A FEATURE-BASED MODEL FOR LEXICAL DATABASES

JEAN VÉRONIS, NANCY IDE

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

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

e-mail : veronis@vassar.edu, ide@vassar.edu

Abstract -- To date, no fully suitable data model for lexical databases has been proposed. As lexical databases have proliferated in multiple formats, there has been growing concern over the reusability of lexical resources. In this paper, we propose a model based on feature structures which overcomes most of the problems inherent in classical database models, and in particular enables accessing, manipulating or merging information structured in multiple ways. Because of their widespread use in the representation of linguistic information, the applicability of feature structures to lexical databases seems 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.

1. INTRODUCTION

There exists a substantial body of research demonstrating that machine readable dictionaries are a rich source of ready-made lexical and semantic information which can be used in natural language processing (for example, Amsler, 1980; Calzolari, 1984; Markowitz, Ahlswede, and Evens, 1986; Byrd *et al.*, 1987; Nakamura and Nagao, 1988; Véronis and Ide, 1990; Klavans, Chodorow, and Wacholder, 1990; Wilks *et al.*, 1990). Much of this research involves the creation of *lexical databases* from original dictionary data, in order to facilitate retrieval and analysis. However, lexical data 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 well to lexical data, and, as a result, current lexical databases exist in a wide variety of (often *ad hoc*) formats. To date, no fully suitable data model for lexical databases has been proposed.

As lexical databases have proliferated in multiple formats, there has been growing concern over the *reusability* of lexical resources. The interchange and integration of data, as well as the development of common software, is increasingly important to avoid duplication of effort and enable the development of large-scale databases of linguistic information (which is the concern of projects such as ACQUILEX, GENELEX, EDR, etc.).

In this paper, we provide a data model that is suited to lexical databases. A strong requirement for such a data model is that it must make lexical information compatible despite its variability in structure across the dictionaries from which it is derived. We show that a model based on *feature structures* overcomes most of the problems inherent in classical database models, and, in particular, enables accessing, manipulating or merging information structured in multiple ways. The feature-based model also allows retaining the particular organization of a given dictionary while at the same time making it invisible to certain retrieval operations. Because of their widespread use in the representation of linguistic information, the applicability of feature structures to lexical databases seems 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.

2. PREVIOUS MODELS

The classical relational model has been proposed to represent dictionaries (Nakamura and Nagao, 1988). However, as Neff, Byrd, and Rizk, 1988, point out, the relational model cannot capture the obvious hierarchy in most dictionary entries. For example, the entry for *abandon* in Fig. 1 has two main sub-parts, one for its verb senses and one for its noun sense, and the two senses of the verb labeled "1" in Fig. 1 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 of relational models obscures this fact.

Neff, Byrd, and Rizk describe a lexical database (the IBM LDB) based on an *unnormalized* (also *Non First Normal Form* or *NF*^{2}) relational data model, in which attribute values may be nested relations with their own internal structure (see Abiteboul and Bidoit, 1984; Roth *et al.*, 1988). Fig. 2 shows the *LDOCE* entry for *abandon* represented in a 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.

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

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. 1. Definition of 'abandon' from *LDOCE*

2.1 Recursive nesting

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. 3).

valeur [valœR] n. f. **A. I. 1.** Ce 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. 3. Part of the definition of 'valeur' in *Zyzomys*

NF[2] models explicitly prohibit recursive embedding of relations. Therefore, the only way to represent the recursive nesting of senses is through the proliferation of attributes such as `SENSE_LEVEL1`, `SENSE_LEVEL2`, etc. to represent the different levels. 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, mulitple queries 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.

2.2 Exceptions

Exceptional cases are characteristic of lexical data. For instance, sense 3 of the word "conjure" in the *OALD* has a pronunciation different from the other senses in the entry, and the entry "heave" in the *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" (Fig. 4).

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]

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 ... [CED]

Fig. 4. Exceptions in dictionary 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. If any attribute can appear in any nested relation, the model

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. 2. NF[2] representation of the entry 'abandon'

becomes ill-defined since the very notion of hierarchy upon which it relies is undermined. Therefore, 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.

2.3 Variable factoring

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

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

We introduce a model for dictionary data based on *feature structures*. We demonstrate the mapping between the information found in dictionaries and the feature-based model, and show how the various characteristics of lexical data, such as recursive nesting of elements, (variable) factoring of information, and exceptions can be handled using well-developed feature structure mechanisms.

Fig. 5 shows how feature structures can be used to represent simple dictionary entries. We will consider feature structures as *typed* (as defined, for instance, by Pollard and Sag, 1987), that is, not all features can appear anywhere, but instead, they must follow a schema that specifies which features are *allowable* (although not necessarily present), and where. 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'petIt@(r)/ *n* person who competes [OALD]

```
form: [orth: competitor      ]
      [hyph: com.peti.tor    ]
      [pron: k@m'petIt@(r)   ]
gram: [pos:  n]
def:  [text: person who competes]
```

Fig. 5. Representation of a simple sense

3.1 Value disjunction and variants

The use of value disjunction (Karttunen, 1984) enables the represention of variants, common in dictionary entries, as shown in Fig. 6. We have added an extension which allows 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 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.

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. 6. 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. 7 shows a typical example where the alternatives are *groups* consisting of orthography and pronunciation.

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... ]
```

Fig. 7. Value disjunction of non-atomic values

3.2 General disjunction and factoring

General disjunction (Kay, 1985) provides a means to specify alternative *sub-parts* of 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 $f: \psi$, a set of feature structures $\{\psi_1, \ldots \psi_p\}$, or a list of feature structures $(\psi_1, \ldots \psi_p)$.

General disjunction allows common parts of components to be factored. 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, but only general disjunction (Fig. 8) captures the obvious factoring and represents the entry cleanly and without redundancy.

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

```
form:  [ pron: 'hQspIt@l@ ]
       ( [orth: hospitaller] )
       ( [geo:  U.S.        ]
         [orth: hospitaler  ] )
gram:  [pos: n]
def:   [text: a person...]
```

Fig. 8. General disjunction

General disjunction provides a means to represent multiple senses, since they can be seen as alternatives (Fig. 9).[1]

Sense nesting is also easily represented using this mechanism. Fig. 10 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. Within the first component of the disjunction, the different senses for the verb comprise an embedded list of disjuncts.

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

```
form : [ orth: disproof ]
       [ pron: dIs'pru:f ]
gram : [ pos: n ]
( //sense 1
  [def: [ text: facts that disprove.. ]]

  //sense 2
  [def: [ text: the act of disproving ]] )
```

Fig. 9. Representation of multiple senses

An important characteristic of this model is that there is no different *type* feature structure for entries, homographs, or senses. This captures what appears to be a fundamental property of lexical data, that is, that the different levels (entries, homographs, senses) are associated with the same kinds of information. Previous models have treated these different levels as different objects, associated wtih different kinds of information, which obscures the more fundamental structure of the information.

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 $\phi_n = \{\psi_1, \ldots \psi_p\}$, 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. 11 represents the feature structure for *abandon* given in Fig. 10. The representation of information as a tree of feature structures, where each node represents a level of hierarchy in the dictionary, reflects structure and factoring of information in dictionaries and captures the fundamental similarity among levels cited above.

3.3 Disjunctive normal form and equivalence

It is possible to define an *unfactor* operator to multiply out the terms of alternatives in a general

[1]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.

disjunction (Fig. 12), assuming that no feature appears at both a higher level and inside a disjunct.[2]

By applying the unfactor operator recursively, it is possible to eliminate all disjunctions except at the top level. resulting (extremely redundant) structure is called the *disjunctive 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. 13a shows the factoring for inflected forms of *alumnus* in the *CED;* the same information could have been factored as it appears in Fig. 13b. Note that we have used *sets* and not *lists* in Fig. 13. 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.

[2]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.

```
form: [ orth: abandon
        hyph: a.ban.don
        pron: @"b&nd@n ]

 ( //homograph 1
   [ gram: [ pos:   v
             gramc: T1 ]

     ( //sense 1
       [ sem: [ scode: ----
                boxc: ----H----T ]
         def: ( [ text: to leave completely and for ever ]
                [ text: desert ] )
         ex:  [ text: The sailors abandoned the burning ship ] ]

       //sense 2
       [ ...                                                  ]

       ... )

   related:[ orth: abandonment ] ]

   //homograph 2
   [ gram: [ pos:   n
             gramc: U ]
     sem: [ scode: ----
            boxc: ----T-----]
     def: [ text: the state when one's feelings and actions...  ]
     ex:  [ text: The people were so excited that they jumped...] ] )
```

Fig. 10. Representation of the entry *abandon* in *LDOCE*

```
                    [ form: [ orth: abandon
                              hyph: a.ban.don
                              pron: @"b&nd@n ] ]

//homograph 1                                   //homograph 2
[ gram: [ pos: v      ] ]                       [ gram: [ pos: n      ]
        [ gramc: T1   ] ]                               [ gramc: U    ]
                                                  sem: [ scode: ----      ]
                                                       [ boxc: ----T-----]
                                                  def: [ text: the state...]
                                                  ex:  [ text: The people..]

//sense 1
[ sem: [ scode: ----                  ]   ...   ...   ...
       [ boxc: ----H----T             ]
  def: ( [ text: to leave...] )
       ( [ text: desert]      )
  ex:  [ text: The sailors...]
```

Fig. 11. Hierarchical Normal Form

```
[ f: a                    [ f: a ]
  { [ g:b ] }             { [ g: b ] }
  { [ h:c ] }             { [ h: c ] }
  { [ g:d ] }             { [ f: a ] }
  { [ h:e ] }             { [ g: d ] }
                          { [ h: e ] }
```

Fig. 12. Unfactoring

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.

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

```
form: ⟨ [ numb: sing
          ⟨ [ gend: masc
              orth: alumnus
              pron: @"l^mn@s ]
            [ gend: fem
              orth: alumna
              pron: @"l^mn@ ] ⟩ ]
        [ numb: pl
          ⟨ [ gend: masc
              orth: alumni
              pron: @"l^mnaI ]
            [ gend: fem
              orth: alumnae
              pron: @"l^mni: ] ⟩ ] ⟩
```

(a)

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

```
form: ⟨ [ gend: masc
          ⟨ [ numb: sing
              orth: alumnus
              pron: @"l^mn@s ]
            [ numb: pl
              orth: alumni
              pron: @"l^mnaI ] ⟩ ]
        [ gend: fem
          ⟨ [ numb: sing
              orth: alumna
              pron: @"l^mn@ ]
            [ numb: pl
              orth: alumnae
              pron: @"l^mni: ] ⟩ ] ⟩
```

(b)

Fig. 13. Two different factorings of the same information

3.4 Partial factoring

The type of factoring described above does not handle the example in Fig. 14, 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.

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. 14. Partial factoring

3.5 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. 15 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.

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]

```
form: [ orth: conjure
        hyph: con.jure
        pron: "kVndZ@(r) ]

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

  ( //sense 1
    [ gram: [ gcode: (VP2A, VP15A) ]
      def:  [ text: do clever tricks...] ]

    //sense 2
    [ gram:    [gcode: VP15B]
      related: [orth : conjure up ] ]

    //sense 3
    [ form:  [pron: k@n"dZU@(r)]
      gram:  [gcode: VP17]
      usage: [reg: formal]
      def:   [text : appeal solemnly...] ]

    ... )
```

Fig. 15. Overriding of values

3.6 Implementation

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.

In Ide, Le Maitre, Véronis (forthcoming), we describe an *object-oriented* implementation which provides the required expressiveness and flexibility. We 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. However, our work suggests that the development of a feature-based DBMS, including built-in mechnisms for disjunction, unification, generalization, etc., is desirable. Such feature-based DBMSs could have applications far beyond the representation of lexical data.

4. 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 a flexibility which seems able to handle the varying structures of different monolingual dictionaries.

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 contribution of discussions with Jacques Le Maitre to the ideas in this paper.

REFERENCES

ABITEBOUL, S., BIDOIT, N. (1984) Non first normal form relations to represent hierarchically organized data. *Proc. 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*. Ph. D. Dissertation, U. Texas at Austin.

BYRD, R. J., CALZOLARI, N., CHODOROW, M. S., KLAVANS, J. L., NEFF, M. S., RIZK, O. (1987) Tools and methods for computational linguistics. *Computational Linguistics,* 13(3/4): 219-240.

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

IDE, N., LE MAITRE, J., VÉRONIS, J. (forthcoming) Outline of a model for lexical databases. *Information Processing and Management.*

KARTTUNEN, L. (1984) Features and values. *Proc. 10th International Conference on Computational Linguistics, COLING'84;* Stanford, California; 28-33; 1984.

KAY, M. (1985) Parsing in functional unification grammar. In: Dowty, D.R., Karttunen, L., and Zwicky, A.M., editors. *Natural Language Parsing;* Cambridge: Cambridge University Press.

KLAVANS, J., CHODOROW, M., WACHOLDER, N. (1990) From dictionary to knowledge base via taxonomy. *Proc. 6th Annual Conference of the UW Centre for the New Oxford English Dictionary;* Waterloo, Ontario; 110-132.

LECLUSE, C., RICHARD, P., (1989). The O2 database programming language. *Proc. 15th VLDB Conference;* Amsterdam; 1989.

MARKOWITZ, J., AHLSWEDE, T., EVENS, M. (1986) Semantically significant patterns in dictionary definitions. *Proc. 24rd 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. *Proc. 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. *Proc. Association for Computational Linguistics Second Applied Conference on Natural Language Processing;* Austin, Texas; 84-92.

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.

VÉRONIS, J., IDE, N., M. (1990) Word Sense Disambiguation with Very Large Neural Networks Extracted from Machine Readable Dictionaries. *Proc. 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.