KOMBE

Communication Aids for the Handicapped

Franz Guenthner
Stefan Langer
Karin Krüger-Thielmann
CENTRUM FÜR INFORMATIONS- UND SPRACHVERARBEITUNG
MUNICH, GERMANY

Nathalie Richardet
Paul Sabatier
GROUPE INTELLIGENCE ARTIFICIELLE
UNIVERSITÉ DE MARSEILLE-LUMINY
MARSEILLE, FRANCE

Robert Pasero
PROLOGIA
MARSEILLE, FRANCE

1993/94

Table of contents

1. Introduction

1.1. Project facts and roles of the contractors

The European project KOMBE started on December 1st, 1991 and finished on June 30th, 1993. It was funded by the Commission of the European Community within the TIDE program (**T**echnology **I**nitiative for **D**isabled and **E**lderly People). The project was carried out by four contractors in Germany and in France. Two university institutes and two industrial partners were involved.

The academic partners were:

CIS

Centrum für Informations- und Sprachverarbeitung

Ludwig-Maximilians-Universität, Munich, Germany

The CIS was responsible for organization and coordination of the various contractors within the KOMBE project

In its work for the project, the CIS concentrated on the linguistic aspects of the German language fragments (Grammar: K. Krüger-Thielmann & S. Langer, Lexicon: A. Bohnhof & J. Koch) and the icon system (K. Krüger-Thielmann & R. Mercader). An important task concerned the requirements of the users. This was treated by R. Mercader in cooperation with all French and German project partners.

GIA

Groupe Intelligence Artificielle[1], CNRS

Faculté des Sciences de Luminy, Marseille, France

The GIA was responsible for the area of language generation, the implementation of the algorithms (Gérard Milhaud), and for the French language fragments (Grammar and Lexicon: R. Pasero, P. Sabatier & Nathalie Richardet).

The two industrial partners in the project were:

---

1 Now LIM (Laboratoire Informatique de Marseille).

PROLOGIA

Marseille, France

PrologIA concentrated on the development of the generic KOMBE system as well as on the communication between the different programming languages. PrologIA integrated the natural language components into the ALS speech aid and developed a graphical user interface. The symbol language system was developed by PrologIA.

SYSTEMA-4

Straubenhardt, Germany

SYSTEMA-4 had to develop interface software for ALS communication aid programs. In addition SYSTEMA-4 was responsible for the user aspects of the project.

1.2. Project summary

The KOMBE project aimed at developing speech aids for persons suffering from various forms of severe motor impairment (in particular, amyotrophic lateral sclerosis, ALS). We designed and implemented grammars and lexicons for the purpose of rapid composition of text. The resulting prototypes show that the system could be used for communication in several pragmatic settings. The project used techniques from computational linguistics and artificial intelligence for the development of new types of language generation devices. The underlying principles allow a considerable speed-up in communication rate on the one hand and a considerable increase in the ease of use on the other. We think that the techniques developed in this project are applicable to rehabilitative uses and that they thus provide useful prothesis for a variety of applications.

The project divided into three, closely intertwined, subprojects:

i) We carried out a thorough investigation of the features of existsing communication aids for patients suffering from amyotrophic lateral sclerosis (ALS) (e.g. "Equalizer", "ALS-KO", "Freedom of Speech" and others), and analyzed the merits and insufficiencies of these systems.

ii) On the basis of this research we constructed the KOMBE system. We designed and implemented grammars and lexicons for German and French. These natural language fragments bear in mind the conceptual and contextual knowledge for a variety of topics in the situation of dialogues between a doctor and his patients. The grammars make essential use of semantic and conceptual information thus allowing for the rapid and contextually appro-

priate composition of messages. Furthermore, we have developed algorithms for the composition of messages for the purpose of rapid composition of text. Two prototypes (a German and a French one) show the possibilities of ALS speech aids. The design and implementation of these enhancements to current speech aids was carried out in the context of techniques used in logic programming in particular with the programming language Prolog.

iii) We established general function-related criteria of evaluation, which can serve as basis for getting one step closer to the aim of normalizing aids for disabled and elderly people in the scope of operating communication and control systems. We produced auxiliary routines for communication aids and existing programs in order to achieve higher communication speed, to design a clearer and easier handling, to keep fatigue as low as possible and thus to reach a way of approach that need not be newly developed at high cost for the requirements of different user groups.

1.3. Project reports

The main results of this project have already been described by several deliverables and the Final Report. This report aims at describing the scientific results of the project in detail. Our work has also been presented in several publications, e. g. GUENTHNER et al. (1992, 1993, 1994). Partial aspects and underlying principles of the system have been described in RICHARDET (1992), LANGER (1993) and GODBERT/PASERO/SABATIER (1993).

2. General results

2.1. Background

Several diseases of the central nervous system (as, for example, ALS) entail increasing motor weekness. As a result, patients suffering from these diseases often have serious speech impairments, which lead to problems with communication. Both the motor weakness and the communication problems aggravate the living conditions of these mainly elderly people.

Starting form the requirements of persons touched by ALS, the goal of the project was to develop a communication aid for all people who are unable to speak and to use a normal keybord. For these users, the ALS KOMBE system allows rapid and contextually appropriate composition of natural language sentences.

While we were studying the possibilities of integrating the KOMBE system as a communicative device for different kinds of people unable to speak, we became aware of the great importance of pictographic languages for communication in the therapeutical and pedagogical domains. Several symbol languages exist which are used for that purpose. Since a lot of nonspeaking people are familiar with symbol languages of that kind, we conceived a second version of our system.

Like the ALS KOMBE system this second version of the communication aid uses the same principle of composition in order to allow the synthesis of syntactically and conceptually appropriate messages in the symbol language.

2.2. State of the art

This state of the art consists of three parts. In the first section we discuss existing methods for word prediction. In the second section we present some existing speech aids for the severely motor impaired and give an evaluation of their functioning. Finally we give a short survey of the state of the art in speech synthesis.

2.2.1. Word prediction

The aim of all word prediction systems is to reduce the input which is necessary to compose an utterance by predicting possible continuations of the sentence beginning. Most of these systems are uniquely based on statistics, considering tuples or triples of word forms in order to predict the following item. There are few systems which make use of some syntactic information, and, as far as we know, none of them is including conceptual knowledge.

PAL (Predictive Adaptive Lexicon) is one of the best-known word-prediction systems for English. A lot of tests have been done with this system, especially the determination of key-saving rates[2]. PAL is based on statistics and recency. The most interesting feature in this system is the fact, that the lexicon is being updated during a session, which means, that it is efficiently adapting to the users needs.

In the word prediction system for English presented in TYVAND/DEMASCO (1993) syntax statistics are used. Each word form is associated with an item of a set of about 130 syntax-tags. The possible continuations of an utterance are calculated by this system in the base of simple statistics and on the base of statistics on triples of the tags. The key saving rates obtained are not much better then with systems using pure statistics, which might be due to the use of randomly chosen syntax categories. No conceptual knowledge is used in this system.

All the systems based on statistics show the following disadvantages when used for communication aids for handicapped people:

- A large corpus from the specific domain would be necessary to have sufficient statistic material. But for the domains relevant for communication aids for disabled persons such a corpus doesn't exist.
- Thus, a lot of sessions are needed to teach the system the relevant utterances.
- There exist no thorough experiences for languages with an elaborated inflexion system such as German and French.

2.2.2. Existing communication aids for motor handicapped

Communication aids for disabled persons show the following main features:

- They minimize the number of actions necessary to compose utterances, in natural language as well as in symbol languages.
- They can be combined with different input devices (joystick, touch screen, mouse, single switch, etc.) adapted to different disabilities.

The option which has been adopted so far to speed up and to facilitate the composition of messages, is mainly the access to a large dictionary of words, symbols or icons. Making use of more or less sophisticated access functions,

---

[2] Key-saving is the percentage reduction in character selections required to encode a piece of text compared to a norm of one key-stroke per character (ARNOTT et al. 1993). Thus this rate is not very appropriate to the context of disabilities, as disabled don't use keyboard; the rate should rather be calculated by selection steps with a single switch.

the disabled user may select an element of the dictionary, or he/she might compose a word letter by letter (like in the system MAINATE).

More elaborated systems in this domain offer word prediction on the base of statistics. At each step of composition, the dictionary elements are sorted in order to offer the most probable elements (like in the system Cowriter).

Few systems make use of some restrained syntactic knowledge, verifying the agreement in number for determiners, nouns, adjectives and verbs (Rapitex and Cowriter).

ALS-KO (from the company SYSTEMA-4, which was involved in this project) is specially tuned to the needs of ALS patients. ALS-KO allows only a very restricted number of possible utterances, as the system possesses only a small number of phrases.

Another well-known communication system is Freedom of Speech, distributed by the company TechAid AG in Zug (Switzerland). Using this program, handicapped persons can enter sentences letter by letter or word by word; the system is delivered with a basic dictionary which can be extended by the user. It is possible to define one's own short forms for often used sentences.

The Canadian firm Madenta offers the word processing system Telepathic® for the Macintosh, which is said to apply some artificial intelligence principles, but which is, following our evaluation of its functioning, purely based on statistics. In combination with other software of the same firm it allows the composition of utterances for the motor impaired. The system includes a speech synthesizer.

The English system Cowriter applies some syntactical knowledge, but only in a very reduced way. It is making use of agreement between different categories (e.g. subject-verb) in order to produce proper continuations.

In France ALS patients can use the system MAINATE (distributed by Aurrand-Lions Informatique, Marseille) which is similar to FOS and Equalizer (latter is the English source of both programs). The system is equipped with a lexicon which can be accessed by selecting the first letter. The user can enter his own words to the dictionary; a French ALS patient has provided us her Mainate lexicon of about 2500 entries. Easy to handle it is nevertheless not very satisfying if we consider the speed of utterance composition.

A complete comparison of the KOMBE system with existing systems like Freedom of speech or MAINATE is not yet possible, because the KOMBE system doesn't dispose of a scanning mode so far. Nevertheless, a theoretic comparison is very interesting; we will calculate the number of actions to be performed by the user when composing the French sentence *je ne peux pas avaler des fruits.*

For this example we assume the dictionary of the ALS patient mentioned above. Let us also suppose that the user already has composed the sentence beginning *je ne peux pas.* Using MAINATE, the user can access the next word *avaler*, which is a word of the dictionary, by entering the letter "a" in the lexical access zone (one action, three clicks); as her dictionary contains nearly 200 entries beginning with "a" (which can't be presented on one screen) she has to leaf through the screens until the screen containing *avaler* is reached (second action) and to select this word (third action, three clicks).

In contrast the number of actions to be performed when using the KOMBE system, is reduced. Making use of the syntactic knowledge the KOMBE system proposes clitic pronouns or infinite verb forms as words following *je ne peux pas.* If these words are presented in alphabetical order the list begins with the infinitives of "a". As the dictionary contains 48 infinitives beginning with "a" *avaler* wouldn't be among the words of the first screen if only syntactic rules are considered. The conceptual constraints, however, reduce this list to the infinitives expressing capacities of a person (because the sentence begins with *je ne peux pas*). This list contains 15 infinitives which can be presented on one screen; the user has only to perform one action. In the case of infinitives beginning with a letter different from "a" (such as *lever*) the user has to perform a further action in order to enter the first letter. He then gets the short list of all infinitives with "l" expressing a capacity and the clitic pronouns.

To summarize, the KOMBE system only requires a few actions (1 or 2) to select the next word whereras the user of MAINATE needs more actions (3) to choose the desired word (each action requiring one or more clicks). In the case of inflected forms the MAINATE user has to complete the root by adding the ending; often used forms can be entered to the dictionary. The KOMBE user gets the right forms automatically, he doesn't have to pay attention to roots and endings.

None of the existing systems presented above is really making use of syntactic and conceptual knowledge in order to predict possible continuations. The systems based on statistics show the disadvantages named in the previous sections.

2.2.3. Speech synthesis

On the market there exist a large number of different speech synthesis boards. Over the last years, a lot of improvements have been made in this domain, and there are systems showing sufficient qualities (e.g. the Swedish multilanguage system INFOVOX): The voice output is understandable and its quality is acceptable. Additionally, many systems offer a lot of possiblities to change parameters as speed, type of voice etc.

Nevertheless, there remains a lot of work to do, especially in the following domains:

- Female voices aren't very satisfactory so far. This is a problem for women using communication systems, as they don't like to use a male voice to pronounce their utterances.
- All parameters which depend on the sentence structure, such as intonation and pitch, cause a lot of trouble. In French the pronounciation of the ending *-ent* and the phenomenon of liaison is highly dependent on the syntactic role of a word in the sentence. No existing system has so far solved these problems in a satisfactory manner.

Unfortunately, the enhancement of an existing voice synthesis system wasn't foreseen in our project, although two of the partners (CIS and GIA) have employees working in this domain.

2.3. Basic principles of the project

2.3.1. Our approach: guided composition of messages

The major innovation of our system consists in the fact that it uses linguistic and conceptual knowledge of the given language (natural language or pictographic language) to facilitate composition of utterances.

The knowledge of syntactic rules allows the system to make word prediction on the basis of rules assuring the grammatical correctness of the resulting sentence. We know precisely by which kind of words (determiner, noun, adjective, verb etc.) a sentence beginning might be completed. The French sentence beginning *je peux avaler des ...* (*I can swallow ...*) might be completed, for example, by a common noun in the plural (*aliments, boisson, baignoires, professeurs*, etc.).

Using the conceptual knowledge, the system can make word predictions on the basis of rules and constraints which assure the correctness of the sense of the sentence. We know more or less precisely, which kind of words permit a meaningful continuation of a sentence beginning. The sentence beginning *je peux avaler des ...* might be completed by common nouns referring to objects which can be swallowed. Common nouns like *baignoires, professeur*s are excluded from this list.

The use of these types of knowledge is an additional aid to the composition of messages as they allow rigourous word prediction. The reduction of the initial list of words allows us to reach our main goal, which is to minimize the number of actions the user has to perform in order to select a new word.

2.3.2. Flexibility

The basic idea is to allow the user as many free formulations as possible, while still maintaining quick input possiblities. To this end the KOMBE system can be used in different modes, which can be characterised by increasing freedom of formulation but decreasing speed of input.

Full mode

This mode allows the full facilities of guided composition of sentences. The first word of a sentence can be chosen from a set of words which represent syntactically possible sentence beginnings. After the choice of the first word, the possiblities for the next word are already constrained by the syntactic rules and the conceptual constraints. If we begin a (German) sentence with *ich*, conjunctions like *und* and, more importantly, verb forms may follow. For this reason, the set of words to follow which are offered by the system is much smaller in this mode than in the following, less restricted modes. The user then selects one word of the possible continuations generated by the system. Conceptual constraints are taken into account in the following way. The continuation of *zieh mir bitte ...* , for example, is most probably to be found among the nominal phrases which designate clothes (such as *den Pullover*). The conceptual knowledge defines relations between verbs (as *an-ziehen*) and the domains of the possible complements (*Pullover* belongs to the domain *clothing*).

Restricted mode

The user can deconnect the conceptual constraints and use the guided composition only with regard to the grammatical aspect. The words proposed by the system fit with the syntactic rules. So only syntactically correct word forms are proposed, which also discharges the user from finding the right endings of a chosen word. In German and French, which have relatively big inflectional paradigms, this is a major advantage.

Free mode

If the user wants to use formulations that are not within the grammar fragment, this is possible, too. For this purpose we offer this third mode. Here the user can enter her/his sentence word for word. Also words which are unknown to the system can be used and have to be entered letter by letter. At the wish of the user the system could enter the new word into the lexicon; this task is done by the module *lexical acquisition* which is described below. It is clear that the user will require substantially longer to enter an utterance using

the free mode than with the techniques including syntactic and semantic knowledge.

With these modes of use, the KOMBE system is highly flexible. Users of different age and different education can profit from the speech aids for various purposes. In many communication situations, the method of guided composition is applicable, because it is a quick method supporting the natural, creative use of language. Finally, users who wish to use words unknown to the system (when they are editing texts, for example) can enter them individually.

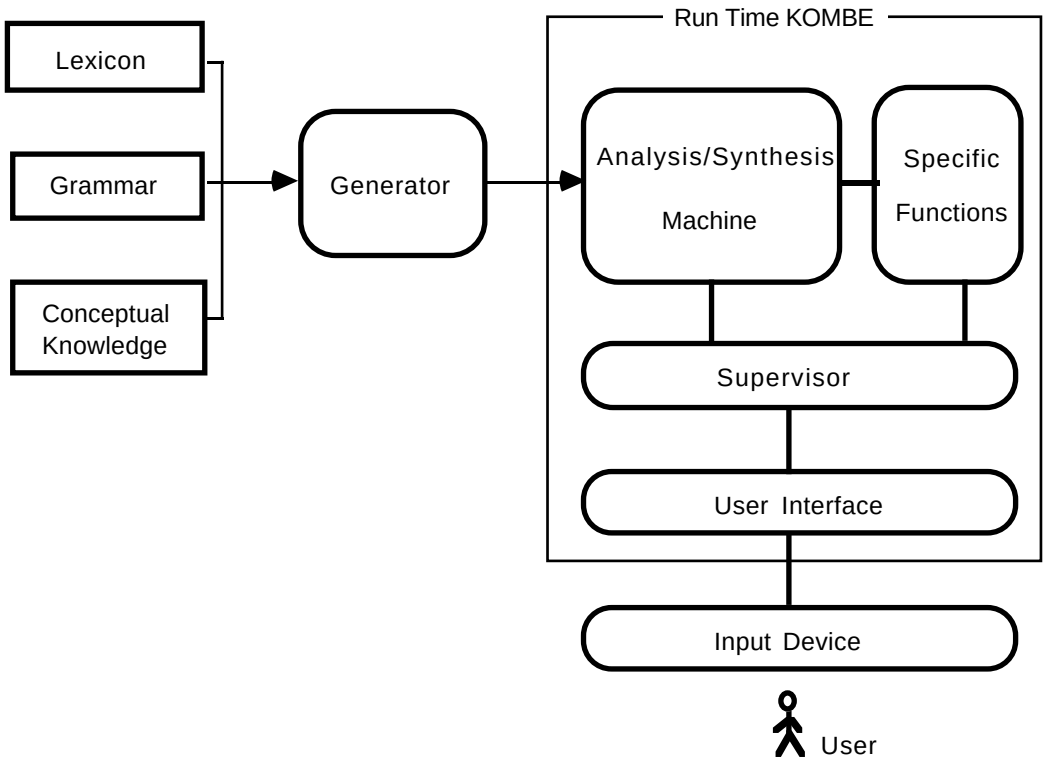
2.4. The KOMBE system

We implemented the above described method of guided composition of messages and constructed two prototypes, one in the framework of natural language communication (in a French and a German version), another suited to communication in a pictographic language. Our goal was to develop a communication aid for disabled people which offers additional features compared with existing products. Our system is making use of rules of syntactic and conceptual well-formedness for the used language, in order to make communication easier. These rules shouldn't be regarded as supplementary constraints for freedom of expression, but rather as a means for facilitating communication in an intelligent way.

The two prototypes have been implemented on the base of the following tools and modules:

- A generic tool, the generator, which allowed us to implement a communication aid making use of whatever language (natural language, symbol language or icon language). On the base of a lexicon, a formal description of the grammar and the conceptual constraints, this tool generates a parsing/generating device which contains the set of functionnalities a disabled person needs to communicate.
- Three modules defining various specific functions, a supervisor which manages the modalities of use, and an interface.

Each implemented system consists of four modules: the machine, the specific functions, the supervisor and the user interface. Different input/output devices can be linked with this system, following the kind of disability.



The generator comprises :

- a compiler for the different kinds of linguistic knowledge: lexicon, grammar (syntactic rules and semantic composition rules) and conceptual model; this knowledge has been formulated by means of formalisms which are described in chapter 4 of this report;
- algorithms for parsing and generation (a destailed description of these algorithms can be found in chapter 3);
- a set of general functions, independent of the language used. The general functions offer, among others, the possibility to change the conversation topic and to compose utterances on three different levels: using the set of words of the lexicon (free mode), using a set of words calculated on the base of the syntactic rules (restricted mode), using a set of words calculated on the base of syntactical rules and conceptual constraints (full mode); the description of the user interface comprises a specification of all these functions;[3]
- the specification of the conversation topic.

---

[3] The interface is described in a project paper: DAMESTOY, Isabelle & Robert PASERO: Définition de l'interface utilsateur de KOMBE – Definition der Benutzerschnittstelle für KOMBE (Interactions of the user with the KOMBE system), 1993.

On the base of this generator we implemented a prototype of a communication aid for natural language, in French and German, for persons suffering from ALS and other severe motor disabled people. In order to achieve this goal we (1) developed a lexicon, a grammar and a conceptual model for a subset of French and German, adapted at the doctor-patient communication.[4] We (2) defined a supervisor allowing to manage the set of functions and an interface, which allows to control the system with a mouse, and realized the interface for two operating systems (Macintosh and MS-DOS).

This user interface comprises specific functions offering searching aids for the words among the proposed words (search and selection by the user or search starting from the word beginning introduced by the user) as well as functions for message editing and file operations (copying or deleting a sentence in a file of typical sentences; inserting a sentence of the file of typical sentences in the editing window).

The use of this generator is a generic approach to develop communication aids which will facilitate enormously the work needed to develop further versions in other languages.

Starting from the same generator, we have also developed a prototype of a symbol language communication aid for persons suffering from cerebral palsy. We used the pictographic language SODI-GRACH. For this goal we performed the following tasks:

- development of a lexicon, a grammar and a conceptual model for a subset of the symbols;
- development of a set of specific functions like the ones used for the natural language systems;
- development of a supervisor and an interface resembling to the ones used for the natural language system.

Chapter 6 of this report gives a detailed description of this symbol language system.

---

4 These two natural language systems are described in detail in chapter 5.

3. The algorithms for the composition of messages

3.1. Outline of the algorithm

The different levels of knowledge are expressed in different modules and then translated into a Prolog program. The task of the algorithms is to bring together this knowledge in order to generate sentences to be used for rapid communication.

The presented algorithm is used to suggest possible continuations of a sentence or possible sentence beginnings to the user of the communication system. In order to reach this goal, the syntax analysis device has to be built in such a way that parts of sentences can be analyzed, and the grammatically and conceptually meaningful continuations can be determined.

To start composing a sentence, the algorithm has to find out all the possible beginnings of a sentence by looking at the right-hand corner of the rule `sentence` and its subrules. These rules are given in the grammar and the lexicon.

When the beginning of a sentence is already chosen, the algorithm determines all the meaningfull continuations by backtracking through all derivations of the given part. The words found by the algorithm are presented to the user, who then selects one of them. This word is then added to the existing part of the utterance. Now the algorithm restarts, finding out the possible continuations of the enlarged part of the sentence, which is thus composed gradually word by word. A sentence can either be accomplished by having no more meaningfull continuations or by the selection of a full stop, question mark or exclamation mark by the user. When one sentence is finished, the algorithm starts again from the beginning, suggesting all possible beginnings of a new sentence.

The following paragraphs contain a description of the technical details of the algorithm.

3.2. Technical description of the algorithm

The top call of the Prolog program produced by the translator is `sentence(D,R)`. From the axiom `sentence` of the grammar, it generates all the couples `(D,R)` where `D` is a derivation tree and `R` its associated semantic representation. Derivation trees are generated according to a top-down, depth-first, left-to-right and non-deterministic strategy. According to this method, a sentence `S` is lexically and syntactically well-formed if `S` is the list of leaves of a

derivation tree `D` generated by `sentence(D,R)`. We have the following Prolog program :

```
lexically_and_syntactically_well_formed(Sentence) :-
     list_of_leaves(Derivation_tree,Sentence),
     sentence(Derivation_tree,Semantic_representation).
```

This program is used both for analyzing and synthesizing sentences. In the analysis mode (i.e. when `sentence` is bound), for reasons of efficiency, as soon as a leaf of a derivation tree is generated, one must verify that this leaf is identical to the current word (or expression) to be analyzed in the sentence. If they are identical, the generation of the current derivation tree continues; otherwise backtracking automatically occurs in the process of the derivation tree generation. In order to do that, the call to the predicate `list_of_leaves` is placed before the call to `sentence`. As defined in [Giannesini et al. 1985], it is coroutined by using the built-in predicate `freeze`.

```
list_of_leaves(Derivation_tree,Sentence) :-
     freeze(Derivation_tree,leaf(Derivation_tree,[],Sentence)).
leaf(Current_leaf_of_derivation_tree,L,[Current_wd_of_sentence|L]):-
     atom(Leaf_of_derivation_tree),
     Current_leaf_of_derivation_tree = Current_word_of_sentence.
leaf(Root(Tree),L1,L2) :-
     freeze(Tree,leaf(Tree,L1,L2)).
leaf(Root(Tree1,Tree2),L1,L2) :-
     freeze(Tree1,leaf(Tree1,L3,L2)),
     freeze(Tree2,leaf(Tree2,L1,L3)).
leaf(Root(Tree1,Tree2,Tree3),L1,l2) ... etc.
```

The first clause `leaf` checks if unification is possible between the current leaf of the derivation tree and the current word of the sentence. If they can be unified, the process goes on, else backtracking automatically occurs in the generation of the derivation tree. This process runs both for analysis (`Current_word_of_sentence` is bound) and for synthesis (`Current_word_of_sentence` is free). In order to do partial synthesis in one pass, one must record the generated set of leaves as possible current words for the sentence. Current leaves must be recorded whenever a current word is unexpected (analysis) or absent (synthesis) in the sentence. In order to do that, we must modify the first clause of `leaf`. We now describe the algorithm in detail.

With each word the process associates an integer corresponding to its position in the sentence. The algorithm needs two counters `Rightmost` and `Current`. The value of `Rightmost` is the integer associated to the rightmost expected word in the sentence. `Rightmost` increases according to the words accepted, but never decreases: backtracking in the generation of the derivation tree has no effect on it. The value of `Current` is the integer associated to the current word

of the sentence. It increases and decreases according to the evolution of the derivation tree generation. The first clause `leaf` is now as follows :

```
leaf(Current_leaf_of_deriv_tree,L,[Current_word_of_sentence|L]) :-
     atom(Leaf_of_deriv_tree),
     test(Current_leaf_of_deriv_tree,Current_word_of_sentence).
```

The `check` procedure is based on the following algorithm :

```
if Current < Rightmost
          if Current_leaf_of_derivation_tree=Current_word_of_sentence
                then continue
          else backtracking

if Current = Rightmost
          if Current_word_of_sentence is free
                then record Current_leaf_of_derivation_tree
                     backtracking
     if Current_word_of_sentence is bound
                then if Current_leaf_of_derivation_tree=
                            Current_word_of_sentence
                     then  erase the recorded leaves
                           Rightmost := Rightmost + 1
                           continue
                     else  record Current_leaf_of_derivation_tree
                           backtracking
```

In order to verify (in analysis mode) or to specify (in synthesis mode) that a sentence `S` is conceptually well-formed, one coroutines an initial constraint `conceptually_well_formed` on the semantic representation `R` associated to `S`. This condition is a call to the rules specifying the conceptual model related to the application domain of the system. The correctness of a semantic representation `R` (associated with a sentence `S`) is verified during its composition.

Finally the core Prolog program ensuring that a sentence is well-formed is the following:

```
well_formed(Sentence) :-
     conceptually_well_formed(Semantic_representation),
     list_of_leaves(Derivation_tree,Sentence),
     sentence(Derivation_tree,Semantic_representation).
```

Here the two last calls `list_of_leaves` and `sentence` express the constraint `lexically_and_syntactically_well_formed` defined above.

4. The formalism for linguistic, conceptual and contextual knowledge

4.1. Introduction

In order to describe natural language utterances used in a communication situation we have to consider various types of knowledge:

- lexical knowledge;
- knowledge of syntax;
- knowledge of semantics;
- conceptual knowledge.

All these types of knowledge are expressed by formalisms described in the following. The files which contain the different forms of information form a data basis, which will be transformed by the compiler into the analysis/synthesis program written in PROLOG.

We decided to base the project on logic programming for several reasons. One of these is the reversability of PROLOG programs, especially the possibility of computing possible continuations of a given beginning of a sentence in an efficient way.

We will illustrate the formalisms associated with the different types of knowledge after the following scheme:

- First we present a formal discription in order to describe the corresponding type of knowledge, the associated problems and the theoretical background of the formalism;
- Then we give a more detailed presentation of the formalism, taking into consideration aspects of its implementation;
- Finally we give a short but complete example, which presents all the files necessary for the KOMBE interface.

4.2. Description of the formalism

4.2.1. Lexical knowledge

A lexicon is a set of elements called *lexical items*. Each lexical item is a triplet of the form:

```
<e,c,g>
```

where

- e is the lexical entry;
- c is the lexical category;
- g is a set of grammatical features.

A *lexical category* (or *grammatical category)* designates a class of words or expressions as, for example, *common noun, article* or *verb*. It represents words or expressions sharing a type of properties (called *features*). Each feature is a attribute-value pair:

```
attribute : value
```

where *attribute* designates the name of the features; *value* is a finite set of possible values. The following features can, for example, be associated with the categories *article, common noun* and *verb* in French:

```
article
[ type : {definite,indefinite},
  gender: {masculine,feminine},
  number: {singular,plural} ]

common noun
[ gender: {masculine,feminine},
  number {singular,plural} ]

verb
[ infinitive: {...} ,
   auxiliary: {avoir,être},
  person: {1,2,3},
  number: {singular,plural},
  mood: {indicative,subjonctive,imperative,...},
  tense: {present,past,future,...},
  construction: {intransitive,transitive, ...} ]
```

Information given in publications on a certain language (dictionaries, grammars etc.) can be seen as a complete set of features associated with each lexical category. We retain only those features sufficient and necessary for our application.

The lexicon will give a category and the values of the features belonging to each lexical entry. For the words *un* and *homme,* for example, it will contain the following information:

```
un                          (lexical entry)
article              (lexical category)
[ type : indefinite ,       (grammatical feature)
  gender : masculine ,      (grammatical feature)
  number : singular] (grammatical feature)

homme
```

```
common noun
[ gender : masculine ,
  number : singular]
```

All lexical items can be defined explicitly. Certain lexical items can also be formed by morphological derivation rules operating on basic lexical items. The plural form *hommes*, for example, can be obtained (or derived) on the basis of the item *homme* by means of morphological derivation rules associated with the category *common noun*. The number feature of the derived item takes then as its value *plural*, all the other features are preserved. By this approach German adjectives as, for example *dankbar*, can be defined as composed of the verb root *dank* and the suffix *bar*; *undankbar* can then be derived by adding the prefix *un*.

It is evident that these morphological derivation rules must consider the exceptions of the given language(s) as it can be seen in the following examples:

(masculine / feminine in French)

actif / active	*ambigu / ambiguë*
époux / épouse	*veuf / veuve*
chat / chatte	*berger / bergère*
blanc / blanche	*neveu / nièce*

(singular / plural in French)

oeil / yeux	*bateau / bateaux*
évantail / évantails	*travail / travaux*
nez / nez	*prix / prix*

(singular / plural in German)

Saal(Säle)	*Aale/Aale*
Turm/Türme	*Wurm/Würmer*

A morphological treatment as described here is suited for French verbs and the "open" word classes (adjectives, nouns and verbs) in German because of the great number of different forms of the same lexical item. Each form is obtained on the basis of the root, the grammatical features (gender, number, tense, mood, person, etc.) and a classification determining the morphological rules to be applied. We implemented a morphological component for French verbs which is used during the analysis/synthesis of sentences. The German lexicon actually contains full forms; when implementing the lexicon, however, we had only to type a small number of forms and obtained the others by

means of the morphological component developed at the CIS[5]; this PROLOG program was also integrated into the user interface which allows the user to introduce new lexical items. We intend to integrate a morphological treatment of adjectives, nouns and verbs into the analysis/synthesis of German sentences corresponding to the treatment of verbs in the French system in order to reduce the number of items in the dictionary.

Furthermore, lexical knowledge concerns constraints of syntactic construction. These conditions can be formulated on the level of features associated with lexical categories. A well-known example of this kind of feature is the specification of the expected syntactic context of a verb. A transitive verb, *singen (to sing)*, for example, takes an object as its complement. This means that a transitive verb is constructed with a subject and an object complement (oc). A more declarative way to express this property is the following notation (where the "_" designates the verb):

```
construction: subject + _ + oc
```

If the verb can take several complements, further constraints must be specified and the formulation by means of features becomes more complex. The French verb *décomposer* and the German *schneiden* can be used in sentences such as:

Max décompose le problème.
Max décompose le problème en plusieurs sous-problèmes.
Katharina schneidet das Fleisch.
Katharina schneidet das Fleisch in kleine Stücke.

These verbs accept subject, object (oc) and a facultative prepositional complement (pc). If there is a pc, it must be in plural form and contain a certain proposition (*en*, *in*). We can note the construction feature as:

```
décomposer:
     construction:          sujet + _ + oc (+ pc)
          constraints:  pc: {prep=en, number=plural}

schneiden:
     construction:          sujet + _ + oc (+ pc)
          constraints:  pc: {prep=in, number=plural}
```

Lexical items can be noted in PROLOG rules as can be seen in the example for some French items:

```
article_indéfini(<<un>,<une>,<des>>) -> ;
article_défini(<<le>,<"l'">,<la>,<les>>) -> ;
nom_commun(<<homme>,<hommes>>,mas) ->;
```

---

5 by Petra Meier-Maier.

```
nom_commun(<<machine,"à",laver>,<machines,"à",laver>>,fém) -> ;
prép_de(<<de>,<"d'">)) -> ;
verbe(<<aimer>,<x>>,groupe_1,<t_emps,m_ode,p_ersonne,n_ombre>,co.nil)
     ->
                traitement_morphologique_verbe(aimer,x,groupe_1,
                t_emps,m_ode,p_ersonne,n_ombre);
verbe(<<donner>,<x>>,groupe_1,
                <t_emps,m_ode,p_ersonne,n_ombre>,co."à".nil) ->
                traitement_morphologique_verbe(aimer,x,groupe_1,
                t_emps,m_ode,p_ersonne,n_ombre);`
```

4.2.2. Syntactic knowledge

The syntactic knowledge is described in terms of a formalism directly derived from *Metamorphosis Grammars* (Colmerauer 1975). This formalism (called MG in the following) can (1) define the strings of a language as well as (2) associate any representations with the strings.

The MG formalism differs from classic formal grammars in allowing complex symbols, called *terms,* (and not only atomic or simple ones) in order to represent terminal and non-terminal symbols. Such a complex symbol can be:

- a variable;
- an atom;
- an expression of the form $k(t_1,...,t_n)$ where:
  - k is an atom of arity n ($n>0$); we say that k has n arguments;
  - t_i is a term, for all i ($1 \le i \le n$).

A variable can represent any unknown term. As a convention, small letters (possibly followed by an integer) are used to note a variable (x, y, x_1). Examples of terms are:

```
gn
     max
     gn(x)
     dét(masculin,singulier)
     phrase(x,y)
     phrase(sans_sujet,avec_objet)
     chat
     13
```

As in classic grammars, we note terminal symbols in brackets:

```
phrase  -->  gn(x)  [ne] verbe(x)  [pas] gn(y)
```

Within a rule, variables with the same name designate the same unknown term. In the above rule the variable x in `gn(x)` and `verbe(x)` represents the

same term. Two terms are said to be the "same" if they can be rewritten by one "common" term. Their rewriting is preceded by a unification operation. The following two terms are unifiable:

```
dét(g,n)
     dét(masculin,singulier)
```

The set of substitutions performed is:

```
{ g=masculin, n=singulier }
```

In contrast, the following terms can't be unified:

```
dét(g,n,m)     and   dét(masculin,singulier)
donner(max,milou,luc)      and   donner(max,x,x)
```

Since a variable can be unified with an infinite number of terms, a non-terminal symbol such as `gn(x)` represents an infinite set of symbols. A symbol containing one or several variables is understood as a pattern of symbols, and the rule the symbol occurs within represents a pattern of rules. The GM rule

```
phrase -->  gn(x)  [ne]  verbe(x)  [pas]  gn(y)
```

designates the set of rules obtained by replacing the occurrences of the variable *x* by one term (whatever) and by replacing the occurrences of the variable y by a term (whatever).

An example of a GM is the following French grammar:

```
phrase    -->   gn(n) verbe(n)
gn(n)     -->   dét(g,n)    nc(g,n)
     gn(n)      -->   np(n)
dét(masculin,singulier)    -->   [un]
     dét(féminin,singulier)      -->   [une]
     dét(g,pluriel)   -->   [les]
nc(masculin,singulier)     -->   [garçon]
     nc(masculin,pluriel)  -->   [garçons]
     nc(féminin,singulier) -->   [fille]
     nc(féminin,pluriel)   -->   [filles]
np(singulier)  -->   [max]
     np(singulier)    -->   [marie]
verbe(singulier)     -->   [chante]
     verbe(pluriel)   -->   [chantent]
     verbe(singulier)      -->   [dort]
     verbe(pluriel)   -->   [dorment]
```

This example illustrates perfectly the method used in this simple case as well as for more complex grammars of French and German that aren't described in detail here.

The PROLOG program associated with the first three rules of this grammar is obtained by the compiler, and has the following form:

```
phrase(ph(x,y))      ->   gn(x,n)     verbe(y,n);
gn(gn(x,y),n)        ->   dét(x,g,n)  nc(y,g,n);
gn(gn(x),n)          ->   np(x,n)
```

This program defines with its first argument the derivation tree corresponding to the above grammar. In conjunction with the program calculating the leaves of a tree, it can analyze and generate sentences.

4.2.3. Semantic knowledge

Knowledge at the semantic level is expressed by rules which – after being interpreted by the compiler – add semantic information to the analysis/synthesis PROLOG program. Equipped with this information, the program can gradually generate a semantic representation of a sentence *S* during the analysis or synthesis of *S*. The system will thus associate a semantic representation with each syntactically well-formed sentence.

In order to represent semantic knowledge we will use a formalism whose basic ideas are beyond the scope of the project. Only a short outline is included here.

The semantic description will follow two basic principles:

- it has to be compositional; that means that the semantic representation of an entity is obtained by composing the semantic representation of the components;
- terminal components of the semantic representation must be expressed in terms of relations operating on individuals.

The theoretical framework will be the λ-calculus, providing a safe and solid theoretical basis.

The basic principle is the following:

- Each syntactic category is linked to a semantic type; the type of a VP, for example, corresponds to the function which takes an individual to obtain a formula.

  The semantic types are:

  - either the basic types *individual (i)* and *formula (o)*;
  - or a complex type noted <*t*,*t'*> associated with the application whose argument is of the type *t* and whose result is of the type *t'* (where *t* and *t'* are types).

  The type of a VP is, for example, noted as <*i*,*o*>.

- Each grammar rule (**syntactic construction rule**) is linked to a **semantic composition rule**:

  syntax rule:

  ```
  phrase --> gn(n) gv(n)
  ```

  semantic composition rule:

  ```
  phrase = [gn gv]
  ```

  The types of *sentence*, *NP* and *VP* are, respectviely: *o,* <<*i*,*o*>,*o*> and <*i*,*o*>.

- Each **lexical element** (or terminal element) is linked to its semantic representation. The verb *chante* is, for example, linked to the following semantic representation of the type <*i*,*o*>*:*

  ```
  λx chante(x)
  ```

We will illustrate these methods by applying them to a (small) French grammar:

```
phrase --> gn verbe
gn --> dét(g) nc(g)
     gn --> np
dét(masculin)  --> [un]
     dét(féminin)    --> [une]
nc(masculin)   --> [garçon]
     nc(féminin      --> [fille]
np   --> [max]
np   --> [marie]
verbe --> [chante]
verbe --> [dort]
```

We associate the following semantic types with the different lexical categories:

```
phrase : o
gn : <<i,o>,o>
det : <<i,o>,<<i,o>,o>>
nc : <i,o>
np : i
verbe : <i,o>
```

Each grammar rules is related to a semantic composition rule:

```
phrase = [gn verbe]
gn = [dét nc]
     gn = λp [p np]
dét = λp λq exist (x,et[p  x],[q x]))
     dét = λp λq exist (x,et[p  x],[q x]))
nc =  λy garçon(y)
     nc =  λy fille(y)
```

```
np = max
     np = marie
verbe =λy chante(y)
verbe =λy dort(y)
```

Consider, for example, the sentence:

max chante

Its semantic representation, `chante(max)`, is obtained by application of the semantic representation of the nominal group, `(λp [p max])`, to the semantic representation of the verbe:

```
verbe (λx chante(x)):
[(λp [p max])  (λx chante(x))] = [(λx chante(x)) max] = chante(max)
```

The semantic representation `exist(x,et(garçon(x),dort(x)))` of the sentence *un garçon dort* is obtained by applying the semantic representation of the gn, `(λq exist(x,et(garçon(x),[q x])))`, to the semantic representation of the verb `(λy dort(y))`:

```
[(λq exist(x,et(garçon(x),[q x]))) (λy dort(y))]
                    =  exist(x,et(garçon(x),[(λy dort(y)) x))
                    =  exist(x, et(garçon(x),dort(x)))
```

The semantic representation of the np `(λq exist(x,et(garçon(x),[q x])))` itsself is obtained by means of application of the semantic representation of the determiner, `(λp λq exist(x,et([p x],[q x])))`, to the semantic representation of the common noun, `(λy garçon(y))`:

```
[(λp λq exist(x,et([p x],[q x])))  (λy garçon(y))]
     =  λq exist(x,et([(λy garçon(y)) x],[q x]
     =  λq exist(x,et([garçon(x),[q x]
```

4.2.4. Conceptual knowledge

According to a general point of view, we assume that a sentence is correct if it is well-formed at three different levels: the lexical, syntactic and conceptual level. A sentence is lexically well-formed if all the words and expressions it contains belong to the lexicon. It is syntactically well-formed if its structure is described by the grammar. A sentence is conceptually well-formed if the relations and the objects it describes are compatible. A sentence as, for example, *La table marche* is conceptually incorrect if the conceptual model allows to prove that the domains associated with the relations *être_table* and *marche* are distinct.

The conceptual knowledge permits:

- constraining the logical representation; one says that the semantic representation is conceptually valid (or not);
- referring to supplementary information.

This knowledge is expressed by the so-called conceptual model. It consists of:

- a set of individuals $\{x_1, x_2, ...\}$;
- a set of relational symbols;
- a set of domains.

The domains are related to each other by inclusion or disjunction. These relations can be expressed by a hierarchical tree. In addition, the model contains a set of constraints:

- for each individual: the domain (or union of domains) it might occur in;
- for each relation: the domain (or union of domains) the arguments might occur in.

Consider the following example of a conceptual model:

```
/*Domains and their hierarchy*/
tout_individu = {animé, non_animé}
     animé = {humain,non_humain}
     humain = {homme,femme}
     non_humain = {chien,chat}
/*Individuals and their domain constraints*/
pierre(humain)
     marie(humain)
/*Relations and their domain constraints*/
voit(animé,tout_individu)
     table(non_animé)
     marche(animé)
     homme(humain)
```

The conceptual model permits us to check whether a set E of atomic formulas is conceptually valid. On the basis of the conceptual model above, it is possible to deduce that the following sets are conceptually valid:

```
{homme(x),marche(x)}
     {marche(pierre)}
```

whereas these are not:

```
{table(x),marche(x)}
     {table(pierre)}
```

4.3. The implemented formalisms

4.3.1. Introduction

In the following we give some precise information about the general form of the natural language component of the KOMBE system before describing each file in detail.

As it is presented here, the implemented formalisms will be used for the French and the German system. The following paragraphs describe the implemented formalism in general and don't describe a special grammar. For reasons of perspecuity we take all examples from the files containing the French grammar.

In writing a grammar, five obligatory files have to be defined, one file is optional. Four of the obligatory files contain the lexical, syntactic, semantic and conceptional knowledge; the fifth one constitutes an interface between syntax and lexicon. This interface allows to define syntax and lexicon independently in order to 1) improve the readability of both files 2) to accelerate the access to the lexicon 3) to allow for accessing an external dictionary (e.g. a lexical database, if this shows up to be usefull.

Finally, the optional file is a PROLOG-file containing additional conditions which are called from the lexicon and/or the syntax. As this additional conditions are optional, this file is optional, too.

In the following paragraphs more detailed information is given concerning the functioning and the syntax of these files. In order to make the presentation easily understandable we have labelled the files by fixed names.[6]

- *GrammaireDoc* for the syntax-file
- *AccesLexDoc* for the interface between syntax and lexicon
- *LexiqueDoc* for the lexicon-file
- *SemantiqueDoc* for the file containing the semantics
- *ConceptuelDoc* for the file containing the conceptuel knowledge
- *ConditionsPro* for the file containing the supplementary conditions

---

6 The KOMBE-system itself is less restrained. Its functioning doesn't depend on the filenames but on their type. So different files of the same type may exist in the same system allowing for testing of different versions during the development of the system.

4.3.2. Syntactic knowledge: the file GrammaireDoc

Before describing the syntax, we have to explain a common notion for the files *GrammaireDoc, AccesLexDoc* and *SemantiqueDoc:* the preterminal. It allows to define syntax and lexicon independently, as it is a non-terminal symbol introduced in the grammar instead of a terminal.

We make a difference between two types of terminals:

- **non variable terminals**: they correspond to a call of an entity of the intern or external lexicon (the link being described in *AccesLexDoc*). One or several terminal symbols in the lexicon are associated with this preterminal symbol, which means that it corresponds to a lexical class. The value of its arguments determines the extension of this class.
- **variable terminals**: In some cases the set of terminal symbols associated with a preterminal is too big to list all entities in the lexicon (or even infinite: e.g. numbers). For this case we have developped the concept of variable preterminals which are associated with an intension formulated in terms of PROLOG-constraints. The corresponding terminals are those verifying this constraints. This link between the preterminal and the constraint is formulated in *AccesLexDoc,* the constraint itself is written down in *ConditionsPro*.

In the grammar the preterminals are treated like nonterminals, being declared as preterminals in the second part of this file. It will be shown below how exactly this link is formulated when describing *AccesLexDoc*.

Now we present the syntax of the file *GrammaireDoc*:

```
règles;
<syntactic rules>
préterminaux_non_variables;
<list of non-variable preterminals>
préterminaux_variables;
<list of variable preterminals>
fin;
```

Elements in bold type are obligatory indications, which are necessary for the compiler.

i) first part: rules

The grammar of the system is defined by means of a formalism corresponding to the metamorphosis grammar. We define rules with a non-terminal head and a tail containing one or several non-terminals, preterminals or conditions.

All non-terminal terms figuring in a tail must appear at least in one other rule as head of the rule. The preterminals will allow us to have access to the terminals, that means, the leaves of the derivation-tree. In the same way, the non-terminals appearing as head of a rule must be contained in the tail of another rule in order to be accessible. Only one terminal symbol, called axiom of the grammar, figures only as head of one or several rules. The call of this predicate initiates the analysis or the synthesis of a sentence, which consists in constructing the corresponding syntactic derivation tree.

As described earlier, the preterminals linking grammar and lexicon are treated like nonterminal symbols. In the syntactic rules they figure only in the tail, where terminals are to be placed.

Terms of the kind condition are treated differently. They express additional conditions concerning one or several features of the non-terminals (or preterminals) of a rule, thus imposing syntacting constraints which guarantee the correctness of the sentence synthesized or analysed. We call them syntactic conditions. Let's give an example: Our French grammar contains sentences beginning with *Donnez...*, after which a common noun must be preceeded by a determiner. In calling an NP we impose this constraint on the feature which determines, whether the NP contains a determiner or not. Thus we assure that a NP without determiner can't appear in this position.

The terms of type condition have the syntax *condition(prolog_predicate)*, where *prolog_predicate* expresses the constraint defined in *ConditionsPro* (or is a system predicate like *dif/2*).

Examples of rules (French grammar):

```
phrase.
     groupe_nominal(suj.t).
     groupe_verbal(suj.t) .
     point_final;
(NP with a proper name)
groupe_nominal(t).
     preposition(t).
     nom_propre(t);
```

(incomplete sentence with a condition forcing the variable x not to be instantiated by *que*)

```
phrase_moins(p.t,x).
     condition(dif(x,que)).
     groupe_nominal_moins(p.t,p.t',suj).
     groupe_verbal(suj.t');
```

ii) second part: non-variable preterminals

The non-terminal symbols which are non-variable preterminals are declared in this part. So this part is simply a list of all non-variable preterminals with their arguments. The elements of this list are separated by a semicolon.

Examples of rules of the part non-variable preterminals:

```
nom_commun0(c.n_ombre.g_enre);
nom_commun1(c.n_ombre.g_enre,t);
verbe0(s_ujet.n_ombre.g_enre);
verbe1(s_ujet.n_ombre.g_enre,t);
```

iii) third part: variable preterminals

Same syntax as the proceeding part.

Exemples de règles de la rubrique preterminaux_variables:

```
nombre(n);
année(a);
```

4.3.3. Interface grammar-lexicon: the file AccesLexDoc

As described earlier, this file constitutes an interface between the grammar and the lexicon. Each non-variable preterminal is associated with the corresponding lexicon call, and each variable preterminal with the corresponding constraint.

As shown earlier, the use of an interface between grammar and lexicon results in a better readability of the grammar and in faster access to the lexicon. Let's give an example:

A non-variable preterminal corresponds to a class of terminals given as a whole in the lexicon. For a non-variable preterminal like `common noun` this class is going to be quite large, slowing down the speed of lexicon access, as a large number of lexicon-rules has to be considered. One possibility to avoid this is to reduce the number of rules to be considered by defining subsets of preterminals with less syntactic features in the following way: if a feature *f* of a non-variable preterminal *P* has *n* possible values, this non-variable preterminal is replaced by *n* non-variable preterminals having the same features as *P* except for *f*. The set of corresponding rules in the lexicon is thus divided in *n* subsets, which will speed up lexicon access considerably.

For example, in

```
nom_commun(genre,nombre)
```

we can eliminate the feature genre, thus getting two new non-variable preterminals:

```
nom_commun_mas(nombre)
nom_commun_fem(nombre)
```

If this was done in the grammar, each rule containing the concerned preterminals would have to be replaced by n rules, making the grammar much slower, but as this is done in the interface, we gain considerably in speed, as the number of rules needn't be augmented.

Now we present the syntax of the file AccesLexDoc:

```
préterminaux_non_variables;
<rules of the part non-variable preterminals>
préterminaux_variables;
<rules of the part variable preterminals>
fin;
```

The file consists of two parts. Elements in bold type are obligatory indications, which are necessary for the compiler.

i) first part: non-variable preterminals

The rules of this part are composed of four elements constructed as follows.

```
<first element,  [ second element .] third element> [ .forth element
    ];
```

`First element`

This element has the form:

```
name_of_the_non-variable preterminal(arg1...argn).variable
```

The rules of this file are called by the grammar-rules. Here we find the preterminals declared in the grammar with instantiated arguments if it is necessary to decide which terminal to choose. Each instantiation corresponds to a terminal.

Other arguments remain as free variables at this stage; they will be instantiated in the lexicon. Each instantiated value identifies one terminal symbol.

`Second element`

This optional part has the form condition*(<Prolog_pred>.{Prolog_pred})*. The condition terms here play a similar role as those in the grammar described above. They allow for setting constraints on the variables, thus reducing the number of terminals called by a rule. This permits a limitation of the number of preterminals declared in the grammar improving its readability and facilitating its development.

The condition terms in the interface allow to use only one preterminal in the grammar, and to apply the additional conditions in AccesLex.

`Third element`

This element has the form:

```
name_of_lexicon_rule[ (arg1', ... ,argm') ].
     <x1,...,xi,variable,xi+2,...xp>
```

The third element contains the predicate for the call of the corresponding terminal in the internal lexicon. It consists of the predicate for the lexicon call and of a n-tuple incoding the position of the terminal in the lexicon rule. This tuple must have the same length as the one in the corresponding lexicon rule.

`Forth element`

This element is optional. Its value is either `con` or `voc`. It expresses a phonological constraint on the terminal form following in the sentence: in French there are certain word-forms which can only preceed a word beginning with a consonant (like ***la***) or a vowel (like *l'*) (cf. the corresponding English phenomena with *a/an*).

Examples of rules in the part non-variable preterminals:

```
<nom_commun0(c.sin.mas).t,nom_commun0_mas.<t,_>>;
<nom_commun0(c.plu.mas).t,nom_commun0_mas.<_,t>>;
<nom_commun0(c.sin.fem).t,nom_commun0_fem.<t,_>>;
<nom_commun0(c.plu.fem).t,nom_commun0_fem.<_,t>>;

<verbe2(suj.sin.g,v1,v2).t,verbe2(v1,v2).<t,_>>;
<verbe2(suj.plu.g,v1,v2).t,verbe2(v1,v2).<_,t>>;
```

(with a condition term and a phonological constraint: rules for the definite article *le, l',la,les*)

```
<article_défini(c.mas.sin).t,
     condition(dif((c,de)).
     condition(dif((c,"à")).article_défini.<t,_,_,_>>. con;
<article_défini(c.g.sin).t,article_défini.<_,t,_,_>>.voc;
<article_défini(c.fem.sin).t,article_défini.<_,_,t,_>>.con;
<article_défini(c.g.plu).t,.condition(dif((c,de)).
     condition(dif((c,"à")).article_défini.<_,_,_,t>>;
```

ii) Second part: variable preterminals

This part consists of rules of the following form:

```
<first element, second element> ;
```

`First element`

It has the following form:

```
name_of_variable_pret(arg1, ..., argn). variable_libre
```

This element is identical to the first part of the rules non-variable preterminals described above. The free variable represents the terminal form.

`Second element`

It has the following form:

```
prolog_predicate(arg1', ..., argi', variable_libre, argi+2', ...,
     argp')
```

This is a PROLOG-predicate having a free variable as one argument. This predicate has to be defined in the file ConditionsPro. It defines the constraint imposed on a terminal, thus assuring that it is one of the terminal forms of a variable preterminal. This equals a definition of the intension of this set.

Examples of rules of the part `variable preterminals`:

```
<année(a).t ,format_année(a,t);
<nombre(n).t ,format_nombre(n,t);
```

4.3.4. Lexical knowledge: the file LexiqueDoc

This file contains the lexicon. It consists of two parts having the following form:

```
terminaux_non_variables;
<rules of the part non-variable preterminals>
terminaux_variables;
<rules of the part variable preterminals>
fin;
```

Elements in bold are necessary indications for the compiler.

i) first part: non-variable preterminals

These rules have the following form:

```
name_of_non_variable_pret(<term_form_1,.....,term_form_n>, seman-
     tic_terminals, phon_indication, [ , arg1, ...., argn] );
```

The rules of this part are called from the interface. Thus we can find here all the rules corresponding to the terminal calls defined in *AccesLexDoc.*

The first argument of the terminal in the n-tuple contains the terminal forms in the same order as supposed in *AccesLexDoc*. The second argument gives a phononogical information (*con* or *voc*), indicating if each of the terminals begins with a vowel or a consonant. The third argument is the "semantic terminal"; this means, it is the identifier which represents the terminal in the semantic representation associated with the preterminal (see paragraph 3.5).

The following arguments are variable in number. They represent other features necessary for determining the syntactic behaviour of the terminal form.

Examples of rules of the part non-variable preterminals:

```
nom_commun1_fem(<<femme>,<femmes>>,femme_de,con,de.n.g);
nom_commun0_mas(<<objet>,<objets>>,objet,voc);
verbe2(<<prend>,<prennent>>,prend,con,dir.n.g,"à".n'.g');
adjectif1(<<amoureux>,<amoureux>,<amoureuse>,<amoureuses>>,amoureux_d
     e,voc,de.n.g);
article_def(<<le>,<la>,<"l'">,<les>>,<>,con);
```

ii) Second part: variable preterminals

These rules have the following form:

```
name_of_variable_preterminal(corresponding_message);
```

At the first sight, a part containing variable preterminals seems inconsistent, as we have seen above that the set of corresponding terminals is infinite and can't be listed in the lexicon. But the entity in the lexicon is in fact the message associated with the preterminal, which will be shown in the window proposing the possible continuations. In this case, the user has to type a word-form as described by the message. It seems appropriate to list these messages in the lexicon, as they supply the information concerning a terminal.

Exemples of rules in the part variable preterminals:

```
année(" une année de 1903 à 1993...");
nombre(" un nombre : 0, 1, 2, ...");
```

4.3.5. Semantic knowledge: the file SemantiqueDoc

This file contains the rules which contain the information necessary for the analyzing-synthesizing device to produce simultaneously to the syntactic structure of a sentence its semantic representation. So each sentence can be associated with a semantic representation, which will be used to verify the semantic constraints. This implies that semantic constraints (e.g. the selection constraints of verbs) are not described in this file. They are formulated in the file *ConceptuelDoc* (see 3.5.).

We adopt the following general features of a semantic representation of a sentence and of its parts:

- it has to be compositional, which means, that the semantic representation of a non-terminal is the result of some composition of the elements of this non-terminal;

- the terminal elements of a semantic representation have to be expressed as relational terms taking as their arguments individuals;
- we use lambda-expressions in order to be able to map a semantic expression into another semantic expression, as described in section 4.2.

Each grammar-rule in GrammaireDoc defining a non-terminal is associated with a semantic composition rule in *SemantiqueDoc* that defines which operations we have to execute in order to map the semantic representation of the elements of the right-hand side into the semantic representation of the head of the rule, the two basic operations being the lambda-abstraction and the application.

To ensure the consistency of these operations, we verify if the lambda-expression representing the head is of the same type as declared in the part `type-declarations`.

In this file the type *individual* is encoded as `ii`, the type formula as `oo`, corresponding to a lambda-expression without abstraction. The types of all preterminals and non-terminals are expressed by means of combinations of these types of the form `<type1,type2>` standing for λ`type1(type2)` and of the empty type. The latter we associate with the preterminal symbols having no semantic role (like the verb *être*, some kinds of prepositions or the *pas* in the negation *ne...pas*).

The application of the lambda abstraction as we have implemented it follows the principles of the lambda-calculus described in section 4.2.

The last element to define is the semantic representation of the preterminals, being the base from which to start the construction of the semantic representation of a sentence. We will call their semantic representation *basic semantic representations*. The basic semantic representation of a preterminal is a lambda expression the type of which is defined in the part type-declarations.

As we don't have access to the terminals at the level of *SemantiqueDoc* we give for each preterminal a generic type declaration where the semantic terminal of the lexicon rule is represented by `nn`. When analyzing/ synthesizing a sentence this `nn` is replaced by the corresponding semantic terminal.

Example:

The basic semantic representation of `nom_commun1` is defined as `<x,<y,nn(x,y)>>`. In the lexicon we have `frère_de` as semantic terminal of the common noun *frère*. This identifier will be integrated into the lambda expression as basic semantic representation, thus having as its result:

```
<x,<y,frère_de(x,y)>>.
```

Finally, we also admit the use of functional elements in the rules for semantic composition. For example we use the connector `et` (and) in a lot of semantic rules. This connector is encoded as `et(P1,P2)`, `P1` and `P2` being lambda-expressions.

In one of our grammars, we use the following rule to describe a verb phrase with a present participle.

```
groupe_verbal_forme_participe(t1,n_eg).
     neg_ne_ou_vide(n_eg).
     verbe_0(participe,t1).
     neg_pas_ou_vide(n_eg);
```

(accepting phrases like: *(ne) marchant (pas)*).

The following semantic rule is associated with the grammar rule:

```
groupe_verbal_forme_participe.<p, <x, et ([p, x],
     [neg_ne_ou_vide, [verbe_0, x]])>>;
```

where *p* is of the type `<ii,oo>` et `x` of the type `ii`.

For the phrase *la femme marchant*, we obtain the following semantic representation:

```
<x, et (femme(x), marche( x))>
```

(The property `<x1,femme(x1)>` has instantiated `p`).

Let's now present the syntax of the file *SemantiqueDoc* in detail. It consists of four parts:

```
symboles_externes;
<rules of the part external symbols>
types;
<rules of the part types>
règles;
<rules of the part rules>
définitions;
<rules of the part definitions>
fin;
```

i) first part: external symbols

Here we list all the functional symbols used in the composition rules and in the basic semantic representations. We describe the type of each of their arguments and the type of the whole expression (constructed by means of the symbol) by rules of the following form:

```
name_of_the_functional_symbol(type_argument_1,...,type_argument_1) =
     type_of_the_furnctional_symbol;
```

Example of a rule in the part external symbols:

```
et(oo,oo) = oo;
```

ii) second part: types

Here the types associated with each preterminal or terminal in the grammar are declared. The declarations have the following form:

```
<name_of_preterminal_without_arg>=<associated_type>;
```

The order of these rules has not to be the same as in the syntax-file.

Examples of rules in the part types:

```
phrase = oo;
adjectif0 = <ii,oo>;
adjectif1 = <ii,<ii,oo>>;
nom_propre = <ii,oo>;
nom_commun0 = <ii,oo>;
nom_commun1 = <ii,<ii,oo>>;
groupe_nominal = <<ii,oo>,oo>;
neg_ne = <oo,oo>;
verbe0 = <ii,oo>;
verbe1 = <ii,<ii,oo>>;
verbe2 = <ii,<ii,<ii,oo>>>;
groupe_verbal = <ii,oo>;
relative = <ii,oo>;
attribut = <ii,oo>;
```

iii) Third part: rules

Here we give the semantic composition rule for each non-terminal element in the grammar.

The rules have the following form:

```
<name_of_the_terminal_without_arg> = <semantic_composition_rules>;
```

In this part, the rules have to appear in the same order as the rules in the grammar.

Normally all the semantic representations of the right hand side of the rule, except the empty one, are used to determine the semantic representation of the head of the rule. Nevertheless, there are some cases where a certain part of an expression is unimportant for the meaning of this expression as a whole (e.g. *Pierre est architecte* has the same meaning as *Pierre est un architecte*, the undefinite article making no contribution to the meaning).

The only operations used are the abstraction and the application, which we express as follows:

- `<x,A>` stands for λxA where `x` is a variable and `A` an expression lacking another expression of the same type as `x` to be a formula.
- `[A,B]` stands for the application of `A` on `B` as presented in part 1.

Examples of the part rules:

```
phrase = [ groupe_nominal, groupe_verbal ];

groupe_nominal = [prep_article,
     < x, [groupe_nominal, [ nom_commun1,x ] ] > ];

groupe_verbal = < x1, [ neg_ne, [ groupe_nominal1,  < x2,
     [ groupe_nominal2, < x3, [ [ [ verbe2,x1 ] ,x2 ],x3 ] > ]  > ] ]
     >;

attribut = <x, [ groupe_nominal, [ adjectif1, x] ] >;
```

iv) Forth part: definitions

Here we define the basic semantic representations of all preterminals to which a type has been attributed in the first part. The rules of this part have the following form:

```
<name_of_preterminal_without_arg> = <associated_RSB>;
```

The order of the definitions is unimportant.

Examples of rules in the part definitions:

```
nom_commun0 = < i , nn(i)>;
nom_commun1 = < i , < j , nn(i,j) > >;

verbe0 = < i , nn(i)>;
verbe1 = < i , < j , nn(i,j) > >;
verbe2 = < i , < j , < k , nn(i,j,k) > > >;

adjectif0 = < i , nn(i)>;
adjectif1 = < i, < j , nn(i,j) > >;

neg_ne_bis = < o , non(o) > ;
```

4.3.6. Conceptuel knowledge: the file ConceptuelDoc

This file contains the rules defining the conceptual knowledge. When the semantic representation of a phrase synthesized/ analyzed is being constructed, the program will verify at each step of the composition of a sentence if it fullfills the conceptual constraints.

Conceptual constraints express selection criteria of predicates, setting constraints on their arguments. The predicate *parler* (to speak), for example, has

as its first argument an entity signifying a human being (except for some speaking parrots).

Each entity in the semantic univers belongs to several classes. First it is defined in the most specific domain it belongs to. Additionally, it belongs to the classes containing the specific one, e.g. a *bee* belonging to the class *animals* and *animals* being a subclass of the class *animated beings*, a *bee* is certainly an *animated being*. In order to avoid describing all these classes for one entity, we first define a hierarchy of concepts specifiying which conceptual classes are subclasses of others and introducing all valid conceptual classes. So the definition of the argument of *parler* (speak) as *human being* allows introducing as argument of this predicate all entities belonging to the class human being or a subclass of *human being* (*man*, *woman* ...).

So the file *ConceptualDoc* consists of two parts, its general form being as follows:

```
domaines;
racine(<root_of_the_tree_of_conceptual_classes>);
<definition_of_subclasses>
contraintes;
<conceptual constraints>
fin ;
```

i) First part: conceptual hierarchy

The first element of this part defines the root of the tree of conceptual classes. This root can be any identifier.

The other rules define the subclasses of one conceptual class. They have the following form:

```
class.subordinated_classes;
```

where `subordinated_classes` is a list containing the classes and expressions of type `or(subordinated_classes1,subordinated_classes2)`, the latter expressing two different subclasses not excluding each other.

The order of the rules needn't be in accordance with the hierarchy.

Example for hierachy rules:

```
racine(tout);
tout.etre_animé.etre_inanimé;
etre_animé.plante.animal.etre_humain;
```

ii) Second part: conceptual constraints

The conceptual constraints have the following syntax:

```
name_of_predicate(arg1,....,arg_n);
```

The arguments have to be one of the concepts defined in the hierarchy above, the predicate must be one of the identifiers given as semantic terminals in the lexicon. The number of arguments has to correspond to the number of arguments of a predicate defined in *SemantiqueDoc.*

Examples for conceptual constraints:

```
parler(humain);
est_fille(femme);
aimer(etre_anime,tout);
```

4.3.7. The optional file ConditionsPro

Let's summarize the role of this file. It contains two different types of PROLOG-predicates:

- the predicates called from the grammar and the interface *AccesLexDoc* by means of introducing a condition term. These predicates express constraints on features of the symbols in these rules.
- the predicates called from the part variable preterminals of *AccesLexDoc*, which express definitions of intensions of the set of terminal forms associated with these preterminals.

4.4. A complete example

Although this example has been drawn from the complete grammar developed within the project, it will only show the significative idea of this grammar. The goal of this section is to describe the different files that are necessary for the functioning of the system on the basis of this very small example.

4.4.1. The file GrammaireDoc

This file describes the syntactic structures allowed in French. We have restrained ourselves the following incisive constraints:

- a sentence is composed of a noun phrase followed by a verb phrase;
- the subject is one of the personal pronouns *je, j'* or *nous;*
- a noun phrase consists of a (possibly empty) preposition followed by an (indefinite or definite) article and a common noun;
- a verb phrase consists of a verb (with 0 or 1 complements) possibly followed by a noun phrase.

This is the file *GrammaireDoc* :

```
/* Grammaire KOMBE */

regles;  /* indication pour le traducteur */

/*****************************************************************/
/* Phrase */

phrase.
     proposition.
     point;

/*****************************************************************/
/* Proposition */

proposition.
     groupe_nominal(sujet.t1).   /* sujet */
     groupe_verbal(sujet.t1);

/*****************************************************************/
/* Groupe nominal (GN) */

         /* je/j' : PronomPersonnelSujet */

groupe_nominal(sujet.t1).
     pronom_personnel_sujet(sujet.t1) ;

groupe_nominal(c.t1).      /* un(e)/des */
     condition(dif(c,sujet)).
     préposition_ou_vide(c.t1).
     article_indéfini(c.t1).
     nom_commun_0(c.t1) ;

groupe_nominal(c.t1).      /* le/la/l'/les */
     condition(dif(c,sujet)).
     préposition_ou_vide_article_défini(c.t1).
     nom_commun_0(c.t1) ;

/*****************************************************************/
/* Groupe verbal */

groupe_verbal(t1).
    verbe_0(t1);

groupe_verbal(t1).
    verbe_1(t1,t2).
    groupe_nominal(t2);
```

```
/*****************************************************************/
/* Préposition_ou_vide */
     /* préposition vide dans le cas sujet ou objet */

préposition_ou_vide(sujet.t).

préposition_ou_vide(objet.t);

préposition_ou_vide(c.t).  /* préposition non vide */
     condition(dif(c,sujet)).
     condition(dif(c,objet)).
     préposition(c.t);

/*****************************************************************/
/* Préposition_ou_vide_article_défini */
     /* traitement de la contraction entre préposition et article */

préposition_ou_vide_article_défini(c).
     préposition_ou_vide(c).
     article_défini(c);
préposition_ou_vide_article_défini(c).
          contraction_préposition_article_défini(c);

/*****************************************************************/

preterminaux_non_variables;      /* indication pour le traducteur */

article_défini(_);
article_indéfini(_);
contraction_préposition_article_défini(_);
nom_commun_0(_);
point;
préposition(_);
pronom_personnel_sujet(_,_);
verbe_0(_);
verbe_1(_,_);

fin;           /* indication pour le traducteur */
```

4.4.2. The file AccesLexDoc

This file defines the interface between the lexicon and the grammar. It allows to specify for each pre-terminal category how the elements of the category are found in the lexicon.

For example, the lexical category `verbe_0` (verb without complement) comprises for each verb the singular and plural form of the first person of indicative present; the following rules specify the access to these elements depending on the number feature:

```
<verbe_0(c.g.sin).t,verb_0_ind.<t,_>>;
<verbe_0(c.g.plu).t,verb_0_ind.<_,t>>;
```

The file AccesLesDoc is the following:

```
/* AccèsLex KOMBE */

preterminaux_non_variables;

<préposition("à".c).t,prép_à.<t>>;
<préposition(de.c).t,prép_de.<t,_>>.con;
<préposition(de.c).t,prép_de.<_,t>>.voc;
<préposition(en.c).t,prép_en.<t>>;
<préposition(par.c).t,prép_par.<t>>;
<préposition(sous.c).t,prép_sous.<t>>;

<pronom_personnel_sujet(sujet.g.sin).t, pro_perso_suj.<t,_,_>>.con;
<pronom_personnel_sujet(sujet.g.sin).t, pro_perso_suj.<_,t,_>>.voc;
<pronom_personnel_sujet(sujet.g.plu).t, pro_perso_suj.<_,_,t>>;

<contraction_préposition_article_défini(de.mas.sin).t,
     contraction_de.<t,_>>.con;
<contraction_préposition_article_défini(de.g.plu).t,
     contraction_de.<_,t>>;
<contraction_préposition_article_défini("à".mas.sin).t,
     contraction_à.<t,_>>.con;
<contraction_préposition_article_défini("à".g.plu).t,
     contraction_à.<_,t>>;

<verbe_0(c.g.sin).t,verb_0_ind.<t,_>>;
<verbe_0(c.g.plu).t,verb_0_ind.<_,t>>;

<verbe_1(c.g.sin,t2).t,verb_1_ind(t2).<t,_>>;
<verbe_1(c.g.plu,t2).t,verb_1_ind(t2).<_,t>>;

<article_indéfini(_.mas.sin).t,art_indéf_sin.<t,_>>;
<article_indéfini(_.fem.sin).t,art_indéf_sin.<_,t>>;
<article_indéfini(c._.plu).t,condition(dif(c,de)).art_indéf_plu.<t>>;

<article_défini(c.mas.sin).t,
     condition(dif(c,de).dif(c,"à")).art_déf.<t,_,_,_>>.con;
<article_défini(_.fem.sin).t,art_déf.<_,t,_,_>>.con;
```

```
<article_défini(c._.sin).t,art_déf.<_,_,t,_>>.voc;
<article_défini(c._.plu).t,
     condition(dif(c,de).dif(c,"à")).art_déf.<_,_,_,t>>;

<nom_commun_0(_.mas.sin).t,nom_com_0_m.<t,_>>;
<nom_commun_0(_.mas.plu).t,nom_com_0_m.<_,t>>;
<nom_commun_0(_.fem.sin).t,nom_com_0_f.<t,_>>;
<nom_commun_0(_.fem.plu).t,nom_com_0_f.<_,t>>;

<point.t,pt.<t>>;

fin;
```

4.4.3. The file LexiqueDoc

This file is a knowledge base containing the set of words listed in groups according to the categories. Each category is associated possibly with some linguistic information used for analysis or synthesis. Each verb allowing one complement, for example, belongs to the category `verbe_1` and is associated with the preposition introducing the complement.

The file LexiqueDoc is the following:

```
/*Lexique KOMBE */

fenetres;

preterminaux_non_variables;

prép_à( <<"à">>,<>,voc ) ;
prép_de( <<de>,<"d'">>,<>,con ) ;
prép_en( <<en>>,<>,voc ) ;
prép_par( <<par>>,<>,con ) ;
prép_sous( <<sous>>,<>,con ) ;

pro_perso_suj(<<je>,<"j'">,<nous>>,je,con);

verb_0_ind(<<mange>,<mangeons>>,manger,con);
verb_0_ind(<<bois>,<buvons>>,boire,con);
verb_0_ind(<<bouge>,<bougeons>>,bouger,con);

verb_1_ind(<<mange>,<mangeons>>,manger,con,objet.t);
verb_1_ind(<<mange>,<mangeons>>,manger,con,de.t);
verb_1_ind(<<bois>,<buvons>>,manger,con,objet.t);
verb_1_ind(<<bois>,<buvons>>,boire,con,de.t);
verb_1_ind(<<bouge>,<bougeons>>,bouger,con,objet.t);
```

```
art_indéf_sin(<<un>,<une>>,<>,voc);
art_indéf_plu(<<des>>,<>,con);

art_déf(<<le>,<la>,<"l'">,<les>>,<>,con);

contraction_de(<<du>,<des>>,<>,con);
contraction_à(<<au>,<aux>>,<>,voc);

nom_com_0_f(<<tête>,<>>,est_tête,con);
nom_com_0_f(<<viande>,<viandes>>,est_viande,con);
nom_com_0_f(<<eau>,<>>,est_eau,voc);
nom_com_0_f(<<hanche>,<hanches>>,est_hanche,voc);
nom_com_0_f(<<fesse>,<fesses>>,est_fesse,con);
nom_com_0_f(<<oreille>,<oreilles>>,est_oreille,voc);
nom_com_0_f(<<nuque>,<>>,est_nuque,con);
nom_com_0_f(<<épaule>,<épaules>>,est_épaule,voc);
nom_com_0_f(<<main>,<mains>>,est_main,con);
nom_com_0_f(<<cheville>,<chevilles>>,est_cheville,con);
nom_com_0_f(<<lèvre>,<lèvres>>,est_lèvre,con);
nom_com_0_f(<<paupière>,<paupières>>,est_paupière,con);

nom_com_0_m(<<fruit>,<fruits>>,est_fruit,con);
nom_com_0_m(<<bras>,<bras>>,est_bras,con);
nom_com_0_m(<<pied>,<pieds>>,est_pied,con);
nom_com_0_m(<<coude>,<coudes>>,est_coude,con);
nom_com_0_m(<<légume>,<légumes>>,est_légume,con);
nom_com_0_m(<<pain>,<>>,est_pain,con);
nom_com_0_m(<<dos>,<>>,est_dos,con);
nom_com_0_m(<<cou>,<>>,est_cou,con);

pt(<<".">>,<>,con) ;

fin;
```

4.4.4. The file SemantiqueDoc

The file SemantiqueDoc specifies the semantics of the different elements of the grammar. This file gives the definitions in the following order:

- the semantic type of the extern symbols used in the semantic representations (`symboles_externes`); in our case only one symbol (*et*) is used. Its definition precises that it is a binary operator defined over two elements of type `oo` and taking a type `oo` as its result, too;

- the semantic type of superior and terminal categories used in the grammar (`types`) and having a semantic function; in our example a sentence is of type `oo`, a verb phrase of type `<ii,oo>`, etc.;
- the composition rules allowing to calculate the semantic representation of a category on the basis of the semantic represenations of its subcategories (`regles`); each composition rule is associated with a grammar rule (*GrammaireDoc*) and vice-versa;
- the semantic representations of the terminal categories (`definitions`) having a semantic function.

Here is the content of the file *SemanticDoc*:

```
/* Sémantique KOMBE */

symboles_externes;

et(oo,oo) = oo ;

types;

phrase = oo ;
proposition = <ii,oo> ;
groupe_verbal = <ii,oo>;
nom_commun_0 = <ii,oo>;
verbe_0 = <ii,oo>;
groupe_nominal = <ii,oo>;
verbe_1 = <ii,<ii,oo>>;
pronom_personnel_sujet = <ii,oo> ;

regles;

phrase = [ proposition, x ];
proposition = <x, et( [groupe_nominal , x] , [groupe_verbal , x] ) >
     ;
groupe_nominal = pronom_personnel_sujet ;
groupe_nominal = nom_commun_0 ;
groupe_nominal = nom_commun_0 ;
groupe_verbal = verbe_0 ;
groupe_verbal =
     < x, et( [ [verbe_1 , x] , y] , [ groupe_nominal , y]) >;
préposition_ou-vide = <> ;
préposition_ou-vide = <> ;
préposition_ou-vide = <> ;
préposition_ou-vide_article_défini = <> ;
préposition_ou-vide_article_défini = <> ;
```

```
definitions;

nom_commun_0 = <x ,nn(x)>;
pronom_personnel_sujet = <x , nn(x)>;
verbe_0 = <x , nn(x)>;
verbe_1 = <x,<y,nn(x,y)>>;

fin;
```

4.4.5. The file ConceptuelDoc

The file ConceptuelDoc specifies the conceptual constraints in the following order:

- a set of domaines and their hierarchy (`domaines`); two domains are either disjunctive or one is included in the other;
- the list of conceptual constraints associated with the relations (`contraintes`); for each relation, the domains of individuals are specified in order to obtain valid relations.

```
/* Conceptuel KOMBE */

domaines;

racine(tout);
tout.être_vivant.chose;
être_vivant.humain.non_humain;
humain.ou(homme.femme,adulte.enfant);
chose.partie_du_corps.aliment.boisson.autre;
partie_du_corps.mobile.non_mobile;

/*****************************************************************/
contraintes;

je(humain) ;

est_tête(mobile);
est_viande(aliment);
est_eau(boisson);
est_hanche(non_mobile);
est_épaule(mobile);
est_main(mobile);
est_lèvre(mobile);
est_paupière(mobile);
```

```
est_fruit(aliment);
est_pied(mobile);
est_coude(mobile);
est_légume(aliment);
est_pain(aliment);
est_dos(non_mobile);

manger(humain);
manger(humain,aliment);
boire(humain);
boire(humain,boisson);
bouger(humain);
bouger(humain,mobile);

fin;
```

5. The ALS prototypes

5.1. The French system

The French ALS system allows the user to compose sentences in order to communicate with a doctor. The topics of dialogue are the following:

- Health
- Hygiene
- Nourishment
- Reflexion

We have defined a core grammar specifying an interesting subset of the French language. Our French core grammar (abbreviated by FCG in the following) is formally described by a set of lexical and syntactic rules.

5.1.1. French lexicon

The lexical coverage of FCG consists of two sets of lexical elements:

- a set of elements which are independant of the domain (a "general" dictionary); these elements are function words of the language and belong to the "closed" categories;
- a set of elements which depend on the domain and, in our context, refer to different situations of handicapped persons (domain dictionary); these elements belong to the "open" categories.

General dictionary

The set of elements which are independant of the domain consists of the following elements:

- definite articles: *le*, *la*, *l'*, *les* ;
- indefinite articles: *un*, *une*, *des* ;
- indefinite adjectives/pronouns: *chaque*, *tout*, *toute*, *aucun*, *aucune* ;
- demonstrative pronouns: *ce*, *cet*, *cette*, *ces* ;
- possessive adjectives: *mon, ma ,mes, ..., leur*, *leurs* ;
- interrogative pronouns: *quel*, *quelle*, *quels*, *quelles* ;
- simple relative and interrogative pronouns: *qui*, *que*, *qu'* ;
- composed forms of relative and interrogative pronouns: *lequel*, *laquelle*, *lesquels*, *lesquelles* ;

- subject personal pronouns: *je, tu, ..., ils, elles* ;
- reflexive pronouns; *me, te, ..., se, s'* ;
- preverbal personal object pronouns *me, te, ..., leur, en* ;
- postverbal personal object pronouns *moi, toi, ..., eux, elles* ;
- negation adverbs: *ne, n', non, pas* ;
- coordination conjunctions: *ou, et* ;
- prepositions: *par, de, à*, etc. ;
- interrogative formes: *est-ce que, est-ce qu'* ;
- hyphen and accessory *t* (as, for example, in *vient-il ?* or *donne-t-il ?*) ;
- full stop, question and exclamation mark: . ? ! ;

In FCG, the following contracted forms are found:

- preposition *à* + definite article: *au* (*à+le*), *aux* (*à+les*) ;
- preposition *de* + definite article: *du* (*de+le*), *des* (*de+les*) ;
- preposition *de* + definite article: *des* (*de+des*) ;
- preposition *à* + relative or interrogative pronoun: *auquel* (*à+lequel*), *auxquels* (*à+lesquels*), *auxquelles* (*à+lesquelles*) ;
- preposition *de* + relative pronoun: *dont* (*de+qu-*), *duquel* (*de+lequel*), *desquels* (*de+lesquels*), *desquelles* (*de+lesquelles*) ;
- preposition *de* + interrogative pronoun: *duquel* (*de+lequel*), *desquels* (*de+lesquels*), *desquelles* (*de+lesquelles*).

Domain dictionary

The lexical elements which depend on the domain of KOMBE refer to the conversations between doctors and patients. We distinguish different conversation subjects; our studies treat the following themes:

- Medical subjects:

  symptoms and pathology;

  medical treatment and care;

  rehabilitation, ergotherapy, speech training, etc.

- Food:

  problems of biting and swallowing (liquid and solid food).

- Hygiene :

  washing (shower, shampooing, etc.)

  toilet (laxative, constipation, etc.)

- Psychological subjects:

  sense of well-being, satisfaction, hope, disappointment, anxiety, solitude, etc.

We have defined the lexical elements associated to these topics (nouns, adjectives, verbs, etc.) and their relations on the conceptuel level.

The French core and domain dictionaries are listed in Annex C.

5.1.2. French grammar

The following passages will give a description of the syntactic coverage of the FCG in terms of examples and syntax rules.

The forms *est* and *sont* (3rd person, present tense) of the verb *être* are used in FCG in attributive and passive constructions.

Sentences

FCG defines *declarative sentences* (or *propositions*), as for example:

Je ne peux plus bouger les jambes.

The definition of propositions, questions and determinative clauses (participle constructions, relative and adjective clauses) in FCG refers to the notion of *proposition* and *incomplete proposition*.

A proposition consists of a subject noun phrase and a verb phrase (VP). This is described by the following rule:

```
proposition -->
         groupe_nominal
         groupe_verbal
```

We have defined three kinds of verb phrases:

- VP can be an **operator** verb followed by an **infinitive** or **completive** clause; a completive clause is introduced by *que* or pronominalized:

```
groupe_verbal(sujet.g.n.p1,t2,n_eg).
         condition(dif(t3,infinitive).
               ou(dif(p1,p2),dif(m_ode,subjonctif))).
         neg_ne_ou_vide(n_eg).
         verbe_opérateur(sujet.g.n.p1,t2,t3.m_ode,c1).
         neg_pas_ou_vide(n_eg).
         complétive(sujet._._.p2,m_ode,n_eg');
    groupe_verbal(t1,t2,n_eg).
         condition(dif(t3,infinitive)).
         neg_ne_ou_vide(n_eg).
         pronom_personnel_complément_préverbal(comp._).
```

```
          verbe_opérateur(t1,t2,t3.m_ode,c1).
          neg_pas_ou_vide(n_eg);
```

The infinitive clause is described by the rule:

```
groupe_verbal(t1,t2,n_eg).
          condition(dif(t3,complétive)).
          neg_ne_ou_vide(n_eg).
          verbe_opérateur(t1,t2,t3.m_ode,c1).
          neg_pas_ou_vide(n_eg).
                infinitive(t1,c1,n_eg');
```

- VP can be the auxilary verb *être* (*to be*) ou *avoir* (*to have*) followed by a qualifying adjective (attributive construction) which can introduce an infinitive or noun clause (`complétive` in French):

```
groupe_verbal(t1,t2,n_eg).
          neg_ne_ou_vide(n_eg).
          être_ou_avoir(t0,t1,t2).
          neg_pas_ou_vide(n_eg).
          suite_être_ou_avoir(t0,t1,n_eg');
```

  with:

```
suite_être_ou_avoir(t0,c.n.g.p1,n_eg).
          condition(dif(t3,infinitive).
                ou(dif(p1,p2),dif(m_ode,subjonctif))).
          attribut(t0,c.n.g.p1,t3.m_ode,c1).
          complétive(sujet._._.p2,m_ode,n_eg);
suite_être_ou_avoir(t0,t1,n_eg).
          condition(dif(t3,complétive)).
          attribut(t0,t1,t3.m_ode,c1).
          infinitive(t1,c1,n_eg);
suite_être_ou_avoir(t0,t1,n_eg).
          adjectif_0(t0,t1);
          suite_être_ou_avoir(t0,t1,n_eg).
          adjectif_1(t0,t1,c1).
          groupe_nominal(c1._,t3,n_eg);
```

- A verbe phrase (`VP`) can be a verb followed by its complements. A verb can have no (`verbe0`), one (`verbe1`) or two (`verbe2`) complements:

```
groupe_verbal(t1,t2,n_eg).
          neg_ne_ou_vide(n_eg).
          verbe_0(t2,t1).
          neg_pas_ou_vide(n_eg);
groupe_verbal(t1,t2,n_eg).
          neg_ne_ou_vide(n_eg).
          verbe_1(t2,t1,c1).
          neg_pas_ou_vide(n_eg).
          groupe_nominal(c1._,t3,n_eg);
```

```
groupe_verbal(t1,t2,n_eg).
          neg_ne_ou_vide(n_eg).
          un_clitique(t1,c1.r1).
          verbe_1(t2,t1,c1).
          neg_pas_ou_vide(n_eg) ;
groupe_verbal(t1,t2,n_eg).
          neg_ne_ou_vide(n_eg).
          pronom_personnel_réfléchi(t1).
          verbe_pron_1(t2,t1).
          neg_pas_ou_vide(n_eg) ;
groupe_verbal(t1,t2,n_eg).
          neg_ne_ou_vide(n_eg).
          verbe_2(t2,t1,c1,c2).
          neg_pas_ou_vide(n_eg).
          groupe_nominal_1(c1._,t3,n_eg).
          groupe_nominal_2(c2._,t4,n_eg);
groupe_verbal(t1,t2,n_eg).
          neg_ne_ou_vide(n_eg).
          un_clitique(t1,c1.r1).
          verbe_2(t2,t1,c1,c2).
          neg_pas_ou_vide(n_eg).
          groupe_nominal(c2._,t5,n_eg);
groupe_verbal(t1,t2,n_eg).
          neg_ne_ou_vide(n_eg).
          un_clitique(t1,c2.r2).
          verbe_2(t2,t1,c1,c2).
          neg_pas_ou_vide(n_eg).
          groupe_nominal(c1._,t3,n_eg);
groupe_verbal(t1,t2,n_eg).
          neg_ne_ou_vide(n_eg).
          pronom_personnel_réfléchi(t1).
          verbe_pron_2(t2,t1,c1).
          neg_pas_ou_vide(n_eg).
          groupe_nominal(c1._,t3,n_eg);
groupe_verbal(t1,t2,n_eg).
          neg_ne_ou_vide(n_eg).
          deux_clitiques(t1,c1.r1,c2.r2).
          verbe_2(t2,t1,c1,c2).
          neg_pas_ou_vide(n_eg);
groupe_verbal(t1,t2,n_eg).
          neg_ne_ou_vide(n_eg).
          deux_clitiques_dont_un_réfléchi(t1,c1._,_).
          verbe_pron_2(t2,t1,c1).
          neg_pas_ou_vide(n_eg);
```

- The VP can be in affirmative or negative form:
  Je peux bouger le bras.

Je n'arrive pas à mâcher.

The negative form is treated by the two goals `neg_ne_ou_vide(n_eg)` and `neg_pas_ou_vide(n_eg)`; the feature `n_eg` indicates if the sentence is negated or not.

Complements of verbs which do not have the form of a pronoun appear on the right hand side of the verb; their order is specified in the lexical rule of this verb. Pronouns as complements are placed before (preverbal personal pronoun) or behind the verb (postverbal personal pronoun). In both cases the complement is treated by the goals `un_clitique(t1,c1.r1)` if there is only one pronoun, and by `deux_clitiques(t1,c1.r1,c2.r2)` or `deux_clitiques_dont_un_réfléchi(t1,c1._,_)` otherwise.

We use incomplete propositions to describe questions and determinative clauses. An incomplete proposition is a sentence without an element on the level of the subject or the verb phrase level. An incomplete proposition has the characteristics of complete sentences:

- affirmative or negative form of the verb phrase;
- active or passive voice of the VP (if the verb accepts an object complement);
- pronominalisation of the subject or the complements of the verb.

Declarative sentences

A declarative sentence is defined by the following rule:

```
phrase.
         proposition(sans_pronom_sujet_postverbal).
         point;
```

Propositions don't permit the inversion of the subject and the verb.

Noun phrases

Within the fragment of FCG noun phrases are defined on the basis of these rules:

```
groupe_nominal(sujet.r1,avec_pronom_personnel,n_eg).
         pronom_personnel_sujet(sujet.r1) ;
/* moi,toi,lui :
               PronomPersonnelComplémentPostverbal
               différent de sujet, objet, à */
groupe_nominal(c.r1,avec_pronom_personnel,n_eg).
         condition(dif(c,sujet).dif(c,objet).dif(c,"à")).
         préposition(c).
         pronom_personnel_complément_postverbal(c.r1) ;
/* (Prép) ... NC0 ... */
```

```
groupe_nominal(t1,avec_nom,n_eg).
          préposition_ou_vide_article_indéfini(t1).      /* un(e)/des
     */
          nom_commun_0(t1) ;
groupe_nominal(t1,avec_nom,n_eg).
          préposition_ou_vide_article_défini(t1).  /* le/la/l'/les */
          nom_commun_0(t1) ;
groupe_nominal(t1,avec_nom,n_eg).
          préposition_ou_vide(t1).
          adjectif_démonstratif(t1).               /* ce(s)/cet(te) */
          nom_commun_0(t1) ;
groupe_nominal(t1,avec_nom,n_eg).
          préposition_ou_vide(t1).
          adjectif_possessif(t1,_).          /* mon,ma,mes/votre,vos
     */
          nom_commun_0(t1) ;
```

If the NP is the subject or object there is no preposition:

```
préposition_ou_vide(sujet._);
préposition_ou_vide(objet._);
préposition_ou_vide(c._).
          condition(dif(c,sujet)).
          condition(dif(c,objet)).
          préposition(c);
```

Infinitive and noun clauses

Infinitive clauses are defined by the following rule :

```
infinitive(c1,n_eg,<k,<i,<j,e>>>).
          préposition_ou_vide(c1).
          neg_ne_neg_pas_ou_vide(n_eg).
          proposition_moins(sujet.r1,infinitif._,n_eg,<k,<i,<j,e>>>);
```

In `proposition_moins` the subject is missing. The verbe phrase is in the infinitive form. An example of an infinitive clause :

Je peux <manger des légumes>.

Noun clauses are defined by the following rules :

```
complétive(t1,m_ode,n_eg).
          préposition_ou_vide(que._).
          groupe_nominal(t1,avec_pronom_personnel,n_eg).
          neg_ne_ou_vide(n_eg).
          verbe_0(m_ode.présent,t1).
          neg_pas_ou_vide(n_eg);
complétive(t1,m_ode,n_eg).
          préposition_ou_vide(que._).
          groupe_nominal_1(t1,avec_pronom_personnel,n_eg).
```

```
          neg_ne_ou_vide(n_eg).
          verbe_1(m_ode.présent,t1,c1).
          neg_pas_ou_vide(n_eg).
          groupe_nominal_2(c1._,t3,n_eg);
complétive(t1,m_ode,n_eg).
          préposition_ou_vide(que._).
          groupe_nominal(t1,avec_pronom_personnel,n_eg).
          neg_ne_ou_vide(n_eg).
          un_clitique(t1,c1.r1).
          verbe_1(m_ode.présent,t1,c1).
          neg_pas_ou_vide(n_eg);
complétive(t1,m_ode,n_eg).
          préposition_ou_vide(que._).
          groupe_nominal(t1,avec_pronom_personnel,n_eg).
          neg_ne_ou_vide(n_eg).
          pronom_personnel_réfléchi(t1).
          verbe_pron_1(m_ode.présent,t1).
          neg_pas_ou_vide(n_eg);
complétive(t1,m_ode,n_eg).
          préposition_ou_vide(que._).
          groupe_nominal(t1,avec_pronom_personnel,n_eg).
          neg_ne_ou_vide(n_eg).
          verbe_2(m_ode.présent,t1,c1,c2).
          neg_pas_ou_vide(n_eg).
          groupe_nominal_1(c1._,t3,n_eg).
          groupe_nominal_2(c2._,t4,n_eg);
complétive(t1,m_ode,n_eg).
          préposition_ou_vide(que._).
          groupe_nominal(t1,avec_pronom_personnel,n_eg).
          neg_ne_ou_vide(n_eg).
          un_clitique(t1,c1.r1).
          verbe_2(m_ode.présent,t1,c1,c2).
          neg_pas_ou_vide(n_eg).
          groupe_nominal_2(c2._,t4,n_eg);
complétive(t1,m_ode,n_eg).
          préposition_ou_vide(que._).
          groupe_nominal(t1,avec_pronom_personnel,n_eg).
          neg_ne_ou_vide(n_eg).
          un_clitique(t1,c2.r2).
          verbe_2(m_ode.présent,t1,c1,c2).
          neg_pas_ou_vide(n_eg).
          groupe_nominal_1(c1._,t3,n_eg);
complétive(t1,m_ode,n_eg).
          préposition_ou_vide(que._).
          groupe_nominal(t1,avec_pronom_personnel,n_eg).
          neg_ne_ou_vide(n_eg).
          pronom_personnel_réfléchi(t1).
          verbe_pron_2(m_ode.présent,t1,c1).
```

```
          neg_pas_ou_vide(n_eg).
          groupe_nominal_1(c1._,t3,n_eg);
complétive(t1,m_ode,n_eg).
          préposition_ou_vide(que._).
          groupe_nominal(t1,avec_pronom_personnel,n_eg).
          neg_ne_ou_vide(n_eg).
          deux_clitiques(t1,c1.r1,c2.r2).
          verbe_2(m_ode.présent,t1,c1,c2).
          neg_pas_ou_vide(n_eg);
complétive(t1,m_ode,n_eg).
          préposition_ou_vide(que._).
          groupe_nominal(t1,avec_pronom_personnel,n_eg).
          neg_ne_ou_vide(n_eg).
          deux_clitiques_dont_un_réfléchi(t1,c1._,_).
          verbe_pron_2(m_ode.présent,t1,c1).
          neg_pas_ou_vide(n_eg);
```

Examples of noun clauses:

Je veux <que vous veniez>.
Je n'aimerais pas <que vous me mentiez>.
J'exige <que vous me disiez la vérité>.

Annotation:

Certain rules treating common nouns contain one of the following goals: `deux_clitiques_dont_un_réfléchi(t1,c1._,_)`, `deux_clitiques(t1, c1.r1, c2.r2)` or `un_clitique(t1,c2.r2)`. The latter occur in fact in the rules defining constructions of a noun and at least one object complement in pronominal form. Thus, these goals represent the different object complements of common nouns which are not expressed by means of noun phrases or object clauses. They cause a call of the following rules:

one clitic pronoun:

```
/* me/te/se (objet)  je me lave/tu te lave/il se lave */
     un_clitique(sujet.r1,objet.r1).
          pronom_personnel_réfléchi(sujet.r1);
/* me/te (objet)  je te lave/tu me lave/il me(te) lave */
     un_clitique(sujet._._.p,objet.g.n.p').
          condition(dif(p,p').dif(p',3)).
          pronom_personnel_complément_préverbal(objet.g.n.p');
/* me/te/se (= à soi)   je me / tu te / il se donne ... */
     un_clitique(sujet.r1,"à".r1).
          pronom_personnel_réfléchi(sujet.r1);
/* me/te (= à)   je te parle/ ... / il te parle */
     un_clitique(sujet._._.p,"à".g.n.p').
          condition(dif(p,p').dif(p',3)).
          pronom_personnel_complément_préverbal("à".g.n.p');
```

```
/* le/la/l'/les */
     un_clitique(_,objet.g.n.3).
          pronom_personnel_complément_préverbal(objet.g.n.3);
/* lui/leur */
     un_clitique(_,"à".g.n.3).
          pronom_personnel_complément_préverbal("à".g.n.3);
/* en (= Complément du verbe en "de'') */
     un_clitique(_,de.g.n.3).
          pronom_personnel_complément_préverbal(de.g.n.3);
```

two clitic pronouns, one of them reflexive:

```
/* me/te/se (= à soi) + le/la/les */
     deux_clitiques_dont_un_réfléchi(sujet.r1,objet.g.n.3,"à
     ".r1).
          pronom_personnel_réfléchi(sujet.r1).
          pronom_personnel_complément_préverbal(objet.g.n.3);
/* me/te/se (= à soi) + en (= Complément du verbe en "de'') */
     deux_clitiques_dont_un_réfléchi(sujet.r1,de.g.n.3,"à".r
     1).
          pronom_personnel_réfléchi(sujet.r1).
          pronom_personnel_complément_préverbal(de.g.n.3);
```

two clitic pronouns:

```
deux_clitiques(t1,t2,t3).
          deux_clitiques_dont_un_réfléchi(t1,t2,t3);
/* me/te(= à) + le/la/les */
     deux_clitiques(sujet._._.p,objet.g.n.3,"à".g'.n'.p').
          condition(dif(p,p').dif(p',3)).
          pronom_personnel_complément_préverbal_1("à".g'.n'.p').
          pronom_personnel_complément_préverbal_2(objet.g.n.3);
/* me/te (= à) + en (= Complément du verbe en "de'') */
     deux_clitiques(sujet._._.p,de.g.n.3,"à".g'.n'.p').
          condition(dif(p,p').dif(p',3)).
          pronom_personnel_complément_préverbal_1("à".g'.n'.p').
          pronom_personnel_complément_préverbal_2(de.g.n.3);
/* le/la/les  + lui/leur */
     deux_clitiques(_,objet.g.n.3,"à".g'.n'.3).
          pronom_personnel_complément_préverbal_1(objet.g.n.3).
          pronom_personnel_complément_préverbal_2("à".g'.n'.3);
/* lui/leur + en (= Complément du verbe en "de") */
     deux_clitiques(_,de.g.n.3,"à".g'.n'.3).
          pronom_personnel_complément_préverbal_1("à".g'.n'.3).
          pronom_personnel_complément_préverbal_2(de.g.n.3);
```

5.2. The German system

5.2.1. General remarks

Like the French ALS system, the German prototype also refers to the conversations between a doctor and his patient. When defining the lexical and syntactic coverage of the German KOMBE system we referred to this special dialogue situation.

We aimed at allowing the user to express his ideas by means of the sentences generated by the system, but we didn't take into consideration paraphrases and synonyms because it will be sufficient for the patient to express his idea in one way.

5.2.2. German lexicon

The lexicon for the German fragment consists of two parts: a core lexicon and a domain lexicon. The core lexicon is independant of the domain of use. It contains function words of the following categories: articles, personal, reflexive, possessive, indefinite and interrogative pronouns, adverbs, auxiliary and modal verbs, conjunctions, prepositions as well as punctuation marks. At present, this core lexicon is not complete, because it only contains the most frequent words of the above categories.

In contrast the domain lexicon is closely connected to the domain of use: it contains words of the open word classes (nouns, verbs, adjectives etc.) specific to the conversation topics of dialogues between the patient and his doctor. The actual German system refers to the following hierarchy of topics:

1. Der Mensch (Körper) *Human being (body)*

a. Medizin *Medicine*

   i. Beschwerden *Symptoms*
   ii. Therapie *Therapy*
   iii. Medikamente *Medicine*

b. Ernährung *Eating and Drinking*

   i. Mahlzeiten *Meals*
   ii. Zubereitung *Preparation/Cooking*

c. Hygiene *Hygiene*

   i. Kleidung *Clothes*
   ii. Körperpflege *Personal hygiene*
   iii. Toilette *Toilet*

d. Beweglichkeit *Mobility*

e. Gedanken *Reflection*

The German core and domain dictionaries are listed in Annex B.

5.2.3. German syntax

Defining the fragment

When defining the fragment of German we took into consideration two aspects. First, statistical aspects were taken into account in order to allow the most frequent constructions of spoken German. On the other hand we tried to restrict the number of alternative constructions expressing the same idea.

Statistical aspects

Up to now, statistical research on the syntax of spoken German is very rare. We only know one corpus of spoken German which is available on material readable by computers: the *Freiburger Korpus*[7]. Unfortunately this corpus doesn't contain many records of personal communication and only one consultation with a dentist. In addition to this consultation we investigated five conversations of every-day life.[8]

For this reason we decided to collect data ourselves. A doctor who treats ALS patients at a university clinic in Munich recorded conversations with her patients; we analyzed these conversations in regard to their syntax and to the words contained there.

The results of our studies on the syntax can be resumed as follows:

- All types of sentences are important: propositions, questions and imperatives. Evidently, there are quite a lot of sentence fragments (apart form the responses to answers) because people often interrupt each other during conversations. In order to allow the users to produce short utterances as well it is important that the **speak** button can always be activated.
- Compound sentences are formed as either coordinate clauses or subordinate clauses of special types. Among the latter, frequent constructions are object, adverbial, conditional and relative clauses as well as indirect questions and object clauses without conjunctions. Infintive constructions and subject clauses rarely occur. Nearly all types of subordinate clauses

---

7 This corpus contains about 500 000 words; parts of it have been published in *Heutiges Deutsch.* We also refer to research on this corpus, cf. JÄGER (1979).

8 They have been drawn form FUCHS (1975) and JÄGER (1979).

without conjunction (except object clauses) and participle constructions have not been found.

- The analysis of verbs and the types of complements showed an interesting result: Some sentence types[9] which are frequently used in the written language[10], are rarely used in spoken German; above all these are sentence types with prepositional objects. In addition, a few sentence types which are rarely used in the written language are frequent in spoken German. As a result we integrated 12 sentence types in the prototype of the KOMBE system.
- Fixed verb systems (Funktionsverbgefüge) were only found in one case. We decided not to consider these constructions because there are simple alternatives (cf. 1.2).
- The use of adverbs and particles is important. Sentence equivalents (*ja, nein, ach, ach so* etc.) aren't yet included.

Excluding equivalent constructions

As we suppose that it will be sufficient for the users of the KOMBE system to express their ideas in one way, it is not very important to allow a number of equivalent constructions. We try to allow the simplest and shortest one among several possibilities. For this reason we refer to research about simplified German.[11] We propose the following simplifications:

- no passive voice because (1) the agent need not be mentioned in this construction (but this would be important in the case of our conversations) and (2) active voice is more easy to use and to understand (COLEMAN);
- the subject has to be put at an initial position of the sentence; this is important in the case of verb second word order in German, where nearly all types of complements can be topicalized; we only allow subjects, adverbs and wh-complements to be put at the beginning of the sentence, and prescribe that the subject (if it is not topicalised) must be the second complement within the sentence.
- no double negation;

---

9 A *sentence type* is a specification of the set of complements of a verb as $NP_{nom} - NP_{akk}$, for example.

10 We refer to DUDEN, *Grammatik der deutschen Gegenwartssprache*, Mannheim 1984[4], p. 634.

11 Above all we refer to the work of A. LERNDORFER (forthcoming) and additionally to COLEMAN (1965) and TEIGELER (1968)

- only one subordination: this has the advantage that (1) the sentences can't be too long and (2) the distance between a verb and its prefix is not too great.

Syntactic structures of the German KOMBE grammar

In German the type of a sentence can be characterized by the position of the inflected verb. This verb form can be placed at three positions within the sentence: at the beginning af the sentence (verb-first order), after the first complement (verb-second order) and after all complements (verb-last order).

The German KOMBE Grammar allows simple questions and complex sentences. Phrase structure rules for the node `sentence` are:

```
Sentence -->   Sentence_verb-first Question-mark
Sentence -->   Sentence_verb-second Full-stop-or-Question-mark
Sentence -->   Sentence_verb-second Subordinate_verb-last
               Full-stop-or-Question-mark
Sentence -->   Subordinate_verb-last Sentence_verb-first Full-stop
```

KOMBE Rules:[12]

```
phrase(e).
     satz(ve1(anfang),e).
     fragezeichen;
phrase(e1.e2).
     satz(ve2(t),e1).
     subord_oder_leer(e2).
     punkt_fragezeichen(t);
phrase(e1.e2).
     satz_ve_letzt(konj(subord.t1),<e',e1>).
     satz(ve1(nach_subord),e2).
     punkt;

subord_oder_leer(nil);
subord_oder_leer(e).
     satz_ve_letzt(konj(subord.t1),<e',e>);
```

Verb-first order is used in alternative questions, imperatives and main clauses following the subordinate clause[13]:

---

12 The semantics was integrated into the German syntax rules and not generated by the compiler, because the necessary programs were not yet available when we developed the German system. The variables *e, e1, e2, ..., i* and *j* refer to semantical entities and formulae.

13 This word order is also used in conditional and concessive clauses without conjunctions (*Sollte es morgen regnen, gehen wir ins Kino*), but this type of clause was not found in the examples of spoken German.

```
Rule:    Sentence_verb-first --> Verb_fin Field Predicate Extra
```

KOMBE Rules:

```
satz(ve1(nach_subord),e.e').
     vv_fin(ve1,m_vv,<s_em_verb,e>).
     feld(m_vv,<<t,nneg>,n_eg>,e').
     praed(ve1,m_vv,n_eg,<s_em_verb,e>);
satz(ve1(s_tell),e1.e2.e).
     vv_aux_mod(m_vv,<i,<j,e1>>).
     feld(m_vv,<<t,nneg>,n_eg>,e2).
     praed(ve1,m_vv,n_eg,<<i,<j,v>>,e>);
```

Examples:

Kannst du mir die Jacke anziehen?
Habe ich dir meine Jacke gegeben?
Wenn ich die Tabletten nehme, lassen die Schmerzen nach.

Verb-second word order is found in propositions, *wh*-questions and verb-second complements (object and subject clauses without conjunctions):

```
Rule:    Sentence -->     Topic Verb_fin Field_minus Predicate Extra
```

KOMBE Rules:

```
satz(ve2(t1),e1.e.e2).
     top(t1,m_vv,<<t,nneg>,n_eg1>,l1,e1).
     vv_fin(ve2,m_vv,<s_em_verb,e>).
     feld_minus(t1,m_vv,<n_eg1,n_eg>,l1,e2).
     praed(ve2,m_vv,n_eg,<s_em_verb,e>);
satz(ve2(t1),e1.e2.e3.e).
     top(t1,m_vv,<<t,nneg>,n_eg1>,l1,e1).
     vv_aux_mod(m_vv,<i,<j,e2>>).
     feld_minus(t1,m_vv,<n_eg1,n_eg>,l1,e3).
     praed(ve2,m_vv,n_eg,<<i,<j,v>>,e>);
```

Examples:

Ich möchte eine Banane essen.
Ich esse eine Banane.

Verb-last word order is a charactersitical feature of subordinate clauses beginning with a conjunction or a relative pronoun:

```
Sentence -->   Conj-or-Rel Field Predicate V_fin Extra
```

KOMBE Rules:

```
satz_ve_letzt(m1,<<j,e>,e1.e2.e'>).
     konj_oder_rel(m1,m2,m_vv,l1,<j,e.e1>).
     feld_minus(m2,m_vv,<<t,nneg>,n_eg>,l1,e2).
     praed(ve_letzt,m_vv,n_eg,<s_em_verb,e'>).
```

```
     vv_fin(ve_letzt,m_vv,<s_em_verb,e'>);
satz_ve_letzt(m1,<<j1,e>,e1.e2.e'.e''>).
     konj_oder_rel(m1,m2,m_vv,l1,<j1,e.e1>).
     feld_minus(m2,m_vv,<<t,nneg>,n_eg>,l1,e2).
     praed(ve_letzt,m_vv,n_eg,<<i,<j,v>>,e'>).
     vv_aux_mod(m_vv,<i,<j,e''>>);
```

Examples:

Ich kenne den Arzt, der sie operiert.
Ich kenne den Arzt, der sie operiert hat.
Ich kenne den Arzt, der sie operieren will.

The field

As just mentioned above, we restrict the relatively free word order within the field, that means between the finite verb (resp. conjunction or relative constituent in verb-last order) and the other parts of the predicate. We defined general rules in order to describe the complements as parts of the field and expressed the constraints for the right order and the possible combinations by means of special conditions.

The field (*feld*) consists of one to three complements (*sg, sg2, sg3*) depending on the main verb of the sentence, the field missing a complement (*feld_minus*) comprises at most two complements. When the last complement has been analyzed the system checks whether the sentence is not missing an obligatory complement (*test_verbrahmen*). If the verb is in the imperative form (what is the case of the second rule) the field has the form of *feld_ minus* which is missing the subject (personal pronoun, 2nd person).

KOMBE Rules:

```
feld(<p,n,t,m>.i_nf.l_iste,<n_eg,n_eg'>,e1.e2).
     condition(dif(p,imp)).
     condition(dif(s_g,praed)).
     sg(np(<p,nom,g,n>.t_yp_subj.def.r),l_iste,l1,<n_eg,n_eg1>,e1).
     sg2(vorg(np(<p,nom,g,n>.t_yp_subj.def.r)),t2,l1,l2,
               <n_eg1,n_eg2>,e2).
     sg3(vorg(np(<p,nom,g,n>.t_yp_subj.def.r),t2),t3,l2,l3,
               <n_eg2,n_eg'>,e2).
     condition(freeze(l3,test_verbrahmen(l3)));
feld(<p,n,t,m>.i_nf.l_iste,n_eg,du_Sie_imp(j).e2).
     condition(bound(p)).
     condition(eq(p,imp)).
     condition(freeze(l_iste,

          streichen(np(<2,nom,g,n>.pron(per).nil),j,l_iste,l1))).
```

```
     feld_minus(np(<2,nom,g,n>.pron(per).nil),
                <2,n,t,m>.i_nf.l_iste,n_eg,l1,e2);
feld(<p,n,t,m>.i_nf.l_iste,<n_eg,n_eg'>,e1.e2.nil).
     condition(dif(p,imp)).
     condition(kopula_verb(i_nf)).
     sg(np(<p,nom,g,n>.t_yp_subj.def.r),l_iste,l1,<n_eg,n_eg1>,e1).
     sg_kop(<t_yp,<p1,nom,g1,n>.r1>,l1,l2,<n_eg1,n_eg'>,e2).
     condition(freeze(l2,test_verbrahmen(l2)));

feld_minus(s_g(<p,nom,g,n>.r),
                <p,n,t,m>.i_nf.v_r,<n_eg,n_eg'>,l_iste,e).
     sg2(top(s_g(<p,nom,g,n>.r)),t2,l_iste,l1,<n_eg,n_eg1>,e).
     sg3(vorg(s_g(<p,nom,g,n>.r),t2),t3,l1,l2,<n_eg1,n_eg'>,e).
     condition(freeze(l2,test_verbrahmen(l2)));
feld_minus(<s_g,t1>,<p,n,t,m>.i_nf.v_r,<n_eg,n_eg'>,l_iste,e).
     condition(dif(s_g,konj)).
     sg2(top(<s_g,t1>),s_g2(<p,nom,g,n>.r),l_iste,l1,<n_eg,n_eg1>,e).
     sg3(vorg(<s_g,t1>,s_g2(<p,nom,g,n>.r)),t3,l1,l2,<n_eg1,n_eg'>,e).
     condition(freeze(l2,test_verbrahmen(l2)));
feld_minus(<s_g,<p,nom,g,n>.r>,
                <p,n,t,m>.i_nf.v_r,n_eg,l_iste,e.nil).
     condition(kopula_verb(i_nf)).
     sg_kop(<t_yp,<p1,nom,g1,n>.r1>,l_iste,l1,n_eg,e).
     condition(freeze(l1,test_verbrahmen(l1)));

feld_minus(konj(t),m_vv,n_eg,l_iste,e).
     feld(m_vv,n_eg,e);
feld_minus(ADV(fak(r)),m_vv,n_eg,l_iste,e).
     feld(m_vv,n_eg,e);
```

The order of the complements within the filed can be described as follows:

- A NP in the nominative case can be put before or behind personal and reflexive pronouns and even between several pronouns. However, it must be placed before further nominal or prepositional phrases:

  Wann hat Petra sich dir vorgestellt?
  Wann hat sich Petra dir vorgestellt?
  Wann hat sich dir Petra vorgestellt?
  Wann hat Petra sich dem Personalchef vorgestellt?
  Wann hat sich Petra dem Personalchef vorgestellt?

- The order of personal and reflexive pronouns depends on the cases of the pronouns:

  $P_{Nom} < P_{Acc} < P_{Dat}$
  Wann hat sie sich dir vorgestellt?

Warum hat man es dir weggenommen?

- Objects realised as NP's have to be placed before prepositional objects:
  Ich halte die Entscheidung für einen Fehler.
  Er dankte dem Ziwi für seine Hilfe.
  Er zeigt ihr die Briefe auf dem Schreibtisch.

All these conditions of ordering two complements `s_g1` and `s_g2` are defined by the rules `nachfolge(s_g1,s_g2)` of the file *ConditionsPro* .

Topicalisation

In German it is possible to topicalise NP's, personal pronouns, prepositional phrases, adverbs, predicatives (adjectives and pronouns), infinitives and participles. As just mentioned above, we only allow subjects, adverbs and *wh*-phrases at the beginning of the sentence. Adverbs occur very frequently at sentence beginnings in spoken German.

KOMBE rules:

```
top(np(<p,nom,g,n>.t_yp.def.r),<p,n,t,m>.i_nf.l_iste,n_eg,l1,e).
     sg(np(<p,nom,g,n>.t_yp.def.r),l_iste,l1,n_eg,e);
top(np(<p,k_as,g,n>.np(wh).r),<p1,n1,t,m>.i_nf.l_iste,n_eg,l1,e).
     sg(s_g(<p,k_as,g,n>.np(wh).r),l_iste,l1,n_eg,e);
top(pp(p_raep.wh.r),<p,n,t,m>.i_nf.l_iste,n_eg,l1,e).
     sg(pp(p_raep.wh.r),l_iste,l1,n_eg,e);
top(ADV(m),<p,n,t,m>.i_nf.l_iste,n_eg,l1,e).
     sg(ADV(m),l_iste,l1,n_eg,e);
```

Predicates

The elements of the predicate (except the finite verb) are the sentence negation *nicht*, infinitives and participles or verb prefixes (which often have the form of prepositions). Fixed verb systems (support verbs) are not treated. Obligatory local or temporal adverbials and predicatives of copula verbs are treated as complements (*sg*). Special attention has been paid to the right place of the sentence negation in the latter cases.

Examples:

Klara hat mir die Jacke gegeben.
Klara will mir die Jacke geben.
Klara hat mir die Jacke nicht gegeben.
Klara zieht die Jacke aus.
Klara zieht die Jacke nicht aus.
Klara hat die Jacke nicht ausgezogen.
Diese Krankheit ist nicht leicht.

KOMBE rules:

```
praed(v_e,s_ynt.<i_nf,voll>.l_iste,n_eg,<s_em_verb,e>).
     negation_praed(i_nf,<n_eg,n_erg>);
praed(v_e,s_ynt.<i_nf,aux>.l_iste,n_eg,<s_em_verb,e>).
     negation_praed(i_nf1,<n_eg,n_erg>).
     condition(infinitiv(i_nf1,v_r1,<s_em_verb,e>,k_lasse,i_nf)).
     condition(verbrahmen(v_r1,s_em_verb,l_iste)).
     part(i_nf1);
praed(v_e,s_ynt.<i_nf,mod>.l_iste,n_eg,<s_em_verb,e>).
     negation_praed(i_nf1,<n_eg,n_erg>).
     condition(infinitiv(i_nf1,v_r1,<s_em_verb,e>,k_lasse,i_nf2)).
     condition(verbrahmen(v_r1,s_em_verb,l_iste)).
     vv_inf(i_nf1);
praed(v_e,s_ynt.<i_nf,pref(p)>.l_iste,n_eg,<s_em_verb,e>).
     negation_praed(i_nf,<n_eg,n_erg>).
     condition(dif(v_e,ve_letzt)).
     praepos_oder_prefix(p);

praepos_oder_prefix(p).
     praepos(p.r1);
praepos_oder_prefix(p).
     prefix(p);
```

Negation

The German KOMBE grammar allows two types of negation. The scope of the negation can be a phrase or the whole sentence. If a phrase is negated it is preceeded by the negation adverb *nicht (not),* it is a negative pronoun (as *niemand – nobody*) or it has a negative determiner (such as *kein – no*). In the case of sentence negation the adverb *nicht* is part of the predicate. As we don't allow double negation, we record by means of a tuple `<t_ype,neg>` if a phrase or the sentence is already negated.

KOMBE rules:

```
/* <t,nneg>          no negation within the sentence
     <t,neg>          phrase negation
     <satz_neg,nneg>  no negation of the sentence (important for
                           copula constructions or obligatory PP)
     <satz_neg,neg>   sentence negation

Sentence negation */
negation_praed(i_nf,<n_eg,n_eg>);
negation_praed(i_nf,<<t,nneg>,<satz_neg,neg>>).condition(free(t)).
     neg_nicht;

/* Phrase negation */
negation_oder_nicht(<n_eg,n_eg>);
```

```
negation_oder_nicht(<<t,nneg>,<t,neg>>).
     neg_nicht;
```

Nominal phrases

Evidence from several investigations of spoken language shows, that certain types of nominal phrases are rarely used by the speakers. In order to restrain the number of words proposed by the system in this domain, we defined a package of rules for nominal phrases large enough to allow comfortable composition of utterances needed for daily life communication, and restrained enough to exclude very complicated constructions only used in written language on a high stilistic level.

In detail we neither admit more than one prepositonal complement, nor do we allow to iterate recursive structures more than twice, deeper embeddings being theoretically possible, but not being used in daily speech. This iteration constraint is achieved by a very simple programming mechanism, which can easily be modified to allow deeper embeddings.

In the following we present the NP-constructions included in the grammar with some examples and the corresponding grammar rules.

Personal reflexive pronouns

Examples:

ich,mich

KOMBE rules:

```
np(m.pron(per).<n_eg,n_eg>.def.l,<j,e>).
     per_pron(m.l,<j,e>);
np(m.pron(refl).<n_eg,n_eg>.def.l,<j,e>).
     refl_pron(m.l,<j,e>);
```

Other pronouns

Examples:

jeder, niemand ;

KOMBE rules:

```
np(<3,k,g,n>.pron(pron).n_eg.def.l,<j,e>).
     pronomen1(<k,g,n>.n_eg.r_est,<j,e>);
```

Proper names

Examples:

Max, Maria, Marseille, München

KOMBE rules:

```
np(<3,k,g,n>.pn.<n_eg,n_eg>.def.l,<j,e>).
     pn(<k,g,n>.nil,<j,e>);
```

Full-NPs

We have designed different rules for possessive and other determiners, as the former exclude certain complements.

The feature `def` is necessary as in German some common nouns have different forms, depending on the definiteness of the article:

Examples:

der Kranke/ein Kranker

KOMBE Rules:

```
np(<3,k,g,n>.np(d_ef).n_eg.d_ef.l,<j,e>).
     det1(<k,g,n>.d_ef.n_eg.d_et_nom.r_est1).
     nbar(<k,g,n>.d_ef.nposs.d_et_nom.l,<j,e>);
np(<3,k,g,n>.np(ndef).<n_eg,n_eg>.def.l,<j,e>).
     poss(<p,k,g,n>.r_est1).
     nbar(<k,g,n>.ndef.poss.l,<j,e>);
```

Rest-NP

On this level we introduce adjectives, different complements and (facultative) relative phrases; this allows for a large number of NP-constructions of a certain complexity like:

Examples:

der junge Arzt mit dem Bart
der junge Arzt, der mich operiert hat
der junge Arzt mit dem Bart, der mich operiert hat

KOMBE Rules:

```
nbar(<k,g,n>.d_ef.nposs.d_et_nom.i_t.l,<j,e.e1.e2>).
     fak_adj(<k,g,n>.d_ef.nil,<j,e>).
     cn(<k,g,n>.d_et_nom.k_ompl.nil,<j,e1>).
     fak_relativsatz(<g,n>.l,<j,e1.e2>);
nbar(<k,g,n>.d_ef.poss.d_et_nom.i_t.l,<j,e.e1>).
     fak_adj(<k,g,n>.d_ef.nil,<j,e>).
     cn(<k,g,n>.d_et_nom.k_ompl.nil,<j,e1>);
```

Adjectives as complements for nouns are facultative. In German their form corresponds to the features of the common noun, depending further on the definiteness of the article .

KOMBE Rules:

```
fak_adj(<k,g,n>.d_ef.nil,<j,e>);
fak_adj(<k,g,n>.d_ef.nil,<j,e>).
     adj(<k,g,n>.d_ef.nil,<j,e>);
```

Different forms of subcategorized noun complements

Some nouns do not subcategorize complements, others take facultatively a certain complement, and finally there are nouns having obligatory complements. A simple mechanism allows to control nested structures.

KOMBE Rules:

```
np_kompl(<fak,_>._);
np_kompl(<fak,x>.i_t).
     np_kompl(x.i_t);

np_kompl(gen.i_t).condition(iterations(i_t,i_t1)).
     np(<3,gen,g,n>._.<<t,neg>,n_eg1>.i_t1,<j,e>);
np_kompl(pp(p_raep).i_t).condition(iterations(i_t,i_t1)).
     pp(p_raep.<<t,neg>,n_eg1>.i_t1,<j,e>);
```

Not subcategorized complements can always be omitted.

KOMBE Rules:

```
np_kompl(frei._);
np_kompl(frei.i_t).
     condition(iterations(i_t,i_t1)).
     pp(von.<<t,neg>,n_eg1>.i_t1,<j,e>);
```

Iteration constraint for embedding of complements

Prolog Rules:

```
iterations(it0,it1) -> !;
iterations(it1,it2) ->;
```

Prepositional phrases

As in German some prepositions are concatenated with the definite article while others are not, a lot of rules are necessary to define prepositional phrases.

Examples:

im Krankenhaus

vor dem Krankenhaus

KOMBE Rules:

```
pp(p_raep.<n_eg,n_eg>.i_t.l,<j,e>).
     praepos_plus_det(p_raep.<k_as,g,n>.r_est).
     nbar(<k_as,g,n>.def.nposs.i_t.l,<j,e>);

pp(p_raep.n_eg.i_t.l,<j,e>).
     praepos(p_raep.k_as.r_est).
     condition(dif(f_orm,np(def))).
     np(<3,k_as,g,n>.f_orm.n_eg.i_t.l,<j,e>);

praepos_plus_det(p_raep.<k_as,fem,n>.nil).
     praepos(p_raep.k_as.r_est).
     det_oder_leer(<k_as,fem,n>.def.nneg.r_est1);

praepos_plus_det(p_raep.<k_as,g,n>.nil).
     praepos(p_raep.k_as.nklit.r_est).
     det_oder_leer(<k_as,g,n>.d_ef.nneg.r_est1);
```

Relative pronouns and relative phrases

Their form is equal to the definite article, except for the genitive form of relative pronoun, which is only used with an additional NP, replacing the article.

Examples:

der Arzt, dessen Behandlung

KOMBE Rules:

```
relativphrase(<3,k,g,n>.<g,n>.nil,<<j,e>,j>).
     condition(dif(k,gen)).
     condition(dif(<k,n>,<dat,plu>)).
     det(<k,g,n>.def.nneg.nil);
relativphrase(<3,dat,g,plu>.<g,plu>.nil,<<j,e>,j>).
     rel_pron(<dat,g,plu>.nil);
```

Special features of programming the German fragment

When implementing the grammar in Prolog, there are several ways to consider the relatively free order of complements in German:

- All possibilities of order are described by a multitude of Prolog rules.
- We define control predicates that supervise how often a type of complement may occur (such as the parenthesis and asterisk of the Kleene notation). A general mechanism accepts just the number and types of objects needed for the main verb of the sentence.

- We adopt the method of incomplete propositions used in FCG in order to define verb-second sentences as consisting of a topic and an incomplete proposition with the topic missing.
- We define a further control mechanism by means of coroutines using *freeze* in order to control the syntactic constraints of the verb of the sentence and the complements being parsed.

A further problem will be the fact that the (main) verb is sometimes known only at the end of the sentence and can't thus be controlling the composition. If the finite verb is an auxiliary or modal verb in verb-first or verb-second order, it would be an efficient solution to quit the normal composition mode from left to right in the sentence, and present participles and infinitives to the user immediately after the finite verb; the system could retain this choice up to the end of the sentence and use it to control the composition of the objects. Despite the advantages, we didn't realize this idea because we were not sure whether the users of the communication aid would be able to choose the verb immediately after the conjunction or relative pronoun; perhaps they would be confused being asked to choose words in such an unusual order.

In order to resolve these problems of implementation we have chosen the following solution:

- In general we use the same formalisms as for the French grammar.
- We adopt the method of incomplete propositions used in FCG (French Core Grammar) in order to define verb-second sentences as consisting of a topic and an incomplete proposition with the topic missing.
- We have defined a further control mechanism by means of coroutines using *freeze* (the rules *sg* and s*treichen*) in order to control the syntactic constraints of the verb of the sentence and the complements being parsed.

  The grammar rule *sg* to parse a complement is coroutined by deleting it in the list of complements given by the main verb of the sentence (by calling *streichen*); the use of *freeze* is necessary in order to avoid the deleting process in a still unknown list. The predicate *streichen* refers to the variable *j* identifying the semantics of the complement; during the deleting process *j* is unified with the corresponding argument in the λ-expression representing the verb semantics.

```
/* Complement of the type NP, but no reflexive pronoun, no double
     negation */

sg(np(m.t_yp.l),l_iste,l1,<n_eg,n_eg'>,e).
     condition(dif(t_yp,pron(refl))).
     condition(freeze(l_iste,
          streichen(np(m.t_yp.l),j,l_iste,l1))).
```

```
     negation_oder_nicht(<n_eg,n_eg1>).
     np(m.t_yp.<n_eg1,n_eg'>.l,<j,e>);

/* A complement of the type 'x' is deleted in the list of obligatory
     complements given by the main verb; the semantics 'j' is unified
     with the argument of the verb semantics given by the construction
     of the verb (the rules 'verbrahmen') */
streichen(x,j,ob(j.x).r,r) ->;
streichen(x,j,<o,i.y>.l,<o,i.y>.r) ->
     dif(x,y)
     freeze(y,streichen(x,j,l,r));

/* the verb 'essen' characterizes sentences of the type 'vr1' (it is
     a transitive verb),
          λ-expression of the type: <<i,<j,<c1,v>>>
          the semantics: v = essen(i,j,c1)
          conjugation: VSW1
          no separable prefix
          auxiliary: 'haben' */

infinitiv("essen",vr1,<<i,<j,<c1,v>>>,essen(i,j,c1)>,
          <VSW1,pref(nil)>,"haben") ->;

/* verb with frames of the type 'vr1' have the λ-expression
     <i,<j,<c1,v>>> and two complements:
          NP:  case: nominative, semantics 'j'
          NP:  case: accusative, semantics 'c1'*/

verbrahmen(vr1,<i,<j,<c1,v>>>,
     <ob,j.np(<p,nom,g_en,n>.r)>.
     <ob,c1.np(<p1,akk,g_en1,n_um1>.r1)>.nil) ->;
```

- The grammar rules accept at most three complements of a sentence; the right order is defined by separate rules (`nachfolge`); the possible combinations within a sentence (as: two nominal phrases in nominative, but not two nominal phrases in genitive) are specified by the rules `moegliche_kombinationen`. This is important in all cases of the main verb at the end of the sentence.

```
/* nachfolge: conditions for complement order. If the first pronoun
     is in nominative the following pronoun can only be one of a
     different case */
     nachfolge(np(<p,nom,g,n>.<pron,t_yp>.r),
                     np(<p1,k_as,g1,n1>.<pron,t_yp1>.r1)) ->
               dif(k_as,nom);

/* moegliche_kombinationen: two complements the nominative of one of
     the complements is already defined by the call in 'feld' or
```

```
     'feld_minus' if there are only two complements, no identity of
     the persons is possible (in order to exclude 'ich will ich ...'
     or similiar sentences) but two complements in nominative  (naming
     different persons) are allowed */
moegliche_kombinationen2(s_g,s_g1) ->
     nicht_zwei_gleiche_pers(s_g,s_g1);

/* moegliche_kombinationen: three complements one complement in
     nominative case is necessary, but if there are two nominative
     complements, a third one is not possible (copula construction)
     (for this reason moegliche_kombinationen fails for 2 nominative
     complements); two complements in dative or genitive case are
     excluded */
moegliche_kombinationen(<np,<p,nom,g,n>.r>,
          <np,<p1,k_as1,g1,n1>.r1>,<np,<p2,k_as2,g2,n2>.r2>) ->
     dif(<k_as1,k_as2>,<gen,gen>)
     dif(<k_as1,k_as2>,<dat,dat>)
     dif(k_as1,nom)
     dif(k_as2,nom);
```

As the rules defined by the predicates `moegliche_kombinationen`, `nicht_zwei_gleiche_pers`, `nachfolge`, `verbrahmen`, `infinitif` and `streichen` represent additional constraints to the grammar rules, their definitions are part of the file ConditionsPro.

5.3. Lexical acquisition

A device for lexical acquisition was implemented for the French and the German version of KOMBE. The aim of this sub-system is to allow the user of the system to add new words to the lexicon. This need might arise when the user doesn't find a word in the lexicon supplied with the system, but which and he or she wants to use frequently. This might be proper names of his friends and parents, or words which aren't very frequent, but which are preferred by the user.

The words that can be added by means of this lexical acquisition interface are only words of the following categories: nouns (proper nouns and common nouns), adjectives, adverbs and verbs. These are the open classes, forming the major part of the lexicon for every language. The other classes consist of a fixed inventory. The frequent words of these categories are already included in the lexicon of the system.

The task of lexical acquistion tool is more difficult in the framework of the KOMBE system than for systems purely based on statistics, because the syntac-

tical and conceptual features of the word to be added are needed to assure that it can be used properly by the analysis/synthesis mechanisms.

In the following paragraphs, we will first make some general remarks about the device, defining the user requirements and the information to be determined. Then we will describe in detail the determination of the morphological, syntactical and conceptual information for French and German, and finally we present some examples for dialogues with the user.

5.3.1. User requirements for lexical acquisition

The system KOMBE makes use of syntactical and conceptuel knowledge. These features are associated with a word and its forms. When the user wants to add a new word, the system has thus to performs the following tasks:

- finding the different word forms (morphological component)
- finding the syntactical features of the word and its forms (syntactical component)
- determining the relevant phonological information (phonological component, only for French)
- finding the conceptuel information associated to the word (conceptual component)

Some of the tasks can be performed by the system itself, others require information exchange with the user of the system.

In order to define the suited procedures, we have to consider the user requirements. As the user of the system has very restrained input media (single switch in the worst case), the most important goal is to reduce the user input as much as possible. All information which can be provided by the system should be calculated in order to reduce the cognitive load for the user. He shouldn't be obliged to type whole word forms, but rather make a choice between preformulated alternatives. In addition, the user doesn't necessarily have any formal education. Thus all tasks he has to perform should be well explained. The dialogues must not contain linguistic vocabulary.

5.3.2. Determining the necessary information

When the device is called from the KOMBE-system, the user has to specify what kind of word he wants to add to the lexicon (noun, adjective, adverb or verb). Then he has to type the word in a canonical form (singular for nouns and adjectives, infinitive for verbs). Now the following components are launched.

Determination of word forms and phonological features

Some word forms can be calculated automatically, following the rules of word-formation in the concerned language. For others there are few possibilities to build a certain form – the user has to choose the right form among the possible ones. The third kind of forms are completely irregular.

After an analysis of the typed word form all other forms which are systematically derivable, are calculated by the system. When there are several concurrent paradigms, the system calculates the possible forms and proposes them to the user, who has to choose the right one. Only in the case of highly irregular paradigms the user has to type the right form (that means to enter it letter by letter).

For French and German nouns, verbs and adjectives, we have implemented efficient algorithms which allow the system to calculate a paradigm from a minimum of forms. This procedure isn't performed for adverbs, which only have one form.

For all French word forms we have to know if they begin with a vowel or a consonant, in order to permit the determination of the right article or pronoun form preceeding the word form. This task can be performed automatically for all word except for such beginning with an *h*. Here we produce two sequences, one with an elided, another with a non-elided form preceeding the word. The user has to choose the correct sequence.

Determination of syntactic features

The relevant syntactic features are depending on the syntactical category. For verbs we have to determine the number and the type of complements. This is done by giving a complement frame and one or several examples for each possible complement structure. The user has to select the right ones.

In the case of common nouns, the user has to determine complements and gender. The first task is performed similarily as in the case of verbs, for the second we propose a choice between several noun phrases with a different form of the determiner, which shows gender agreement with the noun in both French and German.

Proper nouns usually don't take a determiner. Their gender is asked from the user by means of the appropriate coreferent pronoun.

Conceptual features

The task of determining the conceptual features consists in finding the appropriate conceptual class for the arguments of a verb or the noun and the adjective.

In order to select the conceptual class we examine all the different categories included in the tree defining the concepts, beginning from the root. Each category is associated with a clear description, eventually including one or several examples. The user is asked to classify the verb argument, the noun, or the adjective by selecting one of the proposed categories. After one conceptual category has been fixed, a further specification is possible by choosing one of its subcategories. If no further specification is possible, the conceptual class of the item is defined by the chosen category.

Proper names belong to few different conceptual classes. Thus we have defined and implemented a reduced hierarchy of concepts including persons, animals and geographic entities.

Adverbs aren't classified conceptually so far.

5.3.3. Examples for dialogues with the user

The dialogues in the examples are translated into English, the character • indicates a choice to be performed by the user.

i) word form (French verb *jeter*):

```
Please select the right form:
•    je jette
•    je jète
```

ii) complement structure

```
Is this construction possible with the verb jeter?
il jette quelque chose
(comme par exemple : il mange quelque chose)
•    YES
•    NO
```

iii) Gender (German noun *Hund*)

```
Which article is the right one?
•    Der Hund
•    Die Hund
•    Das Hund
```

iv) Conceptual class (*jeter*)

```
X jete Y
In this construction, X is:
```

- a human being
- an animal
- an object
- something else

6. The pictographic system

6.1. Introduction

We developed a second application for handicapped children and teenagers who perfectly understand speech but can't perform nor write any sentences. Our application is based on the iconic code SODI-GRACH[14]. This code refers to about 200 verbs, 250 nouns, and 170 adjectives, adverbs, pronouns and expressions. An icon is associated with each of them. Our system allows users to communicate among one another and to non-handicapped people by means of icons.

Our idea is to apply the generation methods of the KOMBE system to such a symbol language in order to provide a way to communicate by means of images. A sentence will be a sequence of pictures instead of a sequence of words. As an additional feature we wanted the pictures to keep grammatical relations. This means that we use the syntactic rules of the language and a conceptional module to predict and restrain the possible combinations. The algorithms of the KOMBE system are used here without any changes.

We chose to operate with SODI-GRACH, which has rather realistic pictures and is therefore easier to understand. Other reasons which impelled us to take that code were the following facts:

- the symbols are simple and have been drawn by non-speaking people,
- the language is widely known in France, and used by different therapists.

6.2. The elements of the language

The pictographic lexicon includes up to 70 pictures, which were excerpted from the Sodi-Grach catalogue.

The syntactic rules of the language are described by a grammar. We used the same formalisms as for the KOMBE system for natural language (NL).

The conceptual model of the NL KOMBE system, which defines relations between notions, serves to restrain the composition of sentences. This model can also be used here because (nearly[15]) every symbol signifies – as the corre-

---

14 This code has been developed by *Groupement de Recherche pour l'Autonomie et la Communication des Handicapés*.

15 Exceptions are the symbols signifying grammatical functions as *plural*, for example.

sponding word – a concept. As a consequence, the constraints of the conceptual models are not influenced by the fact that we use symbols or words.

6.3. The algorithms

The algorithms of the KOMBE system are defined in a general manner so that they can also be applied to the composition of symbol sequences. This means that the system can propose sentence beginnings and subsequently their possible continuations, even in the case of a symbol language.

6.4. The presentation on the screen

The presentation on the screen follows in essence the same principles as we have shown for the natural language KOMBE system; in this case symbols are shown instead of words. However, a few new routines had to be integrated to get the graphic representation on the screen.

6.5. Applications of a symbol based communication system

A system which is based on symbols instead of words can be applied in several ways, specially if it will be used by children or by elderly people.

6.5.1. "Internal" communication

A classic application of the system is the communication with persons that know the code of the symbols such as therapists, teachers and parents of non-speaking children. In this case the KOMBE system can be seen as a highly elaborated communication tool adapted to the needs of the users.

6.5.2. Learning systems

Some handicapped children (as those suffering from cerebral palsy, for example) have severe problems when learning natural languages; a great number of them can only use symbolic languages. As already mentioned above, most symbol languages have been developed just for those children. As research has shown[16], a further, positive influence on the cognitive capacity of those children has been observed. Some of them are able to learn an alphabetic language after having learned and used a symbol language.

---

16 cf. E. COUNET et al. (1988: 6).

For this reason, it is very interesting to develop powerful computer assisted systems for the purpose of learning a symbol language. The majority of programs, however, only allow the children to acquire lexical items.

The KOMBE system considers two further levels of language, the syntactic rules and the conceptual model. Children using the system can learn to compose sentences that are syntactically correct and – possibly at a later stage – sentences that are conceptually valid ("make sense"), because the system only makes proposals meeting the required conditions.

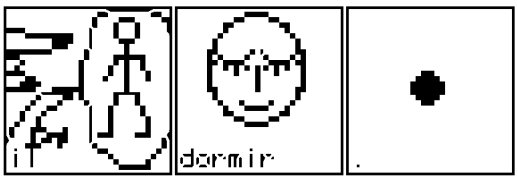
6.5.3. "External" communication

Users of a symbol language as their communication tool can only be understood by people knowing this special language. BLISS users, for example, can be understood by a small number of therapists and parents having learned the code; but they can hardly communicate with other people.

Our system translates a sequence of pictures in a sentence of correct French. This translation program is based upon the following property: Every syntactic structure of the symbol language can be related to a syntactic structure of the natural language (the first structure can be seen as a reduced version of the second one: gaps instead of articles, relative pronouns etc. and without respecting any agreement constraints). This second structure has been integrated as additional argument of the terms into the above mentioned grammar of the symbol language. The translation process consists in using these grammar rules only once, but the rules *list_of_leaves*[17] twice, for the first time to analyze the symbol sequence (the result is the first structure) and for the second time to generate the natural language sentence (or the set of sentences), using the second structure related to the first one.

Examples:

1.) For the symbol sequence



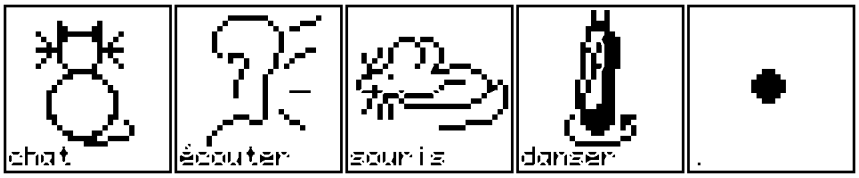
the result is the following natural language sentence:

```
il dort.
```

2.) For the symbol sequence

---

[17] cf. Deliverable 2.2 Algorithms.



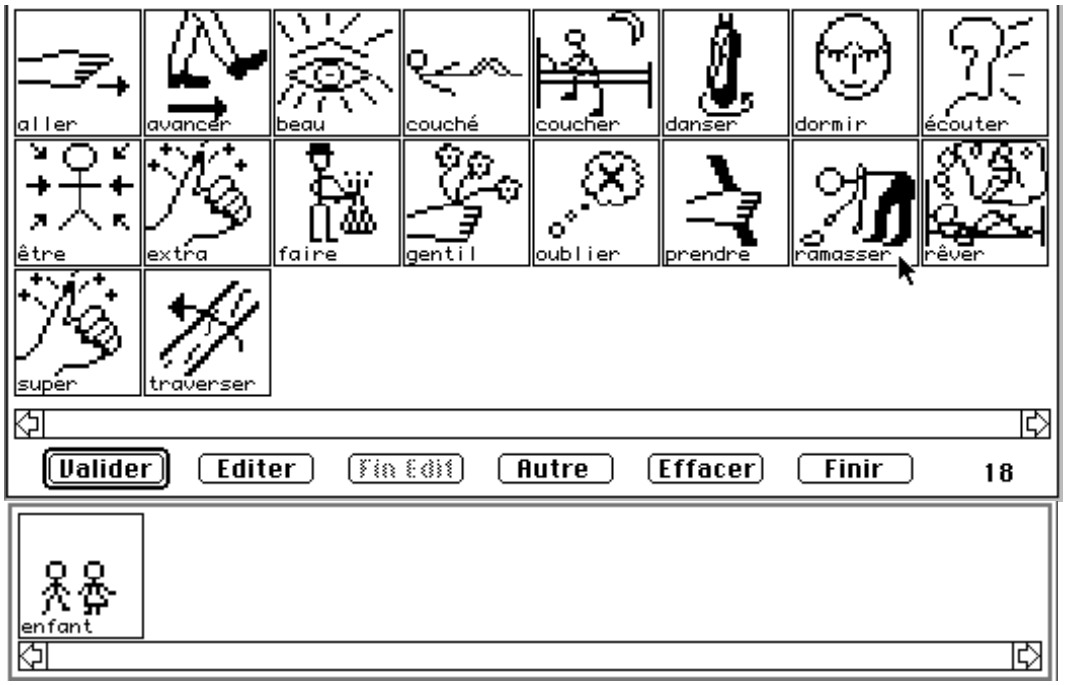
we obtain, among others, the following natural language sentences:

```
le chat écoute la souris qui danse.
le chat écoute les souris qui dansent.
le chat qui écoute la souris danse.
le chat qui écoute les souris danse.
les chats écoutent la souris qui danse.
les chats écoutent les souris qui dansent.
chat, écoutes la souris qui danse.
chat qui écoute la souris, danses.
chats, écoutez les souris qui dansent.
chats qui écoutent la souris, dansez.
chat, écoutons la souris qui danse.
chat qui écoute les souris, dansons.
chats, écoutons la souris qui danse.
```

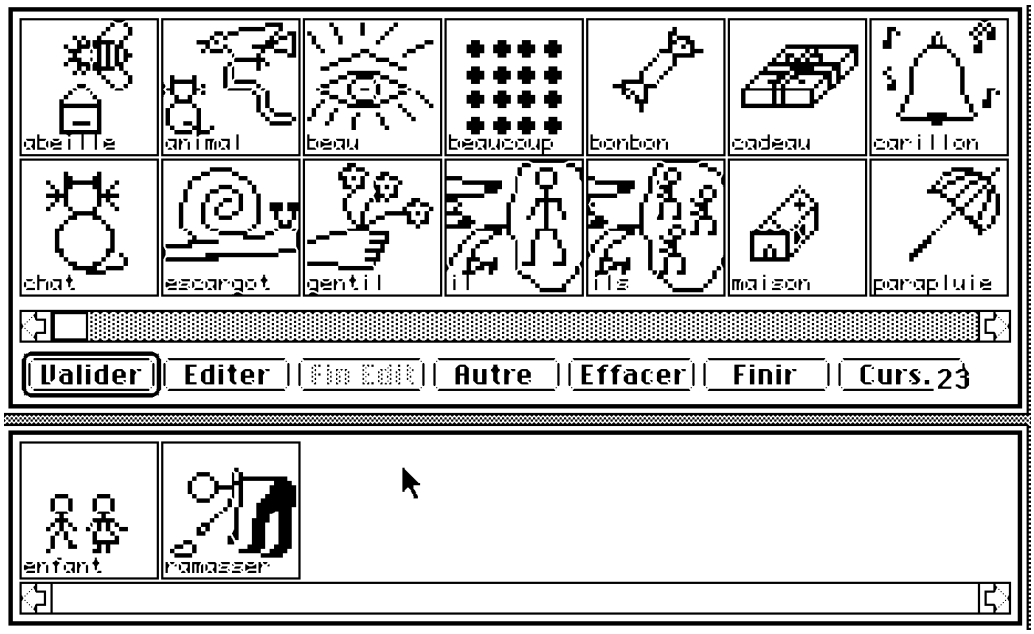
The result is a natural language sentence (or a set of sentences) which can be read by every speaker of this language; as it is written as a completely grammatical sentence it is ready for further treatment (voice synthesizer, text manipulation etc.).

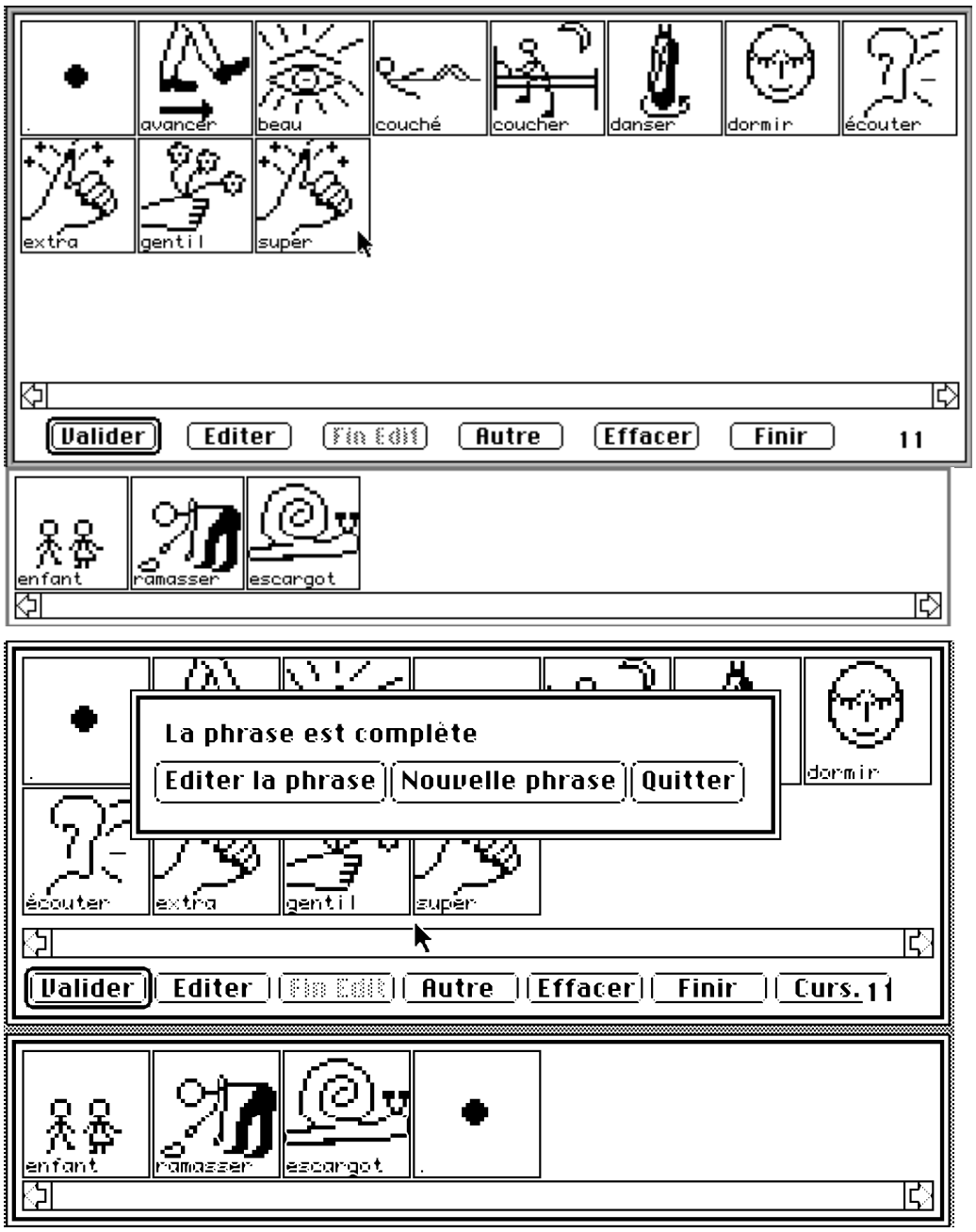
6.6. Example

The following screen outprints show an example of a complete message formulated in the SODI-GRACH-code, composed by a user:



Like in the NL system, users compose their message step by step, that means icon by icon. At each stage possible expected icons are generated by the system according to a grammar, a lexicon and a conceptual model reflecting the appropriate situation. For example, the user can compose the sentence:





Les enfants ramassent un escargot (Children pick a snail).

6.7. Conclusion

In addition to the interesting applications mentioned above, the development of the KOMBE symbol language system contributes to the clarity of use of these languages, because we have to elaborate explicit rules describing the knowledge of the languages, which will be available to all users of the language later on.

Bibliography

ARNOTT, John L., Jeffrey M. HANNAN, Robin J. WOODBURN (1993): *Linguistic Prediction for Disabled Users of Computer-Mediated Communication*, in: The Swedish Handicap Institute (ed.): Proceedings of the ECART2 Conference, Stockholm: Kommentus, section 11.1.

BOLC, Leonard (ed.) (1978): Natural Language Communication with Computers. Berlin: Springer.

BÜHLER, Christian, Helmut HECK (1992): CAi - *A Versatile Communication Aid for the Speech-Impaired*, in: Zagler, Wolfgang(ed.): Computers for Handicapped Persons. Proceedings of the 3rd International Conference, Vienna. Vienna/ Munich: R. Oldenbourg, pp. 88-96.

COLEMAN, E.(1965): *Learning of prose in four grammatical transformations.* In: Journal of Applied Psychology 4, 332 - 341.

COLMERAUER, Alain: *Metamorphosis grammars.* In: Bolc, L. (ed.) 1978, 133 - 189.

COUNET, E. et al. (1988): Communiquer et Apprendre par Pictogrammes. Edition scolaires Erasme; Namur (Bouge).

FUCHS, H.P. & G. SCHANK (eds.) (1975) (: *Texte gesprochener deutscher Standardsprache III. Alltagsgespräche* (=Heutiges Deutsch II,3), München.

GIANNESINI, F., H. KANOUI, R. PASERO & M. VAN CANEGHEM (1985): Prolog. Paris: InterÉditions, 1985.

GODBERT, Elisabeth, Robert PASERO & Paul SABATIER (1993): *Natural Language Interfaces: Specifying and Using Conceptual Constraints.* HCI'93, 5th International Conference on Human-Computer Interaction, Orlando (U.S.).

GUENTHNER, Franz, Karin KRÜGER-THIELMANN, Robert PASERO & Paul SABATIER (1992): *Communication Aids for ALS-Patients*, in: ZAGLER, Wolfgang(ed.): Computers for Handicapped Persons. Proceedings of the 3rd International Conference, Vienna. Vienna, Munich: R. Oldenbourg, pp. 303-307.

GUENTHNER, Franz, Karin KRÜGER-THIELMANN, Robert PASERO & Paul SABATIER (1993): *Communication Aids for Handicapped persons,* in: The Swedish Handicap Institute (ed): Proceedings of the ECART2 Conference, Stockholm. sect.1.4.

GUENTHNER, Franz, Karin KRÜGER-THIELMANN, Robert PASERO & Paul SABATIER (1993): *Guided Composition of Text Used in Communication Aids for Handicapped Persons,* in: Proceedings of the Sixth Biennal Conference of the International Society for Augmentative and Alternative Communication (ISAAC), Maastricht, October 1994, pp. 474-476.

JÄGER, Karl-Heinz (1976): Untersuchungen zur Klassifikation gesprochener deutscher Standardsprache. (= Heutiges Deutsch I/11). München: Hueber.

JÄGER, Karl-Heinz (1979): Texte gesprochener deutscher Standardsprache IV. "Beratungen und Dienstleistungsdialoge". (=Heutiges Deutsch II,4), München 1979. Freiburg: IDS, Forschungsstelle Freiburg.

KAMPHUIS, Harry et al. (1993): *A Comparison of English and Dutch PALs: Predictive Adaptive Lexica in two Languages*, in: The Swedish Handicap Institute (editor): Proceedings of the ECART2 Conference, Stockholm: Kommentus, section 1.1.

LANGER, Stefan (1993): Un système d'acquisition lexicale pour le projet KOMBE. Mémoire de DEA Informatique et Mathématiques, Option Language Naturel; Faculté des Sciences de Luminy, Université Aix-Marseille II.

LEHRNDORFER (forthcoming): Kontrolliertes Deutsch. Doctoral dissertation, University of Munich. (to appear at Tübingen: Narr).

RICHARDET, Nathalie (1992): Un système d'aide à la composition de phrases pour handicapés. Mémoire de DEA Informatique et Mathématiques, Option Language Naturel; Faculté des Sciences de Luminy, Université Aix-Marseille II.

TEIGELER, P. (1968): Verständlichkeit und Wirksamkeit von Sprache und Text. Karlsruhe: Nadolski.

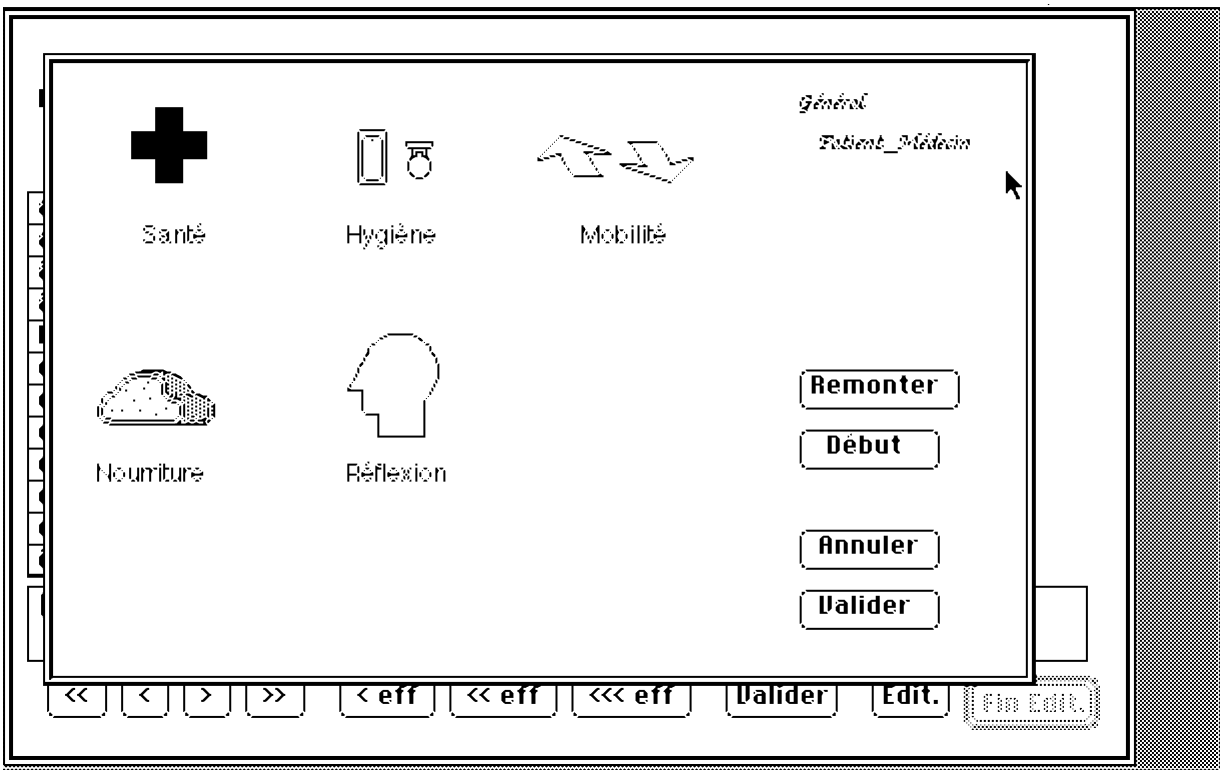
TYVAND, Steinar & Patrick DEMASCO (1993): *Syntax statistics in Word Prediction*, in: The Swedish Handicap Institute (ed.): Proceedings of the ECART2 Conference, Stockholm: Kommentus, section 11.1.

ANNEXES

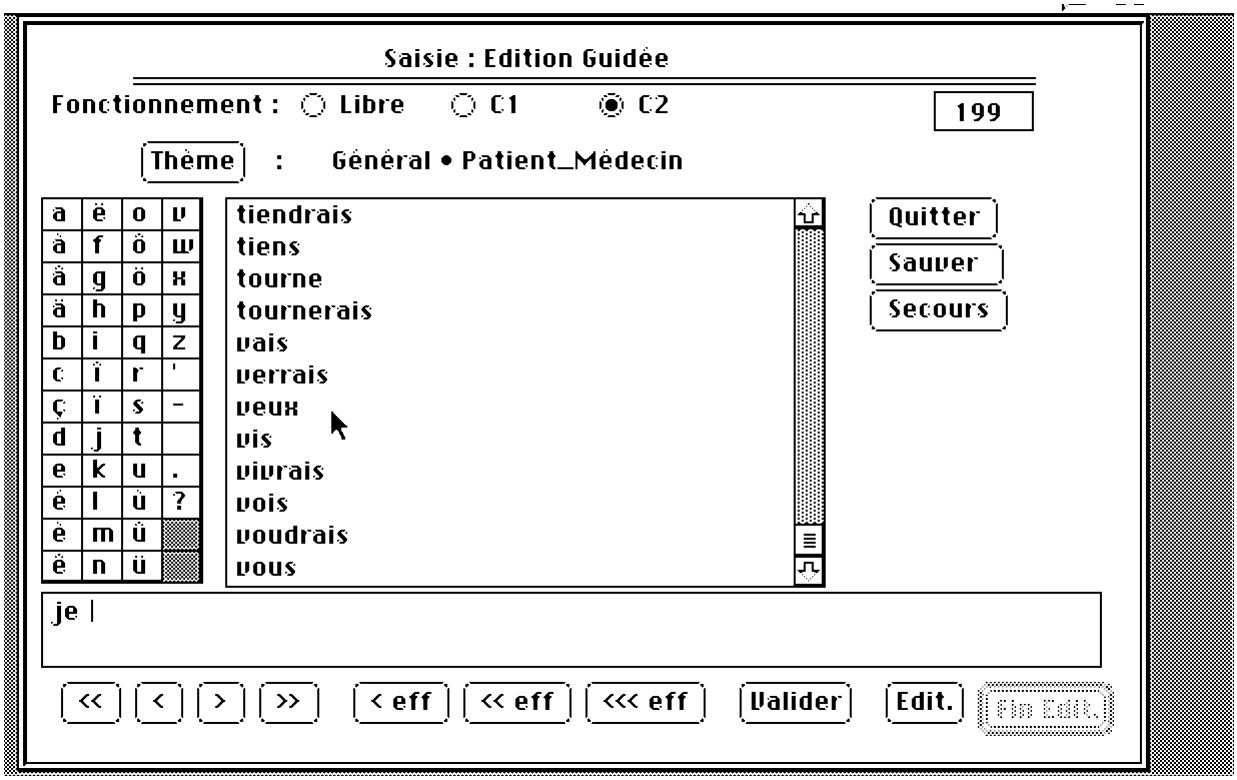
Annex A: Examples of guided composition of sentences

Three modes of composition ("Fonctionnement") are available:

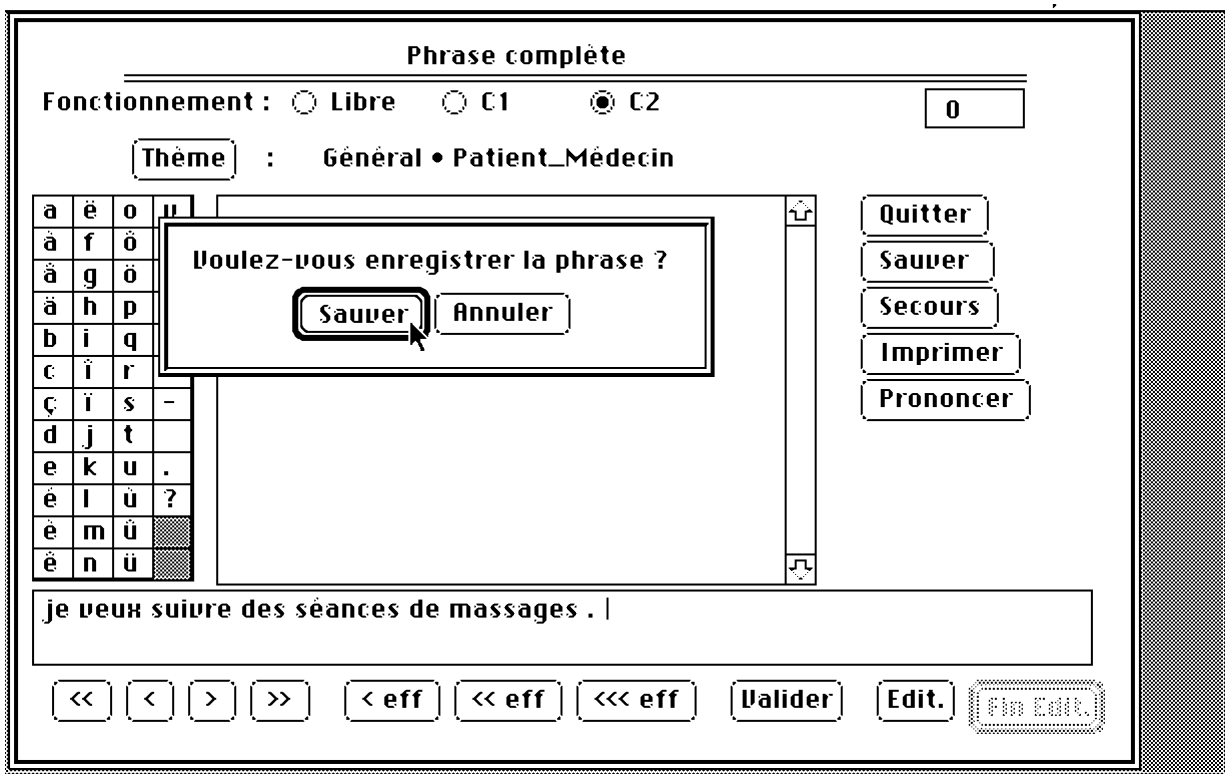
- "Libre" : lexical mode of composition (letter by letter or word by word);
- "C1" : syntactical mode of composition (by grammar rules) ;
- "C2" : conceptual mode of composition (by grammar rules and conceptual rules).



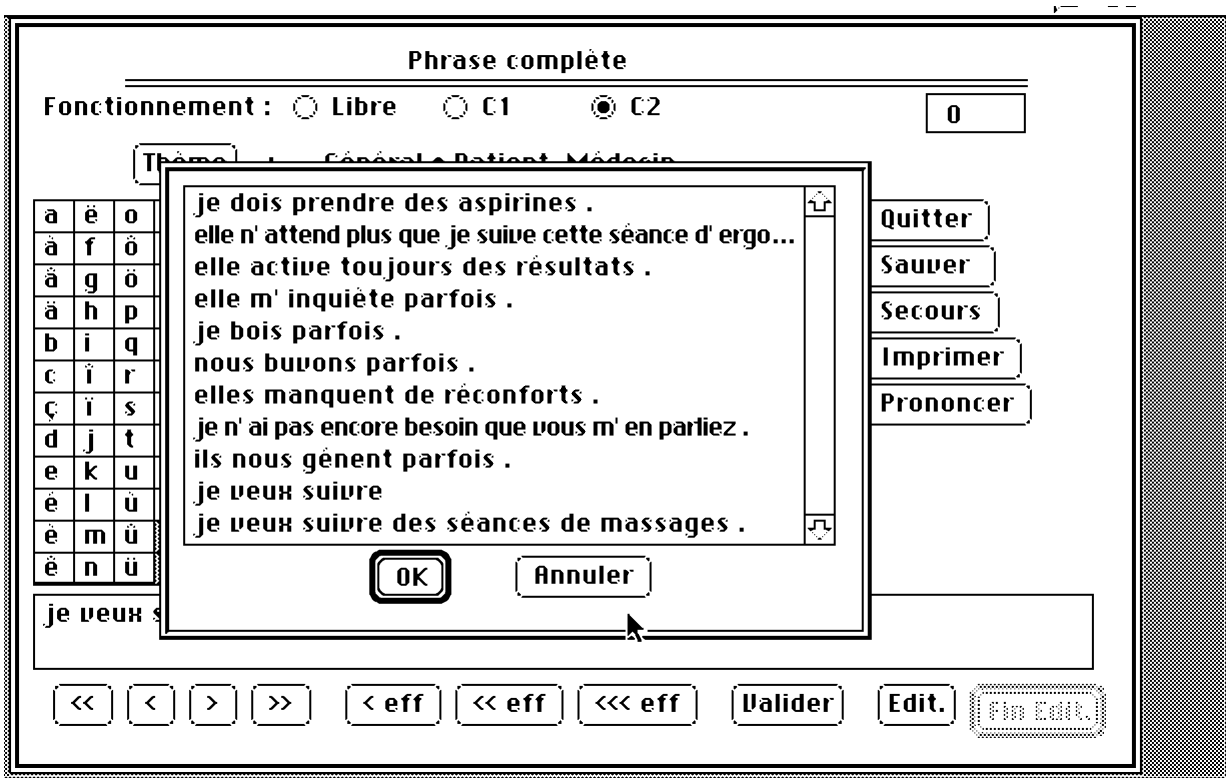
Screen 1: Selection of the conversation theme



Screen 2: Example of guided composition



Screen 3: A sentence is complete



Screen 4: Selecting from a list of saved sentences.

Annex B: German prototype

Table of ideas

The following table shows the ideas which can be expressed by means of the German system.[1]

Topic: Beschwerden – all.doctor.health.symptoms

Subjekt	Modalverben	Verb	Adverb / Negation	Komplement1	Komplement2
ich	können	aufstehen, schlafen	schlecht	–	–
ich	–	aufwachen	oft	–	–
ich	<modal>	sich ausruhen	(nicht) oft	–	–
<Krankheit>, <Schmerzen>	sollen, können, müssen	sich bessern		–	–
ich	müssen, sollen	sich schonen		–	–
<Arzt>	<modal>	untersuchen		<Mensch>, <Körperteile>	–
ich	können	atmen, (ein)schlafen, sprechen, schlucken	kaum, schlecht, (noch) gut, nicht	–	–
ich	können	atmen		durch: Nase, Mund	
ich	–	haben	nie, oft	<Schmerzen> Krämpfe, Zuckungen, Schweißausbrüche, Husten, Allergie, <Krankheit>, Beschwerden	–
ich	–	leiden	nie, oft	an: Krämpfe, Schwächeanfälle, Schweißausbrüche, Allergie, Depressionen	–

1 Words in brackets (<>) refer to the concepts which are defined in the conceptual model.

ich	–	bekommen	nie, oft	<Schmerzen> Krämpfe, Schwäche-anfälle, Schweiß-ausbrüche, (keine) Luft, <Krankheit_ kurz>	–
ich	–	sehen	nie, oft	Doppelbilder	–
ich	–	werden		müde	–
ich	–	sich fühlen		müde, schwach	–
es	–	gehen		mir	gut, schlecht
<Körperteil>	–	tun		mir	weh
<Schmerzen>, Krämpfe, Zuk-kungen, Schweiß-ausbrüche, Husten	–	nachlassen, zunehmen, abnehmen		–	–
<Schmerzen>, Krämpfe, Schwäche-anfälle, Zuckungen, Schweißaus-brüche, Husten	–	werden	nicht	schlimmer, häufiger, weniger	–
<Schmerzen>, Krämpfe, Schwäche-anfälle, Zuckungen, Schweißaus-brüche, Husten	–	sein	nicht	schlimm, stark	–
ich	müssen	husten, schwitzen, sich übergeben,	oft, kaum, selten	–	–
ich	–	sich verschlucken, frieren hinfallen	oft, kaum, selten	–	–
ich	wollen	abnehmen, zunehmen		–	–
es	–	sein	oft, nie	mir	schlecht, übel

Topic: Medikamente – all.doctor.health.medicine

Subjekt	Modalverben	Verb	Adverb / Negation	Komplement1	Komplement2
Arzt	müssen	erhöhen, herabsetzen		Dosis	–
ich	können, wollen, müssen	nehmen, schlucken		<Medikament>	–
ich	können, wollen, müssen	bekommen		<Medikament>, Infusion	–
jemand	können	verschreiben		<Medikament>	mir
ich	müssen	bezahlen	selbst	<Medikament>	–
<Medikament>, <Behandlung>	–	wirken	gut, schlecht	–	–
<Medikament>	–	haben		Nebenwirkungen	–

Topic: Therapie – all.doctor.health.therapy

Subjekt	Modalverben	Verb	Adverb / Negation	Komplement1	Komplement2
ich	sollen, müssen	machen		<Therapie>	–
ich	müssen	bezahlen	selbst	<Therapie>	–

Topic: Ernährung – all.doctor.food

Subjekt	Modalverben	Verb	Adverb / Negation	Komplement1	Komplement2
<Mahlzeit>, Essen	–	sein	schwierigproblemlos	–	–
ich	können	abbeißen	nicht, schlecht, gut	von: <Essen_fest>	–
<Nahrung>	–	haben, enthalten		<Inhaltsstoff> <Zutaten>	–
<Nahrung>, <Inhaltsstoff>, <Zutaten>	–	reizen		mich	–
ich	–	essen	oft, nie	<Essen>	–
ich	–	trinken	oft, nie	<Trinken>	–
ich	–	mögen	nicht	<Essen>, <Trinken>	–

ich	können	kauen	nicht, schlecht, gut	<Essen_fest>	–
ich	können	schlucken	nicht, schlecht, gut	<Essen>, <Trinken>	–
ich	können, wollen	probieren		<Essen>, <Trinken>	–

Topic: Zubereitung – all.doctor.food.cook

Subjekt	Modal-verben	Verb	Adverb / Negation	Komplement1	Komplement2
ich, <Pflege-person>	–	kochen	oft, immer	<Essen_flüssig>, <Essen_weich>, <Essen_fest>, <Getränk_ warm>	–
ich, <Pflege-person>	–	backen	oft, immer	<Essen_fest_ backen>	–
ich, <Pflege-person>	–	braten	oft, immer	<Essen_fest_ nback>	–
<Pflegeperson>	–	pürieren, schneiden, zerkleinern		<Essen_fest>	–
<Pflegeperson>	–	schneiden		<Essen_fest>	in: Stücke
ich, wir, <Pflegeperson>	–	haben		Zauberstab, Pürierstab, Mixer	–

Topic: Mahlzeiten– all.doctor.food.meals

Subjekt	Modalverben	Verb	Adverb / Negation	Komplement1	Komple-ment2
<Pflegeperson>	müssen	füttern	oft, immer	mich	–
ich	können	halten	nicht (mehr)	<Besteck>, Glas, Tasse	–
ich	–	haben		Appetit, Hunger, Durst	–
<Essen>	–	schmecken	nicht, gut, immer	mir	–

Topic: Körperpflege – all.doctor.hygiene

Subjekt	Modalverben	Verb	Adverb / Negation	Komplement1	Komplement2
ich	können	schneiden	nicht, nicht mehr	Fingernägel, Fußnägel	mir
<Pflegeperson>	–	rasieren, frisieren, kämmen, pflegen, schminken		mich	–

Topic: Kleidung – all.doctor.clothes

Subjekt	Modalverben	Verb	Adverb / Negation	Komplement1	Komplement2
ich	können	anziehen, ausziehen	nicht, gut/ schlecht, kaum (noch), nicht mehr, selbst	<Oberbekleidung>, Schuhe, Windel, Strümpfe, mich	–
<Oberbekleidung>	–	passen	nicht, nicht mehr	mir	–

Topic: Waschen/Toilette– toilette

Subjekt	Modalverben	Verb	Adverb / Negation	Komplement1	Komplement2
ich	können	baden, duschen, waschen	gut/ schlecht, nicht, nicht mehr, selbst	mich	–
ich	können	putzen	gut/ schlecht, nicht, nicht mehr, selbst	Zähne	–
ich	können	waschen	gut/ schlecht, nicht, nicht mehr, selbst	<Körperteile, waschbar>	–
ich	können	gehen	nicht, nicht mehr, alleine	zur Toilette	–

Topic: Beweglichkeit – all.doctor.mobility

Subjekt	Modalverben	Verb	Adverb / Negation	Komplement1	Komplement2
Schritt, Bewegung,	–	anstrengen		mich	
Schritt, Bewegung	–	tun		mir	weh
Urlaub, Reise, Kur	–	tun	gut, nicht gut	mir	–
ich	–	stürzen, stolpern, fallen	oft, selten, kaum	–	–
ich	<modal>	gehen, laufen, reisen, schwimmen, sich bewegen, stehen, (auf)stehen, sich hinsetzen, sich hinlegen, sitzen, sich umdrehen	gut/ schlecht, nicht, nicht mehr, kaum	–	–
ich	können	bewegen	gut, schlecht	<bewegl Körperteil>	–
ich	–	brauchen	oft, selten, kaum, schon lange	Rollstuhl, Krücken, Korsett	–
ich	können	beugen, bewegen	gut/ schlecht, nicht, nicht mehr, kaum	<Körperteile äußere>, Kiefer	–
ich	wollen	fahren		in: Urlaub	–

Topic: all.doctor.psycho

Subjekt	Modalverben	Verb	Adverb/ Negation	Komplement1	Komplement2
ich, <Pflegeperson>	<modal>	begleiten		<Pflegeperson>, mich	–
ich, <Arzt>, <Pflegeperson>	<modal>	besprechen		<Behandlung>	
ich, <Arzt>, <Pflegeperson>	<modal>	besprechen		<Behandlung>	mit: mir, <Arzt>, <Pflegeperson>
ich, <Pflegeperson>	<modal>	bleiben		da, hier, im Raum, etc.	

<Pflegeperson>	<modal>	bringen		<Gegenstand>, <Medikament>	–
ich	<modal>	denken		an: Tod, <Mensch>	–
ich	wollen	fragen		nach: <Abstr_Objekt>	–
ich	–	haben	oft	Angst	
ich	wollen, können	sprechen		mit: <Mensch>	–
ich	wollen	wissen		Wahrheit	–

Exemples of German sentences

Every idea cited in the table above can be formulated by a set of sentences. For the purpose of illustration we give a set of German sentences for one of these ideas and the translations into English. We have chosen the following idea:

ich	können, wollen, müssen	nehmen, schlucken		<Medikament>	–

The corresponding set contains the following sentences:

Ich nehme die Tabletten.	*(I take the pills9*
Ich nehme die Tabletten nicht.	*(I don't take the pills)*
Ich habe die Tabletten genommen.	*(I have taken the pills)*
Ich habe die Tabletten nicht genommen.	*(I haven't taken the pills)*
Ich will die Tabletten nicht nehmen.	*(I don't want to take the pills)*
Ich kann die Tabletten nicht schlucken.	*(I can't swallow the pills)*
Muß ich die Tabletten nehmen?	*(Shall/must I take the pills?)*
Muß ich die Tabletten schlucken?	*(Shall/must I swallow the pills?)*
Welche Tabletten muß ich nehmen?	*(Which pills shall I take?)*
Ich nehmen die Tabletten, wenn die Schmerzen zunehmen.	*(I/ I'll take the pills if pain is increasing)*
Wenn die Schmerzen zunehmen, nehme ich die Tabletten.	*(If pain is increasing I/ I'll take the pills)*
Ich nehmen die Tabletten, weil die Schmerzen zunehmen.	*(I take the pills because pain is increasing)*
Ich nehmen die Tabletten, weil die Schmerzen zugenommen haben.	*(I take the pills because pain have increased)*
Weil die Schmerzen zugenommen haben, nehme ich die Tabletten.	*(As pain have increased I take the pills)*

The conceptual module

```
/* Definition of the hierarchy */
domaine;
racine(alles);
alles.Objekt.Abstr_Objekt.Begriff.Eigenschaft.modal_nmod;
Objekt.Belebt.Unbelebt.Teil;
Belebt.Pflanze.Tier_mensch;
Tier_mensch.Tier.Mensch.Mikro_org;
Mensch.ou(Mann.Frau,Erwachsen.Kind).Arzt;

Abstr_Objekt.Gefühl.Gedanke.Tatsache.Dosis.Problem.Ort.
  Ereignis.Tätigkeit.Mass;
Gefühl.Leiden.Freude.Hoffnung.Kraft.Essen_Gefühl;
Leiden.Krankheit.Schmerzen.Symptom.Verletzung.Doppelbilder;
Symptom.Indikator_Ges.Nebenwirkung;
Krankheit.Krankheit_kurz.Krankheit_dauernd;
Krankheit_kurz.Krankheit_kurz_wiederhol;
Essen_Gefühl.Hunger.Appetit.Durst;

Tätigkeit.Arbeit.Gespräch.Behandlung.Reise.Bewegung;
Behandlung.Therapie.Therapie_nselbst.Therapie_arzt;
Objekt.Nahrung.Mittel.Gegenstand.Verkehrsmittel.Schriftliche. toff;
Reise.Reise_an_sich.Urlaub;
Nahrung.Mahlzeit.Essen.Ingredenzien.Trinken;

Trinken.Getränk_kalt.Getränk_warm;

/*fest:schneidbar,weich - nicht schneidbar, nicht alle festen kann man
  backen*/
Essen.Essen_fest.Essen_weich.Essen_flüssig;
Essen_fest.Essen_fest_backen.Essen_fest_nback;

Ingredenzien.Zutaten.Inhaltsstoffe.Aufstrich;

Mittel.Medikament.Seife.Verband;

Medikament.Med_inn.Med_äuss;

Gegenstand.Küchenggst.Möbel.Einrichtung.Bad_Toil.Schmuck.
  Kleidung.Hilfsmittel.Gebäude.Gebäudeteil;
Küchenggst.Geschirr.Besteck.Zerkleinerungsgerät;
/* nur bestimmtes Geschirr wird zum Mund bewegt */
Geschirr.Geschirr_zum_Mund.Geschirr_Tisch;
Kleidung.Kleidung_normal.Kleidung_spezial;
Möbel.Sitzmöbel;

Bad_Toil.Spülung.Wasserhahn;
```

```
Gebäude.Haus.Wirtschaft.Klinik;

Gebäudeteil.Treppe.Wohnung.Zimmer.Toilette;

Einrichtung.Uhr;

Verkehrsmittel.Fahrzeug.Flugzeug;

Schriftliches.Rechnung.Speisekarte;

Stoff.Luft.Geld;

Teil.Teile.Sekretion.Körperteil;
Körperteil.Kt_innere.Muskel.Kt_äussere.Prothese;
Kt_äussere.Kt_bew_putz.Kt_bew_wasch.Kt_unb_putz.Kt_unb_wasch.Kt_andere;
Kt_bew_wasch.Arm.Knie;
Kt_unb_wasch.Po;
Kt_unb_putz.Nase.Nagel;
Kt_andere.Auge.Lid.Mund.Lippe.Kt_innere_bew;

Prothese.Gebiss;

Sekretion.Schweiss.Speichel.Träne;

Mass.Mengenmass.Zeitmass.Gewicht;

Mengenmass.Menge;

Zeitmass.Moment.Zeitinterv.Zeitpunkt;
Zeitinterv.Tag.Woche.Monat.Jahr.Stunde.Minute.Sekunde;
Zeitpunkt.Wochentag.Jahresmonat.Jahreszeit.Tageszeit;

Eigenschaft.weh.übel.qualitativ.frequentiv.methode.zustand_änderung;

qualitativ.gut.schlecht;

/*fuer die Uebereinstimmung mit dem Modalverb*/
modal_nmod.modal.nmod;
modal.mod_anweisung.mod_pers;
mod_pers.kann.will.moechte;
mod_anweisung.darf.kann.muss.soll;

/* Definition der Relationen */
contraintes;

/* Hilfs- und Modalverben */
haben(alles,alles);
können(kann,alles);
```

```
können(kann,alles);
müssen(muss,alles);
sein(alles,alles);
sollen(soll,alles);
werden(alles,alles);
wollen(will,Leiden);
wollen(will,Mensch);

/* Vollverben */

abbeißen(kann,Mensch,Essen_fest);
abbeißen(kann,Mensch,qualitativ);
abnehmen(modal,Mensch);
abnehmen(nmod,Krankheit_kurz_wiederhol);
abnehmen(nmod,Schmerzen);
anstrengen(modal,Bewegung,Mensch);
anstrengen(modal,Mensch);
anziehen(kann,Mensch,Mensch,qualitativ);
anziehen(mod_pers,Mensch,Kleidung);
atmen(kann,Mensch,Mund);
atmen(kann,Mensch,Nase);
atmen(kann,Mensch,qualitativ);
aufstehen(kann,Mensch,qualitativ);
aufwachen(nmod,Mensch);
ausruhen(modal,Mensch);
ausziehen(kann,Mensch,Mensch,qualitativ);
ausziehen(mod_pers,Mensch,Kleidung);
baden(modal,Mensch,Mensch);
besprechen(modal,Mensch,Behandlung);
besprechen(modal,Mensch,Problem);
bessern(kann,Leiden);
bessern(muss,Leiden);
bessern(soll,Leiden);
beugen(modal,Mensch,Arm);
beugen(modal,Mensch,Knie);
bewegen(modal,Mensch,Kt_bew_putz);
bewegen(modal,Mensch,Kt_bew_putz,qualitativ);
bewegen(modal,Mensch,Kt_bew_wasch);
bewegen(modal,Mensch,Kt_bew_wasch,qualitativ);
bewegen(modal,Mensch,Kt_innere_bew);
bewegen(modal,Mensch,Kt_innere_bew,qualitativ);
bewegen(modal,Mensch,qualitativ);
bezahlen(mod_anweisung,Mensch,Rechnung);
bezahlen(muss,Mensch,Medikament);
bleiben(modal,Mensch,Ort);
brauchen(nmod,Mensch,Hilfsmittel);
brauchen(nmod,Mensch,Kleidung_spezial);
brauchen(nmod,Mensch,Medikament);
```

```
denken(modal,Mensch,Ereignis);
duschen(modal,Mensch,Mensch);
einnehmen(modal,Mensch,Med_inn);
erhöhen(kann,Mensch,Dosis);
essen(modal,Mensch,Essen);
essen(modal,Mensch,qualitativ);
fragen(modal,Mensch,Mensch,Abstr_Objekt);
frieren(nmod,Mensch);
frisieren(kann,Mensch,Mensch);
füttern(muss,Mensch,Mensch);
geben(modal,Mensch,Medikament,Mensch);
geben(modal,Mensch,Nahrung,Mensch);
gehen(kann,Mensch,Toilette);
gehen(modal,Mensch,qualitativ);
gehen(mod_anweisung,Mensch,Mensch);
gehen(mod_anweisung,Mensch,Therapie);
gehen(nmod,alles,Mensch,qualitativ);
genießen(kann,Mensch,Nahrung);
haben(kann,Medikament,Nebenwirkung);
haben(nmod,Mensch,Belebt);
haben(nmod,Mensch,Essen_Gefühl);
haben(nmod,Mensch,Leiden);
haben(nmod,Mensch,Zerkleinerungsgerät);
haben(nmod,Nahrung,Inhaltsstoffe);
haben(nmod,Nahrung,Zutaten);
heilen(kann,Arzt,Mensch);
heilen(kann,Arzt,Mensch,Krankheit);
herabsetzen(kann,Mensch,Dosis);
hinfallen(nmod,Mensch);
hinlegen(modal,Mensch);
hinsetzen(modal,Mensch);
hoffen(nmod,Mensch);
hoffen(nmod,Mensch,Ereignis);
husten(muss,Mensch);
kämmen(kann,Mensch,Mensch);
kauen(modal,Mensch,Essen_fest);
kauen(modal,Mensch,qualitativ);
laufen(modal,Mensch,qualitativ);
machen(modal,Mensch,Therapie);
nachlassen(nmod,Krankheit_kurz_wiederhol);
nachlassen(nmod,Mensch);
nachlassen(nmod,Schmerzen);
nehmen(modal,Mensch,Medikament);
passen(nmod,Kleidung,Mensch);
pflegen(nmod,Mensch,Mensch);
probieren(modal,Mensch,Medikament);
probieren(modal,Mensch,Nahrung);
putzen(kann,Mensch,Kt_bew_putz);
```

```
putzen(kann,Mensch,Kt_unb_putz);
reichen(modal,Medikament,Mensch);
reichen(modal,Nahrung,Mensch);
reisen(modal,Mensch);
reizen(modal,Inhaltsstoffe,Mensch);
reizen(modal,Nahrung,Mensch);
riechen(kann,Mensch,Nahrung);
riechen(kann,Mensch,qualitativ);
riechen(nmod,Nahrung,qualitativ);
schlafen(kann,Mensch,qualitativ);
schlucken(modal,Mensch,Medikament);
schlucken(modal,Mensch,Nahrung);
schlucken(modal,Mensch,qualitativ);
schmecken(nmod,Medikament,Mensch);
schmecken(nmod,Medikament,qualitativ);
schmecken(nmod,Nahrung,Mensch);
schmecken(nmod,Nahrung,qualitativ);
schneiden(modal,Mensch,Essen_fest);
schneiden(modal,Mensch,Nagel);
schonen(modal,Mensch);
schwimmen(modal,Mensch,qualitativ);
schwitzen(nmod,Mensch);
sein(modal,alles,alles);
sein(nmod,alles,Mensch,qualitativ);
sitzen(kann,Mensch,qualitativ);
sprechen(modal,Mensch,qualitativ);
sprechen(mod_pers,Mensch,Mensch);
stehen(modal,Mensch,qualitativ);
stolpern(will,Mensch);
trinken(modal,Mensch,qualitativ);
trinken(modal,Mensch,Trinken);
tun(nmod,Bewegung,weh);
tun(nmod,Körperteil,weh);
tun(nmod,Tätigkeit,gut);
tun(nmod,Therapie,gut);
tun(nmod,Therapie,weh);
tun(nmod,Therapie_nselbst,gut);
tun(nmod,Therapie_nselbst,weh);
übergeben(muss,Mensch);
umdrehen(kann,Mensch,Mensch,qualitativ);
untersuchen(modal,Arzt,Mensch);
verschlucken(nmod,Mensch);
verschreiben(mod_pers,Arzt,Medikament,Mensch);
verschreiben(mod_pers,Arzt,Therapie,Mensch);
verschreiben(mod_pers,Arzt,Therapie_nselbst,Mensch);
waschen(kann,Mensch,Kt_bew_wasch);
waschen(kann,Mensch,Kt_unb_wasch);
waschen(kann,Mensch,Mensch,qualitativ);
```

```
waschen(modal,Mensch,Mensch);
werden(nmod,Leiden,zustand_änderung);
werden(nmod,Mensch,Mensch);
wirken(nmod,Medikament,qualitativ);
wirken(nmod,Therapie,qualitativ);
wirken(nmod,Therapie_nselbst,qualitativ);
wissen(will,Mensch,Tatsache);
zunehmen(modal,Mensch);
zunehmen(nmod,Krankheit_kurz_wiederhol);
zunehmen(nmod,Schmerzen);

/*Nomen

Pronomen
Personalpronomina: */

ich(Mensch);
du(Mensch);
er(alles);
sie(alles);
es(alles);
wir(Mensch);
ihr(Mensch);

/* Reflexivpronomina: wie oben definiert
ich(Mensch);
du(Mensch);
wir(Mensch);
ihr(Mensch);
sie(alles); */

/* Indefinite Pronomina: */
niemand(Mensch);
nichts(alles);
jemand(Mensch);
alle(alles);

/* Substantive */
ist_Abendessen(Mahlzeit);
ist_Abführmittel(Med_inn);
ist_Akupunktur(Therapie);
ist_Alkohol(Inhaltsstoffe);
ist_Alkohol(Inhaltsstoffe);
ist_Alkohol(Trinken);
ist_Alkohol(Trinken);
ist_Angst(Leiden);
ist_Arzt(Mensch);
ist_Banane(Essen);
```

```
ist_Beschwerden(Symptom);
ist_Besserung(Ereignis);
ist_Brei(Essen_weich);
ist_Brot(Essen_fest_backen);
ist_Brust(Kt_unb_wasch);
ist_Cortison(Med_inn);
ist_Daumen(Kt_bew_putz);
ist_Depression(Krankheit_dauernd);
ist_Depression(Krankheit_dauernd);
ist_Doktor(Mensch);
ist_Doppelbilder(Doppelbilder);
ist_Dosis(Dosis);
ist_Durst(Durst);
ist_Essen(Essen);
ist_Fieber(Krankheit_kurz);
ist_Finger(Kt_bew_putz);
ist_Fingernägel(Nagel);
ist_Fisch(Essen_fest_backen);
ist_Fleisch(Essen_fest_backen);
ist_Fresubin(Essen_weich);
ist_Freund(Mensch);
ist_Frischzellen(Med_inn);
ist_Frühstück(Mahlzeit);
ist_Fuß(Kt_bew_wasch);
ist_Fußnägel(Nagel);
ist_Gabel(Besteck);
ist_Gebiß(Gebiss);
ist_Gelenk(Kt_innere);
ist_Gemüse(Essen_fest_nback);
ist_Geschirr(Geschirr);
ist_Gesicht(Kt_unb_wasch);
ist_Gespräch(Gespräch);
ist_Getränk(Trinken);
ist_Glas(Geschirr_zum_Mund);
ist_Glied(Kt_bew_wasch);
ist_Hals(Kt_bew_wasch);
ist_Hand(Kt_bew_wasch);
ist_Haus(Haus);
ist_Herz(Kt_innere);
ist_Hitzewallungen(Symptom);
ist_Hoffnung(Hoffnung);
ist_Hunger(Hunger);
ist_Husten(Krankheit);
ist_Infusion(Therapie_nselbst);
ist_Kaffee(Getränk_warm);
ist_Kalzium(Med_inn);
ist_Kapsel(Med_inn);
ist_Kaumuskel(Muskel);
```

```
ist_Kehlkopf(Kt_innere);
ist_Kiefer(Kt_innere_bew);
ist_Klinik(Klinik);
ist_Knie(Kt_bew_wasch);
ist_Knochen(Kt_innere);
ist_Kopf(Kt_bew_wasch);
ist_Kopfschmerzen(Schmerzen);
ist_Korsett(Kleidung_spezial);
ist_Kraft(Kraft);
ist_Krampf(Krankheit_kurz_wiederhol);
ist_Krankenhilfe(Mensch);
ist_Krankheit(Krankheit);
ist_Krücke(Hilfsmittel);
ist_Kuchen(Essen_fest_backen);
ist_Lid(Lid);
ist_Lippe(Lippe);
ist_Logopäde(Mensch);
ist_Logopädie(Therapie);
ist_Luft(Luft);
ist_Mahlzeit(Mahlzeit);
ist_Medikament(Medikament);
ist_Menge(Menge);
ist_Messer(Besteck);
ist_Mittagessen(Mahlzeit);
ist_Mittel(Medikament);
ist_Mund(Mund);
ist_Muskel(Muskel);
ist_Nacken(Kt_unb_wasch);
ist_Nadel(Gegenstand);
ist_Nagel(Nagel);
ist_Nahrungsmittel(Essen);
ist_Nase(Nase);
ist_Nebenwirkung(Nebenwirkung);
ist_Nerv(Kt_innere);
ist_Nervenwasserpunktion(Therapie_arzt);
ist_Niere(Kt_innere);
ist_Nudel(Essen_fest);
ist_Oberkörper(Kt_unb_wasch);
ist_Obst(Essen_fest_nback);
ist_Ohr(Kt_unb_putz);
ist_Operation(Therapie_arzt);
ist_Orthopäde(Mensch);
ist_Patient(Mensch);
ist_Pille(Med_inn);
ist_Po(Po);
ist_Präparat(Medikament);
ist_Praxis(Zimmer);
ist_Problem(Problem);
```

```
ist_Pudding(Essen_weich);
ist_Puder(Med_äuss);
ist_Puls(Indikator_Ges);
ist_Rechnung(Rechnung);
ist_Rehabilitation(Therapie);
ist_Rollstuhl(Hilfsmittel);
ist_Röntgen(Behandlung);
ist_Röntgenbild(Gegenstand);
ist_Rücken(Kt_unb_wasch);
ist_Rückenmark(Kt_innere);
ist_Rückenmarkskanal(Kt_innere);
ist_Rückenmuskulatur(Muskel);
ist_Rückenschmerzen(Schmerzen);
ist_Salbe(Med_äuss);
ist_Salz(Zutaten);
ist_Schluckbeschwerden(Symptom);
ist_Schmerzen(Schmerzen);
ist_Schwäche(Symptom);
ist_Schwächeanfälle(Krankheit_kurz_wiederhol);
ist_Schweiß(Schweiss);
ist_Schweißausbruch(Krankheit_kurz_wiederhol);
ist_Schwindel(Symptom);
ist_Soße(Essen_flüssig);
ist_Spasmus(Symptom);
ist_Speichel(Speichel);
ist_Speichelfluß(Symptom);
ist_Spray(Med_äuss);
ist_Spritze(Gegenstand);
ist_Stoffwechsel(Indikator_Ges);
ist_Stuhlgang(Indikator_Ges);
ist_Suppe(Essen_flüssig);
ist_Symptom(Symptom);
ist_Tablette(Med_inn);
ist_Tasse(Geschirr_zum_Mund);
ist_Tee(Getränk_warm);
ist_Tod(Ereignis);
ist_Treppe(Treppe);
ist_Tropfen(Med_inn);
ist_Urlaub(Urlaub);
ist_Verband(Verband);
ist_Verstopfung(Krankheit);
ist_Wahrheit(Tatsache);
ist_Wasser(Trinken);
ist_Wein(Getränk_kalt);
ist_Wirbel(Kt_innere);
ist_Zahn(Kt_unb_putz);
ist_Zuckungen(Krankheit_kurz_wiederhol);
```

```
/* Eigennamen */
Klara'(Mensch);
Karl'(Mensch);

/*Adjektive*/

ist_gekocht(Essen);
ist_häufiger(zustand_änderung);
ist_heiß(Objekt);
ist_kalt(Objekt);
ist_lecker(Essen);
ist_problemlos(Tätigkeit);
ist_roh(Essen);
ist_satt(Mensch);
ist_schlimmer(zustand_änderung);
ist_schön(alles);
ist_schwierig(Tätigkeit);
ist_stark(Krankheit_kurz_wiederhol);
ist_stark(Schmerzen);
ist_stärker(zustand_änderung);
ist_warm(Objekt);
ist_weniger(zustand_änderung);

/* Adverbien */

tut_gut(gut);
tut_hier(Ort);
tut_nie(frequentiv);
tut_oft(frequentiv);
tut_schlecht(schlecht);
tut_selbst(methode);
tut_übel(übel);
tut_weh(weh);

fin;
```

German dictionary

Modal verbs (7)
wollen
sollen
können
müssen
haben
sein
werden

Full verbs (83)
abbeißen
abnehmen
anstrengen
anziehen
atmen
aufstehen
aufwachen
ausruhen
ausziehen
baden
bekommen
besprechen
bessern
beugen
bewegen
bezahlen
bleiben
brauchen
denken
duschen
einnehmen
erhöhen
essen
fragen
frieren
frisieren
füttern
geben
gehen
genießen
haben
heilen
herabsetzen
hinfallen
hinlegen
hinsetzen
hoffen
husten
kämmen
kauen
kochen
laufen
machen
mixen
nachlassen
nehmen
passen
pflegen
probieren
pürieren
putzen
rasieren
reichen
reisen
reizen
riechen
schlafen
schlucken
schmecken
schmerzen
schminken
schneiden
schonen
schwimmen
schwitzen
sein
sitzen
sprechen
stehen
stolpern
trinken
tun
übergeben
umdrehen
untersuchen
verschlucken
verschreiben
waschen
werden
wirken
wissen
zerkleinern
zunehmen

Common nouns (158)
Abendessen
Abführmittel
Akupunktur
Alkohol
Angst
Arzt
Banane
Beschwerden
Besserung
Brei
Brot
Bruder
Brust
Cortison
Daumen
Depression
Depressionen
Doktor
Doppelbilder
Dosis
Durst
Essen
Fieber
Finger
Fingernägel
Fisch
Fleisch
Frau
Fresubin
Freund
Freundin
Frischzellen
Frühstück
Fuß
Fußnägel
Gabel
Gebiß
Gelenk
Gemüse
Geschirr
Gesicht
Gespräch
Getränk
Glas
Glied
Hals
Hand
Haus
Herz
Hitzewallungen
Hoffnung
Hunger
Husten
Infusion
Kaffee
Kalzium
Kapsel
Kaumuskel
Kehlkopf
Kiefer
Kind
Klinik
Knie
Knochen
Kopf
Kopfschmerzen

Korsett
Kraft
Krampf
Krankenhilfe
Krankheit
Krücke
Kuchen
Lid
Lippe
Logopäde
Logopädie
Luft
Mahlzeit
Mann
Medikament
Menge
Messer
Mittagessen
Mittel
Mund
Muskel
Nacken
Nadel
Nagel
Nahrungsmittel
Nase
Nebenwirkung
Nerv
Nervenwasserpunktion
Niere
Nudeln
Oberkörper
Obst
Ohr
Operation
Orthopäde
Patient
Pille
Po
Präparat
Praxis
Problem
Pudding
Puder
Puls
Rechnung
Rehabilitation
Rollstuhl
Röntgenbild
Röntgen
Rücken
Rückenmarkskanal
Rückenmark
Rückenmuskulatur
Rückenschmerzen
Salbe
Salz
Schluckbeschwerden
Schmerzen
Schwächeanfälle
Schwäche
Schweißausbruch
Schweiß
Schwester
Schwindel
Sohn
Soße
Spasmus
Speichel
Speichelfluß
Spray
Spritze
Stoffwechsel
Stuhlgang
Suppe
Symptom
Tablette
Tasse
Tee
Tochter
Tod
Treppe
Tropfen
Urlaub
Verband
Verstopfung
Wahrheit
Wasser
Wein
Wirbel
Zahn
Zuckungen

Proper names (2)
Karl
Klara

Adjectives (15)
gekocht
häufiger
heiß
kalt
lecker
problemlos
roh
satt
schlimmer
schön
schwierig
stark
stärker
warm
weniger

Adverbs (8)
gut
hier
nie
oft
schlecht
selbst
übel
weh

Definite articles (5)
das
dem
den
der
des
die

Indef. articles (12)
ein
eine
einem
einen
einer
eines
kein
keine
keinem
keinen
keiner
keines

Interrogation pron. (5)
welche
welchem
welchen
welcher
welches

Possessive pron. (6)
mein
meine
meinem
meinen
meiner
meines

Personal pronouns (16)
ich
mir
mich
du
dir
dich
er
sie
es
ihm
ihr
ihn
wir
uns
euch
ihnen

Reflexive pronouns (5)
mich
dich
sich
uns
euch

Indef. pronouns (4)
alles
jemand
nichts
niemand

Relative pronouns (8)
das
dem
den
denen
der
deren
dessen
die

Prepositions (8)
an
auf
aus
mit
nach
über
von
zu

Prepositions with article (3)
vom
zum
zur

Conjunctions (2)
weil
wenn

Negation (1)
nicht

Punctuation marks (4)
.
,
?
!

Annex C: French prototype

Table of ideas

Thème : all.doctor.health.symptom

sujet	verbe opérateur	complé-tive	infini-tive	groupe verbal
	capacité			bredouiller
				déglutir
				dormir
				siffler
				souffler
				avoir métabolisme(9)
				avoir spasme(10)+fièvre(11)
				avoir crampe(12)+mal au dos(13)
				baisser bras(1) + tête
	parvenir		+	bouger <est_mobile>
	pouvoir		+	claquer langue
je	être capable de		+	faire mouvement
j'	être en état de		+	faire hypertension-artérielle(14)
	avoirla possibilité de			fermer yeux
			+	gonfler joue
	avoir des difficultés à		+	incliner tête
	Δ			pencher tête
				plier bras(1)
				redresser tête
				ressentir crampe(12)+mal au dos(13)
				soulever bras(1a)+cuisse(4)
				tourner tête
	volonté			
je	vouloir	+	+	bredouiller
j'	être décidé à		+	souffrir <anatomie>+spasme(10)
	être déterminé à		+	
	Δ			
	besoin			
				bredouiller
je	besoin	+	+	baisser bras(1a)+tête
j'	Δ			bouger <est_mobile>

	envie			
	envie	+	+	
je	aimer(cond)	+	+	bredouiller
j'	désire	+	+	guérir
	souhaite	+	+	
	vouloir(cond)	+	+	
	Δ			
	devoir			
je	devoir		+	dormir
j'	contraint à		+	
	Δ			
	être autonome			
je	être ballonné			
j'	être lourd			<anatomie>
	avoir mal			

Thème : all.doctor.health.therapy

sujet	verbe opérateur	complé-tive	infini-tive	groupe verbal
	capacité			
	parvenir		+	
	pouvoir		+	obtenir scéances d'orthophonie(23) obtenir délai(24)
je	être capable de		+	
j'	être en état de		+	suivre scéances d'orthophonie(23)
	avoir la possib. de		+	
	avoir des diff. à		+	
	Δ			
	volonté			
	vouloir	+	+	
	être décidé à		+	
	être déterminé à		+	
	Δ			
	besoin			
	avoir besoin	+	+	faire gymnastique corrective(22)
je	Δ			obtenir scéances d'orthophonie(23) obtenir délai(24)
j'	***envie***			suivre scéances d'orthophonie(23)
	avoir envie	+	+	
	aimer (cond)	+	+	
	désire	+	+	
	souhaite	+	+	
	vouloir(cond)	+	+	

	intention			
	cesser de		+	
	continuer à		+	
	penser		+	
	proposer de		+	
	refuser de	+	+	
	accepter	+	+	
	attendre		+	
	envisager		+	
	éviter		+	
	Δ			
	impression			
	détester	+	+	
	sentir	+		
	préférer		+	
	en colère de		+	
	triste de		+	
	frustré de		+	
	content		+	
	soucieux de		+	faire gymnastique corrective(22)
je	aimer	+	+	
j'	apprécier	+	+	suivre scéances d'orthophonie(23)
	espérer	+	+	
	Δ			
	devoir			
	devoir		+	
	être contraint à		+	
	Δ			

	savoir			
	considérer	+		
	constater	+		
	croire	+		
	penser que	+		
	réaliser	+		
	savoir	+		
	supposer	+		
	être persuadé	+	+	
	admettre	+		
	Δ			
j'	besoin			gymnastique corrective(22)

Thème : all.doctor.health.medicine

sujet	**verbe opérateur**	**complé-tive**	**infini-tive**	**groupe verbal**
	capacité			
				accepter perfusion
	parvenir		+	absorber cachet(20)
	pouvoir		+	avaler cachet(20)
je	être capable de		+	faire mouvement
j'	être en état de		+	faire examen(16)
	avoir la possib. de		+	obtenir repos(19)+cachet(20)
	avoir des diff. à		+	prendre repos(19)+cachet(20)
	Δ			respecter dose
	volonté			
	vouloir	+	+	avoir traitement(21)+(20)
je	être décidé à		+	faire examen(16)
j'	être déterminé à		+	prendre repos(19)+cachet(20)
	Δ			donner ordonnance à <être_humain>

	besoin			
		+	+	
	avoir besoin			
	Δ			
	envie			faire examen(16)
je		+	+	prendre repos(19)+cachet(20)
j'	avoir envie	+	+	obtenir repos(19)+cachet(20)
	aimer (cond)	+	+	donner ordonnance à <être_humain>
	désire	+	+	
	souhaite		+	
	vouloir(cond)			
	Δ			

	intention			
	cesser de		+	
	continuer à		+	
	penser		+	
	proposer de		+	
	refuser de	+	+	
	accepter	+	+	
	attendre		+	
	envisager		+	
	éviter		+	
	Δ			
	impression			
	détester	+	+	
	sentir	+		
	préférer		+	
	en colère de		+	
	triste de		+	
	frustré de		+	faire examen(16)
	content		+	prendre cachet(20)
	soucieux de		+	
je	aimer	+	+	
j'	apprécier	+	+	
	espérer	+	+	
	Δ			
	devoir			
	devoir		+	
	être contraint à		+	
	Δ			

	savoir			
	considérer	+		
	constater	+		
	croire	+		
	penser que	+		
	réaliser	+		
	savoir	+		
	supposer	+		
	être persuadé	+	+	
	admettre	+		
	Δ			
j'	besoin			<typeMédicament>

Thème : all.doctor.food

sujet	**verbe opérateur**	**com-plétive**	**infini-tive**	**groupe verbal**
	capacité			
				avaler [<aliment>]
				boire [eau(27)]
	parvenir		+	manger [viande(29)]
	pouvoir		+	mastiquer
je	être capable de		+	agripper <vaisselle>
j'	être en état de		+	attraper <vaisselle>
	avoir la possib. de		+	couper [viande]
	avoir des diff. à		+	empoigner <vaisselle>
	Δ			mâcher viande(29)
				prendre repas(28)
				tenir <vaisselle>

	volonté			
	vouloir	+	+	
	être décidé à		+	
	être déterminé à		+	
	Δ			
	besoin			
	avoir besoin	+	+	
	Δ			
	envie			
	avoir envie	+	+	boire [eau(27)]
	aimer (cond)	+	+	manger [viande(29)]
	désire	+	+	prendre repas(28)
je	souhaite	+	+	
j'	vouloir(cond)	+	+	
	Δ			
	intention			
	cesser de		+	
	continuer à		+	
	penser		+	
	proposer de		+	
	refuser de		+	
	accepter	+	+	
	attendre	+	+	
	envisager		+	
	éviter		+	
	Δ			

	impression			
	détester	+	+	
	sentir	+		
	préférer		+	
	en colère de		+	
	triste de		+	
	frustré de		+	
	content		+	
je	soucieux de		+	manger [viande(29)]
j'	aimer	+	+	prendre repas(28)
	apprécier	+	+	
	espérer	+	+	
	Δ			
	devoir			
	devoir		+	
	être contraint à		+	
	Δ			
je j'	avoir faim avoir soif			

Thème : all.doctor.hygiene.bath

sujet	verbe opérateur	complétivee	infinitive	groupe verbal
	capacité			
				baisser bras(1) + tête
				bouger <est_mobile>
	parvenir		+	déboucher flacon(34)
	pouvoir		+	descendre bras(1a)
je	être capable de		+	fermer robinet(35) + bouton
j'	être en état de		+	lever bras(1a)
				ouvrir robinet(35) + flacon(34)
	avoirla possibilité de		+	reboucher robinet(35) + flacon(34)
			+	refermer robinet(35) + flacon(34)
	avoir des difficultés à			utiliser lavabo(37)
	Δ			se laver <être_humain> <lavable>
	volonté			
je	vouloir	+	+	
j'	être décidé à		+	utiliser douche(37)
	être déterminé à		+	
	Δ			
	besoin			
				baisser bras(1a)+tête
je	besoin	+	+	bouger <est_mobile>
j'	Δ			descendre bras(1a)
	envie			
	envie	+	+	
je	aimer(cond)	+	+	
j'	désire	+	+	nettoyer lunettes(36)
	souhaite	+	+	utiliser lavabo(37)
	vouloir(cond)	+	+	
	Δ			
j'	besoin			<accessoires_ toilette>

Thème : all.doctor.hygiene.toilet

sujet	verbe opérateur	complétivee	infinitive	groupe verbal
	capacité			
	parvenir		+	
	pouvoir		+	
je	être capable de		+	
j'	être en état de		+	aller toilettes
				prendre laxatif
	avoirla possibilité de		+	
			+	
	avoir des difficultés à			
	Δ			
	volonté			
	vouloir	+	+	
	être décidé à		+	
	être déterminé à		+	
je	Δ			prendre laxatif
j'				
	besoin			
	besoin	+		
	Δ			
	envie			
	envie	+	+	
je	aimer(cond)	+	+	
j'	désire	+	+	avoir crampe(12) + <élimination>
	souhaite	+	+	
	vouloir(cond)	+	+	prendre laxatif
	Δ			
je	être sujet			<élimination>
	être ballonné			

Thème : all.doctor.psycho

sujet	aux	attr_verb opérateur	com.	inf.	verbe	prép	compl1 (n°classe)	compl2
		<capacité>						
					écrire	-	-	-
je	-	parvenir	-	><	espérer	-	-	-
	-	pouvoir	-	><	mentir	-	-	-
	être	capable de	-	><	réagir	-	-	-
		en état de	-	><	réaliser	-	-	-
j'	avoir	la possib. de	-	><	sourire	-	-	-
		des diff. à	-	><	vivre	-	-	-
					accepter	-	(44+46b+51b+	-
je/j'	-	-	-	-			57b+60+67+68+	
							69+71)	
					activer	-	résultat(61)	-
					admettre	-	(44+46b+49+55+	
							60)	
					amuser	-	<être_humain>	-
					apprécier	-	comportement	-
					arrêter	-	discussion(62)	-
					augmenter	-	espoir(53a)	-
					avertir	-	<être_humain>	-
					avoir	-	(45+50+64a+66	-
							a+73b)	
						de	morale	-
					choisir	-	(51a+60a+70)	-
					comprendre	-	<être_humain>	-
							+(55+61+67+68)	
					compter	sur	<être_humain>	-
							+(56+61+64b+	
							73)	
					connaître	-	(46a+56+57a)	-
					constater	-	(45b+46)	-
					croire	en	<être_humain>	-
							+(52+54+60+63a	
							+71b)	
					découvrir	-	(45a+48+57b)	-
					définir	-	(57+71)	-
					déterminer	-		-
					douter	de	<être_humain>	-
							+(61+63b+66b)	
							<être_humain>	

					écouter	-	(62+72)	-
					ennuyer	-	(57b+65+72)	-
					éviter	-	situation	-
					garder	-	vie(58)	-
					mériter	-	(62+64a)	-
					profiter	de	(61+63)	-
					prolonger	-	discussion(62)	-
					recevoir	-	<être_humain>	-
					recommencer	-	+intérêt(66b)	-
					rencontrer	-	<être_humain> +question(64a)	-
					répondre	à	<être_humain> -	-
					se détendre	pron	situation	-
					s'adapter	pron	(61+64)	-
						pron+à	<être_humain	-
					se souvenir	pro+de	>+ excuse(66)	-
					présenter	- + à		<être humain>

		<volonté>			vivre	-	-	-
					avertir	-	<être_humain>	-
je	-	vouloir	><	><	connaître	-	vérité(57b)	-
	être	décidé à	-	><	conseiller	-	<être_humain>	-
		déterminé à	-	><	consulter	-		-
je/j'	-	-	-	-	découvrir	-	vérité(57b)	-
					définir	-	qualité(57)	-
					dépendre	-	<être_humain>	-
					déranger	-		-
					déterminer	-	qualité(57)	-
					écouter	-	<être_humain>	-
					garder	-	vérité(57b)	-
					manquer	de	chaleur(50b)	-
					questionner	-	<être_humain>	-
					remercier	-		-
					répondre	à		-
					revoir	-		-
					savoir	-	(57b+69+73b)	-
					parler	de	vérité(57)	-
						de + à		<être humain>
					donner	- + à	(50a)+(67a)	
					présenter	- + à	<être_hum>	

		<besoin>			espérer	-	-	-
					mentir	-	-	-
					amuser	-	<être_humain>	-
					augmenter	-	liberté(53a)	-
					avoir		(50+53+63+72+	
j'	avoir	besoin	><	><			74)	
						-	morale(71b)	-
je/j'	-	-	-	-	comprendre	-	(61)+(67)	-
					compter	sur	<être_humain>	-
							+succès(56)	
					connaître	-	(57b)+(67b)	-
					conseiller	-	<être_humain>	-
					consulter	-		-
					définir	-	qualité(57)	-
					déterminer	-		-
					garder	-	(57b)+(65)	-
					penser	à	(60a)+(64)	-
					questionner	-	<être_humain>	-
					recevoir	-	(61)+(63)	-
					réfléchir	à	(60a)+(64)	-
					rencontrer	-	<être_humain>	-
							+intérêt(66b)	
					se détendre	pron	<être_humain>	-
					parler	de	(45b+57+59)	-
						de + à		<être humain>
					donner	- + à	(50a)+(67a)	<être humain>

		<envie>			sourire	-	-	-
			><	><	activer	-	résultat(61)	-
			><	><	amuser	-	<être_humain>	-
			><	><	arrêter	-	discussion(62)	-
j'	avoir	envie	><	><	avoir	-	(50+53+63)	-
	-	aimer(cond)	><	><	choisir	-	(60a)+(70)	-
je	-	désire			connaître	-	<être_humain>	-
	-	souhaite	-	-			+(45b+64c+66b+	
	-	vouloir(con)					67b+71+73b)	
					conseiller	-	<être_humain>	-
je/j'	-	-			consulter	-		-
					découvrir	-	limite(71)	-
					dépendre	de	<être_humain>	-
					déranger	-		-
					déterminer	-	valeur(71)	-
					discuter	de	(45b+46+59)	-
					écouter	-	<être_humain>	-
					éviter	-	(62)+(72)	-
					penser	à	<être_humain>	-
							+(60a)+(64)	
					prolonger	-	(62+64a+65a)	-
					recevoir	-	(61)+(63)	-
					recommencer	-	phrase(62)	-
					réfléchir	à	(60a)+(64)	-
					remercier	-	<être_humain>	-
					rencontrer	-	<être_humain>	-
							+intérêt(66b)	
					répondre	à	<être_humain>	-
						-	+requête(64a)	-
					revoir	-	<être_humain>	-
					savoir	-	(69)+(73b)	-
					se détendre	pron	<être_humain>	-
					parler	de	(45b+57+59)	-
						de + à		<être_
					donner	- + à	(50a)+(67a)	humain>
					présenter	- + à	<être_humain>	<être_hu
							+excuse(66)	main>

		<intention>						
je	-	cesser de	-	><	mentir	-	-	
	-	continuer à	-	><	anéantir	-	espoir(53a)	
	-	penser	-	><	arrêter	-	discussion(62)	
	-	proposer de	-	><	avoir	-	responsabilité	
	-	refuser de	-	><		de	(51b)+(52)	
j'	-	accepter	><	><	comprendre	-	<être_humain>	
	-	attendre	><	><			+comportement(	
	-	envisager	-	><			55)	
	-	éviter	-	><	connaître	-	<être_humain>	
							+(56+57a+64c+6	-
		<savoir>					7b)	
je	-	considérer	><	-	conseiller	-	<être_humain>	
	-	constater	><	-	constater	-	(45b)+(46)	
	-	croire	><	-	croire	en	<être_humain>	
	-	penser que	><	-			+(52+54+60+61	
	-	réaliser	><	-			+63a+66b+67a+	
	-	savoir	><	-			71b)	
	-	supposer	><	-	décourager	-	<être_humain>	
	être	persuadé	><	><	dépendre	de		
j'	-	admettre	><	-	déranger	-		
					discuter	de	problème	-
		<impression>			écouter	-	<être_humain>	
je	-	détester	><	><	éviter	-	vérité	
	-	sentir	><	-	garder	-	(57a)+(65a)	
	-	préférer	-	><	manquer	de	(50b)+(51)	
	être	en colère de	-	><	penser	à	(60a)+(64)	
		triste de	-	><	prolonger	-	(62)+(65a)	
		frustré de	-	><	réfléchir	à	(60a)+(64)	
		content	-	><	rencontrer	-	<être_humain>	
		soucieux de	-	><	revoir	-	<être_humain>	
j'	-	aimer	><	><	s'adapter	pron	situation	
	-	apprécier	><	><	se souvenir	pron	résultat(61)	
	-	espérer	><	><	parler	de	(45b+57+59)	
						de + à		
		<devoir>			présenter	- + à	<être_humain>	
je	-	devoir	-	><			+excuse(66)	
	être	contraint à	-	><				
je/j'	-	-	-	-				

je/j'	-/être	<devoir>	-	><	réaliser			
					accepter	-	(46b+51+57+60+64+67+68+69+71)	-
					admettre	-	(44+46b+49)	-
					avoir	-	(53)+(66a)	-
					chercher	-	réponse(50)	-
					questionner	-	<être_humain>	-
					rencontrer	-	intérêt	-
					répondre	à	<être_humain> +question(64a)	-
je/j'	-/être	<impression>	- /><	- /><	avoir	-	(53)+(66a)	-
					mériter	-	(44)+(60a)	-
					rencontrer	-	intérêt	-
je/j'	-/être	<savoir>	- /><	- /><	chercher	-	aide(50)	-
					mériter	-	(44)+(60a)	-
					rencontrer	-	intérêt	-
je/j'	-	<intention>	- /><	- /><	admettre	-	(44+46b+49)	-
					questionner	-	<être_humain>	-
					recommencer	-	discussion(62)	-
					répondre	à	<être_humain> question(64a)	-

je	être	autonome	-	-	-	-	-	-
		irritable	-	-	-	-	-	
		oppressé	-	-	-	-	-	
		passif	-	-	-	-	-	
		stressé	-	-	-	-	-	
		attristé	-	-	-	par	(47+49a+55+63a+68)	
		averti	-	-	-	de	(46a+56+60a)	
		confronté	-	-	-	à	(44+46b+47)	
		content	-	-	-	de	<être_humain>+(46+49a+55+56a+60+61+68)	
		en colère	-	-	-	-	-	
			-	-	-	contre	<être_humain>+(46b+55+68+71)	
		frustré	-	-	-	de		
		soucieux	-	-	-	de	(51+52+63b+74)	
		sujet	-	-	-	à	(44+45b+46a+55+56b+57)	
j'	avoir	besoin	-	-	-	de	stress(48)	
		peur	-	-	-	de	(50b+51+52)	
							solitude(44)	

The conceptual module

```
/* ConceptDoc  de "sens commun" */

domaines_primaires;

racine(tout);

/* tout = { être_objet , notion_opérateur } ; */
tout.être_objet.notion_opérateur;

/* être_objet = { être_humain , non_être_humain}; */
être_objet.être_humain.non_être_humain ;

/*non_être_humain = { AspectPhysique, MaladieSoins, Anatomie,
  Sensations, Alimentation, ActivitéVitale, Mental, Société, BeauxArts,
  SciencesTechniques, TerrePlantesAnimaux, EspaceTempsLogique,
  ObjetsDivers };*/
```

```
non_être_humain.AspectPhysique.MaladieSoins.Anatomie.Sensations.Alimenta
  tion.ActivitéVitale.Mental.Société.DiversesActivités.BeauxArts.Scienc
  esTechniques.TerrePlantesAnimaux.EspaceTempsLogique.ObjetsDivers;

/* être_humain = { locuteur , auditeur , tiers } ; */
être_humain.locuteur.auditeur.tiers ;

/* Anatomie = { AnatExterne, AnatInterne } ; */
Anatomie.AnatExterne.AnatInterne;

/* AnatExterne= { AvecArticulation, SansArticulation } ; */
AnatExterne.AvecArticulation.SansArticulation ;

/* AvecArticulation= {AnatFermable,
  AnatClaquable,AnatGonflable,AnatDivers} ; */
AvecArticulation.AnatFermable.AnatClaquable.AnatGonflable.AnatDivers ;

/*SansArticulation = {  AnatDiminuable ,AnatAutre } ; */
 SansArticulation.AnatDiminuable.AnatAutre ;

/* AnatInterne= {AnatAir,AnatGénérale } ; */
AnatInterne.AnatAir.AnatGénérale   ;

/* MaladieSoins = { Maladies , Thérapie };*/
MaladieSoins.Maladies.Thérapie ;

/* Maladies = { Perte, AutresMaladies } */
Maladies.Perte.AutresMaladies ;

/* Thérapie = { Médicaments , Soins };*/
Thérapie.Médicaments.Soins;

/* ActivitéVitale = { Vie , Mort };*/
ActivitéVitale.Vie.Mort ;

/* Alimentation = { Aliments, Repas };*/
Alimentation.Aliments.Repas ;

/* Aliments = { AlimentsSolides, AlimentsLiquides};*/
Aliments.AlimentsSolides.AlimentsLiquides ;

/* AlimentsSolides = { AlimentMonoPartie, AlimentPolyPartie};*/
AlimentsSolides.AlimentMonoPartie.AlimentPolyPartie;

/* Mental = { ActivitéIntellectuelle, LangageLittérature,
  MoralePsychologie, Religion };*/
Mental.ActivitéIntellectuelle.LangageLittérature.MoralePsychologie.Relig
  ion;
```

```
/* MoralePsychologie = { PsychoExprimable, PsychoDivers };*/
MoralePsychologie.PsychoExprimable.PsychoDivers ;

/* PsychoExprimable= { Mystèrieux, Faux, AutrePsycho };*/
PsychoExprimable.Mystèrieux.Faux.AutrePsycho ;

/* LangageLittérature = { ConstructionLinguistique,
  ContenantLinguistique, LittératureDiverse };*/
LangageLittérature.ConstructionLinguistique.ContenantLinguistique.Littér
  atureDiverse ;

/* ConstructionLinguistique= { TypeConstruction, RésultatConstruction
  };*/
ConstructionLinguistique.TypeConstruction.RésultatConstruction ;

/*Société = { Argent, Droit, OrganisationSociale, RelationsHumaines };*/
Société.Argent.Droit.OrganisationSociale.RelationsHumaines;

/*RelationsHumaines = { Communication, AutresRelations };*/
RelationsHumaines.Communication.AutresRelations ;

/*Communication = { RelationNégative, RelationDirecte , RelationNeutre
  };*/
Communication.RelationNégative.RelationDirecte.RelationNeutre ;

/*DiversesActivités = { ActesGestes, JeuxSports};*/
DiversesActivités.ActesGestes.JeuxSports;

/*ActesGestes = { Actes, Gestes};*/
ActesGestes.Actes.Gestes;

/*BeauxArts = { Art, Architecture, Maison };*/
BeauxArts.Art.Architecture.Maison;

/*Maison = { MaisonPassage, MaisonContenant, MaisonDénivellation,
  MaisonDivers}*/
Maison.MaisonPassage.MaisonContenant.MaisonDénivellation.MaisonDivers;

/*MaisonPassage= { SansClaquer, PassageRésistant };*/
MaisonPassage.SansClaquer.PassageRésistant ;

/*SciencesTechniques = { Sciences, Techniques, AirEauFeu,
  CouleurLumière, Métaux, Transports };*/
SciencesTechniques.Sciences.Techniques.AirEauFeu.CouleurLumière.Métaux.T
  ransports;

/*TerrePlantesAnimaux = { Agriculture, Animaux, Géographie };*/
```

```
TerrePlantesAnimaux.Agriculture.Animaux.Géographie;

/*EspaceTempsLogique = { Changement, Espace, Formes, Mouvement,
  QuantitéQualitéImportance, RelationLogique, Temps };*/
EspaceTempsLogique.Changement.Espace.Formes.Mouvement.QuantitéQualitéImp
  ortance.RelationLogique.Temps;

/* Temps = { Avenir, Présent, Passé };*/
Temps.Avenir.Présent.Passé ;

/*RelationLogique = { Abstraite, Résultat, Intelligible, AutreLogique
  };*/
RelationLogique.Abstraite.Résultat.Intelligible.AutreLogique ;

/*ObjetsDivers = { Objets, TextilesVêtements };*/
ObjetsDivers.Objets.TextilesVêtements ;

/*Objets = { ObjetsContenants, ObjetsAutres, ObjetsFermables };*/
Objets.ObjetsContenants.ObjetsAutres.ObjetsFermables  ;

/*notion_opérateur = { capacité , besoin , volonté , envie , réflexion
  };*/
notion_opérateur.capacité.besoin.volonté.envie.réflexion;

/*réflexion = { intention, devoir, savoir, impression };*/
réflexion.intention.devoir.savoir.impression;

/*****************************************************************/

domaines_composes;

/* Propriété <est_usuel> */
est_usuel(AvecArticulation,Alimentation,ObjetsDivers);

/* Propriété <est_lavable> */
est_lavable(être_humain,AnatExterne,AlimentsSolides,ObjetsDivers);

/* Propriété <est_mobile> */
est_mobile(être_humain,AvecArticulation,MaisonDivers,MaisonPassage);

/* Propriété <est_contenant> */
est_contenant(ObjetsContenants,MaisonContenant);

/* Propriété <dénivellation> */
dénivellation(MaisonDénivellation);

/* Propriété <se_fermant> */
se_fermant(MaisonPassage,AnatFermable);
```

```
/* Propriété <est_materiel> */
est_matériel(ObjetsDivers,Maison);

/* Propriété <est_comestible> */
est_comestible(Alimentation,Médicaments);

/* Propriété <est_divisable> */
est_divisable(AnatDiminuable,AlimentMonoPartie,AspectPhysique);

/* Propriété <est_faisable> */
est_faisable(Soins,Repas,ConstructionLinguistique,RelationNeutre,Mystèri
  eux);

/* Propriété <valeur_morale> */
valeur_morale(Mental,RelationLogique,Espace,MaladieSoins);

/* Propriété <évoluant> */
évoluant(Temps,Changement);

/* Propriété <impliquant_action> */
impliquant_action(MaladieSoins,ContenantLinguistique,ConstructionLinguis
  tique);

/* Propriété <est_intelligible> */
est_intelligible(être_humain,ActesGestes,LangageLittérature,Organisation
  Sociale,Changement,Résultat,Intelligible,PsychoExprimable,AspectPhysi
  que);

/* Propriété <est_audible> */
est_audible(être_humain,LangageLittérature,Faux,RelationDirecte);

/* Propriété <est_visible> */
est_visible(être_humain,MoralePsychologie,ActesGestes,Changement,Formes,
  QuantitéQualitéImportance,Résultat);

/* Propriété <est_destructible> */
est_destructible(être_humain,Abstraite,Avenir,Vie);

/* Propriété <est_négatif> */
est_négatif(RelationNégative,RelationDirecte,Mystèrieux);

/* Propriété <qui_claque> */
qui_claque(AnatClaquable,PassageResistant);

/* Propriété <est_gonflable> */
est_gonflable(AnatGonflable,AnatAir);
```

```
/* Propriété <est_recevable> */
est_recevable(être_humain,RelationDirecte,RelationNeutre);

/* Propriété <une_situation> */
une_situation(Mental,OrganisationSociale,RelationsHumaines,ActesGestes,Q
  uantitéQualitéImportance,RelationLogique);

non_locuteur(auditeur,tiers) ;

/*********************************************************************/
contraintes;

tu(auditeur);

je(locuteur) ;

est_professeur(tiers);

est_médecin(tiers);

il(tiers) ;
il(non_être_humain);

cela(notion_opérateur);

/* Anatomie */
/* Anatomie.AnatExterne.AvecArticulation.AnatFermable */
est_main(AnatFermable);
est_yeux(AnatFermable);

/*Anatomie.AnatExterne.AvecArticulation.AnatClaquable */
est_doigt(AnatClaquable);
est_langue(AnatClaquable);

/* Anatomie.AnatExterne.AvecArticulation.AnatGonflable */
est_joue(AnatGonflable);
est_ventre(AnatGonflable);

/* Anatomie.AnatExterne.AvecArticulation.AnatDivers  */
est_annulaire(AnatDivers);
est_articulation(AnatDivers);
est_auriculaire(AnatDivers);
est_avant_bras(AnatDivers);
est_cheville(AnatDivers);
est_coude(AnatDivers);
est_épaule(AnatDivers);
est_genou(AnatDivers);
est_index(AnatDivers);
```

```
est_lèvre(AnatDivers);
est_majeur(AnatDivers);
est_membres(AnatDivers);
est_orteil(AnatDivers);
est_paupière(AnatDivers);
est_poignet(AnatDivers);
est_pouce(AnatDivers);
est_torse(AnatDivers);
est_tronc(AnatDivers);
est_hanche(AnatDivers);
est_clavicule(AnatDivers);
est_coccyx(AnatDivers);
est_bras(AnatDivers);
est_jambe(AnatDivers);
est_pied(AnatDivers);
est_tête(AnatDivers);
est_buste(AnatDivers);
est_cou(AnatDivers);
est_dos(AnatDivers);
est_nuque(AnatDivers);

/* Anatomie.AnatExterne.SansArticulation.AnatDiminuable */
est_cheveux(AnatDiminuable);
est_ongles_des_mains(AnatDiminuable);
est_ongles_des_pieds(AnatDiminuable);

/* Anatomie.AnatExterne.SansArticulation.AnatAutre */
est_dent(AnatAutre);
est_crâne(AnatGénérale);
est_cuisse(AnatAutre);
est_oreille(AnatAutre);
est_fesse(AnatAutre);
est_figure(AnatAutre);
est_gorge(AnatAutre);
est_mollet(AnatAutre);
est_nez(AnatAutre);
est_peau(AnatAutre);
est_plante_de_pied(AnatAutre);
est_poitrine(AnatAutre);
est_talon(AnatAutre);
est_visage(AnatAutre);

/* Anatomie.AnatInterne.AnatGénérale */
est_coeur(AnatGénérale);
est_colonne_vertébrale(AnatGénérale);
est_estomac(AnatGénérale);
est_foie(AnatGénérale);
est_lombes(AnatGénérale);
```

```
est_muscle(AnatGénérale);
est_muscle_de_la_mâchoire(AnatGénérale);
est_muscle_du_cou(AnatGénérale);
est_muscle_du_visage(AnatGénérale);
est_nerf_sciatique(AnatGénérale);
est_sacrumn(AnatGénérale);
est_sinus(AnatGénérale);
est_tandon(AnatGénérale);
est_vertèbres_cervicales(AnatGénérale);
est_vessie(AnatGénérale);

/* Anatomie.AnatInterne.AnatAir */
est_poumon(AnatAir);

 /* Maison*/
/* Maison.MaisonPassage.SansClaquer */
est_rideau(SansClaquer);
est_robinet(SansClaquer);
est_robinet_d_eau_chaude(SansClaquer);
est_robinet_d_eau_froide(SansClaquer);

/*Maison.MaisonPassage.PassageRésistant };*/
est_fenêtre(PassageRésistant);
est_porte(PassageRésistant);
est_volet(PassageRésistant);

/* Maison.MaisonContenant }; */
est_baignoire(MaisonContenant);
est_douche(MaisonContenant);
est_lavabo(MaisonContenant);
est_toilettes(MaisonContenant);

/* Maison.MaisonDénivellation */
est_escalier(MaisonDénivellation);
est_marche(MaisonDénivellation);

/* Maison.MaisonDivers*/
est_chaise(MaisonDivers);
est_lit(MaisonDivers);

 /* Objets*/
/* Objets.ObjetsContenants */
est_bol(ObjetsContenants);
est_bouteille(ObjetsContenants);
est_flacon(ObjetsContenants);
est_pot_de_crème(ObjetsContenants);
est_tasse(ObjetsContenants) ;
est_tube_de_dentifrice(ObjetsContenants);
```

```
est_verre(ObjetsContenants);

/* Objets.ObjetsAutres */
est_brosse(ObjetsAutres);
est_brosse_à_dent(ObjetsAutres);
est_couteau(ObjetsAutres);
est_cuiller(ObjetsAutres);
est_cuillère(ObjetsAutres);
est_dentier(ObjetsAutres);
est_fourchette(ObjetsAutres);
est_lunettes(ObjetsAutres);
est_peigne(ObjetsAutres);
est_savon(ObjetsAutres);
est_savonnette(ObjetsAutres);

/* Objets.ObjetsFermables */
est_bouton(ObjetsFermables);

 /* AspectPhysique */
/* AspectPhysique */
est_appétit(AspectPhysique);
est_état(AspectPhysique);

 /* Alimentation*/
/* Alimentation.Aliments.AlimentsSolides.AlimentMonoPartie */
est_chair(AlimentMonoPartie);
est_fruit(AlimentMonoPartie);
est_légume(AlimentMonoPartie);
est_pain(AlimentMonoPartie);
est_viande(AlimentMonoPartie);

/* Alimentation.Aliments.AlimentsSolides.AlimentPolyPartie */
est_sel(AlimentPolyPartie);
est_sucre(AlimentPolyPartie);

/* Alimentation.Aliments.AlimentsLiquides */
est_boisson(AlimentsLiquides);
est_eau(AlimentsLiquides);

/* Alimentation.Repas */
est_déjeuner(Repas);
est_dîner(Repas);
est_petit_déjeuner(Repas);
est_repas(Repas);
est_souper(Repas);

 /* MaladieSoins*/
/* MaladieSoins.Maladies.Perte */
```

```
est_air(Perte);
est_énergie(Perte);
est_force(Perte);
est_goût(Perte);
est_haleine(Perte);
est_larme(Perte);
est_métabolisme(Perte);
est_mucosité_nasale(Perte);
est_odorat(Perte);
est_pouls(Perte);
est_pression_sanguine(Perte);
est_pression_vasculaire(Perte);
est_respiration(Perte);
est_salive(Perte);
est_sueur(Perte);
est_transpiration(Perte);

/* MaladieSoins.Maladies.AutresMaladies */
est_anorexie(AutresMaladies);
est_asphyxie(AutresMaladies);
est_asthénie(AutresMaladies);
est_circulation_sanguine(AutresMaladies);
est_coliques(AutresMaladies);
est_convulsion(AutresMaladies);
est_crampe(AutresMaladies);
est_diarrhée(AutresMaladies);
est_douleur(AutresMaladies);
est_dysarthrie(AutresMaladies);
est_dysphagie(AutresMaladies);
est_dysphonie(AutresMaladies);
est_dyspnée(AutresMaladies);
est_faiblesse(AutresMaladies);
est_fausse_route(AutresMaladies);
est_fièvre(AutresMaladies);
est_hypertension_artérielle(AutresMaladies);
est_hypotension_artérielle(AutresMaladies);
est_lourdeur(AutresMaladies);
est_maladie(AutresMaladies);
est_malaise(AutresMaladies);
est_mal_au_coeur(AutresMaladies);
est_mal_au_dos(AutresMaladies);
est_mal_au_ventre(AutresMaladies);
est_mal_à_la_tête(AutresMaladies);
est_mal_de_coeur(AutresMaladies);
est_mal_de_dos(AutresMaladies);
est_mal_de_tête(AutresMaladies);
est_mal_de_ventre(AutresMaladies);
est_spasme(AutresMaladies);
```

```
est_symptôme(AutresMaladies);
est_toux(AutresMaladies);
est_trouble(AutresMaladies);

/* MaladieSoins.Thérapie.Médicaments */
est_ampoule(Médicaments);
est_aspirine(Médicaments);
est_cachet(Médicaments);
est_dose(Médicaments);
est_gouttes(Médicaments);
est_laxatif(Médicaments);
est_médicament(Médicaments);
est_ordonnance(Médicaments);
est_perfusion(Médicaments);
est_pilule(Médicaments);
est_piqûre(Médicaments);
est_repos(Médicaments);
est_traitement(Médicaments);

/* MaladieSoins.Thérapie.Soins */
est_assistance(Soins);
est_examen(Soins);
est_gymnastique_corrective(Soins);
est_gymnastique_respiratoire(Soins);
est_radiographie(Soins);
est_rééducation(Soins);
est_séances_de_gymnastique_corrective(Soins);
est_séances_de_gymnastique_respiratoire(Soins);
est_séances_de_massage(Soins);
est_séances_d_acuponcture(Soins);
est_séances_d_ergothérapie(Soins);
est_séances_d_orthophonie(Soins);

 /* Mental*/
/* Mental.ActivitéIntellectuelle */
est_difficulté(ActivitéIntellectuelle);
est_jugement(ActivitéIntellectuelle);
est_pensée(ActivitéIntellectuelle);
est_possibilité(ActivitéIntellectuelle);
est_problème(ActivitéIntellectuelle);
est_raisonnement(ActivitéIntellectuelle);
est_responsabilité(ActivitéIntellectuelle);
est_solution(ActivitéIntellectuelle);
est_vérité(ActivitéIntellectuelle);

/* Mental.MoralePsychologie.PsychoDivers */
est_amusement(PsychoDivers);
est_angoisse(PsychoDivers);
```

```
est_apparence(PsychoDivers);
est_appréhension(PsychoDivers);
est_chaleur(PsychoDivers);
est_commodité(PsychoDivers);
est_confiance(PsychoDivers);
est_confort(PsychoDivers);
est_courage(PsychoDivers);
est_défaut(PsychoDivers);
est_ennui(PsychoDivers);
est_épreuve(PsychoDivers);
est_foi(PsychoDivers);
est_frustration(PsychoDivers);
est_gratitude(PsychoDivers);
est_habitude(PsychoDivers);
est_humilité(PsychoDivers);
est_impression(PsychoDivers);
est_malheur(PsychoDivers);
est_morale(PsychoDivers);
est_mur(PsychoDivers);
est_nature(PsychoDivers);
est_peine(PsychoDivers);
est_prétexte(PsychoDivers);
est_réconfort(PsychoDivers);
est_solitude(PsychoDivers);
est_souci(PsychoDivers);
est_stress(PsychoDivers);
est_volonté(PsychoDivers);

/*Mental.MoralePsychologie.PsychoExprimable.Mystèrieux */
est_mystère(Mystèrieux);

/*Mental.MoralePsychologie.PsychoExprimable.Faux */
est_mensonge(Faux);

/*Mental.MoralePsychologie.PsychoExprimable.AutrePsycho */
est_bonheur(AutrePsycho);
est_émotion(AutrePsycho);
est_état_d_âme(AutrePsycho);
est_humeur(AutrePsycho);
est_injustice(AutrePsycho);
est_intérêt(AutrePsycho);
est_joie(AutrePsycho);
est_mutisme(AutrePsycho);
est_plaisir(AutrePsycho);
est_silence(AutrePsycho);

/* Mental.LangageLittérature.ContenantLinguistique */
est_discussion(ContenantLinguistique);
```

```
/* Mental.LangageLittérature.LittératureDiverse */
est_langage(LittératureDiverse);
est_signification(LittératureDiverse);
est_version(LittératureDiverse);

/* Mental.LangageLittérature.ConstructionLinguistique.TypeConstruction*/
est_question(ConstructionLinguistique);
est_requête(ConstructionLinguistique);

/*
  Mental.LangageLittérature.ConstructionLinguistique.RésultatConstructi
  on */
est_phrase(ConstructionLinguistique);

 /* Société*/
/* Société.OrganisationSociale */
est_situation(OrganisationSociale);

/* Société.RelationsHumaines.AutresRelations */
est_amour(AutresRelations);
est_autonomie(AutresRelations);
est_désir(AutresRelations);
est_honte(AutresRelations);
est_présence(AutresRelations);
est_respect(AutresRelations);
est_soutien(AutresRelations);

/* Société.RelationsHumaines.Communication.RelationDirecte */
est_atteinte(RelationDirecte);
est_critique(RelationDirecte);
est_message(RelationDirecte);
est_parole(RelationDirecte);

/* Société.RelationsHumaines.Communication.RelationNégative */
est_tension(RelationNégative);

/* Société.RelationsHumaines.Communication.RelationNeutre */
est_aide(RelationNeutre);
est_excuse(RelationNeutre);
est_information(RelationNeutre);
est_promesse(RelationNeutre);
est_réponse(RelationNeutre);
est_souhait(RelationNeutre);
est_suggestion(RelationNeutre);
est_visite(RelationNeutre);

 /* EspaceTempsLogique*/
```

```
/* EspaceTempsLogique.Formes */
est_aspect(Formes);
est_rigidité(Formes);

/* EspaceTempsLogique.RelationLogique.Abstraite */
est_liberté(Abstraite);
est_relation(Abstraite);

/* EspaceTempsLogique.RelationLogique.Résultat */
est_résultat(Résultat);

/* EspaceTempsLogique.RelationLogique.Intelligible */
est_condition(Intelligible);
est_différence(Intelligible);
est_principe(Intelligible);
est_raison(Intelligible);

/* EspaceTempsLogique.RelationLogique.AutreLogique */
est_occasion(AutreLogique);
est_réalité(AutreLogique);

/* EspaceTempsLogique.Changement */
est_amélioration(Changement);
est_perte(Changement);
est_progression(Changement);

/* EspaceTempsLogique.Espace */
est_date_limite(Espace);
est_limite(Espace);
est_période(Espace);
est_terme(Espace);

/* EspaceTempsLogique.QuantitéQualitéImportance */
est_baisse(QuantitéQualitéImportance);
est_qualité(QuantitéQualitéImportance);
est_valeur(QuantitéQualitéImportance);

/* EspaceTempsLogique.Temps.Avenir */
est_espoir(Avenir);
est_perspective(Avenir);

/* EspaceTempsLogique.Temps.Présent */
est_date(Présent);
est_délai(Présent);
est_urgence(Présent);

/* EspaceTempsLogique.Temps.Passé  */
est_retard(Passé);
```

```
 /* ActivitéVitale*/
/* ActivitéVitale.Vie */
est_vie(Vie);

/* ActivitéVitale.Mort */
est_mort(Mort) ;

 /* DiversesActivités */
/* DiversesActivités.ActesGestes.Actes */
est_mouvement(Actes);

/* DiversesActivités.ActesGestes.Gestes */
est_attitude(Gestes);
est_comportement(Gestes);
est_objectif(Gestes);
est_succès(Gestes);

                          /*******/

/*auxiliaire être ;*/
être(être_humain);

/*auxiliaire avoir ;*/
avoir(être_humain);

                             /*******/

/* attribut exprimant la capacité */
capable_de(être_humain);
en_état_de(être_humain);
la_possibilité_de(être_humain);
difficultés_à(être_humain);

/* attribut exprimant la volonté */
décidé_à(être_humain);
déterminé_à(être_humain);

/* attribut exprimant le besoin */
besoin_de(être_humain);

/* attribut exprimant l'envie */
envie_de(être_humain);

/* attribut exprimant une réflexion */
contraint_à(être_humain);
en_colère_de(être_humain);
persuadé_de(être_humain);
```

```
triste_de(être_humain);
frustré_de(être_humain);

/*adjectif_1 averti_de(valeur_morale ou QuantitéQualitéImportance ou
  Société,être_humain);*/
averti_de(valeur_morale,être_humain);
averti_de(QuantitéQualitéImportance,être_humain);
averti_de(Société,être_humain);

/*adjectif_1 content_de(être_humain ou une_situation , être_humain);*/
content_de(être_humain,être_humain);
content_de(une_situation,être_humain);

content_de(être_humain);

/*adjectif_1 soucieux_de(ActivitéIntellectuelle ou MoralePsychologie ,
  être_humain);*/
soucieux_de(ActivitéIntellectuelle,être_humain);
soucieux_de(MoralePsychologie,être_humain);
soucieux_de(être_humain);

/*adjectif_1 attristé_par(ActivitéVitale ou MoralePsychologie ou
  OrganisationSociale ou être_humain ou est_négatif , être_humain);*/
attristé_par(ActivitéVitale,être_humain);
attristé_par(MoralePsychologie,être_humain);
attristé_par(OrganisationSociale,être_humain);
attristé_par(être_humain,être_humain);
attristé_par(est_négatif,être_humain);

/*adjectif_1 confronté_à { est_intelligible ou valeur_morale ou Société
  ou QuantitéQualitéImportance, être_humain };*/
confronté_à(est_intelligible,être_humain);
confronté_à(valeur_morale,être_humain);
confronté_à(Société,être_humain);
confronté_à(QuantitéQualitéImportance,être_humain);

/*adjectif_1 peur_de { ActivitéVitale ou MoralePsychologie ou
  OrganisationSociale,être_humain };*/
peur_de(ActivitéVitale,locuteur);
peur_de(MoralePsychologie,locuteur);
peur_de(OrganisationSociale,locuteur);
peur_de(être_humain,locuteur);
peur_de(locuteur);

/*adjectif_1 exprimant un état_physique */
besoin(Géographie,locuteur);
besoin(est_usuel,locuteur);
besoin(valeur_morale,locuteur);
```

```
besoin(MaladieSoins,locuteur);
besoin(OrganisationSociale,locuteur);

sujet_à(Maladies,locuteur);
sujet_à(Sensations,locuteur);
mal_à(Anatomie,locuteur);

adapté_à(ou(locuteur,Thérapie),OrganisationSociale);
adapté_à(ou(locuteur,Thérapie),être_humain);

recommandé_à(locuteur,ou(Thérapie,être_humain));

/*adjectif_0 exprimant un état physique*/
accessible(Maison);
avoir_mal(locuteur);
ballonné(locuteur);
bon(ou(Aliments,Repas,ActesGestes,LangageLittérature,Changement,AspectPh
  ysique));
bouché(est_contenant);
critique(ou(AspectPhysique,OrganisationSociale));
dissimulé(ou(ActivitéIntellectuelle,Maladies));
efficace(ou(Thérapie,être_humain));
être_bien(locuteur);
être_mal(locuteur);
faim(locuteur);
fort(QuantitéQualitéImportance);
fréquent(ou(QuantitéQualitéImportance,Maladies));
froid(Aliments);
important(ou(QuantitéQualitéImportance,AspectPhysique,OrganisationSocial
  e,Sensations,Maladies));
lourd(ou(locuteur,ObjetsDivers));
proche(Espace);
propre(est_lavable);
soif(locuteur);
utile(ou(est_usuel,est_matériel));
vide(est_contenant);

/* adjectifs_0 exprimant une réflexion */
   /*réflexion négative*/

autonome(locuteur);
en_colère(locuteur);
irritable(locuteur);
oppressé(locuteur);
passif(locuteur);
stressé(locuteur);

avoir_peur(locuteur);
```

```
                        /*******/

/*verbe opérateur exprimant la capacité ;*/
arriver_à(être_humain);
parvenir_à(être_humain);
pouvoir(être_humain);

/*verbe opérateur exprimant le besoin ;*/

/*verbe opérateur exprimant la volonté ;*/
vouloir(être_humain);

/*verbe opérateur exprimant l'envie ;*/
aimerais(être_humain);
désirer(être_humain);
souhaiter(être_humain);
voudrais(être_humain);

/*verbe opérateur exprimant la réflexion ;*/
      /*verbe opérateur exprimant l'intention ;*/
accepter(être_humain);
attendre(être_humain);
cesser_de(être_humain);
continuer_à(être_humain);
envisager_de(être_humain);
éviter_de(être_humain);
penser(être_humain);
proposer(être_humain);
refuser_de(être_humain);

     /*verbe opérateur exprimant le savoir ;*/
admettre(être_humain);
considérer(être_humain);
constater(être_humain);
croire(être_humain);
penser_que(être_humain);
réaliser_que(être_humain);
savoir(être_humain);
supposer(être_humain);

     /*verbe opérateur exprimant l'impression ;*/
aimer(être_humain);
apprécier(être_humain);
détester(être_humain);
espérer_qqch(être_humain);
sentir(être_humain);
préférer(être_humain);
```

```
     /*verbe opérateur exprimant le devoir ;*/
devoir(être_humain);

                                /*******/

/* verbes 0 */

apparaître(ou(Maladies,Sensations));
augmenter(ou(Maladies,AspectPhysique));
avaler(locuteur) ;
boire(locuteur);
bredouiller(locuteur) ;
déglutir(locuteur) ;
dormir(locuteur);
écrire(locuteur) ;
espérer(être_humain) ;
guérir(locuteur) ;
mâcher(locuteur) ;
manger(locuteur) ;
mastiquer(locuteur) ;
mentir(être_humain) ;
payer(être_humain) ;
pleurer(être_humain) ;
réagir(locuteur) ;
réaliser(être_humain) ;
répondre(est_mobile);
siffler(locuteur) ;
souffler(locuteur) ;
souffrir(locuteur) ;
sourire(locuteur) ;
vivre(locuteur) ;

                                /*******/

/* verbe 1 */

accepter_qqch(être_humain,une_situation) ;

activer(être_humain,évoluant);

agripper(locuteur,est_matériel);

aller(locuteur,Maison) ;

amuser(être_humain,être_humain);
amuser(ou(une_situation,être_humain),être_humain);
```

```
anéantir( être_humain,est_destructible) ;

arrêter(être_humain,impliquant_action);

augmenter(être_humain,évoluant);
augmenter(être_humain,impliquant_action);

avaler_qqch(locuteur,est_comestible);

avertir(être_humain,être_humain);

avoir_qqch(locuteur,DiversesActivités);
avoir_qqch(locuteur,est_faisable);
avoir_qqch(locuteur,est_matériel);
avoir_qqch(locuteur,Aliments);
avoir_qqch(locuteur,OrganisationSociale);
avoir_qqch(locuteur,QuantitéQualitéImportance);
avoir_qqch(locuteur,RelationsHumaines);
avoir_qqch(locuteur,Sensations);

baisser(locuteur,est_mobile) ;

boire_qqch(locuteur,AlimentsLiquides);

bouger(locuteur,est_mobile);

chercher(être_humain,est_visible);

choisir_qqch(être_humain,valeur_morale);
choisir_qqch(être_humain,est_intelligible);
choisir_qqch(être_humain,JeuxSports);

claquer(locuteur,qui_claque) ;

comprendre_qqch(être_humain,est_intelligible);

compter_sur(être_humain,être_humain);
compter_sur(être_humain,MoralePsychologie);
compter_sur(être_humain,RelationsHumaines);
compter_sur(être_humain,Thérapie);

connaître(être_humain,valeur_morale);
connaître(être_humain,est_intelligible);
connaître(être_humain,QuantitéQualitéImportance);
connaître(être_humain,Société);

conseiller(être_humain,être_humain) ;
```

```
constater_qqch(être_humain,valeur_morale);
constater_qqch(être_humain,QuantitéQualitéImportance);
constater_qqch(être_humain,Société);

consulter(être_humain,être_humain);

convenir(ou(Aliments,Repas,une_situation,Thérapie),locuteur) ;

couper(locuteur,est_divisable);

croire_qqch(être_humain,être_humain);

déboucher(locuteur,est_contenant) ;

décourager(être_humain,être_humain);

découvrir(être_humain,ActivitéIntellectuelle);
découvrir(être_humain,MoralePsychologie);

définir(être_humain,ActivitéIntellectuelle);
définir(être_humain,Espace);
définir(être_humain,QuantitéQualitéImportance);

dépendre_de(être_humain,être_humain);

déranger(être_humain,être_humain);

descendre(locuteur,dénivellation) ;

écouter_qqun(être_humain,est_audible);

effrayer(ou(est_négatif,ActivitéVitale,MoralePsychologie,OrganisationSoc
  iale,être_humain),être_humain);

ennuyer(être_humain,être_humain);

étouffer(Aliments,locuteur);

éviter_qqch(être_humain,être_humain);
éviter_qqch(être_humain,Mental);

faire_qqch(locuteur,est_faisable) ;

fermer(locuteur,se_fermant);

géner(est_négatif,locuteur);

gonfler(locuteur,est_gonflable) ;
```

```
laver(locuteur,est_matériel);
laver(locuteur,AlimentsSolides);

manger_qqch(locuteur,AlimentsSolides);

manquer_à(ou(Géographie,est_usuel,valeur_morale,MaladieSoins,Organisatio
  nSociale),locuteur);
manquer_de(être_humain,valeur_morale);
manquer_de(être_humain,est_usuel);
manquer_de(être_humain,être_humain);
manquer_de(être_humain,Géographie);
manquer_de(être_humain,MaladieSoins);
manquer_de(être_humain,OrganisationSociale);

mentir_à(être_humain,être_humain);

nettoyer(locuteur,est_usuel) ;

parler_de(être_humain,ActesGestes) ;
parler_de(être_humain,ActivitéVitale) ;
parler_de(être_humain,EspaceTempsLogique);
parler_de(être_humain,être_humain) ;
parler_de(être_humain,Mental) ;
parler_de(être_humain,Société) ;

prendre_qqch(locuteur,est_matériel);
prendre_qqch(locuteur,Alimentation);

profiter_de(être_humain,est_destructible);
profiter_de(être_humain,Géographie);
profiter_de(être_humain,RelationsHumaines);

questionner(être_humain,être_humain);

recevoir(être_humain,est_recevable);

remercier(être_humain,être_humain);

répondre_à(être_humain,est_audible) ;

respecter(être_humain,valeur_morale) ;
respecter(être_humain,être_humain) ;

ressentir(locuteur,Maladies);
ressentir(locuteur,Sensations);

savoir_qqch(être_humain,valeur_morale) ;
```

```
savoir_qqch(être_humain,QuantitéQualitéImportance) ;
savoir_qqch(être_humain,Société) ;

souffrir_de(locuteur,Anatomie);
souffrir_de(locuteur,AutresMaladies);

suivre(locuteur,Thérapie) ;

utiliser(locuteur,est_usuel) ;

voir(être_humain,est_visible);

/* verbe pron1 */

sadapter(être_humain);
se_détendre(être_humain);
se_laver(locuteur) ;
se_paralyser(est_mobile);

/* verbe2 */

donner(être_humain,RelationLogique,être_humain);
donner(non_locuteur,RelationsHumaines,être_humain) ;
donner(être_humain,RelationsHumaines,être_humain) ;

parler_de_à(être_humain,ActesGestes,être_humain) ;
parler_de_à(être_humain,ActivitéVitale,être_humain) ;
parler_de_à(être_humain,EspaceTempsLogique,être_humain);
parler_de_à(être_humain,être_humain,être_humain) ;
parler_de_à(être_humain,Mental,être_humain) ;
parler_de_à(être_humain,Société,être_humain) ;

présenter(être_humain,est_recevable,être_humain);

/* verbe pron 2 */

se_laver_qqch(locuteur,est_lavable) ;

sadapter_à(être_humain,OrganisationSociale) ;
sadapter_à(être_humain,être_humain) ;
```

French dictionary

Prepositions and articles
au
aux
du
des

Articles
un
une
des
le
la
l'
les

Prepositions (8)
à
contre
de
d'
en
par
sous
sur

Conjunctions
que
qu'

Adverbs and negation particles (15)
encore
guère
jamais
n'
ne
non
parfois
pas
pas encore
pas souvent
pas toujours
plus
plus souvent
souvent
toujours

Subject pronouns
je
j'
nous
tu
vous
il
elle
ils
elles
me
m'
nous
te
t'
vous

Reflexive and object pronouns
se
s'
le
l'
me
m'
nous
te
t'
vous
lui
leur
en
me
m'
nous
te
t'
vous
le
la
l'
les

Accentuated pronouns
moi
nous
toi
vous
lui
elle
eux
elles

Demonstrative pronouns
ce
cet
cette
ces

Possessive pronouns
mon
ma
mes
notre
nos
ton
ta
tes
votre
vos
son
sa
ses
leur
leurs

Operator verbs (34)
accepter
admettre
aimer
apprécier
arriver
attendre
avoir
cesser
considérer
constater
continuer
croire
désirer
détester
devoir
envisager
espérer
être
éviter
parvenir
penser
pouvoir
préférer
proposer
réaliser
refuser
savoir
sentir
souhaiter
supposer
vouloir

Verbs taking noun phrases as their complements (121)
absorber
accepter
activer

adapter
adapter
admettre
agripper
aller
amuser
anéantir
apparaître
apprécier
arrêter
attraper
augmenter
augmenter
avaler
avertir
avoir
baisser
boire
bouger
bredouiller
chercher
choisir
claquer
comprendre
compter
connaître
conseiller
constater
consulter
convenir
couper
croire
déboucher
décourager
découvrir
définir
déglutir
dépendre
déranger
descendre
détendre
déterminer
diminuer
discuter
disparaître
donner
dormir
douter
écouter
écrire
effrayer
empoigner
ennuyer
espérer
étouffer
éviter
faire
fermer
garder
géner
gonfler
guérir
incliner
inquiéter
lever
mâcher
manger
manquer
mastiquer
mentir
mériter
mourir
nettoyer
obtenir
ouvrir
paralyser
parler
pencher
penser
plaire
pleurer
plier
prendre
présenter
profiter
prolonger
questionner
rassurer
réagir
réaliser
reboucher
recevoir
recommencer
redresser
refermer
réfléchir
remercier
rencontrer
répondre
respecter
ressentir
revenir
revoir
révolter
rincer
savoir
siffler
souffler
souffrir
soulever
sourire
souvenir
suivre
tenir
tourner
utiliser
vivre
voir

Predicatives (participles, adjectives etc.) (59)

accessible
adapté
attristé
autonome
averti
ballonné
beaucoup de difficultés
beaucoup de mal
beaucoup de peine
besoin
bien
bon
bouché
capable
chaud
confronté
content
contraint
critique
décidé
des difficultés
déterminé
dissimulé
du mal
efficace
envie
en colère
en état
faible
faim
fermé
fort
fréquent
froid
frustré
important
incapable
insuffisant
inutile
irritable
la possibilité
lourd
mal
mauvais
oppressé
passif
persuadé
peur
proche
propre
recommandé
sale
soif
soucieux

stressé
sujet
triste
utile
vide

Common nouns (305)

aide
air
amélioration
amour
ampoule
amusement
angoisse
annulaire
anorexie
apparence
appétit
appréhension
articulation
aspect
asphyxie
aspirine
assistance
asthénie
atteinte
attitude
auriculaire
autonomie
avant bras
baignoire
baisse
boisson
bol
bonheur
bouteille
bouton
bras
brosse
brosse à dents
buste
cachet
chaise
chaleur
cheveu
cheville
circulation sanguine
clavicule
coccyx
coeur
coliques
colonne vertébrale
commodité
comportement
condition
confiance
confort
convulsion
cou
coude
courage
couteau
crampe
crâne
critique
cuillère
cuisse
date
date limite
défaut
déjeuner
délai
dent
dentier
désir
diarrhée
différence
difficulté
dîner
discussion
doigt
dose
dos
douche
douleur
dysarthrie
dysphagie
dysphonie
dyspnée
eau
émotion
énergie
ennui
épaule
épreuve
escalier
espoir
estomac
état
état d'âme
examen
excuse
faiblesse
fausse route
fenêtre
fesse
fièvre
figure
flacon
foi
foie
force
fourchette
fruit
frustration
genou
gorge
gouttes
goût
gratitude
gymnastique corrective
gymnastique respiratoire
habitude
haleine
hanche
honte
humeur
humilité
hypertension artérielle
hypotension artérielle
impression
index
information
injustice
intérêt
jambe
joie
joue
jugement
langage
langue
larme
lavabo
laxatif
légumes
lèvre
liberté
limite
lit
lombes
lourdeur
lunettes
main
majeur
maladie
malaise
malheur
mal au coeur
mal au dos
mal au ventre
mal à la tête
mal de coeur
mal de dos
mal de tête
mal de ventre
marche
médecin
médicament
mensonge
message
métabolisme
mollet
morale
mort

mouvement
mucosité nasale
murmurs
muscle
muscle de la mâchoire
muscle du cou
muscle du visage
mutisme
mystère
nature
nerf sciatique
nez
nuque
objectif
occasion
odorat
oeil
ongles des mains
ongles des pieds
ordonnance
oreille
orteil
pain
parole
paupière
peau
peigne
peine
pensée
perfusion
période
perspective
perte
petit déjeuner
phrase
pied
pilule
piqûre
plaisir
plante des pieds
poignet
poitrine
porte
possibilité
pot de crème
pouce
pouls
poumon
présence
pression sanguine
pression vasculaire
prétexte
principe
problème
professeur
progression
promesse
qualité
question
radiographie
raison
raisonnement
réalité
réconfort
rééducation
relation
repas
réponse
repos
requête
respect
respiration
responsabilité
résultat
retard
rideau
rigidité
robinet
robinet d'eau chaude
robinet d'eau froide
sacrum
salive
savonnette
savon
séance d'acuponcture
séance d'ergothérapie
séance d'orthophonie
séance de massage
sel
signification
silence
sinus
situation
solitude
solution
souci
souhait
souper
soutien
spasme
stress
succès
sucre
sueur
suggestion
symptôme
talon
tasse
tendon
tension
terme
tête
toilettes
torse
toux
traitement
transpiration
tronc
trouble
troubles du sommeil
troubles sexuels
tube de dentifrice
urgence
valeur
ventre
vérité
verre
version
vertèbres cervicales
vessie
viande
vie
visage
visite
volet
volonté

Punctuation marks (2)

.
?