On Some Aspects of Lexical Standardization

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

In developing and using many large multi-lingual multi-purpose lexicons at CRL, we identified three distinct problem areas (1) an appropriate meta-language (formalism) for representing and processing lexical knowledge (2) a standard generic lexical framework defining a common lexical entry structure (names of features and types of content), and (3) shared universal linguistic types In this paper, we present the solutions developed at CRL addressing dimensions 1 and 2, and we mention the on-going research addressing dimension 3

1 Introduction

We envisage the standardization of lexical resources as a three dimensional process In developing, processing and using large multi-lingual and multi-purpose lexicons, a first set of difficulties lies in the lack of a standard format that is flexible enough to cover many different languages and applications, but sufficiently rigid to enable the use of a single lexical toolset shared across all these languages and applications A standard formalism for encoding lexical knowledge enables the construction of a generic lexical toolset SGML has been used for example for printed dictionaries For computational dictionaries, a good alternative are feature structures (Véronis & Ide 92, Ide & Véronis 95) The second set of problems is almost as acute as the first it is very difficult to design a sound lexical architecture, list the all the features that must be present for a variety of NLP applications, predict the interaction between the various sub-structures, and predict the needs of the various NLP tools that would be accessing the dictionary A standard lexical entry structure which defines the various features and provide guidelines to fill these features is a must for dictionary builders This level has been addressed for example in the Eagles program (Eagles 93) where it is sometimes mixed with the third dimension Finally, the problem of linguistic standards per se is addressed only partially by the definitions of guidelines and the use of a standard lexical entry structure In a multilingual setting, it is probably possible to define multilingual types, such as a standard list of part-of-speech However, this direction is still very much a research area related to the quest for a universal grammar (see e g Cahill & Gazdar 95, 96) Current standardization efforts such as Eagles define standards for content for particular languages only

In Section 2, we present a generic lexical architecture that addresses point one the generic structure of lexical entries and dictionaries, notions of lexical schema and meta-schema, and the generic lexical toolset Section 3 presents the standard structure of lexical entries that is used in structuring a number of computational dictionaries at CRL The standard structure is layered so that a particular dictionary could implement a sub-set of the layers only, while still implementing the standard Furthermore, the structure is flexible enough so that a given layer can be extended (by adding new elements through an inheritance mechanism) for a particular language, but forbids the redefinition of the lexical meta-structure Section 4 mention open problems and on-going research on the topic of a universal lexis and a parameter-based approach to the acquisition of a lexical profile

2 A Generic Lexical Architecture

To support the development of large lexicon, we implemented a Lexical Knowledge Base (LKB) called Habanera (Zajac 97) A Habanera LKB is composed of (1) several monolingual dictionaries, (2) translation relations linking these monolingual dictionaries, and (3) a multilingual dictionary schema that defines a shared multilingual inheritance hierarchy of lexical types for all monolingual dictionaries

The system supports a variety of linguistic architectures Since the design of a lexical architecture is a complex task, flexibility in designing the structure of the LKB 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 any kind of lexical data

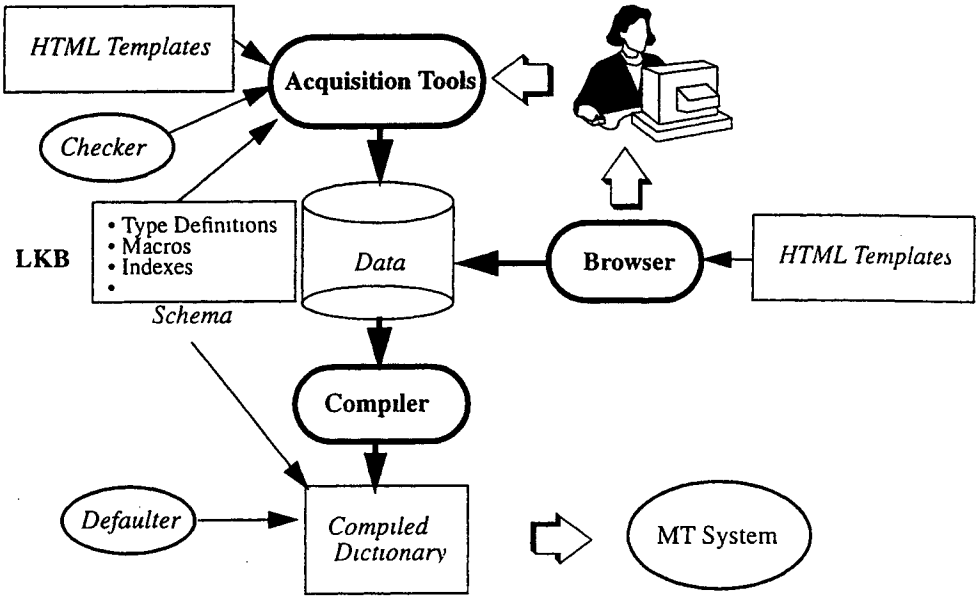

Figure 1 Habanera Architecture

These requirements motivated various initial choices for the design and the implementation of the system We use the Text Encoding Initiative (TEI) definition for printed dictionaries (Sperberg-McQueen & Burnard 94, Chap 12) as a source of inspiration for the definition of a standard dictionary entry structure (definition of the 'meta-schema' in Eagles terminology) However, lexical entries are encoded as Typed Feature Structures (TFS) which is our primary descriptive device for encoding lexical data Typed Feature Structures provide a declarative formalism with a well-defined formal semantics (and associated operations unification and subsumption) which we use instead of SGML to encode lexical entries A set of type definitions specifies what constitute valid lexical entries and play a role similar to a DTD in SGML A type definition specifies the set of features and restrictions on values for types Most of the lexical tools are parametrized by the type definitions which are part of the LKB schema

Multilingual dictionaries are organized as a set of monolingual dictionaries plus translation relations between entries In the case of Knowledge-Based Machine Translation, relations are also defined between word senses and ontological concepts Dictionaries and lexical entries are stored in a a commercial DBMS which allow concurrent access to a dictionary, an important consideration when a dictionary is developed by a team of lexicographers In the database, the format of stored data is independent of the external representation formalism All strings are encoded using Unicode and we use UTF-8 for file exchange (import/export functions)

The system is designed to facilitate acquisition as well as exploitation of lexical resources Acquisition tools are implemented using HTML forms for the acquisition interfaces and additional integrated utilities for checking the correctness of entries, for transcriptions, etc These tools are parametrized by resources (e g , HTML templates, grammars for transcriptions) that are loaded at runtime A dictionary can be accessed interactively through an HTML browser (also parametrized by a set of HTML templates) Natural Language Processing tools such as parser do not access the database Instead, a dictionary is compiled in a compact binary format that allows fast runtime access to entries The dictionary compiler can build several indexes to look-up entries in the compiled dictionary Runtime indexes are compressed tries that provide random access to a compact binary dictionary file

2.1 Dictionaries

The linguist works with a source dictionary where each dictionary entry is structured as a set of sub-entries An entry can for example group together senses for the same lemma, different categories together for the same form, different lemmas in the same derivational family, etc An entry has a unique key (a Unicode string) and a tree of sub-entries At each node of the tree, we attach a feature structure which encodes lexical information The feature structure must follow

the type definitions specified in the dictionary schema The tree of sub-entries defines an inheritance hierarchy Logically, only the leaves are actual entries the compiler traverses the tree of sub-entries, computing inheritance, and generating the compiled dictionary from the set of leaves

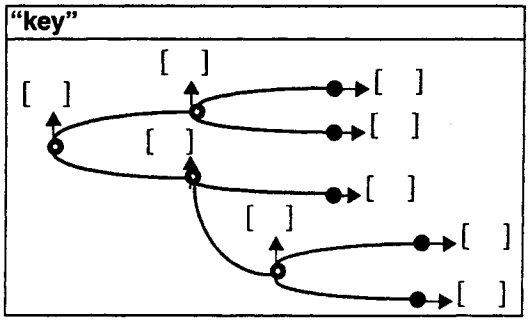

Figure 2 A lexical entry as a tree of feature structures

The dictionary schema contains various information useful for managing the dictionary (1) The schema of entries is specified using Typed Feature Structure definitions (2) The schema of relations among entries, if any A relation must specialize the pre-defined `Relation` type and relations are used to describe synonymy, hyperonymy, etc They are also used to link several monolingual dictionaries to provide translations (3) The set of macros, defining abbreviations for complex feature structures (4) The location of the key in the entry which is used to build the primary dictionary index (each entry has a unique key within a dictionary) (5) The language (as a 3-letter ISO code) (6) Additional indexes that are maintained by the database engine for interactive look-up of entries These indexes are specified as a set of paths in an entry (7) The name of the checker class and of the checker defaulter class

We use (typed) feature structures to model entries and relations (Zajac 98, 92) Each type has a definition, is similar to a class definition in an Object Oriented language the definition of a type specifies what are the allowed features for that type and what is the type of the value for each feature Types are used to define the structure of entries, of relations (links), and of lexical rules Since types can be organized in an inheritance hierarchy, it is possible to define a common framework for describing all dictionaries by defining a cross-language type hierarchy This multilingual type hierarchy specifies dictionary-dependent (that is, language-dependent) elements such as the inventory of morphosyntactic categories by defining super-types that are common to two or more languages, thereby defining a multilingual inheritance hierarchy of lexical types

Only syntactically correct entries are stored in the database However, there are some consistency checks which escape the checking done by the parser as well as the type-checking mechanism provided by the Typed Feature Structure engine For example, all headwords must be written using the alphabet of the language and other characters would not be allowed This kind of checks must be added specifically for each dictionary through the implementation of a checker class that is used by the database before adding entries in a dictionary

An optional defaulter can also be provided for a given dictionary the defaulter analyzes a dictionary entry and applies default rules to fill in missing information For example, if a feature `number` with value `Plural` is filled for a noun, the noun is an irregular plural, otherwise, it is a regular noun and the number feature is not further specified, or, if the dictionary specifies a gender only for feminine nouns, the defaulter might add a masculine gender when it is not specified Entries in the database might have such missing information However, our Typed Feature Structure engine does not provide defaults and a runtime dictionary must include explicitly all the defaults the defaulter is used by the compiler to fill in default information and produce a compiled dictionary where all information is explicitly expanded The compilation process is done as follows on each entry (1) Apply dictionary-specific checks using the checker class (if defined) (2) Apply the defaulter to augment the dictionary entry and solve all the defaults Note that the checker and the defaulter work on the tree of sub-entries, not on individual feature structures (3) Move all information down to the leaves of the tree of sub-entries (compute inheritance) (4) Expand macro definitions (5) Compile a feature structure for each leaf of the sense tree (7) Use type inference to infer the most specific type for each sub-feature structure within a feature structure (8) Type check the feature structures in a feature structure, expand the types of all sub-feature structures by unifying in the definition of the type

Relationships between lexical entries are modeled using binary links (relations), used to describe synonymy relations, derivations relations, translation relations (see Section 1 4), thesaurus relations, etc Any relation defined in the dictionary schema must inherit from the `Relation` type Relations can be given an arbitrarily complex internal structure and can bear information A relation is formally defined as

```
Relation = [dom   Entry,
            range  Entry],
```

For example, in a relation that specifies a cross-reference defined freely by the lexicographer, the domain feature will point to the entry which is the source of the relation and the target entry (range feature) will be identified by providing the key of that entry as in

```
#0=[key  "arm",
    xref  [dom  #0,
                range  [key  "armament"],
                note  "Collective for arm "]]
```

A dictionary browser could interpret these relations by generating hyperlinks between entries for example A dictionary also contains rules which specify productive relations within an entry (see Section 1 3) or among entries within multiple dictionaries or still within a single dictionary (see Section 1 4) The type Relation is used in the definition of translation relations, transfer rules and lexical rules each of these rules are defined as sub-types of `Relation`

2.2 Schema and meta-schema

The Eagles guidelines on standardization of lexical resources (Eagles 93) introduce the distinction between (1) "The meta-schema which defines general well-formdness conditions for the schema", (2) The schema "defines the logical format of language-specific and level-wise linguistic descriptions", and (3) "Instances are the individual lexicons for which there is a translation relation expressed between the individual format of the instance and the 'type' defined by the schema"

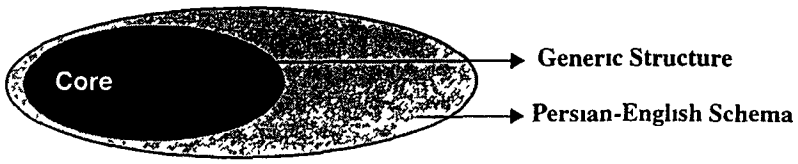

Figure 3 A specific dictionary schema, e g a Persian-English dictionary, specializes the generic schema, which is itself built on a hard-coded core lexical structure

In an Habanera lexical knowledge base, the only fixed structure is the tree of sub-entries, and anything else is defined via the dictionary schema Using the Typed Feature Structure language developed at CRL, it is possible to define dictionary schemata using several layers of abstractions, therefore introducing arbitrary intermediate layers between the meta-schema and the schema proper In this TFS language, sets of type definitions are grouped into modules and sub-modules (a notion similar to the notion of package in programming languages such as Lisp or Java) The use of modules allows to structure a schema as a set of modules introducing additional structures and more specific constraints on the format of an instance In the next section, we will present the lexical structure which is used in CRL dictionaries The schemata of dictionaries are organized as follows A generic module defines the generic structure of a dictionary Language specific modules add to that specification language dependent information (e g a specific inventory of morphosyntactic features) of that is grafted on the generic structure or which specializes the generic structure The generic structure has been inspired by the TEI definition and in presented in Section 3

The set of type definitions specified in the dictionary schema is used by the type-checker which checks that a dictionary entry is well-typed and by the compiler which builds a compact binary representation of a dictionary entry as a feature structure

2.3 Tools

The dictionary browser and editor are parametrized by a set of HTML templates which define the presentation format to be used for displaying feature structures at each level of the tree of sub-entries The mapping of the structure of an entry

to an HTML template relies on naming conventions based on the value of paths to name HTML elements in an HTML document

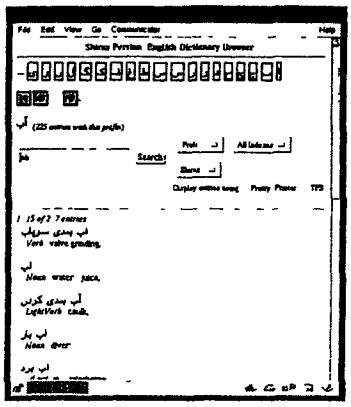

Figure 4 A Habanera Browser for a Persian-English dictionary

Since most Web browsers do not support input methods for languages other than English, input of character strings is done using a transcription A set of transcription tables can be defined by the user and selected in the browser when inputting some character string for e g headwords However, Web browsers support the display of almost any major language[1] and Unicode strings can be directly embedded in HTML documents Habanera also provide import/export functions The format of a dictionary file uses a textual syntax for feature structure (the one used in the examples) The dictionary file encoding is UTF-8

3 Standardizing the Structure of Lexical Entries

The dictionaries developed at CRL shared the same generic structure Each language specific dictionary refines the shared schema by adding language specific information (e g , a specific inventory of morphosyntactic features) The data of a monolingual dictionary is a set of entries corresponding to word senses as described in (Meyer et al . 1990) and (Onyshkevych and Nirenburg, 1994) We distinguish between computational features that are used by NLP components such as parsers (`form`, `gram`, `sem`, `synSem`, `trans`, `rel`, `lexRule`, `usg`),[2] and other features that are used by lexicographers definition (`def`), example (`eg`), etymology (`etym`), cross-reference (`xref`) and note (`note`) The features present for each sub-entry are

```
EntryElements = [
        form  Form,
        gram  Grammar,
        sem  TMR,
        synSem  SynSemMap,
        trans  Translations,
        rel  LexicalRelations,
        lexRule  LexicalRules,
        usg  Usage,
        def  String,
        eg  Example,
        etym  String,
        xref  Xref,
        note  String],
```

The computational features used by NLP components are the following

1 `form` information related to the orthographic form of the word and its morphology (includes morphological features and morphological variants),

2 `gram` information related to the syntactic behavior of the word (includes POS and subcategorization information),

1 With the important exception of Arabic-based scripts

2 The names of most features are taken from the TEI specification

3 `trans` a cross-reference to one or more entries in a target dictionary,

4 `sem` semantic mapping to a conceptual structure,

5 `synSem` information on syntax-semantic linking,

6 `rel` information on paradigmatic (synonyms, antonyms,) and syntagmatic (collocations, co-occurrences,) relations,

7 `lexRule` specification of productive lexical relations among entries within a dictionary (e g , productive morphological derivations),

8 `usg` restrictions on the usage of some word (domain, geographical, temporal,)

In the remainder of this section, we present the structure of the `form` and `gram` features (see Zajac et als 98 for a description of other features)

3.1 Orthography and Morphology

The `form` feature records information about the type of word whether the word is a full word, and acronym, or an abbreviation These types are introduced since typically acronyms and abbreviations are processed differently from ordinary words, for example during a tokenization phase (see e g Grefenstette 94) and words or compounds are processed during or after a morphological analysis the dictionary compiler will produce different runtime dictionaries that include different kinds of information as needed by the various components of the system

The orthography feature records the citation form of the word as well as a list of variants There could also be additional information such as capitalization, hyphenation or syllabification (a useful information for an English morphological analyzer for example)

The morphology records three different kinds of information morphological information that is attached to the word and stored in the lexicon (e g , gender information), inflectional information that is typically computed by a morphological analyzer (and passed to the syntactic analyzer), and derivational information that could be either pre-computed in the lexicon or dynamically computed by a morphological analyzer In our lexical model, we require that each dictionary includes as lexical morphological information the part-of-speech (using the pos feature) and the indication if the word has a regular morphology or not (using the Boolean `regular` feature)

Irregular forms are listed in the dictionary if the value of the `regular` feature is `False` This feature is provided to handle simple cases where a given class of words has only one inflectional paradigm English noun for example can be defined as having only one paradigm for the number inflection, where phonological variants are handled by the morphological processor and anything that falls out of the domain of the morphological processor will be treated as an irregular form Note that the dictionary schema must allow for the inclusion of all inflected forms for irregulars

If the linguist has to define inflectional paradigms, as it is the case in many languages, these paradigms must also be specified in the dictionary schema and should allow for the specification of various stems involved For example, one might consider that English verbs have two paradigms, one where all forms are derived from the citations form (want, wants, wanted, wanted, wanting) modulo phonological changes, one class where some forms must be specified in the lexicon (take, takes, took, taken, taking), and a class of irregulars (be, is, was, been, being) Therefore, English verbs could be classified as regular or irregular, and for regular, they fall in one of two paradigms The reader will have noticed that the morphological model used in the lexicon must be compatible with the model implemented by any morphological processor using the dictionary Our experience has shown that it is not always trivial to reconcile a morphological analyzer developed independently from a dictionary with the dictionary

The structure of the form feature must therefore include the following elements

```
[type  Full | Abbreviation | Acronym,
 orth  [cit  String,      // The citation form
        variants  List], // Optionally, syllabification, capitalization, etc
 morph  [lex  [pos  POS,
               regular  Boolean],
         infl  InflectionalFeatures, // Always unspecified in the dictionary
         deriv  DerivationalStructure]]
```

For example, the form structure of an English entry might look like

```
#0=[key  #k="bring",
    form orth exp  #k,
    sense  #1=[
        morph  [
            lex  [
                pos  eng Type MainVerb,
                regular  True,
                paradigm  1,
                simplePast  "brought",
                pastParticiple  "brought"]]]]
```

where the features for inflectional and derivation information are left unspecified

3.2 Syntax

The gram feature groups all information related to the syntactic behavior of the word The grammar feature gram contains as required features the part-of-speech information (feature `pos`) and the subcategorization frame (feature `frame`) The `frame` feature encodes the subcategorization frame of the predicate expressed as a list of phrasal types The grammar feature may include additional features such as the subcategory, for example Mass/Countable for nouns, or Intransitive/Transitive for verbs, although this is typically better represented by defining the appropriate sub-types for each part-of-speech Additionally, an inflectional feature `infl` is also defined for use by syntactic processors the value of this feature is shared with morphology During processing, a morphological analyzer will produce a set of inflectional features and make them available to syntax through the feature `gram infl` Conversely, a syntactic generator will produce a set of inflectional features for lexical heads and make them available to the morphological generator

The Grammar feature (path `gram` in an entry) has type `Gram` This type is defined as

```
Gram = [pos  POS,
        frame  List,
        infl  MorphInflection],
```

For example, the following (partial) entry specifies two subcategorization frames for the noun "announcement"

```
[key  "announcement",
 gram  [pos  N,
        sense gram frame  < NpComp[head  "that"] >,
        sense gram frame  < NpObl[head  "of"] > ]]
```

4 Conclusion

Standardizing lexicons represent an interesting intellectual and practical endeavor Past experience at CRL in developing, processing and using many large lexicons for several tasks, including machine-translation systems, machine-aided translation tools, and information processing systems shows that a first set of difficulties lies in the lack of a standard format that is flexible enough to cover many different languages and applications, but sufficiently rigid to enable the use of a single lexical toolset shared across all these languages and applications This problems have been addressed by developing a generic dictionary software architecture that is now use to manage several large dictionaries designed for machine-aided translation as well as for machine translation

The second set of problems is almost as acute as the first it is very difficult to start designing a sound lexical architecture from scratch, list the all the features that must be present for a variety of NLP applications, predict the interaction between the various sub-structures, and predict the needs of the various NLP tools that would be accessing the dictionary This has been done many times at CRL and this knowledge is in part incorporated in the generic standard lexical structure briefly presented in Section 3 When developing a new dictionary, the linguist must use a pre-defined dictionary entry structure and follow a set of guidelines for defining the language-specific features This guarantees that the dictionary can be developed and maintained using a standard dictionary management toolset, and that the information contained in the dictionary can actually be used for a variety of NLP applications which requirements are not always obvious for a non-expert The construction of such a standard lexical structure is still however an open task some areas are defined with more precision than others We started from a fairly unconstrained structure and cross-

language work brought out commonalities that have been progressively incorporated in the standard structure Although the standard structure presented in this paper has been now stable for over a year, further research and experimentation could yield new constraints that could be incorporated in the architecture

Finally, the problem of linguistic standards per se is addressed only partially by the definitions of guidelines and the use of a standard lexical entry structure In particular, a standard entry structure imposes a specific organization of the linguistic information encoded in an entry It defines the kind of linguistic information to be encoded and how to structure this information In a multilingual setting, it is probably possible to define multilingual types, such as a standard list of part-of-speech However, our experience on more than 6 different languages show that trying to establish a set of multilingual types is not worth the effort the use of a standard lexical structure allows the linguist to narrow down rapidly on the inventory of language-specific types which can then be listed with relative ease Standardization of lexical content is still a very much open problem, and this research area related to the quest for a universal grammar In the Boas project (Nirenburg & Raskin 98), the linguist defines language-specific properties using a knowledge elicitation system that contains knowledge about the set of possible linguistic parameters and values The linguist is guided through a set of queries and answers, the result of which is a linguistic profile of a language From this language profile, the goal is to generate automatically the set of language-specific lexical properties that define the schema of a dictionary

5 References

1 Lynn Cahill and Gerald Gazdar 1995 "Multilingual lexicons for related languages" In *Proceedings of the 2nd DTI Language Engineering Conference* pp169-176

2 Lynn Cahill and Gerald Gazdar 1996 "Multilingual Lexicons for Related Lexicons" In *Proceedings of AISB'96 Workshop on Multilinguality in the Lexicon*, Brighton, UK, 69-75

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

4 Gregory Grefenstette 1994 "What is a Word, What is a Sentence Problems of Tokenization" Rank Xerox Research Center, Technical Report MLTT-004, April 1994

5 Ide, N , Véronis, J (1995) "Encoding dictionaries" In Ide, N , Véronis, J (Eds) (1995) *The Text Encoding Initiative Background and Context* Kluwer Academic Publishers, Dordrecht, 167-179

6 Sperberg-McQueen, C M , Burnard, L 1994 *Guidelines for Electronic Text Encoding and Interchange*, Text Encoding Initiative, Chicago and Oxford, Chapter 12, "Print Dictionaries", 312-370
`http //etext virginia edu/TEI html`

7 Meyer, I , 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

8 Sergei Nirenburg and Victor Raskin 1998 'Universal Grammar and Lexis for Quick Ramp-Up of MT Systems" Proc of the *17th International Conference on Computational Linguistics* – COLING'98, 10-14 August 1998, Montreal, Canada pp975-979

9 Onyshkevych, Boyan, 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

10 Jean Véronis 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

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

12 Rémi Zajac 1997 "Habanera, a Multipurpose Multilingual Lexical Knowledge Base" NLPRS Workshop on Multilingual Information Processing, *Natural Language Processing Pacific Rim Symposium 1997*, 1-4 December, 1997, Phuket, Thailand

13 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