OUTLINE OF A MODEL FOR LEXICAL DATABASES*

NANCY IDE [1], JACQUES LE MAITRE [2], JEAN VÉRONIS [1,2]

[1] Department of Computer Science, Vassar College
Poughkeepsie, New York 12601, U.S.A.

[2] Groupe Représentation et Traitement des Connaissances,
Centre National de la Recherche Scientifique,
31, Chemin Joseph Aiguier, 13402 Marseille Cedex 09, France

Abstract -- In this paper we show that previously applied data models are inadequate for lexical databases. In particular, we show that relational data models, including unnormalized models which allow the nesting of relations, cannot fully capture the structural properties of lexical information. We propose an alternative feature-based model which allows for a full representation of sense nesting and defines a factoring mechanism that enables the elimination of redundant information. We then demonstrate that feature structures map readily to an object-oriented data model and show how our model can be implemented in an object-oriented DBMS.

1. INTRODUCTION

For 25 years or so, computers have been used in the production of dictionaries. Initially, computers were used primarily for the first step in dictionary-making, that is, the gathering and accessing of examples of word use (see bibliography and surveys in Kipfer, 1983, and Landau, 1984). In the early 1980's, the *COBUILD* dictionary project took this use of computers to the extreme, by compiling a 20 million word corpus of English, generating extensive lists of examples from concordances of the corpus, and creating entries based almost exclusively on these examples (Sinclair, 1987). Computers have also been used extensively for word-processing and typesetting dictionary entries. However, in the last decade lexicographers and publishers began to explore more sophisticated computer involvement in the dictionary-making process, in particular, the use of databases of lexical information (including, for example, pronunciation, part of speech, definition, etymology, etc. for each word) for entry-making. The existence of a *lexical database* provides

*The work described in this paper has been carried out in the context of a joint project of the Department of Computer Science at Vassar College and the Groupe Représentation et Traitement des Connaissances of the Centre National de la Recherche Scientifique (CNRS), which is concerned with the construction and exploitation of a large lexical data base of English and French. An earlier version of this paper was presented at the conference "Intelligent Text and Image Handling" (RIAO 91) in Barcelona, Spain, April 1991.

enormous potential: it can be used to update and maintain dictionaries and check coherency within a dictionary or across related dictionaries, as well as enable the exchange and sharing of information among projects and even the automatic generation of several variant printed versions of a dictionary (e.g., a full version, a concise version, and a pocket version) from a common data source. Beyond this, such databases can be used as the basis for electronic dictionaries on CD-ROM, and for on-line consultation by scholars.

Quite independently, during the 1970's and 80's computational linguists began to develop *computational lexicons* for natural language processing programs. Computational lexicons differ from dictionaries intended for human use in that they must contain much more explicit and specific linguistic information about words and phrases (for example, typical subjects and objects for verbs, semantic features for nouns such as inanimate, human, etc.), and must be encoded in strictly formal structures usable by computer programs. However, large computational lexicons are extremely difficult to develop from scratch, and in the early 1980's computational linguists found that the information typically contained in dictionaries could be exploited in many ways if it were organized in a computer database (see, for instance, Amsler, 1980; Calzolari, 1984; Markowitz, Ahlswede & Evens, 1986; Byrd *et al.*, 1987; Nakamura & Nagao, 1988; Véronis & Ide, 1990; Klavans, Chodorow & Wacholder, 1990; Wilks *et al.*, 1990). No publishers' lexical databases existed at the time, but computational linguists saw that some relatively inexpensive processing of typesetter's tapes from printed dictionaries provided the means to create them. Consequently, lexical databases have been created from a number of printed dictionaries, including the *LDOCE*,[1] *Webster's 7th*, several Collins bilingual dictionaries, etc.

The goal of this paper is to provide a data model that is suited to lexical databases. Lexical data, as is obvious in any dictionary entry, is much more complex than the kind of data (suppliers and parts, employees' records, etc.) that has provided the impetus for most database research. Therefore, classical data models (e.g., relational) do not apply very well to lexical data, although several attempts have been made. In section 2, we review previously applied models and discuss their shortcomings, in an effort to better understand what is required to represent lexical data. A fundamental problem arises from the fact that dictionaries--and therefore databases--organize what seems to be the same kind of

[1]In this paper we will use the following abbreviations for dictionary names:

CED	*Collins English Dictionary*
LDOCE	*Longman Dictionary of Contemporary English*
NPEG	*The New Penguin English Dictionary*
OALD	*Oxford Advanced Learner's Dictionary*
OED	*Oxford English Dictionary*
SOED	*Shorter Oxford English Dictionary*

information (orthography, pronunciation, part of speech, etymology, definitions, etc.) in structurally different ways. A strong requirement for a data model is that it must make lexical information compatible despite this variability in structure. Compatibility is a necessary condition for data sharing and interchange, as well as for the development of general software.

In section 3, we show that a model based on *feature structures* overcomes most of the problems inherent in other models, and in particular enables accessing, manipulating, and merging information structured in multiple ways. The model allows retaining the particular organization of a given dictionary while at the same time making it transparent to certain database operations. There exists a well-developed theoretical framework for feature structures, which are widely used for information representation in linguistics. Their applicability to lexical databases seems therefore natural, although to our knowledge this has not yet been implemented. The use of feature structures in lexical databases also opens up the possibility of compatibility with computational lexicons, which is of considerable interest for computational linguists. A common representation scheme could create a useful bridge between lexicographers and computational linguists, and foster cross-fertilization between the two fields.

To date, feature-based models have not been exploited in commercial database management systems (DBMS), and therefore no implementation in such a system is possible. However, we show that feature structures map readily into object-oriented data models. In section 4 we describe an implementation of our model in the object-oriented DBMS O_2.

2. PREVIOUS WORK

2.1 *Text models*

Dictionaries were first realized electronically as typesetters' tapes for the purposes of publishing. As mentioned above, these tapes subsequently became available to the research community, and several have been processed extensively in order to extract lexical information. In particular, the information about typographical rendering provided in typesetter's tapes (e.g., italics, bold, etc.) is replaced by labels which provide an indication of the content of a field (e.g., headword, pronunciation, etc.), rather than its printed form. In either case, a dictionary is seen as a strictly linear text stream interspersed with *markup* or

tags, which we refer to as the *text model*. There have been some efforts to develop means for information retrieval from dictionaries in this form, since the fundamental textual nature of dictionaries does not immediately suggest the use of traditional database retrieval strategies.

2.1.1 *Typographical markup.* Typographical markup signals a font shift or the presence of special characters, etc., corresponding to the rendering of the dictionary in printed form (Fig. 1). Markup of this kind is said to be *procedural*, because it specifies the procedure (e.g., shift to italic) to be performed when a tag is encountered in linear processing, rather than providing an indication of content of the field (see Coombs, Renear & DeRose, 1987). Although typographic codes are to some extent indicative of field content (for example, part of speech may always be in italics in a given dictionary), a straightforward, one-to-one mapping between typographic codes and content clearly does not exist, since other items, such as semantic field, usage, register, geographical information etc., may be rendered in italics as well. Positional information can be coupled with typographic tags to determine content, but a complex analysis of entry format, which may or may not yield a definitive mapping due to ambiguities, is required. Information retrieval from a dictionary in this form is obviously costly, if it is possible at all.

```
*Cgin*E*S1*E (d*3T*3Fn) *Fn. *%brew *5Q*A1. *Ean alcoholic drink obtained
by distillation and rectification of the grain of malted barley, rye, or
maize, flavoured with juniper berries. *%brew *5Q*A2. *Eany of various
grain spirits flavoured with other fruit or aromatic essences: *Fsloe
gin. *%brew *5Q*A3. *Ean alcoholic drink made from any rectified spirit.
*5Q*5HC18: shortened from Dutch *Fgenever *Ejuniper, via Old French form
Latin *Fj=u-niperus*Gjuniper*E*5I<
```

Fig. 1. Typographical markup (*CED*)

2.1.2 *Descriptive markup.* In a *descriptive* markup scheme, tags provide an indication of the content of the fields they delimit rather than the printed rendering. For instance, instead of tags for italics, bold, etc., tags indicate *headword*, *part of speech*, *pronunciation*, etc. (Fig. 2). The use of descriptive markup enables the retrieval of information by content category from dictionaries in a linear text format. There have been a number of efforts to devise descriptive markup schemes for monolingual dictionaries (see, for example, Tompa, 1989) and to translate the procedural markup of typesetter's tapes into descriptive markup (see, for instance, Boguraev & Neff, in press). More recently, a preliminary common set of descriptive tags for encoding mono- and bi-lingual dictionaries has been proposed (Amsler & Tompa, 1988). This work has been incorporated into the international Text Encoding

Initiative's guidelines for encoding machine readable textual and linguistic data (Sperberg-McQueen & Burnard, 1990).

```
<ent h=gin hn=2><hdw>gin</hdw><pr><ph>dZIn</ph></pr>
<hps ps=n cu=U><hsn><def>colourless alcoholic drink distilled from grain
or malt and flavoured with juniper berries, often drunk with tonic water,
and used in many kinds of cocktail</def></hsn></hps></ent>
```

Fig. 2. Descriptive markup (*OALD*)

Sophisticated retrieval software for tagged text exists (for example, PAT; Gonnet, Baeza-Yates & Snider, 1991), which regards markup as strings of characters embedded in text and basically performs sophisticated string searches. However, texts manipulated by such software must be in a static form, and it is very costly to apply it to texts which are often modified or under development. Further, although such software provides powerful searching capabilities, it is nonetheless limited for contextual searching. The software has no knowledge of the structure of the text, and so searches which involve elements whose relationship is embodied in the structure of the dictionary entry can become prohibitively complex. For example, part of speech is usually given once at the head of an entry, although it applies to all senses. Therefore, to find the part of speech for sense 4 of a given word, information that is physically distant from that sense in the text is required, which in turn demands analysis of the surrounding text. This is accomplished by specifying several string searches and applying Boolean operators to the results. Such operations are often complicated and unintuitive. Therefore, users must typically provide the right combination of requests by hand, and queries by external software are virtually prohibited.

2.1.3 *Grammar-based models.* The problems cited above have led to the development of a more sophisticated model, which superimposes a structure on the text stream by means of a context-free grammar describing the hierarchical structure of a document (Gonnet & Tompa, 1987). The text is represented in the form of "parsed strings", that is, the string itself together with its parse tree according to a given grammar (Fig. 3). This model overcomes the contextual limitations of purely linear models, since the context is available in the parse tree. Also, Gonnet and Tompa (1987) show that it is possible to define the equivalent of many of the operators available in conventional databases with this model. However, although it has much potential interest, like linear models this model appears to be limited to fixed texts which cannot be easily modified or updated. Thus, the model may be

applicable to publically distributed read-only dictionaries, but at this time seems unable to meet the requirements for lexical databases used in publishing and research.

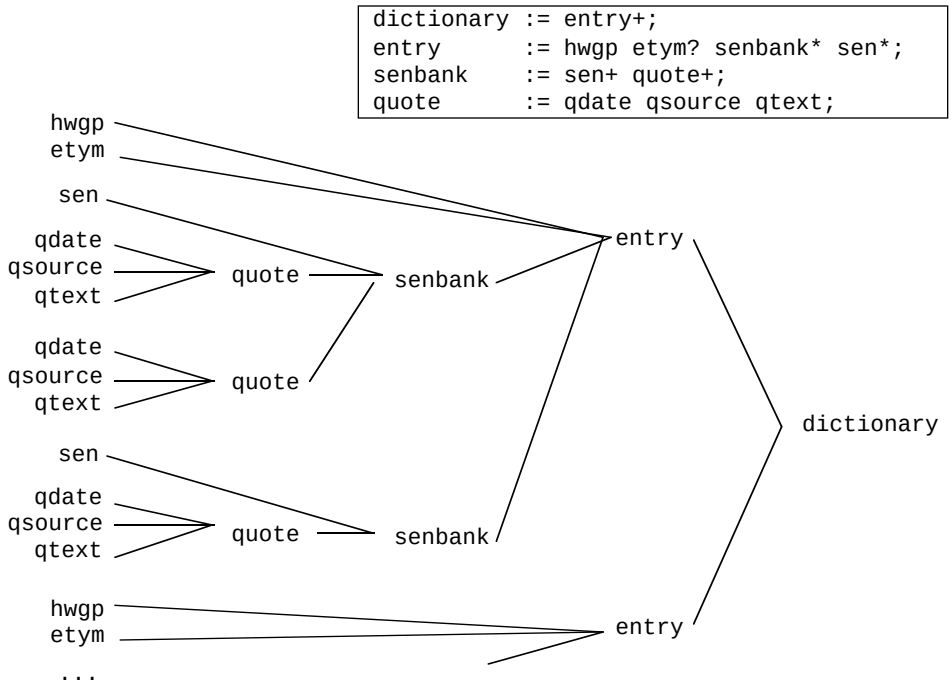


Fig. 3. Simplified grammar and a part of a "parsed-string" for the *OED* (from Gonnet & Tompa, 1987)

2.2 *Relational data models*

Conventional data models for lexical databases have been proposed, primarily for the purposes of research in computational linguistics. These data models have been less popular with publishers and lexicographers, who have traditionally mistrusted such models as too simplistic and/or rigid to allow the editorial freedom lexicographers desire when creating dictionaries (Tompa, 1989).

At the present time, the most common data model is the *relational model.* A relational database consists of a set of *relations* between *entities*. Each role in that relation is called an *attribute.* Conceptually, a relation is a *table* whose columns correspond to *attributes*, and each row, or *tuple*, specifies all the values of attributes for a given entity. Attributes have only *atomic values*--that is, values which, from the DBMS's point of view, cannot be decomposed. In other words, each row-to-column intersection contains one, and only one, value.

The relational model has been suggested for representing dictionary information (see, for example, Nakamura & Nagao, 1988). In this scheme, the dictionary is represented by a

set of relations, each of which includes attributes such as grammar codes, definitions, examples, etc. Fig. 4 gives the definition of "abandon" from the *LDOCE;* Fig. 5, expanded from Nakamura and Nagao (1988), shows the tabular representation of the same entry.[2]

a·ban·don[1] /@'bænd@n/ *v* [T1] **1** to leave completely and for ever; desert: *The sailors abandoned the burning ship*. **2** to leave (a relation or friend) in a thoughtless or cruel way: *He abandoned his wife and went away with all their money*. **3** to give up, esp. without finishing: *The search was abandoned when night came, even though the child had not been found*. **4** (*to*) to give (oneself) up completely to a feeling, desire, etc.: *He abandoned himself to grief | abandoned behaviour*. -- **~ment** *n* [U].

abandon[2] *n* [U] the state when one's feelings and actions are uncontrolled; freedom from control: *The people were so excited that they jumped and shouted with abandon / in gay abandon.*

Fig. 4. Definition of 'abandon' from *LDOCE*

DEFINITION

HW	PS	DN	DF
abandon	v	1	to leave completely and for ever
abandon	v	1	desert
abandon	v	2	to leave (a relation or friend) in a thoughtless or cruel way
abandon	v	3	to give up, esp. without finishing
abandon	v	4	to give (oneself) up completely to a feeling, desire, etc.
abandon	n	0	the state when one's feelings and actions are uncontrolled
abandon	n	0	freedom from control

EXAMPLE

HW	PS	DN	SP
abandon	v	1	The sailors abandoned the burning ship
abandon	v	2	He abandoned his wife and went away with all their money
abandon	v	3	The search was abandoned when night came, even though the child had not been found
abandon	v	4	He abandoned himself to grief
abandon	v	4	abandoned behaviour
abandon	n	0	The people were so excited that they jumped and shouted with abandon/in gay abandon

HW = headword
PS = part of speech
DN = definition number
DF = definition text
SP = example
GC = grammar code
BC = LDOCE "box" code

CODE

HW	PS	DN	GC	BC
abandon	v	1	T1	----H----T
abandon	v	2	T1	--D-H----H
abandon	v	3	T1	----H----T
abandon	v	4	T1	----H----H
abandon	n	0	U	----T-----

Fig. 5. Tables for 'abandon' in *LDOCE* database

This example is derived from a small, simple learner's dictionary with a straightforward internal structure (no deep nesting of senses, etc.), and several pieces of information from

[2]Certain information, such as the *LDOCE* semantic "box codes", appears only in the machine-readable version of the dictionary, and it therefore appears in the database even though absent from the printed version.

the entry text (for example, pronunciation, run-ons, cross-references) have been omitted from the database. However, even this simplified case shows that the relational model poses several problems for representing dictionary entries.

2.2.1 *Fragmentation of data.* The most obvious problem arises from the fact that the number of values for each attribute in dictionary entries varies enormously. For example, entries may include several different pronunciations, parts of speech, orthographic variants, definitions, etc., while some other fields, such as examples, synonyms, cross-references, domain information, geographical information, etc., may be completely absent. To avoid massive duplication of data, the information must be split across several tables, thus fragmenting the view of the data. The more complex the data, the more tables are required, and the more fragmented the view.

This fragmentation in a relational database is real--that is, the different relations represented in different tables are not explicitly connected, but are only logically connected by including attributes with the same domains in the different tables (for example, `HW`, `PS`, and `DN` in Fig. 5). Tuples are connected only if the values for those attributes match. As a result, queries can become complex and unintuitive. For example, Fig. 6 shows the SQL query required to extract all examples given for uncountable nouns whose definitions start with the string "the state...", as for the noun sense of *abandon.*

```
SELECT DEFINITION.HW, EXAMPLE.SP
FROM DEFINITION, EXAMPLE, CODE
WHERE
  DEFINITION.PS = "n" AND
  DEFINITION.DF = "the state *" AND
  EXAMPLE.SP <> null AND
  CODE.GC = "U" AND
  DEFINITION.HW = EXAMPLE.HW AND
  EXAMPLE.HW = CODE.HW AND
  DEFINITION.PS = EXAMPLE.PS AND
  EXAMPLE.PS = CODE.PS AND
  DEFINITION.DN = EXAMPLE.DN AND
  EXAMPLE.DN = CODE.DN
```

Fig. 6. Sample SQL query

A less fragmented view can be obtained by *joining* two or more tables, but the resulting table typically contains an enormous amount of redundant information. Fig. 7 shows the result of joining the tables from Fig. 5, and although very little of the information in an actual dictionary entry is represented here, it is already obvious that the resulting view is cumbersome. Of course, joining makes query construction easier, but the burden is then on

the user to appropriately define the most commonly used views, and on the management system to handle the processing.

HW	PS	DN	GC	BC	DF	SP
abandon	v	1	T1	----H----T	to leave completely and for ever	The sailors abandoned the burning ship
abandon	v	1	T1	----H----T	desert	The sailors abandoned the burning ship
abandon	v	2	T1	--D-H----H	to leave (a relation or friend) in a thought-less or cruel way	He abandoned his wife and went away with all their money
abandon	v	3	T1	----H----T	to give up, esp. without finishing	The search was abandoned when night came, even though the child had not been found
abandon	v	4	T1	----H----T	to give (oneself) up completely to a feeling, desire, etc.	He abandoned himself to grief
abandon	v	4	T1	----H----T	to give (oneself) up completely to a feeling, desire, etc.	abandoned behaviour
abandon	n	0	U	----T-----	the state when one's feelings and actions are uncontrolled	The people were so excited that they jumped and shouted with abandon/in gay abandon
abandon	n	0	U	----T-----	freedom from control	The people were so excited that they jumped and shouted with abandon/in gay abandon

Fig. 7. Joined table from `DEFINITION, EXAMPLE` and `CODE` in Fig. 5.

2.2.2 *Hierarchical structure.* Whether a view is fragmented or joined, there is no representation in a relational database of the obvious hierarchy within most dictionary entries--for instance, it is clear that the entry for *abandon* has two main sub-parts, one for its verb senses and one for its noun sense, and that the two senses of the verb labeled "1" in Fig. 5 are in fact two sub-senses of the first sense given in the entry. These two sub-senses are more closely related to each other than to senses 2, 3, and 4, but the tabular format obscures this fact. Some dictionaries take the grouping and nesting of senses several levels deep in order to distinguish finer and finer grains of meaning. The Hachette *Zyzomys* CD-ROM dictionary, for instance, distinguishes up to five levels in an entry. Fig. 8 shows that in this dictionary "valeur" has two fundamental senses: (A) *value* as *merit* ; and (B) *value* as price. Going deeper, we see that sense A subdivides into two main subcategories: (I) merit of an individual; and (II) the subjective worth of an object. Sense A.I subdivides further into two more subcategories: (1) merit of a person based on general qualities; and (2) bravery or valor, which in turn forms a part of the compound "croix de la valeur militaire", a French military decoration. Flattening this structure into a tabular form obscures the derivational relations captured in the nested arrangement.

valeur [valœR] n. f. **A. I. 1.** par quoi une personne est digne d'estime, ensemble des qualités qui la recommandent. (V. mérite). *Avoir conscience de sa valeur. C'est un homme de grande valeur*. **2.** Vx. Vaillance, bravoure (spécial., au combat). *"La valeur n'attend pas le nombre des années"* (Corneille). ◊ *Valeur militaire (croix de la):* décoration française...
...
II. 1. Ce en quoi une chose est digne d'intérêt. *Les souvenirs attachés à cet objet font pour moi sa valeur*. **2.** Caractère de ce qui est reconnu digne d'intérêt...
...
B. I. 1. Caractère mesurable d'un objet, en tant qu'il est susceptible d'être échangé, désiré, vendu, etc. (V. prix). *Faire estimer la valeur d'un objet d'art...*

Fig. 8. Part of the definition of 'valeur' in Hachette *Zyzomys*

2.3 *Unnormalized relational models*

The need to eliminate redundancy by factoring common pieces of information is well known in database research and has led to the development of *unnormalized* (also *Non First Normal Form* or *NF*^{2}) relational data models, in which attribute values may have a composite structure. That is, attribute values may be either atomic, as in the classical *normalized* relational model, or they may be nested relations with their own internal structure. An algebra and calculus have been proposed for these relations (Abiteboul & Bidoit, 1984; Roth, Korth & Silberschatz, 1988), and a few DBMSs have been developed using the NF^2 model (e.g., VERSO [Bancilhon, 1983] , AIM-P [Pistor & Traunmueller, 1986], DASDBS [Schek, Paul, Scholl & Weikum, 1990]).

Fig. 9 shows how the entry for *abandon* given in Fig. 4 would be represented in the NF^2 model. The outermost table consists of a relation between a headword and some number of homographs. In turn, a homograph consists of a part of speech, a grammar code, and some number of senses; etc. Obviously, this model better captures the hierarchical structure of information in the dictionary and enables the factoring of attributes. Relations which were represented only in matched attribute values between tables in the normalized model are now made explicit, and therefore, complex queries do not require specification of the table connections.

HW	HOMOGRAPH					
	PS	GC	SENSE			
			DN	BC	DEFINITION	EXAMPLE
					DF	SP
abandon	v	T1	1	----H----T	to leave completely and for ever	The sailors abandoned the burning ship
					desert	
			2	--D-H----H	to leave (a relation or friend) in a thoughtless or cruel way	He abandoned his wife and went away with all their money
			3	----H----T	to give up, esp. without finishing	The search was abandoned when night came, even though the child had not been found
			4	----H----T	to give (oneself) up completely to a feeling, desire, etc.	He abandoned himself to grief
						abandoned behaviour
	n	U	0	----T-----	the state when one's feelings and actions are uncontrolled	The people were so excited that they jumped and shouted with abandon/in gay abandon
					freedom from control	

Fig. 9. NF^2 representation of the entry 'abandon'

Neff, Byrd, and Rizk (1988) describe an organization for a lexical database (the IBM LDB) (see also Calzolari, Peters & Roventini, 1990) also based on a hierarchy of attributes, which allows the represention of information in a dictionary entry as a tree (see Fig. 10). Queries are made by filling templates with the same tree structure as the dictionary entry and by indicating desired values and conditions for various attributes (Fig. 11). Therefore, the IBM LDB model is fundamentally an NF^2 model, although it does not use an NF^2 DBMS but is instead built around an *ad hoc* implementation.

Although NF^2 models clearly improve on other models for representing dictionary information, a number of problems still remain. These are outlined in the following subsections.

```
entry
+-hdw: abandon
|
+-superhom
| +-word: abandon
| +-print_form: a.ban.don
| +-hom_number: 01
| |
| +-pronunciation
| | +-primary
| |   +-pron_string: E"b&ndEn
| |
| +-syncat: v
| +-g_code_field: T1
| |
| +-sense_def
| | +-sense_no: 1
| | +-subj_code: ....
| | +-box_code: ....H....T
| | |
| | +-defn
| | | +-def_string: to leave completely and for ever; desert
| | |
| | +-example
| |   +-ex_string: The sailors abandoned the burning ship
| |
| +-sense_def
| | +-sense_no: 2
...
...
| +-run_on
|   +-sense_link: 01
|   +-derivative: abandonment
|   +-d_syncat: n
|   |
|   +-d_code
|     +-g_code_field: U
|
+-superhom
  +-word: abandon
  +-print_form: abandon
  +-hom_number: 02
  +-syncat: n
  +-g_code_field: U
  |
  +-sense_def
    +-sense_no: 0
    +-subj_code: ....
    +-box_code: ....T.....
    |
    +-defn
    | +-def_string: the state when one's feelings and actions...
    |
    +-example
      +-ex_string: The people were so excited that they jumped...
```

Fig. 10. IBM LDB format for 'abandon' in the *LDOCE*

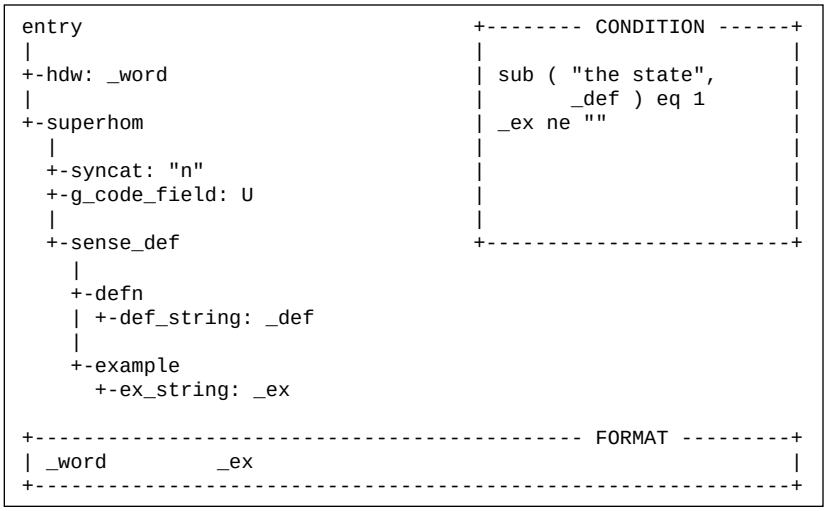


Fig. 11. IBM LDB -- sample query

2.3.1 *Recursive nesting.* Fig. 12 shows an attempt to render the entry for *valeur* from the *Zyzomys* dictionary given in Fig. 8 in the IBM LDB format. It is clear that the IBM LDB model and NF^2 models in general can represent the deep hierarchical structure of the entry. However, NF^2 models do not allow the recursive nesting of relations, and Neff, Byrd, and Rizk (1988) explicitly prohibit recursion in the IBM LDB model. This necessitates the proliferation of attributes such as `sense_def_level_1`, `sense_def_level_2`, etc. to represent the different levels of sense nesting. This in turn demands that queries take into account all the possible positions where a given sub-attribute (e.g., *usage*) could appear. For example, all the queries in Fig. 13 are required to retrieve all nouns which have an archaic *(Vx = vieux)* sense. Since any sense at any level could have this attribute value, it is necessary to query each level.

```
entry
+-hdw: valeur
|
+-superhom
| +-word: valeur
...
| |
| +-sense_def_level_1:
| | +-sense_no: A
| | +-sense_def_level_2:
| | | +-sense_no: I
| | | +-sense_def_level_3:
| | | | +-sense_no: 1
| | | | +-defn
| | | | | +-def_string: Ce par quoi une personne est digne d'estime...
| | | | |
| | | | +-example
| | | | |     +-ex_string: Avoir conscience de sa valeur.
| | | | |     +-ex_string: C'est un homme de grande valeur.
| | | | |
| | | +-sense_def_level_3:
| | | | +-sense_no: 2
| | | | +-usage: Vx
| | | | +-defn
| | | | | +-def_string: Vaillance, bravoure (spécial., au combat)
| | | | +-example
| | | | |   +-ex_string: "La valeur n'attend pas le nombre des années"
| | | | |   +-ex_author: Corneille
| | | | |
| | | | +-sense_def_level_4:
| | | |   +-sense_no: a
| | | |   +-compound: Valeur militaire (croix de la)
| | | |   +-defn
| | | |     +-def_string: décoration française...
...
```

Fig. 12. Attempt to render the entry *valeur* in the IBM LDB format

```
entry
|
+-hdw: _word
|
+-superhom
  |
  +-syncat: "n"
  |
  +-sense_def_level_1:
    +-usage: "Vx"
```

```
entry
|
+-hdw: _word
|
+-superhom
  |
  +-syncat: "n"
  |
  +-sense_def_level_1:
    +-sense_def_level_2:
      +-usage: "Vx"
```

```
entry
|
+-hdw: _word
|
+-superhom
  |
  +-syncat: n
  |
  +-sense_def_level_1:
    +-sense_def_level_2:
      +-sense_def_level_3:
        +-usage: Vx
```

etc.

Fig. 13. Query problem

2.3.2 *Exceptions.* Exceptional cases are characteristic of lexical data. For instance (see Fig. 14):

- Sense 3 of the word "conjure" in the *OALD*, has a different pronunciation from the other senses in the entry.

- In the same entry, the related entry "conjurer, conjuror", although given at the entry level, applies only to sense 1.[3]
- The entry "heave" in both the *OALD* and *CED* shows that inflected forms may apply to individual senses--in this case, the past tense and past participle is "heaved" for all but the nautical senses, for which it is "hove".[4]
- The entry for "breath" from the *SOED* and the entry for "silk" from *NPEG* each specify a special etymology for a particular sense within their respective entries.

Allowing the same attribute at different levels, in different nested relations (for example, allowing a pronunciation attribute at *both* the homograph and sense levels) would require a mechanism to "override" an attribute value at an inner level of nesting. NF^2 models do not provide any such mechanism and, in fact, do not allow the same attribute to appear at different levels. The only way exceptions could be handled in an NF^2 model would be by re-defining the template so that attributes such as pronunciation, inflected forms, etymology, etc., are associated with senses *rather* than homographs. However, this would disable the factoring of this information, which applies to the entire entry in the vast majority of cases. The result would be an effective return to the normalized model.

con•jure /'k^ndG@(r)/ *vt,vi* **1** [VP2A,15A] do clever tricks which appear magical... **2** [VP15B] **~ *up*,** cause to appear as if from nothing... **3** /k@n'dGW@(r)/ [VP17] (formal) appeal solemnly to... **con•jurer**, **con•juror** /'k^ndG@r@(r)/ *n* person who performs conjuring tricks } 1 above. [OALD]

heave /hi:v/ *vt,vi* (*pt,pp* ~d or (6 and 7 below), nautical use, hove /h@Wv/) ... [OALD]

heave (hi:v) *vb.* **heaves, heaving, heaved** *or* (*chiefly nautical*) **hove.** ... **5.** (*past tense and past participle* **hove**) *Nautical.* **a.** to move or cause to move in a specified way ... **b.** (*intr.*) (of a vessel) to pitch or roll. ... [CED]

Breath (breP). [OE. *brœP* odour, exhalation ... The sense 'air in the lungs or mouth' was taken over from OE. *œPm* and *anda* (ME *ethem* and *ande, onde*).] ... [SOED]

silk /silk/ *n* **1** a fine continous protein fibre ... **3** a King's or Queen's counsel ... [ME, fr OE seolc ... (3) fr the silk gown worn by a King's or Queen's counsel] [NPED]

Fig. 14. Exceptions in dictionary entries

2.3.3 *Variable factoring.* Dictionaries obviously differ considerably in their physical layout. For example, in one dictionary, all senses of a given orthographic form with the

[3]Note that the dictionary has to rely on a special mechanism ("} 1 above") to make this specification.

[4]Again, a special rendering mechanism is required to handle this case, since it so grossly violates the usual entry format.

same etymology will be grouped in a single entry, regardless of part of speech; whereas in another, different entries for the same orthographic form are given if the part of speech is different. The *CED*, for instance, has only one entry for *abandon,* including both the noun and verb forms, but the *LDOCE* gives two entries for *abandon,* one for each part of speech. As a result of these differences, the IBM LDB template for the *LDOCE* places the part of speech (`syncat`) attribute at the homograph level, whereas in the *CED* template, part of speech must be given at the level of sense (or "sense group" if some new attribute were defined to group senses with the same part of speech within an entry). This means that the query for part of speech in the *LDOCE* is completely different from that for the *CED*. Further, it means that the merging or comparison of information from different dictionaries demands complete (and possibly complex) de-structuring and re-structuring of the data. This makes data sharing and interchange, as well as the development of general software for the manipulation of lexical data, difficult.

However, differences in dictionary layout are mainly differences in structural organization, whereas the fundamental elements of lexical information seem to be constant. In the example above, for instance, the basic information (orthography, pronuncation, part of speech, etc.) is the same in both the *CED* and *LDOCE,* even if its organization is different. Recognizing this, there have been various efforts to develop a taxomomy of lexical data and a meta-lexical terminology that can generalize across dictionaries and projects (Brustkern & Hess, 1982; The DANLEX Group, 1987).

The only way to have directly compatible databases for different dictionaries in the NF^2 model, even if one assumes that attributes for the same kind of information (e.g., orthography) can have the same *name* across databases, is to have a common *template* across all of them. However, the fixed factoring of attributes in NF^2 models prohibits the creation of a common template, because the template for a given database mirrors the particular factoring of a single dictionary. Therefore, a more flexible model is needed that would retain the particular factoring of a given dictionary, and at the same time render that factoring transparent to certain database operations.

3. A FEATURE-BASED MODEL

In this section we introduce a model for representing information in dictionary entries based on *feature structures*. Feature structures have been heavily used in formal and computational linguistics and natural language processing to encode linguistic information, especially in various grammar formalisms (see, for instance, Kaplan & Bresnan, 1982; Kay,

1985). Their applicability to the information found in dictionaries seems natural and opens up the possibility of compatibility with computational lexicons, although to our knowledge feature structures have not yet been used to represent dictionaries. In addition, there exists a well-developed theoretical framework for the feature structure mechanism which can provide a basis for the model we develop here.

3.1 *Feature structures*

In this section, we give a very brief overview of feature structures. For a more detailed introduction we refer the reader to Shieber (1986). A feature structure is composed of pairs of attributes (called *features*) and their values, which can also be seen as *partial functions* from features to values. Feature structures are graphically represented as a list of features separated from their values by colons, enclosed in square brackets (see Fig. 15a). Values may be atomic or may themselves be feature structures (Fig. 15b and c).

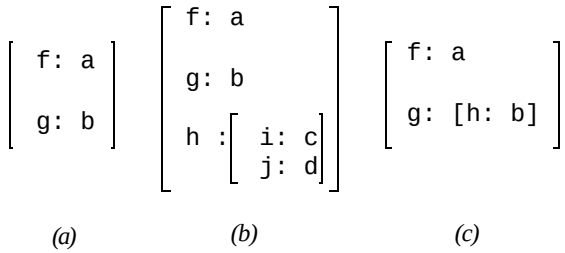


Fig. 15. Feature structure notation

A partial order relation called *subsumption* is defined for feature structures, which determines whether one feature structure is more general than another. A feature structure A is said to *subsume* another feature structure B (noted A $\sqsubseteq$ B) if for each feature *f* of A, there is a feature *f* in B and if the two values A(*f*) and B(*f*) are atomic then A(*f*) = B(*f*), otherwise A(*f*) $\sqsubseteq$ B(*f*). Thus the feature structure (a) subsumes the feature structure (b) in Fig. 15.

An operation called *unification* is also defined to enable the combination of information from feature structures having different, but compatible, information. In formal terms, the unification of A and B (noted A $\sqcup$ the greatest lower bound of A and B according to the subsumption relation--that is, the most general feature structure that is subsumed by both A and B. The dual operation, which consists of taking the least upper bound--that is, the most precise feature structure that subsumes both A and B--is called *generalization* (noted A $\sqcap$ B) (Fig. 16).

Two feature structures are said to be *compatible* if they can be unified, that is, if there exists a feature structure subsumed by both A and B. Otherwise, they are said to be *incompatible.* The feature structure (c) in Fig. 15 is incompatible with both (a) and (b).

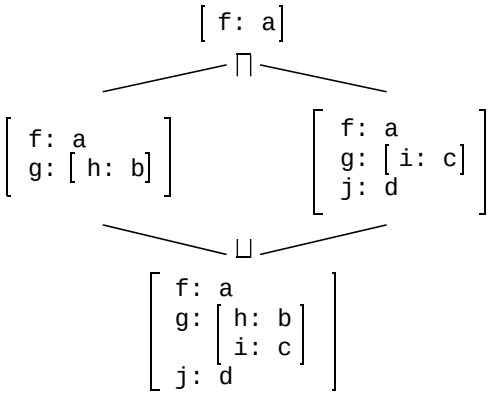


Fig. 16. Unification and generalization

A model based on feature structures can be used to represent simple dictionary entries, as shown in Fig. 17. Our model is *typed* in the sense that not all features can appear anywhere, but instead, must follow a schema that specifies which features are *allowable* (although not necessarily present), and where (see, for instance, Pollard & Sag, 1987). The schema also specifies the domain of values, atomic or complex, allowed for each of these features. For example, entries are described by the type ENTRY, in which the features allowed are *form, gram, usage, def,* etc. The domain of values for *form* is feature structures of type FORM, which consists of feature structures whose legal features include *orth, hyph,* and *pron.* Each of these features has, in turn, an atomic value of type STRING, etc.

com•peti•tor /k@m'petɪt@(r)/ *n* person who competes [OALD]

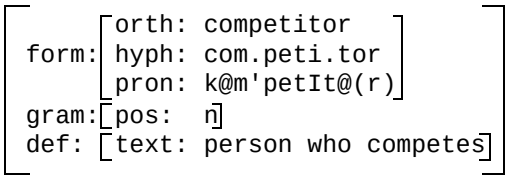


Fig. 17. Representation of a simple sense

3.2 *Value disjunction and variants*

The basic feature-based formalism is not enough to represent the structure of more complex dictionary entries. Fortunately, several authors have proposed extensions that solve many of the remaining problems. Karttunen (1984) shows that *value disjunction* is a natural, linguistically motivated extension, which enables the specification a *set* of alternative values, atomic or complex, for a given feature. The use of value disjunction enables the represention of variants, common in dictionary entries, as shown in Fig. 18. We have added a further extension which enables the specification of either a *set* (noted $\{x_1, \ldots x_n\}$) or a *list* (noted $(x_1, \ldots x_n)$) of possible values. This extension enables retaining the order of values, which is in many cases important in dictionaries. For example, the orthographic form given first is most likely the most common or preferred form. Other information, such as grammatical codes, may not be ordered[5].

biryani *or* **biriani** (ˌbɪrɪ'A:nɪ) *n*. Any of a variety of Indian dishes... [CED]

```
form: [orth: (biryani, biriani)]
      [pron: ,biri'A:ni        ]
gram: [pos:  n                 ]
def:  [text: Any of a variety  ]
      [      of Indian dishes...]
```

Fig. 18. Value disjunction

In many cases, sets or lists of alternatives are not single values but instead *groups* of features. This is common in dictionaries; for instance, Fig. 19 shows a typical example where the alternatives are *groups* consisting of orthography and pronunciation.

[5]Since all of our examples are drawn from existing dictionaries, we have chosen to retain the original ordering rather than make decisions concerning the relevance of the order in which items appear. Therefore, mainly lists appear in the examples given here.

mackle ('mæk@l) *or* **macule** ('mækju:l) *n. Printing.* a double or blurred impression caused by shifting paper or type. [CED]

```
form:  ( orth: mackle  )
       ( pron: 'm&k@l  )
       ( orth: macule  )
       ( pron: 'm&kju:l)
gram:  [ pos:  n       ]
usage: [ dom:  Printing ]
def:   [ text: a double or blurred impression... ]
```

Fig. 19. Value disjunction of non-atomic values

3.3 *General disjunction and factoring*

Kay (1985) proposes an additional extension, called *general disjunction,* that provides a means to specify alternative *sub-parts* of a feature structure. Again, we have extended the mechanism to enable the specification of both sets and lists of sub-parts. Therefore, feature structures can be described as being of the form $[\phi_1, \ldots \phi_n]$, where each ϕ_i is a feature-value pair *f:* ψ, a set of feature structures $\{\psi_1, \ldots \psi_p\}$, or a list of feature structures $(\psi_1, \ldots \psi_p)$. Unification can be extended to disjunctive feature structures: if F= $\{\phi_1, \ldots \phi_n\}$ and G = $\{\psi_1, \ldots \psi_p\}$, computing F $\sqcup$ s taking the disjunction of all the $\phi_i \sqcup \psi_j$, and then discarding all the nonmaximal disjuncts.

General disjunction allows common parts of components to be factored. Fig. 20a shows that without any disjunction, two different representations for the entry for *hospitaller* from the *CED* are required. The use of value disjunction enables localizing the problem and thus eliminates some of the redundancy (Fig. 20b), but only general disjunction (Fig. 20c) captures the obvious factoring and represents the entry cleanly and without redundancy.

hospitaller *or U.S.* **hospitaler** ('hɑspIt@l@) *n.* a person, esp. a member of certain religious orders... [CED]

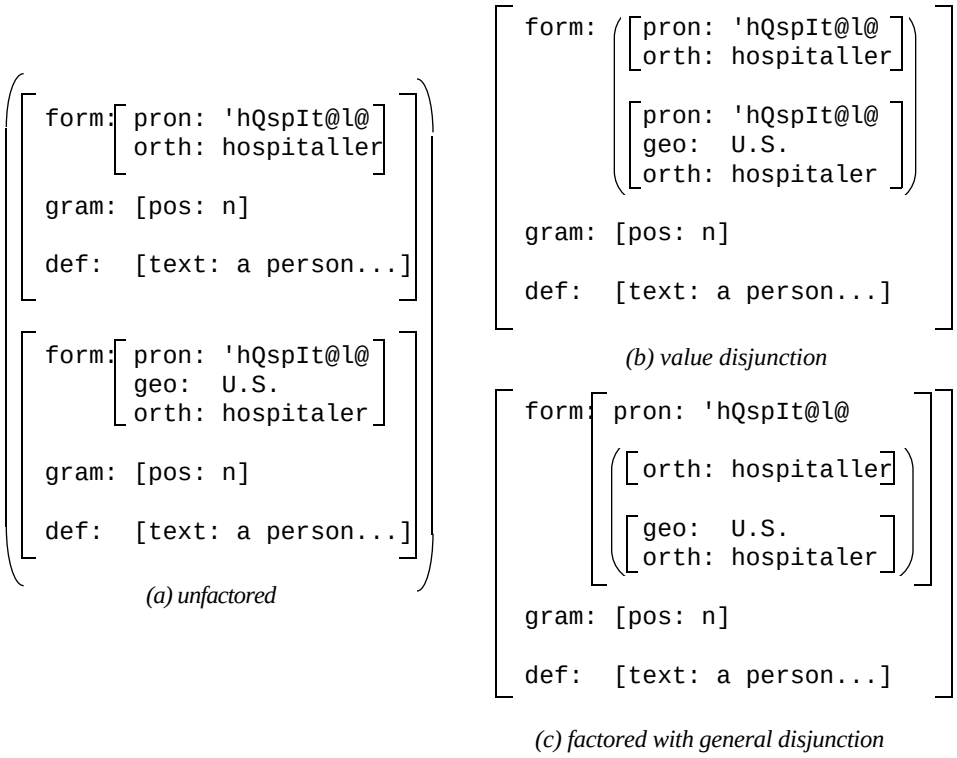


Fig. 20. General disjunction

General disjunction provides a means to represent multiple senses, since they can be seen as alternatives (Fig. 21).[6]

disproof (dɪs'pru:f) *n.* **1.** facts that disprove something. **2.** the act of disproving. [CED]

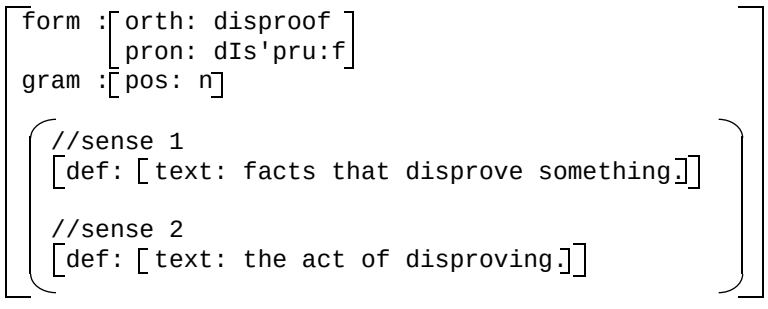


Fig. 21. Representation of multiple senses

[6]Note that in our examples, "//" signals the beginning of a comment which is not part of the feature structure. We have not included the sense number as a feature in our examples because sense numbers can be automatically generated.

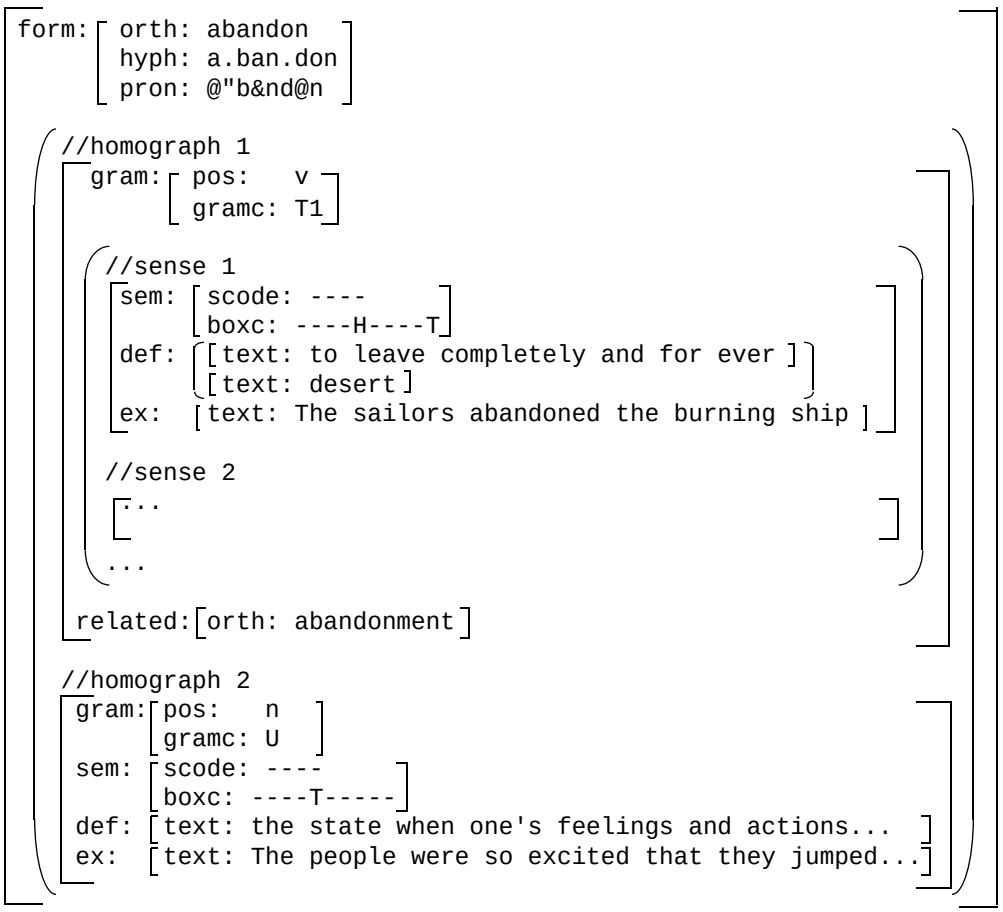


Fig. 22. Representation of the entry *abandon* in *LDOCE*

Sense nesting is also easily represented using this mechanism. Fig. 22 shows the representation for *abandon* given previously. At the outermost level of the feature structure, there is a disjunction between the two different parts of speech (which appear in two separate entries in the *LDOCE*). The disjunction enables the factoring of orthography, pronunciation, and hyphenation over both homographs.[7] Within the first component of the disjunction, the different senses for the verb comprise an embedded list of disjuncts. Note that in this model there is no different *type* feature structure for entries, homographs, or senses, since they potentially contain the same kind of information, as the discussion in

[7]Interestingly, the *LDOCE* gives a separate entry for each part of speech, but gives information about hyphenation and pronunciation only in the entry for the first homograph. This shows that entries are an artifact of printed presentation and do not entirely reflect logical structure. The IBM LDB representation of the *LDOCE* entry for *abandon* loses the information about hyphenation and pronunciation for the second homograph, since there is no provision for the factoring of information in this scheme. The only solution in that model would be to repeat the information for the second homograph.

section 2.3 demonstrates. This reflects a fundamental property of lexical data which is obscured by the layout of print dictionaries.

The entry for *valeur* from the *Zyzomys* dictionary provides an even more complex example of sense nesting (Fig. 23).

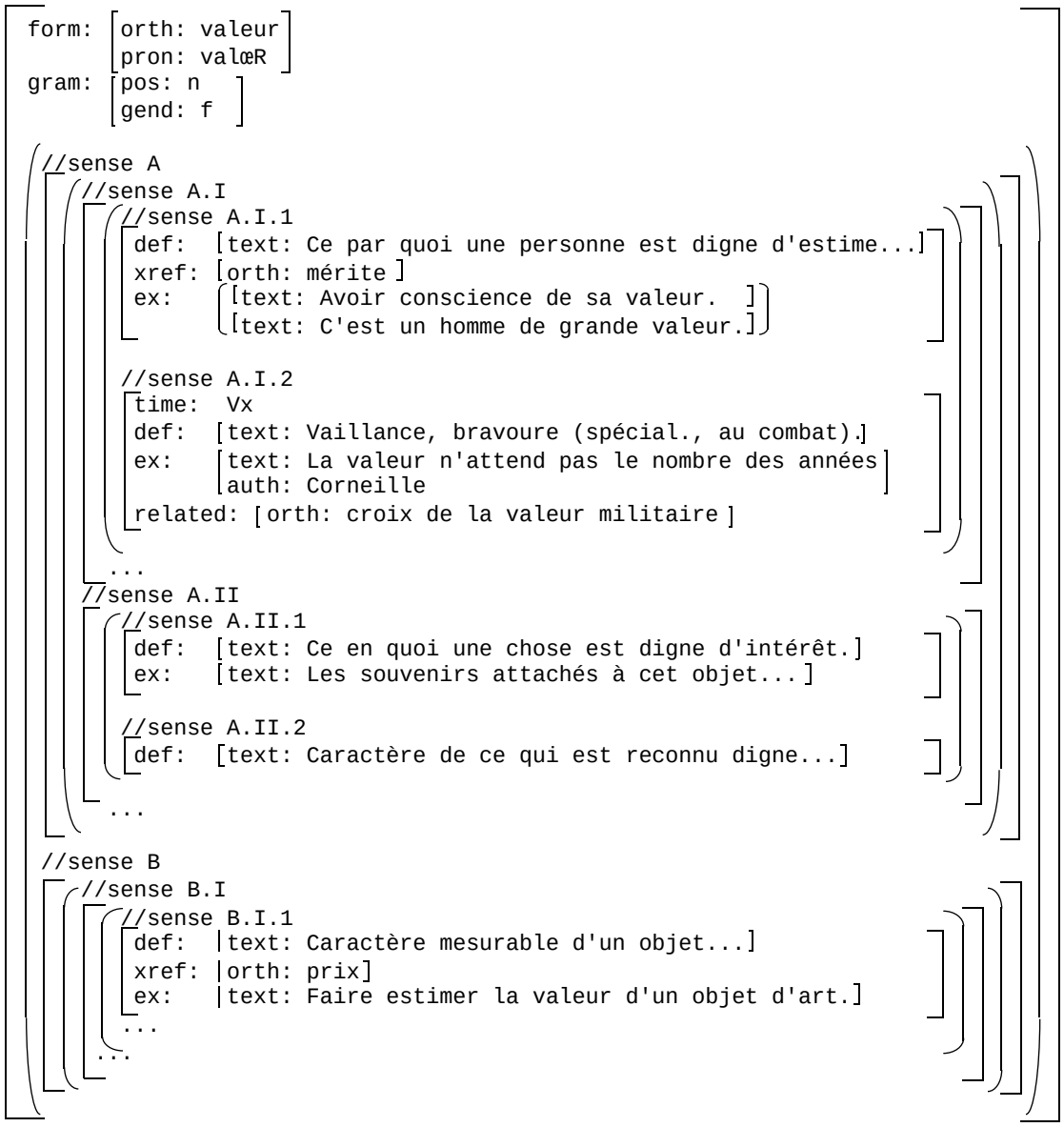


Fig. 23. Representation of the entry *valeur* in *Zyzomys*

Note that we restrict the form of feature structures in our model to a *hierarchical normal form*. That is, in any feature structure F = [$\phi_1, \ldots \phi_n$], only one ϕ_i, let us say ϕ_n = {$\psi_1, \ldots \psi_p$}, is a disjunction. This restriction is applied recursively to embedded feature structures. This scheme enables representing a feature structure as a tree in which factored information [$\phi_1, \ldots \phi_{n-1}$] at a given level is associated with a node, and branches from that

node correspond to the disjuncts $\psi_1, \ldots \psi_p$. Information associated with a node applies to the whole sub-tree rooted at that node. For example, the tree in Fig. 24 represents the feature structure for *abandon* given in Fig. 22. The representation of information as a tree mirrors the hierarchical structure and factoring of information in dictionaries.

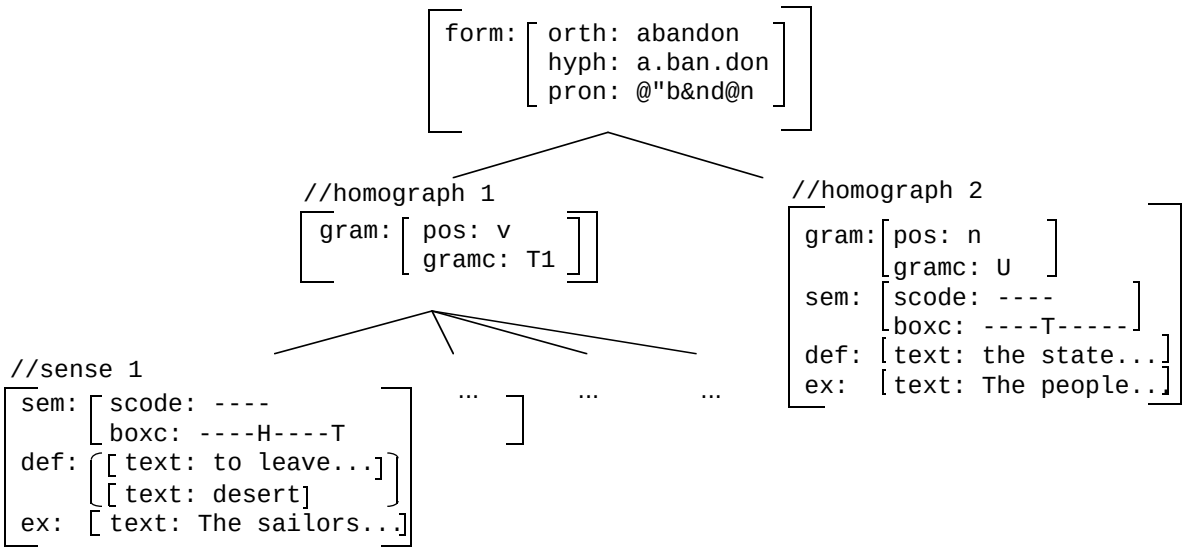


Fig. 24. Hierarchical Normal Form

3.4 *Disjunctive normal form and equivalence*

It is possible to define an *unfactor* operator to multiply out the terms of alternatives in a general disjunction (Fig. 25), assuming that no feature appears at both a higher level and inside a disjunct.[8]

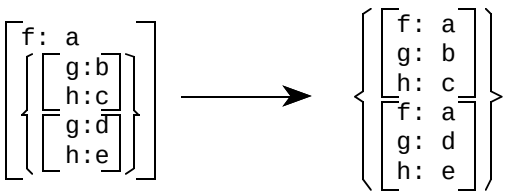


Fig. 25. Unfactoring

By applying the unfactor operator recursively, it is possible to eliminate all disjunctions except at the top level. The resulting (extremely redundant) structure is called the *disjunctive*

[8]Value disjunction is not affected by the unfactor process. However, a value disjunction [f: {a, b}] can be converted to a general disjunction [{[f: a], [f: b]}], and subsequently unfactored.

normal form (DNF). We say that two feature structures are *DNF-equivalent* if they have the same DNF. The fact that the same DNF may have two or more equivalent factorings enables the representation of different factorings in dictionaries, while retaining a means to recognize their equivalence. Fig. 26a shows the factoring for inflected forms of *alumnus* in the *CED;* the same information could have been factored as it appears in Fig. 26b. Note that we have used *sets* and not *lists* in Fig. 26. Strictly speaking, the corresponding feature structures with lists would not have the same DNFs. However, since it is trivial to convert lists into sets, it is easy to define a stronger version of DNF-equivalence that disregards order.

alumnus (@'l^mn@s) *or* (*fem.*) **alumna** (@'l^mn@) *n., pl.* **-ni** (-naɪ) *or* **-nae** (-ni:) … [CED]

(a)

alumnus (@'l^mn@s), *pl.* **-ni** (-naɪ), *or* (*fem.*) **alumna** (@'l^mn@)., *pl.* **-nae** (-ni:)

(b)

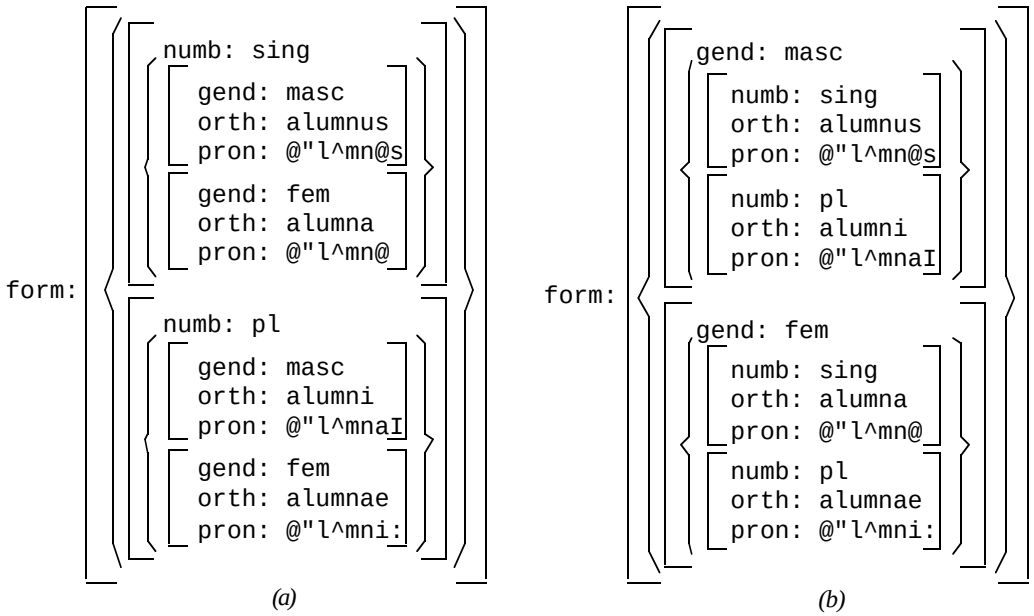


Fig. 26. Two different factorings of the same information

We can also define a *factor* operator to apply to a group of disjuncts, in order to factor out common information. Information can be unfactored and re-factored in a different format without loss of information, thus enabling various presentations of the same information, which may, in turn, correspond to different printed renderings or "views" of the data.

3.5 *Partial factoring*

The type of factoring described above does not handle the example in Fig. 27, where only a part of the grammatical information is factored (pos and subc, but not gcode). We can allow a given feature to appear at both the factored level and inside the disjunct, as long as the two values for that feature are *compatible*. In that case, unfactoring involves taking the *unification* of the factored information and the information in the disjunct (Fig. 28).

ca•reen /k@'ri:n/ *vt,vi* **1** [VP6A] turn (a ship) on one side for cleaning, repairing, etc. **2** [VP6A, 2A] (cause to) tilt, lean over to one side. [OALD]

```
form: [ orth: careen
        hyph: ca.reen
        pron: k@'ri:n ]

gram: [ pos: v
        subc: (tr, intr) ]

( //sense 1
  [ gram: [gcode: VP6A]
    def:  [text: turn (a ship)...] ]

  //sense 2
  [ gram: [gcode: (VP6A, VP2A)]
    def:  [text : (cause to) tilt...] ] )
```

Fig. 27. Partial factoring

```
( //sense 1
  [ form: [ orth: careen
            hyph: ca.reen
            pron: k@'ri:n ]
    gram: [ pos: v
            subc: (tr, intr)
            gcode: VP6A ]
    def:  [ text: turn (a ship)...] ]

  //sense 2
  [ form: [ orth: careen
            hyph: ca.reen
            pron: k@'ri:n ]
    gram: [ pos: v
            subc: (tr, intr)
            gcode: (VP6A, VP2A)]
    def:  [ text : (cause to) tilt...] ] )
```

Fig. 28. Unfactored version of the feature structure in Fig. 27

3.6 *Exceptions and overriding*

We saw in the previous section that compatible information can appear at various levels in a disjunction. Exceptions in dictionaries will be handled by allowing *incompatible* information to appear at different levels. When this is the case, unfactoring will be defined to *retain only the information at the innermost level.* In this way, a value specified at the outer level is *overridden* by a value specified for the same feature at an inner level. For example, Fig. 29 shows the factored entry for *conjure,* in which the pronunciation specified at the outermost level applies to all senses except sense 3, where it is overriden. Fig. 30 gives the unfactored version of the entry.

con•jure /'k^ndG@(r)/ *vt,vi* **1** [VP2A,15A] do clever tricks which appear magical... **2** [VP15B] **~ *up,*** cause to appear as if from nothing... **3** /k@n'dGW@(r)/ [VP17] (formal) appeal solemnly to... [OALD]

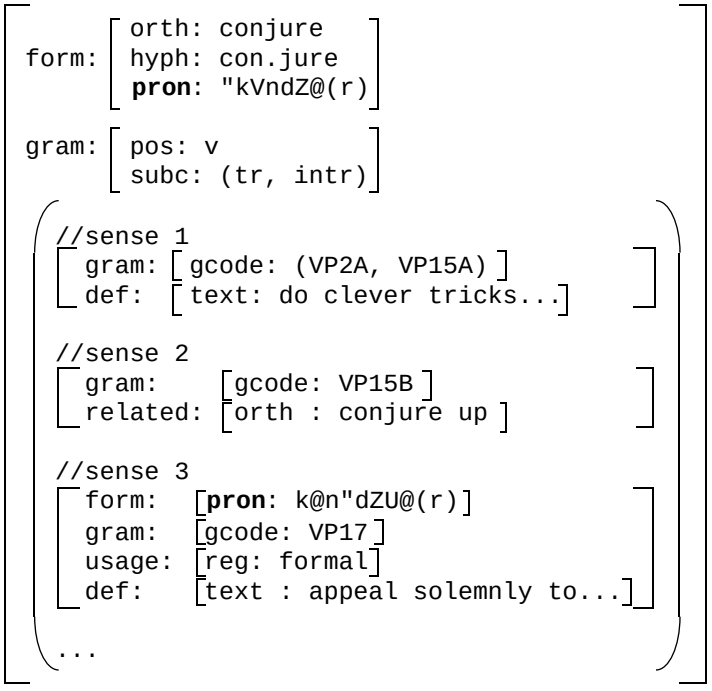


Fig. 29. Overriding of values

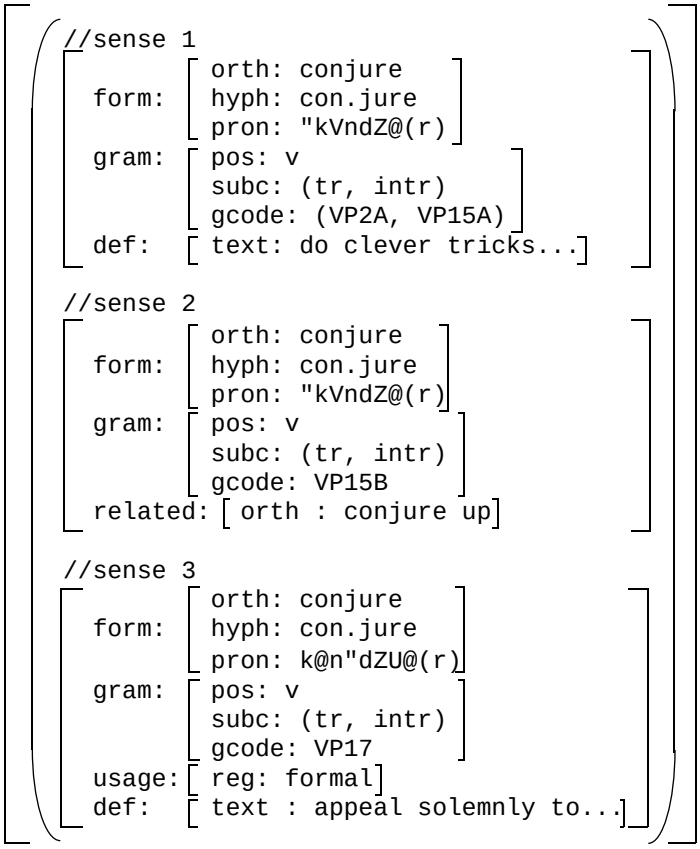


Fig. 30. Unfactored version of the feature structure in Fig. 29

4. IMPLEMENTATION : AN OBJECT-ORIENTED PROTOTYPE

An implementation of the model described above presents difficulties because there exist no DBMSs based on features structures. The feature-based systems developed so far are designed for parsing natural language and are not intended to be used as general DBMSs. Therefore, they typically do not provide even standard database operations. They are furthermore usually restricted to handle only a few hundred grammar rules, and so even the largest systems are incapable of dealing with the large amounts of data that would be required for a dictionary.

We have already seen that existing general DBMSs, including relational and unnormalized systems, are too limited to handle lexical data.[9] However, a new generation of

[9]Some recently developed NF^2 relational models (Schek, et al., 1990) allow recursive nesting of relations, which could provide a more natural means to represent lexical data in a relational system. We have not explored this possiblity.

DBMSs which are *object-oriented* may provide the required expressiveness and flexibility. Object-oriented models allow for highly structured objects by enabling the construction of new types, and in particular, recursive types, as well as providing complex built-in type constructors such as lists and sets. The underlying principle of the object-oriented approach is to eliminate computer-based concepts such as records and fields (the fundamental concepts in the relational model), and enable the user to deal with higher-level concepts that correspond more directly to the real world objects the database represents. Objects within the database, together with all of the attributes associated with them (and even operations and functions for manipulating these attributes) are considered as wholes, whereas in relational models, objects do not exist as wholes but are instead split across the various relations defined in the database.

A number of object-oriented DBMSs are currently operational (for instance, GemStone, GBASE, ONTOS --see Gardarin & Valduriez, 1990). Our implementation uses the O_2 system, which is outlined briefly in section 4.1. Section 4.2 demonstrates how the feature-based model for dictionaries is mapped into the O_2 data model. Section 4.3 describes the O_2 implementation.

4.1 *The O_2 system and model*

O_2 is an object-oriented DBMS specifically designed for "new applications" such as CAD/CAM or editorial information systems and office automation (Deux *et al.,* 1991). The O_2 environment includes:

- an object-oriented "4th generation" programming language (O_2C), which is an extension of C that enables database manipulation and user interface generation;
- a query language (O_2Query), which is an extension of SQL that enables handling complex values and objects. The query language can be used independently and interactively, or it can be called from O_2C;
- a user interface generator (O_2Look), based on Motif and XWindows;
- a object-oriented programming environment (debugger, database browser, etc.).

An O_2 database (see Lecluse & Richard, 1989) consists of a set of *objects*, each of which consists of an identifier-value pair *<i,v>*. The identifier for any object is unique. Values may be either null, atomic (integers, reals, strings, and booleans) or complex, in which case they are defined as follows:

- if n_1, … n_p are attributes and x_1, … x_p are values or identifiers, then *tuple(*n_1*:* x_1, … n_p*:* x_p*)* is a value,
- if x_1, … x_p are values or identifiers then *set(*x_1, … x_p*)* and *list(*x_1, … x_p*)* are values.

For example, the following are objects:

```
<o0, tuple(name: "Fred", spouse: o1, children: set(o2, o3))>
<o1, tuple(name: "Mary", spouse: o0, children: set(o2, o3))>
<o2, tuple(name: "John", spouse: null, children: null)>
<o3, tuple(name: "Paul", spouse: null, children: null)>
```

Objects are grouped in *classes*, which correspond more or less to the traditional notion of abstract data types. Each class is defined by its name, the type of the value of its objects, and a set of *methods*, that is, the set of operations or functions that can be performed on objects of that class. Classes are organized in a class hierarchy, where subclasses automatically inherit methods defined for the superclass. Types for subclasses can be specialized according to a partial order relation (*subtyping*) among types.

4.2 *Mapping feature structures into the O_2 model*

The O_2 and feature structure models bear certain obvious similarities. Simple feature structures correspond to tuples in the O_2 system; features are analogous to attributes. For example, the feature structure

```
[f: a]
[g: b]
```

can be translated into the O_2 object

```
<o1, tuple(f: "a", g: "b")>
```

Complex feature structures are translated into multiple objects in O_2:

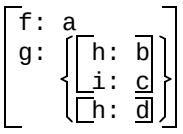
```
[f: a      ]
[g: [h: b] ]
   [i: c]
```

becomes the set of O_2 objects

```
<o1, tuple(f: "a", g: o2)>
<o2, tuple(h: "b", i: "c")>
```

Each feature structure type corresponds to an O_2 class. Since type schemas for feature structures specify all possible features that may appear, when implemented in O_2, attributes corresponding to missing features will have null values.

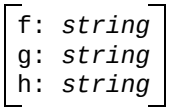
Value disjunction in feature structures is implemented in O_2 using sets and lists. For example,



is translated into

```
<o1, tuple(f: "a", g: set(o2, o3))>
<o2, tuple(h: "b", i: "c")>
<o3, tuple(h: "d", i: null)>
```

O_2 does not provide a built-in mechanism to handle general disjunction. However, general disjunction can be implemented through recursive types. Close examination of the feature structure model above shows that every node in the hierarchical normal form tree (see Fig. 24) has the same type. In O_2, this is implemented by introducing an additional recursive attribute `DISJ` in tuples. For example, if type T for some feature structure is

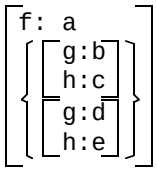


the class definition for T in O_2 is[10]

```
class  T
     type tuple  (f: string,
                  g: string,
                  h: string,
                  DISJ: set (T))
end
```

[10]Note that because of the way we implement general disjunction in O_2, it is necessary to indicate the type of disjunction (set or list) for each feature-structure type at the time it is declared. This is not required in the feature-based model.

Then, the feature structure of type T:



is implemented as

```
<o1, tuple(f: "a", g: null, h: null, DISJ: set(o2, o3))>
<o2, tuple(f: null, g: "b", h: "c", DISJ: null)>
<o3, tuple(f: null, g: "d", h: "e", DISJ: null)>
```

Note that the unfactor operator has to be programmed as a method, since general disjunction is not a built-in mechanism in O_2. The unification operation on which unfactor relies (see section 3.5) is also not built-in.

Fig. 31 shows a few simplified class definitions appropriate for a lexical database. Fig. 32 shows in graphic form the O_2 implementation for the entry *abandon* from the *LDOCE*. Each box in the figure corresponds to an O_2 object.

```
class  Entry
   type tuple  (form: list (Form),
                gram: list (Gram),
                def:  list (Def),
                ...
                DISJ: list (Entry))
end ;

class  Form
   type tuple  (orth: list (string),
                pron: list (string),
                hyph: list (string),
                geo:  list (string),
                DISJ: list (Form))
end ;

class  Gram
   type tuple  (pos:  list (string),
                subc: list (string),
                gend: list (string),
                numb: list (string),
                DISJ: list (Gram))
end ;

...
```

Fig. 31. O_2 class definitions for a lexical database

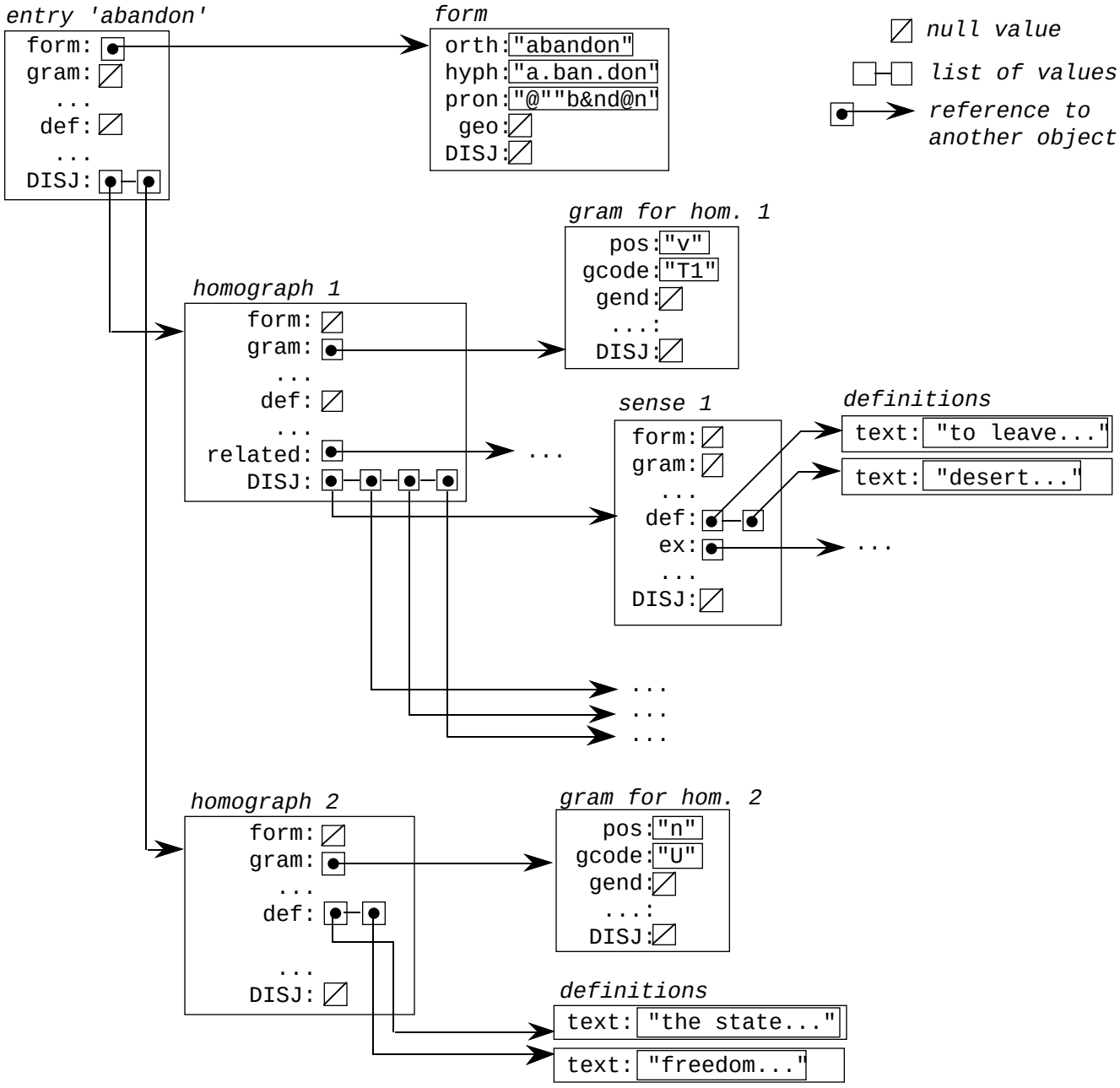


Fig. 32. Representation of the entry 'abandon' in O_2

4.3 *Implementation*

We have implemented a lexical database in a prototype version of O_2. The different classes were defined according to the schema described in the preceding section, and the methods were programmed in O_2C.

4.3.1. *Creation of the lexical database.* The dictionary we have used is the *Zyzomys*, published by Hachette and distributed on CD-ROM. The *Zyzomys* is encoded with mixed markup, including both procedural markup (for example, `/IT` indicates italicized text, `/RO` indicates roman, etc.) and descriptive markup (for example, `/DP` and `/FP` delimit the phonetic transcription, `/ME` marks the orthographic form) (see sections 2.1.1 and 2.1.2 and Fig. 33).

gin [dZIn] n. m. Eau-de-vie de grain aromatisée au genièvre, fabriquée notam. en G-B. - Mot angl. *gin*, adapt. du neerl. *jenever*, =genièvre=.

```
/MDGIN/FD /MEgin  /FE /DPdZin/FP /GE/BGn. m./GG438/GB/FG Eau-de-vie de
grain aromatisée au genièvre, fabriquée /BGnotam./GG445/GB en /BGG.-
B./GG262/GB - Mot /BGangl./GG035/GB /ITgin/RO, adapt. du /BGneerl.
/GG433/GB /ITjenever/RO, =genièvre=.
```

Fig. 33. The entry for 'gin' from the *Zyzomys* and its original encoding

The first step was to analyze the entire dictionary in order to isolate the different logical fields composing each entry, and organize them according to the model defined above. The results of this analysis were encoded in SGML according to a preliminary version of the guidelines for encoding monolingual dictionaries, which we developed while working within the Text Encoding Initiative (Ide, Véronis, Warwick-Armstrong & Calzolari, in press).

Each feature of our model (for example, *form*, *gram*, etc.) corresponds to an SGML element (`<form>...</form>`, `<gram>...</gram>`, etc.--see Fig. 34), as well as to an O_2 class (`Form`, `Gram`, etc.).[11] The dictionary is translated from its SGML format by means of a recursive descent procedure: each time a new opening tag (for example, `<form>`) is encountered, a method in the corresponding O_2 class (for example, `Form`) is triggered, which translates the content of the element concerned into an O_2 value or object.

A few problems were encountered in the process of creating the database, mainly because the version of O_2 used in the experiments was a prototype. In our model, each entry in the dictionary is represented by a tree structure consisting of potentially several dozens of objects. The object manager of our prototype version of O_2 could not handle the hundreds of thousands of objects corresponding to the approximately 50,000 entries of the *Zyzomys*. We therefore limited our experiments to 600 entries. This problem is completely resolved in the commercial version of O_2, which is now available.

[11] Etymology is not included in our database.

```
form: [orth: gin                                    ]
      [pron: dZIn                                   ]
gram: [pos: n                                       ]
      [gen: m                                       ]
def:  [text: Eau-de-vie de grain aromatisée au      ]
      [      genièvre, fabriquée notam. en G.-B.    ]
```

(a) feature structure

```
<entry>
  <form>
    <orth>gin</orth>
    <pron>dZIn</pron>
  </form>
  <gram>
    <pos>n</pos>
    <gen>m</gen>
  </gram>
  <def>
    <text>Eau-de-vie de grain aromatisée au
       genièvre, fabriquée notam. en G.-B.</text>
  </def>
</entry>
```

(b) SGML encoding

Fig. 34. Representations of 'gin' from the *Zyzomys*.

Null values caused a second problem. In both the prototype and commercial versions of O_2, null values are explicitly stored, which, especially for the sparse data in lexical entries, is very space-consuming. However, it is conceivable that future versions of O_2 or other object-oriented DBMSs could solve this problem by not representing null values internally.

4.3.2 *Querying the database*. A typical query asks to extract all entries (or parts of entries) which have certain attribute values (for example, the query for the *LDOCE* given as an example in section 2.2.1, intended to extract all examples for countable nouns whose definitions start with "the state..."--similar queries could be applied to the *Zyzomys*). When the user queries the database, he or she does not know *a priori* how the target definitions are factored (or even if they are all factored in the same way) and therefore at what level certain attributes such as `gram.pos` and `def.text` appear.

We have programmed a query interface in O_2C, which enables the user to formulate queries in an unfactored, "flat" format and determines that attribute values appear at appropriate places within the tree. The retrieval process involves two steps. First, an index for each atomic attribute (`gram.pos`, `gram.gcode`, `def.text`, etc.) enables retrieving all entries with a given value for that attribute. In the *LDOCE* example, the indexes will enable retrieving all entries containing all of the attribute-value pairs in the query (`gram.pos` : "n", `gram.gcode` : "U", `def.text` : "state"). However, because there is no indication in the indexes of where the attribute-value pairs appear in the entry trees, this first step will also

retrieve, for example, entries in which there exist both a noun and a verb homograph, and where "state" appears in the definition text for the verb homograph. Similarly, it will retrieve entries where "n" has been overriden at a lower level and the definition text containing "state" appears at this lower level. Therefore, a second step is necessary, to recursively traverse the trees of the entries retrieved in the first step, and retain only those which match the query exactly.

This solution is not completely satisfactory, since it by-passes the query language and therefore does not take advantage of its features. Ideally, the database query language should include an operator enabling the traversal of recursive structures. Because O_2Query does not allow the definition of new operators, we are working on an extension to the query language LIFOO (developed in part by one of the authors as a functional query language for O_2--see Le Maitre & Boucelma, in press) which is specifically constructed to support the definition of new operators (Boucelma & Le Maitre, 1991).

5. CONCLUSION

In this paper we show that previously applied data models are inadequate for lexical databases. In particular, we show that relational data models, including normalized models which allow the nesting of attributes, cannot capture the structural properties of lexical information. We propose an alternative feature-based model for lexical databases, which departs from previously proposed models in significant ways. In particular, it allows for a full representation of sense nesting and defines an inheritance mechanism that enables the elimination of redundant information. The model provides flexibility which seems able to handle the varying structures of different monolingual dictionaries. Our model may therefore be applicable to a diversity of uses of electronic dictionaries, ranging from research to publication.

We also show how the feature-based model can be implemented in an object-oriented DBMS, and demonstrate that feature structures map readily to an object-oriented data model. Our work suggests that the development of a feature-based DBMS, including built-in mechanisms for disjunction, unification, generalization, etc., is desirable. Such feature-based DBMSs could have applications far beyond the representation of lexical data.

A number of open problems remain for fully specifying the structure and elements of lexical databases. For example, we have not addressed the problems of phrasal elements (such as discontinuous verb phrases and cross-reference phrases embedded in definition or

example text), etymologies (which are themselves complex structured text), etc. Further, it is necessary to test our model across a wide range of monolingual dictionaries in order to ascertain, first, its generality and, second, the exact scope and nature of remaining difficulties.

Acknowledgments -- The present research has been partially funded by the GRECO-PRC Communication Homme-Machine of the French Ministery of Research and Technology, U.S.-French NSF/CNRS grant INT-9016554 for collaborative research, and U.S. NSF RUI grant IRI-9108363. The authors would like to acknowledge the GIP-Altaïr for making available a prototype version of the O_2 DBMS (contract GIP-Altaïr #88-5), and Collins Publishers, Hachette, Longman Group, and Oxford University Press for making their data available for research within the project. The authors would also like to thank Christine Tribocky, Régis Voillaume, and Christiane Fantozzo for their work on the implementation, Jean-Michel Ombrouck for his pre-processing of the Hachette *Zyzomys*, Mary Neff for providing the IBM LDB example, and Frank Tompa for his valuable comments on an earlier draft of this paper.

REFERENCES

Abiteboul, S., & Bidoit, N. (1984). Non first normal form relations to represent hierarchically organized data. *Proceedings of the ACM SIGACT/SIGMOD Symposium on Principles of Database Systems.* Waterloo, Ontario, 191-200.

Amsler, R. A. (1980). *The structure of the Merriam-Webster Pocket Dictionary.* Doctoral dissertation, University of Texas at Austin.

Amsler, R. A., & Tompa, F. W. (1988). An SGML-based standard for English monolingual dictionaries. *Proceedings of the 4th Annual Conference of the UW Centre for the New Oxford English Dictionary.* Waterloo, Ontario, 61-80.

Bancilhon, F., Fortin, D., Gamersan, S., Laubin, J.-M., Richard, P., Scholl, M., Tusera, D., & Verroust, A. (1983). VERSO: A relational backend database machine. In D. K. Hsiao (Ed.), *Advanced Database Machine Architecture*. Englewood Cliffs: Prentice Hall.

Boguraev, B., & Neff, M. S. (in press). From machine readable dictionaries to lexical databases. *International Journal of Lexicography*.

Boucelma, O., & Le Maitre, J. (1991). An extensible functional query language for an object-oriented database system. In C. Delobel, M. Kifer, Y. Masunaga (Eds.), *Deductive and Object-Oriented Databases.* Lecture Notes in Computer Science, Berlin: Springer Verlag.

Brustkern, J., & Hess K. (1982). The BONNLEX lexicon system. In J. Goetschalckx & L. Rolling (Eds.), *Lexicography in the Electronic Age.* Amsterdam: North-Holland.

Byrd, R. J., Calzolari, N., Chodorow, M. S., Klavans, J. L., Neff, M. S., & Rizk, O. Tools and methods for computational linguistics. *Computational Linguistics, 13*(3/4), 219-240.

Calzolari, N. (1984). Detecting patterns in a lexical data base. *Proceedings of the 10th International Conference on Computational Linguistics, COLING'84.* Stanford, California, 170-173.

Calzolari, N., Peters, C., & Roventini, A. (1990). *Computational Model of the Dictionary Entry.* (ACQUILEX Preliminary Report, Esprit Basic Research Action No. 3030). Pisa, Italy: Istituto di Linguistica Computazionale.

Coombs, J. H., Renear, A. H., & DeRose, S. J. (1987). Markup systems and the future of scholarly text processing. *Communications of the ACM, 30*(11), 933-47.

Deux, O. *et al.* (1991). The O_2 System. *Communications of the ACM, 34*(10), 34-48.

Gardarin, G., & Valduriez, P. (1990). *SGBD Avancés, Bases de Données Objets, Déductives, Réparties.* Paris: Eyrolles.

Gonnet, G., & Tompa, F. W. (1987). Mind your grammar: a new approach to modelling text. *Proceedings of the 13th Conference on Very Large Data Bases, VLDB'87.* Brighton, England, 339-346.

Gonnet, G., Baeza-Yates, R. A., & Snider, T. (1991). *Lexicographical indices for text: Inverted files vs. PAT trees* (Technical report OED-91-01). Waterloo, Ontario: UW Centre for the New Oxford English Dictionary and Text Research.

Ide, N., Véronis, J., Warwick-Armstrong, S., & Calzolari, N. (in press). Principles for Encoding machine readable dictionaries. *Proceedings of the Fifth EURALEX International Congress, EURALEX'92.* Tempere, Finland.

Kaplan, R. and & Bresnan, J. (1982). Lexical-functional grammar: A formal system for grammatical representation. In J. Bresnan (Ed.), *The Mental Representation of Grammatical Relations.* Cambridge, Massachussets: MIT Press.

Karttunen, L. (1984). Features and values. *Proceedings of the 10th International Conference on Computational Linguistics, COLING'84.* Stanford, California, 28-33.

Kay, M. (1985). Parsing in functional unification grammar. In D.R. Dowty, L. Karttunen, & A. M. Zwicky (eds.). *Natural Language Parsing*. Cambridge: Cambridge University Press.

Kipfer, B. A. (1983). Computer applications in lexicography: a bibliography. *Dictionaries: Journal of Dictionary Society of North America, 4,* 202-237.

Klavans, J., Chodorow, M., & Wacholder, N. (1990). From dictionary to knowledge base via taxonomy. *Proceedings of the 6th Annual Conference of the UW Centre for the New Oxford English Dictionary.* Waterloo, Ontario, 110-132.

Landau, S. I. (1984). *Dictionaries: The Art and Craft of Lexicography.* New York: Scribner Press.

Le Maitre, J., & Boucelma, O. (in press). LIFOO, un langage fonctionnel de requêtes pour bases de données avancées. *Technique et Science Informatiques.*

Lecluse, C., & Richard, P. (1989). The O_2 database programming language. *Proceedings of the 15th Conference on Very Large Data Bases, VLDB'87.* Amsterdam, 411-422.

Markowitz, J., Ahlswede, T., & Evens, M. (1986). Semantically significant patterns in dictionary definitions. *Proceedings of the 24th Annual Conference of the Association for Computational Linguistics.* New York, 112-119.

Nakamura, J., & Nagao, M. (1988). Extraction of semantic information from an ordinary English dictionary and its evaluation. *Proceedings of the 12th International Conference on Computational Linguistics, COLING'88.* Budapest, Hungary, 459-464.

Neff, M. S., Byrd, R. J., & Rizk, O. A. (1988). Creating and querying lexical databases. *Proceedings of the Association for Computational Linguistics Second Applied Conference on Natural Language Processing.* Austin, Texas, 84-92.

Pistor, P., & Traunmueller, R. (1986). A database language for sets, lists and tables. *Information Systems, 11*(4), 323-336.

Pollard, C., & Sag, I. A. (1987). *Information-based Syntax and Semantics.* CSLI Lecture Notes Series, Chicago: University of Chicago Press.

Roth, M. A., Korth, H. F., & Silberschatz, A. (1988). Extended algebra and calculus for nested relational databases. *ACM Transactions on Database Systems, 13*(4), 389-417.

Schek, H.-J., Paul, H.-B., Scholl, M.H., & Weikum, G. (1990). The DASDBS project: objectives, experiences, and future prospects. *IEEE Transactions on Knowledge and Data Engineering, 2*(1), 25-42.

Shieber, S.M. (1986). *An Introduction to Unification-based Approaches to Grammar*. CSLI Lecture Notes Series, Chicago: University of Chicago Press.

Sinclair, J. M. (1987). *An Account of the COBUILD Project.* London: Collins ELT.

Sperberg-McQueen, M., & Burnard, L. (1990). *Guidelines for the encoding and interchange of machine-readable texts, Draft, Version 0.0.* ACH, ACL, and ALLC.

The DANLEX Group (1987). Descriptive tools for electronic processing of dictionary data. *Lexicographica, Series Maior.* Tübingen: Niemeyer.

Tompa, F. W. (1989). What is tagged text? *Proceedings of the 5th Annual Conference of the UW Centre for the New Oxford English Dictionary.* Oxford, England, 81-93.

Véronis, J., & Ide, N., M. (1990). Word Sense Disambiguation with Very Large Neural Networks Extracted from Machine Readable Dictionaries. *Proceedings of the 13th*

International Conference on Computational Linguistics, COLING'90. Helsinki, Finland, *2,* 389-394.

Wilks, Y., Fass D., Guo, C., MacDonald, J., Plate, T., & Slator. B. (1990). Providing Machine Tractable Dictionary Tools. *Machine Translation, 5,* 99-154.