WPFG

working papers in functional grammar

WPFG no. 19
March 1987

Twelve sentences: a translation procedure in terms of FG

Bieke van der Korst
University of Amsterdam

TWELVE SENTENCES

A translation procedure in terms of Functional Grammar

Bieke van der Korst

Abstract

In this article I will present the outlines of a translation procedure in terms of Functional Grammar. The article is to be seen as an elaboration of the ideas stated in Dik(1979), a memorandum on this subject. On the basis of these ideas a model is formulated describing the procedure for arriving at the (mechanical) translation of underlying predications of one language into underlying predications of another language. The model is subsequently applied to an English sample text, which is the translation of a French fragment of text, in order to see how and to what extent translation along these lines can reproduce the original text. The model is then adjusted in order to cover a number of overlooked aspects. In the last section of the article a Prolog-program is described and presented which is designed to translate a subset of the English predications into the corresponding French predications according to the model.

1. Introduction

Dik(1979) states that predications have several properties which make them the best representations of sentences for (mechanical) translation aims. Predications are the underlying representations of sentences. Sentences are formed by the application of expression rules to predications. Expression rules determine the form and order of the constituents on the basis of the information contained in the predication. Expression rules do not supply the sentences with information not already present in the underlying predication; they only formalize this information. Predications contain all the information necessary for constructing the semantic interpretation of sentences.

The form of predications is argued to be largely invariant across languages. Hence, the underlying predications of two semantically equivalent sentences in two different languages are much more alike then the sentences themselves.This can be illustrated by an English-Japanese pair of sentences:

(1a) I saw that Hanako was swimming
(1b) Watakusi wa Hanako ga oyoide iru no o mita
I Topic H. Subject swimming to be thing Object saw

While the form of these sentences is very different, their meanings are comparable. The underlying predications of (1a&b) are identical except for the lexical items:

(2) DECL {Pret see_V ($d1x_i$: I_{Pro} $(x_i)0$)ExpSubjTop
$miru_V$ $watakusi_{Pro}$
($d1x_j$:[PretProgr $swim_V$
$oyogu_V$
($d1x_k$:Hanako(x_k))$_{AgSubj}$](x_j))$_{GoObj}$}

Translation on the level of predications would appear to be straightforward once the lexical equivalences are established.

In FG, lexical items are seen as predicates and the lexicon consists of predicate frames. Therefore, the equivalences should relate the predicate frames of one language to the predicate frames of another language. The English predicates of (1a) all have a Japanese equivalent:

(3a) see_V $(x_1)_{Exp}$ $(x_2)_{Go}$ $=_{eq}$ $miru_V$ $(x_1)_{Exp}$ $(x_2)_{Go}$
(3b) I_{Pro} $(x_1)0$ $=_{eq}$ $watakusi_{Pro}$ $(x_1)0$
(3c) $swim_V$ $(x_1)_{Ag}$ $=_{eq}$ $oyogu_V$ $(x_1)_{Ag}$

In the simplest case there is a one-to-one relationship between the predicates of the source language and those of the target language. Dik(1979) states that, if it is assumed that there are only simple one-to-one correspondences as in the English-Japanese example, the translation procedure amounts to the following three steps:

I. reconstruction of the underlying predication of the sentence;
II. substitution of the predicates, according to the equivalences;
III. application of the expression rules of the target language grammar.

This procedure is sketched in Figure 1.

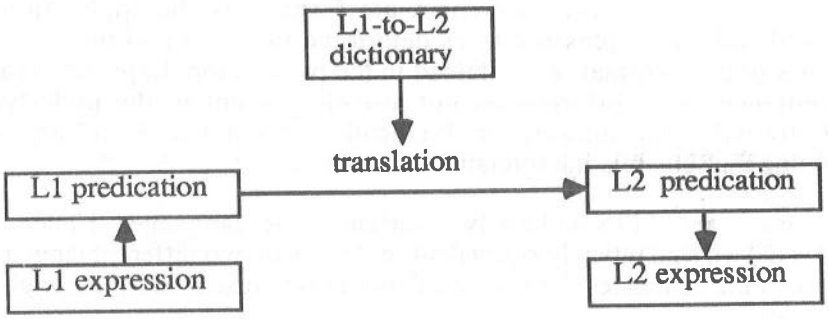

Figure 1. Translation via the predication level

The rest of the memorandum deals with the first step, the (automic) reconstruction of underlying predications, formulating guide lines for a parser strategy.

This article will focus on the second phase of the translation procedure, the actual transfer of predications. In the simplest case, translation of an English predication into a corresponding French predication reduces to a word by word translation, substituting the equivalent French predicates for the English ones. This would be the case if every English predicate could be related to a French predicate with the same meaning and an identical predicate frame. However, it is a fact that not all English words (or predicates) have a French equivalent with exactly the same meaning; there will always be nuances that differ between the two languages. The specific meaning of to hike for example, combining those of to walk and to travel, cannot be expressed in French. In French going by foot is always the same thing: marcher; travelling is another: voyager. Conversely French has two words for you: tu and vous the use of which depends on social factors. In some cases these differences can be traced in selection restrictions on arguments. For example, to bring corresponds with amener or apporter, depending on the nature of the object of the action (whether it walks or not). This could be expressed by the following rules:

(4a) $\text{bring}_V(x_1)_{Ag}(x_2)_{Go} =_{eq} \text{amener}_V(x_1)_{Ag}(x_2{:}\langle\text{animate}\rangle(x_2))_{Go}$
(4b) $\text{bring}_V(x_1)_{Ag}(x_2)_{Go} =_{eq} \text{apporter}_V(x_1)_{Ag}(x_2{:}\langle\text{inanim}\rangle(x_2))_{Go}$

These rules should be read as follows: to bring is equivalent to amener when the second argument is an animate entity, to bring is equivalent to apporter when the second argument refers to an inanimate entity.

But even if we abstract from these differences in meaning and assume that every English predicate can be related to its (closest) corresponding French equivalent, there are still some problems which need to be considered. If the predicate frames of two equivalent predicates are not compatible, exchange of these predicates cannot be performed without the required adjustments. Corresponding predicates can differ in their quantitave and/or qualitative valency:

I. differences in quantitative valency:
- the corresponding predicate can have more arguments than the original, for example:
  to kick somebody (Agent & Goal argument) corresponds with
  donner des coups de pied a quelqu'un (Agent, Goal & Recipient argument);
- the corresponding predicate can have fewer arguments than the original, for example:
  to put somebody to bed (Agent, Goal & Direction argument) corresponds with
  coucher quelqu'un (Agent & Goal argument).

In the first case the substitution will require extension of the number of arguments, in the second case the number of arguments will have to be reduced.

II. differences in qualitative valency:
- the corresponding predicate(s) can impose other selection restrictions on the arguments than the original. Examples are to bring, which corresponds with either apporter or amener (example (4) above) and the French porter, which corresponds with either to wear (with clothes) or to carry (other objects).
- the corresponding predicate can assign other semantic functions to its arguments than the original, for example: to listen (to) with a Direction argument corresponds with écouter with a Goal argument, and to live (in) with a Locative argument corresponds, in some cases, with habiter with a Goal argument.

The first case can be left out of consideration, because this difference does not have formal consequences for the structure of the predication; in the second case semantic functions will have to be changed in order to match with the argument structure of the French predicate.

Two predicates are not fully equivalent (exchangeable) when they differ in nuance or when they differ formally, in the number and semantic functions of the arguments they combine with. I propose to treat such predicates as "idiom", in a broad sense of the word, illustrated by the following scheme:

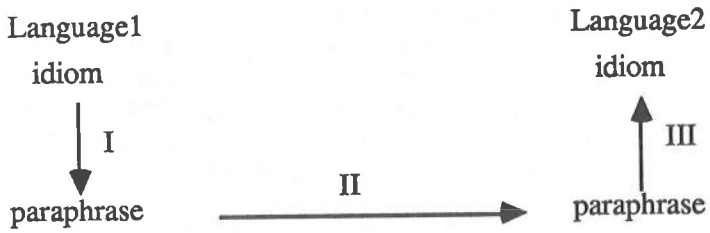

Figure 2. Translation

Figure 2 can be explained as follows. Whenever an element of language1 cannot be translated directly into an element of language2, this element should be treated as idiom, in the sense of a language specific expression. What does one do when one has to translate words that do not figure in the dictionary? The usual procedure will be to try to find other entries, paraphrases which have a translation. In order to arrive at an optimal translation, the next step should consist of trying to find the preferred expression in the target language. This is achieved by assuming to be dealing with a paraphrase, which can be transformed into "idiom". This last step may be called "deparaphrasing". See 2.1. for further explanation.

2. The model

In this section a model is formulated on the basis of the ideas stated in the introduction. This model consists of guiding rules for translation on the level of predications. The validity of these principles is tested by application to a sample text. After a reformulation by extension, the model will prove to be capable of achieving a high degree of precision.

2.1 Formulation

In general, one can say about a theory or a model that it has to satisfy a number of demands. These demands vary across models and depend upon the properties and aims of the model in question. In formulating this model, I had the following demands in mind:

1. The model should be **simple**: it should not handle more than what it is designed for, in this case the translation of predications of one language into predications of another language. The problems concerning the reconstruction of the underlying predications on the basis of sentences and the construction of sentences on the basis of predications are left out of consideration.
2. The model should be **operational:** it has to be suitable for translation into a computer program.
3. The model should be **language-pair independent**, corresponding with the typological adequacy claimed by FG.
4. The model should be **complete**: for every allowed input the process described in the model should provide an output.
5. Finally, the model should be as **correct** as possible. For each source language predication it has to deliver the target language predication which approaches the original one as closely as possible.

This model is based upon the premise that predications can be translated by replacing the relevant predicates by their equivalents in the other language. Corresponding predicates of two languages are related by equivalences, which have the form of rules. At this point I would like to introduce a refinement. This refinement consists of allowing only "pure" equivalences in these rules. Two predicates are purely equivalent when their meanings correspond and their predicate frames are compatible. Equivalences are stated as follows:

$$\text{predicate-L1 } (x_1)Sf1..(x_n)Sfn =_{eq} \text{predicate-L2}[(x_1)Sf1..(x_n)Sfn]$$

where the part between square brackets could be considered as redundant. In a number of cases selection restrictions will be included upon an argument. This was illustrated in (4a&b) above.

When a predicate is not incorporated in the equivalences, there are two possibilities:
- it is a basic predicate, part of the lexicon;
- it is a derived predicate, formed on the basis of another predicate by means of a predicate formation rule (PFR).

In both cases the predicate has to be "paraphrased". Basic predicates are part of the lexicon, which associates them with a meaning definition. Paraphrasing a basic predicate consists of replacing it by its meaning definition. Derived predicates are formed by means of rules. These rules are said to reflect a synchronically productive process. Predicate formation rules are defined by an input predicate type, an output predicate type and a meaning specification, which expresses the change of meaning with respect to the input predicate. Paraphrasing a derived predicate consists of replacing it by this meaning specification as holding for the predicate in question. Note that these principles should work recursively

and cooperatively: it may be the case that the paraphrases are not incorporated in the equivalences either, and then the two possibilities arise again.

The treatment of derived predicates can be optimized in some cases, in particular when the two languages share a predicate formation rule. If both languages have a rule which applies in the same way with the same effect the derived predicates of this class will not have to be included in the equivalences. The rule can be applied in reverse in the source language to obtain the primitive (input) predicate; this predicate is marked for application of the rule in question and translated; after translation the marker triggers application of the corresponding rule in the target language. The reflexive use of verbs, for example, is handled by means of a predicate formation rule in both French and English. The English rule takes away the second argument of a transitive predicate, the French rule does the same thing and supplies the verbal predicate with the reflexive element "se". They both cause a reflexive interpretation, as is illustrated in (5):

(5a) John washes
(5b) Jean se lave

Translation of the one-place verbal predicate wash will be achieved as follows:
- the reflexive argument reduction rule is applied in reverse
- the resulting two-place basic predicate wash marked R is translated
- the two-place French laver serves as input for the French reflexive rule
- output of the rule is one-place se laver

In this way the equivalences will not have to contain the one-place reflexive verbal predicates, if they are derived from transitive verbs (two-place predicates).

The idea of restricting the rules to pure equivalence is based on a choice. One could allow equivalences relating predicates with different valency, possibly even in the form of structure changing operations as illustrated in (6b):

(6a) $\text{listen}_V\ (x_1)_{Ag}\ (x_2)_{Dir} =_{eq} \text{écouter}_V\ (x_1)_{Ag}\ (x_2)_{Go}$
(6b) $\text{listen}_V\ (x_1)_{Ag}\ (x_2)_{Dir} =_{eq} \text{écouter}_V, \text{Dir} \rightarrow \text{Go}$

At first sight the difference only concerns the place where the problem of structural changes is handled. With the approach presented above the problem is taken away from the equivalences, which are left simple and uniform. The actual translation procedure is reduced to lexical substitution. Structural changes are performed before translation. The advantage of this approach lies in the fact that all the necessary information is already present in the fund of the two languages. The fund contains all the lexical information of a language, consisting of the lexicon with basic predicates (and terms) and of predicate (and term) formation rules. It does not seem useful to duplicate this information in the equivalences. All the equivalences have to do is relate the equivalent predicates. The aim is to maximize the lexical substitution.

This approach relies upon the idea that paraphrasing will always lead to a translatable predicate or construction of predicates in the end. The termination of this process is guaranteed by providing equivalences for at least all the lexical primitives, the predicates that are not further defined in the lexicon. In some cases the forced paraphrasing will lead to circumlocution. This will have to be straightened by the third component, called "deparaphrasing" provisionally. The need for such a component does not arise from this problem, as has been motivated in the introduction; this approach only takes the full benefit of it.

This third phase in the translation procedure can be sketched as follows. The output of the transfer component is not considered as the definitive translation, but as a possible paraphrase of the final translation. The principles stated above describe the procedure for arriving at translatable predicates from "idiomatic" predicates. These principles should be applied in the opposite direction in order to arrive at the language preferred predicates from translatable predicates. This reverse application will result in matching the predication or parts of it with the meaning definitions in the lexicon and the meaning specifications of the predicate formation rules of the target language. As an example consider sentence (7a) with the underlying predication (7b):

(7a) John puts the children to bed.
(7b) DECL {Pres put_V ($d1x_1$:$John_{Np}$(x_1)0)$_{Ag}$ (dmx_2:$child_N$(x_2)0)$_{Go}$
($i1x_3$:bed_N(x_3)0)$_{Dir}$}

All the predicates contained in this predication have a pure French equivalent; substution of these predicates results in (7c):

(7c) DECL {Pres $mettre_V$ ($d1x_1$:$Jean_{Np}$(x_1)0)$_{Ag}$
(dmx_2:$enfant_N$(x_2)0)$_{Go}$ ($i1x_3$:lit_N(x_3)0)$_{Dir}$}

The desired idiomatic French expression is coucher, which will be arrived at after inspection of the French lexicon, where this "idiomatic" expression has a place: coucher will be associated with a meaning definition which matches the predicate construction in (7c); the final translation will be (7e):

(7d) $coucher_V$ (x_1)$_{Ag}$ (x_2)$_{Go}$ =df $mettre_V$ (x_1)$_{Ag}$ (x_2)$_{Go}$
($i1x_3$:lit_N(x_3)0)$_{Dir}$
(7e) DECL {Pres $coucher_V$ ($d1x_1$:$Jean_{Np}$(x_1)0)$_{Ag}$
(dmx_2:$enfant_N$(x_2)0)$_{Go}$}

The described model can be illustrated by means of figure 3, which comprises figures 1 and 2 presented in the introduction:

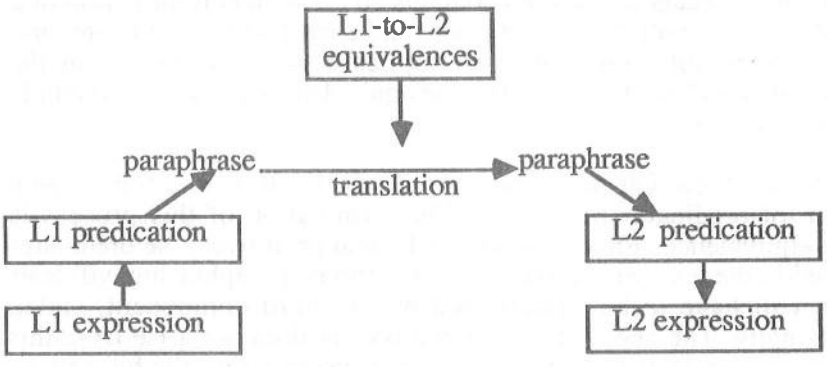

Figure 3. Translation via the predication level

2.2 Test

The model described in the preceding section will now be put to the test of translating the underlying predications of an English translation of a French text into the corresponding French predications. The text given below will have to be mapped onto underlying predications; since there is not yet a parser available for this task, this is done by hand. The translation of predications is achieved by means of lexical substitution of the predicates according to the formulated principles. The lexical equivalences and exact procedures needed for the translation of the English predications will be stated. The result of this translation procedure will be evaluated.

Text

The sample text on which the model will be applied is given below, chapter XII from the English translation of "Le petit prince" (Saint-Exupery(19??)).

> The next planet was inhabited by a tippler. This was a very short visit, but it plunged the little prince into deep dejection.
> "What are you doing there?" he said to the tippler, whom he found settled down in silence before a collection of empty bottles and also a collection of full bottles.
> "I am drinking," replied the tippler, with a lugubrious air.
> "Why are you drinking?" demanded the little prince.
> "So that I may forget," replied the tippler.
> "Forget what?" inquired the little prince, who already was sorry for him.
> "Forget that I am ashamed," the tippler confessed, hanging his head.
> "Ashamed of what?" insisted the little prince, who wanted to help him.
> "Ashamed of drinking!" The tippler brought his speech to an end, and shut himself up in an impregnable silence.
> And the little prince went away, puzzled.
> "The grown-ups are certainly very, very odd," he said to himself, as he continued on his journey.

Note that the translation task is restricted since only this text is considered. On the other hand, the task is complicated since this is a running, existing text rather than carefully selected sample sentences.

Underlying predications

As is to be expected, the mapping of sentences onto underlying predications is not always straightforward. The theory of Functional Grammar does not, yet, provide solutions in all cases. In reconstructing the predications underlying the sentences of the text, decisions have been made at several points which are not fully determined by the theory as it has been developed so far. With each predication the decisions made in constructing it are indicated and motivated. The predications are complete except for pragmatic functions, which are left out of consideration for practical reasons.

(8a) The next planet was inhabited by a tippler.
(8b) DECL {Pret inhabit$_V$ (i1x$_1$:tippler$_N$(x$_1$)0)Po
(d1nextx$_2$:planet$_N$(x$_2$)0)GoSubj}

-The second argument of inhabit has goal-function: it does not take a preposition and can be passivized.
-Next is taken as an ordinal term-operator, not as a restrictor. The distinction between term-operators and restrictors is not always very clear, especially not in singular terms. Restrictors can be seen as qualifying each member of the set of referents individually, while term-operators determine this set as a whole. Since in a phrase as the next three books,next orders the whole set of books rather than each book individually, next qualifies for term operator status.

(9a) This was a very short visit, but it plunged the little prince into deep dejection.
(9b) DECL {Pret {(i1x3:visit$_N$(x3)0:very short$_A$(x3)0)0} ([+prox]1x4)0Subj}
but
DECL {Pret plunge$_V$ (Ax4)FoSubj
(d1x5:prince$_N$(x5)0:little$_A$(x5)0)GoObj
(i-x6:dejection$_N$(x6)0:deep$_A$(x6)0)Dir}

-A very short visit is a (fully specified) term functioning as predicate. The representation is taken from Dik(1980, pp. 90-111), where "John is in the garden" is assigned the following underying predication:
DECL {Pres {(d1x$_i$:garden$_N$(x$_i$)0)Loc} (d1x$_j$:John$_{Np}$(x$_j$)0)Po}
- Very short is a derived predicate. Derived predicates are formed according to some synchronically productive process, as opposed to basic predicates, which are given in the lexicon. The "intensifying" rule could be stated as follows:

input:	predicate$_A$
output:	very predicate$_A$
meaning:	"predicate$_A$ in a high degree"

-Demonstratives are handled by means of term operators, see Dik(1978, pp. 60-63). In this way several subsystems of operators are defined, of which demonstratives are characterized by the distinguishing feature [+/- prox]. Demonstrative pronouns are represented as above; the following examples illustrate the representation of demonstrative adjectives.
([+prox]1x$_i$:boy$_N$(x$_i$))..... this boy
([+prox]mx$_i$:boy$_N$(x$_i$))..... these boys

([-prox]1x_i:boy$_N$(x_i)) .....that boy

([-prox]mx_i:boy$_N$(x_i)).....those boys

-Personal pronouns are handled by means of term operators too. They form another subsystem, where H ("hearer") is expressed by the second person "you", and S ("speaker") by a first person pronoun. In fact this subsystem is characterized by the two distinguishing features [+/- speaker,+/- hearer], and H and S are short for [-s,+h], [+s,-h] respectively. Personal pronouns of the third person will be represented by the anaphoric term operator A, unless there is no referent in the text. In this case one should use [-s,-h], to represent deictic "he". Personal pronouns are not considered to be basic, "ready-made" terms, because at the moment of term-insertion the case or form of the pronoun in the sentence is not yet determined (I vs. me, they vs. them, etc.). In this respect a more abstract representation by means of features has advantages.

- Dejection is an uncountable noun, so the numeral operator is left unspecified in the case of singular. This approach seems more appropriate than specifying a singular term operator which is not expressed with this kind of noun.

(10a) "What are you doing there?" he said to the tippler, whom he found settled down before a collection of empty bottles and also a collection of full bottles.

(10b) DECL {Pret say$_V$ (Ax_5)$_{AgSubj}$
(x7:[INT {PresProgr do$_V$ (Hx_1)$_{AgSubj}$
(Qx_8)$_{GoObj}$
([-prox]x_9)$_{Loc}$}](x_7))$_{GoObj}$
(d1x_1:tippler$_N$(x_1)$_0$:
Pret find$_V$ (Ax_5)$_{ProcSubj}$
(Rx_1)$_{GoObj}$
(x_{10}:[PartPerf settle down$_V$ (x_5)$_{Ag}$
(x_{11}:silence$_N$(x_{11})0)Circ
((i1x_{12}:collection$_N$(x_{12})0
(imx_{13}:bottle$_N$(x_{13})$_0$:
empty$_A$(x_{13})0)Part)
and also (i1x_{14}:collection$_N$(x_{14})0
(imx_{15}:bottle$_N$(x_{15})$_0$:
full$_A$(x_{15})0)Part))$_{Loc\text{-}fr}$(x_{10}))Circ)$_{Rec}$}

-There is a set of predicates, such as say, which can take a predication as argument. Embedded predications take the following form: (x_i:[predicate (x_1)..(x_n)](x_i)). The predication specifies an entity x_i, which can be questioned and referred to by means of an anaphor. Most embedded predications are expressed by means of a that-clause; some can be expressed in direct discourse, between quotation marks. These two different surface forms must have distinctive underlying representations. One way to handle the problem would be to introduce a term operator "quote", next to or instead of the earlier used term operator "sub" for that-clauses. This has the air of an ad hoc solution. The quote-arguments in this text are represented as fully specified predications, that is predications with a predication operator and subject and object assignment; that-clauses are represented as "subpredications". The argument behind this approach lies in the thought that every main clause can be considered a quote. In this way the distinction between direct discourse and subordinate (that-)clauses relies upon the presence or absence of the predication-operator:

DECL {Pres say$_V$ (d1x$_1$:Peter$_{Np}$(x$_1$)0)$_{AgSubj}$
(x$_2$:[PresProgr come$_V$ (d1x$_3$:John$_{Np}$(x$_3$)0)$_{Ag}$](x$_2$))$_{GoObj}$}

for:
Peter says that John is coming.

DECL {Pres say$_V$ (d1x$_1$:Peter$_{Np}$(x$_1$)0)$_{AgSubj}$
(x$_2$:[DECL {Presprogr come$_V$ (d1x$_3$:John$_{Np}$(x$_3$)0)$_{Ag}$}](x$_2$))$_{GoObj}$}

for:
Peter says: "John is coming".

N.B. the distinction between that- and if-clauses has been left out of consideration, as there are no instances of if in this text. The solution stated above serves the purposes of this restricted task, but may prove to be insufficient when other constructions are involved. This is one of the problems that require further investigation.
-Questioned terms and relatives are handled along the lines stated above for demonstratives, by means of the term operators Q and R.
-All predicative phrases have been treated as satellites of circumstance, as

DECL {Pret find$_V$ (Ax$_i$)$_{ProcSubj}$ (Ax$_j$)$_{GoObj}$ (x$_k$:[dead$_A$ (x$_j$)0](x$_k$))$_{Circ}$}

seems to be the correct representation of: He found him dead. Literally: "x_i found x_j in the circumstance x_k defined by x_j being dead".
-Collection is analyzed as taking two arguments, the second of which has partitive-function, which is expressed by the preposition of.
-The semantic function 'locative' requires specification of orientation in some cases, in the predication above fr(ont) is added in order to trigger expression by before. The same holds for the time-function, which in some cases will need specification of aspect. This is another topic which needs further investigation.

(11a) "I am drinking," replied the tippler, with a lugubrious air.
(11b) DECL {Pret reply$_V$ (d1x$_1$:tippler$_N$(x$_1$)0)$_{AgSubj}$
(x$_{16}$:[DECL {PresProgr drink$_V$ (Sx$_1$)$_{AgSubj}$}](x$_{16}$))$_{GoObj}$
(i1x$_{17}$:air$_N$(x$_{17}$)0:lugubrious$_A$(x$_{17}$)0)$_{Manner}$}

(12a) "Why are you drinking?" demanded the little prince.
(12b) DECL {Pret demand$_V$ (d1x$_5$:prince$_N$(x$_5$)0:little$_A$(x$_5$)0)$_{AgSubj}$
(x$_{18}$:[INT {PresProgr drink$_V$ (Hx$_1$)$_{AgSubj}$
(Qx$_{19}$)$_{Rsn}$}](x$_{18}$))$_{GoObj}$}

(13a) "So that I may forget," replied the tippler.
(13b) DECL {Pret reply$_V$ (d1x$_1$:tippler$_N$(x$_1$)0)$_{AgSubj}$
(x$_{20}$:[DECL {(x$_{21}$:[Poss forget$_V$ (Sx$_1$)$_{Proc}$
(0)$_{Go}$](x$_{21}$))$_{Purp}$}](x$_{20}$))$_{GoObj}$}

-Poss is used as a predicate operator indicating potential modality and is supposed to be expressed by the modal auxiliary may. The treatment of modality has not yet been sufficiently developed in FG. The predication in which it occurs constitutes a satellite with purpose function, as the sentence it underlies can be considered incomplete ("So that I may forget").

(14a) "Forget what?" inquired the little prince, who already was sorry for him.
(14b) DECL {Pret inquire$_V$ (d1x$_5$:prince$_N$(x$_5$)$_0$:little$_A$(x$_5$)$_0$:Pret sorry$_A$
(Rx$_5$)$_0$Subj
(Ax$_1$)$_{Ben}$
(already)$_{Ti}$)$_{AgSubj}$
(x$_{22}$:[INT {forget$_V$ (x$_1$)$_{Proc}$
(Qx$_{23}$)$_{Go}$}](x$_{22}$))$_{GoObj}$}

- The adverb already has been represented as a satellite with time function. The form of the satellite term is improvised; it is, in such cases as these, hard to imagine adverbials specifying entities as restrictors. FG does not yet give a full account of the representation of satellites. It is conceivable that a full analysis of such forms as already, not yet etc. will require them to be regarded as the expression of a special subsystem of predicate operators.
- The lack of tense in the embedded clause is accounted for by the absence of a predicate operator in the embedded predication.

(15a) "Forget that I am ashamed," the tippler confessed, hanging his head.
(15b) DECL {Pret confessV (d1x$_1$:tippler$_N$(x$_1$)0)$_{AgSubj}$
(x$_{24}$:[DECL {forget$_V$ (x$_1$)$_{Proc}$
(x$_{25}$:[Pres ashamed$_A$
(Sx$_1$)$_0$](x$_{25}$))$_{Go}$}](x$_{24}$))$_{GoObj}$
(x$_{26}$:[PartPres hangV (x1)Ag
(d1x$_{27}$:head$_N$(x$_{27}$)$_0$:{(x$_1$)$_{Poss}$}(x$_{27}$)0)$_{Go}$](x$_{26}$))$_{Circ}$}

(16a) "Ashamed of what?" inquired the little prince, who wanted to help him.
(16b) DECL {Pret insist$_V$ (d1x$_5$:prince$_N$(x$_5$)$_0$:
little$_A$(x$_5$)$_0$:
Pret want$_V$ (Rx$_5$)$_{PoSubj}$
(x$_{28}$:[Inf help$_V$(x$_5$)$_{Ag}$
(Ax$_1$)$_{Go}$](x$_{28}$))$_{GoObj}$)$_{AgSubj}$
(x$_{29}$:[INT {ashamed$_A$ (x$_1$)$_0$Subj
(Qx$_{30}$)$_{So}$}](x$_{29}$))$_{GoObj}$}

(17a) "Ashamed of drinking!" The tippler brought his speech to an end, and shut himself up in an impregnable silence.

(17b)DECL {Pret [bring$_V$ (d1x$_{31}$:speech$_N$(x$_{31}$)$_0$:{(x$_1$)$_{Poss}$}(x$_{31}$)$_0$)$_{Go}$
(i1x$_{32}$:end$_N$(x$_{32}$)$_0$ (x$_{31}$)$_{Compl}$)$_{Dir}$]
(d1x$_1$:tippler$_N$(x$_1$)$_0$)$_{AgSubj}$
(x$_{33}$:[DECL {ashamed$_A$ (x$_1$)$_0$ b(x$_{34}$:[drink$_V$(x$_1$)$_{Ag}$](x$_{34}$))$_{So}$}](x$_{33}$))$_{GoObj}$}
and
DECL {Pret shut up$_V$ (Ax$_1$)$_{AgSubj}$
(Ax$_1$)$_{GoObj}$
(i1x$_{35}$:silence$_N$(x$_{35}$)$_0$:impregnable$_A$(x$_{35}$)$_0$)$_{Loc}$}

- To bring one's speech to an end is here taken to be a complex (derived) predicate with two arguments (Agent & Goal), of which the second must be a fully specified predication, which underlies a quote (cf. note [1]). There are reasons to believe that this assumed predicate formation rule reflects a productive process, given the following examples:

"Goodmorning, gentlemen", he opened the meeting.
"You're so sweet", he teased me.

- End is analyzed as two-place: x_i is end of x_j.
- Drinking is considered to be a nominalisation, triggered by the structure of the underlying predication: no predicate operator, no arguments are expressed.

(18a) And the little prince went away, puzzled.
(18b)and
DECL (Pret go away$_V$ (d1x$_5$:prince$_N$(x$_5$)$_0$:little$_A$(x$_5$)$_0$)$_{AgSubj}$
(x$_{36}$:[puzzled$_A$ (x$_5$)$_0$](x$_{36}$))$_{Circ}$}

- To go away is considered to be a complex predicate, formed by a predicate formation rule which applies to verbs implying a movement, for example: to send (away), to put (away), to run (away), etc.

(19a) The grown-ups are certainly very, very odd," he said to himself, as he continued on his journey.
(19b)DECL {Pret say$_V$ (Ax$_5$)$_{AgSubj}$
(x$_{37}$:[DECL certainly {Pres very very odd$_A$
(dmx$_{38}$:grown-up$_N$(x$_{38}$)$_0$)$_{0Subj}$}](x$_{37}$))$_{GoObj}$
(Ax$_5$)$_{Rec}$
(x$_{39}$:[{Pret continue$_V$ (Ax$_5$)$_{AgSubj}$
(d1x$_{40}$:journey$_N$(x$_{40}$)$_0$:{(x$_5$)$_{Poss}$}(x$_{40}$)$_0$)$_{Loc}$}](x$_{39}$))$_{Circ}$}

- Certainly is supposed to be the expression of a satellite on the level of illocution, which can be considered as an extension of the domain of the predication operator. The precise form and position of this kind of satellite will be left unspecified. As has been mentioned before, there is still a lot of work to be done in the handling of satellites.

Predicate frames & equivalences

In this section the lexical information concerning the underlying predications is supplied, in particular the part of the English fund involved and the lexical equivalences needed. All predicates are contained in predicate frames, which specify the following information about the predicate:

1. its lexical form
2. its syntactic type: N(ominal), V(erbal) or A(djectival)
3. the number of arguments it combines with
4. the selection restrictions imposed on these arguments
5. the semantic functions of the arguments.

Predicate frames may be either basic, contained in the lexicon, or derived, formed by a productive rule. Each predicate frame is associated with a meaning specification which is again captured in a predicate frame. The equivalences needed to translate these predicate frames are indicated and, where necessary, commented on.

Note, again, that the theory of Functional Grammar is not yet fully developed; the English fund which the presented predicate frames are supposed to be part of has not yet been established. There are many points which require discussion and/or further investigation.

inhabit is a basic verbal predicate whose predicate frame is part of the lexicon. It takes two arguments, with the semantic functions of Positioner and Goal respectively. The first one is selectionally restricted to human beings, the second one is reserved for terms indicating some kind of location. The meaning of inhabit is defined as "to live in".

$\text{inhabit}_V\ (x_1\text{:<human>}(x_1))_{Po}\ (x_2\text{:<location>}(x_2))_{Go}$
$=_{df}\ \text{live}_V\ (x_1)_{Po}\ (x_2)_{Loc}$
$=_{eq}\ \text{habiter}_V\ (x_1)_{Po}\ (x_2)_{Go}$

to inhabit is equivalent to habiter, but this is a one-way equivalence: habiter has another, intransitive, use, similar to to live. On the level of predicate frames there is a pure equivalence, since habiter will be contained in two predicate frames, two entries in the lexicon, one with only a Positioner-argument and one with a second, Goal argument.

tippler is a derived predicate, a 'nomen agentis' (N.A.) formed by a predicate formation rule which can be stated als follows:

input:	$\text{predicate}_V\ (x_1)_{Ag}$
output:	$\text{predicate-(e)r}_N\ (x_1\text{:<male>}(x_1))_0$
meaning:	$\{(x_2\text{:person}_N(x_2)\text{:predicate}_V(x_2)_{Ag})\}\ (x_1)_0$

Since there is a similar rule in French which applies in the same way, the rule can be reversed and the primitive predicate is marked N.A. in order to trigger application of the N.A.-rule after translation. The rule has been applied to the basic predicate tipple, with the following predicate frame and meaning definition:

$\text{tipple}_V\ (x_1\text{:<human>}(x_1))_{Ag}$
$=_{df}\ \text{drink}_V\ (x_1)_{Ag}\ (x_2\text{:alcohol}_N(x_2)_0)_{Go}$

Assuming that there is no French equivalent of to tipple (cf. note [2]), translation will be achieved by replacing it by its defining predicate drink,which does have an equivalent:

$drink_V\ (x_1)_{Ag}\ (x_2)_{Go} =_{eq} boire_V\ (x_1)_{Ag}\ (x_2)_{Go}$

So drink marked N.A. is translated into boire, which will serve as input to the French N.A.-rule.

planet is a basic nominal predicate, defined as "a large heavenly body moving around the sun":

$planet_N\ (x_1)_0$
$=_{df} \{(x_2{:}body_N(x_2)_0{:}heavenly_A(x_2)_0{:}large_A(x_2)_0{:}move_V\ (x_2)_{Proc}$
$(d1x_3{:}sun_N(x_3)_0)_{LocCirc})_0\}\ (x_1)_0$
$=_{eq} planète_{Nf}\ (x_1)_0$

Its French equivalent is planete, a feminine noun. In French there is a formal distinction between feminine and masculine nouns, which is taken to be a categorial distinction: Nf is the category of feminine nouns, Nm that of masculine nouns.

visit is seen as a basic nominal predicate. It is difficult to determine whether the nominal predicate visit is derived from the verbal predicate visit or vice versa. Nominal visit is here taken to be basic, for the sake of simplicity and because there are no decisive reasons to assume the opposite.

$visit_N\ (x_1)_0$
$=_{df} \{(x2{:}time_N(x_2)_0{:}spend_V\ (x_3{:}<human>(x_3))_{Po}\ (x_2)_{Go}\ (x_4)_{Loc})_0\}\ (x_1)_0$
$=_{eq} visite_{Nf}(x_1)_0$

short is a basic adjectival predicate which corresponds with court in many uses; this is the case in the actual context, which is reflected in the meaning definition.

$short_A\ (x_1)_0$
$=_{df} last_V\ (x_1{:}<event>(x_1))_0\ (i1x_2{:}time_N(x_2)_0{:}little_A(x_2)_0)$
$=_{eq} court_A\ (x_1)_0$

plunge is a basic verbal predicate which in this use, again reflected in the selection restrictions, corresponds with plonger:

$plunge_V\ (x_1{:}<inanimate>(x_1))_{Fo}\ (x_2{:}<human>(x_2))_{Go}\ (x_3{:}<state>(x_3))_{Dir}$
$=_{df} cause\ (x_1)_{Fo}\ (x_4{:}[\{(x_3)_{Circ}\}(x_2)_{Po}](x_4))_{Go}$
$=_{eq} plonger_V\ (x_1)_{Fo}\ (x_2)_{Go}\ (x_3)_{Dir}$

prince is a basic nominal predicate, which has several meanings, of which the following must be the intended one in the context of the story (compare: the prince of Monaco).

$prince_N\ (x_1{:}<human,male>(x_1))_0$
$=_{df} ruler_N\ (x_1)_0\ (i1x_2{:}state_N(x_2)_0{:}small_A(x_2)_0)_{Compl}$
$=_{eq} prince_{Nm}\ (x_1)_0$

little is a basic adjectival predicate with several meanings and translations of which the following corresponds with the use in the text:

$little_A\ (x_1)_0$
$=_{df}\ young_A\ (x_1{:}\langle human\rangle(x_1))_0$
$=_{eq}\ petit_A\ (x_1)_0$

dejection is a derived predicate, a nominalisation of the verbal predicate deject:

input:	$predicate_V\ (x_1)..(x_n)$
output:	$predicate\text{-}((a)t)ion_N\ (x_1)_0$
meaning:	$\{(x_2{:}predicate_V\ (x_3)..(x_n))\}\ (x_1)_0$

$deject_V\ (x_1)_{Ag}\ (x_2)_{Go}$
$=_{df}\ cause_V\ (x_1)_{Ag}\ (x_3{:}\ [sad_A\ (x_2)_0]\ (x_3)_0)_{Go}$
$=_{eq}\ abattre_V\ (x_1)_{Ag}\ (x_2)_{Go}$

deep is a basic adjectival predicate with different meanings in different contexts, in this particular context grand is the French equivalent.

$deep_A\ (x_1)_0$
$=_{df}\ strong_A\ (x_1{:}\langle feeling\rangle(x_1))_0$
$=_{eq}\ grand_A\ (x_1)_0$

say is a basic verbal predicate taking three arguments of which the second has to refer to or consist of spoken words. One could impose the selection restriction PREDICATION, but then there would have to be another entry for say when it is combined with "a few words", "nothing", etc. Therefore, <text> has been chosen.

$say_V\ (x_1{:}\langle human\rangle(x_1))_{Ag}\ (x_2{:}\langle text\rangle(x_2))_{Go}\ (x_3{:}\langle animate\rangle(x_3))_{Rec}$
$=_{df}\ express_V\ (x_1)_{Ag}\ (x_4{:}\langle state\ of\ affairs\rangle(x_4))_{Go}\ (x_3)_{Rec}\ (x_2)_{Instr}$
$=_{eq}\ dire_V\ (x_1)_{Ag}\ (x_2)_{Go}\ (x_3)_{Rec}$

do is considered to be one of the lexical primitives whose meaning is not further defined. Do is often used as a substitute or default predicate, as is the case in: "What are you doing?".

$do_V\ (x_1\langle animate\rangle(x_1))_{Ag}\ (x_2{:}\langle act\rangle(x_2))_{Go}$
$=_{df}$ -
$=_{eq}\ faire_V\ (x_1)_{Ag}\ (x_2)_{Go}$

find is a basic verbal predicate which corresponds with trouver.

$find_V\ (x_1{:}\langle animate\rangle(x_1))_{Proc}\ (x_2)_{Go}$
$=_{df}\ discover_V\ (x_1)_{Proc}\ (x_2)_{Go}$
$=_{eq}\ trouver_V\ (x_1)_{Proc}\ (x_2)_{Go}$

settle down is a derived verbal predicate when it takes one argument; the basic predicate settle-down is a two-place predicate meaning: to cause to sit comfortably. The argument reduction rule stated below causes a reflexive interpretation: to sit comfortably. The same rule applies in French, but in French the reflective interpretation is expressed by "se". Dik

(1985) proposes that the reduction be coded by a marker R on the predicate ; then in English R=0, in French R="se". The output of the reversed reduction rule is two-place settle down. The basic predicate settle-down is supplied with a marker for reduction, translated into installer which wil end up as s'installer.

settle-down$_V$-R (x_1:<anim>x_1)$_{Ag}$
argument reduction (refl.):
input: predicate$_V$ (x_1) (x_2)$_{Go}$
output: predicate$_V$-R (x_1)
meaning: predicate$_V$ (x_1) (x_1)

settle-down$_V$ (x_1:<anim>(x_1))$_{Ag}$ (x_2:<anim>(x_2))$_{Go}$
=$_{df}$ cause$_V$ (x_1)$_{Ag}$ (x_3:[comfortable$_A$ (x_2)$_{Po}$](x_3))$_{Go}$
=$_{eq}$ installer$_V$ (x_1)$_{Ag}$ (x_2)$_{Go}$

silence is a basic predicate with a pure equivalent:

silence$_N$ (x_1)$_0$
=$_{df}$ absence$_N$ (x_1)$_0$ (x_2:sound$_N$(x_2)$_0$)$_{Compl}$
=$_{eq}$ silence$_{Nm}$ (x_1)$_0$

collection could be a derived predicate, but according to its intended idiomatic meaning ("pile"), it is considered a lexicalized nominalization.

collection$_N$ (x_1)$_0$ (x_2)$_{Part}$
=$_{df}$ group$_N$ (x_1)$_0$ (x_2)$_{Part}$
=$_{eq}$ collection$_{Nf}$ (x_1)$_0$ (x_2)$_{Part}$

bottle is a basic nominal predicate with a French equivalent:

bottle$_N$ (x_1)$_0$
=$_{df}$ container$_N$(x_1)$_0$ (x_2:<liquid>(x_2))$_{Compl}$
=$_{eq}$ bouteille$_{Nf}$ (x_1)$_0$

full is a basic adjectival predicate. Its meaning, "filled to the top", is represented as follows, with a specified Locative satellite.

full$_A$ (x_1:<container>(x_1))$_0$
=$_{df}$ filled$_A$ (x_1)$_0$ (d1x_2:top$_N$(x_2)$_0$)$_{LocLimit}$
=$_{eq}$ plein$_A$ (x_1)$_0$

empty is a basic adjectival predicate defined as "containing nothing":

empty$_A$ (x_1:<container>(x_1))$_0$
=$_{df}$ contain$_V$ (x_1)$_0$ (0x_2)$_{Go}$
=$_{eq}$ vide$_A$ (x_1)$_0$

reply is a basic verbal predicate, with a similar predicate frame to <u>say</u>:

reply_V $(x_1$:<human>$(x_1))_{Ag}$ $(x_2$:<text>$(x_2))_{Go}$ $(x_3)_{Rec}$
$=_{df}$ answer_V $(x_1)_{Ag}$ $(x_2)_{Go}$ $(x_3)_{Rec}$
$=_{eq}$ répondre_V $(x_1)_{Ag}$ $(x_2)_{Go}$ $(x_3)_{Rec}$

drink in its intransitive use is derived from transitive <u>to drink</u>, by an argument reduction rule, causing a "generalized" interpretation, which is hard to capture in general terms:

drink_V $(x_1$:<human>$(x_1))_{Ag}$
argument reduction (detrans.)

input:	predicate_V (x_1) $(x_2$:<sel.restr.>$(x_2))_{Go}$
output:	predicate_V (x_1)
meaning:	to predicate_V regularly the thing in a particular way corresponding to the selection restriction.

air is a basic nominal predicate with a French equivalent:

air_N $(x_1)_0$
$=_{df}$ appearance_N $(x_1)_0$
$=_{eq}$ air_{Nm} $(x_1)_0$

lugubrious is a basic adjectival predicate with a French equivalent:

lugubrious_A $(x_1)_0$
$=_{df}$ sorrowful_A $(x_1)_0$
$=_{eq}$ lugubre_A $(x_1)_0$

demand is a basic verbal predicate, whose second argument must refer to or consist of a question:

demand_V $(x_1$:<human>$(x_1))_{Ag}$ $(x_2$:<question>$(x_2))_{Go}$ $(x_3$:<human>$(x_3))_{Rec}$
$=_{df}$ ask_V $(x_1)_{Ag}$ $(x_2)_{Go}$ $(x_3)_{Rec}$
$=_{eq}$ demander_V $(x_1)_{Ag}$ $(x_2)_{Go}$ $(x_3)_{Rec}$

forget is a basic verbal predicate corresponding with <u>oublier</u>:

forget_V $(x_1$:<human>$(x_1))_{Proc}$ $(x_2)_{Go}$
$=_{df}$ Neg remember_V $(x_1)_0$ $(x_2)_{Go}$
$=_{eq}$ oublier_V $(x_1)_{Proc}$ $(x_2)_{Go}$

inquire is a basic verbal predicate whose equivalent is a reflexive verb:

inquire_V $(x_1$:<human>$(x_1))_{Ag}$ $(x_2$:<question>$(x_2))_{Go}$
$=_{df}$ ask_V $(x_1)_{Ag}$ $(x_2)_{Go}$
$=_{eq}$ s'enquérir_V $(x_1)_{Ag}$ $(x_2)_{Go}$

sorry is a basic adjectival predicate which cannot be used attributively in this meaning with a human head noun (*a sorry man). This fact has to be captured somehow in the lexicon, but is not relevant within the scope of this article. There is no adjectival equivalent in French: to be sorry for corresponds with plaindre. Sorry in this use of the word can be paraphrased by to pity of which plaindre is a pure equivalent.

$\text{sorry}_A\ (x_1{:}\langle\text{human}\rangle(x_1))_0\ (x_2{:}\langle\text{animate}\rangle(x_2))_{Ben}$
$=_{df}\ \text{pity}_V\ (x_1)_{Ag}\ (x_2)_{Go}$
$\text{pity}_V\ (x_1)_{Ag}\ (x_2)_{Go}\ =_{eq}\ \text{plaindre}_V\ (x_1)_{Ag}\ (x_2)_{Go}$

confess is a basic verbal predicate:

$\text{confess}_V\ (x_1{:}\langle\text{human}\rangle(x_1))_{Ag}\ (x_2{:}\langle\text{fault}\rangle(x_2))_{Go}$
$=_{df}\ \text{admit}_V\ (x_1)_{Ag}\ (x_2)_{Go}$
$=_{eq}\ \text{avouer}_V\ (x_1)_{Ag}\ (x_2)_{Go}$

ashamed is a basic adjectival predicate, since it cannot be derived from an existing verb; nor can it be productively derived from the noun shame. Used attributively it should be translated into honteux; used predicatively it should be translated into avoir honte. The same holds for a group of English adjectives: hungry, thirsty, afraid, etc. Since the difference exists in French it will have to be handled in the French grammar, possibly by means of markers in the lexicon. Ashamed will be related with both corresponding French expressions:

$\text{ashamed}_A\ (x_1{:}\langle\text{human}\rangle(x_1))_0$
$=_{df}\ \text{feel}_V\ (x_1)_0\ (x_2{:}\text{shame}_N(x_1)_0)_{Go}$
$\text{feel}_V\ (x_1)_0\ (x_2)_{Go}\ =_{eq}\ \text{avoir}_V\ (x_1)_0\ (x_2{:}\langle\text{sensation}\rangle(x_2))_{Go}$
$\text{shame}_N(x_1)_0\ =_{eq}\ \text{honte}_{Nf}\ (x_1)_0$

$\text{ashamed}_A\ (x_1{:}\langle\text{human}\rangle(x_1))_0$
$=_{df}\ [\text{feel}_V\ (x_1)_0\ (x_2{:}\text{shame}_N(x_1)_0)_{Go}]\ (x_1)_0$
$=_{eq}\ \text{honteux}_A\ (x_1)_0$

hang is here treated as a basic verbal predicate, corresponding with baisser when the second argument is "a head":

$\text{hang}_V\ (x_1{:}\langle\text{animate}\rangle(x_1))_{Po}\ (x_2)_{Go}$
$=_{df}\ \text{cause}_V\ (x_1)_{Ag}\ (x_3{:}[\text{down}_{Adv}\ (x_2)_{Po}](x_3))_{Go}$
$=_{eq}\ \text{baisser}_V\ (x_1)_{Ag}\ (x_2{:}\langle\text{head}\rangle(x_2))_{Go}$

head is a basic nominal predicate with a French equivalent:

$\text{head}_N\ (x_1)_0$
$=_{df}\ \{(\text{part}_N\ (x_2)_0\ (x_3{:}\text{body}_N(x_3))_{Part}{:}\ \text{contain}_V\ (x_2)_0\ (x_4{:}\text{brain}_N(x_4))_{Go})_0\}\ (x_1)_0$
$=_{eq}\ \text{tête}_{Nf}\ (x_1)_0$

insist is a basic verbal predicate with a French equivalent:

$insist_V$ $(x_1$:<human>$(x_1))_{Ag}$ $(x_2$:<text>$(x_2))_{Go}$
$=_{df}$ say_V $(x_1)_{Ag}$ $(x_2)_{Go}$ $(x_3$:[$persistent_A$ $(x_1)0](x_3))_{Manner}$
$=_{eq}$ $insister_V$ $(x_1)_{Ag}$ $(x_2)_{Go}$

want is a basic verbal predicate with a French equivalent:

$want_V$ $(x_1$:<animate>$(x_1))_{Po}$ $(x_2)_{Go}$
$=_{df}$ $desire_V$ $(x_1)_{Po}$ $(x_2)_{Go}$
$=_{eq}$ $désirer_V$ $(x_1)_{Po}$ $(x_2)_{Go}$

help is a basic verbal predicate with a French equivalent

$help_V$ $(x_1$:<human>$(x_1))_{Ag}$ $(x_2$:<animate>$(x_2))_{Go}$
$=_{df}$ do_V $(x_1)_{Ag}$ $(x_3)_{Go}$ $(x_2)_{Ben}$
$=_{eq}$ $secourir_V$ $(x_1)_{Ag}$ $(x_2)_{Go}$

bring one's speech to an end is a derived predicate, as was motivated earlier. The special rule forming this kind of predicate is formulated below; note that the rule specifies that the second argument must be a (fully specified) predication. All the lexical information of the participating predicates is supplied.

{$bring_V$ $(d1x_3$:$speech_N(x_3)0$:{$(x_1)_{Poss}$}$(x_3)0)_{Go}$ $(i1x_4$:$end_N(x_4)0$ $(x_3)_{Compl})_{Dir}$}
$(x_1$:<human>$(x_1))_{Ag}$ $(x_2$:<text>$(x_2))_{Go}$

special predicate formation rule:
input: say_V $(x_1)_{Ag}$ $(x_2$:PHI(x2))Go $(x_3$:[$predicate_V$ (x_1)..](x3))Circ
output: {$predicate_V$ ..} $(x_1)_{Ag}$ $(x_2)_{Go}$
meaning: no change of meaning

$bring_V$ $(x_1$:<animate>$(x_1))_{Ag}$ $(x_2)_{Go}$ $(x_3)_{Dir}$
$=_{df}$ $cause_V$ $(x_1)_{Ag}$ $(x_4$:[$reach_V$ $(x_2)_{Proc}$ $(x_3)_{Go}](x_4))_{Go}$

$speech_N$ $(x_1)0$
$=_{df}$ {$(x_2$:$speak_V$ $(x_3)_{Ag}$ $(x_4)_{Go})0$} $(x_1)0$
$=_{eq}$ $discours_{Nm}$ $(x_1)0$

end_N $(x_1)0$ $(x_2)_{Compl}$
$=_{df}$ {$(d1lastx_2$:$point_N(x_2)0)0$} $(x_1)0$ $(x_2)_{Compl}$
$=_{eq}$ fin_{Nf} $(x_1)0$ $(x_2)_{Compl}$

shut up is a basic verbal predicate with a French equivalent:

$shut\text{-}up_V$ $(x_1$:<human>$(x_1))_{Ag}$ $(x_2)_{Go}$
$=_{df}$ put_V $away_{Adv}$ $(x_1)_{Ag}$ $(x_2)_{Go}$
$=_{eq}$ $enfermer_V$ $(x_1)_{Ag}$ $(x_2)_{Go}$

impregnable is a basic adjectival predicate with a French equivalent:

$\text{impregnable}_A\ (x_1)_0$
$=_{df}\ \text{impossible}_A\ (x_2\text{:}[\text{enter}_V\ (x_3)_{Ag}\ (x_1)_{Go}](x_2))_0$
$=_{eq}\ \text{impénétrable}_A\ (x_1)_0$

go away is a derived predicate, as was illustrated earlier:

$\text{go}_V\ \text{away}_{Adv}\ (x_1)_{Ag}$
input: $\text{predicate}_V\ (x_1)..(x_n)$
output: $\text{predicate}_V\ \text{away}_{Adv}\ (x_1)..(x_n)$
meaning: $\text{predicate}_V\ (x_1)..(x_n)\ (\text{i}1x_o\text{:place}_N(x_o)0)\text{So}\ (\text{i}1x_p\text{:place}_N(x_p)0)\text{Dir}$

$\text{go}_V\ (x_1)_{Ag}$
$=_{df}$ -
$\text{go}_V\ \text{away}_{Adv}\ (x_1)_{Ag}\ =_{eq}$ s'en aller $(x_1)_{Ag}$

puzzled is a derived predicate, an adjectival predicate formed on the basis of a verbal predicate, but has a basic adjectival French equivalent:

$\text{puzzled}_A\ (x_1)_0 =_{eq} \text{perplexe}_A\ (x_1)_{Fo}$

input: PartPerf predicate_V
output: {.....}$_A$
meaning: no change of meaning

$\text{puzzle}_V\ (x_1)_{Fo}\ (x_2\text{:<human>}(x_2))_{Go}$
$=_{df}\ \text{perplex}_V\ (x_1)_{Fo}\ (x_2)_{Go}$

odd is a basic adjectival predicate:

$\text{odd}_A\ (x_1)_0$
$=_{df}\ \text{strange}_A\ (x_1)_0$
$=_{eq}\ \text{bizarre}_A\ (x_1)_0$

grown-up is either a basic predicate, because of its idiomatic meaning, or a derived nominal predicate, formed on the basis of an adjectival predicate (which in turn is formed on the basis of a verbal predicate, like puzzled). Both possibilities can be handled, but in different ways:

$\text{grown-up}_N\ (x_1\text{:<human>}(x_1))_0$
$=_{df}\ \text{adult}_N\ (x_1)_0$
$=_{eq}\ \text{grande-personne}_{Nf}\ (x_1)_0$

input: $predicate_A\ (x_1)_0$
output: $predicate_N\ (x_1)_0$
meaning: $\{(i1x_2{:}person_N\ (x_2)_0{:}predicate_A(x_2)_0)_0\}\ (x_1)_0$

$person_N(x_1)_0{:}grown\text{-}up_A(x_1)_0$

$person_N\ (x_1)_0$
$=_{df}\ human_N(x_1{:}being_N(x_1)_0)_0$
$=_{eq}\ personne_{Nf}\ (x_1)_0$

$grown\text{-}up_A\ (x_1)_0$
$=_{df}\ adult_A\ (x_1)_0$
$=_{eq}\ grand_A\ (x_1)_0$

continue (on) has two entries in the lexicon with different meanings and predicate frames; its equivalent continuer has the same behaviour:

$continue_V\ (x_1{:}\langle animate\rangle(x_1))_{Ag}\ (x_2)_{Go}$
$=_{df}\ start_V\ again_{Adv}\ (x_1)_{Ag}\ (x_2)_{Go}$
$=_{eq}\ continuer_V\ (x_1)_{Ag}\ (x_2)_{Go}$

$continue_V\ (x_1)_{Ag/Proc}$
$=_{df}\ go\text{-}on_V\ (x_1)_{Ag/Proc}$
$=_{eq}\ continuer_V\ (x_1)_{Ag/Proc}$

journey is a basic nominal predicate:

$journey_N\ (x_1)_0$
$=_{df}\ trip_N\ (x_1)_0$
$=_{eq}\ voyage_{Nm}\ (x_1)_0$

Translation

We shall now consider to what extent the translation process works according to the model developed so far. The English predications are translated into French predications by mechanical substition of the equivalent French predicates for the English ones with the use of the equivalences given above.

(20) DECL {Pret habiter$_V$ (i1x$_1$:buveur$_{Nm}$(x$_1$)0)$_{Po}$
(d1nextx$_2$:planète$_{Nf}$(x$_2$)0)$_{GoSubj}$}

- Note that the category of nominal predicates is specified as to gender in the French predication.
- Tippler was translated as the verbal predicate drink with N.A. marker, the French N.A.-rule takes the stem of the verbal predicate, which in the case of boire is buv-, and adds the suffix "eur".
- The model does not yet provide for the translation of ordinal, and perhaps other lexical, term operators; this has to be supplied.

(21) DECL {Pret {(i1x$_3$:visite$_{Nf}$(x$_3$)0:très court$_A$(x$_3$)0)0} ([+prox]1x$_4$)0$_{Subj}$}
but
DECL {Pret plonger$_V$ (Ax$_4$)$_{FoSubj}$
(d1x$_5$:prince$_{Nm}$(x$_5$)0:petit$_A$(x$_5$)0)$_{GoObj}$
(i-x$_6$:abattement$_{Nm}$(x$_6$)0:grand$_A$(x$_6$)0)$_{Dir}$}

- The translation of conjunctions is another lacuna in the model; there will have to be equivalences for the lexical non-predicative elements which figure in predications.
- Adjectives agree with nouns in number and gender. The number of a noun group is incorporated in a term operator, the gender in the category of the noun. One could think of specifying the adjectival category in accordance with the nominal, but agreement can also be handled by the expression rules, as it occurs in other constructions as well.
- Tres court is formed by the French intensifying rule on the basis of the equivalent of short.

(22) DECL {Pret dire$_V$ (Ax$_5$)$_{AgSubj}$
(x$_7$:[INT {PresProgr faire$_V$ (Hx$_1$)$_{AgSubj}$
(Qx$_8$)$_{GoObj}$
([-prox]x$_9$)$_{Loc}$}](x$_7$))$_{GoObj}$
(d1x$_1$:buveur$_{Nm}$(x$_1$)0:
Pret trouver$_V$ (Ax$_5$)$_{ProcSubj}$
(Rx$_1$)$_{GoObj}$
(x$_{10}$:[PartPerf s'installer$_V$ (x$_5$)$_{Ag}$
(x$_{11}$:silence$_{Nm}$(x$_{11}$)0)$_{Circ}$
((i1x$_{12}$:collection$_{Nf}$(x$_{12}$)0
(imx$_{13}$:bouteille$_{Nf}$(x$_{13}$)0:
vide$_A$(x$_{13}$)0)$_{Part}$)
and also
(i1x$_{14}$:collection$_{Nf}$(x$_{14}$)0
(imx$_{15}$:bouteille$_{Nf}$(x$_{15}$)0:
plein$_A$(x$_{15}$)0)$_{Part}$))$_{Loc\text{-}fr}$](x$_{10}$))$_{Circ}$)$_{Rec}$}

- s'installer is the result of application of the French reflexive argument reduction rule, triggered by translation of settle down marked "R".

(23) DECL {Pret répondre$_V$ (d1x$_1$:buveur$_{Nm}$(x$_1$)0)$_{AgSubj}$
(x$_{16}$:[DECL {PresProgr boire$_V$ (Sx$_1$)$_{AgSubj}$}](x$_{16}$))$_{GoObj}$
(i1x$_{17}$:air$_{Nm}$(x$_{17}$)0:lugubre$_A$(x$_{17}$)0)$_{Manner}$}

(24) DECL {Pret demander$_V$ (d1x$_5$:prince$_{Nm}$(x$_5$)0:petit$_A$(x$_5$)0)$_{AgSubj}$
(x$_{18}$:[INT {PresProgr boire$_V$ (Hx$_1$)$_{AgSubj}$
(Qx$_{19}$)$_{Rsn}$}](x$_{18}$))$_{GoObj}$}

(25) DECL {Pret répondreV (d1x$_1$:buveur$_{Nm}$(x$_1$)0)$_{AgSubj}$
(x$_{20}$:[DECL {(x$_{21}$:[Poss oublier$_V$ (Sx$_1$)$_{Proc}$
(0)$_{Go}$](x$_{21}$))$_{Purp}$}](x$_{20}$))$_{GoObj}$}

(26) DECL {Pret s'enquérir$_V$ (d1x$_5$:prince$_{Nm}$(x$_5$)0:petit$_A$(x$_5$)0:
Pret plaindre$_V$ (Rx$_5$)0$_{Subj}$
(Ax$_1$)$_{Go}$
(already)$_{Ti}$)$_{AgSubj}$
(x$_{22}$:[INT {oublier$_V$ (x$_1$)$_{Proc}$
(Qx$_{23}$)$_{Go}$}](x$_{22}$))$_{GoObj}$}

- Translation of non-predicative adverbials such as already will have to be supplied.
- Plaindre is the equivalent of pity, with Zero and Goal argument.

(27) DECL {Pret avouer$_V$ (d1x$_1$:buveur$_{Nm}$(x$_1$)0)$_{AgSubj}$
(x$_{24}$:[DECL {oublier$_V$ (x$_1$)$_{Proc}$
(x$_{25}$:[Pres honte$_{Nf}$ (Sx$_1$)0](x$_{25}$))$_{Go}$}](x$_{24}$))$_{GoObj}$
(x$_{26}$:[PartPres baisser$_V$ (x1)$_{Ag}$
(d1x$_{27}$:tête$_{Nm}$(x$_{27}$)0:{(x$_1$)$_{Poss}$}(x$_{27}$)0)$_{Go}$](x$_{26}$))$_{Circ}$}

- I assume for the moment that honte with a zero human argument will be supported with avoir in the expression rules. This would be a special feature in French for the mentioned group of nouns designating sensations. Another possibility consists of taking honte as a one-place predicate with an Experiencer argument, which would be expressed by avoir with a non-verbal predicate.

(28) DECL {Pret insister$_V$ (d1x$_5$:prince$_{Nm}$(x$_5$)0:
petit$_A$(x$_5$)0:
Pret dēsirer$_V$ (Rx$_5$)$_{PoSubj}$
(x$_{28}$:[Inf secourir$_V$(x$_5$)$_{Ag}$
(Ax$_1$)$_{Go}$](x$_{28}$))$_{GoObj}$)$_{AgSubj}$
(x$_{29}$:[INT {honte$_{Nf}$ (x$_1$)0$_{Subj}$
(Qx$_{30}$)$_{So}$}](x$_{29}$))$_{GoObj}$}

(29)DECL {Pret [mener$_V$ (d1x$_{31}$:discours$_{Nm}$(x$_{31}$)0:{(x$_1$)$_{Poss}$}(x$_{31}$)0)$_{Go}$
(i1x$_{32}$:fin$_{Nf}$(x$_{32}$)0(x$_{31}$)$_{Compl}$)$_{Dir}$]
(d1x$_1$:buveur$_N$(x$_1$)0)$_{AgSubj}$
(x$_{33}$:[DECL {honte$_{Nf}$ (x$_1$)0
(x$_{34}$:[boire$_V$(x$_1$)$_{Ag}$](x$_{34}$))$_{So}$}](x$_{33}$))$_{GoObj}$}
and
DECL {Pret enfermer$_V$ (Ax$_1$)$_{AgSubj}$
(Ax$_1$)$_{GoObj}$
(i1x$_{35}$:silence$_{Nm}$(x$_{35}$)0:impēnētrable$_A$(x$_{35}$)0)$_{Loc}$}

- Note that the third deparaphrasing component might transform the complex predicate mener son discours a la (sa) fin into achever (son discours).

(30)and
DECL (Pret s'en aller$_V$ (d1x$_5$:prince$_{Nm}$(x$_5$)0:petit$_A$(x$_5$)0)$_{AgSubj}$
(x$_{36}$:[perplexe$_A$ (x$_5$)0](x$_{36}$))$_{Circ}$}

- To go away corresponds with s'en aller, which is idiomatic and difficult to represent in predicate format.

(31)DECL {Pret dire$_V$ (Ax5)$_{AgSubj}$
(x37:[DECL certainly {Pres très très bizarre$_A$
dmx38:grande-personne$_{Nf}$(x38)0)0$_{Subj}$}](x37))$_{GoObj}$
(Ax5)$_{Rec}$
(x39:[{Pret continuer$_V$ (Ax5)$_{AgSubj}$
(d1x40:voyage$_{Nm}$(x40)0:{(x5)$_{Poss}$}(x40)0)$_{Loc}$}](x39))$_{Circ}$}

- The translation of certainly is not yet provided, as was to be expected.
- Grown-up is translated into grande personne; the exact representation of this expression is left unspecified for the moment. In the equivalences the two possibilities are indicated.

Evaluation

As has been signalled, there is a set of lexical elements figuring in the predication which are not handled by the equivalences since they do not act as predicates. These lexical elements form part of a restricted group of syntactic categories: (ordinal) term operators, conjunctions and adverbials. For the moment we could deal with this problem by formulating a second set of equivalences relating the lexical non-predicative elements of the two languages. The sample text would require the following additional equivalences:

next	$=_{eq'}$	suivant
but	$=_{eq'}$	mais
already	$=_{eq'}$	déjà
and	$=_{eq'}$	et
certainly	$=_{eq'}$	décidément

With this addition all lexical elements of the input predications can be translated.

The group of lexical term operators differs from the others, since it is the only syntactic category acknowledged in the FG terminology. Conjunctions are included here because their representation is not yet accounted for by FG. Conjunctions may well be expressed by other means such as operators or satellite constructions. The same holds for adverbials. Some types of adverbials are likely to be treated in terms of (predicate) operators. In that case, the expression of both conjunctions and adverbials will be taken care of by the expression rules, which are part of the languages' grammars. We have been assuming that the grammatical structure of the predication including the operators is not affected by translation. Now it is time to verify this assumption.

2.3 Reformulation

Until now it was assumed that translation of predications is accomplished by substitution of the lexical components of the predication. A closer look shows that this does not suffice. Apart from the lexical elements predications contain grammatical elements: operators and functions. Languages may differ in this respect too. For example, the assignment of syntactic functions varies across languages: each language has a range of semantic functions accessible to subject and object assignment (see Dik(1978, pp. 75-79) for a more complete account of this phenomenon). English and French differ in this respect; English allows Beneficiary arguments to function as subject and as object, French does not:

(1a) Mary was bought a car.
(1b) Peter bought Mary a car.
(1c)*Marie a été acheté une voiture.
(1d)*Pierre a acheté Marie une voiture.

With respect to predicate operators English differs from French too: in French the Progressive aspect is not used as productively as in English, and on the other hand English does not make as strict a distinction between perfective and imperfective preteritum as French does. The mapping of grammatical operators and functions from one language to another need not be a matter of simple one-to-one equivalence. These kinds of differences have to be handled as well and should be incorporated into the model as a kind of syntactic check and adjustment.

FG distinguishes three systems of grammatical operators:
- predication operators, which define the illocutionary value of the predication
- predicate operators, which define the Tense and Aspect of the predication
- term operators, which define definiteness, number and possibly other properties of terms.

FG distinguishes three systems of functions:
- semantic functions, which specify the roles in the state of affairs designated by the predication (Agent, Goal, etc.)
- syntactic functions, which specify the point of view from which the state of affairs is presented (Subject, Object)
- pragmatic functions, which specify the informational status of the constituents of the predication (Topic, Focus, Theme, Tail).

Each of these grammatical systems can be seen as a finite set of elements, of which each language uses a subset. For each language pair the differences on this level are therefore restricted. Ideally there is again a one-to-one correspondence, when the two languages use exactly the same subset of the interlingual set. In theory there are three possible deviations, as is illustrated in figure 4:

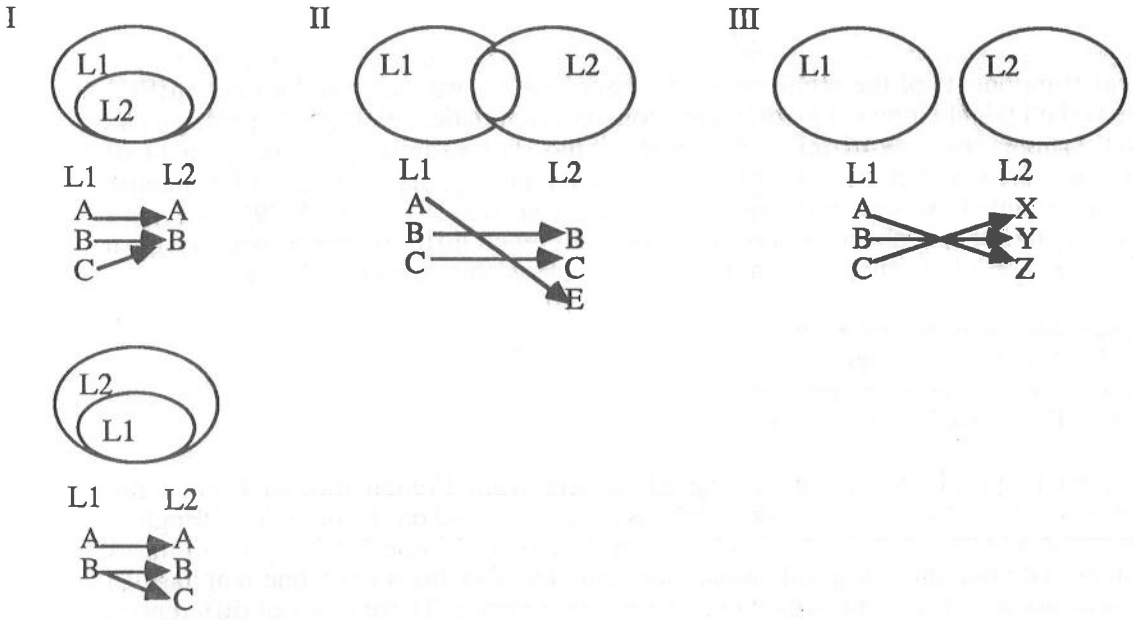

figure 4. Non-correspondences in grammatical systems

Explanation:
I (inclusion): a set of grammatical elements of language1 may include the set of grammatical elements of language2, or vice versa. In that case, an operator or function of one language corresponds with two or more distinguished ones of the other language. Or, alternatively, an operator or function of one language is simply not expressed in the other language.
II (intersection): the sets of grammatical elements of language1 and language2 partly overlap. In that case the elements that are not shared will have to be related.
III (independence): the sets of grammatical elements differ completely. In that case all the correspondences will have to be established.

This part of the model is necessarily language pair dependent: for each particular pair of languages a contrastive analysis will have to be performed in order to establish the correspondences on the grammatical level. A complete contrastive analysis of the English and French grammatical systems is postponed to further investigations. For now, the extensions of the model are restricted to the needs of translation of the sample text.

A. **Predication operators**

The use of predication operators does not vary between English and French.

B. **Predicate operators**

The earlier mentioned difference in the use of progressive and (im)perfective aspect has to be accounted for. This can be done by adopting the following rules:

Pret	->	PretPerf/[+dyn]
Pret	->	PretImpf/[-dyn]
PresProgr	->	Pres
PretProgr	->	PretImpf

Every state of affairs is characterized by the feature [+/- dyn]: actions and processes are dynamic, states and positions are not dynamic. Dynamism is reflected in the semantic function of the first argument (Agent, Force, Processed vs. Positioner, 0). If the English predication contains the predicate operator "Pret" this is specified as perfective if the state of affairs is dynamic and as imperfective if it is not. French does not express the progressive aspect, except when it is stressed, in which case lexical means are adopted (etre en train de).

C. **Term operators**
Lexical term operators, such as the ordinal next, have to be translated with the use of equivalences, for example:

next $=_{eq}$ suivant

D. **Semantic functions**
The sets of semantic functions of English and French do not show differences. If there is a change of semantic function this is caused by a difference on the lexical level.

E. **Syntactic functions**
syntactic function assignment will have to be checked for agreement with the French system. Furthermore, when semantic functions have been changed in the translation procedure, this may have consequences for the assignment of syntactic functions. For example, in the seventh sentence "he was sorry for him" is translated into "il le plaignait". The second (Beneficiary) argument of sorry does not have object function, whereas the Goal-argument of plaindre functions as the syntactic object. The check on syntactic functions will have to account for this kind of adjustments as well.

F. **Pragmatic functions**
pragmatic functions have been left out of the discussion from the beginning.

G. **Special: Coreference**
Unlike English, French makes productive use of the reflexive mode of verbs, both with Goal arguments as well as with Recipient and Beneficiary arguments. Sentences as those under (1) correspond with sentences of the form (2):

(1a) He shut himself up
(1b) He said that to himself
(1c) He bought himself a car

(2a) Il s'est enfermé
(2b) Il se disait cela
(2c) Il s'est acheté une voiture

This can be handled in the following way: if there is a second or third anaphoric argument it will be removed under referential identity and the predicate will be marked for reflexive (rule application).

With these extensions the model can be tested on its validity for the translation of English predications into the corresponding French syntactically correct predications.

2.4 Test

The French predications arrived at by lexical substitution of the predicates will now be checked and adjusted according to the French grammatical system. The principles and rules formulated in the preceding paragraph will be applied where necessary.

(31) DECL {PretImpf habiter$_V$ (i1x$_1$:buveur$_{Nm}$(x$_1$)0)$_{Po}$
(d1suivantx$_2$:planète$_{Nf}$(x$_2$)0)$_{GoSubj}$}

- The first argument of habiter has Positioner function; the predication designates a position, which is a non-dynamic state of affairs. The Pret-operator is specified as imperfective, according to the earlier stated rule.
- The ordinal term operator next is replaced by its French equivalent suivant.

(32) DECL {PretImpf {(i1x$_3$:visite$_{Nf}$(x$_3$)$_0$:très court$_A$(x$_3$)0)0} ([+prox]1x$_4$)0Subj}
mais
DECL {PretPerf plonger$_V$ (Ax$_4$)$_{FoSubj}$
(d1x$_5$:prince$_{Nm}$(x$_5$)$_0$:petit$_A$(x$_5$)0)$_{GoObj}$
(i-x$_6$:abattement$_{Nm}$(x$_6$)$_0$:grand$_A$(x$_6$)0)$_{Dir}$}

- The first of the two coordinated predications designates a (non-dynamic) state; the second designates a process, which is dynamic.
- The conjunction but is replaced by its French equivalent mais.

(33) DECL {PretPerf dire$_V$ (Ax$_5$)$_{AgSubj}$
(x$_7$:[INT {Pres faire$_V$ (Hx$_1$)$_{AgSubj}$
(Qx$_8$)$_{GoObj}$
([-prox]x$_9$)$_{Loc}$}](x$_7$))$_{GoObj}$
(d1x$_1$:buveur$_{Nm}$(x$_1$)$_0$:
PretPerf trouver$_V$ (Ax$_5$)$_{ProcSubj}$
(Rx$_1$)$_{GoObj}$
(x$_{10}$:[PartPerf s'installer$_V$ (x$_5$)$_{Ag}$
(x$_{11}$:silence$_{Nm}$(x$_{11}$)0)$_{Circ}$
((i1x$_{12}$:collection$_{Nf}$(x$_{12}$)0
(imx$_{13}$:bouteille$_{Nf}$(x$_{13}$)$_0$:
vide$_A$(x$_{13}$)0)$_{Part}$)
et aussi
(i1x$_{14}$:collection$_{Nf}$(x$_{14}$)0
(imx$_{15}$:bouteille$_{Nf}$(x$_{15}$)$_0$:
plein$_A$(x$_{15}$)0)$_{Part}$))$_{Loc\text{-}fr}$](x$_{10}$))$_{Circ}$)$_{Rec}$}

- The progressive marker is removed from the Pres-operator.
- And also is translated as et aussi.

(34) DECL {PretPerf répondre$_V$ (d1x$_1$:buveur$_{Nm}$(x$_1$)0)$_{AgSubj}$
(x$_{16}$:[DECL {Pres boire$_V$
(Sx$_1$)$_{AgSubj}$}](x$_{16}$))$_{GoObj}$
(i1x$_{17}$:air$_{Nm}$(x$_{17}$)0:lugubre$_A$(x$_{17}$)0)$_{Manner}$}

- The Pret and PresProgr operators are adjusted to the French aspect system.

(35) DECL {PretPerf demander$_V$ (d1x$_5$:prince$_{Nm}$(x$_5$)0:petit$_A$(x$_5$)0)$_{AgSubj}$
(x$_{18}$:[INT {Pres boire$_V$ (Hx$_1$)$_{AgSubj}$
(Qx$_{19}$)$_{Rsn}$}](x$_{18}$))$_{GoObj}$}

(36) DECL {PretPerf répondreV (d1x$_1$:buveur$_{Nm}$(x$_1$)0)$_{AgSubj}$
(x$_{20}$:[DECL {(x$_{21}$:[Poss oublier$_V$ (Sx$_1$)$_{ProcSubj}$
(0)$_{Go}$](x$_{21}$))$_{Purp}$}](x$_{20}$))$_{GoObj}$}

(37) DECL {PretPerf s'enquérir$_V$ (d1x$_5$:prince$_{Nm}$(x$_5$)0:petit$_A$(x$_5$)0:
PretImpf plaindre$_V$ (Rx$_5$)0$_{Subj}$
(Ax$_1$)$_{GoObj}$
(déjà)$_{Ti}$)$_{AgSubj}$
(x$_{22}$:[INT {oublier$_V$ (x$_1$)$_{Proc}$
(Qx$_{23}$)$_{Go}$}](x$_{22}$))$_{GoObj}$}

- The second argument of plaindre is assigned Object function.
- The adverbial déjà is the French equivalent of already.

(38) DECL {PretPerf avouerV (d1x$_1$:buveur$_{Nm}$(x$_1$)0)$_{AgSubj}$
(x$_{24}$:[DECL {oublier$_V$ (x$_1$)$_{ProcSubj}$
(x$_{25}$:[Pres honte$_{Nf}$ (Sx$_1$)0](x$_{25}$))$_{Go}$}](x$_{24}$))$_{GoObj}$
(x$_{26}$:[PartPres baisserV (x1)Ag
(d1x$_{27}$:tête$_{Nm}$(x$_{27}$)0:{(x$_1$)$_{Poss}$}(x$_{27}$)0)$_{Go}$](x$_{26}$))$_{Circ}$}

(39) DECL {PretPerf insister$_V$ (d1x$_5$:prince$_{Nm}$(x$_5$)0:
petit$_A$(x$_5$)0:
Pret désirer$_V$ (Rx$_5$)$_{PoSubj}$
(x$_{28}$:[Inf secourir$_V$(x$_5$)$_{Ag}$
(Ax$_1$)$_{Go}$](x$_{28}$))$_{GoObj}$)$_{AgSubj}$
(x$_{29}$:[INT {honte$_{Nf}$ (x$_1$)0$_{Subj}$
(Qx$_{30}$)$_{So}$}](x$_{29}$))$_{GoObj}$}

(40)DECL {PretPerf [mener$_V$ (d1x$_{31}$:discours$_{Nm}$(x$_{31}$)0:{(x$_1$)$_{Poss}$}(x$_{31}$)0)$_{Go}$
(i1x$_{32}$:fin$_{Nf}$(x$_{32}$)0 (x$_{31}$)$_{Compl}$)$_{Dir}$]
(d1x$_1$:buveur$_N$(x$_1$)0)$_{AgSubj}$
(x$_{33}$:[DECL {honte$_{Nf}$ (x$_1$)0
(x$_{34}$:[boire$_V$(x$_1$)$_{Ag}$](x$_{34}$))$_{So}$}](x$_{33}$))$_{GoObj}$}
and
DECL {PretPerf s'enfermer$_V$ (Ax$_1$)$_{AgSubj}$
(i1x$_{35}$:silence$_{Nm}$(x$_{35}$)0:impénétrable$_A$(x$_{35}$)0)$_{Loc}$}

- The Goal argument of enfermer, (Ax$_1$), has been removed under referential identity with the first argument; the predicate has undergone the reflexive argument reduction rule.

(41)et
DECL (PretPerf s'en aller$_V$ (d1x$_5$:prince$_{Nm}$(x$_5$)0:petit$_A$(x$_5$)0)$_{AgSubj}$
(x$_{36}$:[perplexe$_A$ (x$_5$)0](x$_{36}$))$_{Circ}$}

(42)DECL {PretPerf se dire$_V$ (Ax$_5$)$_{AgSubj}$
(x$_{37}$:[DECL décidément {Pres tres tres bizarre$_A$
(dmx$_{38}$:grande-personne$_{Nf}$(x$_{38}$)0)0$_{Subj}$}](x$_{37}$))$_{GoObj}$
(x$_{39}$:[{PretPerf continuer$_V$ (Ax$_5$)$_{AgSubj}$
(d1x$_{40}$:voyage$_{Nm}$(x$_{40}$)0:{(x$_5$)$_{Poss}$}(x$_{40}$)0)$_{Loc}$}](x$_{39}$))$_{Circ}$}

- Certainly is translated into décidément.
- The anaphoric Recipient argument (Ax$_5$) has been removed and the predicate has been made reflexive, according to the coreference check.
- The predicate operator of the satellite of circumstance is specified as perfective according to the rule, but according to the nature of such a satellite it should be imperfective. This is one of the further refinements to be realized.

Text

The French predications arrived at by the translation process in two phases can now be expressed. As the French grammar has not yet been developed the expression rules applied here are tentative. Moreover, since pragmatic functions have been left out, certain assumptions have been made about ordering.

(43) La planète suivante était habitée par un buveur.

- The ordinal term operator suivant agrees with the head noun, like other adjectives.
- The "imparfait" is used to express the PretPerf-tense; the "passé simple" for the PretPerf. Perfective is also expressed by the "passé composé", but the passé simple is preferred with written stories.

(44) C'était une visite très courte, mais elle plongea le petit prince en grand abattement.

(45) "Que fais-tu là?", dit-il au buveur, qu'il trouva installé en silence devant une collection de bouteilles vides et aussi une collection de bouteilles pleines.

- The expression rules should incorporate the different inversion rules in French.
- The past participle of a reflexive verb is not accompanied by the reflexive marker "se".
- Quote arguments preferably occupy the first position of the sentence. In fact this is a matter of style.

(46) "Je bois", répondit le buveur d'un air lugubre.

(47) "Pourquoi bois-tu?", demanda le petit prince.

(48) "Pour que j'oublie", répondit le buveur.

- The Poss operator is expressed by the "subjonctif".

(49) "Oublier quoi?", s'enquit le petit prince, qui déjà le plaignait.

(50) "Oublier que j'ai honte", avoua le buveur en baissant sa tête.

- Possessive adjectives are not expressed in French in cases like these, i.e. when they are combined with parts of the body, under referential identity, as in "Il s'est cassé la jambe", "Il avait mal au dos". It is not obvious that the expression rules should handle this phenomenon, since the possessive restrictor is present in the underlying predication. The syntactic checks should then be extended so as to account for this change of structure.
- Nominal predicates of the class of honte, faim, peur, etc. are supplied with the auxiliary avoir for the expression of tense and aspect.

(51) "Honte de quoi?", insista le petit prince, qui désirait le secourir.

(52) "Honte de boire", le buveur mena son discours a la fin et il s'enferma dans un silence impénétrable.

- As has been mentioned before, the final version of this sentence would be something like: "Honte de boire", acheva le buveur.

(53) Et le petit prince s'en fut, perplexe.

(54) "Les grandes personnes sont dēcidément trēs trēs bizarres", il se dit comme il continua sur son voyage.

Comparison with the original

The sentences obtained can be said to be correct French sentences corresponding with the English sentences. Finally the text can be compared to the original in order to see and explain to what extent the sentences may have changed during the translation process. The original French text is given below:

La planète suivante était habitée par un buveur.
Cette visite fut très courte mais elle plongea le petit prince dans une grande mélancolie.
"Que fais-tu la?" dit-il au buveur, qu'il trouva installé en silence devant une collection de bouteilles vides et une collection de bouteilles pleines.
"Je bois", répondit le buveur d'un air lugubre.
"Pourquoi bois-tu?" lui demanda le petit prince.
"Pour oublier", répondit le buveur.
"Pour oublier quoi?" s'enquit le petit prince qui déjà le plaignait.
"Pour oublier que j'ai honte", avoua le buveur en baissant la tête.
"Honte de quoi?" insista le petit prince qui désirait le secourir.
"Honte de boire!", acheva le buveur qui s'enferma définitivement dans le silence.
Et le petit prince s'en fut, perplexe.
"Les grandes personnes sont décidément très, très bizarres", se disait-il en lui-meme durant le voyage.

Some of the differences between the sentences of the preceding section and these can be blamed on the English translator:
- Mélancolie in the second sentence has been translated into dejection, which is more general and corresponds with abattement. Melancoly is a more loaded state of mind.
- In the fifth sentence the anaphoric recipient argument has been omitted.
- The embedded predications of the seventh and eighth sentence have the form of satellites of purpose in the original text, pursuing the form of the first answer of the tippler.
- The tenth sentence shows several changes of construction:
1) achever has been translated by to bring one's speech to an end, which has caused a change of construction. If it had been translated into to conclude or to finish, the direct discourse could have been kept intact;
2) the relative clause has been replaced by a coordinated sentence;
3) the circumstancial satellite définitivement has been transformed into an adjectival restrictor with silence;
4) the prepositional phrase en lui-meme, which might be treated as a kind of satellite, has disappeared in the translation.
- In the twelfth sentence the prepositional construction of the satellite of time "durant le voyage" has been replaced with a predication construction in a satellite of circumstance: "as he continued on his journey", where the definite term "le voyage" has been extended with a possessive restrictor.

There are also some slight changes in the form of the sentences which in most cases can be considered as stylistic differences:
- the use of the subjunctive is rather formal, as is the use of a pendant que-clause. The use of infinitives (with a preposition) and of participles is more common. One could think of a parameter indicating formal or informal style to be given with the translation procedure.
- the use of perfective and imperfective tense is governed by a complex of principles which have to do with the nature and the construction of the text. Generally speaking, in sentences that define the setting of a story the imparfait is used, and in sentences that present the events the passé simple is used. But there are a lot of extra aspects which have to be considered. In this text, for example, it appears that the first and the last sentence are

reserved for the imparfait. The use of tense and aspect within the scope of a text is a topic for further research.

There are a number of aspects which deserve deeper investigation, but in general the model can be said to produce a valid translation of the input text.

3. Prolog program

The findings of the preceding sections are now ready to be translated into a computer program. This program should be able to translate the English predications of the sample text into the corresponding French predications. The program is written in Prolog, a so-called logic programming language.

Prolog statements are to be seen as relationships, expressed by a predicate taking a number of arguments (nuclear predications in terms of FG). Arguments can be constants or variables; constants are known entities, variables are supposed to be bound to an entity. Variables are characterized by an initial capital letter.

The working of Prolog relies upon the existence of a database. A Prolog database consists of clauses, which can be facts or rules. Rules consist of a head and a body, connected by the symbol ":-", which is pronounced "if". The body of a rule consists of one or more statements ("goals"), that is facts or heads of rules, which have to be true in order for the head to be true, for example:

(55) term_operator(X) :- subsystem_op(X).

This rule states that for some entity X (a variable), the property "X is a term operator" holds if "X is a subsystem op" is true. Facts are rules with an empty body, which is the same as a body which always succeeds ("true"), for example:

(56) term_operator(d1).
(57) subsystem_op(prox).

Each predicate is characterized by its name and by the number of arguments it combines with, its quantitative valency or "arity". Predicates are defined by the rules and facts in which they appear as the head; examples (1) and (2) define the predicate "term_operator/1", the predicate with the name "term_operator" and arity 1.
Entering a statement can be seen as asking Prolog to verify it, by matching it with the facts and rules in the database and giving values to the unbound variables. Upon entering (4), Prolog will give (5) as an answer:

(58) term_operator(Op).
(59) Op=prox.

This will be the first solution found, and means: "true when Op is bound to d1". Asking for more solutions (by typing a semicolon) will give (6):

(60) Op=d1;
no.

Lists are the main datastructure used in Prolog. Lists are terms of the form "[Head|Tail]", consisting of constants, variables and/or lists.

Translation is here considered to be a relationship between two predications. The program analyzes the English predication into subparts, which are transferred to the French predication; the in- and output are lists of elements which themselves may be lists. The subparts which the program identifies are stated in the following scheme, describing the syntax of predications.

Predication
[predication, conjunction, predication]
[predication operator, subpredication]

Subpredication
[predicate operator, predicate, category, term, sem/synt function]
[predicate operator, predicate, category, arguments]
[predicate operator, [term, sem function], term, zero(Subj)]
[satellite, sat function]

Arguments
[term, sem/synt function, arguments]
[satellite, sat function, arguments]
[] (= empty list)

Satellite
[adverbial]
[ref, restrictor, ref]
term

Term
[ref]
[subsystem operator, ref]
[ref, predication, ref]
[ref, subpredication, ref]
[term operator, ref, restrictor]
[term operator, ref, restrictors]
[term, conjunction, term]

Restrictors
[restrictor, restrictors]
[]

Restrictor
[subpredication]
if necessary supplied with a dummy predicate operator

Notes
1) A predication may consist of two coordinated predications; by this recursive definition multiple coordinations can be handled. Note that there is a problem here, concerning excessive structure: all multiple coordinations will be analysed as a construction of binary coordinations. This problem has not been handled yet, since it was not relevant for the purposes of this program.
2) A single predication consists of a predication operator and a "subpredication", a notion especially designed for this purpose.
3) A subpredication consists of a predicate operator followed by a predicate with its category (or a term with a semantic function), and one or more arguments or of a single satellite (sentence 6).
4) On two matters a distinction has been made as to the number of elements of a sort, with arguments and with restrictors. One argument or restrictor is taken apart, two or more of them are considered a list. This has been done in order to restrict the number of brackets where possible.

5) Arguments here is a term used for both arguments and satellites; it stands for terms with a semantic function.
6) Satellite terms can take other forms than argument terms in some cases, in particular adverbials and subpredications without tense.
7) Terms can consist of coordinated terms, in the same way as predications. Simple terms consist of:

- only a referent (the index of "x");
- a subsystem term operator with a referent;
- a (sub)predication identifying a referent;
- a term operator, referent and one or more restrictors.

8) A restrictor is a subpredication, which is supplied with a dummy operator in case of absence of the predicate operator. Most nominal and adjectival restrictors lack a tense operator, but are essentially the same as subpredications. In this way the subpredication procedures can be applied to restrictors without redefinition.

Working: stepwise decomposition & transfer
For each subpart of a predication there are certain routines to be run. Some subparts can be transferred unchanged, other subparts have to be further analyzed and some have to be substituted or adjusted. As the predication is analyzed in the course of the program, at each point the appropriate program part is entered. The stepwise decomposition and transfer is guided by the structure of the input predication.

Modularity
Apart from the main "body", the program contains five restricted, more or less independent modules which contain for the largest part language-specific information:

1) Definitions
2) Syntax checks
3) E-fund
4) F-fund
5) Equivalences

ad 1) The definitions contain information about non-lexical elements which may vary across languages: opeators and functions. The definitions correspond with the source language grammar and specify the grammatical features to be expected in the input predication.
ad 2) The syntax checks deal with structural changes of the predication, meeting the requirements imposed by the target language grammar.
ad 3&4) The funds of the two languages contain all the lexical information necessary for the translation procedure. This information consists of predicate frames of fundamental predicates and specifications of predicate formation rules.
ad 5) The equivalences relate the input predicates to the target language fund.

In the following section the full text of the program is given, because reading the program text is the best way to find out the precise working of a Prolog program. The program text is supplied with commentary indicating the different parts and procedures; comments are enclosed between "/*" and "*/".

There are two more things one needs to know about Prolog in order to be able to fully understand the course of the process. First, by trying to satisfy a goal, Prolog runs through its database from top to bottom; the first solution found is the solution adopted. This has consequences for the structure of a program: specific cases are always stated before general cases, partial procedures are stated before overall procedures. Second, when Prolog tries to

resatisfy a goal, because a goal fails at a certain point or because you ask for another solution, it does this by backtracking. Backtracking consists of undoing the last decision and trying to find another fitting one, if this fails then the decision before the last one is undone and so on. There is one way to influence this course of events: the use of a cut ("!"). The cut is a goal that always succeeds, but that fixes the choices upto that point: once a cut is passed, there is no way to resatisfy the current goal. For example, one can state that a predicate is either a basic predicate or a derived predicate:

```
predicate(X)   :-    basic(X).
predicate(X)   :-    derived(X).

basic(drink).
basic(eat).

derived(drink).
```

When Prolog has to find out if some X is a predicate, it will first try to find out if it is a basic predicate before considering the possibility that it is a derived predicate. If one should ask if drink is a predicate, Prolog would answer "yes" on the basis of the fact that drink is a basic predicate. Asking for another solution would give "yes", because drink is a derived predicate too. For eat there is only one solution, so the second answer would be "no". Inserting a cut in the first rule, as illustrated below, has the following effects:

```
predicate(X)   :-    basic(X), !.
```

Now this rule states: "If X is a basic predicate then X is a predicate and there is no need to look any further". So a second answer will always be "no". If X is not a basic predicate the cut is not passed and there is still the possibility that X is a derived predicate.

Program text

The full text of the Prolog program is presented below. The program was designed for translation of the underlying predications of the sample text. In fact it can handle all related predications with the same predicates. The program is a largely simplified reflection of the translation procedure:

1. The form of the predicate frames and predicate formation rules could be kept very simple because of the restricted number of predicates involved. The amount of lexical information needed relies heavily on the capability of the future parser. In the design of this program a number of problems have been left aside as to be handled by the parser. For example, the one-place predicate settle-down is supposed to be marked R by the parser. It appears impractical to incorporate another parsing procedure in the translation process: the parser should convey all the information gained in the parsing process explicitly into the predications.

2. At several points, the grammatical information supplied is restricted to the needs of the input predications. For example, the possibility of recipient- and beneficiary-subjects has been left out of consideration, as they do not appear in the sample text.

3. The complications with string manipulation in Prolog has caused some artificiality in the handling of term operators and derived predicates: where necessary, complex structures have been represented as lists.

In spite of these restrictions, the frame of the program can be said to have general validity. Improvements (extensions and refinements) will not concern the frame and will in most cases take place in the auxiliary, language-specific, modules.

```
/*                        the main 'body'                              */
/*              containing the translation procedures                  */


/*              1. analysis of the predication                         */
/*     transfer of the predication operator and conjunctions           */

/* coordinated predications are separated                              */

translation(E,F)        :-      E=[Pred1,Conj,Pred2],
                                F=[F_pred1,F_conj,F_pred2],
                                translation(Pred1,F_pred1),
                                equi(Conj,F_conj),
                                translation(Pred2,F_pred2).

/* the predication operator is separated from the 'subpredication' */

translation(E,F)        :-      E=[Op,Subpredication],
                                F=[Op,F_subpred],
                                predication_op(Op),
                                subtranslation(Subpredication,F_subpred).


/*              2. analysis of the subpredication                      */
/*           transfer of the predicate and its operator                */

/* first case: the subpredication consists of only a satellite         */

subtranslation(E,F)     :-      E=[S_term,Satf],
                                F=[F_sterm,Satf],
                                satfunction(Satf),!,
                                satelliettransl(S_term,F_sterm).

/* second case: a term functioning as predicate                        */
/* here, this kind of predicate takes only one zero(Subj) argument     */

subtranslation(E,F)     :-      E=[Op,Predicate,Term,Zero],
                                F=[F_op,F_pred,F_term,Zero],
                                zerofunction(Zero),
                                Predicate=[P_term,Sf],!,
                                aspectcheck(Op,F_op,zero),
                                F_pred=[F_pterm,Sf],
                                semfunction(Sf),
                                termtransl(P_term,F_pterm),
                                termtransl(Term,F_term).

/* third case: a one-place predicate with a "pure" equivalent          */

subtranslation(E,F)     :-      E=[Op,Predicate,Cat,Term,Sf],
                                F=[F_op,F_pred,F_cat,F_term,Sf],
                                equivalent(Predicate,F_pred),!,
                                aspectcheck(Op,F_op,F_pred),
                                categorycheck(F_pred,Cat,F_cat),
                                termtransl(Term,F_term),
                                sfunction(Sf).
```

```
/* fourth case: a one-place predicate whose meaning definition      */
/* will be translated                                               */

subtranslation(E,F)      :-    E=[Op,Predicate,Cat,Term,Sf],
                               Fl=[F_op,F_pred,F_cat,F_term,F_sf],
                               meaning(Predicate,Defpred),!,
                               equivalent(Defpred,F_pred),
                               aanp(F_pred,F_cat,F_sf),
                               aspectcheck(Op,F_op,F_pred),
                               termtransl(Term,F_term),
                               syntfunctioncheck(Fl,F).

/* fifth case: a predicate combined with two or more "arguments"    */
/* the predicate has a "pure" equivalent                            */

subtranslation(E,F)      :-    E=[Op,Predicate,Cat,Arguments],
                               Fl=[F_op,F_pred,F_cat,F_arguments],
                               equivalent(Predicate,Eqpred),!,
                               aspectcheck(Op,F_op,Eqpred),
                               categorycheck(Eqpred,Cat,F_cat),
                               argumentstransl(Arguments,Eqarguments),
                               corefcheck(Eqpred,Eqarguments,
                                          F_pred,F_arguments),
                               syntfunctioncheck(Fl,F).


/* sixth case: a predicate combined with two or more "arguments"    */
/* the predicate is translated by means of its meaning definition   */

subtranslation(E,F)      :-    E=[Op,Predicate,Cat,Arguments],
                               Fl=[F_op,F_pred,F_cat,F_arguments],
                               meaning(Predicatel,Defpred),
                               equivalent(Defpred,Eqpred),
                               aanp(Eqpred,F_cat,Arguments,Nwarg),
                               aspectcheck(Op,F_op,Eqpred),
                               argumentstransl(Nwarg,Eqarg),
                               corefcheck(Eqpred,Eqarg,F_pred,F_arguments),
                               syntfunctioncheck(Fl,F).


/*                3. analysis of the "arguments"                    */
/*          transfer of the semantic/syntactic functions            */

/* the list of "arguments" is split up into satellites and terms    */

argumentstransl([],[]).

argumentstransl(E,F)     :-    E=[Term,Sat|Rest],
                               F=[F_term,Sat|F_rest],
                               satfunction(Sat),!,
                               satelliettransl(Term,F_term),
                               argumentstransl(Rest,F_rest).
```

```
argumentstransl(E,F)     :-     E=[Term,Sf|Rest],
                                F=[F_term,F_sf|F_rest],
                                termtransl(Term,F_term),
                                check_obj(Sf,F_sf),
                                argumentstransl(Rest,F_rest).


/*                  4. analysis of (argument-)terms                    */
/*  transfer of term operators, referential indices, conjunctions      */

/* first case: the term consists of only a referential index           */

termtransl(T,T)          :-     T=[X],
                                ref(X),!.

/* second case: the term consists of a subsystem operator and a ref */

termtransl(T,T)          :-     T=[Op,X],
                                subsys(Op),
                                ref(X),!.

/* third case: coordinated terms are separated                         */

termtransl(E,F)          :-     E=[Term1,Coord,Term2],
                                F=[F_term1,F_coord,F_term2],
                                equi(Coord,F_coord),!,
                                termtransl(Term1,F_term1),
                                termtransl(Term2,F_term2).

/* fourth case: the term consists of a predication                     */

termtransl(E,F)          :-     E=[X,Predication,X],
                                F=[X,F_pred,X],
                                ref(X),
                                translation(Predication,F_pred),!.

/* fifth case: the term consists of a subpredication                   */

termtransl(E,F)          :-     E=[X,Subpred,X],
                                F=[X,F_sub,X],
                                ref(X),
                                subtranslation(Subpred,F_sub),!.

/* sixth case: a term with one restrictor                              */

termtransl(E,F)          :-     E=[Op,X,Restrictor],
                                F=[F_op,X,F_restr],
                                equitermop(Op,F_op),
                                ref(X),
                                restrictortransl(Restrictor,F_restr),!.
```

```
/* seventh case: a term with two or more restrictors                    */

termtransl(E,F)          :-    E=[Op,X,Restrictors],
                               F=[F_op,X,F_restr],
                               equitermop(Op,F_op),
                               ref(X),
                               restrictorstransl(Restrictors,F_restr).


/*                 5. analysis of satellite terms                       */
/*         transfer of adverbials and referential indices               */

/* first case: the term consists of only an adverbial                   */

satelliettransl(E,F)     :-    E=[Adv],
                               F=[F_adv],
                               equi_adv(Adv,F_adv),!.

/* second case: the term consists of a subpredication with or           */
/* without tense                                                        */

satelliettransl(E,F)     :-    E=[X,Restr,X],
                               F=[X,F_restr,X],
                               ref(X),
                               restrictortransl(Restr,F_restr),!.

/* third case: the satellite term is treated as an argument term        */

satelliettransl(E,F)     :-    termtransl(E,F).


/*                  6. analysis of the restrictors                      */

/* the list of restrictors is split up until it is empty                */

restrictorstransl([],[]).

restrictorstransl(E,F)   :-    E=[Restrictor|Rest],
                               F=[F_restr|F_rest],
                               restrictortransl(Restrictor,F_restr),
                               restrictorstransl(Rest,F_rest).


/*                 7.  analysis of the restrictor                       */
/*           the restrictor is treated as a subpredication              */

restrictortransl(E,F)    :-    subtranslation(E,F),!.


/* in case of failure the restrictor has to be supplied with a          */
/* dummy predicate operator                                             */
restrictortransl(E,F)    :-    E=[Predicate|Rest],
                               Ext=[dummy_op,Predicate|Rest],
                               subtranslation(Ext,F_ext),
                               F_ext=[dummy_op,F_pred|F_rest],
                               F=[F_pred|F_rest].
```

```
/*                          the definitions                          */
/*   containing information about the English grammatical system    */

predication_op(decl).
predication_op(int).
predication_op(decl_cert).

termop(dl).
termop(il).
termop(dm).
termop(im).
termop(d_).
termop(i_).

termop(X)          :-    subsys(X).

subsys(a).
subsys(h).
subsys(s).
subsys(q).
subsys(r).
subsys(prox).
subsys(nonprox).

ref(0)             :-    !.
ref(N)             :-    integer(N).

sfunction(X)       :-    subj(X).
sfunction(X)       :-    obj(X).
sfunction(X)       :-    semfunction(X).
sfunction(X)       :-    satfunction(X).

subj(X)            :-    subjass(Sem,X).

subjass(ag,agSubj).
subjass(fo,foSubj).
subjass(po,poSubj).
subjass(proc,procSubj).
subjass(zero,zeroSubj).
subjass(go,goSubj).

obj(goObj).
obj(recObj).
obj(benObj).

semfunction(ag).
semfunction(fo).
semfunction(po).
semfunction(proc).
semfunction(zero).
semfunction(go).
semfunction(rec).
semfunction(ben).
semfunction(dir).
semfunction(part).
semfunction(poss).
```

```
satfunction(loc).
satfunction(locfr).
satfunction(ti).
satfunction(circ).
satfunction(manner).
satfunction(so).
satfunction(rsn).
satfunction(purp).

zerofunction(zero).
zerofunction(zeroSubj).

gofunction(go).
gofunction(goObj).
gofunction(goSubj).

dyn(ag).
dyn(fo).
dyn(proc).

not_zero(Op)            :-    not(Op=dummy_op);
                              not(Op=0).
```

```
/*                    the syntactic checks                          */
/*    containing the procedures which correct the predications      */
/*                 according to the French grammar                  */

/*          1. check on syntactic function assignment               */

syntfunctioncheck(Ok,Ok)       :-    Ok=[Op,Pred,v,[Term1,Sfl,Term2,
                                                goSubj|Rest]],!.

syntfunctioncheck(Subpred,Ok) :-     Subpred=[Op,Pred,v,[Term1,Sfl,Term2,
                                                go|Rest]],
                                     subjass(Sfl,Subj),!,
                                     Ok=[Op,Pred,v,[Term1,Subj,Term2,
                                                goObj|Rest]].

syntfunctioncheck(Subpred,Ok) :-      Subpred=[Op,Pred,v,[Term1,Sfl,Term2,
                                                go|Rest]],
                                     subjass(Sem,Sfl),!,
                                     Ok=[Op,Pred,v,[Term1,Sfl,Term2,
                                                goObj|Rest]].

syntfunctioncheck(Subpred,Ok) :-     Subpred=[Op,Pred,v,[Term1,Sfl|Rest]],
                                     not_zero(Op),
                                     subjass(Sfl,Subj),!,
                                     Ok=[Op,Pred,v,[Term1,Subj|Rest]].

syntfunctioncheck(Subpred,Ok) :-     Subpred=[Op,Pred,C,Term1,Sfl],
                                     not-zero(Op),
                                     subjass(Sfl,Subj),!,
                                     Ok=[Op,Pred,C,Term1,Subj].
syntfunctioncheck(OK,OK).

check_obj(recObj,rec)    :-     !.

check_obj(benObj,ben)    :-     !.

check_obj(OK,OK).


/*          2. check on the use of aspect                           */

aspectcheck(presprogr,pres,X) :-     !.

aspectcheck(pret,pretimpf,zero):-    !.

aspectcheck(pret,pretperf,Pred):-    f_predicate(Pred,C,[First|Next]),
                                     dyn(First),!.

aspectcheck(pret,pretimpf,Pred):-    !.

aspectcheck(Ok,Ok,X).
```

```
/*          3. check on the category of nominal predicates          */

categorycheck(Pred,n,nf)        :-    f_predicate(Pred,nf,A),!.

categorycheck(Pred,n,nm)        :-    !.

categorycheck(Pred,Ok,Ok).

/*          4. check on coreference                                 */

corefcheck(P,A,Fp,Fa)           :-    check_on_id(A,Fa),!,
                                      Fp=[se,P].
corefcheck(P,A,P,A).

check_on_id(Arg,Argok)          :-    Arg=[One,Sf1,One,Go|Rest],!,
                                      gofunction(Go),
                                      Argok=[One,Sf1|Rest].

check_on_id(Arg,Argok)          :-    Arg=[One,Sf1,Two,Sf2,One,rec|Rest],!,
                                      Argok=[One,Sf1,Two,Sf2|Rest].

/*          5. structural adjustments with a paraphrase             */

aanp(Pred,C,Sf)                 :-    f_predicate(Pred,C,[Sf]).

aanp(Pred,C,Arg,Argok)          :-    f_predicate(Pred,C,[Sf]),!,
                                      Arg=[X,First|Rest],
                                      Argok=[X,Sf|Rest].

aanp(Pred,C,Arg,Argok)          :-    f_predicate(Pred,C,[Sf1,Sf2]),!,
                                      Arg=[T1,First,T2,Second|Rest],
                                      Argok=[T1,Sf1,T2,Sf2|Rest].

aanp(Pred,C,Arg,Argok)          :-    f_predicate(Pred,C,[Sf1,Sf2,Sf3]),
                                      Arg=[T1,One,T2,Two,T3,Three|Rest],
                                      Argok=[T1,Sf1,T2,Sf2,T3,Sf3|Rest].
```

```
/*                        The English fund                              */
/*     containing the English lexical information needed for the        */
/*                          translation                                 */

predicate(short,a,zero)              :-!.
predicate(tipple,v,[ag,go])          :-!.
predicate(drink,v,[ag,go])           :-!.
predicate(settle_down,v,[ag,go])     :-!.
predicate(odd,a,[zero])              :-!.


meaning(tipple,drink).
meaning(sorry,pity).
meaning(ashamed,shame).
meaning(bring_to_an_end,finish).

specpred(bring_to_an_end).

stem(tipple,tippl)                   :-!.
stem(Pred,Pred).

predicate(Derived,C,A)               :-    derived(M,Prim,[Derived,C,A]).

derived(int,Pred,[[very,Pred],a,A]) :-    predicate(Pred,a,A).

derived(nom_ag,Pred,[[Stem,er],n,[zero]]):-
                                          predicate(Pred,v,[ag|Rest]),
                                          stem(Pred,Stem).

derived(refl,Pred,[[Pred,r],v,[Red]]):-   predicate(Pred,v,[First,
                                                    Second|Rest]),
                                          Red=[First|Rest].

derived(spec,Pred,[[Pred,his,speech],v,[ag,go]]):-
                                          specpred(Pred).
```

```
/*                      The French fund                                  */
/*   containing the predicate frames and predicate formation rules */
/*                  of the French predicates                             */

f_predicate(habiter,v,[po,go])              :-!.
f_predicate(boire,v,[ag,go])                :-!.
f_stem(boire,buv)                           :-!.
f_predicate(planete,nf,[zero])              :-!.
f_predicate(visite,nf,[zero])               :-!.
f_predicate(court,a,[zero])                 :-!.
f_predicate(plonger,v,[fo,go,dir])          :-!.
f_predicate(prince,nm,[zero])               :-!.
f_predicate(petit,a,[zero])                 :-!.
f_predicate(abattement,nm,[zero])           :-!.
f_predicate(grand,a,[zero])                 :-!.
f_predicate(dire,v,[ag,go,rec])             :-!.
f_predicate(faire,v,[ag,go])                :-!.
f_predicate(trouver,v,[proc,go])            :-!.
f_predicate(installer,v,[ag,go])            :-!.
f_predicate(silence,nm,[zero])              :-!.
f_predicate(collection,nf,[zero,part])      :-!.
f_predicate(bouteille,nf,[zero])            :-!.
f_predicate(vide,a,[zero])                  :-!.
f_predicate(plein,a,[zero])                 :-!.
f_predicate(repondre,v,[ag,go,rec])         :-!.
f_predicate(air,nm,[zero])                  :-!.
f_predicate(lugubre,a,[zero])               :-!.
f_predicate(demander,v,[ag,go,rec])         :-!.
f_predicate(oublier,v,[proc,go])            :-!.
f_predicate([se,enquerir],v,[ag,go,go])     :-!.
f_predicate(plaindre,v,[zero,go])           :-!.
f_predicate(avouer,v,[ag,go])               :-!.
f_predicate(honte,nf,[zero])                :-!.
f_predicate(baisser,v,[ag,go])              :-!.
f_predicate(tete,nf,[zero])                 :-!.
f_predicate(insister,v,[ag,go])             :-!.
f_predicate(desirer,v,[po,go])              :-!.
f_predicate(secourir,v,[ag,go])             :-!.
f_predicate(achever,v,[ag,go])              :-!.
f_predicate(discours,nm,[zero])             :-!.
f_predicate(enfermer,v,[ag,go])             :-!.
f_predicate(impenetrable,a,[zero])          :-!.
f_predicate(s_en_aller,v,[ag])              :-!.
f_predicate(perplexe,a,[zero])              :-!.
f_predicate(bizarre,a,[zero])               :-!.
f_predicate(grande_personne,nf,[zero])      :-!.
f_predicate(continuer,v,[ag])               :-!.
f_predicate(voyage,nm,[zero])               :-!.

f_predicate(Derived,C,A)            :-   f_derived(M,Prim,[Derived,
                                                          C,A]).
```

```
f_derived(int,Pred,[[tres,Pred],a,A]):-    f_predicate(Pred,a,A).

f_derived(nom_ag,Pred,[[Stem,eur],n,[zero]]):-
                                           f_predicate(Pred,v,[ag|Rest]),
                                           f_stem(Pred,Stem).


f_derived(refl,Pred,[[se,Pred],v,Red]):-
                                f_predicate(Pred,v,[First,Second|Rest]),
                                Red=[First|Rest].

f_derived(spec,Pred,[[Pred,son,discours],v,[ag,go]]):-
                                f_specpred(Pred).

f_specpred(achever).
```

```
/*                        The equivalences                              */
/*     containing all lexical equivalences & rules for handling         */
/*                        derived predicates                            */

equivalent(inhabit,habiter)          :-!.
equivalent(drink,boire)              :-!.
equivalent(planet,planete)           :-!.
equivalent(visit,visite)             :-!.
equivalent(short,court)              :-!.
equivalent(plunge,plonger)           :-!.
equivalent(prince,prince)            :-!.
equivalent(little,petit)             :-!.
equivalent(dejection,abattement)     :-!.
equivalent(deep,grand)               :-!.
equivalent(say,dire)                 :-!.
equivalent(do,faire)                 :-!.
equivalent(find,trouver)             :-!.
equivalent(settle_down,installer)    :-!.
equivalent(silence,silence)          :-!.
equivalent(collection,collection)    :-!.
equivalent(bottle,bouteille)         :-!.
equivalent(empty,vide)               :-!.
equivalent(full,plein)               :-!.
equivalent(reply,repondre)           :-!.
equivalent(air,air)                  :-!.
equivalent(lugubrious,lugubre)       :-!.
equivalent(demand,demander)          :-!.
equivalent(forget,oublier)           :-!.
equivalent(inquire,[se,enquerir])    :-!.
equivalent(pity,plaindre)            :-!.
equivalent(confess,avouer)           :-!.
equivalent(shame,honte)              :-!.
equivalent(hang,baisser)             :-!.
equivalent(head,tete)                :-!.
equivalent(insist,insister)          :-!.
equivalent(want,desirer)             :-!.
equivalent(help,secourir)            :-!.
equivalent(finish,achever)           :-!.
equivalent(shut_up,enfermer)         :-!.
equivalent(impregnable,impenetrable):-!.
equivalent(go_away,s_en_aller)       :-!.
equivalent(puzzled,perplexe)         :-!.
equivalent(odd,bizarre)              :-!.
equivalent(grown_up,grande_personne):-!.
equivalent(continue,continuer)       :-!.
equivalent(journey,voyage)           :-!.


equivalent(E_pred,F_pred)      :-    derived(M,E_prim,[E_pred|_]),!,
                                     weak_equivalent(E_prim,F_prim),
                                     f_derived(M,F_prim,[F_pred|_]).

weak_equivalent(E,F)           :-    meaning(E,Edef),!,
                                     equivalent(Edef,F).

weak_equivalent(E,F)           :-    equivalent(E,F).
```

```
equi(and,et).
equi(but,mais).

equi_adv(already,deja).

equitermop([Op,Ord],[Op,F_ord])      :-    termop(Op),
                                           ordinaltransl(Ord,F_ord),!.

equitermop(Op,Op).

ordinaltransl(next,suivant).
```

The input and output predications

For each sentence of the sample text the underlying English predication is transformed into a list, according to the syntax of predications as specified in the program. The twelve resulting lists served as input for the translation program. With these input predications the French output predications are given, as they were delivered by the program.

```
1.
I:[decl,
   [pret,inhabit,v,
         [[il,1,[[tippl,er],n,[1],zero]],po,
          [[dl,next],2,[planet,n,[2],zero]],goSubj]]]
O:[decl,
   [pretimpf,habiter,v,
         [[il,1,[[buv,eur],nm,[1],zero]],po,
          [[dl,suivant],2,[planete,nf,[2],zero]],goSubj]]]

2.
I:[[decl,
   [pret,[[il,3,[[visit,n,[3],zero],[[very,short],a,[3],zero]]],zero],
         [prox,4],zeroSubj]],
but,
[decl,
   [pret,plunge,v,
         [[a,4],foSubj,
          [dl,5,[[prince,n,[5],zero],[little,a,[5],zero]]],goObj,
          [i_,6,[[dejection,n,[6],zero],[deep,a,[6],zero]]],dir]]]]
O:[[decl,
  [pretimpf,[[il,3,[[visite,nf,[3],zero],[[tres,court],a,[3],zero]]],zero],
                  [prox,4],zeroSubj]],
mais,
[decl,
   [pretperf,plonger,v,
         [[a,4],foSubj,
          [dl,5,[[prince,nm,[5],zero],[petit,a,[5],zero]]],goObj,
          [i_,6,[[abattement,nm,[6],zero],[grand,a,[6],zero]]],dir]]]]

3.
I:[decl,
   [pret,say,v,
    [[a,5],agSubj,
     [7,[int,
         [presprogr,do,v,
           [[h,1],agSubj,
            [q,8],goObj,
            [nonprox,9],loc]]],7],goObj,
     [dl,1,[[tippl,er],n,[1],zero],
         [pret,find,v,
           [[a,5],procSubj,
            [r,1],goObj,
            [10,[partperf,[settle_down,r],v,
                  [[5],ag,
                   [11,[silence,n,[11],zero],11],circ,
                   [[il,12,[collection,n,
                             [[12],zero,
                              [im,13,[[bottle,n,[13],zero],
                                      [empty,a,[13],zero]]],part]]],
                and,[il,14,[collection,n,
                             [[14],zero,
                              [im,15,[[bottle,n,[15],zero],
                                      [full,a,[15],zero]]],part]]]],
                                              locfr]],10],circ]]]],rec]]].
```

```
O:[decl,
   [pretperf,dire,v,
    [[a,5],agSubj,
     [7,[int,
         [pres,faire,v,
            [[h,1],agSubj,
             [q,8],goObj,
             [nonprox,9],loc]]],7],goObj,
     [d1,1,[[buv,eur],nm,[1],zero],
            [pretperf,trouver,v,
              [[a,5],procSubj,
               [r,1],goObj,
               [10,[partperf,[se,installer],v,
                     [[5],agSubj,
                      [11,[silence,nm,[11],zero],11],circ,
                      [[i1,12,[collection,nf,
                                [[12],zero,
                                 [im,13,[[bouteille,nf,[13],zero],
                                         [vide,a,[13],zero]]],part]]],
                   et,[il,14,[collection,nf,
                                [[14],zero,
                                 [im,15,[[bouteille,nf,[15],zero],
                                        [plein,a,[15],zero]]],part]]]],
                                      locfr]],10],circ]]]],rec]]].

4.
I:[decl,
   [pret,reply,v,
     [[d1,1,[[tippl,er],n,[1],zero]],agSubj,
      [16,[decl,[presprogr,drink,v,[s,1],agSubj]],16],goObj,
      [i1,17,[[air,n,[17],zero],[lugubrious,a,[17],zero]]],manner]]]
O:[decl,
   [pretperf,repondre,v,
     [[d1,1,[[buv,eur],n,[1],zero]],agSubj,
      [16,[decl,[pres,boire,v,[s,1],agSubj]],16],goObj,
      [i1,17,[[air,nm,[17],zero],[lugubre,a,[17],zero]]],manner]]]

5.
I:[decl,
   [pret,demand,v,
     [[d1,5,[[prince,n,[5],zero],[little,a,[5],zero]]],agSubj,
      [18,[int,[presprogr,drink,v,
                [[h,1],agSubj,
                 [q,19],rsn]]],18],goObj]]]
O:[decl,
   [pretperf,demander,v,
     [[d1,5,[[prince,nm,[5],zero],[petit,a,[5],zero]]],agSubj,
      [18,[int,[pres,boire,v,
                [[h,1],agSubj,
                 [q,19],rsn]]],18],goObj]]]
```

```
6.
I:[decl,
   [pret,reply,v,
     [[dl,l,[[tippl,er],n,[l],zero]],agSubj,
      [20,[decl,[[2l,[poss,forget,v,[[s,l],procSubj,
                                    [0],go]],2l],purp]],20],goObj]]]
O:[decl,
   [pretperf,repondre,v,
     [[dl,l,[[buv,eur],nm,[l],zero]],agSubj,
      [20,[decl,[[2l,[poss,oublier,v,[[s,l],procSubj,
                                     [0],go]],2l],purp]],20],goObj]]]

7.
I:[decl,
   [pret,inquire,v,
     [[dl,5,[[prince,n,[5],zero],
             [little,a,[5],zero],
             [pret,sorry,a,[[r,5],agSubj,[a,l],ben,[already],ti]]]],agSubj,
      [22,[int,[0,forget,v,[[l],procSubj,[q,23],goObj]]],22],goObj]]]
O:[decl,
   [pretperf,[se,enquerir],v,
     [[dl,5,[[prince,nm,[5],zero],
             [petit,a,[5],zero],

[pretimpf,plaindre,v,[[r,5],zeroSubj,[a,l],goObj,[deja],ti]]],agSubj,
      [22,[int,[0,oublier,v,[[l],procSubj,[q,23],goObj]]],22,]]]

8.
I:[decl,
   [pret,confess,v,
     [[dl,l,[[tippl,er],n,[l],zero],agSubj,
      [24,[decl,
           [0,forget,v,
              [[l],procSubj,
               [25,[pres,ashamed,a,[l],zeroSubj],25],goObj]]],24],goObj,
      [26,[partpres,hang,v,
              [[l],ag,
               [dl,27,[[head,n,[27],zero],[[[l],poss],[27],zero]]],go]],
                                                              26]circ]]]
O:[decl,
   [pretperf,avouer,v,
     [[dl,l,[[buv,eur],n,[l],zero],agSubj,
      [24,[decl,
           [0,oublier,v,
              [[l],procSubj,
               [25,[pres,honte,nf,[l],zeroSubj],25],goObj]]],24],goObj,
      [26,[partpres,baisser,v,
              [[l],agSubj,
               [dl,27,[[tete,nf,[27],zero],[[[l],poss],[27],zero]]],
                                                    goObj]],26],circ]]]
```

```
9.
I:[decl,
   [pret,insist,v,
     [[dl,5,[[prince,n,[5],zero],
             [little,a,[5],zero],
             [pret,want,v,[[r,5],poSubj,

[28,[inf,help,v,[[5],ag,[a,1],go]],28],goObj]]]],agSubj,
     [29,[int,[0,ashamed,a,[[1],zero,[q,30],so]]],29],goObj]]]
O:[decl,
   [pretperf,insister,v,
     [[dl,5,[[prince,nm,[5],zero],
             [petit,a,[5],zero],
             [pretimpf,desirer,v,
                [[r,5],poSubj,
                 [28,[inf,secourir,v,[[5],agSubj,[a,1],goObj]],28],
                                                        goObj]]]],agSubj,
      [29,[int,[0,honte,nf,[[1],zero,[q,30],so]]],29],goObj]]]

10:
I:[[decl,
   [pret,[bring_to_an_end,his,speech],v,
     [[dl,1,[[tippl,er],n,[1],zero]],agSubj,
      [33,[decl,[0,ashamed,a,
                  [[1],zeroSubj,
                   [34,[drink,v,[1],ag],34],so]]],33],goObj]]],
and,
   [decl,
   [pret,shut_up,v,
     [[a,1],agSubj,
      [a,1],goObj,
      [il,35,[[silence,n,[35],zero],[impregnable,a,[35],zero]]],loc]]]]
O:[[decl,
   [pretperf,[achever,son,discours],v,
     [[dl,1,[[buv,eur],nm,[1],zero]],agSubj,
      [33,[decl,[0,honte,nf,
                  [[1],zero,
                   [34,[boire,v,[1],ag],34],so]]],33],goObj]]],
et,
   [decl,
   [pretperf,[se,enfermer],v,
     [[a,1],agSubj,
      [il,35,[[silence,nm,[35],zero],[impenetrable,a,[35],zero]]],loc]]]]

11.
I:[decl,
   [pret,go_away,v,
     [[dl,5,[[prince,n,[5],zero],[little,a,[5],zero]]],agSubj,
      [36,[puzzled,a,[5],zero],36],circ]]]
O:[decl,
   [pretperf,s_en_aller,v,
     [[dl,5,[[prince,nm,[5],zero],[petit,a,[5],zero]]],agSubj,
      [36,[perplexe,a,[5],zero],36],circ]]]
```

```
12:
I:[decl,
   [pret,say,v,
     [[a,5],agSubj,
      [37,[decl_cert,[pres,[very,[very,odd]],a,
                        [dm,38,[grown_up,n,[38],zero]],zeroSubj]],37],goObj,
      [a,5],rec,
      [39,[pret,continue,v,
            [[a,5],agSubj,
             [dl,40,[[journey,n,[40],zero],[[[5],poss],[40],zero]]],loc]],
                                                              39],circ]]]
O:[decl,
   [pretperf,[se,dire],v,
     [[a,5],agSubj,
      [37,[decl_cert,[pres,[tres,[tres,bizarre]],a,

[dm,38,[grande_personne,nf,[38],zero]],zeroSubj]],37],goObj,
      [39,[pretperf,continuer,v,
            [[a,5],agSubj,
             [dl,40,[[voyage,nm,[40],zero],[[[5],poss],[40],zero]]],loc]],
                                                              39],circ]]]
```

4. Conclusion

Mechanical translation from one natural language into another is an extremely complex matter. In this article a careful start has been made with a translation procedure in terms of Functional Grammar. In the course of the article, from theory to practice, the subject matter has become more and more restricted: starting from general ideas of how translation should cope with interlingual differences in form and/or meaning, ending in a computer program that translates twelve English predications into the corresponding French predications.

The underlying predications of Functional Grammar are considered to be a valid intermediary "language" for translation. The predications underlying translation equivalents are closer to each other than the linguistic expressions themselves; moreover, the underlying predications give a more or less uniform representation of the semantic information contained in the sentences. Predications of two different languages differ naturally in the lexical items they contain; the differences on the lexical level are handled by means of a set of equivalences, relating the corresponding predicates of the two languages and thus forming a kind of bilingual dictionary. The grammatical systems of two languages, which define the structure of the predications, may differ too, but the differences on this level are restricted. All grammatical operators and functions are part of a language independent set, of which each language uses a subset. For each particular language pair the non-correspondences, if any, have to be accounted for by means of rules checking the grammatical structure of the predication and adjusting it to the demands of the language's grammar.

These principles, together with the auxiliary machinery of equivalences and syntactic checks, have been put to the test of translation of a fragment of English into the corresponding fragment of French. The results of this test give cause to believe in the validity of the described model. The computer program embodying these results should be considered as an illustration of the fact that the translation procedure works. The program's design is very restricted and simplified, but it is suitable for refinement and extension. Apart from the topics for further research concerning the theory of Functional Grammar, which are indicated in the text, refinement and extension will be necessary in matters relating to the equivalences and syntactic checks.

Notes

[1] The proposed analysis needs justification. Why not separate the two clauses into two predications as in (1a&b)?

(1a) $(x_{33}$:[DECL {ashamed$_A$ $(x_1)_0$ $(x_{34}$:[drink$_V(x_1)_{Ag}](x_{34}))_{So}$}$](x_3))$

(1b) DECL {Pret [bring$_V$ $(d1x_{31}$:speech$_N(x_{31})_0$:{$(x_1)_{Poss}$}$(x_{31})_0)_{Go}$
$(i1x_{32}$:end$_N(x_{32})_0$ $(x_{31})_{Compl})_{Dir}$]
$(d1x_1$:tippler$_N(x_1)_0)_{AgSubj}$}

and

DECL {Pret shut up$_V$ $(Ax_1)_{AgSubj}$
$(Ax_1)_{GoObj}$
$(i1x_{35}$:silence$_N(x_{35})_0$:impregnable$_A(x_{35})_0)_{Loc}$}

Left aside the problems in determining the exact form of the predication (1a) (How to represent a stand-alone quote-argument?), there are a number of considerations that favour the analysis proposed in the text. There is a typographical indication that the two clauses may be considered to form one sentence: they are not separated by a period. There is an inconsequence here, since the second clause starts with a capital letter. But then again, why doesn't it start on a new line, as similar other sentences do? It seems that the translator has left the choice open, whether there are two sentences or one. The reason for this might be found in the original text (Saint-Exupery(1985)), which says:

(2) 'Honte de boire!' acheva le buveur qui s'enferma définitivement dans le silence.

So, the two clauses originate from one sentence. Finally consider the following examples, taken from two Dutch authors.

(3) 'Met die opgedirkte Amerikaanse trut zeker,' kwam ze nu toch wat meer naar mij toedraaien. (Dolf Cohen, Liegen, loog, gelogen, p.20. (De Bijenkorf bv; 1987))
(4a) 'Rustig nou maar', stelde ik haar gerust (...)
(4b) 'Dat zou ik wel kunnen doen', keek de heer Netjes op zijn gouden Rolex (...)
(Kees van Kooten, Modermismen, p. 94, 108. (De Bezige Bij; 1984))

[2] This assumption is a bit simplistic and, admittedly, influenced by knowledge of the original text. Nevertheless, it is useful in providing an illustration of the paraphrasing process.

References

Dik, S.C.

1978 Functional Grammar. Amsterdam: North Holland.
(Third printing, 1981, Dordrecht: Foris)

1979 Memorandum (How to arrive at a procedure for mechanical translation in terms of Functional Grammar). Ms. Institute for General Linguistics, University of Amsterdam.

1980 Studies in Functional Grammar. London & New York: Academic Press.

1985 Valentie en valentie-operaties in Funktionele Grammatika. In: TTT 5.2, pp. 95-114

Saint-Exupery, A. de

1985 Le petit prince. Bourges: Folio junior.

19?? The little prince. Translator: Woods, K. New York.