[*First International Conference on Language Resources & Evaluation*, Granada, Spain, 28-30 May 1998]

The Habanera Lexical Knowledge Base Management System

Rémi Zajac

Computing Research Laboratory, New Mexico State University
zajac@crl.nmsu.edu

Abstract

Habanera is a multipurpose multilingual lexical knowledge base that is developed at CRL to be used as a central repository of multilingual lexical data. The knowledge base contains a set of dictionaries and relations between entries, within a dictionary (e.g., synonymy) as well as between entries of different dictionaries (e.g., translation). The format of monolingual lexical entries is left relatively free: the top-level follows the TEI structure, and lower-levels are dictionary and language dependent. An entry is encoded using a feature structure representation. Feature structures are typed thus enabling a formal definition of the dictionary schema; type definitions are also used by the database type-checking tools.

1 Introduction

At the Computing Research Laboratory, several monolingual and bilingual dictionaries have been built from a variety of sources: by using MRDs (e.g., a Collins Spanish-English dictionary), dictionaries compiled by hand, or computational dictionaries built in other NLP projects. These dictionaries are used for various NLP applications, including machine translation and other translation tools. The problems encountered in formatting these dictionaries to conform to a single format and in further maintaining these dictionaries were numerous. These problems include for example entries missing essential linguistic information, lack of coherence, non-standard linguistic representations, lack of reversibility. Furthermore, these dictionaries were maintained separately, and serious coherence problems arose for bilingual dictionaries. Since many of our multilingual applications have English as the target language, multiple bilingual dictionaries impose duplication of work for the English side of dictionaries. The Habanera effort is an attempt to smooth away the numerous difficulties in building and maintaining large multilingual resources for various purposes. In particular, the aims are to

- Enhance reuse of lexical resources across CRL projects. Reuse will be fostered by some degree of standardization across dictionaries, and in particular by standardization of the structure of lexical entries. A standard lexical entry should ideally be designed for use by NLP programs as well as for on-line human consultation and should support extension as required by specific applications.
- Offer a framework for easing the cost of lexical acquisition of complex structures for multilingual dictionaries, and in particular syntactic and semantic information as used by knowledge-based NLP systems. The knowledge base should also support various kinds of acquisition scenarii. For example, a dictionary can be built from scratch by lexicographers, but can also be built by reformatting and reusing parts of existing lexical resources such as existing computational lexica or MRDs.
- Guarantee coherence and consistency within and across dictionaries.

Additionally, the system should support a variety of linguistic architectures. Since the design of a lexical architecture is a complex task, flexibility in designing the structure of the lexical knowledge base is an essential feature. This flexibility is provided by allowing for a multi-layered LKB schema in which each layer provides additional constraints on the structure of a lexical entry. This approach is congruent with the distinction made in (Eagles 93) between meta-schemata, schemata and instances. This constraint also means that the system is theory-neutral: one can use the LKB to store LFG, HPSG, or other kind of lexical data.

These requirements motivate various initial choices for the design and the implementation of the system:

- Use of the TEI definition for printed dictionaries as a source of inspiration for the definition of a standard dictionary entry structure (definition of the 'meta-schema').
- Organization of dictionaries as monolingual dictionaries plus translation relations between entries. In the case of knowledge-based MT, relations are also defined between word senses and ontological concepts.
- Use of typed feature structures as the primary descriptive device. Use of type definitions to define the lexical knowledge base schema.
- Use of a commercial OODBMS back-end to store the many lexical resources accumulated at CRL and to

support concurrent users. The format of stored data (objects) is independent of the external representation formalism.

- Use of Unicode for string encoding and manipulation.

Several projects under way at CRL will be using a common lexical knowledge base, accessing various linguistic levels defined in the dictionaries: definitions for Machine-Aided Human-Translation tools, morphology for morphological analyzers and generators, grammar for parsers, etc. Within a year the Habanera lexical knowledge base will contain lexical resources[1] at various stages of development for machine translation to English from the following languages: Arabic, Chinese, Farsi, Japanese, Korean, Russian, Spanish, Serbo-Croatian, and Turkish.

This paper presents an overview of Habanera: the structure of the lexical knowledge base, of dictionaries, of entries and of relations between entries.

2 Management of Lexical Knowledge

The architecture of Habanera distinguishes three different layers:

1. The presentation layer: this is the layer used for the display of the database content. This layer is used in a variety of browsing, editing and management tools. Several presentation models can be defined depending on the needs of various classes of users (see below).
2. The data layer is used by tools manipulating dictionaries and dictionary entries. This layer is accessed through a public API.
3. The storage layer uses a database management system for implementing persistency, concurrent access and security. This layer is not accessible to the user.

Habanera also provides a set of functionalities for database administration, including access rights and recording of changes (change log) for the database itself, and for dictionaries and individual entries.

A Habanera database has a set of dictionaries and a schema which contains a set of type definitions for all dictionaries. This set of type definitions is organized in modules and sub-modules; each dictionary imports a type module. The knowledge base has a set of generic tools related to dictionary creation, construction and evolution. The tools are generic in the sense that they are parameterized by a dictionary schema.

We distinguish between 4 broad classes of users:

- The Administrators create and delete dictionaries, define access permissions, etc. They use an administration tool.
- The Lexicologists define the structure of dictionaries and dictionary entries (the dictionary schema). They also define acquisition procedures and creates on-line help. They use a dictionary management tool.
- The Lexicographers add or modify dictionary entries. They use a dictionary acquisition tool.
- The Programmers build tools for manipulating dictionaries, for example a formatter to extract a dictionary for a particular application (e.g., a morphological generator). They use the Habanera public API.

3 Dictionaries

A dictionary is a set of entries each of which is structured as specified by a dictionary schema. A dictionary may also contain a set of lexical rules which are used to expand the dictionary. For example, semi-productive derivation rules can be used to add new entries which will be manually checked. Fully productive derivation rules can be used by a morphological tool to expand the dictionary in a systematic fashion.

For each entry, the lexicographer has to specify a key which is used to provide a primary index. The dictionary meta-data contains various information useful for managing the dictionary:

- The schema of entries. This schema is specified using Typed Feature Structure definitions and one type in this set of definitions defines the structure of an entry.
- The schema of relations among entries if any. This schema is also specified as TFS definitions. A relation must specialize the Relation type. Relations are used to describe synonymy, hyperonymy, etc. They can also be used to link several monolingual dictionaries to provide translations.
- The language (as a 3-letter ISO code).
- The indexes that must be maintained by the database engine for indexing entries. These indexes are specified as paths in an entry.
- Management information (author, date, reason for modification, etc.).

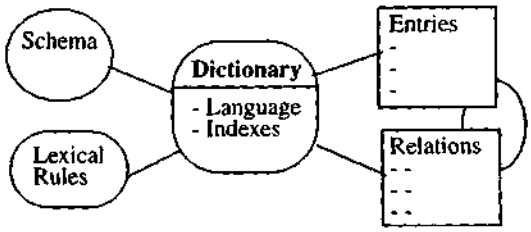


Figure 1: The Dictionary object.

1. Lexical resources for each language include a main dictionary, a dictionary of phrasal expressions, and a dictionary of proper names.

Habanera facilities include an export function which writes a dictionary (or a subset of entries) to a file using a textual format for feature structures. The corresponding import function loads entries in a given dictionary from a file. All strings stored in the database are encoded using Unicode and the import/export files are encoded in UTF-8.

4 Entries

Typed Feature Structures have been chosen for the encoding of lexical information in the database for several reasons: the feature structure format is now used by a large number of computational linguists; this format is very general and allows to structure information in an object-oriented fashion; the format has a very precise computational interpretation which makes it eminently computationally tractable; type definitions allow for type-checking of feature structures; feature structures can also be used directly in various linguistic applications such as parsers or generators. Typed feature structures have been extended to include other useful data types: Boolean, Integer, String, Sequence, Regular Expressions (used for example in multi-word entries)

Habanera does not impose any particular linguistic architecture. The only built-in constraint is that an entry is a feature structure which should be an instance of a type defined in the schema. This is in contrast with many electronic dictionary projects where the linguistic architecture of the lexical database is frozen in the implementation (see e.g., the EC-funded Multilex and Genelex projects). A dictionary entry can be any feature structure. Consistency and coherence are achieved through the use of schemata associated to each dictionary.

A dictionary schema contains a set of type definitions which are used to ensure well-formedness conditions at a logical level: a dictionary entry is an instance of a type and sub-structures in an entry are also instances of types defined in the schema.

In current lexical work at CRL, we have adopted an architecture which roughly follows the distinction between meta-schema, schema and instances as defined in (Eagles 93). Type definitions are partitioned into modules (or 'packages') and modules are related to each other via 'import' relations. In the CRL lexical knowledge base, a type module defines the generic structure of entries in all dictionaries stored in the LKB (Zajac et al. 98). This structure is language-independent and theory-neutral and corresponds to the meta-schema in Eagles terminology. A dictionary schema imports this generic module and then defines language-specific elements.

We roughly follow the TEI specification[1] for names of elements (or attributes, features, zones) although we propose a more rigid structure for entries. This structure also follows more or less the so-called "lexical view" in the TEI specification. One reason to use TEI element names is that TEI is widely known among lexicographers; it covers most of what can appear in a Machine Readable Dictionary (MRD), which will simplify acquisition from MRDs; it also covers many elements that are needed in a computational linguistic application, but not all. We depart from the TEI specification for the specification of translations, and we also add new elements for specifying computational semantic structures such as Text Meaning Representation (TMR) mappings (Onyshkevych & Nirenburg, 1994).

Logically, a dictionary is a set of sense definitions. The format of a dictionary entry allows to group several senses together to constitute a super-entry: each entry is specified as a value of the `sense` feature. If there is no `sense` feature in a super-entry, the super-entry describes a single sense (identical to having only one entry). All features that appear in a `sense` can also appear at the top-level. There could be several `sense` features within a super-entry, and also within a `sense` itself, thereby specifying a tree-like structure of senses that is used to group common information into a sense sub-tree. The interpretation is the usual interpretation of 'default inheritance' as used in many dictionaries where an entry 'inherits', by default, the features defined in the super-entry. A super-entry thus defines a set of senses which are the leaves of the tree of entries. In a dictionary entry, features can be repeated at the same level to encode disjunctive information (see also Véronis & Ide 92).

The `SuperEntry` type defines formally the structure of a super-entry and the Entry type defines the structure of entries. The `SuperEntry` and the `Entry` have a set of features ('zones') in common and the `SuperEntry` has a key which should be unique within the dictionary. The key for entries is derived from the `SuperEntry` key by concatenating the position index within the tree of entries. Thus, entries can be identified uniquely within a dictionary. The `SuperEntry` also has a type which specifies the kind of lexical element of the entries (single word, compound, phrase, inflected form, etc.).

A very simplified entry for 'conjure' looks like this:

```
[key: #k="conjure"
 form: [orth: #k,
        pron: "kVndZ@(r)"] ,
 gram: [subc: tr,
        subc: intr],
 sense: #1=[gram: [ pos: VP2A,
                    pos: VP15A]],
 sense: #2=[gram: [pos: VP15B]],
 sense: #3=[form.pron: "k@n'dZU@(r)",
            gram: [pos: VP17]]]
```

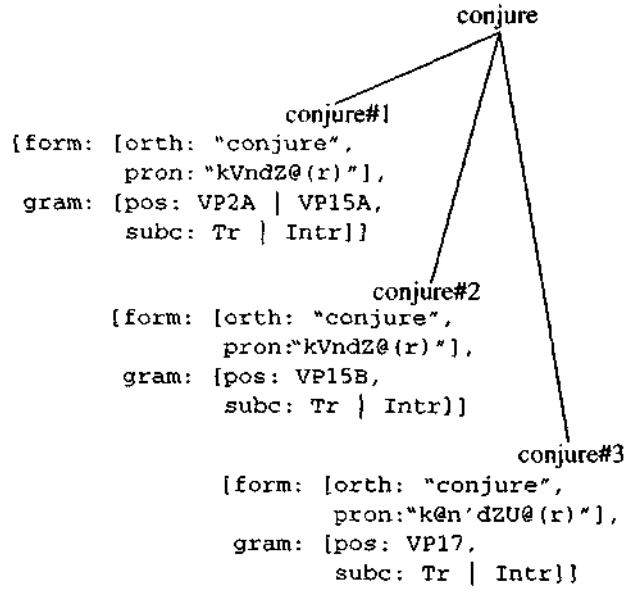
1. See an on-line version of the TEI specifications at http://www.hti.umich.edu/docs/TEI or http://etext.lib.virginia.edu/TEI.html.

Compilation and Type Checking

In the database, dictionary entries are stored as parsed feature structure objects. A dictionary compiler transforms the parsed feature structure into an internal compact representation on which the type checker can operate. This compiled form is used by the unification algorithm and is also used for runtime dictionaries which are accessed by programs (e.g., a morphological analyzer).

The dictionary compiler generates a default runtime index which uses the key feature of the entries.[1] It can also generate additional indexes using paths in entries.

The example given above would be expanded as a tree where entries corresponding to single word-senses are stored at the leaves; the tree itself is simply an index to the sub-senses:



The Habanera checker works on the compiled representation and performs three levels of checking on entries:

- A local consistency check is performed by trying to unify the definition of the entry (as specified in the dictionary schema) with the entry itself: if the unification fails, the entry is tagged as inconsistent.
- A dictionary-wide coherence check is performed by checking that relations which are defined between entries of the dictionary are well-formed and that the two entries of a relation actually exist in the dictionary.
- An LKB-wide coherence check is performed on relations which are defined across dictionaries.

1. Habanera also maintains an index on the parsed entries which is used by the Habanera editor for example. This index uses the key feature.

5 Relations between entries

Relations between entries are specified in the dictionary's schema. They are interpreted in a special way by browsers and editors: a relation defines a hyper-link to another dictionary entry. The Habanera browser provides hyper-link navigation facilities and the editor allows creation of hyper-links, which must obey the dictionary schema. Thus a Habanera knowledge base allows the creation of a complex web of lexical objects. A (binary) relation defines a class of objects which are instances of the relation. A relation has a domain, a range and a set of features and values. The domain and range have labels that are used when specifying a relation from within an entry. For example, an Antonymy relation can be expressed as:

```
Antonymy < Relation;
Antonymy = [domain: Entry,
            range: Entry, ...] .
```

In a dictionary entry *schema*, it would be specified as a normal *feature:type* declaration such as:

```
    usg: [antonym: Antonymy, ...]
```

The dictionary schema also includes the specification **of** which dictionary is the domain of the relation and which * dictionary is the range.

In an *actual entry*, an antonymy relation between the entry and another would simply be specified using the range label to point to the entry itself and the domain label **to** point to the target:

```
#x=[key: "big",
    usg: [antonym: [range: #x,
                    domain:[key: "small",
                            ...]]]]
```

Relations can also be defined between entries in different dictionaries. Since relations are defined using the TFS syntax, a relation can have additional features and relations can be organized in an inheritance hierarchy as well, to , specify for example additional validity conditions on the translation relations. For example, a translation relation between 'dog' and 'Hund' between an English and a German dictionary would appear in the English dictionary as:

```
#0=[key: "dog",
    trans.de: [range: #0,
               dom:  [key:  "Hund"]]]
```

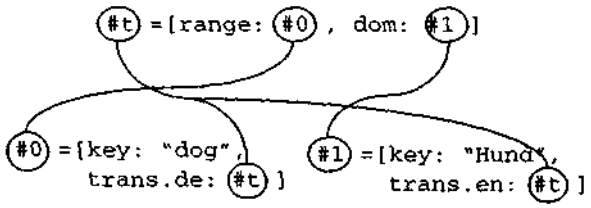
and in the German dictionary as:

```
#l=[key:   "Hund",
    trans.en:  [range:  [key: "dog",
                 dom:  #1]
```

But in the database, these two structures are stored as a single feature structure where the translation relation implements a bidirectional link between the two entries:

```
#l=[key:   "Hund",
    trans.en: #t=[range: #0=[key: "dog",
                            trans.de: #t,
                  dom:  #1]
```

Thus one can view a multilingual dictionary as a single object where each dictionary picks a different root as an entry:



6 Conclusion

The Habanera system provides a flexible and powerful framework which allows the incremental development of a linguistic architecture by providing a layered schema based on sharing of schema modules and by providing inheritance and typing through the use of typed feature structures. This system is totally language independent and theory-neutral and can cater to a large variety of linguistic architectures.

The core of the Habanera lexical knowledge base is implemented in Java using the ObjectStore OODBMS from ObjectDesign. The typed feature structure component, called Tango, has been developed independently. The Habanera core provides a complete API on which a set of tools is being developed:

- An administration tool,
- A lexicologist tool,
- A generic dictionary browser/editor which will include advanced querying facilities.

Current work focuses on development of lexical tools and the construction of bilingual dictionaries between Turkish, Farsi, Korean and English. All dictionary entries share a common structure which defines the language-independent structure of an entry. Language-dependent structures are specified independently for each dictionary.

Acknowledgments This work is part of the Corelli project by the Maryland Procurement Office, Fort George G. Meade, MD under grant MDA904-96-C-1040. Mike Freider is implementing the Tango typed feature structure package, and Jane Freider has implemented the Habanera core and is developing lexical tools. The system is being integrated and tested in a Turkish-English machine translation system under the supervision of Jan Amtrup (Expedition project).

7 References

Daniel Bauer, Frédérique Segond and Annie Zaenen. 1994. "Enriching an SGML-tagged bilingual dictionary for machine-aided comprehension". *Technical Report MLTT-011,* Rank Xerox Research Centre, Grenoble, France, October 1994.

Choy-Kim Chua and Salina A. Amat. 1994. "From a bilingual non-electronic dictionary to a ready-to-print bilingual/trilingual electronic dictionary". Proc. of the *International Conference on Linguistic Applications*, 26-28 July 1994, UTMK-USM, Penang, Malaysia. pp.178-191.

Eagles 1993. "EAGLES Lexicon Architecture" EAGLES Document EAG-CLWG-LEXARCH/B. (http://www.ilc.pi.cnr.it/EAGLES96/lexarch/lexarch.html)

EDR. 1990. Proceedings of the *International Workshop on Electronic Dictionaries*, November 8-9 1990, Oiso, Japan. EDR Technical Report TR-031.

David Farwell, Louise Guthrie and Yorick Wilks. 1993. "Automatically creating lexical entries for ULTRA, a multilingual MT system". *Machine Translation,* 8(3), pp 127-145.

Kevin Knight and Steve K. Luk. 1994. "Building a large-scale knowledge base for machine translation". Proc. of the *12th National Conference on Artificial Intelligence*, AAAI'94.

Kavi Mahesh. 1996. "Ontology Development for Machine Translation: Ideology and Methodology". *Memorandum in Computer and Cognitive Science,* MCCS-96-292, Computing Research Laboratory, New-Mexico State University, Las Cruces, NM.

I. Meyer, B. Onyshkevych and L. Carlson. 1990. "Lexicographic principles and design for knowledge-based machine translation". *Technical report CMT-CMU-90-118,* Carnegie Mellon University, August 13, 1990.

Sergei Nirenburg. 1994. "Lexicon Acquisition for NLP: A Consumer Report". In B.T.S Atkins and A. Zampolli (eds.), *Computational Approaches to the Lexicon.* Clarendon Press, Oxford, UK. pp313-347.

Boyan Onyshkevych and Sergei Nirenburg. 1994. *The lexicon in the scheme of KBMT things.* Memoranda in Computer and Cognitive Science, MCCS-94-277. Las Cruces, N.M.: New Mexico State University. Reprinted as: A lexicon for knowledge-based machine translation, in: Dorr and Klavans 1995 (eds), 5-57.

Antonio Sanfilippo, and Victor Poznanski. 1992. "The acquisition of lexical knowledge from combined machine-readable dictionaries". Proc. of the *3rd Conference on Applied Natural Language Processing*, 31 March - 3 April 1992, Trento, Italy. pp80-87.

Gilles Serasset. 1994. "Interlingual lexical organization for multilingual lexical databases in Nadia". Proc. of the *15th International Conference on Computational Linguistics* - COLING'94, August 5-9 1994, Kyoto, Japan. pp278-282.

Gilles Sérasset. 1994. "Sublim: un système universel de base lexicales multilingues et Nadia: sa spécialisation aux bases lexicales interlingues par acceptions". Ph.D. Dissertation, December 1994, Université Joseph Fourier, Grenoble, France.

C.M. Sperberg-McQueen, Lou Burnard (eds.). 1992. *Guidelines for Electronic Text Encoding and Interchange*, TEI P2, Chap. 12. Text Encoding Initiative, Chicago, Oxford.

Komati Tankage, and Kyoji Umemura. 1994. "Construction of a bilingual dictionary intermediated by a third language". Proc. of the *15th International Conference on Computational Linguistics* - COLING'94, August 5-9 1994, Kyoto, Japan. pp297-303.

Jean Veronis and Nancy Ide. 1992. "A feature-based model for lexical databases". Proc. of the *14th International Conference on Computational Linguistics* - COLING'92, August 23-28 1992, Nantes, France. pp588-594.

Rémi Zajac. 1990. "A Relational Approach to Translation". *Proceedings of the 3rd International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages*, June 1990, Austin, TX, USA.

Rémi Zajac. 1992. "Inheritance and Constraint-Based Grammar Formalisms". *Computational Linguistics* 18/2, Special Issue on Inheritance, June 1992.

Rémi Zajac, Evelyne Viegas and Svetlana Sheremetyeva. 1998. "The Generic Structure of a Lexical Knowledge Base Entry". Ms. Computing Research Laboratory, New Mexico State University.