Recent Trends of Electronic Dictionary Research and Development in Europe

Gilles Sérasset[1]
GETA-IMAG (UJF & CNRS)
BP 53X
38041 Grenoble Cedex
FRANCE

[1] This report has been written in cooperation with the Japan Electronic Dictionary Research Institute Ltd; Mita Kokusai Building Annex, 4-28, Mita 1-chome; Minato-ku, Tokyo 108, Japan.

Contents

Recent Trends of Electronic Dictionary Research and Development in Europe

I. Introduction

Recent developments in Natural Language Processing have highlighted the lack of multi-usage lexical resources. Reusable lexical resources should noticeably reduce the cost of development of Natural Language Processing (NLP) applications. This reusability can be reached by using generic lexical databases as lexical source for NLP applications. This approach also allows easier management of lexical resources. As a result, many projects are conducted in several parts of the world in order to study and try to solve the problem of lexical resources for Natural Language Processing.

The first aim of this report is to make a survey of existing European projects and developments concerning electronic dictionaries. Different approaches have been selected by different projects, some of them focussing on multilingual aspects, others proposing solutions for monolingual aspects.

The second aim of this report is to compare the EDR project in Japan with the major European projects concerning electronic dictionaries.

Among these projects, we will focus on the Multilex (multilingual aspects) and Genelex (monolingual aspects) projects. We also present “Le Lexicaliste," a commercial software for monolingual lexical database management.

This report begins with a presentation of the European lexical database projects. We distinguish the monolingual and the multilingual approaches. Then, we present the different functionalities a lexical workstation should provide. Examples of such functionalities, as seen in different European projects, are given whenever possible. Finally, we propose a systematic comparison between the EDR, Multilex and Genelex projects.

We give in annex 3 detailed analytic reports concerning Multilex, Genelex and Le Lexicaliste.

II. Lexical Databases in Europe

Among lexical databases, we distinguish three classes of systems, “classical” electronic dictionaries, “specialized” systems and “multi-usage” systems. We define each class of systems and present some of the research projects or commercial software packages that belong to the class.

1. “Classical” electronic dictionaries

This class groups dictionaries that are dedicated to human use. Among these dictionaries, we find classical paper dictionaries made available in electronic format and electronic dictionaries for human use in general.

1.1. Acquila

Acquila is a management software for multilingual terminological databases running on PCs. It deals with 2 language simultaneously and can manage 14 languages.

Each term is associated with its linguistic environment (source language, target language) and a context (domain, activity of the company, name of the company).

1.2. Collins On-Line

Collins is a well-known dictionary publisher. It proposes an on-line version of its bilingual dictionaries: Collins On-Line, on PC.

Six pairs of languages (bilingual) are available. Users are allowed to write and store personal notes and phrases.

1.3. Dicologique

Dicologique is an interactive dictionary on PC. It gives access to 100,000 words in its French version.

Based on a semantic network linking different concepts and on a domain hierarchy, it gives access to the database from a domain, a lemma or a quasi-definition (e.g. Horse and (Red not Orange) will return “rouan” and “aubère” (2 kinds of red horses)).

Dicologique is distributed by MEMODATA, 23 rue des Boutiques, 14000 Caen, France, Tel: (+33) 31 95 05 08, fax: (+33) 31 95 54 21.

1.4. The “Trésor de la Langue Française” (TLF)

The Treasure of French Language is a textual database developed by the National Institute of French Language (INALF) of CNRS. This textual database stores 2,500 texts in French. 80% of these texts are literary texts, the other 20% are scientific texts from many different domains. These texts cover the period from XVIIth to XXth century. This represents a total of 170 million words.

The TLF database is accessible on-line through the network. Only 30% of the TLF is available to the public (due to copyright reasons).

1.5. Multiterm

Multiterm is a multilingual terminological database management system. It is flexible enough to allow the user to define its entry format (up to 16 languages and 80 attributes).

This system allows the user to associate 2 pages of text to each entry. Cross-references can also be defined.

This system is commercialized by INK, 107 avenue Parmentier, 75011 Paris, Tel: (+33 1) 40 21 65 65, Fax: (+33 1) 40 21 07 10.

"Classical" electronic dictionaries are dedicated to the human use. This class of dictionaries seems rather mature now, and many commercial software packages are available. Ongoing research in this area aims at simplifying access to the information.

2. "Specialized" systems

"Specialized" systems are electronic dictionaries developed as lexical knowledge sources for certain applications. Some are used only by one kind of application, while others may be shared by several applications.

2.1. Ariane

GETA has developed a Machine Translation (MT) system generator named Ariane. It appeared quickly that the dictionary management problem was a crucial point to handle. Hence, some tools have been developed to make dictionary management easier.

Each MT system developed under the Ariane system has its own dictionary, which is scattered in various "components" (morphological analysis, lexical transfer, morphological generation). These dictionaries are written in Ariane's specialized languages (ATEF, TRANSF, SYGMOR), that is, using special formats and syntax. A tool (Visulex) has been developed to allow the visualization and the modification of the lexical information of a given MT system written in Ariane. It is limited in that it allows lexicographers to manipulate the dictionaries of only one language pair at a time.

In this database, a lexical unit consists of a family of words that are inter-related by lexico-semantic functions. For example, the lemmas "to construct", "construction", "reconstruction", and "constructive" all belong to the same lexical unit, usually named "construct-V". Visulex is a first approach towards factorization of lexical resources, but the format used is very close to the format of the application.

2.2. BDTAO

B'VITAL has developed a lexical database from which Ariane's dictionaries may be automatically produced. This database is specialized for MT, but is independent of the particular MT system at hand, and is reversible (completely in analysis and generation, and for terminology only in transfer).

This database handles "fork" dictionaries (1 source language --> n target languages). The lexical unit is a family of words that are related by lexico-semantic functions, as above. The structure of a lexical unit is a flat attribute value structure. It is separated in 2 sections. The first section contains monolingual information on the lexical unit. The second section contains the different translations of the lexical unit in several languages.

2.3. BDLex

BDLex is a research project conducted by the IRIT laboratory of the university of Toulouse. This project aims at the construction of a monolingual lexical database for French.

The lexical unit is the lemma. BDLex has now 25,000 entries (i.e. about 300,000 inflected forms). The structure contains morphological and phonological information.

Here are four BDLex entries:

lemma	HG	PHON	FPH	HP	CL_PHON	NS	F	CS	GN	CF
nabab	11	/nA/bAb		11	/NA/DAD	2		N	Mn	01
nabi	11	/nA/bi		11	/NA/DI	2		N	Mn	01
nabot	11	/nA/bo	t"	11	/NA/DE	2		N	gn	01
nacelle	11	/nA/s&l	e	11	/NA/SEL	2		N	Fn	81

where the abbreviations are: HG: homographic number, PHON: phonetic, FPH: phonetic ending, HP: Homophonic number, CL_PHON: phonetic class, NS: number of syllables, F: frequency, CS: syntactic class, GN: variation in gender and number and CF: flexional class.

2.4. METAL

For its MT generator system METAL 3.0, SIEMENS has developed some tools to handle the manipulation of dictionaries. There are 2 types of lexicons in METAL 3.0: monolingual lexicons and transfer lexicons. The features in a transfer lexicon are fixed across all language pairs. In monolingual lexicons, the features are defined on a per language basis through a specification file.

A lexicon entry consists of a set of features and their values. The value of a feature can be of any of the following types:

INTEGER	Any number in the range of -228+1 and 228-1.
STRING	A sequence of characters in double quotes, e.g. “lexicon”.
SYMBOL	A Lisp symbol
BOOLEAN	Either the symbol T or NIL.
SET	A non-duplicating, non ordered sequence of INTEGERs, STRINGs and/or SYMBOLs.
LIST	A list is an ordered sequence of INTEGERs, SYMBOLs, STRINGs and/or LISTs.

The user can define his/her own structures through a specification file. This file contains 2 basic parts, feature-value definitions, and lexical category definitions. The syntax of the definitions is a Lisp syntax. As an example, we give the definition of a feature and of a lexical category.

The first part (the `defeat` macro) defines a feature called `a-cl` that is a set with 6 possible values and a maximum of 7 possible combinations of values.

The second part (the `defcat` macro) defines a lexical category name `det` that contains one required feature (`ca`) and 3 optional features (`a-cl`, `abb` and `plc`).

```
(defeat a-cl :set
             :values (FMNP1 FMNP2 FMNP3 FMNPS1 FMNPS2 FMNPS3)
             :allocate 7
             :pretty-name “Anaphoric class”)
```

```
(defcat det :optional (A-CL FMNP1 FMNP2 FMNPS3)
            :optional ABB
            :required CA
            :optional PLC
            :default (PLC NF)
            :check-consistency t
            :pretty-name “Determiner”)
```

2.5. LADL dictionaries

LADL (Université Marne la vallée) has developed DELAF, an inflected form dictionary for simple words containing 600,000 forms and DELACF, an inflected form dictionary for compound words containing 150,000 forms.

In these dictionaries, the entries are represented as finite state automata. This allows a powerful parsing of texts. A simple entry is represented as an automaton containing the morphological and syntactical information (see example below).

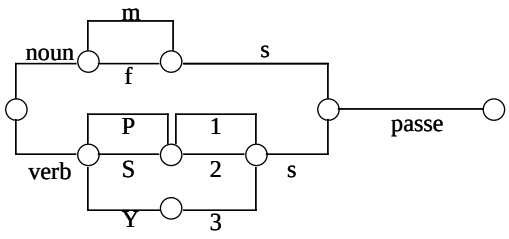


Figure 1: The automaton representation of the inflected French form “passe”.
The abbreviations are m: masculine; f: feminine; s: singular; 1,2,3: nth person; P: present; S: subjunctive and Y: imperative

3. Multi-usage systems

Multi-usage systems are systems that have been developed without any particular application in sight. These systems can be used by different applications (MT systems, language understanding systems, grammar checker, …).

These systems are rather new and, generally, in the state of research. We will present 4 of those systems. Three of them are research projects, while the fourth is a commercial software. We list them in chronological order.

3.1. Acquilex

Acquilex is a CEC-DG XIII-ESPRIT project. It is concerned with the acquisition of lexical knowledge for natural language processing systems, semi-automatically from machine-readable versions of conventional dictionaries for English, Spanish, Italian and Dutch.

The Acquilex project involves the following partners:
- University of Cambridge, United Kingdom,
- University of Amsterdam, The Netherlands,
- Universitat Politecnica de Catalunya, Bacelona, Spain,
- Instituto di Linguistica Computazionale, Pisa, Italy,
- Universita di Pisa, Italy,
- Cambridge university press, United Kingdom,
- Bibliograf, Spain,
- Van Dale Lexicografie, The Netherlands.

Work on Acquilex can be divided in 2 areas: the development of a methodology and the construction of software tools to create lexical database from machine readable dictionaries and the subsequent construction of illustrative theoretically-motivated, lexical knowledge base fragments from the databases, using a further set of software tools designed to integrate, enrich and formalize the database information.

In Acquilex dictionaries, the lexical unit is a word-sense.

Acquilex has chosen to encode the lexicon as a Typed Features Structures (TFS) system. The TFS system allows the lexicographer to define the valid features and their values. The inheritance mechanism allows the factorization of information on units of the same class.

Each lexical unit (word sense) is defined by its morphology, syntax and semantic. The semantic description of the lexical unit is rather complex. A lot of information is present in the model. For example, the word “water” will contain a description of its constituents (oxygen and hydrogen), its taste, its degree of alcohol, its physical state,…

3.2. Genelex

GENELEX is a Eureka project involving French, Italian and Spanish partners.

The objective of Genelex (GENEric LEXicon) is to construct a generic dictionary in various European languages (for now, French, Italian and Spanish). This involves not only the development of a generic dictionary but also the definition of a strategy guaranteeing that it will remain generic.

The dictionary should be considered as a large lexical database, with no direct connection to any NLP application. The lexicons dedicated to applications will be obtained by the extraction of the data designed from the generic dictionary in a form adapted to needs.

For Genelex, A lexical Unit is a word-sense, defined by the relations between a morphological unit, a syntactical unit and a semantic unit.

Genelex has chosen to encode the dictionary in an entity-relationship format. This allows the visualization of a lexical unit as a graph. It also allows to place any element of information on an equal footing (i.e. no node is privileged for information retrieval, whereas a tree structure or a Typed Feature Structure privileges the root).

This project is presented more in details in annex A.2.

3.3. Le lexicaliste

Le lexicaliste is a dictionary generator commercialized by SITE-EUROLANG. It was originally developed to meet in-house needs for large translation and document processing jobs.

Le lexicaliste is generic. Lexical entries can be imported from various sources (other dictionaries, files, term lists,…) and defined on the basis of multiple attribute criteria within a very rich set of possible linguistic data structures. When lexical data is required for a given application, a dictionary can be generated and exported to that application in Standard Generalized Markup Language (SGML) format. Le lexicaliste runs under the Oracle RDBMS on Sun workstations with X/Windows Motif user interface.

Le lexicaliste provides a way to define the desired structures through a description of the attributes. It also allows users to define a lexical and a semantic network. When the structure is defined, the user can write constraints and default rules with a 4th Generation Language (4GL).

Le lexicaliste also provides a friendly user interface that contains hypertext tools for navigation within the dictionary. The user can easily browse the dictionary by accessing a word or navigating in the lexical and semantic networks.

This project is presented in more detail in annex A.3.

3.4. Multilex

MULTILEX is a CEC - DG XIII - ESPRIT project. The aim of this project is to define standards for multilingual lexical databases. The planned duration is 3 years and is being executed in 2 phases with 2 coordinators. Languages considered are essentially those of the European Community.

There are different standards to propose: standards on linguistic information (what is in the database) and architecture (how the entries, the dictionaries,…, are organized), standards on tools (how to manipulate the database), logical structure (how the information is represented) and software architecture (how the tools are organized).

The first phase aimed at the definition of the different standards (linguistic standard, representation formalism, definition and prototyping of tools). The second phase will be more application-oriented.

During the first phase of the project, Multilex has performed the following tasks:
- Definition/simulation of the standards,
- Prototyping of some tools,
- Coding of some entries (English, French, German and Italian).

For Multilex, the lexical unit is a word sense.

The architecture of a Multilex database is based on monolingual and bilingual dictionaries.

Lexical units are represented with a formalism based on Typed Feature Structures. A language has been defined in order to describe the structure of the lexical units. Another one allows the lexicographer to code coherence and integrity rules. Every operation on lexical units is done in this formalism. After that, the units are stored in a relational database.

Multilex has also developed a linguistic standard for the representation of lexical units of European languages. This standard appears as the maximal set of information that is common to the European languages.

Some tools have been defined (browser/editor, integrity checker, defaulter, transcriptors and Import/Export tool). Some of them have been prototyped.

This project is presented more in detail in annex A.1.

III. Lexical Workstations in Europe

In this paragraph, we present some lexical databases developed in Europe. These lexical workstations are parts of the lexical database projects presented before.

We have divided the projects in 2 classes: monolingual projects and bilingual projects.

First, we present the limitations of our study. Then, we will present the different projects.

We explain the choice of the projects and we give the different functionalities we take into account in our study. Then, we will present each of the projects.

1. Limitations of the study

In this chapter, we explain the choice of the projects and we give the different functionalities we take into account in our study.

1.1. Choice of the projects

We will present 4 projects, two of them handling monolingual information, the others handling multilingual information.

Some tools have been developed by the university of Pisa within different projects. These tools can be conceived as different modules of a lexicographic workstation. We have chosen to present this lexicographic workstation, because it provides a complete view of a lexical workstation, with access to textual corpora, creation of and access to a Lexical Database System (LDB) and integrated access to texts and dictionaries.

Le lexicaliste provides many functionalities that make it a real lexical workstation. It does not provide access to lexical corpora, but it is easy to use and provides a very efficient editing environment. We chose to present this project because of its very powerful environment for editing, browsing, defaulting and coherence checking.

METAL, the MT system developed by Siemens, contains a multilingual database. We have chosen to present this part of METAL because of its very powerful defaulting mechanism.

Multilex has defined a whole set of tools for a lexical workstation. Some of these tools have been prototyped. We have chosen to present this project because it provides rather complete specifications for a great variety of tools.

We could have chosen the Genelex project to illustrate functionalities of lexical workstations. Such a workstation has been developed that presents the entries in a graphic format. Unfortunately, we have not seen any demonstration of this workstation.

We could also have presented the Acquilex project, but the majority of the tools provided by Acquilex is part of the lexical workstation developed at Pisa University.

These 4 projects are quite representative of the current state of European Research and Development in lexical data bases.

1.2. Functionalities

In this part, we present the main functionalities a lexicographic workstation must provide.

The most important functionality is, of course, an **editing/browsing** tool. All projects provide this kind of tool, but with different approaches. For all projects, we will look into the possibilities offered by this tool (kind of access, kind of editing methods).

A **defaulter** is also an important part of the workstation, as it facilitates the editing of an entry. We will see what kind of default is possible for each project.

An **integrity and coherence checker** is a very powerful tool for database maintenance. We will see the different tools provided by the projects and the way to define the coherence rules.

A **transcriptor** allows the database to store multilingual data in different writing systems without worrying about the type of DataBase Management System (DBMS) which is used. It allows the manipulation of multilingual structure, using a standard character set and the rendering of the information in the appropriate writing system.

Import/export tools are also very important. Obviously, a lexical database is of no use without an export tool. An import tool allows the use of existing lexical resources for the database.

Access tools to textual corpora help the lexicographer in determining the different uses of lexical units and in collecting examples. It also allows a semi-automatic extraction of lexical information from the texts.

2. Monolingual lexical workstations

2.1. PISA

Pisa university has developed an integrated system mainly based on two prototypes: a textual database system and a lexical database system.

2.1.1. Access to textual corpora

The textual database system stores and handles very large corpora, which are all available on-line. Basic queries can be addressed to the texts at the word-form level. The system can:

- find a given word
- compute its frequency
- display word-contexts of a specified length
- search for co-occurrences, including those of non-adjacent words
- expand a context to a whole paragraph.

2.1.2. Creation of a lexical database

Creation of a LDB can be obtained in two different ways: inputting data by a lexicographer, acquiring data from machine readable dictionaries. Both these acquisition or creation strategies are based on a preliminary definition of the structure of the lexical entry.

When acquiring data from one or more existing dictionaries, the template serves first of all as a guide for the “parsing” procedure of the dictionaries, and can be used as a basis for exchange, for comparison and for unification of data coming from different sources.

When data entering or updating is done interactively by the lexicographer, a procedure has been implemented which allows input and update in a very friendly way. In this case, the definition has the function of guiding the lexicographer by giving prompts on fields to be filled.

The Pisa workstation does not provide tools for consistency checking and defaulting.

2.1.3. Access to a lexical database

The Italian lexical database is at present composed of 2 monolingual Italian dictionaries and one bilingual dictionary (Italian-English).

There are many access functions which allow the user to query a database from many different perspectives. Among these functions, we can find modules that allow to put queries through different filters and which create and retrieve taxonomies in a lexical database.

2.1.4. Integrated access to texts and dictionaries

The possibilities offered by the lexical database of retrieving sets of words on the basis either of their morphology or of their belonging to a certain semantic class (defined by means of a given hyperonym) were exploited to allow more sophisticated queries to the corpus of texts in the textual database. Thus, the textual database and the lexical database are linked as 2 modules of a more powerful system. As a result one can submit queries to the corpus of texts, through the lexicon. For example:

- Using the morphological engine of the lexical database, one can retrieve all occurrences of the inflected word-forms of a given lemma.
- One can also retrieve all occurrences of all hyponyms of a given term.

2.2. Le lexicaliste

Le lexicaliste provides a very user-friendly lexical workstation. Although it does not include access to a textual database, it provides a complete set of tools for a lexicographer.

2.2.1. Creation of a dictionary

The set of attributes and relations representing the information for articles and lexical and semantic networks are defined in a system of reference.

Everybody can read this system of reference, but only the person responsible for the dictionary can modify it.

The system of reference stores the declaration of the attributes and relations with their respective properties. For example, one can declare that `transitivity`:

- is an attribute
- is of a scalar type
- admits two values: `transitive` and `intransitive`
- has only one value at a time
- applies only to verbs.

In the system of reference, the different objects are grouped in 5 classes:

- attributes of a lemma (e.g.: `category`)
- attributes of a word sense (e.g.: `transitivity`, `definition`)
- attributes of flexional rules (e.g.: `number`, `gender`)
- lexical relations (e.g.: `abbreviation`)
- semantic relations (e.g.: `hyperonymy`).

A graphical interface gives access to:

- the list of attributes and relations
- the description of an attribute or a relation
- the list of possible values of an attribute
- the description of an attribute’s value.

A modification of the system of reference is not an usual operation during the article management phase.

There are 2 ways to create an article interactively:
- from an existing word-sense with a cut and paste operation
- by giving a key, a category and a type.

2.2.2. Coherence checking and defaulting

Some values are interdependent. For example, if a verb is pronominal, its auxiliary is the verb “être” (to be). The administrator of the dictionary can express these interdependencies using a 4GL (4th Generation Language).

These interdependencies can be seen as constraints always to be verified. In that purpose, the administrator gives an error message to be displayed if a constraint is not verified.

These interdependencies can also be seen as defaulting rules. In that case, the default value is automatically generated during the creation of an entry, depending on the values already stated.

2.2.3. Access to a dictionary

One can consult the dictionary by accessing one of its articles. There are 4 different ways to have access to an article:
- direct access by the lemma
- indirect access from a concept, via the semantic network
- indirect access from another article via the lexical network
- indirect access from any word found in any textual field (hypertext relation).

When an article is selected, the senses of the words are displayed. One can look at all information associated to each sense. One can also browse the database via the lexical or the semantic networks, or via hypertext relations (by selecting a word or a set of words in a textual field and searching another article from these words).

2.2.4. Import/export tools

2.2.4.1. Importing

The importation mechanism is fully parametrizable. We can distinguish 2 cases:
- importing articles
- importing lexical or semantic network.

If an imported article already exists in the dictionary, The old article found in the dictionary is copied in a trace file and replaced by the imported one (if the imported article is coherent).

If an imported article does not exist in the dictionary, it is simply created.

If an imported link exists, the direct and inverse links are deleted from the dictionary, the new link is inserted and its inverse is automatically generated.

If an imported link does not exist in the dictionary, it is created and its inverse is automatically generated.

A concept that appears in a semantic network and that is not in the dictionary is automatically created.

The source SGML file must be consistent with the Lexicaliste's DTD (Document Type Definition). Consistence is checked by Le Lexicaliste during import.

2.2.4.2. Exporting

The data is exported to an SGML file. Filters can be defined by the administrator for the export mechanism.

The user can export either the articles, or the lexical or semantic networks.

3. Multilingual lexical workstations

3.1. METAL

For its MT generator system METAL 3.0, SIEMENS has developed some tools to handle the manipulation of dictionaries. Among these tools, we focus on the defaulting mechanism that is very interesting.

The defaulter consists of one language independent module, called "basic" and several language specific modules, one for each language. This facilitates maintenance and modification of the system as well as extension of additional languages.

3.1.1. Dictionary look-up

Before a canonical form is defaulted, the word is always looked up in the lexicon under a given category and language. If the word is found in the database, the entry is retrieved and no defaulting takes place.

3.1.2. Separator analysis

If a separator is found in the string (a blank or a hyphen for instance), the word following the separator is looked up in the database under the given category and language. If an entry is found, then the new entry is derived from the one found in the database.

3.1.3. Prefix analysis

This function is comparable to the one described above except that it searches the canonical form for a prefix that has been defined as legal for the given language and category. It is called if the canonical form does not have a separator.

3.1.4. Suffix analysis

If no separator or prefix can be found, then the canonical form ending is analyzed by a function that compares it with a table of endings. Searching the table of endings is done sequentially. When the first match is encountered, the corresponding lexical unit and feature value combination are generated.

The table of endings is sorted in order to apply the most specific ending-pattern first. Joker characters are allowed in an ending pattern.

3.1.5. Standard entries table

If no string or pattern matches the canonical form ending, then the word is defaulted according to a standard table of entries. This table contains default entries for all categories of the given language.

3.1.6. Language specific tables

The heuristics described above are applied successively to the canonical form to be defaulted. They are basically the same for all languages and categories, although the exact content for tables will differ across languages and categories. In addition to these general heuristics, the defaulter has tables that are used for handling phenomena that are specific to only one language, or even only one category of a language.

3.2. Multilex

The Multilex consortium has prototyped some tools. We will not discuss these prototypes, but rather present the tools that have been completely specified by Multilex.

Multilex only provides tools for a lexical database. It has not planned to include tools for a textual database.

This project is presented in more detail in annex A.1.

3.2.1. Editor / browser

The editor will visualize the needed features and their possible values. A user-friendly interface will be used with e.g. check boxes for boolean features, pop-up menus for features with a fixed set of possible values,…

It will check information and try to detect errors as soon as possible allowing the lexicographer to correct them.

The tool itself will expand some information to a full attribute/value list and will solve abbreviated information, generate defaults when no input is given and carry out the inheritance rules.

The scope of a dictionary browsing tool is to provide the user with fast and flexible access to the lexical information. This implies facilities to navigate through the different fields which constitute an article in order to retrieve, view and manipulate information of interest in whatever part of the dictionary it is stored. Ideally, in order to optimize the consultation process, the user will also have access to lemmatisation procedures.

The browser proposes a series of functions which can be used to have access to the dictionary. Any information contained in the dictionary should be usable as a key to access other information.

The data query language used must be easy to formulate and as close as possible to natural language. It must enable the user to formulate an unrestricted variety of queries, to specify conditions on attributes of the entries, to select only the desired attributes and to format answers to queries.

The user can simply look up lexical items (single word forms or all the forms of a lemma in a monolingual dictionary, select translations, and display target language information in a multilingual dictionary,…). The browser provides simple search functions and allows the user to define his own search function.

3.2.2. Defaulter

The defaulter produces default lexicon entries to be proof-read and post-edited by the human coder. The default entries are derived by various heuristics, using general linguistic knowledge (lexical rules) and knowledge retrieved from entries existing in the lexical database. The defaulter mainly supports the coding task to be performed by a human coder. It

does so by defaulting lexical information for words to be added to the lexical database. It is used before the start of the real translation task.

The defaulter mechanism selected by Multilex is the same as for METAL's defaulter.

3.2.3. Integrity and coherence checker

Each time a lexical unit is added or modified, the system will have to check some coherence and integrity constraints.

A *constraint* is a rule defined by the lexicographer. Activated contraint sets must be verified before and after each transaction.

There are different levels of constraint:

1. *Warning*: constraints of level 1 are just warnings: when the constraint is overridden, a message is passed to the lexicographer, but all treatments are allowed. The warning disappears as soon as the lexicographer validates the entry. These constraints can be used to detect potential errors.

2. *Delay*: constraints of level 2 can be overridden, but, in that case, the lexicographer receives a message and some treatments are forbidden on the concerned entries. An extraction request will run properly, but the detected entries will not appear in the extracted file. Interactive treatments such as browsing and editing are allowed. These contraints can be used to handle temporarily incomplete entries.

3. *Critical*: constraints of level 3 can't be overriden. If a transaction overrides such a constraint, it will be cancelled (rollback) (all necessary information will be saved in a log to allow debugging).

There are different kinds of constraint:

1. *Integrity* rules apply to an article of the lexical database. They ensure that none of the article of the lexical database have an ill-formed configuration.

2. *Local coherence* rules apply to different articles of the same dictionary. They ensure that the dictionary is coherent.

3. *Global coherence* rules apply to different articles of different dictionaries of the lexical database. They ensure some coherence between dictionaries (if a sense is present in a monolingual dictionary, it must be an entry of bilingual dictionaries). This kind of constraint will be developed in another chapter.

Constraints have 3 main parts:
- an extraction expression which specifies the objects of the database that are concerned with the given constraint,
- a boolean expression that must be verified by the concerned objects,
- a miscellaneous part that gives various information on the constraint (level, comment,…).

3.2.4. Transcriptor

The Multilex tools used by the lexicographer must handle different writing systems. However, the relational databases considered for physical storage use a unique, normalized set of characters (typically, ISO 646 or 8879). Hence, Multilex has proposed a formal way to define writing systems, as well as a minimal transcription for coding natural language strings at the Multilex Internal Format (MIF) and Multilex DataBase (MDB) levels.

The backward and forward motion between the minimal transcription and its normal rendering is done by a transcriptor defined by the linguist.

A transcriptor generator has been fully developed during the first phase of Multilex. It takes as a parameter a finite state automaton defined by the linguist, and returns a transcriptor based on this automaton.

IV. Comparison between EDR, Genelex and Multilex

1. Overview

Making a comparison between EDR, Genelex and Multilex is not very easy. One must not forget the differences between these projects.

First of all, they have different profiles:
- EDR started on April 26, 1986. It involves ¥14 billion by the end of fiscal year 1994.
- Genelex started in September 1990. It involves about 250 million FF (i.e. about ¥6 billion).
- Multilex started on December 1st, 1990. It involves about 4.5 million ECU (i.e. about ¥730 million).

These projects use *different approaches*. Genelex and Multilex mainly take care of linguistic descriptions of lexical items, whereas EDR is more interested in developing a multilingual database based on concepts. The former projects define formalisms to handle different linguistic descriptions, whereas the latter mainly develops a complete architecture to store and link concepts and words.

These projects have *different goals:*
- EDR aims at building a multilingual dictionary. This dictionary will consist of different kinds of dictionaries: word dictionaries, concept dictionaries, co-occurrence dictionaries and bilingual dictionaries.
- Multilex proposes a standard for multilingual electronic dictionaries. It proposes an architecture based on monolingual and bilingual dictionaries.
- Genelex aims at the construction of generic monolingual lexical databases. Up to now, it has proposed a generic model for French.

These projects aim at *different usages:*
- EDR dictionaries will be dictionaries for computers.
- Multilex and Genelex dictionaries will contain information for the computer and for people.

2. Comparison criteria

2.1. Quantitative criteria

We take account of the following quantitative criteria:

Number of entries in the dictionaries
We will give the planned number of entries for each project.

Cost of the project
We will give the cost of each project in Japanese Yens.

Duration and invested effort of the project
We will give the duration and man-year effort invested in the projects.

Number of languages involved in the study
We will give the languages that are taken into account and for multilingual approaches, which pairs of languages are studied.

2.2. Qualitative criteria

We take into account the following qualitative criteria:

2.2.1. About the lexical information

What is the linguistic architecture?
The linguistic architecture is the organisation of the different dictionaries inside the database.

How rich is the linguistic model?
We will see what kind of lexical information is present in the model.

Presence of "classical" information?
"Classical information" is information that can be found in "classical" paper dictionaries.

If "classical information" is present, what is its form
We will see if "classical information" is given in an implicit or in an explicit form.

What kind of semantical information is present?
We will see the kind of semantic information that is present and its form.

2.2.2. About the lexical database system

Is it linguistically independent?
We will see if the lexical database system allows modifications in the linguistic structure of the entries.

Is it implemented?
We will see the implementation(s) of the systems.

What platform supports the system?
We will see on what plateform the system is implemented.

Does the system support multiscript information?
We will see if the system can handle multiscripts or not.

Does it provide some readable transcription for non roman scripts?
We will see if readable transcriptions are used in the different levels of the system (such transcriptions are very useful for debugging purposes and also provide a readable rendering of the information on non-multiscript terminals).

Does the system use a commercialized database software?
We will see what kind of database management system is used by the different projects.

2.2.3. About the functionalities of the lexical workstation

Is there a way to import and export entries (or parts of entries)?
We will look at the import/export mechanism if any.

Does the system support SGML standards?
We will see if the system uses SGML (or TEI) standards internally or as an input or output of importing/exporting mechanism.

Is there a way to define constraints on the entries?

We will see if the lexicographer can define constraints on the entries for an automatic coherence checking.

Is there a defaulting mechanism?
We will look at the defaulting mechanism. We will see if this mechanism can be defined by the linguist.

3. Comparison

3.1. Quantitative criteria

3.1.1. Number of entries

EDR plans to develop large dictionaries of about 300,000 words for each English and Japanese (200,000 of general vocabulary, 100,000 of terminological vocabulary). EDR also develops dictionaries of 400,000 concepts, dictionaries of 300,000 co-occurrences (for each English and Japanese) and dictionaries of 300,000 bilingual entries (for each Japanese-English and English-Japanese).

Each GENELEX partner has his own dictionaries:
- SEMA (from LADL tables): about 70,000 morphological units (among which 4,000 verbs with a complete syntactic description),
- IBM: about 50,000 morphological units,
- GSI-ERLI: about 68,000 simple morphological units, and about 15,000 compound morphological units.

Multilex is in its research phase. This criterion is not relevant for this project. However, an example dictionary consisting of about 600 terms in 5 languages has been produced in the first phase. A core dictionary (that will be freely distributed) of 3000 terms in 5 languages will be developed in the second phase.

3.1.2. Cost of the project

EDR costs about ¥14 billion.

Genelex costs about ¥6 billion.

Multilex costs about ¥730 million.

3.1.3. Duration and invested effort of the project

EDR will cover 9 years and involve about 1200 man-years.

Genelex will cover 4 years. It involves about 250 man-years.

Multilex will cover 3 years. It involves about 432 man-months, or about 36 man-years.

3.1.4. Number of languages involved in the study

EDR handles English and Japanese (Japanese-English and English-Japanese dictionaries are also considered).

Genelex handles French, Italian and Spanish (but only for monolingual information).

Multilex handles European languages in general in its theoretical studies. But it priviledges English, French, German and Italian for its monolingual dictionaries.

3.2. Qualitative criteria

3.2.1. About the lexical information:

What is the linguistic architecture?

EDR (mainly) adopts a interlingual architecture. This architecture is based on a dictionary of concepts where language independent concepts are described and linked to lexical entries in each language. EDR has also developed bilingual dictionaries for English-Japanese and Japanese-English.

Genelex works on generic dictionaries for monolingual information. They define generic models for French, for Italian and for Spanish.

Multilex adopts a classical multi-bilingual architecture. This architecture is based on 2 types of dictionaries: monolingual dictionaries and transfer dictionaries. A bilingual dictionary is the joining of 2 transfer dictionaries (one for each direction).

How rich is the linguistic model?

The EDR model for monolingual entries provides grammatical information for a word form. The semantic information is a link to a concept. As a concept is described in the concept dictionary, the information provided is rather rich. However, an entry is a word form. This choice provides good results for Japanese and English, but it is not very well adapted for flexional languages like French (about 10 word forms by lemma). The monolingual information is represented as a list of attributes.

The Genelex model provides rich information for each lemma. This information, represented as a graph, includes morphology, syntax and semantics.

Multilex describes an entry as a typed feature structure. Multilex defines the linguistic structure of the entries in a very complete way. The monolingual part of this structure contains 3 main parts: morphology, syntax and semantics. Multilex has defined structures for terminological information. Multilex also describes unidirectional bilingual entries. These entries mainly contain a test part and a transformation part.

Presence of "classical" information?

According to the published documentation, the EDR system does not represent any "classical" information like definitions or etymology, because it is designed to be used by computers. However, concepts are accompanied by quick descriptions in natural language (in Japanese and English).

The Genelex model will handle "classical" information.

Multilex is developed for both automatic and human use. In this framework, "classical" information is present (definition in natural language, phonetics, etymology,…).

If "classical information" is present, what is its form?

Classical information is not present in EDR's dictionaries.

This part of the Genelex model is under development.

In the Multilex model, “classical” information is given in an explicit form. A part of the semantic structure gives an explicit (machine processable) form to the definition in natural language. The definition is also given as a simple string in order to be used by human.

What kind of semantic information is present?

EDR provides a concept description by means of relations between concepts. EDR has defined a set of 32 relations and 5 attributes (i.e. unary relations) to describe the semantic network for EDR Corpus (according to April 1990's technical report). 18 main relations are used to describe the concepts in the dictionary. EDR has also developed a hierarchy of concepts which acts as a classification of concepts.

The semantic part of the Genelex model is under development.

Multilex provides two kinds of semantic information. The first is a description of acceptions by means of structured semantic features. The second one is based on the definition in natural language, made explicit (machine processable) as a set of atomic relations between acceptions.

3.2.2. About the lexical database system:

Is it linguistically independent?

The linguistic information is located in the word dictionaries. Although the basic format of EDR’s system is fixed according to the data type (e.g. there are only one right adjacency attribute and one left adjacency attribute for each headword), there is a great deal of flexibility in describing grammatical attributes. The descriptive framework for grammatical attributes is a flat list of attribute-values with no explicit structure among them. It is possible to add any value or comment in this field.

The Genelex model is defined in a SGML Document Type Definition (DTD) and the description of some points that remain implicit in the DTD are given in a separate document, in natural language. It is possible to change the model without rewriting any tools. However, the model remains a graph.

Multilex provides a language of description of the lexical structure of an entry. With this language, a linguist can express its own structure. The limitation of the independence is that the linguist must represent its structure using a typed feature structure schema.

Is it implemented?

There is only one lexical database system per partner. Parts of this system are developed in different laboratories and integrated at the main office.

Genelex lexical database system is implemented. There is one common implementation for all partners.

In its first phase, Multilex has only prototyped some tools. The lexical database system itself is to be implemented in the second phase.

What platform supports the system?

The system was previously developed on a Sigma-station (Fujitsu). Now, it is developed in C on a Sun Sparc-Station using japanese Xwindow manager (extended with the public domain Wnn system as a front end for Japanese characters entry.

Genelex: the lexical database system is developed in C++, under MOTIF, on a Sun4 and on a RS6000.

Multilex is in its research phase.

Does the system support multiscript information?

EDR handles Japanese and English writing systems. The multiscript information is supported by the Japanese window manager. EDR must use a transcription for non-English characters (e.g. *Götterdämmerung* is indexed as *G"otterd"ammerung*). A transcription of Japanese is used to be stored in the commercial databases (EUC: Extended Unix Code).

Genelex: not for the moment. The interchange format is based on the ISO 8879 standard.

Multilex is not yet implemented, but a transcription mechanism has been fully defined in order to be able to provide multiscript information. The only limitation in rendering will be due to the platform. Multilex also provides a way to define writing systems.

Does it provide some readable transcription for non roman scripts?

EDR uses EUC: (Extended Unix Code) as a transcription of japanese characters. This transcription is not readable. However the basic Japanese Unix stations are using this code, so it can be read easily.

Up to now, Genelex only deals with roman languages.

Multilex has defined a powerful transcriptor generator mechanism. With this mechanism, it is possible to use readable transcription for non roman scripts. With this generator, Multilex has built transcriptors for TEI, internal, and non roman (Thai,…) representations.

Does the system use a commercialized database software?

EDR system is available with *Sybase* and *Oracle* (commercialized Relational database management systems). It can also use *DIOS* (a Database Management Systems especially developed for the project).

The Genelex lexical database system can use any database software. Relational and object databases, as well as the “All in Memory” approach, have been implemented and tested.

Multilex plans to use commercial software. The only constraint is the use of a relational, Structured Query Language (SQL) based database management system.

3.2.3. About the functionalities of the lexical workstation:

Is there a way to import and export entries (or parts of entries)?

Any part of the EDR dictionaries (and corpus) can be exported to an SGML text form and to plain text form. The importing mecanism is not yet implemented.

Genelex: the entries can be exported and imported via a SGML format. Extraction and merging mechanisms will be developed later.

Multilex provides an import/export mecanism. However, this mechanism is not yet completely specified.

Does the system support SGML standards?

EDR provides an SGML form for all dictionaries. Japanese characters are given in EUC.

SGML is the communication standard for Genelex model.

Multilex provides a SGML/Text Encoding Initiative (TEI) based format for import and export of data.

Is there a way to define constraints on the entries?

EDR provides a way to check the validity of attributes by way of definition tables. However, High level constraints like "Attributes X and Y can not be present in the same entry" can not be expressed to check the validity of the dictionaries.

Genelex provides a way to define constraints on entries. These constraints will be automatically checked.

Multilex allows the linguist to define his/her constraints. These constraints will be automatically checked at each transaction. Multilex makes a distinction between 3 types of constraints (integrity (constraints on a single entry), local coherence (constraints involving entries of the same dictionary) and global coherence constraints (constraints involving entries of different dictionaries)) and 3 levels of constraints (warning, delay and critical constraints).

Is there a defaulting mechanism?

This criterion is not really relevant for EDR which has already achieved its indexation phase. However, EDR has defined a hierarchy of concepts. This hierarchy consists of a classification of concepts. It is made to minimize the concept descriptions by factorizing the information. This hierarchy acts as a defaulting mecanism for concepts.

Genelex provides a defaulting mechanism.

Multilex provides 2 kinds of defaulting mechanisms. First, the built-in inheritance mechanism of the Multilex formalism (typed feature structures). Second, an entry completion mechanism that uses tables of prefixes and suffixes.

3.3. Recapitulation

We give in the following table a brief recapitulation of the comparison.

	EDR	Genelex	Multilex
Number of entries	English: general: 200,000 term: 100,000 Japanese: general: 200,000 term: 100,000 Concepts: 400,000 Bilingual: 300,000 Cooccurrence: 300,000	Core dict: 3,000 Hachette: 55,000 Notre Temps: 60,000 +25000 proper nouns SEMA: 70,000 MUs IBM: 50,000 MUs GSI-ERLI: 68,000 simpleand 15,000 compound MUs.	This project is not yet in indexation phase
Cost of the project	¥14 billion	¥6 billion	¥730 million
Duration and invested effort	9 years ≈ 1200 man-years	4 years 250 man years	3 years 36 man-years
Number of languages involved	Japanese and English	French, Italian and Spanish (monolingual only)	European languages are taken in account for the study
Linguistic architecture	Interlingual (via a concept dictionary)	monolingual	Multi-bilingual
Linguistically independent	0	+	+
Classical information	-	+	+
Semantic information	+	?	+
Platform	C, Xwindow, Sun SparcStation	C++, MOTIF, Sun4 + RS6000	
Multiscript	Japanese and English	-	++
Transcriptions	+		++
Database software	Sybase, Oracle and DIOS (Implemented for the project)	Relational + object oriented + all in memory	relational (SQL based)
Import/export	+	+	+
SGML standards	+	+	+ (TEI)
Constraint checking	+	+	+
Defaulting mechanism	+	+	+

V. Conclusion

We have made a survey of recent trends in electronic dictionary research in Europe. During this survey, we have studied lexical database systems in Europe, with “classical”, “specialised” and “multi-usage” systems. Then, we have studied European lexical workstations, with monolingual and multilingual workstations.

Finally, we have made a comparison between the Japanese EDR project and 2 European projects on lexical databases: Genelex and Multilex. These projects are very different: they have different goals, different centers of interest, and different usages in view.

The Japanese EDR project proposes a very powerful architecture, based on a concept database (with concept description) that acts as an interlingua.

The Genelex project proposes a new kind of structure, based on an entity-relationship model. This structure seems very powerful in some cases.

Multilex proposes some standards for a multilingual database system, consisting of a very complete linguistic standard and a very powerful and very flexible software architecture.

On the other side, it seems that EDR would have to develop another linguistic architecture to tackle new languages, as the current one is adapted to Japanese and English, but would be difficult to use for flexional languages like French (about 10 word forms per lemma).

Genelex has developed a model for French. No multilingual consideration has been taken in account. Moreover, the entity-relationship model is well adapted in some cases, but other cases would be treated more simply with standard basic models (trees, feature structures,…).

Multilex has proposed a rich set of standards for lexical database software. The linguistic architecture is based on a multi-bilingual dictionary approach. It will be interesting to study an approach based on an interlingual dictionary.

All the projects studied here impose constraints on the linguistic level. That is true even of multiusage projects, that are designed to be used by a great variety of applications. Each of these projects imposes its own notion of “lexical unit” (lemma, word-sense, concept) and its own logical structure (Typed Feature Structures, Entity-relationship model, automata, trees, …).

With these constraints, a user can not use the same system to consult 2 databases coming from different places. It is not possible to handle the different dictionaries under the same lexical database system (neither under the same machines). This problem is critical for data acquisition from other electronic dictionaries.

The next step in lexical database research is to define a system which can manipulate heterogeneous structures (trees, Typed Feature Structures, graphs,…). With such a system, it would be possible to manipulate the currently existing electronic dictionaries with the same tools, under the same system. Of course, this would not solve the problem of data sharing and data acquisition, as the linguistic structures would remain different. But this would simplify the acquisition and merging by providing a common workstation for the different sources. Such a system will also be very useful to study new lexical structures that merge different aspects of the different basic structures (e.g. a structure where the entry is an automaton associated with a word-sense tree decorated by Typed Feature Structures).

A.1 Multilex

I. Introduction

1. Overview

MULTILEX is a CEC - DG XII - ESPRIT project. The aim of this project is to define standards for multilingual lexical databases. The planned duration is 3 years and is being executed in 2 phases with 2 coordinators. Languages considered are essentially those of the European Community.

The participants of the first phase of this project were:
- Triumph-Adler AG, Germany
- CAP Gemini Innovation, France
- Lexicon Srl, Italy
- Philips, The Netherlands
- L-CUBE, Greece
- Siemens Nixdorf, Germany
- Siemens Nixdorf, Spain
- GETA IMAG, University of Grenoble & CNRS, France
- ASSTRIL (Paris VII), France
- University of Pisa, Italy
- University of Manchester Institute of Science and Technology, United Kingdom
- Vrije Universiteit Amsterdam, The Netherlands
- University of Surrey, United Kingdom
- University of Bochum, Germany
- University of Münster, Germany.

The first phase aimed at the definition of the different standards (linguistic standard, representation formalism, definition and prototyping of tools). The second phase will be more application-oriented.

The first phase ended on 30 June 1992. The second phase is just beginning. This report covers only the results of the first phase.

2. Goals

The aim of the Multilex project is to propose and experiment standards for multilingual lexical databases.

There are different standards to propose: standards on linguistic information (what is in the database) and architecture (how the entries, the dictionaries,… are organised), standards on tools (how to manipulate the database), logical structure (how the information is represented) and software architecture (how the tools are organised).

3. Summary of results

During the first phase of the project, Multilex has performed the following tasks:

- Definition/simulation of the standards,
- Prototyping of some tools,
- Coding of some entries (English, French, German and Italian).

The main task of the first phase was to define the standards for multilingual lexical databases. Some prototypes of tools for the lexicographer have been developed with these standards during the first phase, but the main experiments and developments will be performed during the second phase of Multilex.

II. Linguistic architecture

In this section, we will focus on the linguistic architecture of a Multilex dictionary. We will present the organisation of the different dictionaries, the concept of Lexical Unit as defined by Multilex and, finally, the organisation of the information inside the Lexical Unit.

1. Overview

A Multilex database consists of a set of dictionaries. There are 2 types of dictionaries: monolingual and bilingual dictionaries.

The general architecture of MULTILEX foresees one monolingual dictionary per language and two bilingual dictionaries per pair of languages (see figure 1).

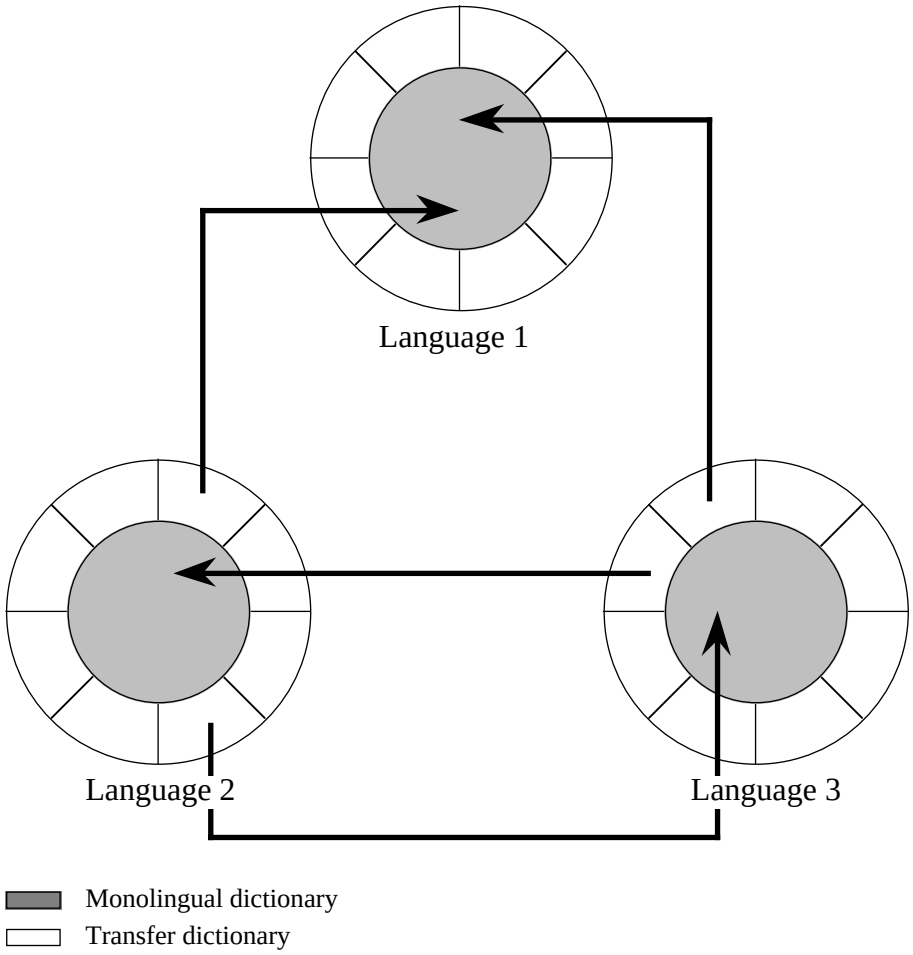


Figure 1: Multilex dictionary model

A dictionary is a set of articles. There is only one type of article in a dictionary.

A monolingual dictionary consists of a set of Lexical Units (LU).

A bilingual dictionary is made of translation units relating source language dictionary entries and target language dictionary entries.

2. Monolingual part: the Lexical Unit

A Lexical Unit (LU) identifies lexical information for one and only one meaning.

2.1. Identification

The LU identifier is a 2 face object consisting of:
- a canonical form,
- a discriminating mark (e.g. a number) distinguishing different units having the same canonical form.

The canonical form is defined as follows:

A canonical form is the orthographic standard form ("citation form") fixed in each of our languages by convention, except for a number of exceptions in which the standard is not fixed conventionally (e.g. 'colineau' or 'colinot' in French, or 'yoghurt', 'yogurt' or 'yoghourt' in English). In such cases, the canonical form is arbitrarily chosen among the possible citation forms.

In case of homography, the Multilex standard binds an additional mark to the LU identifier to account for homography. This homography mark must be different from the meaning discrimination mark.

For example, the LUs having as a canonical form 'louer' are identified by the combination of the 3 following elements:

canonical_form = louer, homograph_number = 1, meaning = 1: `[louer_1_1]`
canonical_form = louer, homograph_number = 1, meaning = 2: `[louer_1_2]`
…
canonical_form = louer, homograph_number = 2, meaning = 1: `[louer_2_1]`
canonical_form = louer, homograph_number = 2, meaning = 2: `[louer_2_2]`
canonical_form = louer, homograph_number = 2, meaning = 3: `[louer_2_3]`

2.2. Specification

Each LU is specified by:

Name	Definition	Optionality	Nb
GPMU	a configuration of graphic, phonological and morphological attributes. Can be shared by several LUs	Obligatory	any
Syntactic description	1. a set/structure of syntactic attributes relevant to the LU. 2. elementary semantic features for syntactic arguments disambiguation	Obligatory	1
Semantic description	expression of semantic contents	optional	1
Cross-references	set of labelled/named links between LUs to be defined by the user. Ex: synonyms, antonyms, superordinates, conceptual links, etc.	optional	1
Transfer	bilingual transfer	optional	any
Maintenance record	set of maintenance forms	obligatory	1

So, a LU can be seen as the structure presented at figure 2.

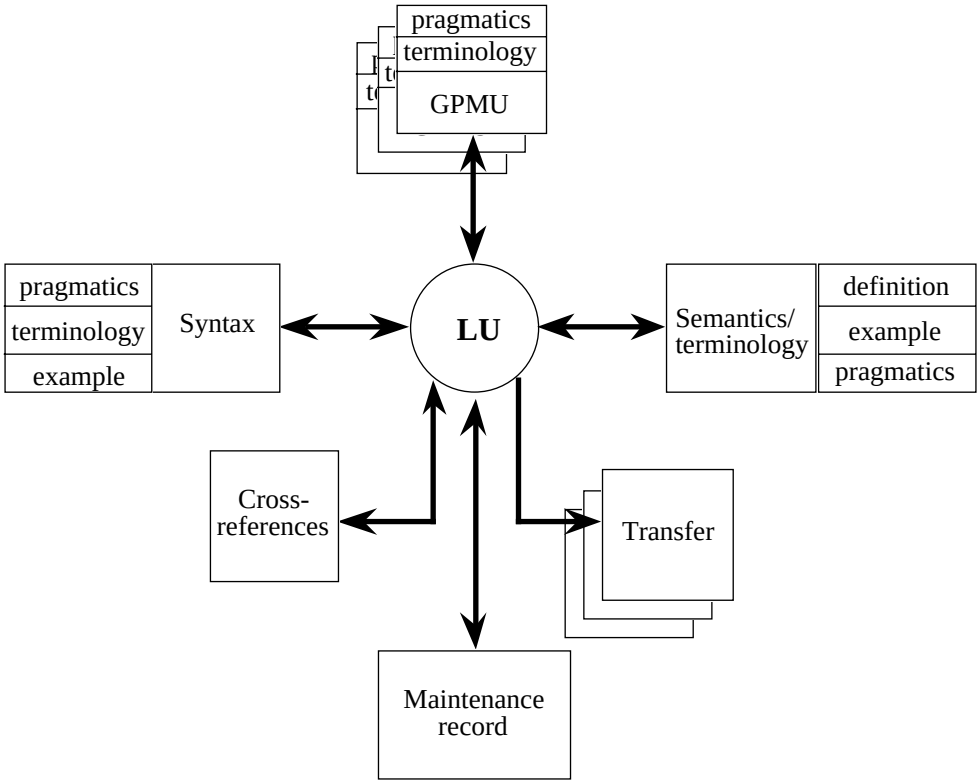


Figure 2: General view of a Lexical Unit.

In the following paragraphs, we will present the different parts of this structure. A more detailed specification of the linguistic information present in these parts will be given in the next chapter.

2.3. Graphical, Phonological and Morphological Unit (GPMU)

In the Multilex standard, the orthographic, phonologic and morphological levels are organized in such a way that interrelated graphic, phonological and morphological attribute-value pairs are kept all in the same set (a GPMU) and a new GPMU is created for each variation and its consecutive effects on other surface behaviours.

For example, the French noun ‘colinot’ which has a variation ‘colineau’ has two GPMU (see figure 3). This is how Multilex shows that the formation of the plural does not follow the same rule for ‘colinot’ (plural ‘colinots’) and ‘colineau’ (plural ‘colineaux’).

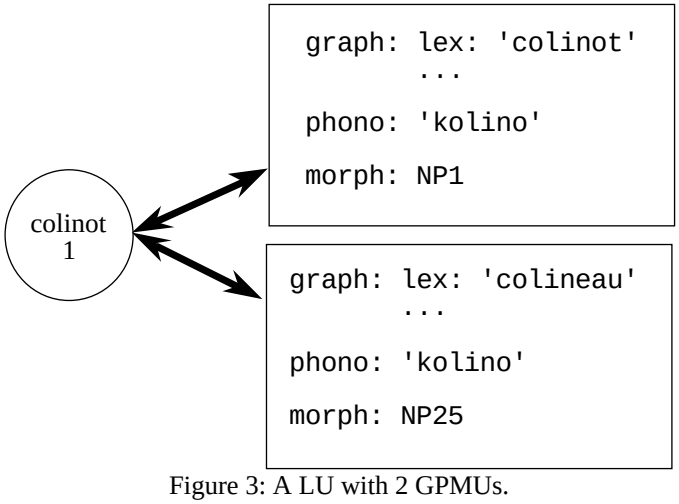


Figure 3: A LU with 2 GPMUs.

Because of the interdependence of all the surface features, Multilex chooses to link the pragmatic information to the GPMU as a whole.

Take for example the French noun ‘paiement’, which has a less frequent variant ‘payement’. These two spellings have different pronunciations but are derived from the same verb ‘payer’, according to 2 variant conjugation models (‘je paie’ or ‘je paye’). Should we decide that these words are orthographic, morphological or phonological variants ? This question makes sense when introducing pragmatic information. ‘payement’ is less frequent than ‘paiement’, but this might be related to a more general remark stating that the pronunciation with the semi-consonant yod is more rarely used than the alternate pronunciation without yod.

In order to allow an LU to account for domain idiosyncrasies (providing that the LU in question is semantically unique), specific domain information constraining the selection of GPMUs is stored with each GPMU.

2.4. Syntax

There is one syntactic description per LU. However, in rare cases, variant syntactic constructions are semantically equivalent (correspond to the same LU). Syntactic variations can be dealt with in the Multilex model.

a. **Specific attributes** can be added in case of regular syntactic variations.
For example: A well defined category of English verbs taking 3 arguments may have two equivalent constructions: ‘somebody gives someone something’ is equivalent to ‘somebody gives something to someone’. Since this particularity concerns many verbs and since it is language dependent, a specific syntactic attribute is created to convey this information which will be efficiently dealt with by a rule in a specific application framework.

b. For unconstrained alternation variations, Multilex allows **multi-valued attributes.** More than one attribute of the syntactic description can receive multiple values, provided that the possible combinations of values are compatible.
For example: A meaning of the French verb ‘comparer’ can be expressed either by both the syntactic structure ‘comparer X à Y’ and ‘comparer X avec Y’. ‘à’ can be substituted to ‘avec’ and vice versa without additional change of the structure.

c. When incompatibility of multi-values occurs, Multilex allows **duplicating the syntactic description**.

Other informations are related to the syntactic structure:
- kinds of pragmatic information
- terminological information
- examples.

2.5. Semantics

Multilex provides a set of elementary features which are commonly agreed upon as a base for syntactic disambiguation. The semantic object attached to each LU consists of 2 parts, one of which is an obligatory elementary semantic feature or list of elementary semantic features. Functionnally, it is a syntactic information and is stored with the syntactic features. The other part — which is optional — is the extensive expression of the semantic content.

A paper dictionary style definition is necessary to help translators, linguists, lexicographers or any other human users to choose the right meaning. Hence, this text definition is obligatory.

Since a term LU can be used in several sub-languages, it may bear more than one terminological specification in the limit of the descriptive apparatus provided by the terminological description of lexical items.

An obligatory example at the semantic level is required to illustrate the meaning and to place it into a contextual situation for better understanding, the meaning being one of a term or a general language word.

Optional pragmatic information can be linked to an LU to make more precise the status of a notion designated by the LU or the status of the link between the canonical form and the semantic content of the LU.

2.6. Transfer

This part of the LU provides a direct link to all the transfer entries in the different bilingual dictionaries that have this LU as source LU. It is simply a list of transfer entries identifiers.

As a general resource for different applications, a Multilex dictionary may contain LUs for which no transfer to any language is wanted or LUs for which the sets of transfers are heterogeneous. No restriction or special requirement bears on the number or the absence of transfer specification. But it is possible to express and use some constraints to detect the presence or absence of a specification anywhere in the structure (see below, Integrity and coherence checking).

2.7. Cross-references

A cross reference is defined by 2 elements:

- a link type and
- a LU list

where the values of the link type refer to synonymy, antonymy, genericity, specificity,… Link reversibility has to be defined and a reverse link type can be specified.

2.8. Maintenance record

An obligatory maintenance record is linked to each LU and is updated — a maintenance sheet is added — every time a change is made in the information related to this LU or in the GPMUs it inherits from. A distinct (also obligatory) maintenance record is linked to each bilingual transfer.

Maintenance information (for each edition in which a change has been made) should at least contain the following features:

- lexicographic STATUS: new - old - deleted (i.e. soft deletion)
- OWNER: name of the owner of the article (i.e. the company)
- PRODUCT: the product for which the transfer is valid within the owner's production
- AUTHOR: name of the editor
- CREATION DATE: date of first edition
- LAST UPDATE: date of last edition
- MAILBOX: for lexicographers to communicate information about the wourk in progress (e.g. a message "refer to the LUs of 'left'" in the mailbox of the LUs 'right' as a warning that these entries must be consistent). These comments are optional.

3. Bilingual part: the transfer entry.

3.1. Identification

The transfer entry is uniquely identified by the identifier of the bilingual dictionary in which it appears and an arbitrary number, unique within the dictionary.

3.2. Specification

All transfers relate LUs. They give only the type of the relation between a source language LU and LUs of the target language and leave the LU descriptions to the monolingual dictionary. Transfers are unidirectional.

The links established between the linguistic levels by the model guarantee that all monolingual information necessary for the transfer is available through the LU identifier.

A transfer can either relate a source LU to one target LU or to a combination of LUs when there is no single LU in the target language to account for the meaning that the source LU conveys. For example, the French word ‘permissionnaire’ has no corresponding word in English and is translated by a noun phrase ‘soldier on leave’.

III. Linguistic information

In this chapter, we will focus on the linguistic information that can be represented in a Multilex dictionary. We will see in detail the orthographical, morphological, syntactical and semantical data. Then, we will look at the information contained in the transfer part.

1. Orthography

Orthography is strictly concerned with the spelling of a word in the relevant writing system(s) (for Multilex purpose, the alphabets of the European Community Languages). Under one view, it is concerned with the correct or accepted spelling, according to actual or perceived rules and regulations; under another view, it is concerned with the principles that underly spelling.

The orthographical information is located in GPMUs.

We give below the list of features that have been defined in order to code orthographical information.

Feature	Description	Possible values	Default
`Lex`	String representing the canonical form of a word, with upper and lower case characters distinguished.	a string	

`Lex_type`	Indicates full forms of words or abbreviated variants thereof, e.g. BBC is the acronym of British Broadcasting Corporation.	`full`
		`abbreviation`
		`acronym`
		`truncation`
		`symbol`
		`elliptical_form`
			`full`

`Geography`	A linguistic region for the language in question reflecting dialect differences.	a linguistic geographical region	according to language in question

`Temporal`	Places language use in a time continuum.	`contemporary`
		`archaic`
		`obsolete`
			`contemporary`

`Social_group`	Identify a social group for whom a variant is (interacts with the attribute ‘style’).	`all` a social group	`all`
`Subject_field`	Used for terminology. Not yet specified for general language.	`general` a subject field	`general`

`Style`	sociolinguistic variation in language use.	`neutral`
		`formal`
		`informal`
		`standard`
		`nonstandard`
		`substandard`
		`euphemistic`
		`offensive`
		`taboo`
		`poetic`
			`neutral`

Feature	Description	Possible values	Default
Status	This attribute relates to the acceptibility of a term.	preferred progressive accepted unofficial	
Frequency	Gives the frequence of a word or of the spelling of a word.	common less_common uncommon rare	common
Hyphenation	Gives a list of hyphenation possibilities	list of hyphenation variants	

2. Morphology

The morphological information is located in GPMUs.

We give below the list of features that have been defined in order to code morphological information.

Feature	Description	Possible values	Default
Cat	indicates the morpho-syntactic category of the entry.	A Adjective Adverb Noun P Verb Subordinator Coordinator Det Pronoun Num Int Punctuation Aux Nocat	
Entry_type	gives the type of the entry which is described	lemma inflected_word word_combin derived_word	lemma

The next features depends on the type of entry given in `entry_type`.

If the entry is a **lemma** (`entry_type = lemma`):

Feature	Description	Possible values
`inflection_prop`	describes the inflectional properties.	a complex type depending of the category

`Boundness`	distinguishes the entries according to the degree of boundness.	`clitic`
		`enclitic`
		`proclitic`
		`prefix`
		`suffix`
		`infix`
		`free_form`

If the entry is an **inflected word** (`entry_type = inflected_word`):

Feature	Description	Possible values
`inflection_desc`	describes the inflection.	a complex type depending of the category

`Boundness`	distinguishes the entries according to the degree of boundness.	`clitic`
		`enclitic`
		`proclitic`
		`prefix`
		`suffix`
		`infix`
		`free_form`

`lemma`	refers to the entry that represent the lemma of the word of which it is an inflected form.	a GPMU identifier

If the entry is a **word combination** (`entry_type = word_combin`):

Feature	Description	Possible values
`components`	specifies the words the word combination consists of.	a sequence of GPMU identifiers

If the entry is a **derived word** (`entry_type = derived_word`):

Feature	Description	Possible values

`derivation`	describes the type of derived form.	`dim`
		`compar`
		`superl`

`derived_from`	refers to the word it has been derived from.	a GPMU identifier

3. Syntax

We will present the list of features that have been defined in order to code syntactical information.

In order to describe the structure that a LU can enter, Multilex choses to give the list of explicit structures, i.e. in the form N0V0N1.

In this chapter, we will use the French notation for these structures (e.g. N0êtreAdj) rather than the English notation (e.g. N0beAdj).

First, we give the types that are used in all the descriptions regardless of the category of the LU.

Type name	Description	Possible values

`Cat_type`	indication of the category	`V`
		`Adj`
		`Adv`
		`N`
		`Prep`
		`Conj`
		`Det`
		`Pro`
		`Card`
		`Nocat`
		`FS`[2]
		`frozen_NP`

`SubCatInfo`	subcategorization information.	`Nhum`
		`Nhumc`
		`Nan`
		`Nur`
		`Nloc`
		`Nun`
		`Nbp`
		`Nc`
		`Nart`
		`QuP`
		`QuPsubj`
		`V1W`

`AddP_type`	additional properties (symetry,…).	`sym(N1,N2)`
		`sym(N0,N1)`
		…

[2] FS stands for ‘Frozen Sentence’.

The following features depend on the category of the entry.

If the entry is a **verb**:

Feature	Description	Possible values
`cat`	the category	Cat_type
`Example`	an example	a string

`MainS`	main structure (a structureV)	`N0V0`
		`N0V0N1`
		`N0V0àN1`
		`N0V0deN1`
		`N0V0N1àN2`
		…

`SubCat`	subcategorization	list of SubCatInfo
`DerS`	derived structures	list of structureV
`Auxiliary`	auxiliary used with the verb in compound tenses.	Type of auxiliary (depends of the language).

`Passive`	this optional feature describes the passivation of the verb. It consists in a list of the following possible values:	`N1êtreVppparN0`
		`N1êtreVppdeN0`
		…

`AddP`	additional properties	list of AddP_type
`reflexivity`	indicates the reflexivity of the verb	`yes` `no`
`operator`	for operator verbs, gives the relationship between the verbal construction and the underlying adjectival (or verbal) construction. For example, the verbal entry of ‘to make’ corresponding to the operator use of this verb will receive the following argument: `operator = {(N0VN1Adj, N1beAdj)}` which gives the relationship between the 2 following sentences: ‘Max made him mad’ and ‘he is mad’. a structure Op is: StructureV * (structureV or structureAdj)	list of structureOp
`paspartattr`	to indicate if the passive participle of the verb can be used attributively.	boolean
`io_subj_orderallowed`	to indicate if the order indirect object-subject in non-topicalized structures is possible	boolean
`nonoblcontrol`	to indicate whether nonobligatory control is allowed or not (obligatory control only, or nonobligatory control as well).	boolean

Feature	Description	Possible values
`Particle`	specifies if the verb takes a particle, and if so which particle (for certain languages only).	an identifier to a LU of type particle
`Reflexivity`	indicates whether the verb is a reflexive verb or not (for certain languages only)	boolean

`Subc`	subclassification of verbs	`MainVerb`
		`ModalVerb`
		`PerfAux`
		`ProgAux`
		`PassAux`
		`Causative`

`Perfauxes`	indicates which auxiliary verb(s) is (are) used to form perfect tenses	set of LU identifiers
`CompDel`	indicates whether deletion of the complementizer (the subordinate conjunction) is allowed or not (for certain languages only)	boolean
`Intensionality`	indicates the possibility to interpret the direct object intentionally (for certain languages only)	boolean
`BridgeVerb`	indicates if the verb allows finite sentencial complements from which an element is extracted	boolean

If the entry is a **frozen sentence**:

Feature	Description	Possible values
`cat`	the category	Cat_type
`Example`	an example	a string

`MainS`	main structure (a structureFS)
	A structureFS is a structure of frozen sentences. Fixed arguments (C1) are linked with pointers to the corresponding GPMU through the SubCat argument.
		`C`
		`N0V0C1`
		`N0V0C1àN2`
		`N0V0N1àC2`
		…

`SubCat`	subcategorization. the type SubCatInfoFS is a SubCatInfo or C0=GPMU or C1=GPMU or C2=GPMU.	list of SubCatInfoFS
`DerS`	derived structures	list of structureFS
`Auxiliary`	auxiliary used with the verb in compound tenses.	Type of auxiliary (depends of the language).

If the entry is a **noun**:

Feature	Description	Possible values
cat	the category	Cat_type
Genders	indicates the grammatical gender of nouns	masc fem neut
Sexes	indicates the natural sexes (biological gender) of the noun	set of (masculine feminine)
Subc	subclassification of nouns	propernoun commonnoun … (still to be determined)
requiresdet	indicates whether the noun requires the presence of the definite determiner (for proper nouns only)	boolean
AAR	indicates if the noun is subject to the animate accusative rule found in certain languages	boolean
relpro	indicates what kind of relative pronoun the noun allows or requires	relproclass1 relproclass2
masscount	indicates whether the noun is a count noun or a mass noun	boolean
impersonal	indicates if it can be used in impersonal sentences such as “it is winter”	boolean
Inalienable	indicates if the relevant lexical unit is inalienable	boolean
DetPred	indicates if the noun can be used without any determiner or article present when used predicatively	boolean
vocative	indicates if the noun can be used to address people, without any article, e.g. mr., mrs.	boolean
Example	an example	a string
MainS	main structure (a structure N) The structureN contains a pointer to the support verb they are involving	N0VsupN N0VsupNàN1 N0VsupNdeN1 …
SubCat	subcategorization.	list of SubCatInfo
DerS	derived structures	list of structureN

If the entry is an **adjective**:

Feature	Description	Possible values
`cat`	the category	Cat_type
`uses`	indicates how the adjective can be used	set of (`attributive` `predicative` `nominalized`)
`Position`	give the position that an adjective can enter in a noun phrase: before the noun or after the noun or both.	`before` `after` `before&after`
`reflexivity`	indicates if the adjective requires a reflexive pronoun (certain languages only)	boolean
`advFormation`	indicates whether the adjective can also be used as an adverb (e.g. in Dutch) or to indicate that an adverb can be derived from it in a fully regular and semantically transparent manner (e.g. French 'rapide' -> 'rapidement' and English 'quick' -> 'quickly')	boolean
`copulas`	indicates which verbs can be used as copulas with the adjective	set of reference to LUs of copulas
`Example`	an example	a string
`MainS`	main structure (a structure Adj)	`N0êtreAdj` `N0êtreAdjàN1` `N0êtreAdjdeN1` …
`SubCat`	subcategorization.	list of SubCatInfo
`DerS`	derived structures	list of structureAdj

4. Semantics

The semantic specification of the dictionary must distinguish and interrelate the meanings or the significations of the words. If a canonical form of a word has several distinct meanings, it is enough to enumerate and to determine these; it is not necessary to specify a disambiguating procedure which produces the meaning of an ambiguous word occurence in a text. Such disambiguating procedure is part of a parser. It will make use of dictionary information and will thus be partially determined by it. The same holds for the use of dictionaries in text generating procedures; they will use the dictionary information but the selection of the appropriate word in a given context will only be partially determined by the dictionary. Multilex is mainly concerned with the representation format in which semantical analysis should be expressed. Three different formats have been advocated in Multilex:

- a consistent current reference format based on human user dictionary formats
- a logically regimented format
- a typed feature structure format.

4.1. Reference format of semantic explanation

In this approach, there are two essential parts of the semantical section for a Lexical Unit, the Lexical Sentence and the Explanation/Definition.

For example, the semantic part of 'praise' entry will consist of:
- Lexical Sentence: so. praise so. for sth.
- Explanation/Definition: to speak of with admiration and approval.

4.2. Regimentation of the ordinary language entries of current semantical reference representation

This approach consists in a reorganization of the explanation exposed in the preceding approach.

For example, the semantic part of the 'praise' entry will be reorganised in:
- Lexical Sentence: so.x praises.1 so_or_sth.y
- Explanation/Definition:
  so.x expresses.1 sth.z
  and sth.z is: approval.1.4 for sth.r
  and sth.z is: strong.3.1
  and sth.r is: qualities.2.1 of so_or_sth.y
  or sth.r is: achievements.1 of so_or_sth.y

4.3. Typed Feature based approach

This approach assumes that a Explanation/Definition has to provide the meaning as a construction of terms where each term (and term combination) designates a concept whose semantics is determined by its position in the semantic hierarchy, and by its set of features (attributes and values).

Within the semantic feature approach, the formalism proposed for representing semantic information is the Typed Feature Structure (TFS) Representation System. The general characteristics of the representation system proposed by Multilex are on the one hand that it uses a typed unification-based representation language, on the other hand that it incorporates default inheritance.

The TFS system we are referring here consists of:
- a type system consisting of a type hierarchy plus constraints
- a default inheritance mechanism
- lexical rules.

Multilex has proposed a core set of basic features. As an example, we give here the hierarchy of semantic features for nouns (see Figure 4).

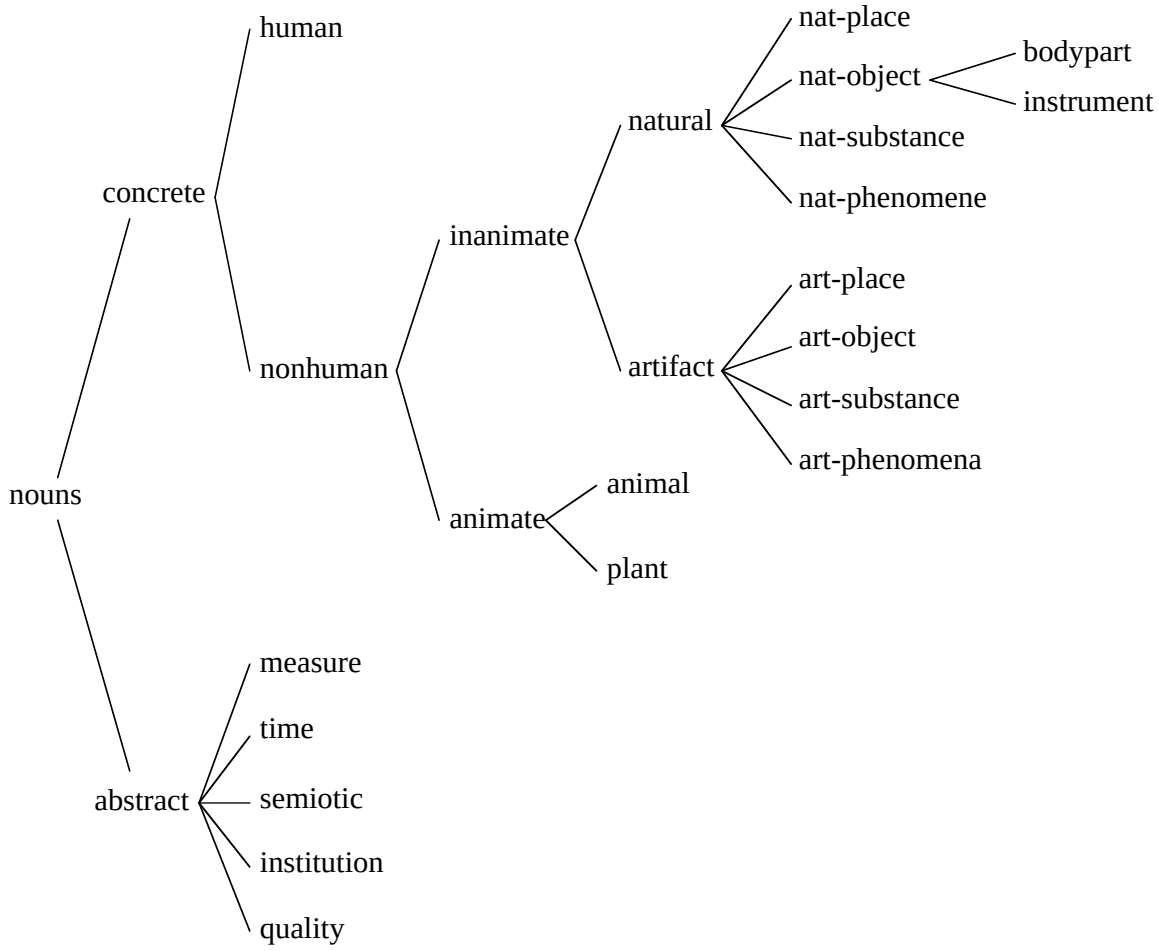


Figure 4: Hierarchy of semantic features for nouns.

5. Transfer information

An entry in a transfer dictionary consists of:

Feature	Description	Values
`Target_dict`	an obligatory specification of the target language	the name of a dictionary
`Mapping`	an obligatory mapping of a source LU into a target LU (or a target paraphrase)	a source and a target LU

`Equivalence`	an indication of equivalence type.
	Specifying the equivalence between LUs of two different languages consists in indicating wether the contents of the source LU will be completely, partially or not translated into the target language.
		`complete`
		`hyperonym`
		`related`
		`variant`
		`explanation`
		`borrowing`
		`near_equivalent`

Feature	Description	Values
`Test`	an optional test field. The test field contains a series of syntactic and semantic constraints that have to be fullfilled for a given transfer to be chosen. A distinction is made between formal tests to be used by MT systems and informal tests that are human processable.	a test
`Transformation`	an optional transformation field Transformations allow to map the syntactic predicate-argument structure of the source LU onto the syntactic predicate-argument structure of the target LU. Multilex provides the means to express transformation statements in a standardised form (see below).	a transformation
`examples`	some examples for human users.	a string
`maintenance`	an obligatory maintenance record (see linguistic architecture).	

Note about the mapping:

A transfer generally links a source LU to a target LU. However, there are cases in which a source LU (simple or compound) has not necessarily a corresponding LU (simple or compound) in the target dictionary.

When the lexical gap is also a cultural gap, it can be filled by a target paraphrase which is generally the semantic definition of the source LU rendered in the target language. This definition cannot be used directly by an MT system. It is merely an information for the human translator to consult at post-editing time.

E.g., French 'khâgne_lu_1' -> English 'art class preparing entrance exam for the Ecole Normale Supérieure'

More frequent is the case in which the lexical gap can be filled by a phrase which is valid as a translation although it is not frozen enough in the target language to figure as an entry of the dictionary. In this case, the target phrase will be a combination of target LUs.

E.g., French 'permissionnaire_lu_1' -> English (soldier_lu_y on leave_lu_z).

Note about the transformations:

Multilex gives three basic operations in order to express transformations:

- **Deletion** of an argument form: arguments that are present in the source syntactic frame may be unwanted in the target language. For example, one meaning of the verb 'to eat' has an object which is a meal (or some food) according to which the translation of the verb into French is different. Therefore, **after a bilingual test has identified the object argument as 'lunch'**, the transfer will state its deletion as an action to be performed when using the verb 'déjeuner'.
  E.g., English 'eat_lu_x '-> French 'déjeuner_lu_y'
  (transf:delete source_arg:N1)
  Example: English 'he eats lunch in a fast-food' -> French 'il déjeune dans un fast-food'.

- **Argument reshuffling:** A different case consists in the mapping of source language arguments onto different target language arguments:
  E.g., English ‘like_lu_x’ -> French plaire_lu_x
  (transf:map source_arg:N0 target_arg:(à N1))
  (transf:mapsource_arg:N1 target_arg:N0)
  Example: English ‘Mary likes him’ -> French ‘Il plait à Mary’.

- **Information addition:** In some cases, the translation requires that additional LUs be added to the equivalent target LU because the source LU holds specific information which is not included in the target LU.
  E.g., English ‘hammer_lu_x’ -> French ‘enfoncer_lu_y’
  (transf:add target_phrase:(avec un marteau_lu_z))
  Example: Enlish ‘to hammer a nail’ -> French ‘enfoncer un clou avec un marteau’.

IV. Software architecture

In this section, we will see the standards on software architecture for a Multilex database system. We will begin by an overview of the software architecture and then, we will see in detail the different levels that appear in this architecture.

Multilex software architecture is organized in 3 levels (see figure 5):

- *Multilex Database (MDB) level:* the different objects of the MULTILEX database will be shared, using some appropriate coding, in a relational model (based on SQL). The MDB level will be completely transparent to the different users (lexicographers, linguists, end-users…).
- *Multilex Internal Format (MIF) level:* it is the level which is accessed by the different tools. The different objects of the MULTILEX database are represented using the MULTILEX Internal Format (see chapter 'Logical Structure').
- *Multilex Presentation Format (MPF) level:* it is the abstract level at which the lexicographer works. The differents tools must provide outputs and accept inputs at this level. The abstract structure is defined by the lexicographer. It is the interface between the user and the system.

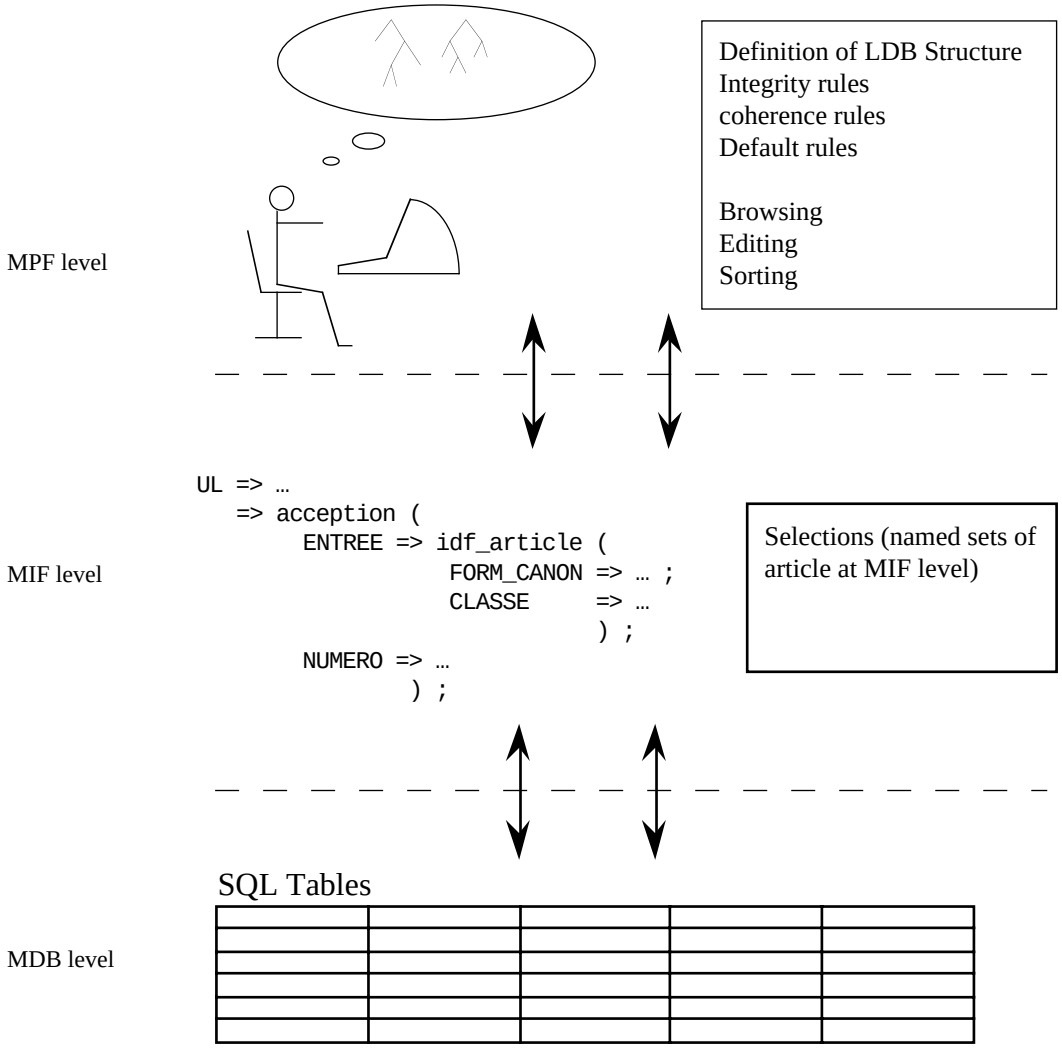


Figure 5: the different levels

The functioning is based on the backward and forward motion between the MPF, MIF and MDB levels.

As an example of the functioning of this architecture, we will see the processes for evaluating a query in a Multilex database. The query is expressed by the user at MPF level, and coded at MIF level. Then, it is translated and evaluated at MDB level. If the involved set of articles is already present at MIF level, the query is directly evaluated (see figure 6).

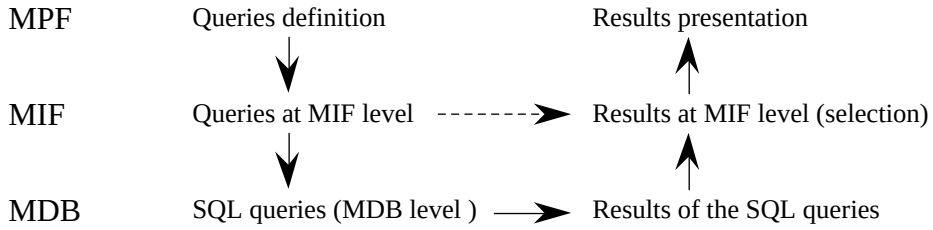


Figure 6: queries evaluation in MULTILEX

V. Logical structure

All linguistic standards have to be represented in a logical structure in order to be stored and manipulated by the computer. In this section, we will focus on the linguistic structure chosen by Multilex as a standard for representing lexical information. We will begin with a short introduction to the chosen structure. Then, we will see some examples of Lexical Units represented in this structure.

The structure represented here belongs to the Multilex Internal Level. All the tools will be designed to manipulate this structure.

1. Typed Feature Logic based structure

The Multilex Internal Format (MIF) has to be flexible (flexible enough to be theory independant), linguistically relevant (to be able to present information in a form which is relevant to linguists), declarative, tractable (to be stored in a relational database management system).

To fulfill these requirements, Multilex has opted for a MIF based on Typed Feature Logic (TFL).

TFL can be seen as a development of unification formalisms in which the feature bundles are constrained by assigning types. Type hierarchies can be built allowing inheritance of data about constraint and feature values, and giving very efficient data representations. There are also potential advantages in data coherence and processing efficiency of the sort realised by typing in computer programming languages.

The syntax and semantics are similar to that of the POLYGLOSS system (Emele and Zajac 1990).

1.1. The Attribute Value Matrices

The basic unit of the MIF is the typed Attribute-Value Matrix (AVM), also known as 'Feature Structures'. In a typed AVM, all nodes are typed, and the type of the AVM is the type of its initial node.

As an example, the following AVM is of type `SYNT`, with a feature, `syntax`:

```
SYNT [ syntax: HEAD [head: [syncat: v],
                     agr:    AGR  [person: 3,
                                   number: sing]]].
```

The type `SYNT` is supposed to have been previously declared. We will see how to declare a type in the next section.

1.2. Types and constraints

Types can be complex or atomic. A complex type has a structure, and the attributes which form this structure are defined with their possible values (complex or atomic). An atomic type has no further structure, and is simply a list of possible values that a feature can take. A complex type is declared as follows:

```
SYNT = [syntax: HEAD].
```

A simple type is declared as a disjunction of symbols, as follows:

```
NUMBER = sing | plur.
```

Equality constraints can be encoded in the type declaration: this is consistent with unification between different parts of the type structure, and is captured by using variables in type declarations. The scope of a variable is the type definition where it appears. Variable symbols start with a #.

The following example shows a PERSON which has a spouse which is also a PERSON.

```
#x=PERSON[spouse: #y=PERSON[spouse: #x]].
```

The inclusion of the variables #x and #y adds 2 constraints:

- the spouse of #x is #y and
- the spouse of #y is #x.

To allow more flexibility in the definition of logical constraints, Multilex will allow the constraint of inequality between types, written by using an inequality symbol: =/= in place of the =.

1.3. The type hierarchy

The type hierarchy is a set of types which is partially ordered by the subsumption relation: <TYPE,⊆>. As such it has the following properties:

- Reflexivity: t ⊆ t for any t in <TYPE,⊆>,
- Anti-symmetry: if t ⊆ s and s ⊆ t in <TYPE,⊆> then s = t,
- Transitivity: if t1 ⊆ t2 and t2 ⊆ t3 then t1 ⊆ t3.

The hierarchy has a maximal element T such that t ⊆ T for any t in <TYPE,⊆>, and a minimal element ⊥ such that ⊥ ⊆ t for any t in <TYPE,⊆>. Information is inherited monotonically from type to sub-type.

A type hierarchy is defined by declaring a super type, and a set of sub-types as follows:

```
SUPER_TYPE = SUB_TYPE1 | SUB_TYPE2 | SUB_TYPE3.
```

the symbol '|' is the join operator, and would be interpreted as disjunction.

Multilex hierarchies may be defined top down or bottom up. Bottom up definition is as follows:

```
SUB_TYPE = SUPER_TYPE1 & SUPER_TYPE2.
```

The symbol & is the meet operator.

A type definition can be built using combinations of the meet and join operators. The convention is as follows.

```
A = A1 | A2.
B = B1 | B2.
X = A & B & C & D.
```

is equivalent to:

```
A = A1 | A2.
B = B1 | B2.
X = (A1 | A2) & (B1 | B2) & C & D.
```

The conjonction and disjunction operators can also be used in the construction of feature terms, in which case they are interpreted as the meet on feature terms and the join of feature terms respectively.

1.4. Special types

Multilex uses special types for lists and strings. These are introduced here as a purely syntactic measure.

Lists are written using angle brackets, instead of using the clumsy first/rest notation. For example '< a b c d >'. The empty list is written '<>'. List elements can be accessed by naming each element in the list, or by using a 'head/tail' convention as follows:

```
<head . tail>
```

Strings are simply written between double quotes.

1.5. Macros

Macros are abbreviations for commonly used sub-structures. An example is:

```
3PS := AGR [person: 3, number: sing].
```

The left hand side is the macro name, the right hand side is the substructure, and the delimiter is ':='. The macro can be included in a subsequent AVM.

```
[syntax: [head: [syncat: v], agr: 3PS]].
```

Macros are similar to types, except that they are not included in the inheritance hierarchy. They can be used in feature structures in the same way as type symbols.

2. Examples

As an example, let's see the representation of the French lexical entry 'comparer' (to compare). We will restrict the semantic part to the definition in natural language.

```
;;   COMMON TYPES
;;   ============

BOOL = + | -.


;;   LEXICAL UNIT STRUCTURE
;;   ======================

SIGN = [gpmu: GPMU,
        syn: SYN,
        sem: SEM,
        example: T,
        cross_ref: T,
        maintenance: T,
        transfer: T].
;;   THE GPMU
;;   ========

GPMU = [orth: ORTH,
        phon: T,
        morph:MORPH,
        prag: PRAG].


;;   THE ORTHOGRAPHY
;;   ===============

ORTH = [lex: T,
        lex_type: ORTH_TYPE].

ORTH_TYPE = full | abbreviation.
```

The linguistic structure is defined in the same way (by defining the types: MORPH, PRAG, SYN and SEM). After that, we will define some macros and then the GPMUs. Finally, we will give the entries.

```
;;   MACROS
;;   ======

VBITRANS := [syn: [cat: verb, arity: 3]].

;;   SUBCATEGORISATION INFORMATION
;;   ============================
;;   This is again a combination of macros and types, to allow
;;   for sets of subcategorization lists.

SUBCATA := SUBCATA1 | SUBCATA2.

SUBCATA1 := < [cat: np] [cat: pnp, ptype: a!2] >.
SUBCATA2 := < [cat: np] [cat: pnp, ptype: avec] >.

SUBCATB := < [cat: np] >.

;;   THE GPMUS
;;   =========

COMPARER_GPMU = GPMU [orth: [lex: “comparer”,
                             lex_type: full],
                      phon: “kO~pare”,
                      morph: [cat: verb,
                              entry_type: lemma,
                              boundness: free_form,
                              inflection_prop: regular],
                      prag: [temporal: contemporary,
                             style: unmarked,
                             frequency: common,
                             social_group: all,
                             subject_field: general,
                             status: any,
                             geography: allfrench]].

;;   THE LEXICAL UNITS
;;   =================

comparer_lu_1 = SIGN &
                VBITRANS &
                [gpmu: COMPARE_GPMU,
                 syn:[subcat: SUBCATA,
                      ext_arg: +,
                      example: “comparer un e!1crivain avec un autre:
                                comparer un e!1crivain a!2 un autre:
                                comparer plusieurs artistes”,
                      controller: none,
                      nonoblcontrol: -,
                      active: +,
                      passive: +,
                      reflexive: -,
                      subc: main,
                      perfauxes: avoir_V_1,
                      bridgeverb: -,
                      polarity: neutral,
                      class: activity],
                  sem:[def: “e!1xaminer les rapports de ressemblance
                             et de différence”]].
```

```
comparer_lu_2 = SIGN &
                VBITRANS &
                [gpmu: COMPARE_GPMU,
                 syn:[subcat: SUBCATB,
                      ext_arg: +,
                      example: “comparer un e!1crivain avec un autre:
                                comparer un e!1crivain a!2 un autre:
                                comparer plusieurs artistes”,
                      controller: none,
                      nonoblcontrol: -,
                      active: +,
                      passive: +,
                      reflexive: -,
                      subc: main,
                      perfauxes: avoir_V_1,
                      bridgeverb: -,
                      polarity: neutral,
                      class: activity],
                 sem:[def: “Rapprocher en vue d’assimiler;
                            mettre en paralle!2le”]].

comparer_lu_3 = SIGN &
                VBITRANS &
                [gpmu: COMPARE_GPMU,
                 syn:[subcat: SUBCATA,
                      ext_arg: +,
                      example: “comparer un e!1crivain avec un autre:
                                comparer un e!1crivain a!2 un autre:
                                comparer plusieurs artistes”,
                      controller: none,
                      nonoblcontrol: -,
                      active: +,
                      passive: +,
                      reflexive: -,
                      subc: main,
                      perfauxes: avoir_V_1,
                      bridgeverb: -,
                      polarity: neutral,
                      class: activity],
                 sem:[def: “Rapprocher des personnes ou des choses
                            de nature ou d’espe!2ce diffe!1rentes,
                            dans une comparaison”]].
```

VI. Tools

Multilex aims at proposing standards for a multilingual lexical database. Among these standards, Multilex has defined different tools for the lexicographer. In this section, we will see the different tools that have been defined.

1. Editor

1.1. Overview

The editor will offer a user-friendly interface in order to create the different entries. It will use all the possible information to facilitate the work of the lexicographer, by handling inheritance, shortcuts, presentation and management of the different features according to the already stated values,…

1.2. Functionalities

1.2.1. External interface

The editor will visualize the needed features and their possible values. A user-friendly interface will be used with e.g. check boxes for boolean features, pop-up menus for features with a fixed set of possible values,…

It will check information and try to detect errors as soon as possible to allow the lexicographer to correct them.

1.2.2. Full entry generator

The tool itself will expand some information to a full attribute/value list and will solve abbreviated information, generate defaults when no input is given and carry out the inheritance rules.

1.2.3. Inheritance rules

A classical inheritance mecanism will be implemented in order to default values. Like every default value this one has to be checked by hand by the lexicographer.

1.3. Prototyping

A prototype of a dictionary editor called DicEdit has been developed on PC with Windows. It allows the lexicographer to edit entries with a very simple interface. It also checks some constraints (absence of obligatory values,…) and stores the data in a compressed format equivalent to MIF.

Another prototype has been developed using a structured document editor called Grif™ on Sun. Multilex considers a dictionary as a special kind of structured document. Using an editor that takes the description of a structured document as a parameter, Multilex has been able to prototype an editor in a very quick and generic way.

2. Browser

2.1. Overview

The scope of a dictionary browsing tool is to provide the user with fast and flexible access to the lexical information. This implies facilities to navigate through the different fields which constitute an article in order to retrieve, view and manipulate information of interest in whatever part of the dictionary it is stored. Ideally, in order to optimize the consultation process, the user will also have access to lemmatisation procedures.

2.2. Functionalities

The browser proposes a series of functions which can be used to have access to the dictionary. Any information contained in the dictionary should be usable as a key to access other informations.

The data query language used must be easy to formulate and as close as possible to natural language. It must enable the user to formulate an unrestricted variety of queries, to specify conditions on attributes of the entries, to select only the desired attributes and to format answers to queries.

The user is able to simply lookup lexical items (single word forms or all the forms of a lemma in a monolingual dictionary, select translation and display target language informations in a multilingual dictionary,…). The browser provides simple search functions and allow the user to define his own search function.

2.3. Prototyping

Multilex partners have reused a graph editor (written in Prolog on Sun) in order to visualize the structure of the dictionary. It allows the lexicographer to select LUs and to manipulate them graphically.

3. Defaulter

3.1. Overview

The defaulter produces default lexicon entries to be proof-read and post-edited by the human coder. The default entries are derived by various heuristics, using general linguistic knowledge (lexical rules) and knowledge retrieved from entries existing in the lexical database. The defaulter mainly supports the coding task to be performed by a human coder. It does so by defaulting lexical information for words to be added to the lexical database. It is used before the start of the real translation task.

3.2. Functionalities

3.2.1. System structure

The defaulter system consists of one language independant module, called “basic” and several language specific modules, each for one language. This facilitates the maintenance and modification of the system as well as the extension for additional languages.

The basic module includes:
- the external interfaces
- the interface between the defaulter and the lexical database

- functions that implements the heuristic strategies and a set of general utilities used throughout the defaulter, such as functions for handling lexicon entries copied from the database and functions for generating specific features and values.

The basic routines and heuristics use language specific tables, functions and control tables residing in the language specific modules.

3.2.2. External interfaces

The defaulter operates in two different modes: word-driven and file-driven.

The function `DEFAULT_WORD` takes as input a canonical form, a lexical category and a language. It returns monolingual lexical information in the form of feature value pairs.

The function `DEFAULT_FILE` takes as input a file with bilingual transfer entries and produces two monolingual files with defaulted entries out of it, one for the source and one for the target language.

3.2.3. Interface to the lexical database

Before a canonical form is defaulted, the word is always looked up in the lexicon under a given category and language. If the word is found in the database, the entry is retrieved and no defaulting takes place.

3.2.4. Separator analysis

If a separator is found in the string (a blank or an hyphen for instance), the word following the separator is looked up in the database under the given category and language. If an entry is found, then the new entry is derived from the one found in the database.

3.2.5. Prefix analysis

This function is comparable to the one described above except that it searches the canonical form for a prefix that has been defined as legal for the given language and category. It is called if the canonical form does not have a separator.

3.2.6. Suffix analysis

If no separator or prefix could be found, then the canonical form ending is analysed by a function that compares it with an endings table. Searching the endings table is done sequentially. When the first match is encountered, then the corresponding lexical unit and feature value combination are generated.

The ending table is sorted in order to apply the most specific ending-pattern first. Joker characters are allowed in an ending pattern.

3.2.7. Standard entries table

If no string or pattern matches the canonical form ending, then the word is defaulted according to a standard entries tables. This table contains default entries for all categories of the given language.

3.2.8. Language specific tables

The heuristics described above are applied successively to the canonical form to be defaulted. They are basically the same for all languages and categories, although the exact

content for tables will differ across languages and categories. In addition to these general heuristics, the defaulter has tables that are used for the handling of phenomena that are specific to only one language or even only one category of a language.

4. Integrity checker

4.1. Overview

Each time a Lexical unit is added or modified, the Multilex system will have to check some coherence and integrity constraints.

A *constraint* is a rule defined by the lexicographer. Activated contraint sets must be verified before and after each transaction.

There are different levels of constraint:

1. *Warning*: constraints of level 1 are just warnings: when the constraint is overridden, a message is passed to the lexicographer, but all treatments are allowed. The warning disappears as soon as the lexicographer validates the entry. These constraints can be used to detect potential errors.

2. *Delay*: constraints of level 2 can be overridden, but, in that case, the lexicographer receives a message and some treatments are forbidden on the concerned entries. An extraction request will run properly, but the detected entries will not appear in the extracted file. Interactive treatments such as browsing and editing are allowed. These contraints can be used to handle temporarily incomplete entries.

3. *Critical*: constraints of level 3 can't be overriden. If a transaction overrides such a constraint, it will be canceled (rollback) (all possible informations will be saved in a log to allow debugging).

There are different kinds of constraint:

1. *Integrity* rules apply on an article of the lexical database. They ensure that none of the article of the lexical database have an ill-formed configuration.

2. *Local coherence* rules apply on different articles of the same dictionary. They ensure that the dictionary is coherent.

3. *Global coherence* rules apply on different articles of different dictionaries of the lexical database. They ensure some coherence between dictionaries (if a sense is present in a monolingual dictionary, it must be an entry of bilingual dictionaries). This kind of constraint will be developed in another chapter.

These constraints have 3 main parts:
- an extraction expression which specifies the objects of the database that are concerned by this constraint,
- a boolean expression that must be verified by the concerned objects,
- a miscelaneous part that gives some information on the constraint (level, comment,…).

4.2. Definition of the constraints

These constraints have 3 main parts:
- an extraction expression which specifies the objects of the database that are concerned by this constraint,
- a boolean expression that must be verified by the concerned objects,

- a miscellaneous part that gives different information on the constraint (level, comment,…).

4.2.1. Integrity constraints

We can make the distinction between “structural constraints” and “linguistic constraints”.

Structural constraints operate on the structure of an article. Linguistic constraints operate on the content of an article. There is no distinction in the way to express these constraints.

Here are two examples of integrity constraints:

- **The maintenance record of an article is obligatory**
```
for_all (article) : present(article.Maintenance)
        { &level critical }.
```

- **For every article if attribute cat has the value** `verb`, **then** `synt_prop` **must have the value** `verb_synt_prop`
```
for_all (article) :
            ((article.syntactic_descr.cat = verb) <==>
             (article.syntactic_descr.synt_prop = verb_synt_prop))
     { &level critical
       &comment "check coherence of category and syntactic properties"
     }.
```

4.2.2. Local coherence constraints

local coherence constraints ensure that the dictionary is coherent. We give some examples of these constraints:

- **Attribute** `cat` **must have the same value in syntax and morphology** (this constraint applies on an article and on GPMUs). Remark: `#a` denotes the variable `a`.
```
for_all (article[syntactic_descr : [cat : #cat]
                 GPMU : #list]) :
            (for_all (GPMU) : ( in-list(GPMU.GPMU_name, #list) ==>
                                (GPMU.morphology.cat = #cat)
                              )
            )
        { &level critical
          &comment "Attribute cat must have the same value in syntax
                    and morphology"
        }.
```

4.2.3. Global coherence constraints

Global coherence rules apply to different articles of different dictionaries of the lexical database. They ensure some coherence between dictionaries (if a sense is present in a monolingual dictionary, it must be an entry of bilingual dictionaries).

The only difference between these rules and integrity/local coherence rules is the way the objects of the database are defined: we add the dictionary in which the object will be found (in the example below, we use the `french_english`, `french` and `english` dictionaries).

Here are some examples of global constraints:

- **For all entries of the bilingual dictionary, the source (and target) article must exist in the source (target) dictionary** (The example is shown for a French English dictionary)

```
for_all (french_english::fraeng_link[ mapping :
                                       [source_lu : @source_key,
                                        target_lu : @target_key]]) :
            ( exist(french::article [ lu_name : @source_key])
              and
              exist(english::article[ lu_name : @target_key]))
          { &level delay
            &comment "A link must refer to an existing article"
          }.
```

5. Transcriptor

The Multilex tools used by the lexicographer must handle the different writing systems. On the other side, a relational database only use a normalized set of characters. Hence, Multilex defines writing systems in a fomal way and proposes a minimal transcription for coding natural language strings at the MIF and MDB levels.

The backward and forward motion between the minimal transcription and its normal rendering is done by a transcriptor defined by the linguist.

In order to be concrete, here is an example of a multilingual text (English, French and Vietnamese[3]), with its minimal transcription and its normal rendering:

Minimal Transcription	Normal Rendering
:TRANS TMIN :WRSYS ENG *MRS. **PE!1LE!1 HAD TO BUY A JEWEL-CASE IN THE **MATY SHOP IN *BESANC!5ON FOR *CHRISTMAS.	Mrs. PÉLÉ had to buy a jewel-case in the MATY shop in Besançon for Christmas.
:TRANS TMIN :WRSYS FRE *MME. **PE!1LE!1 A DU!3 ACHETER UNE BOI!3TE A!2 BIJOUX CHEZ **MATY A!2 *BESANC!5ON POUR *NOE!4L.	Mme. PÉLÉ a dû acheter une boîte à bijoux chez MATY à Besançon pour Noël.
:TRANS TMIN :WRSYS VIE *BA!A **PE!3!D-LE!3!D D!9A!C PHA!BI MUA MO!3!ET HO!3!EP D!9O!3!A TRANG SU!19!D TA!EI HIE!3!EU **MATY *O!14!B XU!19!D *BO!19-DA!12NG-XO!3NG NHA!3N DI!EP LE!3!C *NO!3-EN.	Bà PẾ -LẾ đ ã phả i mua mộ t hộ p đồ trang sứ c tạ i hiệ u MATY ở xứ Bơ -dă ng-xông nhân dị p lễ Nô-en.

Multilex proposes to define the elements necessary to handle multiple writing systems in multiligual lexical data bases (and perhaps also in multilingual documents) in four sections :

- Character Sets
- Writing Systems
- Usages
- Normalised Codes.

---

[3] Multilex aims at the standardisation of lexicons for European languages, but the study on writing systems has been done with consideration to non European writing systems.

5.1. Character Sets

A character set is defined as a set of typographic signs used for writing some natural languages, augmented by attributes such as various diacritics (accents, tone…) or case (upper and lower). Character sets include alphabets such as Latin, Cyrillic, or Arabic, syllabaries like Kanas or Hangul, and ideogram sets like Hanzi, Kanji, etc. It is also necessary to take into account punctuation sets, numeral sets (Arabic, Roman, Thai, Chinese…), mathematical symbols, semigraphic symbols, etc.

A character set includes basic characters, possibly enriched with diacritics and case and compound characters having the status of autonomous characters in some writing systems (for example : ñ in Spanish, or sz in Hungarian…) or in some normalised codings (e.g. œ). The definition of a character set includes that of its "minimal transcription" (based on ISO 646). That transcription is readable and highly portable. Other transcriptions may be declared in order to take into account the needs of various data entry methods, or the facilities offered by specific devices.

For each set, a certain number of imperative constraints may be defined (for example: feasibility or unfeasibility of the combination between basic characters and diacritics).

For each set, a certain number of orderings is defined too; these orderings will be used by the different sorts (example: ASCII ordering, EBCDIC ordering, standard lexicographic orderings taking into account diacritics and case, etc.).

5.2. Writing Systems

A writing system uses a certain number of character sets. For instance, the French writing system uses Roman characters, Roman punctuation marks, Arabic digits, and some special signs (%, &, $, £, @, §…). A writing system may impose some specific constraints (for example, the cedilla is used only on c in the French writing system, whereas it is used on a and e in the Polish writing system).

In regard to a given writing system, there are usually several methods for sorting character strings. A sorting method can be defined by choosing an ordering of the character sets and an ordering for each set. For example, in the French writing system, letters usually come before numbers, numbers before punctuation marks, and punctuation marks before special signs. For sorting letters, the case has often a lower priority than the diacritics.

5.3. Usages

A usage is a rule (or a set of rules) which can apply to one or more writing systems. The first kind of usage concerns the direction of writing, which can go left to right and top to bottom (English, French…) or from right to left and top to bottom (Arabic, Hebrew, Farsi, Hindi…), or from top to bottom and right to left (Chinese, traditional Japanese…) etc.

A second kind of usage concerns context-sensitive presentation. Such usages describe typographic changes like ligatures (for example : ae -> æ, oe -> œ, fi -> fi in French), or character shape changes depending on the context, as we find in the Arabic or Greek writing systems (an Arabic character can take four different shapes, depending on whether it stands by itself, or at the beginning, in the middle or at the end of a word).

A third kind of usage concerns hyphenation. Some writing systems, such as Arabic, Thai, Japanese, or Chinese, have no hyphenation, while other writing systems may have several possible sets of hyphenation rules.

A fourth kind of usage concerns fine typography. For example, double punctuation marks (, . : ; ? ! "…) must be preceded by a small space in French, but not in English. Other rules of that kind might be defined to control kerning (e.g. kerning A V gives AV).

5.4. Normalized Codes

Finally, Multilex includes a section to describe the mappings from normalized codes such as various ISO norms, EBCDIC, GB2312-80, JIS… to characters. A given code (e.g. that of ñ in ISO-025) may correspond to several characters (here n+~ and the autonomous ñ), possibly in several character sets.

5.5. Prototyping

Multilex has developed a tool that performs the forward and backward motions between the internal transcription and the normal rendering of strings. This tool is fully parametrizable by the linguist. In order to define a particular transcriptor (e.g., for French strings), the linguist has to define a finite state automaton. Multilex has developed 4 transcriptors:

- from minimal transcription to normal rendering
- from normal rendering to minimal transcription
- from minimal transcription to TEI transcription
- from TEI transcription to minimal transcription.

6. Import/Export

Import from other databases and export to other databases or to application dictionaries are planned. Import/Export facilities are allowed by a SGML exchange structure following the TEI guidelines. This part has not been covered by the first phase of Multilex. It will be done during the second phase.

The SGML exchange structure will be generated by a generic tool guided by a high level language which describes equivalence between the internal structure and the export structure.

The SGML structure will be generated according to the TEI guidelines.

VII. Bibliography

Multilex reports

Brust-Braun M. and U. Knops 1992, *The Multilex Database Structure*. Multilex report, February 1992.

Fedder L. 1992, *The Multilex Internal Format*. Multilex report, June 1992.

Garcia X and M. Meya 1992, *Remote Access to a Multilex Site*. Multilex report, February 1992.

Hook S. P. and K. Ahmad 1992, *Terminology Management Systems: Definition of the Standard and Database Interface*. Multilex report, February 1992.

Khatchadourian H. 1992, *Tools, functional specifications*. Multilex report, February 1992.

Knops U. and G. Thurmair 1991, *Defaulter*. Multilex report, November 1991.

Knops U., W. Martin and N. Modiano 1992, *Multilingual Description of Lexical Items.* Multilex report, June 1992.

Martin W., E. Ten Pas, H. Demeersseman, J. Paul and M. Vliegen 1992, *Terminological Description of Lexical Items*. Multilex report, June 1992.

McNaught J., S. Smith, J. Odijk, D. Clemenceau, E. Roche, M. Gross, H. Schnelle and C. Kunze 1992, *Monolingual Description of Lexical Items*. Multilex report, June 1992.

Modiano N. 1992, *Linguistic Architecture*. Multilex report, June 1992.

Peters C. and E. Picchi 1991, *Dictionary Browsing Tool Requirements*. Multilex report, November 1991.

Phan H. K. and C. Boitet 1992, *Defining Writing Systems and Transcriptions for Multilingual Lexical Databases.* Multilex report, February 1992.

Sérasset G. 1991, *Ψ-termes et dictionnaires.* Multilex report, November 1991.

Sérasset G. 1992, *Defining a Database, an example*. Multilex report, June 1992.

Sérasset G. and C. Boitet 1992, *General Tools definition*. Multilex report, January 1992.

Spanu A. 1992, *Proposal for a typed feature structure (TFS). Representation system for semantic information.* Multilex report, February 1992.

Other reports

Aït-Kaci H. 1986, *An Algebraïc Approach to the Effective Resolution of Type Equations*, Theoretical Computer Science, vol. 45 : pp. 293-351.

Aït-Kaci H. and P. Lincoln 1988, *LIFE : a Natural Language for Natural Language*, MCC, TR ACA-ST-074-88, février 1988, p. 27.

Aït-Kaci H. and R. Nasr 1986, *LOGIN : a Logic Programming Language with Built-in Inheritance*, JLP, vol. 3 : pp. 185-215.

Battista Varile G. and A. Zampolli 1992, *Synopses of American, European and Japanese Projects Presented at the International Projects Day at Coling 1992*. Linguistica Computazionale vol. VIII, July 1992, p. 71.

Emele M. and R. Zajac 1990, *Typed Unification Grammars*. Proc. of the 13th International Conference on Computational Linguistics, COLING-90, Helsinki, August 1990.

Emele M. et al. 1990, *Organising linguistic knowledge for multilingual generation*, Proc. of COLING-90, Helsinki, vol. 3/3, pp. 102-107.

Phan H. K. and C. Boitet 1992, Multilinguisation d'un éditeur de documents structurés, application à un dictionnaire trilingue. Proc. of COLING-92, Nantes, vol. 3/4, pp. 966-971.

A.2 Genelex

I. Introduction

1. Overview

Genelex is a Eureka project involving about 250 man-years. The participants of Genelex in France are:

- Bull,
- GSI-Erli,
- Hachette,
- IBM,
- LADL,
- SEMA,
- In Italy:
  - Lexicon,
  - University of Pisa,
  - SERVEDI (UTET-PARAVIA),
- In Spain:
  - SALVAT,
  - TECSIDEL,
  - University of Barcelona.

2. Goals

The objective of Genelex (GENEric LEXicon) is to construct a generic dictionary in various European languages (for now, French, Italian and Spanish). This involves not only the development of a generic dictionary but also the definition of a strategy guaranteeing that it will remain generic.

The dictionary should be considered as a large lexical database, with no direct connection to any NLP application. The lexicons dedicated to applications will be obtained by the extraction of the data designed from the generic dictionary in a form adapted to needs.

The Genelex project has three types of tasks to carry out:

- defining a data model representing the structure of a word (or term) and allowing to store it into a database.
- constructing a set of software tools adapted to the model and allowing to convert into the Genelex model the existing dictionaries and to manipulate the generic dictionary (browsing and updating) and to extract specific dictionaries. The software tools that Genelex is developing are the following:
  - a software package to help a lexicographer to merge different dictionaries,
  - a software package to parse texts in order to update the generic lexicon,
  - a lexicographer workstation,
  - a generator software package (generating a sub-lexicon dedicated to an application).
- effectively constructing a generic dictionary in a specified linguistic model using Genelex software tools.

3. Summary of results

These different tasks were already approached in the context of a French feasibility study which finished in September 1990.

Today, the French model is developed and used by the partners.

The model currently handles the following amount of lexical entries (each partner has its own dictionary):

- SEMA (from LADL tables): about 70,000 morphological units (among which 4,000 verbs with a complete syntactic description),
- IBM: about 50,000 morphological units,
- GSI-ERLI: about 68,000 simple morphological units, and about 15,000 compound morphological units.

A Genelex User Group has been created. The contact is:
Marc Nossin,
GSI Erli,
1 place des Marseillais,
94227 Charenton-le-pont cedex,
Tel: +33 1 48 93 81 21,
Fax: +33 1 43 75 79 79,
e-mail: marc.nossin@erli.gsi.fr

II. Overview

The Genelex model is based on an entity relationship schema.

The global schema of an entry is given in figure 1.

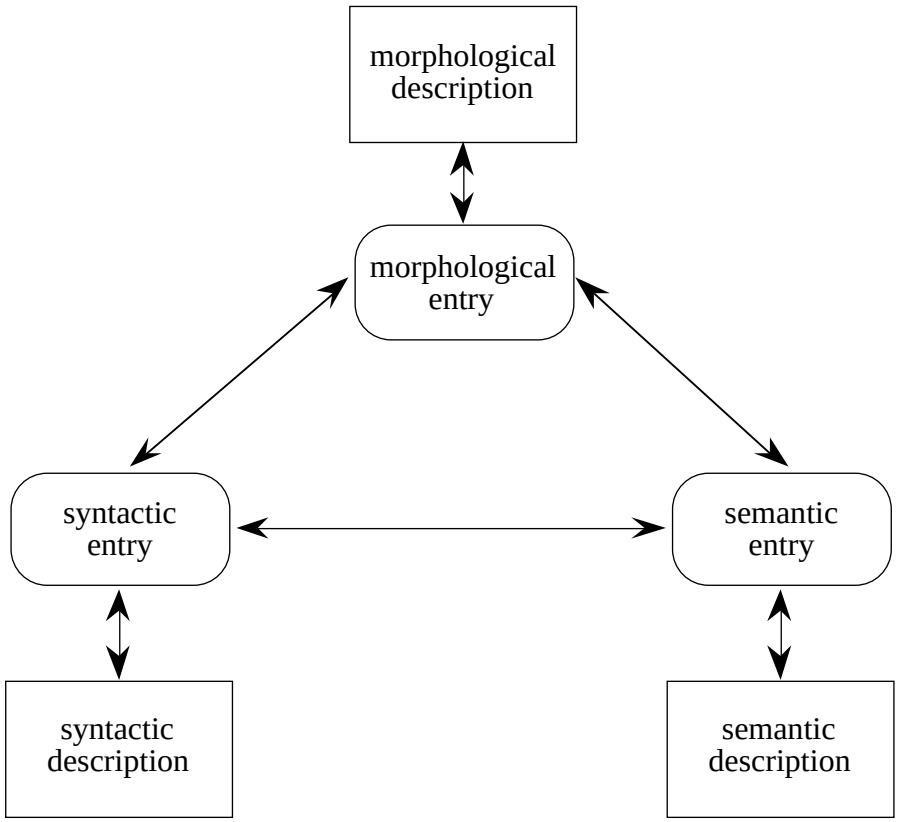


Figure 1: A reduced view of the Genelex model.

Many constraints have been expressed in this schema (types of objects, types of relations, cardinality of relations, …). However, this schema is not adapted to express rules. Hence, some constraints are expressed in natural language in a separated document.

A DTD (Definition Type Document) has been written in SGML as a concrete representation of the conceptual model. Some of the constraints in natural language have been expressed in the DTD.

III. Morphology

2.1. Morphological Units (MU)

Morphological Units are separated in two groups:

- Autonomous morphological units are units that appear individually in the language.
  Ex: *demain* (tomorrow), *aujourd'hui* (today)
  matin (morning), *après-midi* (afternoon)
  se la couler douce (to take it easy)

- Non autonomous morphological units are units that appear only in derived words, locutions, compound words or set phrases.
  Ex: *parce (appears only in 'parce que' (because))*
  tohu (appears only in 'tohu bohu' (hubbub))
  bohu (appears only in 'tohu bohu' (hubbub))
  aequo (appears only in 'ex aequo')
  -tion (appears only in derived words like 'definition').

2.1.1. Autonomous Morphological Units

Autonomous morphological units have a grammatical category, one or more syntactic properties and eventually one or more meanings.

Autonomous MUs can be simple, derived or composed.

A MU which can not be inflected is defined by a graphy (possibly with variants) and a grammatical category.

A MU which can be inflected is defined by an equivalence class of inflected forms and a category. E.g.: *{cheval, chevaux}*, *{peau-rouge* (masc. sing.), *peau-rouge* (fem. sing.), *peaux-rouges* (masc. plur.), *peaux-rouges* (fem. plur.)*}*.

Some criteria to decide on the splitting of MUs have been defined:
- The most general splitting criterion is the category. E.g.: the noun *autiste*: *{autiste, autiste, autistes, autistes}* (autist) and the adjective *autiste*: *{autiste, autiste, autistes, autistes}* (autistic) are two distinct morphological units.
- The gender is a splitting criterion for nouns unless the meaning is equivalent (apart of the sex or gender).
- The signification can be a splitting criterion if no link (etymology, derivation, rhetoric) can be established between 2 significations. The lexicographer decides whether to split or not.
- The function of a pronoun is a splitting criterion. E.g.: le (object) and il (subject) are 2 distinct MUs.

2.1.1.1. Simple MUs

A simple morphological unit is identified by a string of alphabetical characters, excluding the space, the hyphen, the apostrophe.

2.1.1.2. Derived MUs

In terms of character strings, these units do not differ from simple MUs. Nevertheless, these units are "analyzable", in the sense that they consist of:

- 0 to N prefixes,
- one base,
- 0 to N suffixes.

The derivation process is a recursive process (see figure 2).

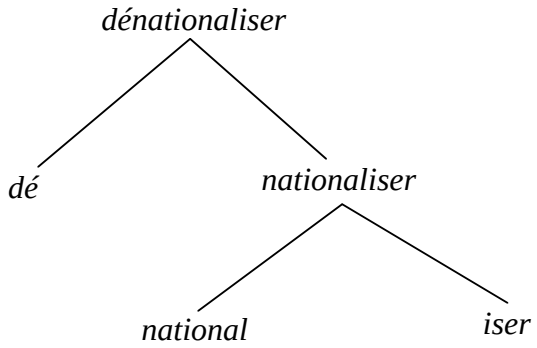


Figure 2: The structure of a derived MU.

2.1.1.3. Compound MUs

Compound MUs are complex expressions which satisfy at least one of the following criteria:

a. one of the component appears only in this complex expression
   E.g.: *aujourd'hui ('hui' appears only in this expression)*
   au fur et à mesure ('fur' appears only in this expression).

b. morphological particularity
   E.g.: *une deux-chevaux (this expression is feminine and singular, but 'chevaux' is masculine and plural)*
   un peau rouge (this expression is masculine, but 'peau' is feminine).

c. graphical particularity
   E.g.: *garde-malade.*

d. indivisible compound (no insertion allowed)
   E.g.: *à force de.*

e. not semantically composed
   E.g.: a *sage-femme* (midwife) is not a wise *(sage)* woman *(femme).*

A compound MU is defined by its components. A component can be either an autonomous MU or a non autonomous MU.

Each compound MU has a grammatical category and a set of combination of morphologic features (combTM). E.g.: *peau rouge*, catgram: Nom, combTM: ms mp fs fp.

A compound MU has N composition relations, one for each component. Each relation relates the compound MU, one of its component and the compositional inflection mode that describes the compound. It also give the order of the component in the compound and the joining character preceding the component (such as hyphen, quote or space).

2.1.2. Non autonomous Morphological Units

Non autonomous morphological units are units that appear only in derived words, locutions, compound words or set phrases.

A grammatical category can be associated to these units. In such a case, it is chosen arbitrarily, or according to etymological considerations.

2.2. Characteristics of Morphological Units

2.2.1. Phonetic and graphic systems

2.2.1.1. Graphic system

The MUs are identified by their graph i.e. by a string of characters, accentuated or not, in upper or lower case. [a-z, A-Z, ç, é, è, ë, ê, á, à, â, ä, å, æ, œ, ÿ, î, ï, ô, ö, ó, ù, û, ü, ñ, É, È, Ë, Ê, Á, À, Â, Ä, Î, Ï, Ô, Ö, Ó, Ù, Û, Ü], space, hyphen, apostrophe and underline (mark of possible hyphenation).

The coding of accentuated characters is the coding defined by the TEI. This coding is transparent to the lexicographer.

A MU can have graphical variants, called by Genelex Graphical Morphological Units (GMU).

Example: *clé, clef* 1 MU, 2 GMU.

The graph of a compound MU is calculated from its components rather than stored in a GMU. The graphical variation of a compound has two origins:

- variation on the joining character. E.g.: *col-vert, colvert* (these variations are stored in the composition relation);
- graphical variation of a component. E.g.: *porte-clef, porte-clé* (these variations are variations of the component).

2.2.1.2. Phonetic system

A phonetic transcription identifies the pronunciation of a MU. This transcription consists in a string of phonemes.

A MU can have phonetic variants.

The phonetic system of a compound is analoguous to the graphic system.

The symmetry between phonetic and graphic informations is represented in the Genelex model.

2.2.2. Grammatical categories

The Genelex consortium had long discussions to determine whether the model should use the traditional categories (noun, verb, article, adjective, pronoun, preposition, adverb, conjunction, interjection, particle).

Genelex has introduced the category “determiner” in the standard list. This category includes the traditional articles, demonstrative and interrogative adjectives, indefinite adjectives, possessives, and cardinals numerals. E.g. *les, deux, certains, notre* (the, two, some, our).

Genelex has defined the following categories: noun, adjective, determiner, adverb, verb, preposition, pronoun, interjection, conjunction and particle.

2.2.3. Morphological attributes

Adjectives, determiners, nouns and pronouns admit the attributes gender and number.

Verbs admits the attributes mood, tense, person, gender and number.

Adverbs, prepositions, conjunctions and interjections admit no morphological attributes.

Genelex has defined the following attributes:

Attribute	Possible values
Gender	`masculin, féminin, neutre`
Number	`singulier, pluriel`
Mood	`indicatif, subjonctif, conditionnel, impératif, infinitif, participe`
Tense	`présent, imparfait, passé simple, futur, passé`
Person	`1, 2, 3`

2.2.4. Inflected forms

A graphical inflected form consists of a graph and a combination of morphological attributes.

Example: *buvait* (drank):
Mood: indicatif, Tense: imparfait, Person: 3, Number: singulier,
boire (to drink):
Mood: infinitif, Tense: présent, Person: 1/2/3, Number: sing/plur,
grandes (big):
Gender: féminin, Number: pluriel.

Only verbs, adjectives, adjectival determiners and nouns have an inflectional system.

The inflection is the description of an equivalence class of inflected forms.

The inflectionnal mode is the method used to construct an equivalence class of inflected forms for a MU.

2.2.4.1. Inflection of simple MUs

2 mechanisms are allowed for the construction of inflected forms:
- addition of an ending to a base,
- deletion and addition of characters to a graphical or phonemical MU.

The first mechanism is interesting for verbal inflection since:
- the base is always different from the graphical MU,
- one can occasionally select a base among a set of bases.

Example: *chanter* (to sing)

indpre30p	RADG1: *chant*	ADJUNCT: *ent*	-> *chantent*
finir (to finish)			
indpre30p	RADG1: *fin*	ADJUNCT: *issent*	-> *finissent*
aller (to go)			
indpre20p	RADG1: *all*	ADJUNCT: *ons*	-> *allons*
indpre30p	RADG2: *v*	ADJUNCT: *ont*	-> *vont*

The second mechanism is adapted to some inflected forms (like *{empereur, impératrice, empereurs, impératrices}* (emperor, empress,…)).

A “joker” is allowed in the deletion and addition of characters. The joker is noted “$”. It can be replaced in a pattern by one or more characters.

Example: deletion: *é$er*, adjunct: *è$e*.
célébrer, il célèbre (to celebrate)
assécher, il assèche (to dry)
exécrer, il exècre (to execrate)
céder, il cède (to give up)

Inflectional variants are allowed:

- variation of the base: it is possible to use 2 different bases for the same inflected form.
  E.g.: *asseoir* (to seat) indipf10s
  RADG1: *assey* ADJUNCT: *ais* -> *asseyais*
  and RADG2: *assoy* ADJUNCT: *ais* -> *assoyais*

- variation of the ending:
  E.g.: *media* (media) mp
  RADG0: *media* ADJUNCT: *s* -> *medias*
  RADG0: *media* ADJUNCT: -> *media*

- combination of variations:
  E.g.: *scenario* (scenario) mp
  RADG0: *scenario* ADJUNCT: *s* -> *scenarios*
  RADG1: *scenari* ADJUNCT: *i* -> *scenarii*

2.2.4.2. Inflection of compound MUs

Inflected forms of a compound MUs are constructed from the inflected forms of its componants.

An inflectional mode for compounds is located in the composition relations linking the compound and its componants. This inflectional mode allows to select the relevant inflected form for the component.

Example: The compound MU for *Un peau rouge, des peaux rouges, une peau rouge, des peaux rouges* will have the following 2 relations:

composition relation	
UM-C *peau rouge*	UM-S *peau*
inflectional mode	
Compound	Component
ms	fs
mp	fp
fs	fs
fp	fp

composition relation	
UM-C *peau-rouge*	UM-S *rouge*
inflectional mode	
Compound	Component
ms	fs
mp	fp
fs	fs
fp	fp

2.2.5. Etymology

A MU can have competitive etymons, a string of etymons or one or more etymons (composition or lexical derivation).

Example: *mécanisme*: du latin *mechanisma*
élastique: du latin *elasticus*, du grec *elasis*
plénipotentiaire: du latin *plenus* et *potentia*.

2.3. Some examples of MUs

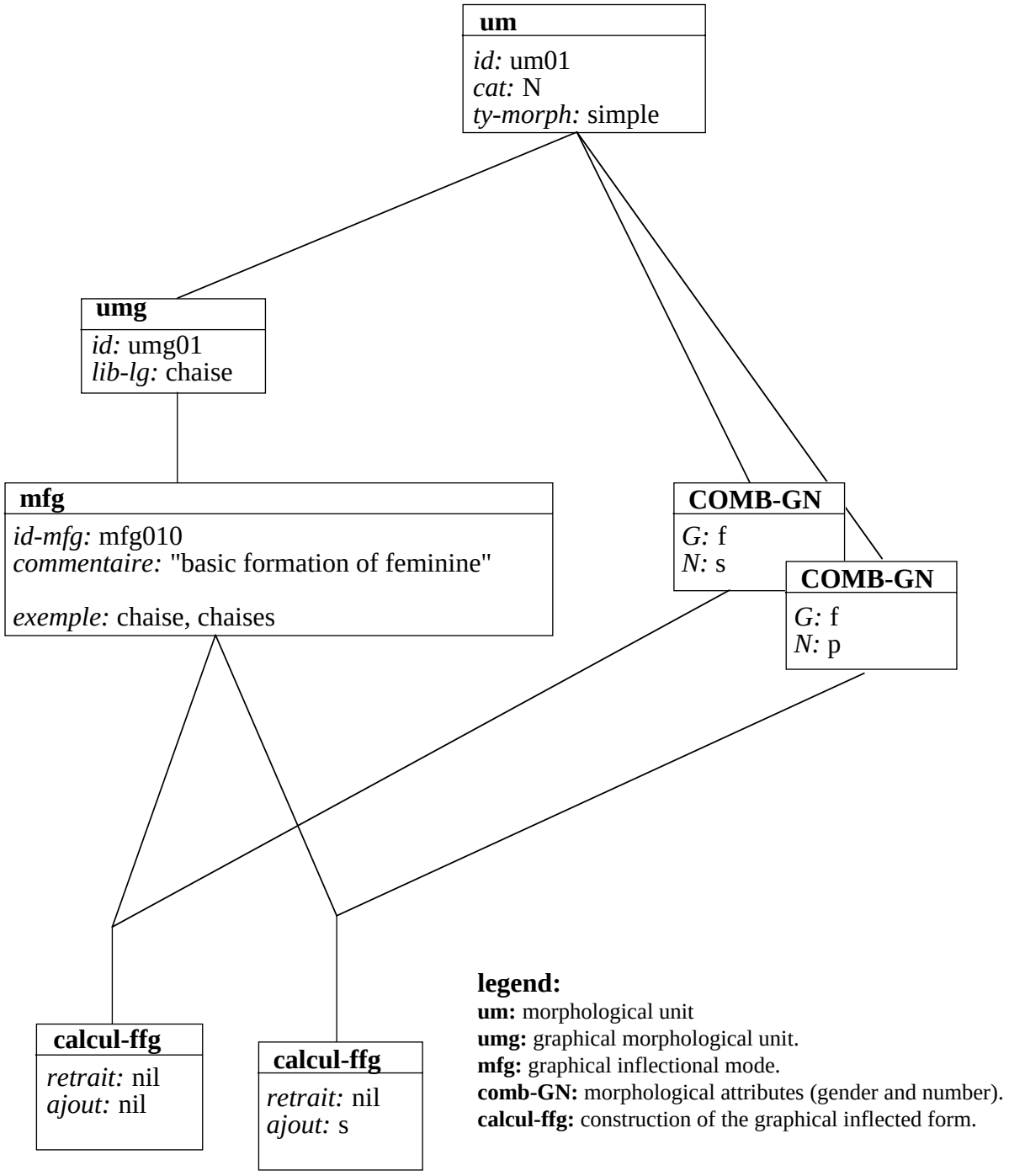


Figure 3: The morphological unit for the noun “chaise” (chair).
This noun is regular

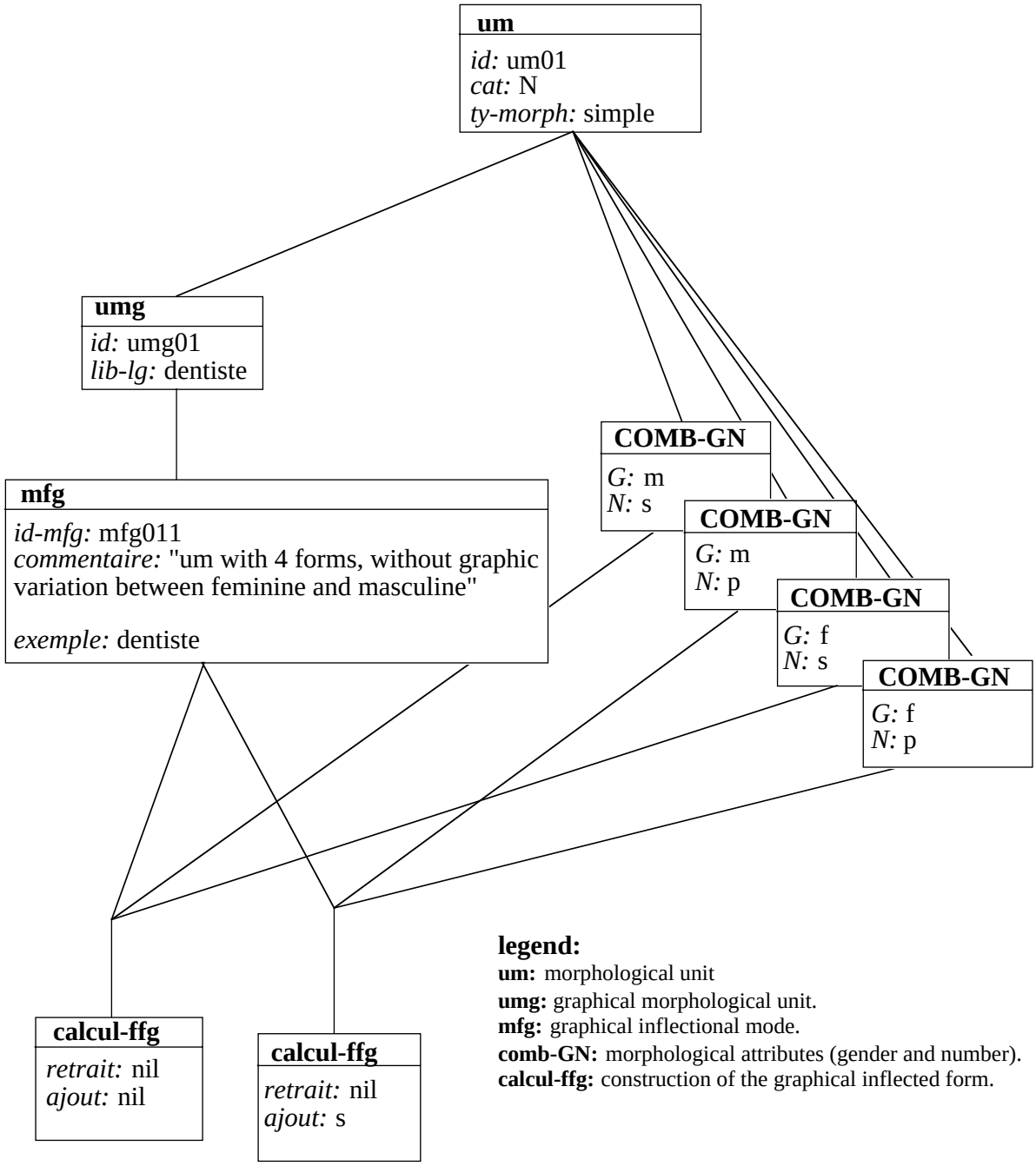


Figure 4: The morphological unit for the noun “dentiste” (dentist).
This noun has no graphical variation between masculine and feminine.

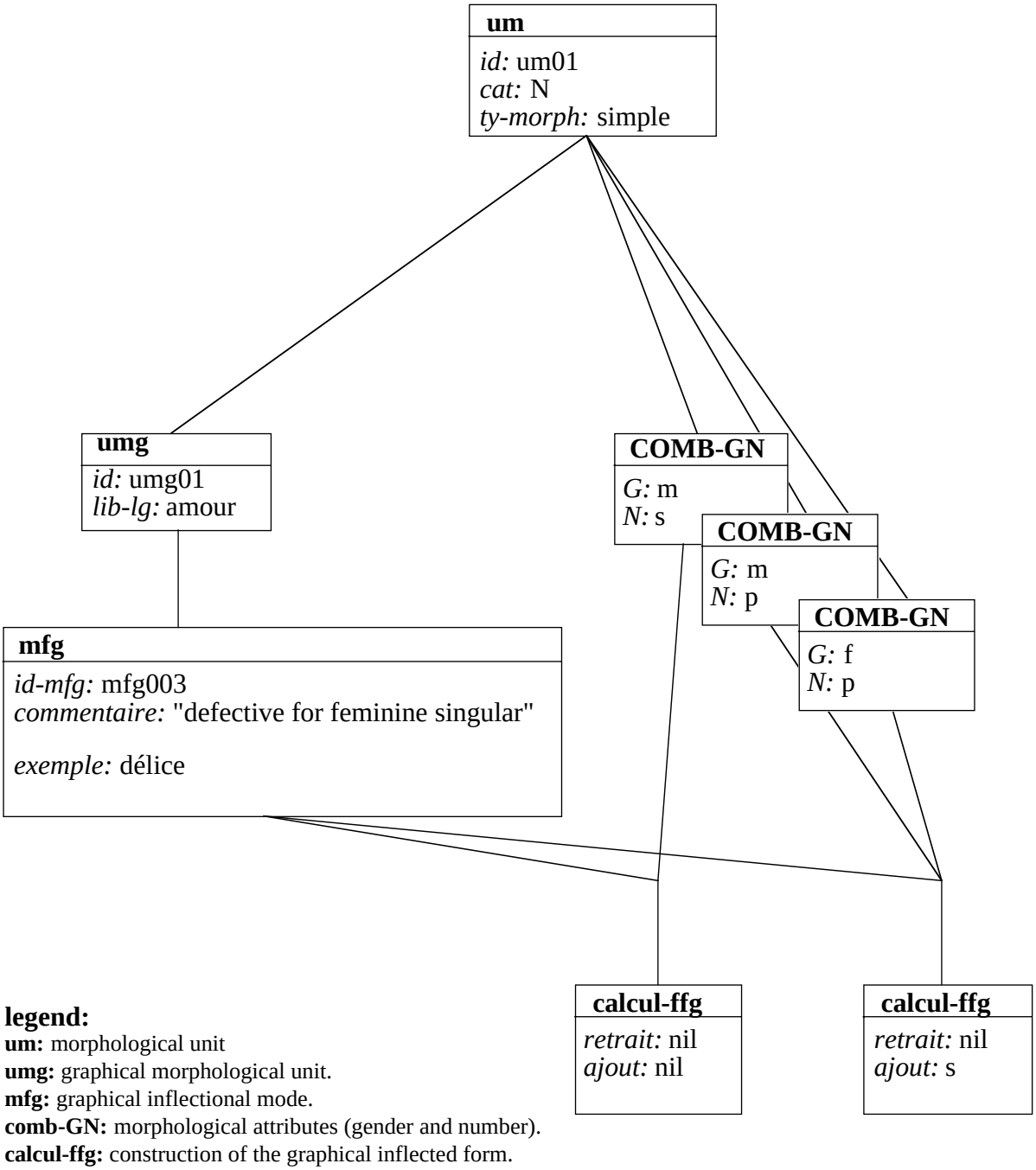


Figure 5: The morphological unit for the noun “amour” (love).
This noun is defective in feminine singular.

IV. Syntax

A morphological unit can have one or more syntactic behaviors.

The Syntactic Units (SynU) describe these syntactic behaviors.

The following model for simple words is valid for compounds, as the representation of the syntactic compounds is based on the same formalism.

3.1. Basic notions

3.1.1. Syntactic Units

3.1.1.1. Definition

The Syntactic Unit (SynU) is the entry at the syntactic level.

It describes one and only one of the syntactic behaviors of a morphological unit.

Each morphological unit is related to at least one Syntactic Unit (SynU). If a Morphological Unit (MU) has several syntactic behaviors, it will be related to several Syntactic Units (SynU).

A SynU corresponds to one and only one MU.

3.1.1.2. Splitting criteria

The formal splitting criterion of SynU is the fact that a SynU has one and only one basic construction criterion (see definition of construction criterion in the next paragraph). An example of splitting is given in figure 6.

Pierre arrive à partir (Pierre manages to leave)	1 SynU
Pierre arrive à Paris (Pierre arrives at Paris)	1 SynU
Pierre arrive (Pierre arrives)	

Figure 6: Example of splitting.

3.1.2. Construction criteria

Each syntactic unit is described by a set of construction criteria, i.e., one basic construction criterion and 0 to N construction criteria derived from the basic one.

The construction criterion is defined by a list of positions P_i ($0 \leq i \leq N$), with their optionality and their solidarity.

3.1.3. Position

3.1.3.1. Formal definition

The position is an element of the definition of a construction criterion.

Formally, a position is defined by a triplet: distribution, function and thematic role.

A distribution on a position is the set of syntagma that can instanciate this position.

A position has 0 or 1 function and 0 or more thematic roles.

Each element of this triple can be empty, but not at the same time.

There are different types of position:

- HEAD: This position supports the restrictive features presented later.
- P_i ($0 \leq i < 4$): where i gives the rank of the position among the other.

Example:
Ces problèmes la regardent (These problems concern her)
Qu'elle ne vienne pas la regarde (The fact that she does not come concerns her)

```
P0      SN
        P[mode:infinitif]
        P[mode:subjonctif]
```

Les enfants dorment (The children are sleeping)

```
P0      SN
```

3.1.3.2. Properties

Position and construction criteria

A construction criterion contains 1 to 4 positions, with, possibly, the head.

A same lexical unit can have one or more complementation schema based on its positions.

Example:
voler transitive (to steal)

```
ccb: P0 HEAD (P1)
```

voler intransitive (to fly)

```
ccb: P0 HEAD
```

arriver (to arrive)

```
ccb: P0 HEAD (P1)
P1: SP[sscat:lieu]
```

P1 is optional and defined by a SP, subset of prepositions included in the "prepositions of place".

arriver modal (to manage)

```
ccb: P0 HEAD P1
P1: SP[lex:à]
    P[mode:infinitif]
     [lex:à]
```

P1 is obligatory.

Head position

The head position identifies the place where the entry is inserted, and may have an associated set of restrictions (morphological, auxiliaries, aspect,…).

Linearity

"Position" does not refer to the strict meaning of place as the constraint of surface linearity has been rejected.

In case of optional position, the omission of a position must not lead to the redefinition of the following positions. E.g.: the structure `P0 ((P1) (P2))` can be realized by `P0 P1 P2`, `P0 P1`, `P0 P2` and `P0`. The omission of P1 does not lead to the redefinition of `P2` as `P1`.

The surface order is not constrained. E.g. *Je promets de venir à Pierre*, *Je promets à Pierre de venir* (I promise Jean to come) correspond to the same construction.

It is possible to constrain the surface order, when necessary, by "fixing" a position to its rank (using an exclamation point). E.g.:
Il craint de Marie qu'elle ne vienne (He is afraid Marie will come).

```
ccb: P0 HEAD !P1 P2
P0: SN
P1: SP[lex:de]
P2: P[sscat:complétive].
```

Solidarity

Some positions stick together, i.e., no insertion is allowed between them. This constraint can be expressed with a hyphen between the 2 positions: `P1-P2`.

Optionality

Some positions are optional. This optionality is expressed with brackets. E.g.:
- *Pierre parle de sa soirée à Marie* (Pierre speaks about his evening with Marie)
  Pierre parle à Marie (Pierre speaks with Marie)
  Pierre parle de sa soirée (Pierre speaks about his evening)
  Pierre parle (Pierre speaks)

```
ccb: P0 HEAD ((P1) (P2))
```

It is also possible to express rare cases, either by giving all the possible extensions without brackets or by giving more then one bracketted construction. E.g.:

```
- ccb: P0 HEAD (P1) ((P2) (P3)) &
       P0 Head P2 ((P1) (P3))
i.e.:
  ccb: P0 HEAD P1 P2 P3
       P0 HEAD P1 P2
       P0 HEAD P1 P3
       P0 HEAD P1
       P0 HEAD P2
```

inter-conditioned positions

In some cases, the realization of a position can constrain the realization of another position. E.g.:

- *Pierre répond que c'est exact* (Pierre answers that it is true)
  Pierre répond à la question (Pierre answers to the question)
  Qu'il ait une telle attitude répond à la question (His attitude answers the question)

but not: *Qu'il ait une telle attitude répond que c'est exact* (His attitude answers that it is true)

```
ccb: P0 HEAD ((P1) (P2))
P0: SN
    P[sscat:complétive]
     [mode:subjonctif]
HEAD: V[lex:self]
P1: SP[lex:à]
P2: P[sscat:complétive]
     [mode indicatif]
cond: si P0 == P[sscat:complétive]
                [mode:subjonctif]
      alors P2 != P[sscat:complétive]
                   [mode indicatif].
```

Positions as functions

One can associate a function name to a position. These functions are relative to the head.

Example:

- *Jean aime Marie* (Jean loves Marie)

```
ccb: P0 HEAD ((P1) (P2))
P0 sujet
P1 objet-direct
```

Thematic role of the position

Genelex allows the specification of the thematic role of a position. A first list of possible values has been defined:

Agent	Entity that makes a deliberate action
But	Destination of the action (physic or psychic)
Destinataire	human destination of an action that implies a transfer (physic or psychic)
Instrument	instrument, cause, means of the action
Loc	place of the action
Patient	person that is affected by the action
Source	place or person that is the origin of the action
Temps	temporal entity (+ duration)
Thème	the rest

It is possible to define other values.

3.1.4. Syntagma

A syntagm is a given realization of a position. A syntagm is formally defined by a syntagmatic label, associated with a set of constraints (rewriting constraints and restrictive features).

The labels are taken among syntactic symbols with terminal categories. These symbols are supposed to be known and described in a grammar outside the dictionary.

When an entry imposes a structural constraint on the realization of a syntagm, it is possible to express these constraints with the formalism of construction criterion. E.g.:

- *Il est très intéressant de remarquer cela* (It is very interesting to notice this)
  Il est très intéressant que tu remarques cela (It is very interesting that you notice this)

```
ccb: P0 HEAD P1
P0: PRO[lex:il] [sscat:impersonnel]
HEAD: V[lex:être]
P1: SADJ: P0 HEAD P1
    P0: SADV
    HEAD: ADJ[lex:self]
    P1: P[mode:infinitif]
         [lex:de]
        P[sscat:completive]
         [mode:subonctif]
```

This introduces recursively the possibility to code syntactic trees.

3.1.5. restrictive features on position and syntagma

The syntagma can be constrained by a set of features that specify as precisely as possible a realization of a position.

A constraint on a feature is expressed between square brackets, by giving the feature and its value. Several constraints can be added to a syntagm. E.g.:

```
V[aux:être]
 [pronominal:se]
 [temps:composé]
 [aspect:processif]
```

In the following paragraphs, we will describe the features used to express constraints.

3.1.5.1. Lexical features

Restriction on the lexical realization of one of the components of the syntagm.

E.g.:

`SP[lex:à]`
SP with the preposition “à”.

`V[lex:self]`
“self” is a keyword to refer to the described entry.

3.1.5.2. Morphological features

Restriction on the value of a morphological feature (mood, tense, person, gender, number) of a syntagm or one of its components. E.g.:

```
SN[nombre:pluriel]
```

3.1.5.3. Morpho-syntactic features

These features are:

- morpho-syntactic subcategory of one of the components of the syntagm,
- value of the auxiliary,
- value of the non referential pronoun of the verb in a compound,
- value of the negation when it is obligatory in a verbal construction.

Example:

```
SN[sscat:def]
V[aux:avoir]
V[pronominal:se]
V[neg:ne-pas]
```

3.1.5.4. syntactico-semantic features

Coreference indexes manipulated in distributional grammars can be used. This allows the representation of reflexive and reciprocal verbs.

Example: *Marie se regarde* (Marie looks at herself)

```
ccb:  P0 HEAD P1
P0:   SN[coref:i]
HEAD: V[lex:self]
P1:   PRO [coref:i][sscat:faible]
```

3.1.5.5. semantic features

These features can express the aspect of a verb, as well as denotational conditions and semantic classes.

3.1.6. Transformations

One can store transformations between Syntactic Units, between construction criteria and between syntagma. The lexicographer decides which level will be used for transformations. Genelex only gives recommendations:

- transformations between Syntactic Units should be reserved for distinct semantic usages for different syntactic structures (e.g., transitive or intransitive usage of neutral verbs).

- transformations between construction criteria should be reserved for different syntactic forms that do not imply fundamental semantic differences (e.g., passivation).

- transformations between syntagma should be reserved for identical syntactic structures where a syntagm at a position is a transformed into another syntagm at the same position.

As an example, we will show how passivation is represented as a transformation between construction criteria:

Jean casse la branche (Jean breaks the branch)

La branche est cassée par Jean (The branch is broken by Jean)

```
ccb:  P0 HEAD P1
P0:   SN
HEAD: V[lex:self]
P1:   SN

calcul: “passif en par”
 P0 = P1
 P1 = P0

cct:  P0 Head P1
P0:   SN
HEAD: V[lex:self]
P1:   SP[lex:par]
```

3.2. Definition and properties of syntactic compounds

3.2.1. General definition

Syntactic compounds are groups of syntactically well-formed words. They are complex elements with the following characteristics:

- No graphical separator (except blank and apostrophe),
- General agreement rules are respected among the components,
- Syntactically well-formed and, possibly, the properties of modification, substitution, transformation, insertion, deletion and moving are applicable.

3.2.2. Coverage

These syntactic compounds cover:

- **set expressions**: proverbs, maxims, some nouns, adverbs, conjunctions, etc. (*bec de gaz* (gaslamp), *à l'avenant* (in keeping), *au moment où* (at a time)). A terminal category can be associated to a set expression (*bec de gaz* (gaslamp) is a noun,…). If one can't assign a category, the internal syntax of the expression is sufficient to describe it.

- **support verb - noun**: This allows to store the couple support verb / noun as an expression (*poser/donner sa démission* (to hand in/tender one's resignation)).

- **quantifier noun - quantified noun**: This allows to code privileged pair of words such as *meute* and *loups* (pack and wolves) or *essaim* and *abeilles* (swarm and bee).

One can have acces to these compounds through several entry points, namely those of their different components.

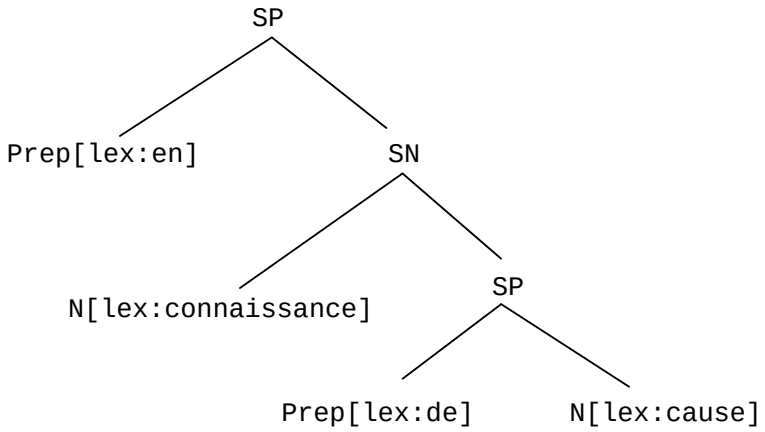
3.2.3. Constraints

Some constraints can be attached to a syntactic compound. Genelex considers lexical, morphological, morpho-syntactic, structural, semantic and solidarity constraints.

3.2.3.1. Lexical constraints

Lexical constraints on syntactic compounds are expressed by the same formalism as constraints on simple syntactic units. Such a constraint applies to all the leaves ("total lexicalisation") or to the majority of them ("partial lexicalisation").

Example: *En connaissance de cause* (with full knowledge of the facts):

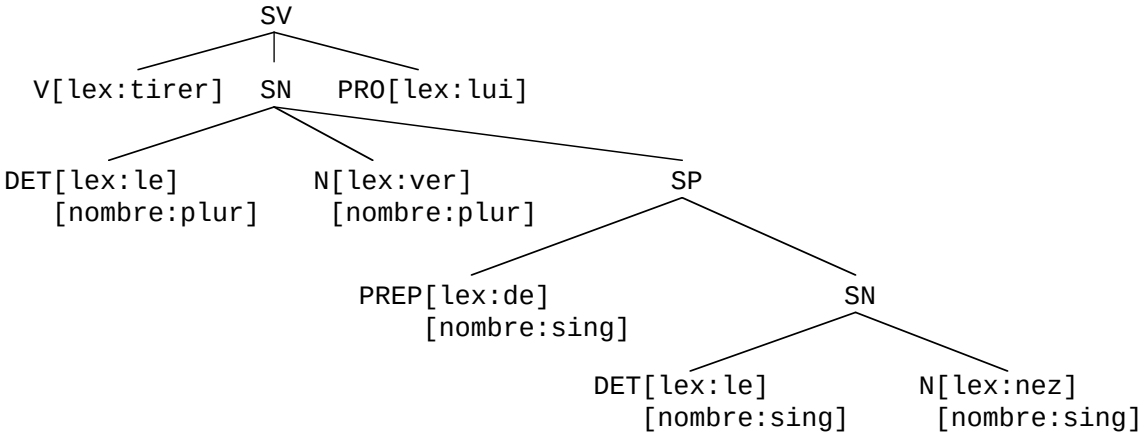


Example: *En connaissance de cause* (with full knowledge of the facts).

3.2.3.2. Morphological and morpho-syntactic constraints

A syntactic compound can have morphological constraints (mood, tense, person, gender, number, etc.) as well as morpho-syntactic constraints on the category of a component.

Example: *lui tirer les vers du nez* (to worm information out of him):



3.2.3.3. Coreference constraints

Coreference constraints are also allowed on syntactic compounds. This allows to express the coreference between components of the compounds (using the feature coref as seen for simple syntactic units) and between a component of the compound and a component of the phrase in which it appears (in that case, one has to code the structure of the phrase to refer to it).

3.2.3.4. Structural constraints

In certain cases, a syntactic compound has an optional component in the syntagm that will become obligatory.

Such constraints are expressed with simple words, using construction criteria to describe the rewriting on a syntagm.

3.2.3.5. Solidarity constraints

Some components show solidarity, i.e., they do not allow any insertion between them. The solidarity between positions is noted with a dash. E.g.: *boites aux lettres* (mailbox) will have the construction:

```
cc: HEAD - P0 - P1 - P2
HEAD: N[lex:boite]
P0: PREP[lex:à]
        [nombre:plur]
P1: DET[sscat:déf]
       [nombre:plur]
P2: N[lex:lettre]
     [nombre:plur]
```

3.2.3.6. Semantic constraints

As for simple syntactic units, semantic restrictions are available at the level of syntax (aspect, semantic class).

3.2.4. Syntactic properties

As a syntagm, the syntactic compound can be affected by syntactic operations (movement, deletion, substitution and agreement).

As for simple syntactic units, it can be necessary to freeze the place of a position. The formalism used is the same as above (“!”).

The realization of a syntagm can be unique or multiple. In case of multiple realizations, it is possible to code the different realizations.

For example, *appareil de/à projection* (projection equipment) will have the following construction:

```
cc: N: HEAD P0
HEAD: N[lex:appareil]
P0: SP: HEAD P0
        HEAD: PREP[lex:à]
              PREP[lex:de]
        P0: N[lex:projection]
```

One of the components of the syntactic compound can be deleted (not realized on the surface level). The deletion can apply to a specifier, a modifier or to the head itself. The optionality is noted with brackets (as simple SynU).

Agreement of components of a compound is expressed with the help of a coreference feature. The agreement of one of the components and an element outside the compound is handled by the grammar and not by the lexicon.

3.2.5. complementation

One has generally to specify the complementation schema of a compound syntactic unit. “self” is a reserved word that allows to refer to the described entry. It can be used to refer to a compound as a unit. In that case, the value of self is given in the internal structure of the compound.

Using this mechanism, the description of the complementation of a compound is analoguous to the description of the complementation (construction) of a simple syntactic unit.

A.3 Le Lexicaliste

I. Introduction

1. Overview

"Le lexicaliste" is a dictionary generator. It was originally developed to meet in-house needs for large translation and document processing jobs.

Le lexicaliste is generic. Lexical entries can be imported from various sources (other dictionaries, files, term lists,…) and defined on the basis of multiple attribute criteria within a very rich set of possible linguistic data structures. When lexical data is required for a given application, a dictionary can be generated and exported to that application in SGML format. Le lexicaliste runs under the Oracle RDBMS on Sun workstations with X/Windows Motif user interface.

Le lexicaliste provides a way to define the desired structures via a description of the attributes. It also allows the definition of a lexical and a semantic network. When the structure is defined, the user can write constraints and default rules with a 4th Generation Language (4GL).

Le lexicaliste also automatically provides a user-friendly interface that contains hypertext tools for navigation within the dictionary. The user can easily browse the dictionary by accessing a word or navigating in the lexical and semantic networks.

2. Goals

This system has been developed for CNET (National Center for Telecommunications) by SITE in order to build natural language understanding applications. CNET needed a large, permanently upgradable database of French words, containing easily exploitable phonetic, morpho-syntactic and semantic information. The lexical processing has to be done on a single workstation, and, given the scale of investment, the product has to be reusable for future, unspecified lexicon-based applications. The bases are not used directly by applications, but are used as sources for specialized dictionaries. Le lexicaliste is designed to manage about 200,000 entries (simple or compound words) by dictionary without notable lost of performance.

As a result, Le Lexicaliste has been developed. It is actually dedicated to monolingual dictionaries, but studies will be done to extend this system to multilingual databases.

3. Summary of the results

Le Lexicaliste is a commercialized system. A demo version is available for 1000 FF (about $200) at: SITE-EUROLANG, BP 35, 94701 Maisons Alfort Cedex, France. Tel: +33 1 45 13 05 00, Fax: +33 1 45 13 05 59.

II. Organization of the system

1. System architecture

The separation of objects illustrates the genericity of the system. This separation is done according to 3 levels:

- the data (articles, information associated to each meaning, lexical and semantic links),
- the referential (that define the set of attributes, the lexical and semantic relations used in articles),
- the meta-referential (that defines the set of characteristics of referential's objects, i.e., properties, domain of the attributes, …).

The objects of the three levels are redefinable. This allows the adaptation of the system to another linguistic theory or to another language.

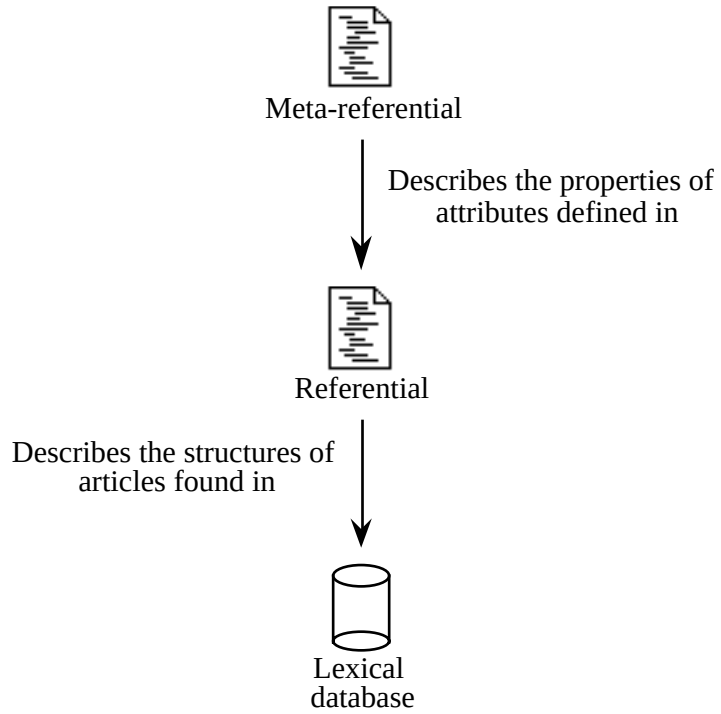


Figure 1: The three levels of definition of a dictionary.

The meta-referential gives characteristics of the objects of the referential. For example:

- properties of relation (acyclic,…),
- properties of attributes (monovaluated,…),
- cardinality of relations (1-N,…)
- …

The domain of an attribute can be defined at the level of the referential (for a priori fixed domains, e.g., flexional model, transitivity, usage,…). This domain can be simple (set of values) or hierarchical (hierarchy of values).

Inside the referential, attributes are grouped according to 5 classes:

- attributes of the lemma (e.g. `category`),
- attributes of a word-sense (e.g. `transitivity` and `definition`),
- attributes of the flexional rules (e.g. `number` and `gender`),
- lexical relations (e.g. `abbreviation`),
- semantic relations (e.g., `hyperonymy`).

The definition of an attribute is composed of different fields:

- `A_comment`: information given by the system administrator for himself,
- `U_comment`: information given by the system administrator for the user,
- `Info_type`: indicates the window containing the information,
- `Attr_type`: indicates the object the attribute is associated to,
- `Dependancy`: indicates the lemmas the attribute is associated to,
- `Properties`: indicates the properties of the attribute (meta referential),
- `Domain`: indicates the domain of the attribute,
- `Same_as`: indicates an attribute that has the same domain.

The tables of the database are automatically generated according to the referential and the meta-referential.

The interface is automatically generated according to the referential and the meta-referential. A collection of standard interaction objects (buttons, textual fields, pop-up menus,…). The choice of the appropriate object is done according to characteristics (meta-referential) of attributes (referential).

2. Linguistic architecture

2.1. Architecture of an article

An article is defined as a tree of word senses. This article is identified by:

- a lemma,
- a syntactic category,
- a type of term (simple, compound,…).

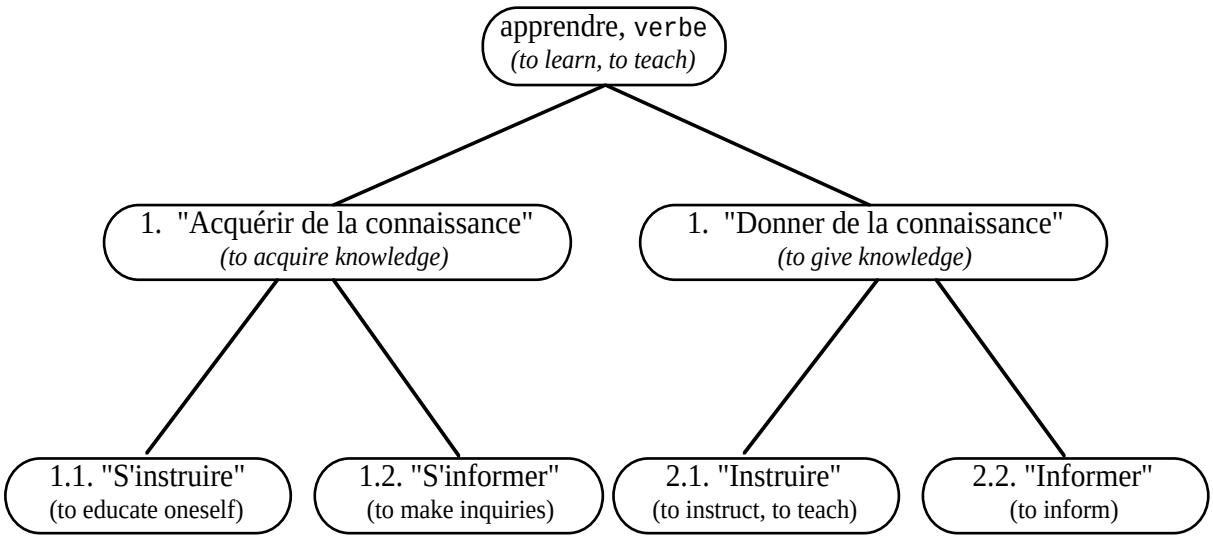


Figure 2: Example of article structure.

Each word sense is defined by information, expressed by instanciation of an attribute or a structure of attributes, or by instanciation of a relation.

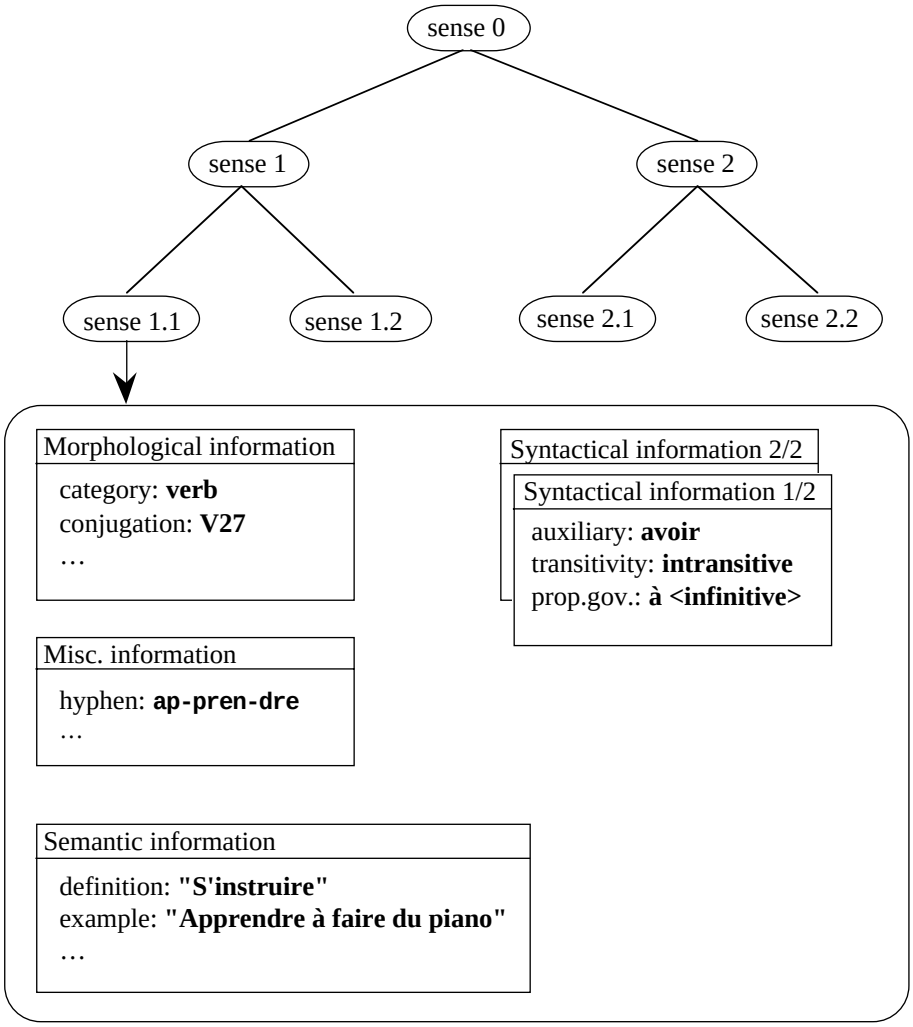


Figure 3: Information associated to a word-sense.

A classical inheritance mechanism is defined in the word-sense tree.

2.2. Lexical and semantic network

The system manages two sets of relations (i.e., 2 distinct networks):
- lexical relations (defined on a set of word-senses),
- semantic relations (defined on a set of concepts).

The lexical relations link 2 word-senses at a lexical level:
- "appt" (*apt*) `is-abreviation-of` "appartement" (*apartment*),
- "clef" (*key*) `is-orthographical-variant-of` "clé" (*key*),
- "apprentissage" (*apprenticeship*) `is-nominalisation-of` "apprendre" (*to learn*)

A lexical relation is a link at the word-sense level. This allows users to express some cases as "blanchir" (*to make sth. white*) that can be nominalized in "blanchissage" (*washing*) or in "blanchiment" (*whitewash (of a reputation*)), depending on the selected word-sense.

The semantic relations link 2 concepts at a semantic level:
- "chaise" (*chair*) `is-a` "meuble" (*furniture*),
- "poisson" (*fish*) `has-a-connotation-with` "mer" (*sea*),
- "malaria" (*malaria*) `is-synonymous-with` "paludisme" (*paludisme*).

Each word-sense can be associated with a concept by a particular attribute, the semantic predicate. Each concept can be reciprocally associated with one or more word-senses.

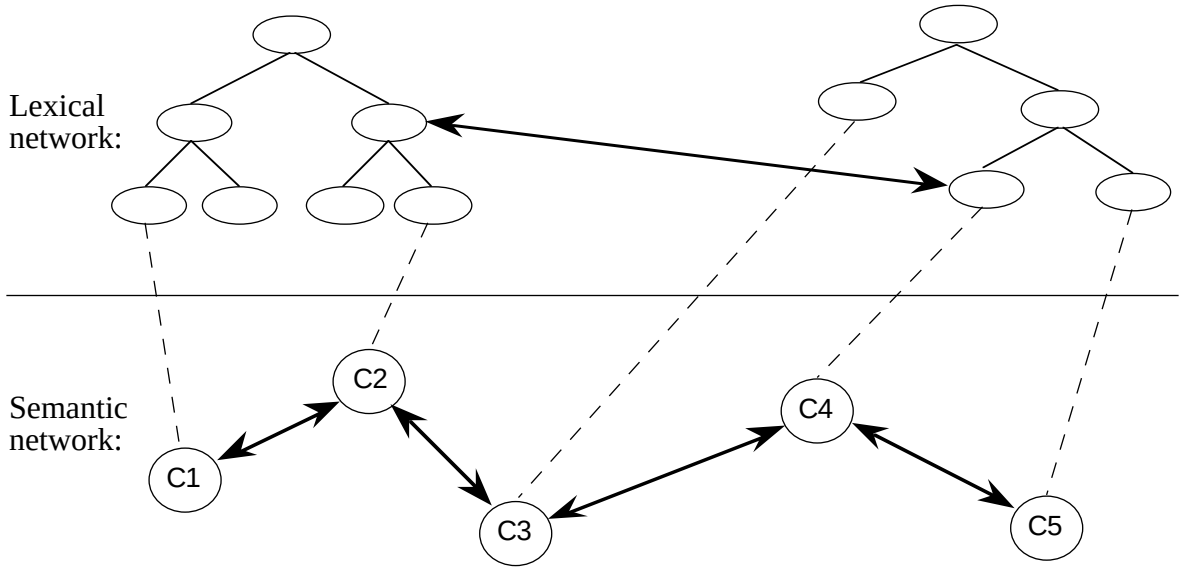


Figure 4: Lexical and semantic networks.

III. Defining coherence and default rules

In order to decrease creation and maintenance costs of a dictionary, a 4GL (4th Generation Language) has been defined. This language allows the user to define declaratively constraints on information associated to a word-sense.

There are 2 ways to use these constraints:
- checking the integrity of an article (constraints must be verified on each word-sense of an article),
- defaulting values of attributes during the creation or the updating of an article.

With the 4GL, one can express the following constraints:
- a pronominal verb obligatorily uses “être” (*to be*) as auxiliary,
- an impersonal verb does not accept passive form,
- the `transitivity` attribute certainly takes value “direct” for a verb ending with “iser” (this default value will be proposed),
- a verb ending with “ger” has V1 or V2 as value for `conjugation` attribute.

Example of constraint declaration:

```
DECL-MESSAGES

msg-aux "l'attribut auxiliaire n'est pas defini"
msg-transit "l'attribut transitif n'est pas defini"
msg-transObj2 "l'attribut transObj2 doit etre defini"
msg-frmPassif "l'attribut frmPassif doit etre defini"

DECL-CONTRAINTES

// Syntactic attributes for verbs

SI cat = verb ALORS

 // the default auxilary is “avoir” (to have)
 aux DEFINI DEFAUT {avoir} MESSAGE msg-aux

 // if the verb end with “ter”, the conjugation is
 // V3 or V3H or V3Q (default V3).
 si cle = "*ter" alors
       mm dans { V3, V3H, V3Q } defaut V3
 fsi

 // An intransitive verb does not admit passive (by default)
 SI transit = intrans ALORS
       passiv DEFAUT non
 FSI
FSI
```

Constraints that can not be expressed with the 4GL can be written directly in C.

IV. Conclusion

Le Lexicaliste is a very good solution for monolingual dictionary problems. It is generic and can be easily adapted to different linguistic theories. Of course, the system imposes some constraints on the theory (e.g., a lexical unit is composed of a tree of word-senses), but it allows real flexibility. For example, the lexicographer can freely choose whether to group 2 word-senses under the same lexical unit or to split them in 2 different lexical units (with same lemma and same category). The lexicographer can also define the structure of the articles and define relations between these articles.

When the structure of the dictionary is defined, the lexicographer can use a very user-friendly, automatically generated, interface to browse and edit articles. The cost of creation and maintenance is also decreased by automatically checking the constraints defined by the linguist with a powerful 4th Generation Language. The constraints are used by the system as coherence constraints and also as defaulting rules.

Finally, Le Lexicaliste provides mechanisms to upload/download linguistic information from/to a well known standard format, SGML, thereby conforming with the TEI guidelines. This mechanism allows the generation of linguistic application dictionaries, by extracting only the information that is relevant to the application.