terminology) to the new structure, corresponding to the conditions on the left-hand side.

The 'target schema' details not only the new tree structure in (16), but also, by way of confirmation, an indication that the old node 2 has disappeared: /*←2/ (the asterisk is again an anti-node or non-existent node). If a rule is to create a new node the confirmation would be, e.g. /6←*/, i.e. with the new node and anti-node in the reverse positions.

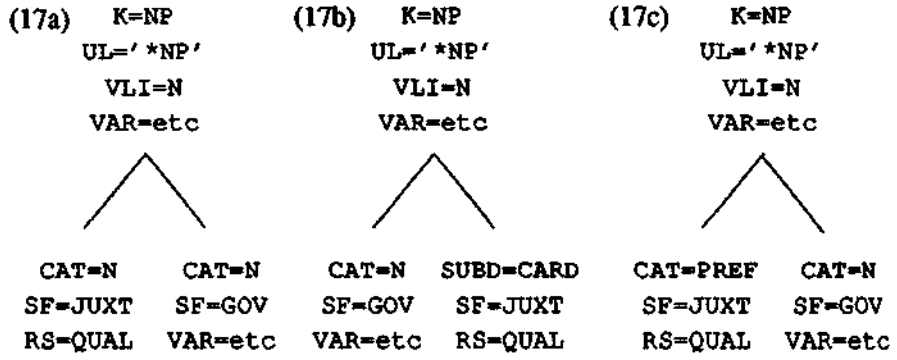
(16)



The affectation part of the right-hand side is a mixture of copying of information, adding new information, and performing complex operations. In line 11 we have a case of simple copying: the attribute–value assignments for the new node 4 are copied from the old node 4. In line 22, the old node 1 is copied onto a new node 1 and three new attributes are added: K, UL and VLI with values NP, *NP and N respectively. Lines 12–21 illustrate a series of complex assignments. Here we see affectations dependent on values of other attributes, expressed by logical operators: –SI– means 'if', –NE– 'not equal', –ALORS–

expressed by logical operators: –SI– means 'if', –NE– 'not equal', –ALORS– 'then', –SINON– 'else' and –FSI– 'end if'. The value assignments are to be as specified in lines 13–16 if SUBD(4) is not CARD, otherwise they are to be as specified in lines 17–20. The former would be the case for (14a) and (14c) above, and the latter would apply to (14b). It should be noticed that assignments can either be given in terms of literal values, as in SF(4):=GOV (i.e. the syntactic function (SF) of node 4 must be GOV); or they can be given in terms of other variables, as in VAR(1):=VAR(4). Furthermore, just as we saw in the case of the conditions, affectations can also be expressed in formats, whenever sets of affectations are used repeatedly in different rules. The mechanism and motivations for its use are the same as for condition formats.

The three tree structures in (17) show the end result of applying this rule.



The formalism can be criticised for being less declarative than intended; note the presence of the procedural 'if–then' in the affectation part. The possibility of including such procedures allows rules to be combined just because they have similar conditions and affectations, even though they may refer to essentially different constructions. In this example, the linguist has been encouraged to deal with three different types of compound noun on formal rather than linguistic grounds. The noun+number type (14b) differs clearly from the other two; it is the only one where the left daughter becomes GOV (17b) and it has one particular condition, tested in line 4 and repeated in the affectation at line 12.

13.5.3 TRANSF and SYGMOR

As we have already remarked, the TRANSF formalism used for lexical transfer is a slightly less powerful tree transducer. TRANSF takes trees as input but unlike ROBRA does not have the power to rearrange them; it is able only to alter the labelling on branches. Thus it has the pattern-matching capability of ROBRA, i.e. locating arbitrarily complex tree structures in the data structure, but it cannot create new tree structures. It is entirely appropriate for lexical transfer. Any structural changes required by the choice of target LUs are handled by the subsequent application of ROBRA (as described above.)

The rule-writing formalism SYGMOR was designed for morphological generation. Its function is to convert labelled trees into strings of characters (including punctuation marks.) There are two transducers; the first converts trees to strings of items, each of which is a bundle of attribute–value pairs; the second transducer maps these bundles onto actual strings. The whole process, though relatively simple, can be quite flexible and includes, for example, string transformation rules which allow regular morphological rules to be captured (e.g. the doubling of certain consonants in English *-ing* forms: *put* → *putting*).

13.6 Concluding remarks

The GETA system is rightly regarded as the archetypal 'second generation' MT system, i.e adopting a linguistics-oriented highly modular indirect translation approach, clearly separating algorithmic and linguistic processes, a stratified approach to analysis and generation, with little semantic processing and no AI-type 'understanding' and no interactive facilities. It represented a considerable advance on earlier direct 'first generation' systems of the pre-ALPAC era but it also exhibits some characteristic shortcomings of this design type.

The main problem lies in the use of the high-level rule-writing formalisms. In order to provide as general a set of formalisms as possible, the computer scientists at GETA had to design particularly powerful formalisms and software. In effect, they are like programming languages, albeit attuned to a special purpose. But this means that linguists have very few constraints, other than some undefinable notion of good programming practice, and the ideals of declarative programming can quickly become submerged. A good example is the use of 'flags' in rules: as a set of rules becomes more and more complex, it becomes increasingly difficult for the linguist to predict exactly what the effect will be of adding or changing a rule here or there. Since the underlying architecture is that of a production system, the linguist is not supposed to incorporate control information in rules, i.e. information which tells the interpreter which rule to apply next. However, for the sake of efficiency and avoidance of spurious computations, it is tempting to specify affectations in one rule which are to be tested in the condition part of another rule, i.e. 'flagging' that the first rule has applied so that the second rule can 'know' this. There is no problem if the 'flags' have an intrinsic linguistic validity, but if they do not, the declarative nature of the system is undermined. A criticism of the GETA approach is that this type of 'cheating' is in fact inevitable and necessary, and that when a system gets to a substantial size the idealised declarative model may be simply inappropriate. Whether later systems based on Prolog and other declarative languages are open to the same criticism remains to be seen.

A second, more local, problem concerns the linguistic representation used in GETA. The multi-level tree structure aims to capture simultaneously surface and deep structure, but in some cases there is a conflict between these two. For example, discontinous elements (e.g. in English phrasal verbs) must be represented as discontinuities at the surface level, but at the deeper level ought really to be merged as single nodes. This is done to a certain extent using the value of the LU

(lexical unit) attribute, but the tree structures still include redundant or partially empty branches which represent surface elements with no deeper roles.

The strengths of the GETA system are its linguistic and computational techniques, particularly in the areas of morphological and syntactic analysis and transfer. Its weaknesses are the inadequate treatment of semantics and the poverty of its lexicographic databases. The only relatively large-scale test was that of the Russian–French version on abstracts in space science and metallurgy with a dictionary of some 7,000 Russian lexical units.

These deficiencies, however, are far outweighed by the substantial pioneering achievements of the GETA research group, which influenced profoundly the direction of MT research from the mid 1960s. The innovative theoretical work on Ariane reached a peak in the early 1980s. Since then the focus of attention in the GETA group has changed slightly. At the time of his death, Vauquois was working on a new, more declarative rule-writing formalism, called 'static grammars'. During the 1980s there has been much work at GETA on improving the environment for linguists working on MT, with better debugging tools and special rule-writing packages which allow linguists to write rules in a more familiar way and which automatically compile the actual formalism, and so on. The most recent new area of research at GETA, now under the direction of Christian Boitet, is the development of an interactive 'dialogue-based' MT system (cf. section 18.5). It is a sign that this long-standing research group remains at the forefront of MT research.

13.7 Sources and further reading

The GETA system is exceptionally well documented both in the public domain of collections on MT and Computational Linguistics, and in conference proceedings, as well as in the semi-private domain of GETA's own technical reports. The information in this chapter has been derived from all of these sources, too numerous to list.

The Grenoble group's early attempts at MT are documented in Vauquois' (1975) book. A representative collection of articles by Vauquois throughout his life has been published under the editorship of Christian Boitet (Vauquois, 1988). Other major general articles about the GETA system are Boitet and Nédobejkine (1981), Vauquois and Boitet (1985), Boitet (1987,1989a), Guilbaud (1987). Whitelock and Kilby (1983) and Hutchins (1986, 1988) give outsiders' views of the system and further references.

The project to commercialize the GETA system, called Calliope, is described in Brunner (1985); Joscelyne (1987) is a journalistic appraisal of its failure. Work on environments for rule writers is described in Boitet and Gerber (1986). Static grammars are described by Vauquois and Chappuy (1985).