ELSNET 97 Summer School

Lexicon Development for Language and Speech Processing

DRAFT Course Materials

Computational Lexicography for Speech and Language

Dafydd Gibbon

Address: Universität Bielefeld
Fakultät für Linguistik & Literaturwissenschaft
Postfach 100131
D–33501 Bielefeld

Phone: +49 521 106 3510
Fax: +49 521 106 6008

Email: gibbon@spectrum.uni-bielefeld.de
Web: http://coral.lili.uni-bielefeld.de/

Contents

1 Aspects of lexicography

This collection of course materials borrows from various materials, partly published and partly unpublished course materials. It is heterogenous, uneven, and currently in very rough shape, but contains a variety of different kinds of material relevant to spoken language lexicography.

The collection starts with an overview of spoken language lexica and why resources are needed, following this with a review of types of lexical information at different levels of linguistic analysis and a chapter on lexical representation and inference. Two practical chapters cover elementary lexical databases and UNIX tools in and around lexicography.

1.1 System lexica

Both written and spoken language systems, separately and in integrated form such as in dictation software, are becoming increasingly versatile, and a central task in developing such a system is the collation of lexical information. Lexical information is required both as a means of characterising properties of words in a spoken language corpus in a lexical database or knowledge base, and for the development of practically all system components. In related areas such as natural language processing (NLP) and computational and theoretical linguistics, the lexicon has come to play an increasingly central role. The lexicon of a spoken language system may be designed for broad or narrow coverage, for specific applications, with a particular kind of organisation, and optimised for a specific strategy of lexical search. Since the construction of a lexicon is a highly labour-intensive and thus also error-prone job, a prime requirement is for formalising lexical representations and automating lexicon development as far as possible, and in re-using lexical resources from existing applications in new developments.

The main object of this chapter is to provide a framework for relating such concepts to each other and for the formulation of recommendations for development and use of lexica for spoken language systems. In this introductory section, some basic concepts connected with the use and structure of lexica in spoken language systems are outlined. In the following sections, specific dimensions of spoken language lexica are discussed in more detail. Particular attention is paid to lexical properties related to *inflectional morphology*, which is far more important for many other languages than it is for English, and other aspects of morphology which are important for the treatment of out-of-vocabulary words. Discussion is restricted to spoken language lexica as system development resources; non-electronic lexica for human use (e.g. pronunciation dictionaries in book form) are not considered. Features common to spoken and written language lexica, such as syntactic and semantic information in lexical entries, are only mentioned in passing; see the report of the EAGLES Working Group on Computational Lexica on these points. The close relation between spoken language lexica and speech corpora results in overlap with the Spoken Language Corpus chapter of this handbook.

The following sections of the chapter are concerned with basic features of spoken language lexica, types of lexical information, lexicon structure, lexical access, and lexical knowledge acquisition for spoken language lexica.

1.2 Lexical resources

At the present time, information about lexica for spoken language systems is relatively hard to come by. One reason for this is that such information is largely contained in specifications of particular proprietary or prototype systems and in technical reports with restricted distribution. With the advent of organisations for coordinating the use of language resources, such as ELRA (the *European Language Resources Association*) and the LDC (the *Linguistic Data Consortium*), access to information on spoken language lexica is becoming more widely available.

Another reason for difficulties in obtaining information about spoken language lexica is that there is not a close relation between concepts and terminology in the speech processing field on the one hand, and concepts and terminology in traditional lexicography on the other. natural language processing and computational linguistics. Components such as Hidden Markov Models for word recognition, stochastic language models for word sequence patterns, grapheme-phoneme tables and rules, word-oriented knowledge bases for semantic interpretation or text construction are all concerned with the the identity and properties of words, lexical access, lexical disambiguation, lexicon architecture and lexical representation, but these relations are not immediately obvious within the specific context of speech technology. Stochastic word models, for instance, would not generally be regarded as a variety of lexicon they evidently do provide corpus-based lexical information about word collocations.

A terminological problem should be noted at the outset: in the spoken language technologies, the term *linguistic* is often used for the representation and processing in sentence, text and dialogue level components, and *acoustic* for word models. With present-day systems, this terminology is misleading. The integration of prosody, for example, requires the interfacing of acoustic techniques at sentence, text and dialogue levels, and linguistic analysis is involved at the word level for the specification of of morphological components in systems developed for highly inflecting languages or for the recognition of out-of-vocabulary words, or for using phonological information in structured Hidden Markov Models (HMMs).

It is useful to distinguish between system lexica and lexical databases. The distinction may, in specific cases, be blurred, and the unity of the two concepts may also be rather loose if the system lexicon is highly modular, or distributed among several system components, or if several different lexical databases are used. However, the distinction is a useful one. The distinction between lexica and lexical databases will be discussed below. Since the kinds of information in both these types of lexical object overlap, the term "spoken language lexicon" will generally be used in this chapter to cover both types. The following overview is necessarily selective.

Lexica for spoken language are used in a variety of systems, including the following:

- Automatic spelling correctors (spelling is determined to a large extent by phonological considerations).
- Medium and large-vocabulary automatic speech recognition (ASR), as in systems such as SPICOS (cf. Höge et al. 1985; Dreckschmidt 1987; Ney et al. 1988; Thurmair 1986), HEARSAY-II (cf. Lesser et al. 1975; Erman 1977; Erman and Lesser 1980; Erman and Hayes-Roth 1981), SPHINX (cf. Lee et al. 1990), ISADORA (cf. Schukat-Talamazzini 1993), or, for example in automatic dictation machines such as IBM's TANGORA (cf. Averbuch et al. 1986, 1987; Jelinek 1985) and DragonDictate by Dragon Systems (cf. Baker 1975a,b, 1989; Baker et al. 1992).
- Speech synthesis in text-to-speech systems, for example in reading machines, speaking clocks. For further speech synthesis applications, various relevant studies such as Allen et al. (1987), Bailly and Benoît (1992), Bailly (1994), Van Coile (1989), Klatt (1982, 1987), Hertz et al. (1985), Van Hemert et al. (1987) can be consulted.
- Interactive dialogue systems, with speech front ends to databases and enquiry systems and synthesised responses (see for instance Brietzmann et al. 1983; Niemann et al. 1985, 1992; Bunt et al. 1985).
- Speech-to-speech translation systems as developed in the ATR and VERBMOBIL projects, which use various speech recognition techniques, including continuous speech recognition, recognition of new words, word spotting in continuous speech. For speech translation systems see for instance Rayner et al. (1993) and Woszczyna et al. (1993).
- Lexica and encyclopaedias on CD-ROM with multimedia (including acoustic) output.
- Research and development of spoken language processing systems, in the process of which broader based lexica for written language, coupled with tools such as grapheme-phoneme converters, may be used as sources of information.

Spoken language lexica may be components of systems such as those listed above, or reusable background resources. System lexica are generally only of local interest within institutes, companies or projects. Lexical databases as reusable background resources which are intended to be more generally available raise questions of standardised representation, storage and dissemination. In general, the same principles apply as for Spoken Language Corpora: they are collated, stored and disseminated using a variety of media. In research and development contexts, magnetic media (disk or tape) were preferred until recently; in recent years, local magnetic storage and wider informal dissemination within projects or other relevant communities is conducted via the Internet using standard file transfer protocols, electronic mail and World-Wide Web search and access. Large lexica, and corpora on which large lexica are based, are also stored and disseminated in the form of ISO standard CD-ROMs.

The following brief overview can do no more than list a number of examples of current work on spoken language lexicography. At this stage, no claim to exhaustiveness is made, and no valuation of cited or uncited work is intended.

- A number of general lexica with information relevant to spoken language have already been available on CD-ROM for quite some time, including the Hachette and Robert (9 volume) dictionaries for French, the Oxford English Dictionary, the Duden dictionary for German, and the Franklin Computer Corporation Master 4000 dictionary with acoustic output for 83000 words (cf. Goorfin 1989).

- Several lexica with more restricted circulation have been developed in the context of speech technology research and development. Companies such as IBM, and telecom research and development institutes such as CNET in France have developed large lexica (CNET, for instance, has a 55000 word and 12000 phrase lexicon).
- University and other research institutes have also constructed large lexica; in France, for example, such institutes as ENST in Paris, ICP in Grenoble (cf. Tubach and Bok 1985), Paris (cf. Plenat 1991), for a pronunciation dictionary of abbreviations) and IRIT in Toulouse (the BDLEX project) have worked on large spoken language lexica. The BDLEX-1 lexicon coordinated by IRIT (cf. Pérennou and De Calmès 1987) contains 23000 entries, and BDLEX-2 (cf. Pérennou et al. 1991, 1992; Pérennou and Tihoni 1992) contains 50000 entries; a set of linguistic software tools permits the construction of a variety of daughter lexica for spelling correction and lemmatisation, and defines a total of 270000 fully inflected forms.
- The Belgian BRULEX psycholinguistic lexicon contains information on uniqueness points (the point in a letter tree where a word form is uniquely identified), lexical fields, phonological patterns and mean digram frequencies for 36000 words (cf. Content et al. 1990).
- In the United Kingdom, the Alvey project resulted in many tools and lexical materials (cf. Boguraev et al. 1988).
- In the Netherlands, the Nijmegen lexical database CELEX (cf. Baayen 1991), also available on CD-ROM, contains components with 400000 Dutch forms, 15000 English forms and 51000 German forms, together with an access tool FLEX.
- For German, lexical databases for spoken language lexica have been constructed by companies such as Siemens, Daimler-Benz, IBM and Philips, as well as in university speech technology departments (e.g. Munich, Erlangen, Karlsruhe, Bielefeld), and in the VERBMOBIL project (Gibbon 1995; Gibbon and Ehrlich 1995); these have been made available on the World-Wide Web with interactive form interfaces.
- Work in computational lexicology and computational phonology has led to the development of structured lexicon concepts for spoken language such as ILEX (cf. Gibbon 1992a; Bleiching 1992a) based on the DATR lexical knowledge representation language (cf. Evans and Gazdar 1989, 1990); the DATR language has been applied to word form lexica in the multilingual SUNDIAL project (cf. Andry et al. 1992) by the German partner Daimler-Benz and in the German VERBMOBIL project (cf. Gibbon 1993).
- The European Commission has funded a number of projects, particularly within the ESPRIT programme, in which questions of multilingual spoken language system lexica have been addressed, albeit relatively indirectly (POLYGLOT, SUNDIAL, SAM, SAM-A), as well as lexicography projects such as MULTILEX in the ESPRIT programme (cf. Heyer et al. 1991), GENELEX in the EUREKA programme (cf. Nossin 1991) and ACQUILEX, which concentrate on multi-functional written language lexica, though extension of the results to spoken language information has been provided for by the adoption of general sign-based lexicon architectures (see the results of the EAGLES Working Group on Computational Lexica).

1.3 Spoken language lexica

A spoken language lexicon may be a component in a system, a *system lexicon*, or a background resource for wider use, a *lexical database*, in each case containing information about the pronunciation, the spelling, the syntactic usage, the meaning and specific pragmatic properties of words; lexica containing subsets of this information may also be referred to as spoken language lexica, though the simpler cases are often simply referred to as wordlists. Where there is little danger of confusion, the term *spoken language lexicon* will be used to refer indifferently to either a *spoken language system lexicon* or a *spoken language lexical database*. A lexical databse may be general purpose, or orientated towards specific tasks such as speech recognition or speech synthesis, and restricted to a specific scenario. For system development and evaluation it is generally critical to define an agreed *word-list* with a well-defined notion of *word* (e.g. a fully inflected word form), and an associated complete and consistent pronunciation dictionary for grapheme-phoneme conversion and language model construction.

A spoken language lexicon is defined as a list of representations of lexical entries consisting of spoken word forms paired with their other lexical properties such as spelling, pronunciation, part of speech (POS), meaning and usage information, in such a way as to optimise lookup of any or all of these properties. This definition covers a wide range of specific types of spoken language lexicon, . At the one end of the spectrum are lists in which orthography provides a more or less indirect representation of a spoken word form pronunciation augmented by tabular pronunciation dictionaries and conversion rules. At the other end are declarative knowledge bases with attribute-value matrix representation formalisms and inheritance hierarchies with associated inference machines, by means of which details of lexical information are inferred from specific premises (entries) about individual lexical items and general premises (rules) about the structure of lexical items. Between these extremes are optimised representations, and other application directed special lexicon types based, for instance, on the different requirements for pronunciation tables for speech recognisers and for speech synthesisers.

Both in speech recognition and in speech synthesis, the different kinds of spoken language lexicon are generally orientated towards the *forms* of words rather than towards their *distribution* in larger text or utterance units, or their *meaning* and *use* in context. Furthermore, where possible closed sets of fully inflected words which are actually attested in corpora are preferred to the construction of words on morphological principles, though rule-based word construction is increasing in importance in projects concerned with highly inflecting languages or aimed at the recognition of spontaneous continuous speech in which out-of-vocabulary words or ad hoc coinages (nonce forms) are encountered. In addition to out-of-vocabulary words, systematic noise events may also require inventarisation in a lexical database.

1.4 Lexical databases and system lexica

The distinction between lexical databases and system lexica is a useful one, though in practice more complex distinctions are required. The main characteristics of the two kinds of lexical object are outlined below.

LEXICAL DATABASE: A spoken language lexical database is often a set of loosely related simpler databases (e.g. pronunciation table, index into a signal annotation file database, stochastic word model, linguistic lexical database with syntactic and semantic information).

- Purpose:
  - Resource for system development (training, evaluation; construction of stochastic language models).
  - Definition of vocabulary coverage.
  - Basis for vocabulary consistency maintenance.
  - Reference point for integrating different kinds of lexical information.
  - Source of information for investigation of vocabulary structure.
- Structure:
  - Generally fixed record structures, with fields for different types of lexical information, and strings as values in fields.
  - Often identification of *lexical key (lexical identifier)* with *orthographic word form*. A problem with orthographic keys, particularly with large vocabularies: is the existence of homographs, i.e. lexical items wth the same spelling but different pronunciation (heterophonous homographs) and/or meaning, a potential source of "orthographic noise". Additional serial numbering may be used to distinguish between homographs.
  - Alternative for larger databases with more complex linguistic information: Unique identification of word as a more abstract unit with a formal identifier and specific properties including orthography, pronunciation, syntax (POS), semantics, etc. on an equal footing.
  - Implementation generally conforming to local laboratory standards as a database of ASCII strings, created and accessed by means of standard UNIX tools and UNIX shell scripts, or C programmes; in more complex environments with a commercial database such as ORACLE; occasionally as knowledge bases in higher-level languages such as Prolog or specialised languages such as DATR.
- Content:
  - Main lookup key (in general an orthographic representation, perhaps supplemented by numbering to distinguish homographs).
  - Database entries may be *fully inflected forms*, *uninflected stems*, or *morphemes* (generally *morphs*, i.e. the phonemic forms of morphemes), or all of these; other inventories containing units such as phonemes, diphones or syllables, may be included.
  - Pronunciation (in canonical phonemic representation, perhaps including pronunciation variants.
  - Subword boundaries between units such as syllables, morphs (phonemic forms of affixes, lexical roots), derived stems and constituents of compound words.
  - Syntactic category (part of speech, POS, e.g. Noun, Adjective, Article, Pronoun, Verb, Adverb, Preposition, Conjunction, Interjection) or subcategory (e.g. Proper vs. Common Noun, Intransitive vs. Transitive vs. Ditransitive vs. Prepositional, etc., Verb).
  - Semantic categories (in general scenario-specific, i.e. restricted to a given domain or application).
  - Corpus information: frequency statistics (of varying complexity, up to sophisticated language models; concordance information (i.e. list of contexts of occurrence for each word, usually generated on demand); signal annotations.

  - Further information: concordance (textual context), links to speech files.
  - Implementation:
    - commercial relational or object-oriented database,
    - UNIX ASCII database core with access by UNIX script languages, C or C++ programmes,
    - in-house custom databases or knowledge bases.

SYSTEM LEXICON: Lexical information (i.e. properties of words) referred to during the speech recognition or synthesis process may not be concentrated in one identifiable lexicon in a given system.

- Purpose: Definition of those properties of words required for recognition, parsing and understanding, or for planning, formulation and synthesis.
- Structure: In general separate modules for different properties of words with different functions within the system (which are often not regarded as having anything at all to do with a lexicon)
  - In speech recognition: Modules such as the word recogniser (typically based on *Hidden Markov Model* technology), which identifies word forms, i.e. recognition oriented lexical access keys, often phoneme strings derived from orthographic keys and a pronunciation dictionary, the stochastic language model (which defines statistical properties of words in their immediate contexts as *bigrams*, *trigrams*, etc.), and the linguistic lexicon with syntactic and semantic information, linked to an application-specific database or knowledge base.
  - In speech synthesis: Modules which map orthographic forms (in text-to-speech systems) or conceptual or semantic representations (in concept-to-speech systems) to word structures in terms of morpheme sequences, word prosody (e.g. accentuation), and pronunciation (in terms of phonemes), supplemented by detailed rules for phoneme variants in different contexts and for timing and other relevant parametric information.
- Content: Application specific; subsets of information defined in the lexical database resource, as outlined under “Structure”.

1.5 Coverage of lexica

Spoken language lexica differ in coverage and content in many respects from lexica for written language, although they also share much information with them. Written language lexica are generally based on a stem, neutral or canonical morphological form (e.g. nominative singular; infinitive), or headword concept, in which generalisations over morphologically related forms may be included. This principle leads to fairly compact representations. Spoken language lexica for speech recognition are generally based on fully inflected word forms, as in dictation systems with about 20000 entries. Depending on the complexity of inflectional morphology in the language concerned, the number of fully inflected word form entries is larger than the number of regularly inflectable entries in a dictionary based on stems or neutral forms by a factor from 2 or 3 to several thousand, depending on the typology of the language concerned. Speech synthesis systems for text-to-speech applications do not rely exclusively on extensive lexica, but also use rule-based techniques for generating pronunciation forms and prosody (speech melody) from smaller basic units.

An orthographically oriented lexicon generally includes a canonical phonemic transcription, based on the citation form of a word (the pronunciation of a word in isolation) which can be utilised, for example, in sophisticated tools for automatic spelling correction or “phonetic search” in name databases. However, this is not always adequate for the requirements of speech recognition systems, in which further details are required.

A spoken language lexicon may also contain information about pronunciation variants, and often includes prosodic information about syllable structure, stress, and (in tone and pitch accent languages) about lexical tone and pitch accent, with morphological information about division into stems and affixes. Spoken language lexica are in general much more heavily orientated towards properties of word *forms* than towards the *distributional* and *semantic* properties of words.

It may happen that a canonical morphological form or a canonical pronunciation does not actually occur in a given spoken language corpus; this would be of little consequence for a traditional dictionary, but in a spoken language dictionary it is necessary to adopt one of the following solutionsr for a discussion of solutions to the sparse data problem in language modelling):

1. Use the canonical phonemic form, but mark it as non-occurring; additionally, incorporate the attested form.
2. Adopt an attested form as canonical morphological form (e.g. nouns occurring only in the plural such as French *ténèbres* ‘darkness’, English *trousers*, German *Leute* ‘people’).

At a more detailed level, orthography (the division of word forms into standardised units of writing) and phonology (the division of word forms into units of pronunciation) are related in different ways in different languages both to each other and also to the morphology (the division of word forms into units of sense) of the language. The orthographic notion of “syllable” serves, in general, in written language lexica for defining hyphenation at line breaks and certain spelling rules (and may even refer to morphological prefixes and suffixes); for this purpose, morphological information about words is also generally required. In spoken language, however, the phonological notion of “syllable” is quite different; it refers to units of speech which are basic to the definition of the well-formed sound sequences of a language and to the rhythmic structure of speech, and forms the basis for the definition of variant pronunciations of speech sounds. Alphabetic orthography involves a close relation between characters and phonemes; in syllabic orthography (Japanese ‘Kana’) characters are closely related to phonological syllables; in logographic orthography (Chinese), characters are closely related to simplex words (cf. numerals in European languages: the spelling “7” is pronounced /zi:bən/, /sɛt/, /sɛvən/, and so on).

When complex word forms are put together from combinations of smaller units, different *alternations* of orthographic units (letters) often occur at the boundaries of the parts of such words (*telephone* + *y* = *telephony*; *lady* + *s* = *ladies*). Similarly, morphophonemic alternations occur in such positions (*wife* – /waɪf/ singular vs. *wives* – /waɪvz/ plural). Furthermore, additional kinds of lexical unit are required in the lexicon of a spoken language dialogue system: discourse particles, hesitation phenomena, pragmatic idioms, such as greetings, or so-called *functional units* (sequences of functional words which behave as a phonological unit: *n'est-ce pas*, /nɛspa/) and *clitics* (functional words which combine with lexical words to form a functional unit, cf. *I'm coming*, /aɪm kʌmɪŋ/).

Criteria for the coverage of lexica for spoken language processing systems are heavily corpus determined, and differ considerably from criteria for coverage of lexica for traditional computational linguistics and some areas of natural language processing. In theoretical computational linguistics, interests are determined by systematic fragments of natural languages which reveal interesting problems of representation and processing. In natural language processing, maximally broad coverage is often the goal. In spoken language lexica as currently used in speech technology, lexica are always oriented towards a particular well-defined corpus which has often been specifically constructed for the task in hand. When speech technology and natural language specialists meet, for instance in comprehensive dialogue oriented development projects, these differences of terminology and priorities are a potential source of misunderstanding and disagreement, and joint solutions need to be carefully negotiated.

The main coverage criteria for spoken language lexica may be summarised as follows.

- Completeness (all word types in the training corpus and test corpora).
- Minimality (only word types in the training and test corpora).
- Consistency (with respect to the training and test corpora and other related data types).
- Generality (projection of the word form set on to related forms not in the corpus).
- Informativity (the types of lexical information associated with lexical entries).

The first four criteria define *quantitative* or *extensional coverage*, the fifth defines *qualitative* or *intensional coverage* of the lexicon.

These criteria pertain to words; if other units, such as idioms, are involved, the criteria apply analogously to these.

The first three extensional criteria are essentials for the current state of speech technology. Conventional expectations in written language processing, i.e. in natural language processing and computational linguistics, are widely different, and are expressed in the fourth criterion. Clearly the second and fourth criteria clash; the relation to relevant corpora must therefore be carefully flagged in a spoken language lexicon. The degree of extentional coverage (which for a speech recognition system generally has to be 100 %) is sometimes expressed in terms of the notions of degree of static coverage (ratio of in a corpus which are contained in a given dictionary to the number of words in the corpus) and the degree of dynamic coverage or saturation (the probability of encountering words which have previously been encountered); the latter value is generally higher than the former (cf. Ferrané et al. 1992).

1.6 The lexicon in spoken language recognition systems

A spoken language recognition system is generally divided into two components: the recognition component and the search component. In the recognition component, intervals of the speech signal are mapped by probabilistic systems such as Hidden Markov Models, Neural Networks, Dynamic Programming algorithms, Fuzzy Logic knowledge bases, to word hypotheses; the resulting mapping is organised as a *word* *lattice* or *word graph*, i.e. a set of word hypotheses, each assigned in principle to a temporal interval in the speech signal. The term *word* is used here in the sense of “lexical lookup key”. The keys are traditionally represented by orthography, but would be better represented in a spoken language system by phonemic transcriptions. in order to avoid orthographic noise due to heterophonous homographs. The search component enhances the information from the speech signal with top-down information from a language model in order to narrow down the lexical search space. In spoken language recognition system development, a corpus based lexicon of orthographically transcribed forms is used as the basis for a pronunciation lexicon (pronunciation dictionary); the lexicon is often supplemented by rules for generating pronunciation variants due to informal speech styles (phonostylistics) or speaker and dialect variants. The pronunciation lexicon is required in order to tune the recognition system to a specific corpus by statistical training: frequencies of distribution of words in a corpus are interpreted as the prior (*a priori*) probabilities of words in a given context. These prior probabilities may be based on the absolute frequencies of words, or on their frequencies relative to a given context, e.g. *digram* (*bigram*) frequencies. The functionality of spoken language lexica may be summarised in the following terms.

- Off–line functions
    - Fully inflected word form list construction
    - Pronunciation table (lexicon) construction
        - Synthesiser development
        - Recogniser (forced alignment, stochastic training)
        - Orthographic transcription (Transliteration) checking
        - Integration of word and sentence prosody
        - Integration of morphology for
            - describing new words
            - sparse data training with stems and word classes
    - Frequency table construction
    - Word distribution frequency tables
    - Language models
    - Coverage definition for inter-project coordination
- On–line functions
    - Search for word forms for inclusion in word lattice
        - orthographic (conventional technology)
        - phonological (new architectures)
    - Definition of criteria for lexicon architecture and lookup

2 Types of lexical information

2.1 Lexicon models and lexical representation

A given system lexicon or lexical database is based on a *lexical information model* or a *data model*; often the model is intuitively constructed, or based on notions taken from traditional school grammar, but scientifically motivated models are becoming available. A model of lexical information will make at least the following distinctions:

- Lexical objects: The basic objects (such as words) described in a lexicon. It is becoming customary in lexicography and computational linguistics to refer to the *lexical sign*, i.e. an object associated with attributes denoting orthogonal kinds of lexical information. A second kind of lexical object is the *lexical sign class* or *archi-sign* in which similar lexical objects are grouped together, each characterised by subsets of the lexical information required to characterise specific lexical signs. These class-based generalisations may be organised in terms of *implication rules (redundancy rules)*, , subsumption lattices, *type hierarchies*, or *default inheritance hierarchies.*
- Lexical information: In a theoretically well-founded lexicon which satisfies formal criteria of consistency and coverage criteria such as empirical completeness and soundness, types of lexical information are *orthogonal*, i.e. of different types which complement each other. These orthogonal types of lexical information are often labelled with *attribute* names, and the items of information regarded as the *values* of these attributes. Values may be complex, expressed as nested attribute-value structures. The types include orthography, pronunciation, syntactic distributional properties, meaning, and pragmatic properties of use in context (e.g. speech act type, stylistic level). See also the results of the EAGLES Working Group on Formalisms.

Lexicon models for lexical databases and system lexicons are part of the overall conceptual framework required for lexicon development. Modern approaches to lexicon development provide suitable lexical representation languages for formulating and integrating the different kinds of lexical information specified in a lexicon model and assigning them to lexical objects, and implementations for these representation languages (cf. Andry et al. 1992). In recent work, the following useful distinctions are sometimes made:

- Lexicon formalism: A specially designed logic programming language such as DATR, or an algebraic formalism such as attribute-value matrices, or appropriate definitions in high level languages such as LISP or Prolog, with compiler concepts for translating these languages into conventional languages for efficient processing. Imperative languages such as C are sometimes used directly to represent smaller lexicons, or where speed of access is at a premium, but this is not a generally recommended practice.
- Lexicon theory: A coherent and consistent set of expressions formulated in a well-defined formalism and interpreted with respect to a lexicon model.
  - General lexicon theory: A general theory of lexical objects and information, for instance a theory of lexical signs and their representation.
  - Specific lexicon theory: A given lexicon formulated in a lexicon formalism on the basis of a lexicon model.
- Lexicon model: Specification of the domain denoted by a lexicon theory, conceptually independent of the theory itself (cf. the notion of a *data model* for a database). A different definition is also common: the general structure of the objects and attribute-value structures in a formal lexicon. A lexicon model specifies the following kinds of information:
  - Types of lexical object and structure of lexical entries.
  - Types of lexical information associated with lexical objects in lexical entries.
  - Relations between lexical objects and structure of the lexicon as a whole *lexicon architecture.*
- Linguistic framework: In recent large projects such as VERBMOBIL, general linguistic frameworks such as HPSG (Head-Driven Phrase Structure Grammar) have been used.

The aspects of representation and architecture will be dealt with in a later section. The following subsections are concerned with the main kinds of lexical information required for spoken language lexical entries.

2.2 A simple sign model for lexical properties

Lexical information is often regarded as a heterogeneous collection of idiosyncratic information about lexical items. An assumption such as this makes it hard to discuss lexical information systematically and, moreover, from the point of view of contemporary lexicography, it is wrong. For this reason, a simple unifying informal model of lexical signs, related to a view which is current in computational linguistics and computational lexicography, is used for the purpose of further discussion (see Figure 2.1 for the basic ILEX — Integrated Lexicon — sign model, and Figure 2.2 for a more general level of a multi–level sign).

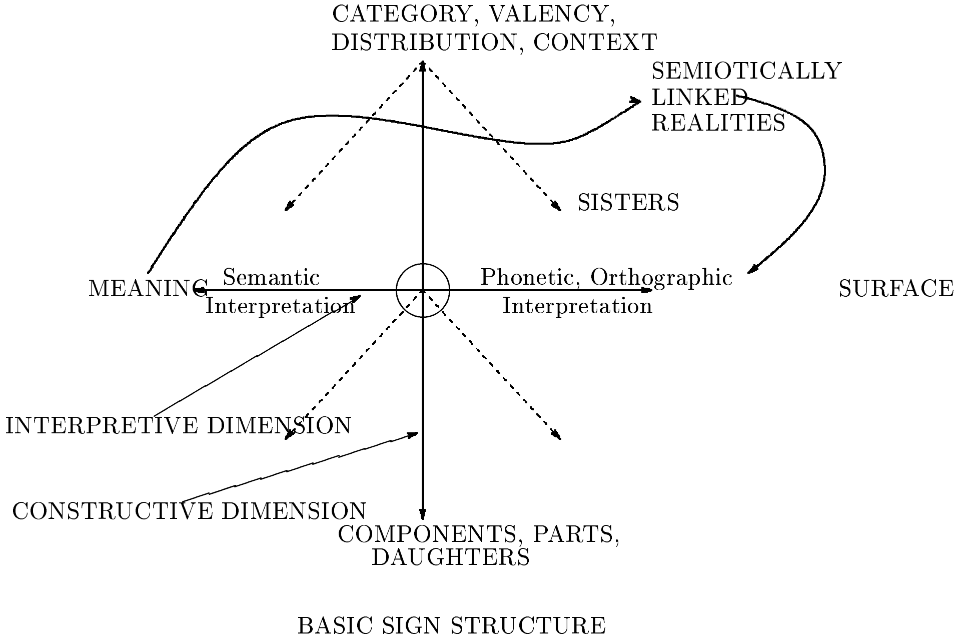

CATEGORY, VALENCY, DISTRIBUTION, CONTEXT
SEMIOTICALLY LINKED REALITIES
SISTERS
MEANING
Semantic Interpretation
Phonetic, Orthographic Interpretation
SURFACE
INTERPRETIVE DIMENSION
CONSTRUCTIVE DIMENSION
COMPONENTS, PARTS, DAUGHTERS
BASIC SIGN STRUCTURE

Figure 2.1: ILEX sign model.

In general terms, a sign is a unit of communication with identifiable form and meaning. Lexical signs have specific *ranks*, such as *word* or *phrase* (for phrasal idioms), and include: *words*, *phrasal idioms* and other items such as *dialogue control particles* (*er*, *uhm*, *aha* etc.). It may also be argued that even smaller units such as morphemes also have sign structure. Lexical signs thus range, in principle, over fully inflected word forms, morphs (roots, affixes), stems (roots or stems to which affixation has applied), lemmas (or lemmata), often represented by an orthographic form, and phrasal items (idioms).

Lexical signs are characterised by the following four basic types of information:

1. Surface properties: *orthographic* and *phonological* representation; for pronunciation, several different levels of transcription are possible (morphophonemic, phonemic, phonetic).
2. Semantic properties: *semantic* and *pragmatic* representation.
3. Distributional properties: *syntactic category* and *subcategory* (e.g. *Verb*, *Transitive Verb*).
4. Compositional properties: *head* and *modifier* (*complement* or *specifier*) constituents; word formation is recursive:
   [[[[*mouse*] [*trap*]] [[*repair*] [*shop*]]] [*owner*]]

The first two types are referred to as *interpretative* properties, since they interpret the basic sign representation in terms of the real world of phonetics (or writing) and the real world of meaning, while the second two types are referred to as *structural* (or *syntactic*, in a general sense of the term) properties. Complex signs are constructed compositionally from their constituent signs and derive their properties compositionally from these. Non-lexical signs include, for example, freely invented compound words, such as the example given above, or almost any sentence in this book.

The following sections will be devoted to the four main types of lexical information, referring to them as *surface*, *content*, *grammatical* and *morphological* information, respectively.

In the examples given below, a basic computer-readable attribute-value syntax is used, based on the kind of spoken language lexical representation in DATR used by Andry et al. (1992). The name of the lexical sign (which is not necessarily its orthography) is written with an initial upper case letter and followed by a colon, attribute names can be either word-like atoms or sequences of atoms (in the latter case, permitting an indirect representation of more complex attribute structures); they are enclosed in corner brackets and separated from their values by an equality sign, and the lexical sign is terminated by a period. The SAMPA notation is used below.

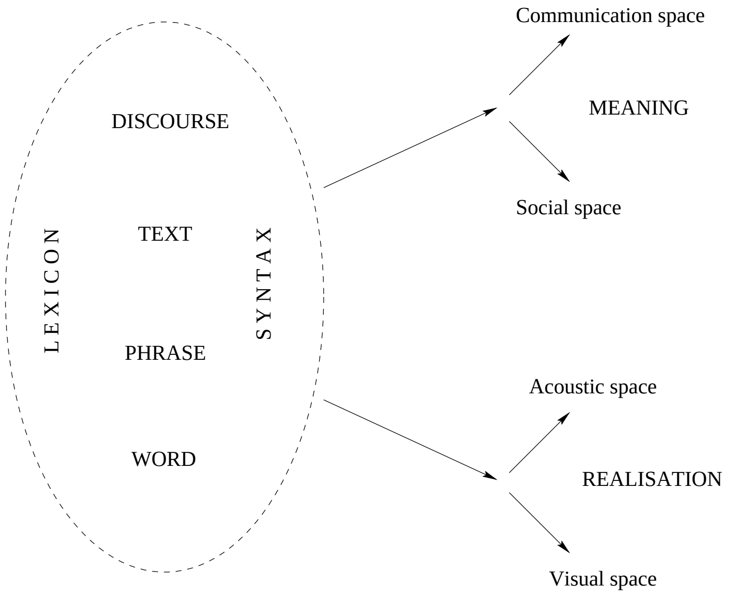


Figure 2.2: Sign model with ranks, lexicon and interpretations.

```
Table:
  <surface orthography>      = table
  <surface phonetics sampa> = teIbl
  <semantics>                = artefactual horizontal surface
  <distribution>             = noun common countable
  <composition>              = simplex z_plural.
```

In the case of complex signs, the meaning of the sign is a function of the meanings of its parts and the pronunciation of the sign is a function of the pronunciations of its parts. These functions may be partly idiosyncratic with lexical signs; this is shown in the pronunciation and meaning of words like English "dustman":

```
Dustman:
 <surface orthography>      = dustman
 <surface phonetics sampa>  = dVsm@n
 <semantics>                = 'municipal garbage collector'.
```

The *pronunciation* and *meaning* of this complex lexical sign are not in all respects a general compositional function of its parts, for example the pronunciation of *dustman* is not /dʌstmæn/ but /dʌsmən/, nor is a dustman necessarily only concerned with dust:

```
Dust:
  <surface phonetics sampa> = dVst
  <semantics>                = 'just visible particles of
                               solid matter'.

Man:
  <surface phonetics sampa> = m{n
  <semantics>                = 'male adult human being'.
```

In contrast, the *spelling* and the *distribution* of the complex sign are perfectly regular functions of the spellings of the parts and the distribution of the head (i.e. `Man`) of the sign, respectively.

In perfectly regular cases, there would therefore be no necessity to include complex words in the lexicon. Such cases are practically non-existent, however, since complex words are in general partially idiosyncratic; in a comprehensive spoken language lexicon, both complex words and their parts therefore need to be included. For most current practical purposes, in which *potential words* (unknown words or ad hoc word formations) do not need to be treated in addition to *actual words* (those contained in a lexicon), complex words can be listed in full as unanalysed forms.

Modern computational lexicographic practice attempts to reduce the redundancy in a lexicon as far as possible: fully regular information in compounds can be *inherited* from the parts of the compounds, while idiosyncratic information is specified locally. In a case like this, a lexical class is specified for defining the structure of compounds, and "inheritance pointers" are included. The result is a *hierarchical lexicon structure*, in which macro-like cross-references are made to other lexical signs (analogous to cross-references in conventional dictionaries), but also to whole classes of lexical signs (*archi-signs*).

2.3 Lexical units

2.3.1 Kinds of lexical unit

Intuitively, the prototypic lexical unit is a word. This definition has a number of catches to it, however, because the notion of word is not as simple as it seems, and because lexical phrases (idioms) also exist. The intuitive notion of *word* has "fuzzy edges", as in the following cases:

1. Words may contain other words (e.g. compound words such as *database*, *Sprachtechnologie*).
2. Words have different status in respect of their phonetic realisations and their meaning; compare the difference between function words, e.g. *to*, *for* with reduced pronunciations and structural meanings, and content words, e.g. *word*, *spell*, which refer to real world objects, properties, event types, abstract concepts.
3. Words may be merged with other words in informal speech (*cliticisation*). Examples of clitics are English *'s* in *he's* – /hi:z/, French *l'* in *il l'a vu* – /il la vy:/, German *'m* in *auf'm Tisch* – /aʊfm tɪʃ/.
4. Particular types of word formation such as spelling and acronym formation may require special attention: *ecu* – /i:kˈˈju:/, /i:si:ˈˈju:/.
5. Words may be *inflected word forms*, making *sound* (singular) and *sounds* (plural) into different words. On the other hand, words may be regarded as a class of inflectionally related forms (a *paradigm*), i.e. *sound* and *sounds* then belong to the same word, which may be characterised by a canonical inflected form (e.g. nominative singular), or by the stem shared by the forms and identified by linguistic analysis, or by a number or other abstract label. In speech technology, the *inflected word form* is the standard definition. In standard dictionaries, the *paradigm* definition of word is used, represented by a *headword* or *lemma*, generally the canonical inflectional form such as nominative singular, in orthographic representation.
6. Lexical units may need to be larger than the word (e.g. phrasal idioms).
7. Lexical units may need to be smaller than the word: *Semantically oriented* morphological word subunits (word constituents) include
   - word stems minus inflections; indivisible word stems are *lexical morphemes*);
   - constituent words words formed by compounding (composition);
   - constituent prefixes, stems and suffixes in words formed by derivation.

   Pronunciation oriented phonological word subunits include syllables and their parts; phonological subunits do not necessarily correspond closely with morphological subunits.
8. Linguistic textbooks distinguish between several different views of words as lexical units, depending on which kind of lexical sign information is regarded as primary:
   - The *phonological word* (based on its conformity to the phonotactic structure of a language).
   - The *prosodic word*, based on its conformity to the accentuation and the rhythm patterning of the language.
   - The *orthographic word* (for instance, as delimited by spaces or punctuation marks).
   - The *morphological word* (based on the indivisibility and fixed internal structure of words).
   - The *syntactic word* (based on its distribution in sentences).
9. The lexical word as a *type*, as opposed to an *occurrence* of the type in larger units, and a *token* of the type in a corpus of speech or writing.

The central meaning for the purpose of spoken language lexica will be taken to be the *morphological word.*

Lexical units (entries, items) are assigned sets of properties; these identify the lexical units as signs, and determine the organisation of the lexicon. In practical contexts, the choice of lexical unit and the definition of priorities among its properties may be important for procedural reasons, i.e. in determining ways in which a lexicon may be most easily accessed: through orthography, pronunciation, meaning, syntactic properties, or via its morphological properties (stem, inflection). The application-driven decision on the kind of lexical unit which is most suitable for a given purpose is a non-trivial one. However, for many practical purposes fairly straightforward guidelines can be given:

- The form of a lexical item, in particular its orthography, is often used as the main identifying property for accessing the lexicon.
- However, access on phonetic grounds, via the phonological form, is evidently the optimal procedure for speech recognition, and access on conceptual semantic or syntactic grounds is evidently the optimal procedure for speech synthesis.
- The use of orthography as an intermediate stage in speech recognition is a useful and widespread heuristic which generally does not introduce significant numbers of artefacts into the mapping from speech signals to lexical items, but is not recommended for complex systems with large vocabularies, except as a means of visualisation in user interfaces.
- For text-to-speech applications orthography is likely to be the optimal lexical access key.

2.3.2 Fully inflected form lexica

It has already been noted that fully inflected form lexica and lexical databases are fairly standard for speech recognition. Where a small closed vocabulary is used, and new, unknown or ad hoc word formations are not required (as with most current applications in speech synthesis and recognition), fully inflected word forms are listed. This procedure is most convenient in languages with very small inflectional paradigms; for languages of the agglutinative type, in which large numbers of inflectional endings are concatenated, the procedure rapidly becomes intractable. In other applications, too, such as speech synthesis, it may be more tractable to generate fully inflected word forms from stems and endings.

An example of a language with few inflections is English, where (except for a few pronouns) only nouns and verbs are inflected, and even here three forms exist for nouns (uninflected, genitive and plural) and four for verbs (uninflected, third person singular present, past, and present participle; irregular verbs in addition have a different past participle form – the verb *to be* is, as always, an extreme case). English is therefore not a good example for illustrating inflectional morphology (in other areas of morphology, i.e. in word formation, languages appear to be equally complex).

French is much more complex, with inflections on adjectives, and large verb paradigms; note that orthographic inflection in French has more inflectional endings than are distinguished in phonological inflection. German also has complex inflectional morphology, with significantly more endings on all articles, pronouns, nouns, adjectives and verbs, increasing the size of the vocabulary over the size of a stem-oriented lexicon by a factor of about 4.

In extremely highly inflecting languages such as Finnish, the number of endings and the length of sequences of endings multiply out to increase the vocabulary by a factor of over 1000. Special morphological techniques have been developed (e.g. two-level morphology) to permit efficient calculation of inflected forms and to avoid a finite but unmanageable explosion of lexicon size for highly inflecting languages (cf. Koskenniemi 1983b; Karttunen 1983). These techniques have so far not been applied to any significant extent in speech technology (but cf. Althoff et al. 1996).

The figures cited refer only to the sets of forms. When the *form-function mapping*, i.e. the association of a given inflected form with a morphosyntactic category, is considered, the figures become much worse. A single inflected adjective form such as *guten* in German has 44 possible interpretations which are relevant for morphosyntactic agreement contexts (cf. Gibbon 1995), with 13 *feminine* readings, 17 *masculine* readings, and 14 *neuter* readings, depending on different cases (*nominative*, *accusative*, *genitive* and *dative*) and different determiner (article) categories (*strong*, *weak* and *mixed*). It is possible to reduce the size of these sets by means of default-logic abbreviations in a lexical database, but for efficient processing, they ultimately need to be multiplied out. Similar considerations apply to other word categories, and to other highly inflecting languages.

Complex inflectional properties in many languages other than English imply that, for these languages, large vocabulary systems with complex grammatical constructions require prohibitively large fully inflected form inventories. Although the sets of mappings involved can be very large, the inflectional systems of languages define a finite number of variants for each stem, and therefore it may make sense in complex applications in speech recognition to define a rule-based "virtual lexical database" or "virtual lexicon" which constructs or analyses each fully inflected word form on demand using a *stem* or *morph lexicon* with a *morphological rule component* (Althoff et al. 1996; Bleiching et al. 1996a; Geutner 1995).

2.3.3 Stem and morph lexica

Lexica based on the morphological parts of words, coupled with lexical rules for defining the composition of words from these parts, are not widely used in current speech recognition practice. They are useful, however, in expanding lexica of attested forms to include all fully inflected forms, for instance for word generation and speech synthesis, and in tools which verify the consistency of corpus transcriptions and lexica.

Terminology in this area is somewhat variable. In the most general usage, a *stem* is any uninflected item, whether morphologically simple or complex. However, intermediate stages in word formation by affixation, and in the inflection of highly inflected languages, are also called stems. The smallest stem is a *phonological lexical morph* or an *orthographic lexical morph*, i.e. the phonological or orthographic realisation of a *lexical morpheme.* Since stems may vary in different inflectional contexts, as affixes do, it is necessary to include information about the morphophonological (and morphographemic) alternations of such morphemes:

```
Knife:
  <surface phonology singular>   = naIf
  <surface phonology plural>     = naIv + z
  <surface orthography singular> = knife
  <surface orthography plural>   = knive + s.
```

The use of morphological decomposition of the kind illustrated here has been demonstrated to bring some advantages in medium size vocabulary speech recognition in German (cf. Geutner 1995); for languages like English, with a low incidence of inflections, the advantage is minimal.

In a stem lexicon, the basic lexical key or *lemma* is the stem, which is represented in some kind of normalised notation. The most common kind of normalised or canonical notation has the following two properties:

1. Canonical inflected form: With morphologically inflected items, a “normal form” such as the *infinitive* for verbs or the *nominative singular* for nouns is used.
2. Canonical orthography: A standardised orthographic representation of the canonical inflected form is used.

For specific purposes, in which lexical entries need to be accessed on the basis of a specific property, indexing based, for instance, on the canonical phonemic representation, either of a fully inflected form or of the canonical inflected form, or even of the stem itself, may be required; for stochastic language models, for example, a tree-coded representation may be the optimal representation. Phonemic representation is dealt with in more detail below.

2.3.4 The notion of “lexical lemma”

As in the *knife* example, one particular form, for instance orthographic, of an entry is often used as a headword or lemma. From a technical lexicographic point of view, this form then has a dual function:

1. It names the entry.
2. It also represents one of its properties, namely its spelling.

In spoken language lexicography, this distinction is central, and ignoring it may lead to confusion. This applies particularly in the context of spoken language lexicography, where the primary criterion of access by word form is phonological.

When homographs occur (e.g. *bank* as a financial institution or as the side of a river), an additional consecutive numbering is used, e.g. $bank_1$, $bank_2$, etc.

The concept of an *abstract lemma*, deriving from recent developments in computational linguistics and their application to phonology and prosody, may be used in order to clarify the distinction (cf. Gibbon 1992a): an abstract lemma may have any convenient unique name or number (or indeed be labelled by the spelling of the canonical inflected form, as already noted); all properties have equal status, so that the abstract lemma is neutral with respect to different types of lexical access, through spelling, pronunciation, semantics, etc. The examples of lexical entries given so far are based on the concept of an abstract lemma. The neutrality of the abstract lemma with respect to particular properties and particular directions of lexical access make it suitable as a basic concept for organising flexible lexical databases. A lexicon based on a neutral abstract lemma concept is the basic form of a *declarative lexicon*, in which the structure or the lexicon is not dictated by requirements of specific types of lexical access (characteristics of a *procedural lexicon*, but by general logical principles. The distinction between declarative and procedural lexica is a relative one, however, which is taken up in the section on spoken language lexicon architectures. For practical applications, a lexical database will need to be procedurally optimised (= indexed) for fast access.

2.4 Lexical properties and lexical relations in spoken language

The complex relations between orthographic, phonological, syntactic and semantic properties of lexical units make a theoretically satisfying definition of "lexical sign" quite elusive. Lexical relations are either *paradigmatic*, and define classes of similar items, or *syntagmatic*, and define complex items in terms of relations between their parts.

Present discussion is restricted to the main paradigmatic relations in traditional terms. The expression of these relations in terms of *semantic features*, *semantic markers* or *semantic components* is not dealt with explicitly, though it figures implicitly in the attribute-value structures which are referred to in the examples.

The syntagmatic relations (semantic roles; collocational relations; syntactic subcategories, valencies) are more complex. Introductions to linguistics may be consulted on syntagmatic relations in sentences (constituent structures and dependency structures). For further information on semantic properties, reference should be made to standard textbooks such as Lyons (1977) or Cruse (1986). Reference should also be made to the results of the Eagles Computational Lexica Working Group.

The following systematised versions of traditional definitions express the main paradigmatic relations between lexical signs.

1. The main relations of *form* between lexical signs are as follows:

   Homonymy: Two words with the same orthographic and phonological forms, but different syntactic categories and/or meanings are homonyms. Example: *mate* /meɪt/ 'friend' or 'final state of play in a chess game'.

   Homography: Two words with the same orthographic form and different phonological forms are (heterophonic) homographs. Example: *row* /rəʊ/ 'horizontal sequence', /raʊ/ 'noise, quarrel'.

   Homophony: Two words with the same phonological form and different orthographic forms are (heterographic) homophones. Example: *meet* /mi:t/ 'encounter' – *meat* /mi:t/ 'edible animal tissue'.

   Heterography: Two orthographic forms of the same word are heterographs. Example: *standardise* – *standardize* /stændədaɪz/.

   Heterophony: Two phonological forms of the same word are heterophones. Example: *either* /aɪðə/ – /i:ðə/ 'disjunction'.

2. The main relations of *function* between lexical signs:

   Hyperonymy: If the meaning of one word is entailed by the meaning of another, it is a hyperonym of the other (a superordinate term relative to the other). Example: *book* is a hyperonym of *manual* as the meaning of *book* is implied by the meaning of *manual* (in one of its readings).

   Hyponymy: The converse of hyperonym. If the meaning of one word entails the meaning of another, it is a hyponym of the other (a subordinate term relative to the other). Example: *manual* is a hyponym of *book* as the meaning of *manual* implies the meaning of *book*.

   Co-hyponymy: Two words are co-hyponyms if and only if there is a word which is a hyperonym of each (in the same reading of this word). Example: *manual* and *novel* are co-hyponyms in relation to *book*.

   Synonymy: Two words are synonyms if and only if they have the same meaning (or at least have one meaning in common), i.e. if the meaning of each entails the meaning of the other. They are partial synonyms if either has additional readings not shared by the other. They are full synonyms if they have no reading which is not shared by the other. Example: *manual* and *handbook* are partial synonyms (*manual* is also, among other things, a term for a traditional organ keyboard). Full synonyms are rare. By implication, synonyms are also co-hyponyms.

   Antonymy: Two words are antonyms (a) if they are co-hyponyms with respect to given meanings, and (b) if they differ in meaning in respect of those details of the same meaning which are not shared by their hyperonym. Example: *manual* and *novel* are antonyms. Note that the term is sometimes restricted to binary oppositions, e.g. *dead* – *alive*.

In addition to these lexical relations, there are a number of syntagmatic complexities which hold between different types of information.

- Semantically, recursion in word formation is unrestricted, with left- or right branching, or centre-embedding.
- Morphologically, recursion is restricted to flat, linear concatenation, as in:
  Spracherkennungsevaluationsmethode –
  Sprach#er+kenn+ung#+s#evalu+ation#+s#method+e,
  or *operationalisation* – *oper+at+ion+al+is+at+ion*, which can be efficiently described and implemented by finite state devices.

- Morphophonological modifications of the basic concatenative structure occur, with superimposed word stress or tone patterns, vowel and consonant modifications, as in *telephone – telephony*, *bring – brought.*
- So-called *bracketing paradoxes* occur because of the different morphological structures determined by semantics and phonology; the most well-known example is *transformational grammarian*, semantically bracketed as ((*transformational grammar*) *ian*), morphologically bracketed as (*transformational* (*grammar ian*)).
- Note, too, that morphological (lexical, semantic oriented) bracketing does not necessarily correspond with non-lexical, phonologically motivated syllabic bracketing, as in *operation operate – ion* – /ɔ. pə. r'ˈeɪ. ʃən/.

2.5 Surface information: orthography

Orthography has been used in several different roles in spoken language lexica, some of which have already been noted:

1. Convenient general reference labels for words, due to the high level of awareness of, familiarity with and standardisation of orthography in literate societies.
2. Convenient identifying names for lexical entries, for “normal lemma” forms, and for headwords in complex lexicon entries which group related words together.
3. Convenient identifying names for word hypotheses in word lattices, as lexical lookup keys.
4. Visualisation of word hypotheses in a development system.
5. Representation of the orthographic properties of words (the main function).

Each of these functions is distinct and needs to be kept conceptually separate in order to avoid confusion. The functions (1) and (2) are not particularly problematic. Function (3) is traditionally a feature of speech recognition systems for relatively small vocabularies. The larger the vocabulary, however, the greater the danger of introducing unnecessary *orthographic noise*, i.e. intrusive artefacts due to homography (words with identical spelling and different pronunciation); for this reason, in new architectures, phonological (e.g. phonemic or autosegmental) representation in word graphs may be preferred. Function (4) is unproblematic, though similar reservations as with (3) are to be noted. Function (5) is the main function and is obviously essential for written output of any kind; however, it is often confused with both functions (2) and (3). Care with consistent orthography is obviously essential.

Orthography has the advantage of being highly standardised, except for certain regional variants (British and American English; Federal, Swiss, and Austrian German) and variations in publishers’ conventions (e.g. British English *ise/–ize* as in *standardisation/standardization*, capitalisation of adjectives in nominal function in German, as *die anderen / die Anderen*, or variations in hyphenation conventions and the spelling of compound words; variation is found particularly in the treatment of derived and compound word s (e.g. separation and hyphenation) and in the use of typographic devices such as capitalisation). Orthography is given further attention in the section on lexical representation.

A standard orthographic transcription is often used for convenience as a means of representing and accessing words in a spoken language lexicon. This has several reasons:

1. Familiarity to all educated speakers of the language.
2. High level of standardisation in comparison with theory-influenced phonological transcriptions.
3. Sufficient proximity to phonological form, at least in European languages, ensures a reasonably close mapping to pronunciation at the level of whole words (not necessarily in the details of grapheme to phoneme mapping) in small vocabularies in some languages (French and English are notorious exceptions).

Most European languages have highly regulated orthographies, the use of which is associated with social and political rewards and punishments. Official orthographic reforms, which typically generate much controversy among the general public, may necessitate some re-implementation of spelling checkers and grapheme-phoneme converters (cf. the ongoing reform of German orthography).

For use in spoken language lexica, particularly in word lists used for training and testing recognisers, consistency is essential and often additional conventions are required in order to meet the criterion of general computer readability in the case of special letters and diacritics. Although it cannot be regarded as a standard, it is becomming common practice to use the ASCII codings or their LaTeX adaptations for specific countries. For example, a standard computer-readable orthography for German has become widely accepted for German speech recognition applications which marks special characters, in particular those with an *Umlaut* diacritic, as shown in Table II:2.

The results of the EAGLES Working Groups on Text Corpora and Lexica should be consulted on orthographic and other matters pertaining to written texts.

Table 2.1: Computer readable ASCII orthography for German

Standard orthography	ASCII orthography
Äpfel	`''Apfel`
ändern	`''andern`
Öl	`''Ol`
östlich	`''ostlich`
Überzug	`''Uberzug`
über	`''uber`
heiß	`hei''s`

2.6 Surface information: pronunciation

Pronunciation information is much more application specific (and indeed theory specific) than orthographic information. Standardly, information about phonemic structure is included in the form of a *phonemic transcription* of a standard *canonical* or *citation form* pronunciation, i.e. the pronunciation of a word in isolation in a standard variety of the language. Often the phonemic transcription is enhanced by including prosodic information such as the stress position (Dutch, English, German), the type of tonal accent (Swedish), syllable and word boundaries in compound words, and word and phrase boundaries in phrasal idioms. Morphological information (morph boundaries, as well as the boundaries of words and phrases) is relevant to stress patterning, and is sometimes also included.

A particularly thorny question is the inclusion of information about pronunciation variants, of which there are two main types, *rule-governed* allophonic and phonostylistic variants, and idiosyncratic *lexical* variants. The following rules of thumb can be given:

- Pronunciation lexica for synthesis generally require one standard (canonical) pronunciation; however, variants of these with different prosodic contexts may be required.
- Pronunciation lexica for recognition require a distinction to be made between variants of the same word, and variants which are associated with the same spelling but different words (heterophonic homographs).
- Strictly speaking, pronunciation lexica for recognition require only lexical variants to be listed which are idiosyncratic and cannot be predicted by rule (e.g. English *either* /aɪðə/ – /iːðə/). Variants which are general and regular (such as the reduction of schwa + liquid or nasal to a syllabic liquid or nasal) can be calculated using pronunciation rules (phonological rules): English *running* /rʌnɪŋ/ – /rʌnɪn/, German *einem* /aɪnəm/ – /aɪnm̩/ – /aɪm/).

Although *phoneme* is a technical term with somewhat different definitions in different theoretical contexts, and although there are technical arguments due to Generative Phonology (cf. Chomsky and Halle 1968) which show that the notion of phoneme leads to inconsistencies, the core of phoneme theory is relatively standard. In linguistics handbooks, the phoneme is commonly defined as the minimal distinctive (meaning-distinguishing) unit (temporal segment) of sound. In the following fairly standard definition, the distinctiveness criterion is implicit in the concept of a *system*; the concept of a *sound* (= *phone*, *allophone*) covers possible variants of a phoneme (e.g. English aspirated word-initial /p/ as opposed to unaspirated /p/ in the context /sp. . . / (Crystal 1985, p. 228):

> PHONEME (PHONEMIC(S)) The minimal unit in the sound SYSTEM of a LANGUAGE . . . Sounds are considered to be members of the same phoneme if they are phonetically similar and do not occur in the same ENVIRONMENT.

A fairly complete definition is thus based on *distinctiveness*, *minimality*, *phonetic similarity* and *distributional complementarity*. Phoneme definitions are differential or relational definitions, illustrated by the notion of minimal difference between two words in minimal pairs such as the items in the set of English words *pin–tin–kin–fin–thin–sin–shin–chin–bin–din–gin–win–Lynne–Min–Nin*, (in standard SAMPA computer readable phonemic transcription: /pɪn – tɪn – kɪn – fɪn – θɪn – sɪn – Sɪn – tʃɪn – bɪn – dɪn – ʤn – wɪn – lɪn – mɪn – nɪn/) (the last three are names). Phonemes defined in this way are further classified as bundles of phonological distinctive features. Operationally, phonemes are defined by procedures of segmentation and classification (reflected, for example, in the recognition and classification components of automatic speech recognition systems):

- Segmentation is the procedure of isolating minimal distinctive temporal phonetic segments (*phones*).
- Classification is the procedure of classifying phones as *allophones* (phonetic *alternants* of the same *phoneme*, on the grounds of distinctiveness, minimality, phonetic similarity and complementary distribution (i.e. their occurrence in complementary contexts as contextual variants of that phoneme).

In contrast to orthographic representations, which for social and cultural reasons, are highly standardised common knowledge, lexical representations of pronunciation are theory and application specific. The most widely used representations in pronouncing dictionaries for human use, such as in foreign language teaching, and in spoken language systems, are *phonemic transcriptions.*

Phonemic descriptions are available for several hundred languages, and phonemic transcriptions based on these are suitable for constructing roman orthographies for languages which have orthographies based on different principles (e.g. syllabic or logographic) or no orthography at all. For a given language, phonemic descriptions differ peripherally (for instance, it is controversial whether diphthongs and affricates are to be analysed as one phoneme or two?). Phonemes are in general the units of choice for practical phonological transcriptions in spoken language system lexica. Other, more specialised types of representation such as the feature matrix representations required by all modern phonological descriptions, and autosegmental lattice representations, or metrical tree graph and histogram representations (cf. Goldsmith 1990) are increasingly finding application in experimental systems (cf. Kornai 1991; Carson-Berndsen 1993a; Kirchhoff 1996; Church 1987b,a) because of their richness and their more direct relation to the acoustic signal, in contrast to phonemic representations. However at the lexical level, they can generally be calculated relatively easily from the more compact, but less general, phonemic representations. Because of the widespread use of phonemes, the concept is discussed in more detail below; for fuller explanations, textbooks on phonology should be consulted.

The central question in phonological lexical representation, in cases where the notion of phoneme alone is not fully adequate, is that of the *level of representation* (*level of description*, *level of abstraction*). There are three main levels, each of which is an essential part of a full description, and which needs to be evaluated for all but the simplest applications, *morphophonemic*, *phonemic*, and *phonetic*, which are characterised below.

MORPHOPHONEMIC: The morphophonemic level provides a simplification of phonological information with respect to the phonological level; the simplifications utilise knowledge about the morphological structure of words, and permit the use of *morphophonemes*, (a near-synonym is *archiphoneme*) which stand for classes of morphologically and phonologically related phonemes.

A standard example of a morphophoneme is the final obstruent in languages with final obstruent devoicing, including Dutch and German. For example, the phonemic representation German *Weg* /ve:k/ 'way' – *Wege* /ve:gə/ 'ways' corresponds to a morphophonemic representation {ve:G} – {ve:G+ə}, which simplifies the description of the stem of the word. The morphophoneme {G} stands for the phoneme set {/k/, /g/}, and selection of the appropriate member of the set (the appropriate feature specification) is triggered by the morphological boundary and neighbouring phonological segments. Alternatively the morphophoneme may be said to consist of the underspecified feature bundle shared by /k/ and /g/, or more technically, the feature bundle which *subsumes* the feature bundles of /k/ and /g/.

An example from English is the alternation /f/ – /v/ in plural formation in one class of nouns, as in *knife* /naɪf/ – *knives* /naɪvz/, which can be represented morphophonemically as {naɪV} – {naɪV+z}. The morphophoneme {V} stands for the phoneme set {/f/, /v/}. Here, too, selection of the phoneme (specification of the underspecified subsuming feature bundle) is determined by the morphological boundary and the phonological properties of neighbouring segments.

A corresponding level is necessary for the description of spelling: cf. variations such as English *y–ie* in *city – cities*, or German *s–ss–ß* as in *Bus* – *Busse*, *Kuß* (*Kuss* in the new orthography) – *Küsse* and *Fuß* – *Füße*.

Morphophonemic representations augmented by realisation rules are a useful *compression technique* for reducing lexicon size:

- Lexica can be stem-based, and thus have fewer entries, and all inflections can be automatically calculated by rule for any stem in the lexicon.
- Morphotactic and morphophonological rules can be used for extending lexica of fully inflected attested forms, and for checking such lexica for consistency.

For requirements such as these, the use of morphophonemic representations, supplemented by morphological construction rules and morphophonemic mapping rules is recommended (Koskenniemi (1983b), Karttunen (1983), Ritchie et al. (1992), Bleiching et al. (1996a) for descriptions of various practical approaches).

There are no standard conventions for the representation of morphophonemesmorphophonemics, whether computer readable or not (but see the SAMPA alphabet for French); capital letters are often used in linguistics publications. Note that this use of capital letters at the morphophonemic level should not be confused with the use of ASCII upper case codes in the SAMPA alphabet at the phonemic level.

Citations of *morphophonemic* representations are often delimited with brace brackets {...}.

PHONEMIC: The phonemic level is a standard intermediate level corresponding to criteria outlined in more detail below. The standard European computer readable phonetic alphabet is SAMPA: this alphabet is used for the main languages of the European Union, and is recommended for this purpose. The internationally recognised standard alphabet for phonemic representations is the International Phonetic Alphabet (IPA). The IPA alphabet is used for the most part in the text of this book. One of the main functions of the International Phonetic Association since its inception over 100 years ago has been to coordinate and define standards for this alphabet.

Until relatively recently, the special font used for the IPA has made it difficult to interface it with spoken language systems, and for this reason a number of computer-readable encodings of subsets of the IPA have been made for various languages (cf. Allen 1988; Esling 1988, 1990; Jassem and Łobacz 1989; Ball 1991). The standard computer phonetic alphabet for the main languages of the European Union is the SAMPA alphabet, developed in the ESPRIT SAM and SAM-A projects (cf. Wells 1987, 1989a, 1993b,a; Llisterri and Mariño 1993). SAMPA is widely used in European projects, both for corpus transcription and for lexical representations (see also the chapter on Spoken Language Corpora).

However, there is a standard numerical code for IPA symbols (cf. Esling (1988, 1990)), and developments in user interfaces with graphical visualisation in recent years are leading to the increasing use of the IPA in its original form, particularly in the speech lab software which is used in spoken language system development.

Citations of *phonemic* representations are standardly delimited by slashes /.../.

PHONETIC: At the phonetic level further details of pronunciation, beyond the phonemically minimal features, are given. Since the relation between the phonemic and the phonetic level can be described by general rules mapping phonemes to their detailed realisations (allophones) in specific contexts (cf. Woods and Zue 1976), it is strictly speaking redundant to include these regular variants in a lexicon. However, for reasons of efficiency, detailed phonetic word models for speech recogniser training or for speech synthesis may be calculated using phonological rules and stored. Essentially this is a software decision: whether to use tables (for efficiency of lookup) or rules (for compactness and generality) for a given purpose.

A specific version of the phonetic level of transcription is *phonotypic* transcription, defined as a mapping from the phonemic level using regular phonological rules of assimilation, deletion, epenthesis (cf. Autesserre et al. 1989); this level is frequently used for generating additional word models to improve speech recognition. Since the amount of phonetic detail which can be processed depends heavily on the vocabulary size and the number of phonological rules which are considered relevant, no general recommendation on this can be given.

There is no widely used standard ASCII encoding of the entire IPA for computer readable phonetic representations and therefore no recommendations can be given on this. A proposal by John Wells, the originator of SAMPA, is under discussion. Currently, individual laboratories use their own enhancements of phonemic representations. However, the fuller encodings mentioned in connection with the phonemic level of transcription are eminently suitable for interface purpose at the phonetic level, and will no doubt be increasingly used where more detailed phonetic information is required.

Citations of *phonetic* forms are standardly delimited by square brackets [...].

2.7 Prosodic information

The area of word prosody, and, more generally, the description of other prosodic units which have quasi-morphemic functions, is gradually emerging as an important area for spoken language lexica. For present purposes, prosodic properties are defined as properties of word forms which are larger than phonemes. Further specification in phonetic terms (e.g. F0 patterning) and in semantic terms (e.g. attitudinal meaning) may also be given but is not essential for present purposes.

One type of lexical information on prosody pertains to phonological or morphological properties of words, such as Swedish pitch accents, or stress positions in words. Some aspects of word prosody are predictable on the basis of the regular phonological and morphological structure of words, but some are idiosyncratic. Examples in English where word stress is significant include the noun-verb alternation type as in *export* – /''ekspɔ:t/ (Noun), /eksp''ɔ:t/ (Verb). In German, word stress is significant for instance in distinguishing between compound separable particle verbs and derived inseparable prefixed verbs as in *übersetzen* – /''y:bɐzɛtsən/ (compound) vs. /y:bɐz''ɛtsən/ (derivation).

It has been shown (cf. Waibel 1988) that taking word prosody into account in English can produce a significant improvement in recognition rate.

In addition, there is lexical information associated with prosodic units which occur independently of particular words, and therefore may themselves be regarded as lexical signs and be inventarised in a prosodic lexicon (cf. Aubergé 1992). To give a highly simplified example in a basic attribute-value notation, a prosodic lexicon for an intonation language might have the following structure.

```
Terminal_1:
  <phonetics pitch> = fall
  <semantics>       = statement or instruction.

Terminal_2:
  <phonetics pitch> = rise
  <semantics>       = question or polite instruction.
```

This kind of information, in which prosodic categories function as a kind of *morpheme* with an identifiable meaning, is generally not regarded as lexical information, but treated as a separate layer of organisation in language. Intonation is being taken increasingly into account for *prosodic parsing* in two main senses of this term:

1. Analysis of speech signal in respect of the *fundamental frequency* (*F0*, *F-zero*) trajectory, for speech recognition, in relation to words, sentences and dialogue units.
2. Analysis of sentence structure for the generation of intonation patterns in speech synthesis.

Prosodic representation in the lexicon is in general restricted to the prosodic properties of words, such as stress position in English, Dutch, and German words, or tonal accent in Swedish words, or to rhythmically relevant units such as the syllable and the foot. For spoken language processing in which prosody plays a role, it is also necessary to include an inventory of prosodic forms, and their meanings, which play a role at the sentence level, independently of specific words: i.e. a prosodic lexicon.

It should be borne in mind that in linguistics, "prosody" currently has a broader meaning, and covers all properties of pronunciation which are not directly concerned with defining consonants and vowels. Prosody in this sense covers, for example, syllable structure and phonological word phonotactics, as well as the more traditional categories of intonation, accent, and phrasing.

The IPA defines symbols for representing lexical and non-lexical types of prosody, and a subset (for word prosody) has been encoded in the SAMPA alphabet. However, the state of knowledge in the area of prosody is less stable than in the area of segmental word structure, and a range of different conventions is available (cf. Bruce 1989); in this area, there are SAMPA "dialects", for instance replacing SAMPA `"` and `%` for primary and secondary stress by the more iconic `'` (single quote) and `''` (two single quotes) or `"` (double quote).

The ToBI (Tones and Break Indices) transcription, originally developed for American English, has now been applied to several languages.

In oriented spoken language lexicography within the VERBMOBIL project, attribute-based formal representations of prosodic features in the lexicon have been developed using the ILEX (Integrated Lexicon) model and the lexical knowledge representation language DATR (cf. Bleiching 1992a; Gibbon 1991).

There is an increasing tendency no longer to regard prosodic representations as totally exotic and quite unlike anything else. But there is still insufficient consensus on lexical prosodic features to permit generally valid recommendations to be made for prosodic representations in the lexicon. For most purposes, plain SAMPA or ToBI style symbols will be adequate. For covering new ground with extended lexica for use with discourse phenomena at the dialogue level, a lexical knowledge representation language with a more general notation, as illustrated above, may be more appropriate.

2.8 Grammatical information: morphology

2.8.1 Types of morphological information

Morphology is concerned with *generalisations* about words as lexical signs, in respect of surface form, meaning, distribution and composition. More generally, morphological information is information about semantically relevant word structure; the smallest morphological unit is the *morpheme*, often defined as the *smallest meaningful unit* in a language. Morphemes should not be confused with phonological units such as the *phoneme syllable* and its constituents, which are used for describing the structure of words from the point of view of their *pronunciation*, without direct reference to *meaning*. For applications of morphology to speech recognition see Althoff et al. (1996), Bleiching et al. (1996a), Geutner (1995).

The domain of morphology may be divided in terms of the *functions* of morphological operations, i.e. inflectional *agreement* or *congruence* vs. *word formation*, or in terms of the *structures* defined by morphological operations, i.e. *affixation*, (*prefixation*, *suffixation*, *infixation* or *prosodic modification*) vs. compounding (*concatenation* of *stems* or *words*). These two dimensions can be represented as follows:

OPERATION	AGREEMENT	WORD FORMATION
AFFIXATION	Inflection	Derivation
STEM/WORD CONCATENATION		Composition

There is a gap in the table with regard to the use of stem or word concatenation for agreement; however, so-called *periphrastic constructions* with verbs, typically with auxiliary verbs and participles or infinitives, may be assigned to this slot, prepositions relate to nouns in a comparable way. Compare English *John will come* with French *Jean viendra*, or English *Give it to the cook* with German *Geben Sie es dem Koch.* English lacks an inflectional future, but has periphrastic (phrasal) modal or infinitive complement future forms such as *John will come tomorrow*, *John is going to come tomorrow*, as well as the present tense as a general or neutral tense form, as in *John comes tomorrow* (contrast with anecdotal narrative, such as "You know something? This morning Julie comes in and there's this pigeon sitting on her desk ...").
There are other intermediate cases which sometimes present difficulties in classification and where the solution is not always immediately obvious:

- Are the degrees of comparison of adjectives *inflections* or *derivations* (i.e. positive *loud*, comparative *louder*, superlative *loudest*)?
- Are participles *inflected forms* of verbs or *derivations* as deverbal adjectives?
- Are infinitives of verbs *inflected forms* or *derivations* (cf. *They want an answer* vs. *They want to know*; *To be or not to be*; *He begged to come*, *He decided to come*)?

Traditional treatments often treat these forms together with inflections, presumably because of their regularity and the involvement of suffixation. They are generally better treated as derivations, however, because they have different syntactic distributions from other inflections of the same stems, and may be additionally inflected as adjectives or nouns (cf. the orthographic form of the perfect participle in French with *être* verbs: *Elle est venue – She has come.*

2.8.2 Applications of morphology

Morphological structuring is useful for the following tasks:

- The treatment of large vocabularies for speech recognition and synthesis by means of rule-based generation of inflected forms from stems.
- The prediction of new (unattested, unknown) words for speech recognition on the basis of known principles of word composition, and known attested parts of words.
- Rule-based assignment of stress patterns.
- Word recognition by stem spotting.
- Construction of subword language models for speech recognition.

There are two main ways of structuring words internally into word subunits (word constituents):

1. SEMANTIC ORIENTATION. On morphological grounds, word forms may be decomposed into smaller meaningful units, the smallest of which are morphs, the phonological forms of morphemes; an intermediate unit between the morph and the word form is the stem.
2. PHONOLOGICAL ORIENTATION. On phonological grounds, word forms may be decomposed into smaller pronunciation units, the smallest of which are phonemes; an intermediate pronunciation unit is the syllable.

It is important to note that *decomposition into syllables is not isomorphic with decomposition into morphs.* For example, *phonological* has the syllable structure /fɔ . nə . lɔ . ʤɪ . kəl/ and the morph structure /fɔn + ə+ lɔʤɪk + əl/, which are quite different from each other.
In addition to phonological decomposition, in the written mode word forms may be decomposed into smaller spelling units, graphemes, each consisting of one or more characters. An intermediate orthographic unit is the orthographic break (orthographic syllable), which is in general only needed for splitting words at line-breaks and does not correspond closely to either syllable or morph boundaries but combines phonological, morphological and orthographic criteria.
It has already been noted that in many languages, syllables and morphs do not always coincide; morphs may be smaller than or larger than syllables.
For the core requirements of speech recognition, in which a closed vocabulary of attested fully inflected word forms is generally used, morphological structuring is not necessary. Phonological structuring into

syllables, demisyllables, diphone sequences or phonemes is widely used in order to increase statistical coverage and to capture details of pronunciation (cf. Browman 1980; Ruske and Schotola 1981; Ruske 1985).

A brief outline of the main concepts in morphology, as they affect spoken language lexica will be useful in developing spoken language lexica (for more detail a textbook in linguistics should be consulted, e.g. Akmajian (1984)):

MORPHOLOGY: Morphology is the definition of the composition of words as a function of the meaning, syntactic function, and phonological or orthographic form of their parts. The morphology of spoken language is fundamentally the same as the morphology of written language in respect of meaning, syntactic function, and the combinability of *morphemes.* It differs in respect of morphophonological alternations, which differ from spelling alternations, and word prosody (for instance word stress patterns). General definitions are given here; examples are given below.

Morphotactics (word syntax) is the definition of the composition of words as a function of the forms of their parts.

Inflection is that part of morphology which deals with the adaptation of words to their contexts within sentences: on the basis of *agreement* (*congruence*), e.g. between subject and verb.

Word formation is that part of morphology which deals with the construction of words from smaller meaningful parts.

Derivation is that part of word formation which deals with the construction of words by the concatenation of stems with affixes (prefixes and suffixes).

Compounding (composition) is that part of word formation which deals with the construction of words by concatenating words or stems.

SIMPLE MORPHOLOGICAL UNITS: Traditional terminology varies in this area. A standard but incomplete definition of a *morpheme*, for instance, is that it is "the minimal meaning-bearing unit of a language". This definition is not entirely satisfactory, however, and for present purposes the sign-based model and the unit of *word* will be used as the starting point.

A *morpheme* is the smallest abstract sign-structured component of a word, and is assigned representations of its meaning, distribution and surface (orthographic and phonological) properties. More informally, morphemes are parts of words defined by criteria of form, distribution and meaning; i.e. they have meanings and are realised by orthographic or phonological forms (morphs). They have no internal morphological structure.

Traditionally, the two main kinds of morpheme are:

- *Lexical morphemes*, characterised by membership of a large, potentially open class, with meanings such as properties and roles of objects, states and events.
- *Grammatical morphemes*, characterised by membership of a closed class, defined by their distribution with respect to larger units such as sentences or complex words (e.g. inflectional and derivational endings; function words such as prepositions, articles).

Morphs are, in traditional linguistics, the orthographic or phonological forms (realisations) of morphemes. *Orthographic morphs* consist of graphemes (either single letters or fixed combinations of letters); in traditional phonology, *phonological morphs* consist of phoneme sequences with a prosodic pattern (e.g. word stress).

Roots or *bases* (lexical morphs) are the morphs which realise lexical morphemes and inflectable grammatical morphemes, and function as the smallest type of *stem* in *derivation* and *compounding. Affixes (prefixes, suffixes)* are morphs which realise the inflectional and derivational beginnings and endings of words.

A *free morph* is a morph which can occur on its own with no affixes or prosodic modifications as a separate word; a *bound morph* is a morph (generally an affix) which always occurs together with at least one other morph (typically a *stem* in the same word.

COMPLEX MORPHOLOGICAL UNITS: The structure of words is, like the structure of sentences, defined recursively, since the vocabulary of a language (including new coinages) is potentially unlimited. The functional and formal classification of morphological word structure (compounding and derivation, see above) takes this into account. Where 'out of vocabulary words' are likely to be encountered, *morphotactic rules* and a *morphological parser* or *morphological generator* may be required in order to supplement the lexicon. The condition of recursive structure does not apply to inflection, which, given a finite set of stems, defines a finite set of fully inflected word forms (in agglutinative languages possibly an extremely large finite set):

INFLECTIONAL AFFIXATION: A word (fully inflected word) is a stem morphologically concatenated with a full set of inflectional affixes, e.g. English *algorithm* + *s* = *algorithms* or German *ge* + *segn* + *et* + *en* 'blessed' (plural participle or adjective).

DERIVATIONAL AFFIXATION: A stem is

- either a root (i.e. lexical morph), e.g. *tree*, *algorithm*

- or a stem morphologically concatenated with a derivational affix, e.g. *algorithm* + *ic*, *algorithm* + *ic* + *al* + *ly*, *non* + *algorithm* + *ic* + *al* + *ly*, etc.

COMPOUNDING: A compound word is a word morphologically concatenated with a word or a stem.

MORPHOPHONOLOGICAL AND ORTHOGRAPHIC ALTERNATIONS: The operation of *morphological concatenation* is defined for present purposes as "concatenation and modification of segments at morph boundaries by boundary phenomena." The details of pronunciation and spelling are altered in morphologically complex items. An example of morphophonological alternation is /f/ – /v/ in *knife* /naɪf/ – /naɪvz/. An example of orthographic alternation is *y* – *i* – *ie* in *fly*, *flier*, *flies*. These alternants can be described by rules:

1. Morphophonological rules are rules (analogous to spelling rules) which describe morphophonological alternations, i.e. the differences between pronunciations of parts of composite words and pronunciations of corresponding parts of simplex words.
2. Spelling rules are rules which describe spelling alternations, i.e. the differences between spellings of parts of composite words and the spellings of corresponding parts of simplex words.

A standard technology for formulating spelling rules and morphophonological rules is *Two-Level Morphology* (cf. Koskenniemi (1983b), Karttunen (1983); cf. Ritchie et al. (1992)).

2.9 Grammatical information: sentence syntax

2.9.1 Grammar: statistical language models

Language models are a major area of research and development, and crucial for the success of a speech recognition system.

In speech recognition systems, the mapping of the digitised acoustic forms of words on to symbolic representations for use as lexical lookup keys is performed by stochastic speech recognisers, which may incorporate information about the phonological structure of a language to a greater or lesser extent, with *word models* for matching with the acoustic analysis of the speech signal. Details of standard practice can be easily be found in the literature (cf. Waibel and Lee 1990).

In written language processing, a comparable task is Optical Character Recognition (OCR), and in particular, handwriting recognition; there is no comparable task in conventional natural language processing or computational linguistics, where letters are uniquely identified by digital codes, and dictionary access may be trivially optimised by encoding letter sequences into tries (letter trees, letter-based decision trees). However, in linguistic terms, in each case the task is the identification of word forms as lexical keys.

2.9.2 Sentence syntax information

Syntactic information is required not only for parsing into syntactic structures for further semantic processing in a speech understanding system, but also in order to control the assignment of prosodic information to sentences in *prosodic parsing* and *prosodic synthesis.*

Syntactic information is defined as information about the distribution of a word in syntactic structures. This is a very common, indeed "classical", but specialised use of the words "syntax" and "syntactic" to pertain to phrasal syntax, i.e. the structure of sentences. Other more general uses of the terms for linguistic units which are larger or smaller than sentences are increasingly encountered, such as "dialogue syntax", "word syntax" (for morphotactics within morphology).

Within this classical usage, the term *syntax* is sometimes opposed to the term *lexicon*; the term *grammar* is sometimes used to mean syntax, but sometimes includes both phrasal syntax and the lexicon.

Strictly speaking, a stochastic language model is a probabilistic sentence syntax, since it defines the distribution of words in syntactic structures. However, the notion of syntactic structure used is often rather elementary, consisting of a short fixed-length substring or window over word strings, with length two (bigram) or three (trigram). It is also used with quite a different function from the classical combination of sentence syntax and sentence parser.

Sentence syntax defines the structure of a (generally unlimited) set of *sentences.* Syntactic lexical information is traditionally divided into information about *paradigmatic* (classificatory; disjunctive; element-class, subclass-superclass) and *syntagmatic* (compositional; conjunctive; part-whole) relations. The informal definitions of these terms in linguistics textbooks are often unclear, metaphorical and inconsistent. For instance, temporally parallel information about the constitution of phonemes in terms of distinctive features is sometimes regarded as paradigmatic (since features may be seen as intensional characterisations of a class of phonemes) and sometimes as syntagmatic (since the phonetic events corresponding to features occur together to constitute a phoneme as a larger whole). The relation here is analogous to the relation between intonation and sentences, which are also temporally parallel, and in fact treated in an identical fashion in contemporary computational phonology. From a formal point of view, this is purely a matter of perspective: the internal structure of a unit (syntagmatic relations between parts of the unit) may be seen as a property of the unit (paradigmatic relation of similarity between the whole unit and other units). In lexical knowledge bases for spoken language systems it is crucial to keep questions of syntagmatic distribution and questions of paradigmatic similarity apart as two distinct and complementary aspects of structure.

The *part of speech* (*POS*, *word class*, or *category*) is the most elementary type of syntactic information. One traditional set of word classes consists of the following: Noun or Substantive, Pronoun, Verb, Adverb, Adjective, Article, Preposition, Conjunction, Interjection. POS classifications are used for tagging written corpora (texts or transcriptions), for the purpose of information retrieval or for the training of class–based statistical language models; fairly standard POS tagsets have defined for a number of *taggers* (automatic tagging software; see the results of the EAGLES Working Group on machine readable corpora).

Two main groups of POS category are generally identified:

1. *Lexical categories* are the open classes which may be extended by word formation: Noun, Verb, Adjective, Adverb.
2. *Grammatical categories* are the closed classes which express syntactic and indexical relations: Pronoun and Article (anaphoric and deictic relations), Preposition (spatial, temporal, personal relations etc.), Conjunction (propositional relations), Interjection (dialogue relations).

The granularity of classification can be reduced by grouping classes together (this particular binary division is relevant for defining stress patterns for example) or increased by defining subcategories based on the *complements* (object, indirect object, prepositional or sentential object, etc.) of words (in various terminologies: their *valency* or *subcategorisation frames*, *case frames*, transitivity properties). For further information, introductory texts on syntax, e.g. Sells (1985) or Radford (1988) may be consulted.

In theoretical and computational linguistics, grammars are classified in terms of the *Chomsky hierarchy* of formal languages which they *generate* (i.e. define), and often represented as equivalent automata. Some aspects of this classification are connected with stochastic language models. For further information, standard computer science compiler construction literature can be consulted.

3 Lexical representation and inference

3.1 Basic philosophy

The following survey covers a number of the concepts which have been prominent in linguistic discussions for the past ten to fifteen years as guidelines for constructing what might be termed a "good linguistic description", that is, in effect, "a good theory". After this discussion, the term plan is outlined.

3.2 Domains, models, theories

Linguistics is partly *doing* language description, so to speak, but also *thinking* about how to do language description. Here it is useful to start with the idea that any language description (grammar, lexicon, pronunciation or spelling guide, idiom collection, guide to usage) not only embodies a theory but *is* a theory, in a sense, with the following properties which also characterise a fully developed formal theory:

1. **Domain:** It is about a specific, quite well-defined facts of language.
2. **Model:** It is based on a specific perspective on language (e.g. as a system of signs, or a set of rules, or a hierarchy of relations).
3. **Notation:** It uses specific representations of facts as seen from a given perspective.

Such theory may not have all the properties in detail which a fully developed formal theory would have; the point is, though, that **there is no such thing as a description of language without theoretical assumptions**. In particular, the theory itself is just that subset of sentences within a particular notational system which are true with respect to the model for the domain concerned.

The notion of theory just described may be termed a *special theory*, i.e. a description of a specific language. A useful distinction is that between a special theory and a *general theory*, which specifies properties which special theories for all possible languages should have. Much of the following discussion is a this level; it is a somewhat abstract level of discussion, but ultimately based on common sense ideas about what makes for sensible procedures in producing valid descriptions of languages.

3.3 Linguistics and the lexicon: principles and rules

In respect of general and specific linguistic theories, the traditional distinction between grammars and lexica, with grammars being the main objects of interest and lexica being somewhat uninteresting compendia of pedantic details, has been replaced in linguistic theory of the past 25 years by the idea that phrases and sentences are constructed solely by combining lexical items on the basis of their lexical properties, and thus that the grammar as an autonomous component plays a secondary role, if any.

In major modern theories of grammar, and in computational formalisms, the notion of grammatical rule has been consequently been reduced to a few very general principles of compositionality which interact with the properties of lexical items to define the more complex structures of language. Grammars such as these include Chomsky's *Government and Binding (GB)*, Bresnan & Kaplan's *Lexical Functional Grammar (LFG)*, Pollard & Sag's *Head Driven Phrase Structure Grammar (HPSG)*. Linguistically oriented computational formalisms include Kay's *Functional Unification Grammar (FUG)*, Shieber's *PATR-II*, and varieties of unification-based categorial grammar (e.g. Karttunen's *Lexical Radicalism* or Steedman's *Typed Categorial Grammar*).

This kind of view has not always been held; it contrasts with older views, in which a large number of rules, often of very different types, needed to be specified in order to define syntactic structures.

While the older approaches often failed to see the wood for the trees, the newer approach is faced with the problem that the domain of language canot be adequately described as an obvious, simple, neat structure like a regularly cut jewel, but is full of odd details and apparent exceptions. It is this property which provided the challenge which led to the treatment of syntactic facts as properties of lexical items, at the one end of the scale, which are joined into composite (but simply organised) structures at the other end of the scale by very general principles.

3.4 Declarative and procedural properties of theories

When trying to find a way of discussing special theories within the framework of a general theory, problems of special theories such as the regularity and complexity of structures, and the homogeneity or heterogeneity of the operations required to define composite structures, play a prominent role. There are a number of basic distinctions which help to provide some useful structure for such discussion.

An approach which reduces the number of operations required to define complex structures as far as possible, in the extreme case just to one rule type or even one rule, is often called *declarative.* An approach at the other extreme, which uses many different kinds of rule whose interactions with one another have to be individually specified, is often called *procedural.* But there is no purely procedural or purely declarative theory: any theory has a declarative core of structure to be operated on, however simple, and any theory has at least one rule of inference to process it, however general.

Combining this distinction with basic assumptions about how a theory relates to reality (e.g. *observed* or *intuited* experience of language), we can distinguish two main kinds of preference which underlie decisions to pick the one rather than the other type of approach, either for a theory as a whole, or for a part of a theory:

1. **Empirical evidence:**
   - Observational completeness preference: Prefer the approach which expresses all the basic facts in the domain concerned.
   - Observational soundness preference: Prefer the approach which expresses only the basic facts in the domain concerned.
   - Internal evidence preference: Prefer the approach which permits the most general expression of interrelations between distributional facts about linguistic structures.
   - External evidence preference: Prefer the approach which permits the best predictions about performance in related domains (e.g. language history, language production and perception, language learning, language pathology, language teaching, language engineering software)

5. **Formal evidence:**
   - Notational completeness preference: Prefer the notation which expresses all the facts and operations which are required by the empirical model (including being logically sufficiently general).
   - Notational soundness preference: Prefer the notation which expresses only the facts and operations which are required by the empirical model (including being logically consistent).
   - Declarative simplicity preference: Prefer the approach which permits the simplest representations of linguistic facts.
   - Procedural simplicity preference: Prefer the approach which permits the simplest operations over linguistic representations.

The meaning of the empirical preferences is probably fairly obvious; the formal preferences are probably not quite so clear. These can be illustrated by examples such as the following:

- It is easier to define a very small domain (e.g. of the *John saw the man with a telescope* type), and provide a formally sound and complete theory (a so-called "toy grammar" or "toy lexicon") for its linguistic properties than to define a large domain (e.g. of the type *all the issues of "The Times" in 1994* and provide a complete and sound theory for it. It is probably impossible to provide a sound and complete theory of the competence and performance of an actual native speaker (though arbitrary idealisations of native speakers and their performance in terms of small domains are easy to imagine).
- A declaratively simple model (such as *sequences of three words in corpora*, i.e. a *trigram model* of grammar), with general and simple statistical processing principles may be preferred for finding general statistical properties of very large domains, while complex feature structures and inter-word relations may be preferred for intelligent support of word processors (e.g. spelling, grammar and style checkers); in other contexts, combinations of these strategies may be preferred.

3.5 Development of lexical syntax

This view of theory development is more comprehensive, and also more useful, than the thirty-year-old *competence-performance* distinction of Chomsky (1965), which replaced the thirty-year-older theory of linguistic behaviour developed by Bloomfield. In a sense, it combines features of the two approaches and refines them with insights from neighbouring disciplines.

In very general terms, this development has seen four phases:

1. The logical syntaxes of the sixties (categorial grammar, Montague grammar and their successors).
2. The more descriptive "lexicalist" theories of syntax, starting with Chomsky's *Aspects of the Theory of Syntax.*
3. Some theories of the eighties which try to reduce syntax to a very few very general rules of composition (preferably just one), as in GB theory, or the unification grammars in computational linguistics, or the blends of these found in LFG or HPSG.

4. The development of interestingly structured lexica using ideas from Artificial Intelligence and formal logic, as in HPSG type hierarchies and the DATR default inheritance formalism developed by Gazdar & Evans (and also widely used by Bielefeld students and project researchers).

3.6 Compilation of syntactic facts

3.6.1 Facts and theories

- Formalism: System of writing with properties such as explicitness, complexity, expressivity
- Theory: subset of sentences of a formalism which are true wrt a model; formulations of facts
- Model: objects and relations between objects - facts (formal, or empirically testable)
- Interpretation: function from formalism to model

3.6.2 Types of lexical information

1. Matrix of syntagmatic x paradigmatic facts
2. Syntagmatic facts
    - ID hierarchy level information (morpheme-derivation-compound-phrase-sentence-text)
    - ID sister-dependency information
    - Phonetic interpretation
        - LP prosodic hierarchy level information
        - LP temporal ordering information
    - Semantic interpretation
        - Quantificational and predicate-argument information
        - Indexical and modal information
3. Paradigmatic facts
    - Phonological mapping classes at each level (elementary: features & segments; compositional)
    - Semantic mapping classes at each level (semantic fields)
    - Daughter (and daughter dependency) classes at each level; head features
    - Mother, sister and sister-dependency classes at each level; note lexical projection
5. Actual facts ...

3.7 Aspects of HPSG theory and representation

3.7.1 1987: Attribute–value matrix (AVM) template

3.7.1.1 Main AVM

$$<\mathit{entry}, \left[\begin{array}{ll}
\text{PHON concat}(\ldots) & \\
\text{SYN} & \left[\begin{array}{ll}
\text{LOC} & \left[\begin{array}{ll}
\text{HEAD} & \left[\begin{array}{ll}
\text{MAJ} & \{\text{N,V,A,P,D,ADV}\} \\
\text{CASE} & \{\text{NOM,ACC,GEN}\} \\
\text{VFORM} & \{\text{FIN,BSE,PSP,PRP,PAS,INF,GER}\} \\
\text{NFORM} & \{\text{THERE,IT,NORM}\} \\
\text{PFORM} & \{\text{OF,ON,TO,FROM,}\ldots\} \\
\text{AUX} & \{+,-\} \\
\text{INV} & \{+,-\} \\
\text{PRD} & \{+,-\}
\end{array}\right] \\
\text{SUBCAT} & \left\langle \text{NP}_{\boxed{1}},\text{NP}_{\boxed{2}},\text{NP}_{\boxed{3}} \right\rangle \\
\text{LEX} & \{+,-\}
\end{array}\right] \\
\text{BIND} & \left[\begin{array}{ll}
\text{SLASH} & \{\text{ (box-indexed?) entries on subcat lists }\} \\
\text{REL} & \{\text{ variable-indices }\} \\
\text{QUE} & \{\text{ variable-indexed WH-items }\}
\end{array}\right]
\end{array}\right] \\
\text{DTRS} & \left[\begin{array}{ll}
\text{FILLER–DTR} & \ldots \\
\text{HEAD–DTR} & \ldots \\
\text{COMP–DTRS} & \langle \ldots \rangle
\end{array}\right] \\
\text{SEM} & \left[\begin{array}{ll}
\text{CONTENT} & \text{circumstance} \\
\text{INDICES} & \text{index}
\end{array}\right]
\end{array}\right] >$$

3.7.1.2 Details of semantic AVM

$$\text{circumstance} = \left\{ \text{named, conjoined, quantified} \right\}$$

$$\text{named circumstance} = \begin{bmatrix} \text{RELN} & \\ \text{AGENT} & \boxed{i} \\ \text{PATIENT} & \boxed{j} \\ \text{OBJECT} & \boxed{k} \\ \text{POL} & \left\{ \text{ONE,ZERO} \right\} \\ \ldots & \end{bmatrix}$$

$$\text{conjoined circumstance} = \begin{bmatrix} \text{CONN} & \{\text{AND,}\ldots\} \\ \text{JUNCTS} & \{\text{p,q}\} \end{bmatrix}$$

$$\text{quantified circumstance} = \begin{bmatrix} \text{QUANT} & \begin{bmatrix} \text{DET} & \{\text{FORALL}, \ldots\} \\ \text{IND} & \begin{bmatrix} \text{VAR} & \boxed{1} \\ \text{REST} & \begin{bmatrix} \text{RELN} \\ \text{INST } \boxed{1} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$$\text{index} = \left\{ \begin{bmatrix} \text{VAR} & \boxed{1} \\ \text{REST} & \begin{bmatrix} \text{RELN} & \text{NAMING} \\ \text{NAME} & \\ \text{NAMED} & \boxed{1} \\ \text{POL} & \text{ONE} \end{bmatrix} \end{bmatrix} \right\}$$

3.7.2 1987: Principles and Rules

3.7.2.1 Universal grammar

$\text{UG} = \text{P}_1 \wedge \ldots \wedge \text{P}_n$

3.7.2.2 Language–specific principles (e.g. English)

$\text{EP} = \text{P}_{n+1} \wedge \ldots \wedge \text{P}_{n+m}$

3.7.2.3 Lexical items of English

$\text{EL} = \text{L}_1 \vee \ldots \vee \text{L}_p$

3.7.2.4 Rules (e.g. of English)

$\text{ER} = \text{R}_1 \vee \ldots \vee \text{R}_q$

3.7.2.5 Particular grammar (e.g. of English)

$\text{English} = \text{P}_1 \wedge \ldots \wedge \text{P}_n \wedge \text{P}_{n+1} \wedge \ldots \wedge \text{P}_{n+m} \wedge (\text{L}_1 \vee \ldots \vee \text{L}_p \vee \text{R}_1 \vee \ldots \vee \text{R}_q)$

3.7.3 Principles

The following principles govern the sharing (and flow of) information through an AVM.

3.7.3.1 Head Feature Principle

In a headed structure, share the head features of mother and daughter:

$$\left[\text{DTRS}\ _{headed-structure}[\ ] \right] \Rightarrow$$

$$\begin{bmatrix} \text{SYN|LOC|HEAD} & \boxed{1} \\ \text{DTRS|HEAD–DTR|SYN|LOC|HEAD} & \boxed{1} \end{bmatrix}$$

3.7.3.2 Subcategorisation Principle

In a headed structure, the head daughter subcategory is the concatenation of the complements and the mother's subcategory:

$$\left[\text{DTRS}\ _{headed-structure}[\ ] \right] \Rightarrow$$

$$\begin{bmatrix} \text{SYN|LOC|SUBCAT } \boxed{2} & \\ \text{DTRS} & \begin{bmatrix} \text{HEAD–DTR|SYN|LOC|SUBCAT} & \text{append}(\boxed{1}\ , \boxed{2}) \\ \text{COMP–DTRS} & \boxed{1} \end{bmatrix} \end{bmatrix}$$

3.7.3.3 Binding Inheritance Principle

The value of a binding feature on a sign is the set union of the values on its daughters, minus those elements which have become bound.

3.7.3.4 Semantics Principle (preliminary version)

$$\left[\text{DTRS}\ {}_{headed-structure}\left[\ \right]\right] \Rightarrow$$

$$\begin{bmatrix}\text{SEM}|\text{CONT} & \boxed{1}\\ \text{DTRS}|\text{HEAD–DTR}|\text{SEM}|\text{CONT} & \boxed{1}\end{bmatrix}$$

3.7.3.5 Semantics Principle (second version)

$$\left[\text{DTRS}\ {}_{headed-structure}\left[\ \right]\right] \Rightarrow$$

$$\begin{bmatrix}\text{SEM} & \begin{bmatrix}\text{CONT} & \boxed{1}\\ \text{IND} & \text{collect–indices}(\boxed{2})\end{bmatrix}\\ \text{DTRS}\ \boxed{2} & \begin{bmatrix}\text{HEAD–DTR}|\text{SEM}|\text{CONT}\ \boxed{1}\end{bmatrix}\end{bmatrix}$$

3.7.3.6 Semantics Principle (third version)

$$\left[\text{DTRS}\ {}_{headed-structure}\left[\ \right]\right] \Rightarrow$$

$$\begin{bmatrix}\text{SEM} & \begin{bmatrix}\text{CONT} & \text{combine–semantics}(\boxed{1}\,,\boxed{2})\\ \text{IND} & \text{collect–indices}(\boxed{3})\end{bmatrix}\\ \text{DTRS}\ \boxed{2} & \begin{bmatrix}\text{HEAD–DTR}|\text{SEM}|\text{CONT} & \boxed{1}\\ \text{COMP–DTRS} & \begin{bmatrix}\text{SEM}|\text{CONT}\ \boxed{2}\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

```
combine–semantics(A,L) =
if     A has type circumstance and
       B has type quantifier
then   $\begin{bmatrix}\text{QUANT} & \text{B}\\ \text{SCOPE} & \text{A}\end{bmatrix}$
else   A
```

3.7.3.7 Semantics Principle (fourth version)

$$\left[\text{DTRS}\ {}_{headed-structure}\left[\ \right]\right] \Rightarrow$$

$$\begin{bmatrix}\text{SEM} & \begin{bmatrix}\text{CONT} & \text{combine–semantics}(\boxed{1}\,,\boxed{2})\\ \text{IND} & \text{collect–indices}(\boxed{3})\end{bmatrix}\\ \text{DTRS}\ \boxed{2} & \begin{bmatrix}\text{HEAD–DTR}|\text{SEM}|\text{CONT} & \boxed{1}\\ \text{COMP–DTRS} & \boxed{2}\end{bmatrix}\end{bmatrix}$$

Definition of ‘successively–combine–semantics’:

```
successively–combine–semantics(A,L) =
if     length(L) = 0
then   A
else   successively–combine–semantics(combine–semantics(A,SEM|CONT of first(L)), rest(L))
```

3.7.3.8 Unification of Local Principles

The general compositionality principle of defining a property of a whole as a function of the corresponding properties of its parts is defined for the whole sign by unifying the local principles:

$$\left[\text{DTRS}\ {}_{headed-structure}\left[\ \right]\right] \Rightarrow$$

$$\begin{bmatrix}\text{SYN}|\text{LOC} & \begin{bmatrix}\text{HEAD} & \boxed{1}\\ \text{SUBCAT} & \boxed{3}\end{bmatrix}\\ \text{SEM} & \begin{bmatrix}\text{CONT} & \text{combine–semantics}(\boxed{5}\,,\boxed{2})\\ \text{IND} & \text{collect–indices}(\boxed{4})\end{bmatrix}\\ \text{DTRS} & \begin{bmatrix}\text{HEAD–DTR} & \begin{bmatrix}\text{SYN}|\text{LOC} & \begin{bmatrix}\text{HEAD} & \boxed{1}\\ \text{SUBCAT} & \text{append}(\boxed{2}\,,\boxed{3})\end{bmatrix}\\ \text{SEM}|\text{CONT} & \boxed{5}\end{bmatrix}\\ \text{COMP–DTRS}\ \boxed{2} & \end{bmatrix}\end{bmatrix}$$

3.7.4 Rules

Grammar rules are partially specified phrasal signs.

3.7.4.1 Rule 1

Phrasal (nonlexical) signs in English have one complement daughter:

$$\begin{bmatrix} \text{SYN}|\text{LOC}|\text{SUBCAT} & \langle\ \rangle & \\ \text{DTRS} & & \begin{bmatrix} \text{HEAD–DTR}|\text{SYN}|\text{LOC}|\text{LEX} & - \\ \text{COMP–DTRS} & \langle\ [\ ]\ \rangle \end{bmatrix} \end{bmatrix}$$

Unification of Rule 1 with head and subcat principles:

$$\begin{bmatrix} \text{SYN}|\text{LOC} & \begin{bmatrix} \text{HEAD} & \boxed{1} \\ \text{SUBCAT} & \langle\ \rangle \end{bmatrix} \\ \text{DTRS} & \begin{bmatrix} \text{HEAD–DTR}|\text{SYN}|\text{LOC} & \begin{bmatrix} \text{HEAD} & \boxed{1} \\ \text{SUBCAT} & \langle \boxed{2} \rangle \\ \text{LEX} & - \end{bmatrix} \\ \text{COMP–DTRS} & \langle \boxed{2} \rangle \end{bmatrix} \end{bmatrix}$$

Unification with English Constituent Ordering Principle (note the ordering of COMP before HEAD in this analysis):

$$\begin{bmatrix} \text{PHON} & \text{concat}(\boxed{3}\,,\boxed{4}) \\ \text{SYN}|\text{LOC} & \begin{bmatrix} \text{HEAD} & \boxed{1} \\ \text{SUBCAT} & \langle\ \rangle \end{bmatrix} \\ \text{DTRS} & \begin{bmatrix} \text{HEAD–DTR} & \begin{bmatrix} \text{PHON} & \boxed{4} \\ \text{SYN}|\text{LOC} & \begin{bmatrix} \text{HEAD} & \boxed{1} \\ \text{SUBCAT} & < \boxed{2} > \\ \text{LEX} & - \end{bmatrix} \end{bmatrix} \\ \text{COMP–DTRS} & \langle \boxed{2}\ [\boxed{3}]\ \rangle \end{bmatrix} \end{bmatrix}$$

3.7.4.2 Rule 2

$$\begin{bmatrix} \text{SYN}|\text{LOC}|\text{SUBCAT} & \langle\ [\ ]\ \rangle \\ \text{DTRS}|\text{HEAD–DTR}|\text{SYN}|\text{LOC} & \begin{bmatrix} \text{HEAD}|\text{INV} & - \\ \text{LEX} & - \end{bmatrix} \end{bmatrix}$$

3.7.4.3 Rule 3

$$\begin{bmatrix} \text{SYN}|\text{LOC}|\text{SUBCAT} & \langle\rangle \\ \text{DTRS}|\text{HEAD–DTR}|\text{SYN}|\text{LOC} & \begin{bmatrix} \text{HEAD}|\text{INV} & + \\ \text{LEX} & + \end{bmatrix} \end{bmatrix}$$

Unification of Rule 3 with head, subcat and ECO:

$$\begin{bmatrix} \text{PHON} & \text{concat}(\boxed{4}\,,\boxed{5}\,,\boxed{6}) \\ \text{SYN}|\text{LOC} & \begin{bmatrix} \text{HEAD} & \boxed{1} \\ \text{SUBCAT} & \langle\ \rangle \\ \text{LEX} & - \end{bmatrix} \\ \text{DTRS} & \begin{bmatrix} \text{HEAD–DTR} & \begin{bmatrix} \text{PHON} & \boxed{4} \\ \text{SYN}|\text{LOC} & \begin{bmatrix} \text{HEAD} & \boxed{1}\ \text{headfeatures} \\ \text{SUBCAT} & < \boxed{2}\,,\boxed{3} > \\ \text{LEX} & + \end{bmatrix} \end{bmatrix} \\ \text{COMP–DTRS} & \langle \boxed{2}\ [\text{PHON}\ \boxed{6}]\ ,\ \boxed{3}\ [\text{PHON}\ \boxed{5}]\ \rangle \end{bmatrix} \end{bmatrix}$$

$$\text{headfeatures:} \begin{bmatrix} \text{MAJ} & \text{V} \\ \text{VFORM} & \text{FIN} \\ \text{INV} & + \\ \text{AUX} & + \end{bmatrix}$$

3.7.5 Temporal order of constituents

3.7.5.1 Constituent order principle

$$_{phrasal-sign}[\ ] \Rightarrow$$

$$\begin{bmatrix} \text{PHON order–constituents}(\boxed{1}) \\ \text{DTRS}\ \boxed{1} \end{bmatrix}$$

3.7.5.2 Linear Precedence Constraint 1 (LP1)

$\text{HEAD}\big[\text{LEX} +\big] < [\ ]$

3.7.5.3 Linear Precedence Constraint for Head–Final Languages

$[\ ] < \text{HEAD}\big[\text{LEX} +\big]$

3.7.5.4 LP2 (first formulation)

$\text{COMPLEMENT} \ll \text{COMPLEMENT}$
in case the lhs is less oblique than the rhs.

3.7.5.5 LP2 (second formulation)

$\text{COMPLEMENT} \ll \text{COMPLEMENT}\big[\text{LEX} -\big]$

3.7.5.6 Focus Rule (LP3)

$\big[\text{MAJ } \neg\text{N}\big] < \big[\text{FOCUS} +\big]$

3.7.5.7 LP2 (final formulation)

$\text{COMPLEMENT}\big[\text{MAJ } \neg\text{V}\big] \ll \big[\text{LEX} -\big]$

3.7.5.8 Scrambling Principle

$$\begin{bmatrix} \text{PHON interleave–constituents}(\ \boxed{1}\ ) \\ \text{DTRS } \boxed{1} \end{bmatrix}$$

3.8 Comparison of AVM versions

3.8.1 1987

$$\begin{bmatrix} \text{PHON} & \\ \text{SYN} & \begin{bmatrix} \text{LOC} & \begin{bmatrix} \text{HEAD} & \dots \\ \text{LEX} & \dots \\ \text{SUBCAT} & \dots \end{bmatrix} \end{bmatrix} \\ \text{DTRS} & \\ \text{SEM} & \begin{bmatrix} \text{CONTENT} & \dots \\ \text{INDICES} & \dots \end{bmatrix} \end{bmatrix}$$

3.8.2 1994

$$\begin{bmatrix} \text{PHON} & \\ \text{SYNSEM} & \begin{bmatrix} \text{LOC} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \text{HEAD} & \dots \\ \text{SUBCAT} & \dots \end{bmatrix} \\ \text{CONTENT} \dots & \\ \text{CONTEXT} \dots & \end{bmatrix} \\ \text{NONLOC} \dots & \end{bmatrix} \\ \text{DTRS} & \\ \text{QSTORE} & \end{bmatrix}$$

3.8.3 1996

$$\begin{bmatrix} \text{PHON} & \text{(apparently non–existent)} \\ \text{SYN} & \begin{bmatrix} \text{HEAD} \dots & \\ \text{VAL} & \begin{bmatrix} \text{SPR} & \langle\ \rangle \\ \text{COMPS} & \langle\ \rangle \end{bmatrix} \\ \text{GAP} & \dots \end{bmatrix} \\ \text{SEM} & \big[\text{INDEX } i\big] \\ \text{ARG–ST} & \big[\boxed{1}\,,\boxed{2}\big] \end{bmatrix}$$

3.9 Toward operational model semantics for DATR

DATR is a widely used lexicon representation language with very simple syntax and unexpectedly complex semantics. Rather than dealing with it in depth here, an operational model semantics for DATR in Prolog will be described, followed by an application to the description of compound noun phrases in German.

```
/*
---- This is an executable Prolog file --------


MINIDATR: A minimal core DATR engine in Prolog

          Dafydd Gibbon

           U Bielefeld
     gibbon@spectrum.uni-bielefeld.de

           August 1993
Bug removed September 1996

Notice: This document contains draft information from
        a section of a preliminary version of a
        VERBMOBIL deliverable (TP 5.3-P1).
        It is distributed in this form to assist
        partners in advance planning.


This document contains a simple version of
the core DATR inference engine in Prolog in order
to illustrate the principles of DATR inference to
Prolog programmers. Note that in minor details
it departs slightly from DATR conventions:
- nonstandard nodenames are permitted;
- the knowledge base must be pre-sorted to permit
  'longest path first' inference;
- queries include the theory name.

Note also that this is not a directly usable
implementation: there is no user interface, no
DATR-Prolog interpreter, no DATR-specific trace
or debugging, no attention paide to efficiency,
etc. The aim is to provide a minimal 'core DATR
standard inference' interpreter in logical style.


1 Illustration of a DATR theory: a 'microlexicon'

MINILEX.DTR

Tablecloth: <>            == Compound
            <ilex>        == lemma
            <relation>    == (for covering)
            <modifier>    == "Table:<>"
            <head>        == "Cloth:<>".

Table:      <>            == Simplex
            <ilex>        == lemma
            <meaning>     == (horizontal surface to put things on)
            <orthography> == (t a b l e).

Cloth:      <>            == Simplex
            <ilex>        == lemma
            <meaning>     == (variety of textile)
            <orthography> == (c l o t h).

Compound:   <>            == Word
            <ilex>        == generalisation
            <type>        == compound
            <meaning>     == ("<head meaning>" "<relation>"
                              "<modifier meaning>")
            <orthography> == ("<modifier orthography>" "<head orthography>").

Simplex:    <>            == Word
            <ilex>        == generalisation
            <type>        == simplex.

Word:       <ilex>        == generalisation
            <type>        == word.


Theorems:

Tablecloth:<relation>=(for covering).
Tablecloth:<meaning>=(variety of textile for covering horizontal surface to
                     put things on).
Tablecloth:<orthography>=(t a b l e c l o t h).
Table:<orthography>=(t a b l e).
Table:<relation>=undefined.


2 A Prolog translation of MINILEX.DTR: MINILEX.PRO knowledge base.

Note the 'longest path first' reordering of the theory in Prolog.

Tablecloth: <>            == Compound
            <ilex>        == lemma
            <relation>    == (for covering)
            <modifier>    == "Table:<>"
            <head>        == "Cloth:<>".
*/
datr_sentence(minilex,'Tablecloth',[ilex],[lemma]).
datr_sentence(minilex,'Tablecloth',[relation],[for,covering]).
datr_sentence(minilex,'Tablecloth',[modifier],[[gnp,'Table',[]]]).
datr_sentence(minilex,'Tablecloth',[head],[[gnp,'Cloth',[]]]).
datr_sentence(minilex,'Tablecloth',[],[[ln,'Compound']]).
/*
Table:      <>            == Simplex
            <ilex>        == lemma
            <meaning>     == (horizontal surface to put things on)
            <orthography> == (t a b l e).
*/
datr_sentence(minilex,'Table',[ilex],[lemma]).
datr_sentence(minilex,'Table',[meaning],[horizontal,surface,to,put,things,on]).
datr_sentence(minilex,'Table',[orthography],[t,a,b,l,e]).
datr_sentence(minilex,'Table',[],[[ln,'Simplex']]).
/*
Cloth:      <>            == Simplex
            <ilex>        == lemma
            <meaning>     == (variety of textile)
            <orthography> == (c l o t h).
*/
datr_sentence(minilex,'Cloth',[ilex],[lemma]).
datr_sentence(minilex,'Cloth',[meaning],[variety,of,textile]).
datr_sentence(minilex,'Cloth',[orthography],[c,l,o,t,h]).
datr_sentence(minilex,'Cloth',[],[[ln,'Simplex']]).
/*
Compound:   <>            == Word
            <ilex>        == generalisation
            <type>        == compound
            <meaning>     == ("<head meaning>" "<relation>"
                              "<modifier meaning>")
            <orthography> == ("<modifier orthography>" "<head orthography>").
*/
datr_sentence(minilex,'Compound',[ilex],[generalisation]).
datr_sentence(minilex,'Compound',[type],[complex]).
datr_sentence(minilex,'Compound',[meaning],
   [[gp,[head,meaning]],[gp,[relation]],[gp,[modifier,meaning]]]).
datr_sentence(minilex,'Compound',[orthography],
   [[gp,[modifier,orthography]],[gp,[head,orthography]]]).
datr_sentence(minilex,'Compound',[],[[ln,'Word']]).
/*
Simplex:    <>            == Word
            <ilex>        == generalisation
            <type>        == simplex.
*/
datr_sentence(minilex,'Simplex',[ilex],[generalisation]).
datr_sentence(minilex,'Simplex',[type],[simplex]).
datr_sentence(minilex,'Simplex',[],[[ln,'Word']]).
/*
Word:       <ilex>        == generalisation
            <type>        == word.
*/
datr_sentence(minilex,'Word',[ilex],[generalisation]).
datr_sentence(minilex,'Word',[type],[word]).

minilex(Node,Path,Value) :- datr(minilex,Node,Path,Value).
/*


3 A DATR test theory, MINITEST.DTR

This theory illustrates the seven cases of DATR standard
inference. The relevant theorems are the following:

A:<> = (via node A via node B via node C undefined).
A:<1> = (via node A Rule 1).
A:<2> = (via node A Rule 2).
A:<3> = (via node A Rule 3).
A:<4> = (via node A Rule 4).
A:<5> = (via node A via node C Rule 5).
A:<6> = (via node A Rule 6).
A:<7> = (via node A Rule 7).
A:<1 2> = (path <1 2> extends path <1>).
A:<nest a> = (via node A nested global path with a).
A:<nest b> = (via node A nested global path with rubbish).

The MINITEST.DTR theory:

A:<>             == (via node 'A' B)
  <1>            == <one>
  <2>            == <two>
  <3>            == <three>
  <4>            == <four>
  <5>            == <five>
  <6>            == <six>
  <7>            == <seven>
   <seventh>       == 'Rule 7'
   <1 2>           == (path '<1 2>' extends path '<1>')
   <param>         == alpha.

B:<>              == (via node 'B' C)
   <one>           == 'Rule 1'
   <two>           == C:<second>
   <three>         == C
   <four>          == <fourth>
   <five>          == "C:<fifth>"
   <six>           == "C"
   <seven>         == "<seventh>"
   <fourth>        == 'Rule 4'
   <nest>          == <elsif "<param>">
   <elsif alpha a> == 'nested global path with a'
   <elsif>         == 'nested global path with rubbish'.

C:<>              == (via node 'C' D)
   <6>             == 'Rule 6'
   <second>        == 'Rule 2'
   <three>         == 'Rule 3'
   <fuenf>         == 'Rule 5'.

D:<>              == undefined
   <fifth>         == "<fuenf>".
```

4 A BNF description of core DATR syntax

```
<theory>     ::= <sentence> | <sentence> <theory>
<sentence>   ::= <node> : <equations>
<equations>  ::= . | <equation>
<equation>   ::= <lhs> == <rhs>

<lhs>        ::= "<" <atomseq> ">"
<atomseq>    ::= nullseq | <atom> <lhseq>

<rhs>        ::= <valseq> | ( <valseq> )
<valseq>     ::= nullseq | <val> <valseq>
<val>        ::= atom | <descriptor> | " <descriptor> "
<descriptor> ::= <node> : <path> | <node> | <path>
<path>       ::= "<" <valseq> ">"

<atom>       ::= <a_char> <s_seq> | ' <charseq> '
<node>       ::= <n_char> <s_seq>

<charseq>    ::= nullseq | <char> <charseq>
<s_seq>      ::= nullseq | <s_char> <s_seq>

<res_char>   ::= {:, <, >, =, (, ), ", .}
<char>       ::= {char(0),...,char(127)}
<p_char>     ::= {char(33),...,char(127)} | \ <char>
<s_char>     ::= <p_char> - <res_char>
<n_char>     ::= {A,...,Z}
<a_char>     ::= <s_char> - <n_char>
```

5 MINITEST.DTR to MINITEST.PRO translation (outline)

```
<sentence>   ::= datr_node(<theory>,<node>,<lhs>,<rhs>).
<lhs>        ::= Prolog list of atoms, perhaps empty,
                  e.g. [], [a], [attribute,list]
<rhs>        ::= Prolog list of <descriptor> expressions, perhaps empty,
                  e.g. [], [a], [aa,bb], etc.
<descriptor> ::= expression of one of the following types,
               corresponding to each of the 7 DATR inference rules as a
               Prolog atom or a tagged list:
               (1) <atom>
               (2) [lnp,<node>,<path>] for local node-path
               (3) [ln,<node>] for local node
               (4) [lp,<path>] for local path
               (5) [gnp,<node>,<path>] for global node-path
               (6) [gn,<node>] for global node
               (7) [gp,<path>] for global path
<node>       ::= <atom>
<atom>       ::= Prolog atomic symbol
<path>       ::= <rhs>
```

6 Illustration of DATR-Prolog translation for MINITEST.DTR

This line-by-line illustration includes "longest path first" pre-sorting:

```
/* A:<1 2>          == (’ path <1 2> extends path <1>’).        */
datr_sentence(minitest,’A’,[1,2],[’ path <1 2> extends path <1>’]).
/* A:<1>            == <one>.                                   */
datr_sentence(minitest,’A’,[1],[[lp,[one]]]).
/* A:<2>            == <two>.                                   */
datr_sentence(minitest,’A’,[2],[[lp,[two]]]).
/* A:<3>            == <three>.                                 */
datr_sentence(minitest,’A’,[3],[[lp,[three]]]).
/* A:<4>            == <four>.                                  */
datr_sentence(minitest,’A’,[4],[[lp,[four]]]).
/* A:<5>            == <five>.                                  */
datr_sentence(minitest,’A’,[5],[[lp,[five]]]).
/* A:<6>            == <six>.                                   */
datr_sentence(minitest,’A’,[6],[[lp,[six]]]).
/* A:<7>            == <seven>.                                 */
datr_sentence(minitest,’A’,[7],[[lp,[seven]]]).
/* A:<seventh>      == ’Rule 7’.                                */
datr_sentence(minitest,’A’,[seventh],[’ Rule 7’]).
/* A:<param>        == alpha.                                   */
datr_sentence(minitest,’A’,[param],[alpha]).
/* A:<>             == (via node ’A’ B).                        */
datr_sentence(minitest,’A’,[],[’ via node A’,[ln,’B’]]).

/* B:<one>          == ’Rule 1’.                                */
datr_sentence(minitest,’B’,[one],[’ Rule 1’]).
/* B:<two>          == C:<second>.                              */
datr_sentence(minitest,’B’,[two],[[lnp,’C’,[second]]]).
/* B:<three>        == C.                                       */
datr_sentence(minitest,’B’,[three],[[ln,’C’]]).
/* B:<four>         == <fourth>.                                */
datr_sentence(minitest,’B’,[four],[[lp,[fourth]]]).
/* B:<five>         == "C:<fifth>".                             */
datr_sentence(minitest,’B’,[five],[[gnp,’C’,[fifth]]]).
/* B:<six>          == "C".                                     */
datr_sentence(minitest,’B’,[six],[[gn,’C’]]).
/* B:<seven>        == "<seventh>".                             */
datr_sentence(minitest,’B’,[seven],[[gp,[seventh]]]).
/* B:<fourth>       == ’Rule 4’.                                */
datr_sentence(minitest,’B’,[fourth],[’ Rule 4’]).
```

```
/* B:<nest>           == <elsif "<param>">.                        */
datr_sentence(minitest,’B’,[nest],[[lp,[elsif,[gp,[param]]]]]).
/* B:<elsif alpha a> == ’nested global path with a’.             */
datr_sentence(minitest,’B’,[elsif,alpha,a],[’ nested global path with a’]).
/* B:<elsif>          == ’nested global path with rubbish’.       */
datr_sentence(minitest,’B’,[elsif],[’ nested global path with rubbish’]).
/* B:<>               == (via node ’B’ C).                         */
datr_sentence(minitest,’B’,[],[’ via node B’,[ln,’C’]]).

/* C:<6>              == ’Rule 6’.                                 */
datr_sentence(minitest,’C’,[6],[’ Rule 6’]).
/* C:<second>         == ’Rule 2’.                                 */
datr_sentence(minitest,’C’,[second],[’ Rule 2’]).
/* C:<three>          == ’Rule 3’.                                 */
datr_sentence(minitest,’C’,[three],[’ Rule 3’]).
/* C:<fuenf>          == ’Rule 5’.                                 */
datr_sentence(minitest,’C’,[fuenf],[’ Rule 5’]).
/* C:<>               == (via node ’C’ D).                         */
datr_sentence(minitest,’C’,[],[’ via node C’,[ln,’D’]]).

/* D:<fifth>          == "<fuenf>".                                */
datr_sentence(minitest,’D’,[fifth],[[gp,[fuenf]]]).
/* D:<>               == undefined.                                */
datr_sentence(minitest,’D’,[],[’ undefined’]).

minitest(Node,Path,Value) :- datr(minitest,Node,Path,Value).
/*


Outline of the MINIDATR.PRO inference engine

datr(Theory,Node,Path,Value).
       - defines initial DATR global and local query environments
       - 1 clause
    datr_connect(Theory,Gnode,Gpath,Lnode,Lpath,Value).
       - accesses DATR theory with query environments
       - 1 clause
    datr_rhs(Gnode,Gpath,Lnode,Prefix,Suffix,Rhs,Value).
       - evaluates RHS of DATR equations as lists
       - 2 clauses
    datr_rule(Gnode,Gpath,Lnode,Prefix,Suffix,Descriptor,Value).
       - 7 DATR inference rules
       - 7 clauses, for evaluation of atoms and of local and
         global node-path, node, and path descriptors.
    append(Prefix,Suffix,Whole).
       - expresses RHS sequence evaluation and path extension
       - 2 clauses (standard definition).

The following queries illustrate standard DATR inference.

   Query                     Response

minitest(’A’,[],X).        X = [ via node A, via node B, via node C,undefined]
minitest(’A’,[1],X).       X = [ via node A, Rule 1]
minitest(’A’,[2],X).       X = [ via node A, Rule 2]
minitest(’A’,[3],X).       X = [ via node A, Rule 3]
minitest(’A’,[4],X).       X = [ via node A, Rule 4]
minitest(’A’,[5],X).       X = [ via node A, via node C, Rule 5]
minitest(’A’,[6],X).       X = [ via node A, Rule 6]
minitest(’A’,[7],X).       X = [ via node A, Rule 7]
minitest(’A’,[1,2],X).     X = [ path <1 2> extends path <1>]
minitest(’A’,[nest,a],X).  X = [ via node A, nested global path with a]
minitest('A',[nest,b],X).  X = [ via node A, nested global path with rubbish]


7 Code for the MINIDATR inference engine

Note that this minimalistic core DATR inference engine allows
nonstandard node-names, has no DATR variables, and does not
include the 'longest path first' default connection condition.
This means that the 'longest path first' ordering must be
pre-defined in the Prolog knowledge base.
*/

datr(Theory,Node,Path,Value) :-
 atomic(Theory),
 atomic(Node),
 nonvar(Path),
 atompath(Path),
 var(Value),
 datr_connect(Theory,Node,Path,Node,Path,Value),!.

atompath([]) :- !.
atompath([First|Rest]) :-
 atomic(First),
 atompath(Rest).

datr_connect(Theory,Gnode,Gpath,Lnode,Lpath,Value):-
 datr_sentence(Theory,Lnode,Prefix,Rhs),
 append(Prefix,Suffix,Lpath),!,
 datr_rhs(Theory,Gnode,Gpath,Lnode,Prefix,Suffix,Rhs,Value).


datr_rhs(Theory,Gnode,Gpath,Lnode,Prefix,Suffix,[],[]).

datr_rhs(Theory,Gnode,Gpath,Lnode,Prefix,Suffix,[First|Rest],Value) :-
 datr_rule(Theory,Gnode,Gpath,Lnode,Prefix,Suffix,First,First_value),
 append(First_value,Rest_value,Value),!,
 datr_rhs(Theory,Gnode,Gpath,Lnode,Prefix,Suffix,Rest,Rest_value).


/* DATR Inference Rule 1                                     */
datr_rule(Theory,Gnode,Gpath,Lnode,Prefix,Suffix,Expr,[Expr]) :-
 atomic(Expr).

/* DATR Inference Rule 2                                     */
datr_rule(Theory,Gnode,Gpath,Lnode,Prefix,Suffix,[lnp,Node,Path],Value) :-
 datr_rhs(Theory,Gnode,Gpath,Lnode,Prefix,Suffix,Path,Path_value),
 append(Path_value,Suffix,Extension),
 datr_connect(Theory,Gnode,Gpath,Node,Extension,Value).

/* DATR Inference Rule 3                                     */
datr_rule(Theory,Gnode,Gpath,Lnode,Prefix,Suffix,[ln,Node],Value) :-
 append(Prefix,Suffix,Extension),
 datr_connect(Theory,Gnode,Gpath,Node,Extension,Value).

/* DATR Inference Rule 4                                     */
datr_rule(Theory,Gnode,Gpath,Lnode,Prefix,Suffix,[lp,Path],Value) :-
 datr_rhs(Theory,Gnode,Gpath,Lnode,Prefix,Suffix,Path,Path_value),
 append(Path_value,Suffix,Extension),
 datr_connect(Theory,Gnode,Gpath,Lnode,Extension,Value).

/* DATR Inference Rule 5                                     */
datr_rule(Theory,Gnode,Gpath,Lnode,Prefix,Suffix,[gnp,Node,Path],Value) :-
```

```
 datr_rhs(Theory,Gnode,Gpath,Lnode,Prefix,Suffix,Path,Path_value),
 append(Path_value,Suffix,Extension),
 datr_connect(Theory,Node,Extension,Node,Extension,Value).

/* DATR Inference Rule 6                                      */
datr_rule(Theory,Gnode,Gpath,Lnode,Prefix,Suffix,[gn,Node],Value) :-
%% Bug. Removed 11.09.96. DG
%% append(Gpath,Suffix,Extension),
 datr_connect(Theory,Node,Extension,Node,Extension,Value).

/* DATR Inference Rule 7                                      */
datr_rule(Theory,Gnode,Gpath,Lnode,Prefix,Suffix,[gp,Path],Value) :-
 datr_rhs(Theory,Gnode,Gpath,Lnode,Prefix,Suffix,Path,Path_value),
 append(Path_value,Suffix,Extension),
 datr_connect(Theory,Gnode,Extension,Gnode,Extension,Value).
```

3.10 Relating AVM and DATR representations: compound nouns

3.10.1 Lexical signs and the Inheritance Lexicon

3.10.1.1 Lexical signs

What are signs, in linguistic terms? Do signs consist of other signs, in the way that sentences like *Let's listen to Charlie Byrd!* have constituents, or compound words like *mousetrap repair shop owner* are made up of other words? Or is the quality of being a sign rather a holistic one which only attaches to utterances or even dialogues in context, from 'Hi' to the entire proceedings of a business meeting? The present approach to the theory of word formation (the ILEX approach) encompasses the following assumptions about signs:[1]

1. All signs are pairs of some observable form and a meaning.
2. All signs are compositional in principle, down to their smallest phonological constituents.
3. Every language user is familiar with an inventory of more–or–less fixed signs, a *lexicon*, as well as with non–lexical, freely constructed signs.
4. Lexical signs are assigned to a scale of well–defined ranks, corresponding to linguistic levels of description from phoneme–size through morphemes, simple, derived and compound words, phrases and proverbs to ritualised exchanges, in an *idiomaticity hierarchy*; the *word* is a *basic rank*.
5. At each rank, linguistically significant generalisations are formulated in terms of inheritance relations for sets of inventorised lexical items at this rank: phonology (better, prosody) is the set of generalisations about speech sounds, morphology the set of generalisations about form and meaning of words, syntax the set of generalisations about form and meaning of phrasal idioms, and so on.
6. At any given rank, a sign has, in principle, four properties: its *surface* (physical appearance, e.g. the forms represented by the transcription /ɹˈæ.tl.sneɪk/, or the spelling *rattlesnake*), its *meaning* (its relation to the situation of use, including objects it refers to, speaker and addressee), its *category* (its co–occurrence with other signs in linguistic structures), and its *parts* (its internal structure or 'child' constituents, which are in general weighted in terms of *head* and *modifier* constituents).
7. The surface and the meaning of a sign are its *interpretative* properties, and the category and parts are its *compositional* properties.

There are interesting special cases. For example, the traditional phoneme is an inventorised item with no parts, no semantic interpretation and purely structural 'meaning'; the morph *cran* in *cranberry* has no parts at the same rank (morphology), and no semantic interpretation (except in Norfolk, where it means 'a basket of the type freshly caught herring are kept in'). Leprechaun items such as 'zero morphemes' and 'traces', for those who believe in them, have no parts and no phonetic interpretation, but a category and a semantic interpretation.

Recent work in syntax, notably within the paradigm of Head–driven Phrase Structure Grammar, HPSG (cf. Pollard and Sag (1987), Pollard and Sag (1994)), has revived a similar structuralist notion of sign to that outlined here, and formalised it as an attribute–value matrix (AVM). In this approach, a taxonomy (type hierarchy) of sign types is defined, from the most general type *sign* to the most specific types, individual words; each sign type is characterised by a set of appropriate attributes and appropriate

[1] A number of variants of the template outlined here have been known since the early nineties as the *ILEX* (*Inheritance LEXicon* or *Integrated LEXicon*) model; lexica based on the model have generally been formulated as DATR theories. Many published and unpublished ILEX/DATR 'microlexica' have been implemented on the basis of this model.

values, and generalisations over more specific sign types are expressed by *inheriting* the properties of more general sign types along the branches of the sign taxonomy.

It is not yet clear how to integrate lexical problem areas into the word and sentence oriented (albeit lexicalistic) HPSG approach. The HPSG model contains three relevant kinds of entity: a base inventory of words, lexical rules of inflection, word–formation, subcategorisation and semantic selection which define an extended inventory of words, and principles of composition linking the 'head–daughter' (head part) and the 'complement–daughters' (modifier parts) of a phrase by concatenation and unification or other appropriate operation. Problem areas for this model currently still include the following:

1. *Idioms*, which are clearly lexical signs, but not elementary ones.
2. *Sentence prosody* and *word prosody*, which are involved in compositionality, but by complex varieties of *prosodic association* and not just by *concatenation.*
3. *Compositional principles* for the morphology of inflection, derivation and compounding, including compositionality in *morphophonology* and *morphographemics.*
4. *Degrees of irregularity* in the lexicon.
5. *Degrees of compositionality* in syntax and morphotactics.
6. *Markedness relations* based on neutralisation or familiarity.
7. *Compositional lexical semantics* (hard, if lexical items have no parts).

The present study addresses these problems and proposes an integrated, sign–based solution to lexical explanations. In the following sections, an HPSG–related theoretical framework and an operational DATR model for this theory are used to describe English compounds: linguistic concepts closely related to HPSG are described and implemented with representation techniques from DATR. After a summary of the main directions in Inheritance Lexicon Theory, modelling conventions for the inheritance lexicon are characterised, a summary of lexical properties of the main types of English noun, in particular noun compounds, is given, followed by an account of the DATR lexical knowledge representation formalism. An operational DATR model for English nouns is discussed, and a sample analysis is presented. The main results and conclusions are outlined in the final section.

3.10.1.2 Inheritance Lexicon Theory

A number of approaches to lexical theory are emerging in computational linguistics which address these problems and attempt to integrate descriptions in the known lexical problem areas. A central role is played by the *inheritance lexicon paradigm*, initiated by Flickinger Flickinger (1987). Inheritance Lexicon Theory (ILT) has been developed in three main directions, each with slightly different linguistic assumptions and conventions for lexical representation.

HPSG: In the HPSG lexicon, lexical signs are represented by AVM representations and classified into types, with more specific types inheriting generalisable properties from more general types. Lexical rules project a base lexicon on to a much larger (perhaps infinite) lexicon; the rules cover lexicon extension in morphology (inflections, derivations, compounds), syntax (complex subcategories such as *passive*), semantics (selectional conditions for disambiguating polysemy).

OOL: In the Object–Oriented Lexicon (OOL) lexical items are represented as *objects* (*classes* and *instances*) in an *object hierarchy*, in which objects communicate by *message–passing*, and *methods* for handling the messages are defined for each object. More specific objects inherit general methods from more general objects, and therefore methods do not necessarily have to be fully specified for any given object. Object–oriented representations originated as a means of representing one type of semantic network in Artificial Intelligence, generally implemented as functions in LISP, but have resulted in well–known class–oriented programming languages such as SmallTalk, C++ and Java. The OOL concept was introduced by Daelemans Daelemans (1987).

DATR: DATR is a lexical knowledge representation language developed by Evans and Gazdar (summarised in Evans and Gazdar (1996)). In DATR, the basic unit is the *node* (roughly comparable with the *type* in the HPSG approach and the *object* in the OOL approach) organised into a *default inheritance hierarchy.* Each node in the hierarchy is characterised by a set of *attribute–value equations* (more precisely, equations pairing attribute paths and values), in which any *path* may only occur once, and each *value* evaluates to a sequence of atomic constituent values (possibly null). The constituent values directly specified for particular nodes may be atomic, or inherited from more general nodes. Since a node may therefore inherit values from several other, more general nodes, but only if constrained by a unique attribute, DATR is said to have *orthogonal multiple inheritance.* Default inheritance means that a value of a given attribute may be specified more than once in an inheritance path, in which case the values at lower (more specific) nodes in the hierarchy override values.[2]

[2] Representative studies using DATR models have been carried out by Gibbon on Arabic paradigms and Kikuyu tone

3.10.2 Modelling conventions for the Inheritance Lexicon

3.10.2.1 Basic modelling conventions

Traditionally, the linguistic structure of signs is characterised in terms of three basic notions: *level of representation (abstraction, description)*; *syntagmatic relation*; *paradigmatic relation.* The *level of representation (abstraction, description etc.)*, includes compositional levels of morphology, syntax, text, and interpretative levels of semantics, phonetics. At each level, structure is further defined by *syntagmatic relations*, including concepts of *dependency*, *valency* and *headedness*, and by *paradigmatic relations*, including concepts of *markedness.* Syntagmatic relations are *part–whole* and *part–part* relations and paradigmatic relations are *similarity* relations which define classes of linguistic units and oppositions between sub–classes. These notions will be characterised in more detail below.

Level of representation (abstraction, description etc.): A coherent set of descriptive categories together with methodological criteria and formal representation devices for these categories. Levels are assigned to a scale of well–defined ranks corresponding to linguistic levels of description from phoneme–like units through morpheme–like units, simple, derived and compound words, phrases, sentences (including idioms and proverbs) to ritualised exchanges. At each rank a distinction between *lexical* and *nonce (ad hoc)* items is defined, and the rank scale of lexical items constitutes an *idiomaticity hierarchy.* The *word* is a *basic rank* in the sense of Rosch's notion of basic category Rosch (1978).

A distinction is made at each rank between signs and their *co–interpretation* in terms of phonetic and orthographic *surface form* and *meaning.* The duality of co–interpretation, shared by many linguistic theories, explicates the traditional semiotic triangle in terms of a sign for which there exists on the one hand a model of surface form (sound or writing, gesture, scent etc.), and on the other hand a model of situational meaning. Whether the sign and its two types of interpretation are assigned cognitive (conceptual, mentalistic) interpretations in addition to the behavioural and observational criteria for surface (and, in part, semantic) interpretations is more a question of a linguist's epistemological stance than of direct empirical consequence.

The pair of interpretation functions co–interprets items at different ranks such as the *phoneme*, the *morpheme*, the *word*, the *sentence*, the *turn* or *dialogue contribution*, the *dialogue.* Mapping functions between ranks and rank–specific interpretative models define the overall architecture of a linguistic theory.

Syntagmatic relation: A compositional relation, definable as

1. a part–whole (dominance) relation between parent categories and child categories (constituents), for example *head–of*, *modifier–of*, or
2. a part–part relation between sibling categories, e.g. dependency or valency relations, *affix–to*, *initial*, or
3. a transitive generalisation of these simple relations to more indirect relations (e.g. *head feature projection* as a generalisation of the part–whole relation, or *SVO* surface order as a generalisation of the simple part–part relation).

A fundamental distinction between (possibly universal) *immediate dominance (ID)* or part–whole relations and (partly language specific) *linear precedence (LP)* or temporally and spatially interpretable part–part relations is made in most computational grammars. For example, the ID structure of compound words in English and French is similar, but English is 'right–headed' whereas French is 'left–headed' and uses interfixed prepositions: *peau–rouge* 'redskin', *épingle à cheveux* 'hairpin', *pain d'épice* 'gingerbread'. In the ILEX approach, the core type of syntagmatic relation is the ID relation, and the LP relation is generalised to the *quasi-linear precedence (QLP)* relation in order to include prosodic association for suprasegmentals in speech, highlights and layout in writing. The QLP relation plays a similar role in surface form interpretation to *logical form (LF)* in semantic interpretation. A distinction is therefore made between *compositional syntagmatic relations* and *interpretative syntagmatic relations*; it is the latter which generally features in traditional descriptions. In current theories of syntax, syntagmatic relations are formalised as operations of compositionality, e.g. the *slash* and *position* operations in categorial grammar, *rewrite* and *concatenation* operations in phrase structure grammar, and the ID and LP relations of unification grammar.

A straightforward definition of a syntagmatic relation is as follows:

(Gibbon (1990)), Reinhard & Gibbon on Arabic and Kikuyu (Reinhard and Gibbon (1981)), Gibbon on German compounds (Gibbon (1992b)), Cahill on morphophonology in the lexicon (Cahill (1993a)), Bleiching on German morphology and lexical prosody (Bleiching (1992b), Bleiching (1994)), Corbett & Fraser on Russian inflection (Corbett and Fraser (1995)), Bleiching, Drexel & Gibbon on German inflection (Bleiching et al. (1996b)), Gibbon, Tseng & Folikpo on Ewegbe tone (Gibbon et al. (????)).

$$\forall x, y, z\ SynRel(x,y,z) \equiv Part(x,z) \wedge Part(y,z) \wedge f(FS(x), FS(y)) = FS(z)$$

where at most one of x or y or z may remain uninstantiated, $SynRel$ is a syntagmatic relation, and FS is a feature structure (i.e. AVM). For example,

$$Spelling(jellyfish) = f_{spell}(Spelling(jelly), Spelling(fish))$$
$$Pronunciation(jellyfish) = f_{pron}(Pronunciation(jelly), Pronunciation(fish))$$
$$Meaning(jellyfish) = f_{mean}(Meaning(jelly), Meaning(fish))$$

This formula expresses *Frege's Principle (FP)* (cf. Cresswell (1973)) of compositionality, i.e. the principle that a property of the whole is a function of this property of the parts, whereby the function f may be concatenation, unification, slash cancellation, etc., depending on the formalism used. FP is generally applied only to semantic interpretation; in the present approach it is also applied to surface form interpretation. The function f_{mean} is less general in this case than the surface interpretation functions, and needs components to account for metaphor and ellipsis.

Paradigmatic relation: A generalisation relation, characterising similarity between signs in terms of one or more sign properties, defining *sets* or *classes*, *elements of sets*, and *set–subset inclusion*, with the usual set theoretic operations of union, intersection, and the formation of set theoretic relations as tuples. Sign properties are defined in terms of feature structures, and similarity is defined in terms of the subsumption ($\sqsubseteq$) operation[3]. Traditionally, paradigmatic relations define semantic fields, syntactic categories, phonological natural classes, and distributional classes of all kinds. Leaving aside some technical details, the terms used may be defined straightforwardly as follows, with feature structures representing complex lexical properties of quantifiable lexical objects, FS_i (the *subsumer*) and FS_j (the *subsumed*) are feature structures consisting of attribute–value (AV) pairs, and '$\rightarrow$' and '$\equiv$' represent conditional and biconditional propositional functions respectively:

Subsumption: $\forall\ x\ FS_i \sqsubseteq FS_j \equiv FS_j(x) \rightarrow FS_i(x)$
Paradigmatic relation: $\forall\ i,j\ ParaRel(i,j) \equiv \exists\ k\ FS_k \sqsubseteq FS_i \wedge FS_k \sqsubseteq FS_j$

Paradigmatic generalisations are expressed as inheritance relations between subclasses and classes, and among the subclasses of a given class. This concept is explained in the following sections.

3.10.2.2 Subsumption hierarchies, taxonomies and generalisation

The subsumption relation can be understood as a relation of *implication* which relates more specific to more general concepts in conceptual taxonomies. In formal terms, subsumption defines a lattice, a kind of partial ordering, which may be represented as a directed acyclic graph. The hierarchical graphs defined by subsumption need not be trees, but can be more general kinds of graph in which child nodes are *re–entrant*, i.e. a child node may have more than one parent node. However, commonly a subsumption lattice has a core tree structure, with superimposition of more than one tree, or of other cross–classifiying structures. The subsumption relation may be seen as a *generalisation* relation, in that the subsumer expresses a generalisation over the subsumed.

Examples of lexical subsumption are shown in Figure 3.1, which illustrates some of the following points:

1. The semantic properties of *horse* subsume the semantic properties of *stallion.*
2. The semantic properties {*male, animal*} subsume the semantic properties of *stallion*
3. The semantic properties of *horse* subsume the semantic properties of *mare.*
4. The phonological properties of *lamp* subsume the phonological properties of *streetlamp*
5. Heads subsume the constructions whose heads they are.
6. $\begin{bmatrix}\text{MANNER obstruent}\end{bmatrix} \sqsubseteq \begin{bmatrix}\text{MANNER obstruent}\\ \text{VOICING unvoiced}\end{bmatrix}$
7. Archiphonemes subsume their phoneme members.

3.10.2.3 Generalisation hierarchies and inheritance

If $FS_i \sqsubseteq FS_j$, as in any of the cases illustrated above, then the subsumed FS_j is redundant if all its AV pairs are completely specified. Consequently, the information in subsumer FS_i may be subtracted from FS_j, leaving a non–redundant set of AV specifications, and a redundancy rule can be formulated which will allow the 'missing' features to be inferred or 'added in'. This is standard procedure in the rule notation of generative phonology and morphology:

[3] In some sources, the symbol is reversed by analogy with the subset relation over the extensions of the feature structures.

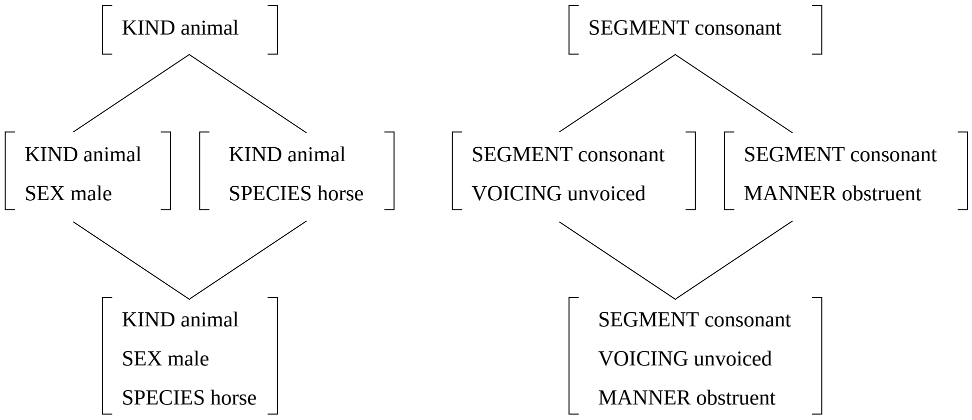

KIND animal
KIND animal SEX male
KIND animal SPECIES horse
KIND animal SEX male SPECIES horse
SEGMENT consonant
SEGMENT consonant VOICING unvoiced
SEGMENT consonant MANNER obstruent
SEGMENT consonant VOICING unvoiced MANNER obstruent

Figure 3.1: Reentrant subsumption graphs

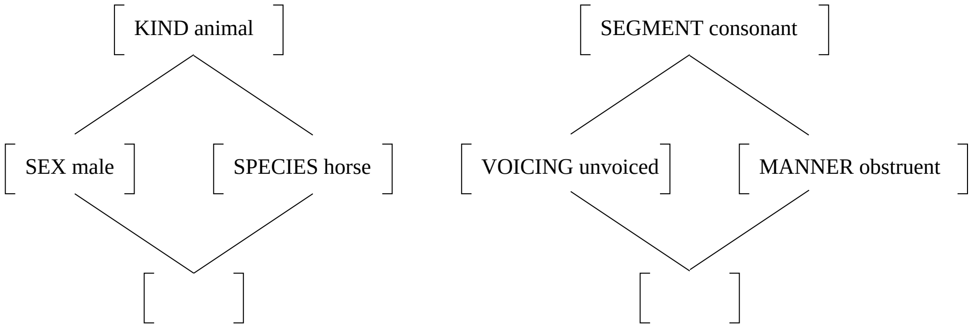


Figure 3.2: Reentrant inheritance graphs

$$\begin{array}{lll}
\text{The phonological redundancy rule:} & \begin{bmatrix}\text{MANNER obstruent}\end{bmatrix} \quad \rightarrow & \begin{bmatrix}\text{VOICING unvoiced}\end{bmatrix} \;/\; \_\; \# \\
\text{expands conventionally to:} & \begin{bmatrix}\text{MANNER obstruent}\\ \text{VOICING [ ]}\end{bmatrix} \# \rightarrow & \begin{bmatrix}\text{MANNER obstruent}\\ \text{VOICING unvoiced}\end{bmatrix} \# \\
\text{or, in terms of subsumption:} & \begin{bmatrix}\textit{archi–segment}_i\\ \text{MANNER obstruent}\end{bmatrix} \# \sqsubseteq & \begin{bmatrix}\textit{archi-segment}_j\\ \text{MANNER obstruent}\\ \text{VOICING unvoiced}\end{bmatrix} \#
\end{array}$$

The subtraction operation between a subsumed AVM_1 and a subsumer AVM_2 yields a non–redundant AVM_3 in an *inheritance* relation with AVM_2. The inheritance relation whereby AVM_3 inherits the features of AVM_2, and thereby reconstitutes AVM_1, s the inverse of the subtraction operation, and is expressed as a special case of unification: $\text{AVM}_3 = \text{AVM}_2 \sqcup \text{AVM}_3$, where $\text{AVM}_3 \sqcap \text{AVM}_2 = 0$. The generalisation (feature intersection) operator '$\sqcap$' is defined as the set of features shared by AVM_3 and AVM_2 and and the specialisation (unification) operator '$\sqcup$' is defined recursively for compatible AVMs: two attribute–value pairs unify either if the values are identical atoms, or if an attribute in one AVM is not specified in the other, or if the values of identical attributes in the AVMs unify. Under the type inheritance operation expressed by unification, the AVMs in Figure 3.2 and the AVMs in Figure 3.1 are equivalent. The elementary case of non–recursive unification has been familiar in linguistics since the introduction of the *lexical insertion* operation by Chomsky Chomsky (1965); Shieber Shieber (1986) summarises the more general unification operation used in unification grammars.

In the DATR formalism, a form of *default inheritance* is defined, in which the subsumption relation and the unification operation do not hold. Instead, there is a *default–override* relation and a *default unification* operation. In the default–override relation, a value for a given attribute may be specified more than once in the same inheritance path, and the specification of the lower (more specific) class overrides the specification of the higher (more general) class. In a famous illustration, Tweety, *qua penguin* cannot fly, but Tweety, *qua bird* can fly. Clearly, the penguin specification is more specific than the bird specification, therefore the dispositional predicate 'cannot fly' overrides the dispositional predicate 'can fly'.

In the ILEX version of Inheritance Lexicon Theory, default inheritance is used in order to explain exceptions and subregularities of this kind.

3.10.2.4 Signs, archi–signs, and generalisation over signs

The four main properties of a sign have complex values whose structure is summarised in the following nested attribute value template (with illustrative values inserted), which will be referred to as the ILEX template:

$$\left[\begin{array}{ll} \text{LEMMA} & \textit{pussy–willow} \\ \text{STRUC} & \left[\begin{array}{ll} \text{CAT} & \textit{compound\_noun} \\ \text{PARTS} & \left[\begin{array}{ll} \text{HEAD} & \textit{willow} \\ \text{MODI} & \textit{pussy} \end{array}\right] \end{array}\right] \\ \text{INT} & \left[\begin{array}{ll} \text{MEAN} & \left[\begin{array}{ll} \text{EVENT} & \textit{state} \\ \text{QUALIA} & \left[\begin{array}{l} \text{RELN}\ \textsc{resemble}(\textit{willow,pussy}) \end{array}\right] \\ \text{TECH} & \textsc{salix caprea pendula} \\ \text{INDEX} & j \end{array}\right] \\ \text{SURF} & \left[\begin{array}{ll} \text{PHON} & \text{/pʊsɪ\#wˈɪləʊ/} \\ \text{ORTH} & \text{“pussy–willow”} \end{array}\right] \end{array}\right] \end{array}\right]$$

The attributes have the following interpretations (abbreviations in parentheses):

LEMMA: Name of the lexical entry; the lowest type in the inheritance hierarchy; it can be compared with types in HPSG (except that the ILEX approach uses default inheritance lattices, while HPSG uses type subsumption lattices).

STRUCTURE (STRUC): The syntagmatic properties of the sign.

CATEGORY (CAT): The relation of a head sign to its parent and siblings (cf. HPSG 'HEAD' and 'SUBCAT' attributes).

PARTS: The constituents of a sign (cf. HPSG 'DTRS').

HEAD: The head constituent (cf. Zwicky Zwicky (1993)).

MODIFIER (MODI): The non–head constituents of a sign (cf. HPSG 'COMP'); for noun compounds, generally a single item.

INTERPRETATION (INT): The basic semiotic properties of a sign.

MEANING (MEAN): The semantic interpretation attribute.

EVENT: Taken from Generative Lexicon Theory (cf. Pustejovsky Pustejovsky (1996)).

QUALIA: Taken from Generative Lexicon Theory.

RELATION (RELN): Taken from HPSG–flavoured semantic role structure.

TECHNICAL (TECH): Indicates a technical meaning from a special sublanguage.

INDEX: Taken from HPSG–flavoured situation semantics.

SURFACE (SURF): The phonetic/orthographic interpretation attribute.

PHON: Phonetic interpretation (with prosodic association and concatenation, when represented in full detail).

ORTH: Orthographic interpretation.

Values which are shared by a class of signs (i.e. values defining paradigmatic similarity relations) may be generalised by applying the operator '⊓' to the AVMs of the signs. In this case, the values are *inherited* from the 'archi–sign' representing this class, and need not be represented explicitly for each member of the class. Inheritance therefore expresses implication, the paradigmatic relation which constitutes taxonomies. For example, *serenity* inherits certain phonological properties from the archi–sign representing the class of English words affected by tri–syllabic shortening; *bake* inherits the details of its inflections from the archi–sign representing the class of all weak verbs; *chair* inherits certain general semantic properties from the archi–sign representing all items of furniture; *surfboard* inherits compositional properties from the archi–sign representing the class containing *skateboard* and *blackboard*, and in particular it inherits 'head features' such as CAT from its HEAD PART *board*.

The inheritance of properties from (or by) a PART is commonly referred to as *feature percolation*, and defines the notion of compositionality in attribute–value terms.

The four main kinds of inheritance, which are closely related to mechanisms in the DATR lexical knowledge representation language, are listed in Table 3.1.

Orthogonal multiple inheritance simply means that the values of several different specified attributes may be inherited from different archi–signs or types, rather than from a single archi–sign for the CAT

Table 3.1: AVM inheritance operations.

Symbol:	Type of inheritance:
→	Paradigmatic inheritance from an archi–sign
←	Orthogonal multiple inheritance from an archi–sign
⇓	Syntagmatic inheritance from a PART
⇑	Lexical insertion of a property of a PART into an interpretation template (or an evaluable path)

attribute. In general, any attribute which is not explicitly specified inherits its value from the archi–sign; the notation given here permits explicit expression of this relation.

3.10.2.5 Surface compositionality and semantic compositionality

The concepts of *lexical compositionality* and *partial lexical compositionality* can now be illustrated in terms of the ILEX template (see Table 3.2). Immediate Dominance compositionality is represented by the STRUC attribute, and compositional interpretation is indicated by parentheses which represent the application of a semantic or phonetic operator (the first element in the enclosed list) to its operands (the remaining list elements). The notions SEMANTICALLY_LINK and PROSODICALLY_LINK are defined in terms of default unification. The operation of hyphenation is straightforward concatenation of the parts with an intervening hyphen, with the concatenation operation interpreted as a spatial precedence relation. compositionality is defined in general terms for all interpretative features, but each type of interpretation specifies its own operators.

Table 3.2: Paradigmatic and syntagmatic inheritance for *pussy–willow.*

(1) Lexical sign:

$$\begin{bmatrix} \text{LEMMA} & \textit{pussy–willow} \\ \text{STRUC} & \begin{bmatrix} \text{CAT} & \rightarrow \textit{compound\_noun} \\ \text{PARTS} & \begin{bmatrix} \text{HEAD} & \Downarrow \textit{willow} \\ \text{MODI} & \Downarrow \textit{pussy} \end{bmatrix} \end{bmatrix} \\ \text{INT} & \begin{bmatrix} \text{MEAN} & \begin{bmatrix} \text{QUALIA} & \begin{bmatrix} \text{RELN} & \textsc{resemble} \end{bmatrix} \\ \text{TECH} & \textsc{salix caprea pendula} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

(2) Lexical archi–sign:

$$\begin{bmatrix} \text{LEMMA} & \textit{compound\_noun} \\ \text{STRUC} & \begin{bmatrix} \text{CAT} & \rightarrow \textit{noun} \end{bmatrix} \\ \text{INT} & \begin{bmatrix} \text{MEAN} & \begin{pmatrix} \textsc{semantically\_link} \\ \Uparrow\text{INT}|\text{MEAN} \\ \Uparrow\text{STRUC}|\text{PARTS}|\text{MODI}|\text{INT}|\text{MEAN} \\ \Uparrow\text{STRUC}|\text{PARTS}|\text{HEAD}|\text{INT}|\text{MEAN} \end{pmatrix} \\ \text{SURF} & \begin{bmatrix} \text{PHON} & \begin{pmatrix} \textsc{prosodically\_link}, \\ \Uparrow\text{STRUC}|\text{PARTS}|\text{MODI}|\text{INT}|\text{SURF}|\text{PHON}, \\ \Uparrow\text{STRUC}|\text{PARTS}|\text{HEAD}|\text{INT}|\text{SURF}|\text{PHON} \end{pmatrix} \\ \text{ORTH} & \begin{pmatrix} \textit{hyphenate}, \\ \Uparrow\text{STRUC}|\text{PARTS}|\text{MODI}|\text{INT}|\text{SURF}|\text{ORTH}, \\ \Uparrow\text{STRUC}|\text{PARTS}|\text{HEAD}|\text{INT}|\text{SURF}|\text{ORTH} \end{pmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

An interesting feature is the operation of *lexical insertion*, the constraints on which are specified by the '⇑' inheritance type and expressed as *attribute paths*, i.e. nested AVMs with only one attribute specified per recursion,

In the illustration, the LEMMA *pussy–willow* paradigmatically inherits properties by default inheritance from the archi–sign *compound_noun*, and syntagmatically inherits from the head *willow* 'salix' and the modifier *pussy* 'felis'. Those ILEX template properties for *pussy–willow* which are not specified are completed by unification via inheritance: either percolated up from the head *willow* or inherited from the archi–sign *compound_noun*. The LEMMA *pussy–willow* is seen to be partially rather than fully compositional in that the value for the attribute path INT|MEAN|QUALIA|RELN is specified idiosyncratically. At a higher level in the inheritance path, the value for INT|MEAN|QUALIA|RELN may be specified differently, e.g. as IS_A; the more specific value overrides the more general value.

A lexical sign which inherits *all* its INT properties from the properties of its PARTS, and its general compositional properties from its CAT attribute (such as function application, concatenation, association), and is not otherwise idiosyncratically specified for INT (i.e. has no default–overrides), is *totally compositional.*

A lexical sign which inherits *none* of its INT properties from properties of PARTS, all of these properties being specified idiosyncratically, is *totally noncompositional.* An extreme example of a sign which is totally non–compositional is a hesitation particle interjection such as 'er', i.e. /ə:/; however, even this is debatable because the /ə/ is associated with a flat stylised intonation and together with this intonation has a 'phatic' channel–sustaining function.

A lexical sign which inherits *some* of its INT properties from properties of its PARTS, others being specified idiosyncratically, or which does not inherit compositional properties from the most general subsumer in the inheritance graph, is *partially compositional.*

The totally compositional and totally non–compositional or idiosyncratic cases are 'ideal types' corresponding to absolute or zero adherence to Frege's Principle. Lexical signs, in the general case, exhibit varying degrees of partial compositionality (or, conversely, *exceptionality* or *irregularity*), measurable by their depth in the type inheritance hierarchy. The concept of a scale of compositionality applies not just to semantics, but also to surface form.

For example, orthography is partially compositional: in *ladies' fingers* 'okra', the ORTH of the plural *fingers* is a function of the ORTH of the PARTS *finger* and *s*, but the ORTH of the genitive plural *ladies'* is a more specific function of the PARTS *lady* and *s*.

The PHON property is also only partly compositional. The plural /fɪŋgəz/ appears at first sight to be a general compositional function of the PARTS /fɪŋgə/ and /z/, namely concatenation (interpreted as temporal immediate precedence: /fɪŋgə/ $\prec^\circ$ /z/). However, the compositional function is in fact a more complex morphophonological function which is sensitive to the MANNER and VOICING specifications of the stem–final segment. Morphophonology therefore defines a scale of partial phonetic compositionality.

Perhaps the most interesting cases are the MEAN–SURF parallels in partial compositionality which characterise diachronically lexicalised compounds. For example, the ORTH of *dustman* is perfectly compositional. The MEAN (in informal terms) is, however, only partially compositional: 'municipally employed professional refuse collector', whereby

1. the collective noun 'refuse' (rubbish, garbage) has a very general semantic paradigmatic relation to 'dust',
2. the deverbal derivation 'collector' characteristically denotes a male agent,
3. further details are elliptical, a typical feature of compounds.

But the PHON property is also only partially compositional: /dʌsmən/, and not /dʌstmæn/, i.e. the final consonant of /dʌst/ is elided and the vowel of /mæn/ is weakened. Partial compositionality of this kind has to be specified idiosyncratically for each lexical item concerned; this is the kind of partial compositionality which, on the diachronic dimension, has led in time to the total non–compositionality of PHON and ORTH with words like *woman* = $f_{diachron}$(*wife*,*man*) or *husband* = $f_{diachron}$(*house*,*bond*).

3.10.2.6 Lexical items as structural semiotic types

The notion *lexical item* is used to cover any lexical sign type but also other inventorisable items such as affixes and phonemes, whose lexical status in linguistics is controversial. Some examples of structural semiotic characterisations of these items are given below.

Phoneme: A minimal sign with no MEAN specification and no PARTS (*pace* proponents of distinctive features and autosegmental lattices; sub–morphemic morphological composition is not at issue here).

Morpheme: A sign with elementary MEAN specification, the PHON of whose PARTS is specified for a concatenation of *phonemes.*

Lexical morpheme, lexical base, root: A *simple stem*; a *grammatical morpheme* is an *affix.*

Cranberry morph: A *morpheme* with no specification for MEAN.

Word: A *word* (in English) is specified recursively for all four structural semiotic properties:

1. an uninflectable root, or
2. an inflectable root with an inflection, or
3. a derivation terminated by an inflected suffix, or
4. a compound terminated by a word.

Stem: A lexical root, or an item to which an affix is attached to form a derivation or an inflection, or to which a word or another stem is attached to form a compound word.

Derivation: A complex *stem* consisting of a single *root* attached to an *affix*; the type *affix* covers prefixes, suffixes, infixes, interfixes, introfixes (intercalations), superfixes, and 'attached to' covers the relevant

compositional part–part operations.
Compound: A complex *stem* consisting of more than one *root*, each of which may be the centre of a *derivation* and may be inflected; a compound *word* must terminate in an inflected *root* or an inflected *derivational suffix.*
Phrasal idiom: A lexical sign licensed by the principles and rules of sentence structure, with some PARTS unspecified according to the *frozenness hierarchy* of idiomaticity.
Lexical prosody: A *superfix* item with semiotic properties like those of *phonemes* or *morphemes*, but which is not concatenated but prosodically associated with other phonemes or morphemes. Prosodic association is interpreted as *temporal overlap* ($X \circ Y$) of phonetic events, while concatenation is interpreted as *immediate precedence* ($X \prec^{\circ} Y$) of temporal events Carson-Berndsen (1993b). A more general relation of *precedence* ($X \circ \mathrm{Y}$) is often used.
Nonce word: A sign licensed by the *word* constraints, but not inventarised as a lexical sign.
Phrase, sentence: A sign licensed by the *phrasal idiom* constraints, but not inventarised as a lexical sign. In the view represented by the ILEX model, all sign types are *grounded in lexical signs* of the corresponding ranks. Morphology is thus seen as the discipline dealing with generalisations over lexicalised words, syntax in the traditional sense of the term is seen as the discipline dealing with generalisations over phrasal idioms, and so on.

3.10.3 A selection of English noun compound types

Four of the main kinds of compound noun in English (*tatpurusa*, *bahuvrihi*, *dvandva*, *synthetic*) will suffice to demonstrate the ILEX approach.

Tatpurusa (endocentric) compounds: In endocentric or tatpurusa compounds, the MEAN of the whole is subsumed by the MEAN of the HEAD of the PARTS. A milk–bottle is a bottle, a mouse–trap is a trap: MEAN(*bottle*) $\sqsubseteq$ MEAN(*milk–bottle*), MEAN(*trap*) $\sqsubseteq$ MEAN(*mouse–trap*).
There are metaphorical variants: a pineapple is not an apple, but functionally similar or jocularly relatable to an apple (maybe when seen from a considerable distance or eaten blindfolded after a hot curry). The MEAN of *apple* still subsumes the MEAN of *pineapple*; the MEAN of both is subsumed by the MEAN of *fruit.* The inheritance structure of 'pineapple' is very similar to that of 'pussy–willow', illustrated above, but but with a metapor relation RESEMBLE which applies both to the head and the modifier ('something like an apple which grows on something like a pine').

Bahuvrihi (exocentric) compounds: In bahuvrihi compounds, the MEAN of the whole is not subsumed by the MEAN of the HEAD of the PARTS, but by an elliptical 'understood' semantic category.
The simplest kinds of exocentric compound are items such as 'redskin' or 'longlegs', paraphasable informally as 'SOMEONE who will typically HAVE *skin* which is kinda *red*' and 'SOMEONE who will typically HAVE *legs* which are kinda *long*', with a 'has property' relation. Capitalisation indicates elliptical terms, parentheses indicate elliptical relations which are characteristic of the kind of compound concerned, italics indicate overt components. Capitalised and bracketed items are the largest factors in the partial compositionality of exocentric compounds.
A more complex type is *pickpocket*, i.e. 'SOMEONE who will typically professionally surreptiously *pick*[=extract] VALUABLES from someone else's *pocket*'. Exocentric compounds are modelled with more deeply nested inheritance structures than endocentric compounds.

Dvandva (coordinate) compounds: The parts of coordinate compounds occur in a fixed order, and are morphologically headed, but semantically have no head–modifier structure. The functor is, basically, conjunction. Examples of this relatively simple type are 'fighter–bomber', which is both a fighter and a bomber.

Synthetic compounds: The second element of a synthetic compound a derived noun whose ending enters into the same semantic construction as its stem and the preceding noun. Examples of this type are *busdriver*, *screwdriver.* The ORTH derivational structure of *busdriver* is bracketed as

$$\begin{aligned}&\mathrm{ORTH}(\mathit{busdriver}) =\\&\textsc{concat}_{orth}(\mathrm{ORTH}(\mathit{bus}),\mathrm{ORTH}(\mathit{driver})) =\\&\textsc{concat}_{orth}(\mathrm{ORTH}(\mathit{bus}),(\textsc{concat}_{orth}(\mathrm{ORTH}(\mathit{drive}),\mathrm{ORTH}(\mathit{er})))).\end{aligned}$$

However, the MEAN structure is bracketed differently (omitting some details):

MEAN(*busdriver*) =
λx (SEMANTICALLY_LINK(MEAN(*drive*),(MEAN(x),MEAN(*bus*))))

Some apparent synthetic compounds involve so–called *bracketing paradoxes*, which can be explained as different compositional structures defined for SURF and MEAN attributes. One classical case has the semantic bracketing ((*transformation al grammar*) *ian*), i.e.

λx(SEMANTICALLY_LINK(PROFESSIONALLY_PRODUCE,
MEAN('x'),
SEMANTICALLY_LINK(MEAN('al'),
MEAN('transformation')),
MEAN('grammar'))))

versus the morphological bracketing ((*transformation al*) (*grammar ian*)).

3.10.4 The DATR formalism

3.10.4.1 Theories and models

A *theory* such as the AVM–based account of English compounds sketched above, may simultaneously describe any number of *models*. A model may be formal, such as a set–theoretic representation of an empirical domain, or more informal, as is generally the case in descriptive linguistics, formulated in plain text enriched with symbols and line drawings. A theory is simply a subset of sentences in a formalism for which a model exists in terms of which the sentences can be interpreted.

One kind of formal model for a theory is an 'implementation', i.e. an interpretation of the theory in terms of an operational knowledge representation language or programming language. This is actually a special case of a more general kind of formal interpretation; interpretations for AVMs have been given, for example, in terms of finite state automata (see Kasper and Rounds (1986)). An interpretation of a theory in terms of a different but perhaps more well–known formalism permits conclusions to be drawn about whether the theory is complete (describes all it is supposed to describe) and sound (does not describe anything it is not supposed to describe). If the interpretation function is bijective, then in principle the model could be regarded as the theory and the theory as the model; this is then just a question of perspective.

In this sense, the lexical representation formalism DATR will be used to provide an operational model for the theory which permits quick consistency checking of complex theories by the automatic deduction of hypotheses. Descriptions in DATR are, however, generally referred to as 'theories'.

DATR 'theories', used here as 'operational models' for AVM theories, are sets of DATR sentences. DATR sentences are pairs of a *node* and a set of equations, each of which is a pair of an attribute path and a value.

In the ILEX approach, therefore, a lexicon is an AVM theory which describes an operational model formulated in DATR; this model can itself be seen as an empirical theory which is interpreted (like the AVM theory) in terms of an empirical model with observationally identifiable categories.

3.10.4.2 DATR syntax

The syntax of DATR expresses three kinds of hierarchical structure:

1. Syntagmatic:
    1. Nested attribute value structures (here used to represent ID relations between and property assignment to signs),
    2. Hierarchies of sequences, with property percolation through the hierarchy expressed by 'local inheritance', and lexical insertion expressed by 'global inheritance',
3. Paradigmatic: class inclusion (or implication) hierarchies expressed by local inheritance.

In DATR, nested AVMs are represented as nodes paired with conjunctions of equations. The left–hand side of each equation is an attribute path with attributes represented as atoms[4]:

```
<struc parts modi int surf>
```

The right–hand side is a sequence of value expressions which may be either atoms or inheritance descriptors. There are two main kinds of inheritance descriptor, those which denote *local inheritance* and those which denote *global inheritance*, and in each case there are three subtypes of descriptor which constrain inheritance from different positions in the inheritance hierarchy: by specification of a *node–path pair*, a

[4] DATR nodes are character strings starting with an upper case character, or declared character strings; DATR atoms are either character strings starting with a lower case character, or character strings enclosed in single right quotes, or declared character strings.

node alone, or a *path* alone. For each of these seven cases, i.e. atomic value expressions and the three types each of local and global inheritance, there are seven inference rules.

An important feature of DATR is that paths on the right–hand side are *evaluable*, that is, they have exactly the same formal structure as an entire right–hand side sequence, and may thus contain any value expressions, not just atoms. In particular including other paths, which may in turn include nested value expressions, and so on.

A selective version of the initial example *pussy_willow*, incorporating local (paradigmatic) and global (syntagmatic) inheritance, can be rendered in DATR as follows, with the IPA transcription characters rendered in a slightly modified version of the SAMPA ASCII coding of Wells (cf. Wells (1989b)), in which '/' is used to denote lexical stress:

```
% Query definitions (node-path pairs):
% All nodes except those declared under ‘hide’,
% combined with all paths declared under ‘show’:
# hide Noun Compound_noun .
# show <int mean> <int surf> .
% Lexical entry ranks (simplex and compound nouns):
Willow:
  <>                     == Noun
  <int mean qualia reln> == salix
  <int surf phon>        == ’w/I1@U’
  <int surf orth>        == willow.
Pussy:
  <>                     == Noun
  <int mean qualia reln> == felis
  <int surf phon>        == ’pUsI’
  <int surf orth>        == pussy.
Pussy_willow:
  <>                     == Compound_noun
  <struc parts head>     == "Willow:<>"
  <struc parts modi>     == "Pussy:<>"
  <int mean qualia reln> == ’ RESEMBLE ’
  <int surf reln orth>   == ’-’.
% Paradigmatic inheritance hierarchy (<int surf reln> has default null value):
Compound_noun:
  <>              == Noun
  <int surf reln > ==
  <int mean>      == "<int mean qualia reln>" ’(’
                     "<struc parts head int mean qualia reln>" ,
                     "<struc parts modi int mean qualia reln>" ’)’
  <int surf>      == "<struc parts modi int surf>"
                     "<int surf reln>"
                     "<struc parts head int surf>".
Noun:
  <>              ==
  <int mean>      == "<int mean qualia reln>".
```

The empty path, which appears as a left–hand–side under each node, is the path with no attributes specified. This is the most general path, and indicates the inheritance path to the next more general node or class. Any values which are explicitly specified in an equation associated with the current class override values of the same attributes specified at a higher (more general) node; in this case, the INT values are exhaustively specified, so only information about the category itself is locally inherited.

Information about the parts is globally inherited from each part lemma, the head *Willow* and the modifier *Pussy.* In HPSG terms, the HEAD features are inherited from the head or HEAD-DTR, and the COMP features are inherited from the modifier or COMP-DTRS.

Global inheritance means that the parts concerned are treated quite independently of each other and of the larger unit, ensuring compositionality (which can be modified if necessary for descriptive reasons). Among the DATR equations that can be derived from the theory are the following:

```
Pussy:< int mean > = felis .
Pussy:< int surf phon > = pUsI .
Pussy:< int surf orth > = pussy .
Willow:< int mean > = salix .
Willow:< int surf phon > = w/I1@U .
Willow:< int surf orth > = willow .
Pussy_willow:< int mean > =  RESEMBLE  ( salix , felis ) .
Pussy_willow:< int surf phon > = pUsI w/I1@U .
Pussy_willow:< int surf orth > = pussy - willow .
```

The DATR inheritance rules were the starting point for the definition of the paradigmatic and syntagmatic inheritance relations used in the AVM–based theory introduced in the preceding sections. For this reason, there is a simple mapping between the inheritance and compositionality operators used in the AVM theory, and the six inheritance operations defined for DATR, though not all the DATR possibilities are exhausted in the AVM theory (see Table 3.3). Atomic values are basically the same in each formalism.

Table 3.3: Inheritance operations.

DATR operation	DATR notation	AVM notation
Local node:path inheritance	`A:<b c d>`	→
Local node inheritance	`A`	A special case of →
Local path inheritance	`<b c d>`	← (also a special case of →)
Global node:path inheritance	`"A:<b c d>"`	⇓
Global node inheritance	`"A"`	Rarely used.
Global path inheritance	`"<b c d>"`	⇑

3.10.4.3 DATR rules of deduction

The DATR rules of deduction will be explained here in procedural terms (though a declarative explanation may also be given, see Evans and Gazdar (1996)). The inference rules are of four types: an initialisation rule, a query connection (matching) rule, a path extension rule, and finally an inference rule for each of the seven value expression types.

Environments, initialisation and modification: The DATR rules of deduction refer to a *local environment* and a *global environment.* Each environment consists of a pair of variables, one for evaluation of the local node–path pair, node, and path descriptors, and the other for evaluation of global node–path pair, node and path descriptors. The global environment is initialised to the value of the query node–path pair, and re–defined by the global inheritance descriptors. When the global environment is initialised and whenever it is changed, the variables in the local environment are copied into the local environment. Environment changes are encapsulated for the inheritance descriptor concerned, whether local or global, and do not affect sibling descriptors in the same sequence. However, the same local and global environments are valid for all paths at all depths of recursion in the descriptor concerned.

Matching: The matching of a query attribute path with the paths on the left–hand side of a DATR equation is based on two operations over the local environment and the theory, *connection* and *extension.*

Connection: The local environment connects with a NODE:PATH==SEQUENCE equation defined in a theory iff

1. NODE is identical to the node in the local environment,
2. PATH is a prefix of the path in the local environment, e.g.: the local environment path
   ```
   <int mean qualia reln>
   ```
   matches the following prefixes (whereby the identical path and the zero path both count as prefixes):
   ```
   <int mean qualia reln>
   <int mean qualia>
   <int mean>
   <int>
   <>
   ```
3. PATH is the longest path under NODE which is also a prefix of the path in the local environment. For example, given the local environment path `<int mean qualia reln>`, and two competing paths under NODE which are prefixes of this path,
   ```
   <int mean qualia>
   <int mean>
   ```
   the match is with `<int mean qualia>`: 'the longest path wins'. This principle defines *default inheritance* in DATR.

Extension: The path in a connected local environment consists of a matching prefix and an extension suffix (possibly zero); in the preceding example, `<int mean qualia>` is the matching prefix and `<reln>` is the extension suffix; the matched local environment can be represented by `<int mean qualia || reln>`. Extension is the concatenation of all paths in an equation (however deeply embedded, in both local and global inheritance descriptors) with the extension suffix, for example, with the local environment and matching equation

```
<int mean qualia reln>
<int mean qualia> == Semantics:<qualia>
```

The extension of the equation is

```
<int mean qualia reln> == Semantics:<qualia reln>.
```

The following notation will sometimes be used for clarity:

`<int mean qualia || reln> == Semantics:<qualia || reln>.`

This mechanism expresses a form of constraint propagation for orthogogonal inheritance through the inheritance network.

Inheritance: The right–hand side of a connected and extended equation is evaluated according to seven rules of inference or inheritance rules, one for atoms and three each for inheritance descriptors in the local and global environments. The inheritance rules define how the value expressions on the right–hand side of DATR equations are to be evaluated. Evaluation consists of finding a value for a DATR query, i.e. a node–path pair, by recursive application of the seven inference rules to the elements of sequences and evaluable paths.

Inference rules:

1. DATR sequences and DATR atoms:
   DATR sequences evaluate to a concatenation of the values of their parts, i.e. sequences of atoms.
   Rule I: DATR atoms evaluate to themselves.
2. DATR local inheritance:
   Rule II: Local NODE:PATH descriptor. Substitute NODE for the node and PATH (after evaluation and extension) for the path in the local environment, and connect the local environment with the theory.
   Rule III: Local NODE descriptor. Substitute NODE for the node in the local environment, and connect the local environment with the theory.
   Rule IV: Local PATH descriptor. Substitute PATH (after evaluation and extension) for the local environment path, and connect the local environment with the theory.
3. DATR global inheritance:
   Rule V: Global NODE:PATH descriptor. Substitute NODE for the node in the global environments, and PATH (after evaluation and extension) for the global environment path; copy the global environment to the local environment and connect the local environment with the theory.
   Rule VI: Global NODE descriptor. Substitute NODE for the node in the global environment; copy the global environment to the local environment and connect the local environment to the theory.
   Rule VII: Global PATH descriptor. Substitute PATH (after evaluation and extension) for the global environment path; copy the global environment to the local environment and connect the local environment to the theory.

The following is an example[5] of the inference steps involved in deriving the DATR sentence `Pussy_willow:< int mean > = RESEMBLE(salix,felis).`

```
=0,0,0> LOCAL Pussy_willow:< || int mean > == Compound_noun
        GLOBAL Pussy_willow:< int mean >
RULE III.(NODE)
=1,0,0> LOCAL Compound_noun:< int mean > == "< int mean qualia reln >"
              ( "< struc parts head int mean qualia reln >" ,
              "< struc parts modi int mean qualia reln >" )
        GLOBAL Pussy_willow:< int mean >
RULE VII.(GPATH)
=2,0,0> LOCAL Pussy_willow:< int mean qualia reln > ==  RESEMBLE
        GLOBAL Pussy_willow:< int mean qualia reln >
RULE I.(ATOM)
 RESEMBLE
RULE I.(ATOM)
(
RULE VII.(GPATH)
=2,0,2> LOCAL Pussy_willow:< struc parts head || int mean qualia reln > == "Willow: <  > "
        GLOBAL Pussy_willow:< struc parts head int mean qualia reln >
RULE V.(GNODE:GPATH)
=3,0,0> LOCAL Willow:< int mean qualia reln > == salix
        GLOBAL Willow:< int mean qualia reln >
RULE I.(ATOM)
salix
RULE I.(ATOM)
,
RULE VII.(GPATH)
=2,0,4> LOCAL Pussy_willow:< struc parts modi || int mean qualia reln > == "Pussy: <  > "
        GLOBAL Pussy_willow:< struc parts modi int mean qualia reln >
RULE V.(GNODE:GPATH)
=3,0,0> LOCAL Pussy:< int mean qualia reln > == felis
        GLOBAL Pussy:< int mean qualia reln >
RULE I.(ATOM)
felis
```

[5] The derivation was produced with the ZDATR interpreter, Schillo (1996). The numbers indicate depth of local inheritance, path inheritance, and position in the right–hand–side sequence; the '‖' sequence separates the matched prefix of the local environment from the remaining suffix, and the RULE number refers to the DATR inference rule which applies to the current value expression under evaluation.

```
RULE I.(ATOM)
)
[Query 4 (12 Inferences)] Pussy_willow:< int mean > =  RESEMBLE (salix,felis).
```

3.10.5 An operational DATR model for English compounds

3.10.5.1 Descriptive scope of the model

The model described in the following pages is constructed on the lines outlined in the preceding sections, with a few minor modifications; for example, the AVMs operationalised in the model are flatter, and the descriptive scope of the model is much broader, but the model contains additional relatively informal attribute specifications. There are also many possible 'style options' for modelling in DATR, which will not be discussed here.

The descriptive scope of the model includes the following:

1. simplexes and inflection;
2. compound types tatpurusa, dvandva and bahuvrihi;
3. informal compositional semantic interpretation;
4. phonetic interpretation (pre- and postmorphophonemic representations);
5. orthographic interpretation (pre- and postmorphographemic representations);
6. compositionality generalised for meaning and surface interpretation at all ranks;
7. morphophonological finite state transducer;
8. morphographemic finite state transducer.

Table 3.4: Terms used in the DATR model.

Term:	Description:
cat	category, cf. 'CAT' in AVM
compound	morphological category specification
graph	morphographemic interpretation
head	cf. 'HEAD' in AVM
mass	value for mass noun
modi	cf. 'MODI' in AVM
morph	morphological attribute
operator	compositionality operator, cf. 'RELN' in AVM
orth	orthographic (post-morphographemic) interpretation
phon	phonetic interpretation (including stress marks)
plur	plural inflection
mean	semantic interpretation, cf. AVM 'MEAN'
sing	singular inflection (default value)
stem	morphological category specification
stress	lexical stress
surf	surface interpretation (default is morphophonemic)
plain	inflectional status of modifier

The morphophonological and morphographemic finite state transducers demonstrate how one formalism can be used to operationalise different theories, in this case not as an AVM based theory modelled with directed acyclic graphs, but as automata of the kind used in two-level morphology Koskenniemi (1983a), modelled with directed cyclic graphs. The terms are used are listed in Table 3.4.

Not all aspects of the model can be discussed in the present context, but some of the lexical specifications which can be inferred by application of the inheritance rules are illustrated here with ther synthetic compound *busdriver*:

```
Busdriver:<surf graph>           = bus-drive+er.
Busdriver:<surf graph orth>      = bus-driver.
Busdriver:<surf>                 = bVs#draiv+@.
Busdriver:<surf phon>            = //bVs/draiv@.
Busdriver:<mean>                 = {{{one_OF_{agent|instrument}}
                                     _CAN_{{action_OF_{move_vehicle}}}}
                                           _AFFECT_{{one_OF_{public_road_vehicle}}}}.
Busdriver:<plur surf graph>      = bus-drive+er#+s.
Busdriver:<plur surf graph orth> = bus-drivers.
Busdriver:<plur surf>            = bVs#draiv+@#+/Z.
Busdriver:<plur surf phon>       = //bVs/draiv@z.
Busdriver:<plur mean>            = {{{more_than_one_OF_{agent|instrument}}
                                     _CAN_{{action_OF_{move_vehicle}}}}
                                           _AFFECT_{{one_OF_{public_road_vehicle}}}.
```

The overall architecture ofthe model is shown in Figure 3.3 using the compound *housewife*, following the ILEX model; only the broad outline is represented. Lexical entries, the targets of global inheritance, are indicated by square nodes, the local inheritance hierarchy by elliptical nodes.

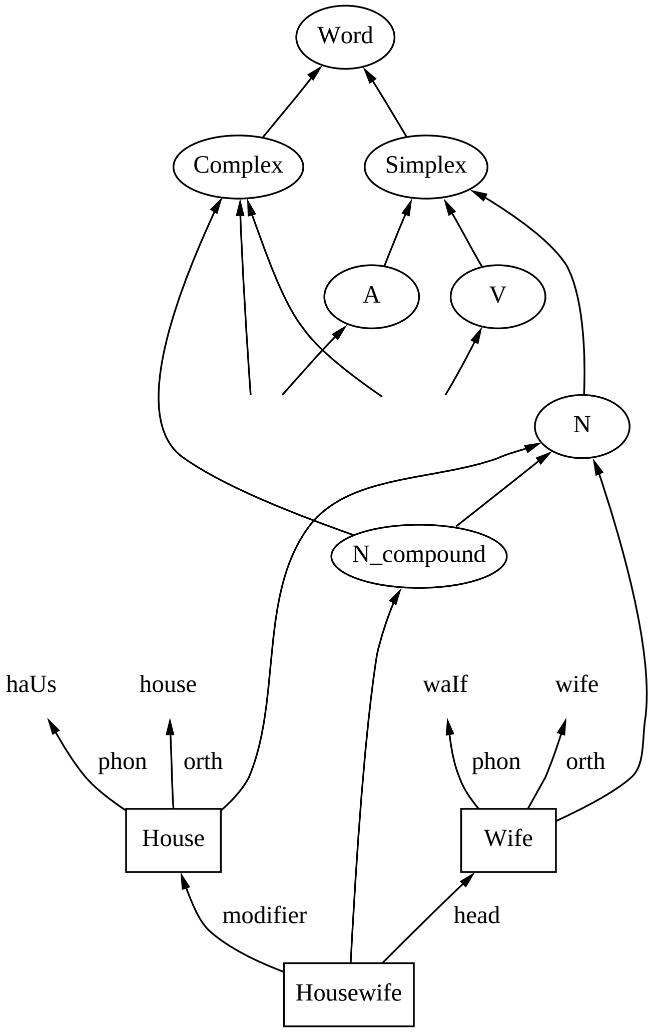


Figure 3.3: Architecture of ILEX–based noun compound theory

3.10.5.2 DATR model: lexicon extract

Simplexes:

```
Pale:
  <>                    == Adjective
  <modi mean>           == rather_white
  <modi surf graph>     == p a l e
  <modi surf>           == p e I l.
Face:
  <>                    == Noun
  <modi mean>           == front_of_head
  <modi surf graph>     == f a c e
  <modi surf>           == f e I s.
```

Derivational suffix:

```
Er:
  <>                    == Noun_suffix
  <modi mean>           == agent|instrument
  <modi surf graph>     == e r
  <modi surf>           == @.
```

Derivations:

```
Bomber:
  <>                    == Noun_derivation
  <modi>                 == "Bomb:<plain>"
  <head>                 == "Er:<>"
  <operator mean>        == CAN.
Driver:
  <>                     == Noun_derivation
  <modi>                 == "Drive:<plain>"
  <head>                 == "Er:<>"
  <operator mean>        == CAN.
```

Standard tatpurusa representation:

```
Mousetrap:
  <>                     == Noun_compound
  <operator mean>        == FOR
  <modi>                 == "Mouse:<plain>"
  <head>                 == "Trap:<>".
Mousetrapcheese:
  <>                     == Noun_compound
  <operator mean>        == FOR
  <operator surf graph> == _
  <modi>                 == "Mousetrap:<plain>"
  <head>                 == "Cheese:<>".
```

Two–stage bahuvrihi representation:

```
Paleface2:
  <>                     == Noun_compound:<>
  <operator mean>        == HASPROP
  <modi>                 == "Paleface:<plain>"
  <head mean>            == someone.
Paleface:
  <>                     == Noun_compound:<>
  <operator mean>        == IS
  <modi>                 == "Pale:<plain>"
  <head>                 == "Face:<>".
```

Dvandva representation:

```
Fighterbomber:
  <>                     == Noun_compound
  <operator mean>        == AS-WELL-AS
  <operator surf graph> == -
  <modi>                 == "Fighter:<plain>"
  <head>                 == "Bomber:<>".
```

3.10.5.3 Noun inheritance hierarchy

The top–level node, *Sign*, is completely unspecified and has the null value. At the *Word* node, information about inflectional neutralisation and constraints on the interpretation mapping is specified. For reasons which cannot be argued here, the default interpretation for HEAD SURF is the null value.

```
Sign:
<>                       == .
Word:
<>                       == Sign
<plur surf phon>         == Morphophon:<Interpretation>
<surf phon>              == Morphophon:<Interpretation>
<plur surf graph orth>   == Morphograph:<Interpretation>
<surf graph orth>        == Morphograph:<Interpretation>
<surf>                   == Interpretation
<plur surf>              == Interpretation
<mean>                   == Interpretation
<plur mean>              == Interpretation
<plain>                  == Interpretation
<plain plur mean>        == <plain mean>
<operator sing>          == "<operator>"
<operator plur>          == "<operator>"
<operator plain mean>    == "<operator mean>"
<operator mean>          == OF
<sing>                   == <>
<modi sing>              == "<modi>"
<modi plur>              == "<modi>"
<head plur>              == "<head>"
<head surf>              ==
<head mean>              == <indiv "<mean indiv>">
<indiv exists>           == APPLICATION
<indiv mass>             == some
<indiv>                  == one
<head plur mean>         == more_than_one
Noun:
<>                       == Word
<cat surf>               == noun
<plain sing>             == <plain>
<plain plur>             == <plain sing>
<head plur surf graph>   == #+ s
<head plur surf>         == #+ /Z .
Noun_compound:
<>                       == Noun
<cat morph>              == compound
<plain operator>         == "<operator>"
```

```
<operator surf>         == #.
Noun_derivation:
<>                      == Noun
<cat morph>             == derivation
<operator surf>         == +.
```

3.10.5.4 Co–interpretation for semantics and surface form

Semantic and phonetic interpretation are, in principle, treated identically, with a number of specific constraints concerned with the assignment of recursive brackets to MEAN and the entirely analogous recursive assignment of lexical stress to SURF PHON, depending on the morphological category.

```
Interpretation:
<>                      == First Operator Second
<plain>                 == <>.
First:
<>                      == StressOp "<modi>"
<mean>                  == { "<head mean>" _
<plur mean>             == { "<head plur mean>" _ .
Second:
<>                      == "<head>"
<mean>                  == _ { "<modi mean>" } }
<plur mean>             == _ { "<modi plur mean>" } }.
Operator:
<>                      == "<operator>".
StressOp:
<>                      ==
<surf phon>             == <stress "<cat morph>">
<plur surf phon>        == <surf phon>
<stress stem>           == /
<stress compound>       == /
<stress>                == .
```

3.10.5.5 Surface interpretation: morphophonemic and morphographemic mapping

The morphographemic mapping maps characters from the lexical level to the post–lexical level taking into account specific restrictions on character mapping at inflectional boundaries. In the general (default) case (the ‘elsewhere condition’), a DATR variable ‘$char’ defines the identity mapping. Boundary diacritics are deleted. Theoretically this traditional use of boundary diacritics is not optimal, but a more adequate treatment would go beyond the scope of the paper.

```
Morphograph:
  <>        ==
  <+>       == <>
  <#+>      == <>
  <#>       == <>
  <##>      == <>
  <$char>   == $char <>
  <e + e>   == e <>
  <e #+ e>  == e <>
  <y + s>   == i e s <>
  <y #+ s>  == i e s <>
  <s #+ s>  == s e s <>.
```

Very much like the morphographemic mapping, in the morphophonemic mapping, the plural morphophoneme ‘/Z’ is realised dependent on its left context as one of /s, z, ɪz/. Other segments, another case of the ‘elsewhere condition’, are realised unchanged using a DATR variable ‘$phon’. Phonemes and (as with spelling) boundary diacritics are not the theoretically optimal choice for phonetic interpretation, but a full feature lattice treatment is not possible in the present context.

```
Morphophon:
  <>          ==
  <$phon>     == $phon
  <+>         == <>
  <#+>        == <>
  <#>         == <>
  <##>        == <>
  <p #+ /Z> == p s <>
  <t #+ /Z> == t s <>
  <k #+ /Z> == k s <>
  <f #+ /Z> == f s <>
  <T #+ /Z> == T z <>
  <s #+ /Z> == s I z <>
  <S #+ /Z> == S I z <>
  <z #+ /Z> == z I z <>
  <Z #+ /Z> == Z I z <>
  </Z>        == z <>.
```

3.10.6 A sample analysis

The complexity of the theory is shown by derivations generated by the operational DATR model. In order to derive the post-lexical phonetic representation of the synthetic compound *busdriver*, 173 DATR inferences (rule applications) are required, in order to derive the simplex plural form *buses*, 44 inferences are needed. It will be sufficient to illustrate the process using a simplex plural, *buses*, as the general definition of head–modifier based on interpretative compositionality covers all morphological ranks.

Initial local inheritance:

```
=0,0,0> LOCAL Bus:< || plur surf phon > == Noun
        GLOBAL Bus:< plur surf phon >
RULE III.(NODE)
=1,0,0> LOCAL Noun:< || plur surf phon > == Word
        GLOBAL Bus:< plur surf phon >
RULE III.(NODE)
=2,0,0> LOCAL Word:< plur surf phon > == Morphophon:< Interpretation >
        GLOBAL Bus:< plur surf phon >
RULE II.(NODE/PATH)
RULE III.(NODE)
```

Assignment of linear precedence and lexical stress to inflected word:

```
=3,1,0> LOCAL Interpretation:< || plur surf phon > == First Operator Second
        GLOBAL Bus:< plur surf phon >
RULE III.(NODE)
=4,1,0> LOCAL First:< || plur surf phon > == StressOp "< modi >"
        GLOBAL Bus:< plur surf phon >
RULE III.(NODE)
=5,1,0> LOCAL StressOp:< plur surf phon > == < surf phon >
        GLOBAL Bus:< plur surf phon >
RULE IV.(PATH)
=6,1,0> LOCAL StressOp:< surf phon > == < stress "< cat morph >" >
        GLOBAL Bus:< plur surf phon >
RULE IV.(PATH)
RULE VII.(GPATH)
=7,2,0> LOCAL Bus:< || cat morph > == Noun
        GLOBAL Bus:< cat morph >
RULE III.(NODE)
=8,2,0> LOCAL Noun:< || cat morph > == Word
        GLOBAL Bus:< cat morph >
RULE III.(NODE)
=9,2,0> LOCAL Word:< || cat morph > == Sign
        GLOBAL Bus:< cat morph >
RULE III.(NODE)
=10,2,0> LOCAL Sign:< || cat morph > ==
        GLOBAL Bus:< cat morph >
RULE I.(ATOM)
=7,1,0> LOCAL StressOp:< stress > ==
        GLOBAL Bus:< plur surf phon >
RULE I.(ATOM)
RULE VII.(GPATH)
```

Assignment of morphophonemic (lexical) representation:

```
=5,1,1> LOCAL Bus:< || modi plur surf phon > == Noun
        GLOBAL Bus:< modi plur surf phon >
RULE III.(NODE)
=6,1,0> LOCAL Noun:< || modi plur surf phon > == Word
        GLOBAL Bus:< modi plur surf phon >
RULE III.(NODE)
=7,1,0> LOCAL Word:< modi plur || surf phon > == "< modi >"
        GLOBAL Bus:< modi plur surf phon >
RULE VII.(GPATH)
=8,1,0> LOCAL Bus:< modi surf || phon > == b V s
        GLOBAL Bus:< modi surf phon >
RULE I.(ATOM)
b
RULE I.(ATOM)
V
RULE I.(ATOM)
s
RULE III.(NODE)
```

Interpretation of (null) inflection operator:

```
=4,1,1> LOCAL Operator:< || plur surf phon > == "< operator >"
        GLOBAL Bus:< plur surf phon >
RULE VII.(GPATH)
=5,1,0> LOCAL Bus:< || operator plur surf phon > == Noun
        GLOBAL Bus:< operator plur surf phon >
RULE III.(NODE)
=6,1,0> LOCAL Noun:< || operator plur surf phon > == Word
        GLOBAL Bus:< operator plur surf phon >
RULE III.(NODE)
=7,1,0> LOCAL Word:< operator plur || surf phon > == "< operator >"
        GLOBAL Bus:< operator plur surf phon >
RULE VII.(GPATH)
=8,1,0> LOCAL Bus:< || operator surf phon > == Noun
        GLOBAL Bus:< operator surf phon >
RULE III.(NODE)
=9,1,0> LOCAL Noun:< || operator surf phon > == Word
        GLOBAL Bus:< operator surf phon >
RULE III.(NODE)
```

```
=10,1,0> LOCAL Word:< || operator surf phon > == Sign
        GLOBAL Bus:< operator surf phon >
RULE III.(NODE)
=11,1,0> LOCAL Sign:< || operator surf phon > ==
        GLOBAL Bus:< operator surf phon >
RULE I.(ATOM)
RULE III.(NODE)
=4,1,2> LOCAL Second:< || plur surf phon > == "< head >"
        GLOBAL Bus:< plur surf phon >
RULE VII.(GPATH)
```

Assignment of plural morphophoneme:

```
=5,1,0> LOCAL Bus:< || head plur surf phon > == Noun
        GLOBAL Bus:< head plur surf phon >
RULE III.(NODE)
=6,1,0> LOCAL Noun:< head plur surf || phon > == #+ /Z
        GLOBAL Bus:< head plur surf phon >
RULE I.(ATOM)
#+
RULE I.(ATOM)
/Z
```

Morphophonemic mapping:

```
=3,0,0> LOCAL Morphophon:< b || V s #+ /Z > == b <  >
        GLOBAL Bus:< plur surf phon >
RULE I.(ATOM)
b
RULE IV.(PATH)
=4,0,1> LOCAL Morphophon:< V || s #+ /Z > == V <  >
        GLOBAL Bus:< plur surf phon >
RULE I.(ATOM)
V
RULE IV.(PATH)
=5,0,1> LOCAL Morphophon:< s #+ /Z > == s I z <  >
        GLOBAL Bus:< plur surf phon >
RULE I.(ATOM)
s
RULE I.(ATOM)
I
RULE I.(ATOM)
z
RULE IV.(PATH)
=6,0,3> LOCAL Morphophon:< > ==
        GLOBAL Bus:< plur surf phon >
RULE I.(ATOM)
[Query 49 (44 Inferences)] Bus:< plur surf phon > = bVsIz.
```

3.10.7 Discussion and prospects

The goal of this contribution to Inheritance Lexicon Theory is to take a step towards a solution of problems such as the integration of morphology, idioms, and lexical prosody, to introduce a general notion of compositional sign and compositional co–interpretation for surface and semantic interpretation at all structural ranks.

In pursuing this goal, the concept of inheritance was introduced and used to account for both paradigmatic and syntagmatic generalisations, including ID and LP relations and morphographemic and morphophonemic mappings. Starting with a theory based on attribute–value matrices, a formal description of English compounds was outlined. As a heuristic device for investigating the complex implications of the theory, a technique for developing an operational DATR model for the theory was outlined, and an operational DATR model was presented in some detail. An explicit mapping from the theory to the model was not defined.

The results demonstrate the flexibility of the ILEX methodology, and provide a vivid illustration both of the complexity of natural language, in terms of the length and depth of the derivation of interpretative representations. But the results also demonstrate the elegance and simplicity of natural language lexical items, in terms of highly underspecified lemma entries. The operational model demonstrates for the first time that it is possible to integrate a variety of different facts about compositionality in the lexicon in a homogeneous, theoretically well–founded and computationally tractable fashion, without sacrificing linguistic perspicuity.

As well as adding a dimension of compositionality to the basic structuralist concept of a sign, the multiply linked lattice structures of inheritance lexicon methodology contribute towards a new interpretation of another basic structuralist position in respect of the structure of language: *un système où tout se tient.*

4 Elementary lexical databases

The area of *lexicon structure* deals with the organisation of information in lexica. Models for lexical information, and types of lexical information, are dealt with in the preceding sections. Terminology varies considerably in this area. The structure of a spoken language lexicon may be seen from the following points of view:

Lexical formalisms, lexicon representation languages: Representation conventions of various types (symbolic notations, programming languages, database languages, logical formalisms, purpose-designed knowledge representation languages), which are suitable for formulating *lexical models.*

Lexicon architecture: The choice of basic objects and properties in the lexicon, and the structure of the lexicon as a whole, such as a table of items, a trie (decision tree), an inheritance hierarchy, a semantic network, a database.

4.1 Spoken language lexicon formalisms

Spoken language lexicon formalisms (representation languages) may be broadly classified according to their use:

1. Linguistically and phonetically based working notations.
2. Implementation languages for the operational phase.
3. Algebraic and logical formalisms for formal definition.

Where an ad hoc solution is required for a very small lexicon, and where lexicon structure is simple, a lexicon may be written directly in a standard programming language suitable for high-speed runtime applications, traditionally Fortran but more recently C, or in a higher level language such as LISP or Prolog. Recent developments are moving towards high level knowledge representation languages which are specifically designed to meet all three of the above criteria equally well, in that they are useful working notations, have efficient implementations, and are formally well-defined.

Some of these are also used for general written language lexica. A more detailed classification of formal representation systems may be given as follows:

1. General data structures (lists, tables or matrices, tree structures designed for optimal lexical access).
2. Programming languages (C for efficiency; LISP or Prolog for flexibility).
3. Database systems.
4. General text markup languages such as SGML.
5. Knowledge representation languages (inheritance networks, semantic networks, frame systems).
6. Linguistic knowledge representation languages, commonly based on attribute-value logics.
7. Lexical knowledge representation languages (attribute based inheritance formalisms) such as DATR.

General data structure definitions for these representations are required for developers and for theoretical work on the complexity and efficiency of lexica and lexicon processing. Standard textbooks on data structures and algorithms should be consulted for this purpose.

Conventional programming languages are generally used for performance reasons in runtime systems. They may also be used to implement small or simple lexica directly, in particular for rapid prototyping of these; this is not optimal software development practice, however, and not to be recommended for developing large or complex lexica, in particular those with highly structured linguistic information.

Database management systems (DBMSs) are widely used for general lexical resource management, including large-scale lexica with rich information which needs to be accessed flexibly and efficiently. In the Sam project, an Oracle database management concept for spoken language corpora and lexica was developed (cf. Dolmazon et al. 1990).

General text markup languages are used for integration with large, pre-analysed written corpora in the development of natural language processing systems and in statistical basic research in computational linguistics, but so far have not been used in spoken language system development (cf. the results of the Eagles Working Group on Text Corpora). Implementations of SGML are readily available.

Knowledge representation languages (KRLs) are used for developing complex semantic and conceptual knowledge representations, and for integrating spoken language front ends with knowledge based systems; see Schröder et al. (1987), Sagerer (1990); more generally, cf. Bobrow and Winograd (1977), Brachman and Levesque (1985), Charniak and McDermott (1985), De Mori et al. (1984), Young et al. (1989).

Linguistic formalisms in general are discussed in the results of the EAGLES Working Group on Grammar Formalisms, which should be referred to in this connection.

Lexical knowledge representation languages (LKRLs) are a relatively new development. They are coming to be used in knowledge acquisition for integrated lexica which contain a variety of complex lexical information from phonology through morphology and syntax to semantics and pragmatics. They provide a means of bridging the gap between complexity of lexical information and easy-to-read representations using sign-based lexicon models. A LKRL which has been used in several natural language processing and language and speech projects is DATR (cf. Evans and Gazdar 1989; Cahill 1993b; Cahill and Evans 1990; Andry et al. 1992; Gibbon 1991, 1993; Bleiching 1992a; Langer and Gibbon 1992). This is the language which is used for basic attribute-value representations in this chapter. A number of public domain DATR implementations are available and can be obtained from the Web sites.

4.2 Lexicon architecture and lexical database structure

Lexicon architecture pertains to the *choice of basic objects and properties* in the lexicon, and to the *overall structure* of the lexicon. More formally, it defines the relation which assigns lexical properties to lexical entries.

The term "architecture" generally refers to the structure of system lexica, but the term is also justified in connection with lexical database structure, particularly when more complex relational or object-oriented structures are concerned.

The basic objects in terms of which an architecture may be defined were discussed in the section on lexical information for spoken language.

The overall structure of a spoken language lexicon is determined by a range of declarative, procedural and operational criteria such as the following:

- The complexity of the information assigned to lexical entries.
- The complexity of the relations defined between lexical entries.
- The particular subset of objects and properties defined for a given application lexicon.
- Linguistic and logical compression techniques such as redundancy rules or, more generally, inheritance hierarchies.
- Task driven directionality of access.
- Variety of information required for access (from phonological to pragmatic).
- Performance requirements of software (including lingware) size and speed of access.
- Techniques of acquisition and maintenance (with respect, for instance, consistency).

At the one extreme is the ideal notion of a fully integrated sign-based model with non-redundant specification of entries and property inheritance; in between is the efficient database management system used for large scale lexical databases, and at the other extreme is the simple pronunciation table which is the starting point for the training of speech recognition devices.

The choice of lexicon architecture on the basis of parameters such as those listed above, and taking into account practical constraints from the actual working environment, is application specific. There is no single principle of organisation which applies to all lexica.

The closest approximation to a neutral form of spoken language lexicon organisation is a sign-based general background lexicon organised as a database with flexible access. Such a lexicon is basically knowledge acquisition oriented, and can function as a source for the specialised lexica required for different speech synthesis and recognition applications. Specialised models for sublexica which are optimised for particular applications can then be formulated, and sublexica can be automatically compiled out of the main lexicon into application-specific notations and structures.

The organisation of a lexicon determines the general properties of the formalism to be used with the lexicon. Conversely, available formalisms determine tractable forms of lexicon organisation in terms of data structures, algorithms and programming environments (cf. Knuth 1973; Carbonell and Pierrel 1986; Rudnicky et al. 1987; Lacouture and Normandin 1993). Object-oriented system architectures, with local encapsulation of all aspects of representation and processing, permit the construction of hybrid systems with functionally optimised components; by analogy, the lexicon itself can be conceived as a hybrid object system if required.

This is in effect the situation in current speech recognition technology, in which a more or less large set of HMMs representing words, for instance, can be seen as a procedurally sophisticated lexicon with acoustically driven lookup of keys which are then used to access the main lexicon. Although the standard perspective is to see the two components as separate, they can be seen as objects which are both located in hybrid spoken language system spoken language system lexicon components.

Current research on new object-oriented interactive incremental spoken language system architectures raises many new questions about the role of a lexicon. One major question is whether the lexicon is an object (or system of objects) in its own right, or whether the lexicon is distributed over the system components and is thus a *virtual lexicon*, or which components of the system, e.g. morphology and word semantics, or sentence parsing and propositional semantics, interact directly. Questions such as these are the subject of ongoing basic research, and it would be premature to make specific recommendations at this point.

For a broader discussion of lexicon architectures, the work of the EAGLES Working Group on Computational Lexica should be consulted.

4.3 Lexicon architecture and the structure of lexical databases

The architecture of a lexicon, in particular of a lexical database, is determined partly by the types of declarative knowledge it contains, partly by considerations of access and interaction with other databases or modules. The main features of spoken language lexical databases have already been discussed. In practice, a spoken language lexical database is often a set of loosely related simpler databases (e.g. pronunciation table, signal annotation file, stochastic word model, and a main lexical database with syntactic and semantic information), and is part of a larger complex of databases involving speech signal files, transcription files (orthographic and phonemic), and annotation (labelling) files which define a function from the transcriptions into the digitised speech signal. However, in the interests of consistency it is helpful to take a more general lexicographic point of view, and to see a lexical database for spoken language development as a single database, in which relations between lexical items and their properties at all levels, from acoustics through word structure to syntax, semantics and pragmatics are defined.

The major problem in deciding how to organise a lexical database is the *ambiguity* of word forms. In a spoken language system, the focus is on the *pronunciation*, i.e. on *phonemic word forms* (not the orthography, though this is often used as a conveniently familiar form of representation). The key issue here is *homophony*, i.e. a phonemic word form associated with at least two different sets of lexical information, and thus logically involving a *disjunction* in the database.

In a simple traditional database model based on fixed-length records, in which each field represents a specific attribute of the entity which the record stands for, there is a record for each lexical entry associated with a homophone, uniquely identified by a serial number. However, for specific applications such as the training of a speech recogniser it is convenient to have just one record for each word form. In a database which is optimised for this application, the disjunction required by the homphone is *within a single record*, rather than distributed over alternative records which share the same value for the pronunciation attribute. Structures of this type are typically used in pronunciation dictionaries (pronunciation lexica, pronunciation tables) for speech recognition. Disjunctive information of this kind within the lexical database corresponds to non-deterministic situations and the use of complex search algorithms in actual spoken language systems.

4.3.1 A simple database type: Pronunciation tables

Pronunciation tables (pronunciation dictionaries) hardly correspond to the intuitive concept of a lexical database, which implies a fairly high degree of complexity, but they are nevertheless a useful source of practical examples of a simple lexical database structure.

Pronunciation tables define the relation between orthographic and phonemic representations of words. Often they are defined as functions which assign pronunciations (frequently a set of variant pronunciations) to orthographic representations; this is an obvious necessity for text-to-speech lexica, but in speech recognition applications in which orthographic transcriptions (which are easier to make and check than phonemic transcriptions) are mapped to phonemic representations for the purpose of training speech recognisers, the use of a pronunciation table of this type is relevant.

A pronunciation table which involves pronunciation variants (see below) provides a simple illustration of the orthographic noise problems, represented by disjunctions in the database.

Pronunciation tables have to fulfil a number of criteria, in particular the criterion of unambiguous notation, of consistency with orthographic transcriptions and other transcriptions of a particular corpus, and of simple and fast processing.

General proposals for the interchange of lexical information about word forms, including morphological,

Table 4.1: Frequently used symbols

Boundaries	
morpheme:	+
word:	#
liaisonless group:	##
phonological syntagma:	§ (in phrasal entries)
Phonemes (in IPA or SAMPA notation), including a notation for the French archiphonemes.	
Phonological diacritics	
latency mark	”
(for consonants pronounced in liaison contexts or morphological linking)	
consonant deletion mark	’ (e.g. for final consonants)

Table 4.2: Verbmobil diacritics

Boundaries	
morpheme:	+
stem-inflection boundary:	#+
word in compounds:	#
word in phrases:	##
syllable:	.
primary stress:	'
secondary stress:	'' (two single quotes)
Additional conventions	
The boundaries # and ## are both coextensive + and . boundaries.	
Where + and . boundaries are coextensive, . is written before +.	
The stress marks ' and '' are written immediately before the vowel, not before the syllable.	

phonological and prosodic information, have been made for different languages. They do not have standard status at the current time, but they are sufficiently similar to justify recommendation. A standard for French has been described (cf. Pérennou and De Calmès 1987; Autesserre et al. 1989), containing the features tabled in Figure II:2.

For the spoken language lexicon in the German Verbmobil project the same basic principle has been adopted (cf. Bleiching and Gibbon 1994), with extensions for incorporating prosodic information, as in Table II:2.

Table II:2 shows an extract from the Verbmobil pronunciation table in the Verbmobil WIF (Word form Interchange Format) convention; following current practice, it is organised according to orthographic keys.

The convention has been designed to permit the removal of information which is not required, or the selection of useful subsets of the table using simple UNIX tool commands; the use of “'” for primary and secondary stress permits simple generalisation over both.

4.3.2 Example of wordlist construction: Verbmobil demonstrator

4.3.2.1 Preliminary remarks

The Verbmobil demonstrator wordlist was defined on the basis of consultations with all partners between March 1994 and May 1994, with the aim of defining the vocabulary for the Verbmobil demonstrator, due in February 1995. This was a non-trivial enterprise which required the explicit development of somewhat

Table 4.3: Extract from the Verbmobil pronunciation table

ASCII orthography	Extended SAMPA transcription
Angst	?'aNst
Annahme	?'an#n"a:.m+@
Apparat	?a.pa.r'a:t
April	?a.pr'Il
Aprilwoche	?a.pr'Il#v"O.x+@
Arzttermin	?'a6tst#tE6.m"i:n
Aschermittwoch	?"a.S6#m'It#v"Ox
Auf_Wiederh"oren	?aUf##v'i:.d6#h"2:.r+@n
Auf_Wiederschauen	?aUf##v'i:.d6#S"aU.+@n
Auf_Wiedersehen	?aUf##v'i:.d6#z"e:.+@n
August	?aU.g'Ust ?'aU.gUst
Augustwoche	?aU.g'Ust#v"O.x+@
Ausweichm"oglichkeit	?'aUs#v"aIC#m"2:k.+lIC.+kaIt

complex lexicographic criteria for spoken language, which are outlined and then specified in detail below. The main logistic problems can be summarised as follows. From the point of view of the speech recognition groups, the wordlist had to be corpus-based. From the point of view of the linguistic groups, the wordlist had to contain sensible generalisations (for instance, to contain full inflectional paradigms, or domain specific closed word classes such as days of the week, or months, or numbers for dates, calendar weeks, and years). For the speech groups and some of the language groups, it made sense to define the entry as a *fully inflected word form*, and this definition was adopted. For the morphology groups, and for medium term lexicon work, it made sense, however, to regard the uninflected stem as the basic lexical entry. In particular, the wordlist was technically defined as a submatrix of the full Verbmobil lexicon matrix. Finally, the discrimination ability of the speech recognition components for continuous spontaneous speech was estimated to define a limit of about 1300 words, which imposed a top limit.

These partially conflicting requirements made it necessary to define the lexicographic criteria explicitly in order to avoid as far as possible (an impossible task, of course) misunderstandings of different kinds from the speech and language oriented groups in respect of their different criteria.

These questions pertain to the definition of *extensional coverage criteria*, that is, to the selection and number of entries in the wordlist.

Complementary to the extensional coverage criteria are the *intensional coverage criteria*, that is, the number and type of *attributes* assigned to each entry (or, in straightforward database terms, the number of fields in each entry record). Initially, the intensional coverage was defined in terms of two attributes, orthography and pronunciation. However, this was not the end of the story, because the different speech recognition subcomponents for word recognition, syllable handling, prosody, morphological word-structuring, etc., required different kinds of information under the pronunciation attributes. Finally, the problem of ambiguous orthographies for pronunciation variants (heterophonic homographs) had to be treated.

The result of discussions and negotiations was a compact wordlist format, whose main properties may be summarised in the followoing terms:

- German standard orthography as defined by the Duden Publishing Company was adopted; borderline cases (such as contracted preposition-article sequences) were adopted from the Handbook of the Verbmobil corpus project (Teilprojekt 14).
- The encoding of German standard orthography followed the operational norm defined by the *de facto* standard file *german.sty* commonly used as a Latex document type for German (see below).
- A pronunciation transcription was developed to include the following properties:
    - Phonemic transcription according to the international SAMPA conventions for German (some local SAMPA variants differ slightly, e.g. in using /Q/ for glottal stop; however, this is defined in international SAMPA as a variety of open rounded back vowel and was therefore considered unsuitable) Length marks for long vowels were included, as these are widely used, though they are, strictly speaking, redundant, and are not used in the Duden *Aussprachewörterbuch* (Pronunciation Dictionary).

- Word stress: Primary and secondary lexical stress are encoded (to be distinguished from phonetic accent in context). The international SAMPA encoding with double quote (") and percent (%) was found to be inconvenient for a number of ASCII oriented processing environments, and replaced by a single quote (') for primary stress and two single quotes ('') for secondary stress. In LaTeX, the latter is generally indistinguishable from as a double quote. This notation has the advantage of being iconic; for example, tertiary stress (in compound words) can be simply included as three single quotes (''').
- Word boundaries: Word boundaries are included as a hash sign '#', which is a standard notation in linguistics. In compound words, word boundaries are expressed as a single hash sign ('#'). In phrasal idioms, word boundaries are expressed as a double hash sign '#').
- Syllable boundaries: Word boundaries are simultaneously syllable boundaries. Those syllable boundaries which are not simultaneously word boundaries are expressed as a period sign ('.'), which is a standard notation for syllables in linguistics.
- Morph boundaries: Word boundaries are simultaneously morph boundaries. Those morph boundaries which are not simultaneously word boundaries are expressed as a plus sign ('+'). Morphs are the phonemic representations (or, in orthography, the orthographic representations) of morphemes (in distinction to more structural or semantic characterisations of morphemes). Note that morphs and syllables are frequently not co-extensive: Morphs may contain more than or less than one syllable, and morphs and syllables may overlap (cf. Verbindung /vE6.bIn.d+UN/ for overlap of the morph /bInd/ with the syllable /dUN/.

The following sections contain a translation of the brief lexicographic specification distributed with the official wordlist Version 1.1 (May 1994).

4.3.2.2 Goals

1. Negotiation of consensus between different ASR groups, data collection groups, lexicon group
2. Definition of lexical coverage for system
3. Provision of information of different types for speech recognition
4. Definition of compact and flexible *Wordform Interchange Format* (WIF)

4.3.2.3 Coverage and negotiated selection criteria

1. Words in artificial reference dialogue
2. Words from 10 selected dialogues
3. Completion of function word sets
4. Completion of closed semantic sets (calendar terms etc.)
5. Initial limitation to 1300 entries for test purposes

4.3.2.4 Wordlist types

The demonstrator wordlist as defined above should not be confused with the following list types:

1. Canonical pronunciation dictionaries for speech synthesis with no alternative lexical variant pronunciations.
2. Corpus based lists of fully inflected forms for the definition of word models in language recognition, which contain lexical variants of different kinds, frequency lists, etc.
3. Wordlists with phonetic (sublexical) pronunciation variants, coupled with pronunciation detail rules.

4.3.2.5 Lexicographic concept for demonstrator wordlist

4.3.2.5.1 Declarative main lexicon

The main VERBMOBIL lexicon is based on a declarative concept and uses a mall set of related lexical formalisms (e.g. STUF3 as general formalism, DATR for mophology) and a flexible and efficient lexical database concept.

4.3.2.5.2 Procedurally specialised daughter lexica

The demonstrator wordlist is a specialised daughter lexicon of the main lexicon.

4.3.2.5.3 Pronunciation table

The wordlist has the form of a table, pairing single orthographic wordforms with (list of) phonological representations, according to current conventions in ASR.

4.3.2.5.4 Orthography

Retention of upper case for substantives; TEX conventions for Eszet and umlauted vowels (”s, ”a, ...); optional separator hyphen for compounds; separator underscore for phrasal entries.

4.3.2.5.5 Complex pronunciation representation

Pronunciation representation with information required by different partners:

- canonical phonemic transcription with defined phoneme set
- syllable separator (point)
- morph separator (plus mark)
- word separator in compounds (double cross), implies morph & syllable boundary
- lexically and morphologically derived primary and secondary accent marks
- alternative non-predictable pronunciations
- heterophonous homographs kept separate

4.3.2.5.6 Simple UNIX processing

The coding of the wordlist must be suitable for elementary processing with UNIX tools based on regular expressions, such as sed, grep.

4.3.2.5.7 Simple database function

With UNIX tools, emulate simple access database functions:

- the set of unstressed monosyllables
- the set of compounds
- the set of polysyllabic noninflected simplex words e.g.:
  ’The set of morphologically simple unstressed monosyllables with short vowels’
  =def grep -v ”[.’+#:]” bodyfile ¿ tinyfile

4.3.2.5.8 Convertibility

Unique convertability into other currnt ASCII encodings, e.g. with different phoneme conventions, different syllable marks, etc. is required.

4.3.2.5.9 Definition of characters and symbols

Orthography

- Capitals: Initial letters of nouns.
- ” German characters ”s, ”a, ”o, ”u, ”A, ”O, ”U
- $ Initial marker for letter names (spellings) $A, ... $Z
- _ Underscore as word separator in phrasal expressions
- - Hyphen as word separator option for compounds

Column separator

- 1 spaces (ASCII 32).
  May easily be replaced by other separator,
  e.g. `sed -e "s/  /*/g" < blankfile > starfile`
  First occurrence of result, etc., can be distinguished:
  e.g. `sed -e "s/[*]/  /1" < starfile > ulmfile`

Canonical phonemic transcription

- ? Standard SAMPA coding of glottal stop is preferred in the lexicon over the Kiel variant ’Q’ (introduced for use where punctuation marks occur in transcriptions).
  Kiel code: `sed -e "s/[?]/Q/g" < thisfile > Qfile`

Word prosodic transcription

- ’ (single quote) morpho-lexical primary accent+

- " (two single quotes) morpho-lexical secondary accent+
  Kiel code: `sed -e "s/’’/\"/g" < thisfile > dquofile`
  Del sec. str.: `sed -e "s/’’//g" < thisfile > nsacfile`
  Del all str.: `sed -e "s/’//g" < infile > naccfile`

Pronunciation separators

- . Syllable separator. Not in Kiel code.
  Erlangen code: `sed -e "s/\./|/g" < infile > outfile`
- + Morph separator.
  Syllable and morph separators are partially independent. Where they occur together, the syllable separator is written before the morph separator.

- # Word separator in compounds, including those containing unique morphs, and separable prefixes.
  The word separator is simultaneously syllable and morph separator.

- ## Word separator in phrasal entries. The phrasal word separator is simultaneously syllable and morph separator.
  Must be processed before simple word boundary mark, e.g. if a different code is required:
  `sed -e "s/\#\#/ /g" < infile > outfile`
- Delete all separators: `sed -e "s/[\.`#]//g” ¡ infile ¿ outfile+

Note: In Kiel code, ’+’ is used redundantly to mark absence of accent in function words;
’#’ is deleted when lexical de-stressing in highly lexicalised compounds occurs. This usage is not compatible with standard linguistic usage as used in this wordlist format.

4.3.3 More complex lexical databases

In a complex project, lexical information from several sources may need to be integrated in a fashion which permits flexible further development work even when the information cannot easily be reduced to a logically fully consistent and well-defined system. A situation such as this will arise when alternative modules, based on different principles, are to be made available for the same system. For instance, two different syntactic components will define different forms of syntactic ambiguity and be associated in different ways with semantic ambiguities. And morphological ambiguities arise with inflected forms in highly inflecting languages. In order to achieve any kind of integration, at least the word form representations will need to be consistent. The hybrid information sources will have to be represented as conjunctions of the values of independent attributes (i.e. fields within a record), with separate disjunctions, where relevant, within fields.

In general, spoken language projects have been based on the idealised notion of a *single*, *well-defined*, *consistent* and *complete*; this situation might reasonably be expected to correspond to the reality of a system developed in a single laboratory at one specific location. However, larger scale projects need to be able to cope with hybrid lexical information of the kind just outlined. A project of this type is the Verbmobil project funded by the German government, with international participation.

A general product–oriented solution would obviouly use a product standard database but an illustration of the typical R&D style UNIX database is given here for the sake of simplicity as an example of a database structure designed for hybrid lexical information.

1. Internal database structure (standard UNIX database format):
   - database: header records followed by body records
   - header: header_record_1 header_record_2 header_record_3
   - body: body_record_1 . . . body_record_n
   - header_record_1: (record containing attribute names, i.e. field names)
   - header_record_2: (record defining internal conjunctive/disjunctive structure of attribute values, i.e. field contents)
   - header_record_3: (record containing source of information)
   - body_record_i: (record containing values for a given entry)

2. Example of record structure:
   - Header: (the designations A3 etc. refer to projects delivering particular types of information)
     Record 1: Orth A3 B1 C1 D1

     Record 2: Orth A3.Phon B1.Wortart,B1.Kasus,B1.Genus,B1.Num,
     B1.Detagr,B1.Definit,1.Semobj,B1.Semattr C1.Syncat1_C1.Syncat2 D1.Syncat
     Record 3: reference.ort a3joha.lex b1naeve.lex c1jung.lex d1peters.lex
   - Body:
     Mutter mU!t6 nomen_akk,fem,sg,@empty@,@empty@,Raute,@empty@;
     nomen,nom,fem,sg,@empty@,@empty@,Raute,@empty@ Nom,OBJEKTTYP nom
   - Note that the spaces designate conjunction (i.e. field separators), while the semicolons designate disjunction
3. Example of human-readable formatting

```
Entry 372: Mutter
  Orth: Mutter
  A3:   mU!t6
  B1:   nomen,akk,fem,sg,@empty@,@empty@,Raute,@empty@
        nomen,nom,fem,sg,@empty@,@empty@,Raute,@empty@
  C1:   Nom,OBJEKTTYP
  D1:   nom
```

On UNIX systems, laboratory-specific acquisition and access routines for ASCII lexical databases are frequently writen with sandard UNIX script languages for ASCII character stream processing. If the resources are available to produce fully specified C and C++ programmes, then of course this is to be preferred. The UNIX tools are useful for prototyping and *ad hoc* format conversion and informal exchange within the speech development community, but are not to be recommended for commercial use.

4.3.4 Querying lexical databases

A complex database will permit access according to a wide variety of criteria, based on Boolean query combinations involving matching of field contents, selection of keys, and so on. However, flexible access to UNIX text databases can be easily designed, using knowledge about the structure of the database, the types of entries, and so on.

The following example illustrates basic UNIX script programming for human-readable format conversion (transformation of selected named attributes of a database record into the attribute format given above). It has been used in the interactive VM–HyprLex Web interface for the Verbmobil lexicographic database.

```
#!/bin/sh
# dbviewr
# D.Gibbon, 20.11.1994
# Prettyprint of single entries
# and attributes in lexicon database,
# with regular expression matching.
# Uses UNIX tools:
#  gawk (i.e. gnu awk), sed, tr
# (Note: sed and tr are used for illustration, and would
#        normally be emulated in gawk)
# Database structure:
# Header: Record 1: Fields containing attribute names.
#         Record 2: Other information.
# Body:   Records >2: Database relation.
# Query usage: Usage: dbviewr dbname attribute* regexp

# Two-line database header
# Version and date stamp:
Firstline="bielefeld.lexdb.v2.1 VM-TP5.9 31 Jan 1995 DG UBI"

# Attribute names associated with columns of database:
Attrib="BIorth BIorthseg BImorpro BIlemma BIorthstem BImorprostem BIflex BICD1 BICD2 BICD3 BICDall KICanon BIdi
```

```
# Basic command line parameter tests:

if [ $# -lt 3 ]
 then
 echo "Error: Possibly no parameters were defined for the \"SELECT Marked\" option."
 exit
fi

if [ $1 = help -a -f $2 ]
 then
 echo "Internal error. Please mail bug report to administrator."
 echo $Firstline
 echo $Attrib
 exit
fi

if [ ! -f $1 ]
 then
 echo "Internal error. Please mail report bug to administrator."
 echo File $1 does not exist.
 exit
fi


# The GNU version of the awk script language is used:
gawk '

# Transfer the keyword from the command line to an awk variable:
BEGIN {keyword = ARGV[ARGC-1]}

# Identify the attributes in the first record whose values
# are to be queried.
NR == 2 {
   {
   for (i=2 ; i < ARGC ; i++)
     {
       for (j=1 ; j <= NF ; j++)
           if (ARGV[i] == $j) {
              attrib[j] = "yes"
              attname[j] = $j
       }
     }
   }
   {
     for (i = 2 ; i < ARGC ; i++)
          ARGV[i]=""
   }
}

# Find required keyword entry/entries in body of database,
# print required values and set 'found' flag:
NR > 2 && $1 ~ keyword {print "\nEntry " NR-2 ":", keyword
        {for (i=1 ; i <= NF ; i++)
        if (attrib[i] ~ "yes") {print "  " attname[i] ":\t" $i}}}

# Print message if no entry was found for the keyword.

 ' $* |

# Format disjunctive field contents with linebreaks and spaces:
sed -e "s/;;/\//g" |
sed -e "s/;/&     /g" |
sed -e "s/=>/&;      /g" |

# Store result in temporary file while counting:
tr ";" "\012" | tee $TMP.0 |

# Count number of hits and emit into standard output:
gawk '
BEGIN {sum=0}
$1~/Entry/ {sum++}
END{print "<br><b>Number of matches =",sum,"</b>"}
    '

# Emit temporary file into standard output and remove:
cat $TMP.0
rm -f $TMP.0
```

For an overview of related format conversion techniques, see Aho et al. (1987), Dougherty (1990), Wall and Schwartz (1991).

4.4 Lexical knowledge acquisition for spoken language

The most general declarative perspective on a spoken language lexicon which is required for lexicon acquisition is that of the "omniscient lexicographer": the lexicographer "knows", in principle, all the possible categories and relations which may hold between a sign, its meaning and pronunciation, and other signs; all properties of a sign have equal status in terms of possible access. This idealised view, while useful for general lexical databases, is not appropriate for the construction of more specialised spoken language lexica for the classical types of spoken language system, unless these are derived from a more general lexical database as sublexica.

A prerequisite for lexical acquisition is to define the following items:

- An application dependent lexicon model which defines relevant types of lexical information.
- A lexical database model which defines the storage structure for lexical information.
- A characterisation of the sources of lexical data (e.g. orthographic transcriptions, other databases; manual information provided systematically or *ad hoc* by a lexicographer)
- Definition of procedures for constructing a lexicon from the data on the basis of these models and sources.
- Definition of procedures for validating the consistency of the database.
- Definition of procedures for extracting complex lexical information (e.g. coverage statistics for attributes and attribute combinations).
- Tools for implementing the procedures.

The logistics involved in these procedures are entirely non-trivial in a large project or consortium, and especially so with a complex spoken language understanding system or speech-to-speech translation system.

4.5 Types of knowledge source

The types of lexical knowledge source for a spoken language system depend largely on the application. There are few general sources of lexical spoken language material (for instance with pronunciation and general frequency information) for any language. The construction of such a source is a major task which requires concerted action on a large scale by specialists of a whole language engineering community. It is a formidable task for many theoretical and practical reasons, but nevertheless one which will require a great deal of effort in the coming years. The two major sources of lexical knowledge for spoken language lexical systems are:

1. existing dictionaries (to some extent),
2. application specific corpora (to a large extent),
3. results of descriptive, theoretical and computational linguistics (to some extent).

There is still a definite lack of general resources in the area (cf. the introduction to this chapter), and the construction of application-derived, generalisable resources will be a major task for any project and for the entire spoken language community in the coming years.

General lexical material is required for the lexical knowledge in general coverage text-to-speech systems, as well as for broad application pronunciation tables for speech recognition.

4.5.1 Dictionaries

Useful sources of information are generally available dictionaries, particularly pronouncing dictionaries, provided that they adhere to accepted standards of consistency and expressiveness of notation, and are available in electronic form. An overview of some sources was given at the beginning of this chapter, and reference should be made to the results of the EAGLES Working Group on Computational Lexica for further examples.

4.5.2 Corpora

Spoken language lexica are application specific, and necessarily so when corpus-derived frequency information is needed. An example of a corpus-derived lexicon type for speech recognition was given above. Another type of corpus-derived lexicon is the diphone word list widely used in speech synthesis technology; for this, phoneme label alignment with the speech signal is required, with the aid of which diphones are defined in the signal for further processing. The chapter on Spoken Language Corpora contains detailed information on procedures of corpus treatment, and the results of the EAGLES Working Group on Text Corpora should also be consulted.

4.5.3 Acquisition tools

At the current state of the art, there are few generally available tools for constructing spoken language lexica, either by extraction from existing dictionaries or from corpora. Lexicon construction usually takes place "in house" in individual laboratories or project consortia; lexicon formats consequently vary greatly. For information on general acquisition tools in the sense of lexicographers' work benches, reference should be made to the results of the EAGLES Working Group on Computational Lexica. It is not appropriate in this context to go into the vast domain of Machine Learning and its application to the (semi–)automatic acquisition of lexica from data.

Of greatest practical use for the development of spoken language lexica in the area of word forms are the tools required for creating different kinds of word form list and word form table from corpora; the general parameters associated with acquiring syntactic, semantic and pragmatic information are not unique to spoken language lexica (though the details, for instance of spoken language dialogue, indeed differ greatly from spoken to written language).

It is a common practice is either to write custom-made programmes in C, or, where speed of processing is not at a premium, to use standard UNIX script languages for processing orthographic transcriptions. Neither of these procedures is particularly difficult, because of the relatively straightforward and well-understood procedures and associated algorithms.

The simplest approach for many applications where processing time is not critical, for instance with small lexica, or where batch-style processing is acceptable, is to use UNIX tools such as *grep*, *tr*, *sed*, *uniq*, *cut*, *tail*, *spell* and *awk*. For descriptions of these tools, a UNIX manual or textbook, or the man page on-line information on a UNIX system should be consulted; techniques for specific database oriented UNIX tools are described by Aho et al. (1987), Dougherty (1990), Wall and Schwartz (1991).

An example of database formatting was given above. Simple examples of UNIX tool applications are illustrated in grossly simplified form below in order to convey an idea of the sort of corpus pre-processing required for ASCII-based spoken language lexicon acquisition.

- Orthographic transcription to word list:

```
#!/bin/sh
# Simple wordlist generator
echo Wordlist generator
tr -sc 'A-Za-z' '\012' < $1 | sort | uniq > wordlist.srt
echo Wordlist in file 'wordlist.srt'
```

- Orthographic transcription to frequency list:

```
#!/bin/sh
# Simple word frequency generator
echo Word frequency generator
tr -sc 'A-Za-z' '\012' < $1 | sort | uniq -c > wordlist.frq
echo Wordlist in file
'wordlist.frq'.
```

- Orthographic transcriptiontranscription!orthographic to digram frequency table:

  ```
  #!/bin/sh
  # Simple digram table generator
  echo Digram generator
  tr -sc 'A-Za-z' '\012' < $1 > lines.txt
  tail +2 lines.txt > tailed.txt
  paste lines.txt tailed.txt | sort | uniq -c > digrams.tab
  echo Digram frequency table in file 'digrams.tab'.
  ```

  Digram frequency information of this type is the basis for the construction of statistical *language models.*; this simple illustration is, however, not to be compared with state of the art technology.

4.5.4 Acquisition logistics

The main issues in lexical acquisition are *extensional* and *intensional* coverage, i.e. the range of entries and the range of properties of lexical entries, and *consistency.* The main steps in acquisition are as follows:

1. Definition of scenario.
2. Definition of word list (inflected forms, for highly inflecting languages perhaps word stems).
3. Definition of corpus transcription conventions, frequently canonical orthographic (not impressionistically modified); general SGML conventions are available but not yet widespread in speech technology.
4. Definition of lexical representation conventions and a filter to derive thlexical representations from the transcriptions.
5. Deployment of grapheme–phoneme conversion (often a combination of stochastic and rule–based conversion with manual checking).
6. Construction of a pronunciation lexicon (orthography–phonology pairs), including lexical (non–rule–derivable) pronunciation variants.
7. Construction of pronunciation variant rule base and derivation of regular pronunciation variants.
8. Integration of other lexical information (syntactic, semantic, pragmatic).
9. Calculation of corpus statistics for lexical items and intensional coverage statistics for fields in the database..
10. Deployment for calculation of language model.
11. Deployment for training of stochastic recognisers.
12. Definition of recognition and dialogue strategies for coping with out–of–vocabulary items.

In a consortium, a major part of the logistics is concerned with negotiation towards mutually agreed goals, adaptation of formats for or by particular partners, and provision of specific information (e.g. syllable boundaries, morphological information) for different approaches or different tasks in speech technology. An overview of the logistics for lexical acquisition in the Verbmobil consortium is given in Figure 4.1; this includes morphological analysis for the completion of non–attested paradigm elements for inflected words. The shaded blocks are those central to lexicon acquisition.

An example of a corpus transcription filter for adapting Verbmobil orthographic dialogue corpus transcription conventions is the `trlfilter` (transliteration filter) software. The term 'transliteration' is Verbmobil terminology for 'orthographic transcription'. The filter is implemented as a UNIX script, and is used as an operational standard device to ensure consistency between the local lexica used by partners, often with local transcription conventions, in different modules. The original design has been considerably modified by Daniela Steinbrecher; the degree of parametrisation is shown in the following descriptions:

```
##  Usage:
##  trlfilter [-aghiprstuvw] InPath [OutPath]
##  or
##  trlfilter -R <file> [-m] [OutPath]
##
##  -R : create reference files from turns listed in file
##  -m : generalization of human noise (<HUM>)
##
##  -a : angle brackets enclosing hesitation particles are not removed
##  -g : no generalization of noise, interruptions and breaks
##  -h : header for output files
```

```
##  -i : removes turnID
##  -p : punctuation is not removed
##  -r : annotates ambiguous reduced wordforms
##  -s : spelling sequences are replaced with <SPELL>
##  -t : removes turns containing:
##          1) <;T>   technical break
##          2) /-     interrupted speech
##          3) +/ /+  restart
##          4) =      word fragment
##  -u : removes interrupted words (X_ _X constructions)
##  -w : interrupted words are joined
```

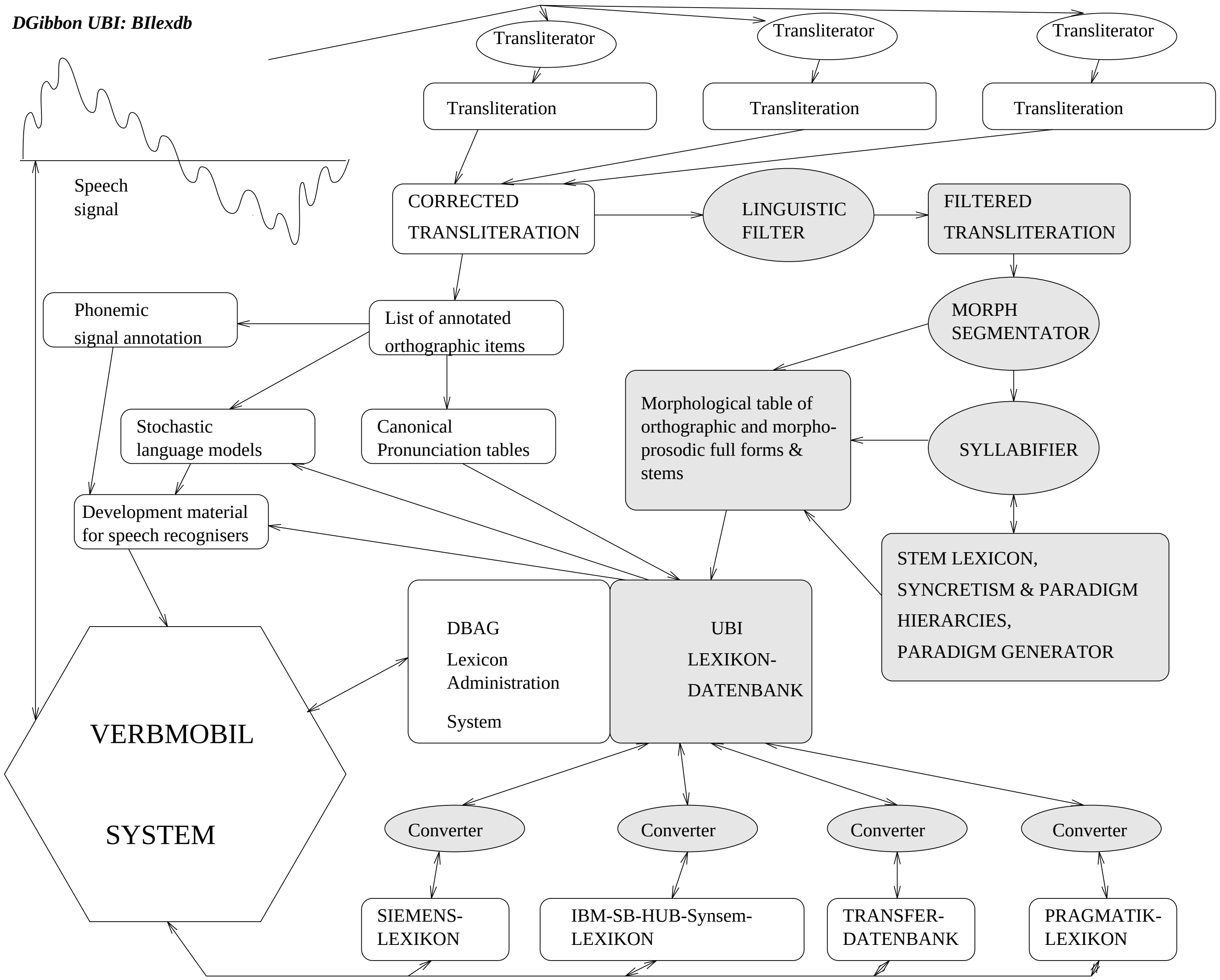

DGibbon UBI: BIlexdb
Transliterator
Transliterator
Transliterator
Transliteration
Transliteration
Transliteration
Speech signal
CORRECTED TRANSLITERATION
LINGUISTIC FILTER
FILTERED TRANSLITERATION
Phonemic signal annotation
List of annotated orthographic items
MORPH SEGMENTATOR
Stochastic language models
Canonical Pronunciation tables
Morphological table of orthographic and morpho-prosodic full forms & stems
SYLLABIFIER
Development material for speech recognisers
STEM LEXICON, SYNCRETISM & PARADIGM HIERARCIES, PARADIGM GENERATOR
DBAG Lexicon Administration System
UBI LEXIKON-DATENBANK
VERBMOBIL SYSTEM
Converter
Converter
Converter
Converter
SIEMENS-LEXIKON
IBM-SB-HUB-Synsem-LEXIKON
TRANSFER-DATENBANK
PRAGMATIK-LEXIKON

Figure 4.1:

5 UNIX tools in and around lexicogrphy

5.1 Overview

The availability of the public domain distributions of LINUX, and relatively inexpensive but powerful processors and large memories in modern PCs, have made UNIX very popular outside the research institutions which were its traditional domain. This chapter cannot replace good introductions to UNIX tools or to programming techniques, but it is intended to convey some of the flavour of rapid prototyping with UNIX text oriented tools, an activity less politely known as UNIX hacking.

A powerful feature of these tools is the use of regular expressions, i.e. generalisations over sets of linear strings, for matching and substitution in texts. It is no coincidence that regular expressions are at the heart of modern computational linguistics, particularly in computational phonology and morphology, but also in syntax and dialogue management for spoken language. There are many tasks which can be performed practically be means of UNIX tools which have a theoretically satisfying correlate in terms of comptational models of language, including, for instance, syllabification in phonology, morphophonemic and morphographemic rules, or rule–based grapheme–to–phoneme conversion. Other, more conventional uses include the creation of lexical databases, KWIC (Key Word In Context) concordances, file format conversions such as the generation of readable LaTeX or HTML formats from database entries.

5.2 Generally useful programmes

The following tools are particularly useful for a variety of purposes. Some are standardly provided with UNIX or the most widespread UNIX windows server, known as X-Windows or X11. installations. Others need to be downloaded from various ftp and web sites worldwide. This selection is small, personal, and no doubt everyone will find some 'essential' item missing.

FORMATTING, PRINTING: Document description languages or formatting languages are required for many purposes. A document formatted in LaTeX, such as this text, goes through at least two stages of re–formatting before printing: first, it is converted into dvi (device independent) format, and then it is converted, for example, into PostScript.

- latex: Standard scientific and technical document description language (formatting language), originally a set of standard high–level text structure macros for books, articles, reports etc., to save using detailed defniitions in the TeX formatting language. Although these high–level macros are often referred to in the Computer Science literature as 'logical' structure, this is of course nonsense to a linguist, since they have very little to do with logic or logical form; the macros may be seen from a linguistic point of view as text–linguistic descriptions.
- latex2html: A well-thought out programme for converting the 'latent' hypertext structures in LaTeX documents into hypertext, with sequential and hierarchical navigation concepts; it is not foolproof with complex LaTeX structures and special characters.
- dvips: Format converter from dvi to PostScript.
- psnup, psselect, ... A family of format converters to print several PostScript logical pages on one physical page, select pages from a PostScript document, etc.
- ps2ascii: Format converter to extract ASCII text from a PostScript file.
- ghostscript: Format converter for PostScript files into other formats.

GRAPHIC PREVIEW: Graphics programmes for previewing formatted files before printing.

- xdvi: A fast previewer for dvi format.
- ghostview: A slow previewer for PostScript format; uses ghostscript.

EDITING: There are many editors available for UNIX environments; the following two are oriented towards standard terminals.

- vi: The standard UNIX full screen editor for VT100 terminals, xterm, for which every UNIX freak should know the basic jargon. It supports search by regular expression, can automatically update files, and sends a friendly email about how to find the file you were working on when the system crashed. It works everywhere, anywhere, and mostly even when nothing else works.
- emacs: A sophisticated standard editor originally written in LISP, contains its own variety of LISP as a macro language, supports editing and running programmes from the same environment, has structure highlighting for everything from LISP to HTML.

GRAPHICS: A variety of tools are available for graphics construction and editing; the following are just a small selection.

- xfig: For drawing and editing vector graphics. A wide variety of graphics formats, including LaTeX picture macros, can be exported. The xfig files are object definitions in ASCII format, and can easily be used for the automatic generation of vector from numerical or other data using UNIX stream editing techniques. Figure 5.1 was generated using this technique.

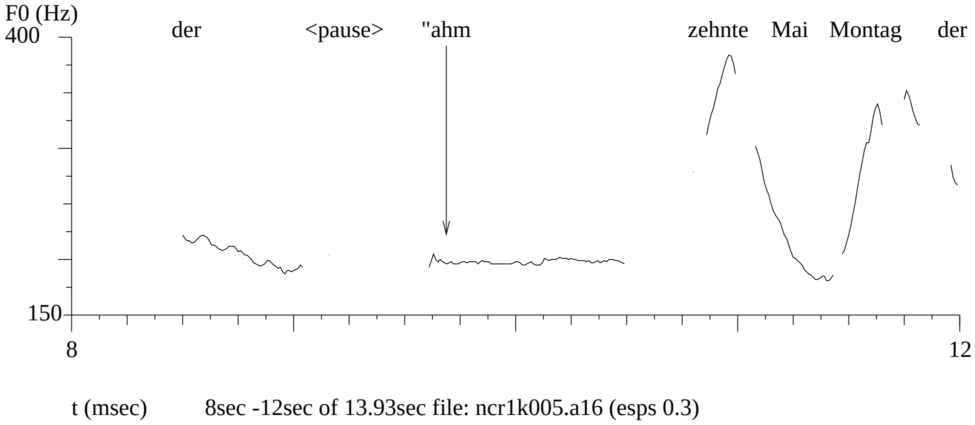
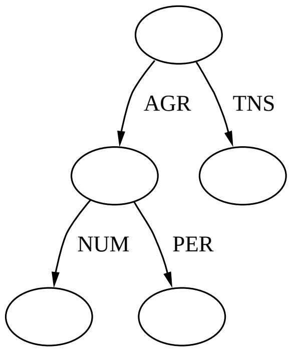


Figure 5.1: Discourse particle intonation

- xv: For editing and reformatting bitmap graphics.
- dot: A neat and simple programme developed at AT&T research for automatically drawing directed graphs. For example, the graph in Table 5.2 was drawn this way (the attribute–value structure was drawn with a LaTeX macro). The 'dot' definition from which this graph was generated will be shown below in the context of file reformatting.

Table 5.1: Attribute–value matrix and directed graph

AVM	*translates to*	DAG
$\begin{bmatrix} \text{AGR} = \begin{bmatrix} \text{NUM} = \text{plural} \\ \text{PER} = \text{second} \end{bmatrix} \\ \text{TNS} = \text{past} \end{bmatrix}$		

- pstopnm: Format converter to standard pnm/ppm graphic format (beware: huge output files!)
- ppmtogif: Format converter from standard pnm/ppm graphic format to other formats (here to gif).

Net: These Internet communication programmes are presumably well known, and are currently generally part of any installation.

- telnet
- ftp
- elm
- lynx
- netscape

Script languages:

A script language is a programming language in which instructions are written line by line, and which processes a UNIX text line by line. This technique provides an extremely convenient basis for constructing lexica, concordances and hypertexts as UNIX databases.

- sed: A standard programme for automatically editing and reformatting UNIX text files.
- awk, gawk: sed and much more, including branches, loops and other control structures.
- perl: gawk and much more, including associative data structures and faster processing (latex2html and many other common programmes are written in perl).

5.3 Pipes and strings and scripts and things

Before using UNIX tools it is necessary to understand a few basic things about processing under UNIX: the basic elements are characters, concatenated into sequences called character streams.

The newline character (ASCII octal `\012`, decimal `10`) segments of the character stream into lines; note that MS texts have an additional carriage return character (ASCII octal `\015`, decimal `10`) before the newline character. Tools called dos2unix and unix2dos are available which perform the required conversion, but a simple UNIX tool application which converts these characters into blanks is the following:

```
+tr "\015" " "+
```

The character streams are sent to a UNIX process (a UNIX programme running at a specific time with a specific process number) from a file, or through a 'pipe' (a channel which links the output of one process to the input of another without intervening files), as in the following example:

```
cat dosfile | tr "\015" " " > unixfile
```

The file 'dosfile' is sent by cat (file listing) through a pipe (symbolised by '—') to tr for processing, and tr sends it into the file 'unixfile'. Input from a pipe or the keyboard is referred to as 'standard in' or 'stdin', and output to a pipe or to the screen is referred to as 'standard out' or 'stdout'.

A typical UNIX shell script, in which several programmes can be used to send streams in sequence to each other through pipes or files has already been shown in connection with database structures. A script for the most common UNIX shell language always starts with the first line `#!/bin/sh` which instructs UNIX to execute the command /bin/sh, in this case to produce a shell or new working environment for the following programmes.

5.4 A selection of UNIX tools

There are a number of good books on programming with UNIX tools, therefore details are not given here. Immediate help on the use of these tools can also be obtained from UNIX 'man pages' (manual pages); e.g. the UNIX command 'man comm' provides the definition of the 'comm' programme (if the man pages are installed, of course).

Name	*Description*	*Typical use*
cat	List the contents of a file or files.	Useful for inspecting files, or for starting a pipe with a UNIX text stream.
cd	Change directory.	Often needed for local processing of directories with intermediate and final results.
chmod	Change read–write–execute permissions on files (important when creating UNIX scripts, modifying email files.	
comm	Compare sorted files with results in 3 columns, with differences and shared lines.	Useful for emulating set differences and intersections in extracting sub–databases.
cp	Copy files.	
cut	Select columns of a UNIX text file.	gawk is generally more flexible for database processing.

date	Generate the date as an ASCII string.	Use to produce time stamps on your automatically generated HTML pages.
echo	Write a string to a file or the screen.	Provides information on progress in long scripts, and writes strings (for instance with a datestamp).
exit	Terminate a UNIX shell.	In scripts, exit after an error, or insert an exit to aid in debugging.
gawk	The GNU version of awk, the UNIX answer to BASIC (but more powerful and flexible).	Use for any programming task where speed is not the prime requirement (in which case, use C); combines the facilities of sed, cut etc. with standard programming language control constructs.
grep	Selects lines from a UNIX text stream with regular expression matching.	Very useful for creating concordances and subdatabases.
head	Extracts the first part of a file.	Useful for extracting fixed–length headers or inspecting the beginning of a file.
join	Creates a new database from two sorted database files using shared keys (items in the leftmost column).	Useful for adding new attributes to databases, or for selecting a sub–database on the basis of a list of keys.
mkdir	Create a sub–directory.	May be needed for making a directory to store intermediate or final results.
mv	Rename a file.	Useful for manipulating temporary files.
paste	Adds new new columns to a database by appending rows from each input file in the order in which they occur in the files.	Add columns to databases (but only after correctly padding empty fields).
rev	Reverse order of characters in lines.	Use it to produce rhyming dictionaries or do suffix chopping.
rm	Delete files.	Remove existing temporary files before and/or after proceeding.
sed	Line–by–line stream editor for substring substitutions in UNIX text files on the basis of regular expressions.	Indispensible for re–formatting, removing punctuation marks, even finite–state–phonology based grapheme–phoneme conversion.
sleep	Delay for a certain number of second.	Useful for slowing down screen information in debugging.
sort	Sorts lines in a file according to various criteria.	sort -u is the basis for emulating set union of UNIX text files.
spell	Reports spelling errors.	Saves all kinds of trouble.
tail	Extracts the last part of a file.	Useful for removing headers or inspecting end of file.
tee	Writes the stream of data in a UNIX pipe into a file.	Useful for creating intermediate outputs or temporary files for debugging in a pipe environment.
touch	Creates an empty file.	Useful at the beginning of repeated file append operations.

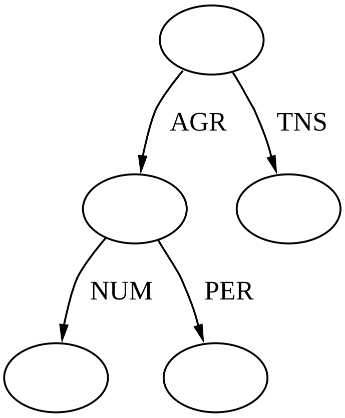
tr	Translates characters.	Useful for introducing or removing control characters such as newline or tab, or changing between upper and lower case.
uniq	Removes duplicates from a sorted list.	Useful for creating sets of non–identical records in databases.
wc	Counts the number of bytes, words, lines in an ASCII file.	Useful for coarse validation of selection operations with grep and gawk.

5.5 Reformatting a graphics definition: simple illustration

Visualisation is an essential procedure for planning and testing complex structures; formatting and re–formatting tools aid this process. A basic attribute–value structure can be visualised as a feature bundle or as a directed graph, as already shown. Both of these visualisations can be generated from the same basic structure of triples.

However, the formats required by UNIX–based facilities can sometimes be simplified further; this is an illustration of a tool which translates simpler formats, which may be more appropriate for manual construction into more complex formats, in order to avoid the error–prone typing of details. An example of such a tool is shown below, which takes the definition shown at the left of Table 5.2 and creates the file shown in the centre, converting this description into a PostScript file, which is reproduced at the right.

Table 5.2: Re–formatting a graph description.

node1 node2 AGR
node1 node5 TNS
node2 node3 NUM
node2 node4 PER
	digraph avm {
	node1 [label=""]
	node2 [label=""]
	node3 [label=""]
	node4 [label=""]
	node5 [label=""]
	node1 -> node2 [label="AGR"]
	node1 -> node5 [label="TNS"]
	node2 -> node3 [label="NUM"]
	node2 -> node4 [label="PER"]
	}
		

```
#!/bin/sh
# makedot
# D. gibbon
# 7 Jul 1997
# Create AT&T dot file from list
# Input: List of arc triples (node node label).

INFILE=$1
INFILENAME=‘echo $1 | sed "s/\..*//g"‘

echo digraph $INFILENAME { > tmp.out

# Making node definitions
cat $INFILE |
gawk '{print $1,$2}' |
tr " " "\012" |
sort -u |
sed "s/^.*$/& [ label=\"\" ]/g" >> tmp.out

# Making edge definitions
cat $INFILE |
gawk '{print $1 " -> " $2 " [ label=\"" $3 "\" ]"}' >> tmp.out
echo "}" >> tmp.out

mv -f tmp.out $INFILENAME.dot
dot -Tps -o $INFILENAME.ps $INFILENAME.dot
```

5.6 Generating a Web hyperconcordance with UNIX tools

The availability of lexical resources is a big problem, and the consistency of databases in distributed cooperative projects is extremely important. The World Wide Web is increasingly being used to overcome these problems, and a number of Web applications are illustrated here.

The first is a rather simple application, colouring HTML files to get away from classic Web grey. The second puts an arbitrary preformatted text into an HTML file and surrounds it with appropriate HTML markup and a date stamp (no excuse for having pages without date stamps any more). The third application is rather complicated, namely the construction of a hyperconcordance, i.e. a Web concordance which utilises hypertext structure. The application was written for this text, and can be improved in many ways. Examples can be found on my Web pages.

5.6.1 Colouring an HTML document

using a UNIX loop to colour all HTML files in a directory if one does not like classic Web grey. The primary aim is to colour the background but, if desired, the text, link text, etc. may also be coloured. This script was written to change the output of large latex2html documents, as it is tedious and unreliable to do this manually.

```
#!/bin/sh
# htmlcolor
# D. Gibbon, 11 May 1996
# Change background colour, etc., of html documents

COLOR=$1

for file in `ls *.html`

do
     echo Colouring $file ...
     sed "s/<[Bb][Oo][Dd][Yy][^>]*>/<body bgcolor=$COLOR>/g" $file > tmp
      mv -f tmp $file
done
```

5.6.2 Putting arbitrary text into an HTML file

The following script simply takes a file as input and emits the text as preformatted HTML to standard out, from where it can be directed to a file, for instance: '`makehtml myfile > myfile.html`'.

```
#!/bin/sh
# makehtml
# D. Gibbon
# 9 July 1995
# Create the skeleton of an html document

# Local variables
SERVER="coral.lili.uni-bielefeld.de"
NAME="Dafydd Gibbon"
TITLE=$1

# Standard variables set using UNIX commands
USER=`whoami`
TIMESTAMP=`date`

INFILE=$1
```

```
#---------------------------------------------------
echo '<!doctype html public "-//W30//DTD W3 HTML 2.0//EN">'
echo '<!-- Header and start of body -->'

echo '<html><head>'
echo '<title>'$TITLE'</title>'
echo '</head><body bgcolor=fffff0><address>Universit&auml;t Bielefeld -
     Fakult&auml;t f&uuml;r Linguistik und Literaturwissenschaft</address><hr>'

# ---------------------------------------------------
echo ''

echo '<!-- BODY OF HTML DOCUMENT -->'
echo '<pre>'
echo ''
echo ''

cat $INFILE

echo ''
echo ''
echo '</pre>'

echo '<!-- Conventional footer -->'
echo '<hr>'
echo '<address>'
echo '<a href="http://'$SERVER'/~'$USER'/"> '$NAME'</a>,'
echo $TIMESTAMP
echo '</address>'
#---------------------------------------------------
echo '<!-- End of body and file -->'
echo '</body>'
echo '</html>'
echo ''
```

5.7 A hyperconcordance for the web with UNIX

The concordance described here is a static concordance which is compiled at one time, and creates a complete fixed hypertext. Several examples can be found on my web pages. This technique has the disadvantage that it is inflexible (the search categories are fixed) and can generate very large files (of the order of 1000 greater than the text itself).

A different technique creates dynamic concordances, using active online search in the text database; this technique is also illustrated on my Web pages.

An example of the output of the static concordance for “Old Possum’s Book of Practical Cats”, by T. S. Eliot, for the word ‘away’ is:

```
AWAY

[f=6/ 1663( 5882), N/types=0.00360794, N/tokens=0.00102006]

[]    (1) And his bosun, TUMBLEBRUTUS, he too had stol'n [ away ] -
[]    (2) If you put it [ away ] on the larder shelf.
[]    (3)    But the dogs and the herdsmen will turn them away.
[]    (4) The big Police Dog was [ away ] from his beat -
[]    (5) 'It must have been Macavity!' - but he's a mile away.
```

A click on the left–hand button will show the whole immediate text environment for the line concerned. The architecture ofthe hyperconcordance is shown in simplified form in figure 5.2. The concordance consists of three basic objects, the TEXT, the Alphabetic List, and the Frequency List. The entries in the alphabetic list point to the concordance objects, which are sets of contexts for the word concerned, as

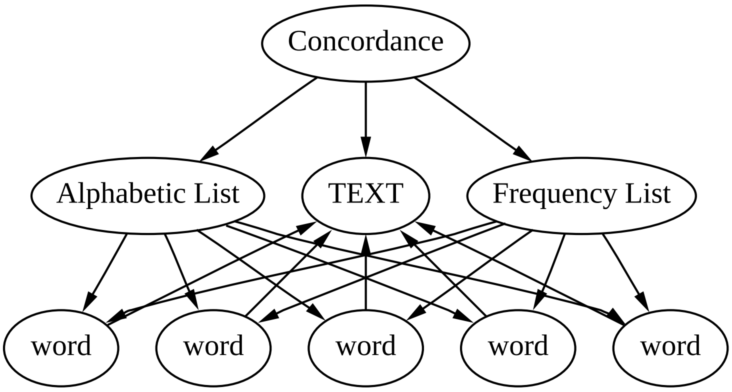


Figure 5.2: Hyperconcordance structure.

illustrated above. Each of these contexts in turn points back to the text. The Frequency List is similar, but the words are ordered into ranks by frequency.

```
WORD LIST SORTED BY FREQUENCY

 REF RANK WORD COUNT/TYPES(TOKENS)

 []      1 THE        341/1663(5882)
 []      2 AND        257/1663(5882)
 []      3 A          179/1663(5882)
 []      4 OF         127/1663(5882)
 []      5 TO         122/1663(5882)
 []      6 HE          99/1663(5882)
 []      7 IN          94/1663(5882)
 []      8 YOU         88/1663(5882)
 []      9 IS          77/1663(5882)
 []     10 HIS         75/1663(5882)
 []     11 THAT        58/1663(5882)
 []     12 CAT         56/1663(5882)
 []     13 I           54/1663(5882)
```

The concordance is generated by a complex script which takes a plain text as input, and produces the hyperconcordance as output. In this version, a number of variables have to be set manually in the script which would be better obtained from the command line. The script is deliberately written using a variety of UNIX tools rather than, say, gawk or perl, in order to show the different specialisations which UNIX tools offer.

The programme follows the strategy of first constructing a LaTeX file which can be printed as a paper document, and then converting this into HTML.

1. Variable definitions.
2. Preprocessor: Construct LaTeX file header.
3. Preprocessor: Add line labels to input text, for back–reference from concordance (this permits the including LaTeX section heads if the input text file is more than a chunk of text).
4. Create a file with all LaTeX document markup removed for converting into concordance (in case the input text was already in LaTeX), and process punctuation marks.
5. Sort and count the words in the file.
6. Construct frequency list.
7. Process for upper case letters, and (not very efficient, this) create a database of pairs of the original text and the normalised text.
8. Cycle through words in the word list ...
9. Cycle through lines in the text ...
10. Write matches into the concordance.
11. Format with LaTeX (and patch the LaTeX .aux file to cure an obscure feature of latex2html behaviour).

12. Generate hyperconcordance with latex2html.
13. Colour the hyperconcordance files.

Remember: the code was produced *ad hoc* for UNIX teaching purposes and needs re–designing for more general applications.

```
#!/bin/sh
# html2lex
# D. Gibbon
# 25 May 1997
# Convert LaTex text to hyperlexicon

# Hypertext concept:
# 1. Text (terminal).
# 2. Frequency dictionary: words linked to concordance entries.
# 3. Concordance: words linked to lines in text.
#
# Possible extensions:
# Text with links to frequency dictionary entries.
#
# Note: lines are fixed.

INPUT=‘echo $1 | sed "s/\..*//g"‘
echo $INPUT
NAME="Beckett"
TITLE="CONCORDANCE: Beckett Excerpt"
INPUTFILE=$1
LATEXFILE=${INPUT}lex.tex

rm -f tmp.*
cleantex

##########################################################################
# Preprocessor: Adding LaTeX line labels and newfiles

echo Preprocessor: constructing dictionary files in LaTeX ...

##########################################################################

echo "\\documentclass[11pt,twoside,a4paper]{article}"      > tmp.a
echo "\\\begin{document}"                                 >> tmp.a
echo "\\\title{$TITLE}"                                   >> tmp.a
echo "\\author{Samuel Beckett}"                           >> tmp.a
echo "\\date{Designed 2 July 1997 for David C. Weichert\\\\\\\Constructed
     \\\today}" >> tmp.a
echo "\\maketitle"                                        >> tmp.a
echo "\\\tableofcontents"                                 >> tmp.a

echo "\\section{$NAME}\n"                                 >> tmp.a
dos2unix $INPUTFILE                                       >> tmp.a

echo "\\section{$TITLE}"                                  >> tmp.a
echo "\\input{$INPUT.flist}"                              >> tmp.a
echo "\\input{$INPUT.conc}"                               >> tmp.a
echo "\\end{document}"                                    >> tmp.a


cat tmp.a |

sed "s/^  *//g" |

gawk ’
BEGIN {
```

```
LINECOUNT=1
TEXTLINE="false"
NEWSECTION="false"
PREVIOUS=""
}

$0 ~ /\\end\{document\}/ {
TEXTLINE="FALSE"
}

$0 ~ /\\section/ {
$0="\n" $0 "\n"
NEWSECTION="true"
TEXTLINE="true"
}

$0 !~ /\\section/ && $0 !~ /\\input/ && $0 != "" && TEXTLINE=="true" {
if (PREVIOUS=="" && NEWSECTION=="false")
ADDLINE="\\ \\\\\n"
else ADDLINE=""
$0 = ADDLINE "\\label\{line:" LINECOUNT "}" $0 "\\\\"
LINECOUNT++
NEWSECTION="false"
}

$0 ~ /\\input/ {
TEXTLINE="false"
}

{
if ($0 != "") print
PREVIOUS=$0
}
' > $LATEXFILE

###########################################################################
# Remove LaTeX document markup

echo Removing LaTeX document and text markup ...

cat $LATEXFILE |
grep -v "\\\documentclass" |
grep -v "\\\begin{document}" |
grep -v "\\\end{document}" |
grep -v "\\\usage" |
grep -v "\\\maketitle" |
grep -v "\\\tableofcontents" |

# Remove LaTeX text markup
sed "s/\\\textit{//g
s/\\\textbf{//g
     s/\\\textsc{//g
     s/\\\[^{]*{[^}]*}//g
     s/\\\ //g
     s/\\\\\\\\//g
     s/}//g" |

# Remove empty lines (change!)
grep "[A-Za-z]" |
tee $LATEXFILE.txt |

tr "[.,;:!?\`()]" " " |
sed "s/’ / /g
     s/’$//g
     s/^- / /g
     s/ -$/ /g
     s/ - / /g
     s/~/ /g" |

tr " " "\012" |
grep . |

tr "[A-Z]" "[a-z]" |
sort |
tee tmp.list | wc -l > tmp.count


########################################################################

echo  Sorting, counting ...

cat tmp.list |
uniq -c |
sort -rn |

gawk ’ {print $2,$1} ’ | tee tmp.frq |
sort |
tee tmp.lex |
gawk ’{print $1}’ |
tr "[a-z]" "[A-Z]" > tmp.caps
paste tmp.lex tmp.caps |
tr "\011" " " | tee $LATEXFILE.lex |
wc -l >> tmp.count

########################################################################

echo Wordlist sorted by frequency ...

echo "\\subsection{\\\textbf{WORD LIST SORTED BY FREQUENCY}}"
     > $INPUT.flist.tex

echo "\\\begin{tabular}{crlr}" >> $INPUT.flist.tex
echo "\\\textbf{REF} & \\\textbf{RANK} & \\\textbf{WORD} &
     \\\textbf{COUNT/TYPES(TOKENS)}\\\\\\" >> $INPUT.flist.tex

# Add type and token constants to frequency count file:
TYPESTOKENS=‘cat tmp.count | tr "\012" " " | gawk ’{print "\\\/" $2 "(" $1")"}’‘
echo Types and tokens: $TYPESTOKENS

cat tmp.frq |
sed "s/$/$TYPESTOKENS/g" |
tr "[a-z]" "[A-Z]" |
sed "s/^/\\\textbf{/g
     s/ /} /g" |
gawk ’{
      gsub(/\\textbf\{/,"",$1)
      gsub(/\}/,"",$1)
# Because of GUMBIE~CAT ...
      gsub(/[~\047]/,"-",$1)
# Create reference to the line in which the word occurs:
      print "\\ref\{word:"$1"\} "NR " " $0
      }’ |
sed "s/ / \& /g
     s/$/\\\\\\\\/g" >> $INPUT.flist.tex
echo "\\end{tabular}" >> $INPUT.flist.tex

##########################################################################
# Process text file for CAPs

echo Processing text file for capitals, normalising punctuation marks ...

cat $LATEXFILE.txt |
tee tmp.0 |
# Surround by blanks to aid search.
# Note on apostrophe:
# - Non-final apostrophe preserved.
# - Ambiguity between nf apostrophe and single quote, and both removed.
# - Code single quote, eg with =’?
sed "s/[.,;:?!]/ & /g
#     s/’[^ ]/ & /g
#     s/’$/ &/g
     s/^/ /g
     s/$/ /g
     s/  */ /g" |
tr "[A-Z]" "[a-z]" > tmp.1

# Paste text, lower case text, upper case text:
echo Paste text, lower case text, upper case text ...
paste tmp.0 tmp.1 > tmp.2

# Housekeeping:
rm -f tmp.0 tmp.1
rm -f $INPUT.conc.tex

# for each word in wordlist (UNIX loop),
# for each line in text,
# collect lines which match a word in the wordlist

OPCOUNT=‘cat tmp.count | gawk ’BEGIN{PROD=1} {PROD=PROD*$0} END {print PROD}’‘
echo ’Making concordance (’$OPCOUNT’ matching operations) ...’

for WORD in ‘cat $LATEXFILE.lex | tr " " "_"‘
do

echo $WORD ... | sed "s/.*_//g"

echo $WORD |
cat tmp.count - tmp.2 |

##############################################################################
gawk ’

# Load token count from tmp.count
NR==1 {TOKENS=$0}

# Load type count from tmp.count
NR==2 {TYPES=$0}

# Load lc name, frequency, uc name from pipe
NR==3 {
       split($0,WordFreq,"_")
       WORD=WordFreq[1]
       FREQ=WordFreq[2]
       WORDCAP=WordFreq[3]
       WORDGUMBIE=WORDCAP
#        gsub(/[~\047]/,"-",WORDGUMBIE)
       COUNT=0
      }

# For each new word introduce a subsection
NR==4 {
      print "\n\\subsection\{\\textbf\{"WORDCAP"\}\}"
      print "\n\\texttt\{\[f="FREQ"\/"TYPES"("TOKENS"), N\/types="FREQ/TYPES",
             N\/tokens="FREQ/TOKENS"\]\}\\label\{word:"WORDGUMBIE"\}\\\\\n\n"
      }

# For each occurrence create a concordance line
# NR>3 && $0 ~ /[^A-Za-z]"WORD"[^A-Za-z]/ {
NR>3 && $0 ~ " "WORD" " {
      COUNT++
      gsub("[ .!?:;,]"WORD"[ .!?:;,]"," \\textbf\{[&]\} ")
      split($0,Output,"\t")
      printf("\\pageref\{line:%d\} (%d) %s\n",NR-3,COUNT,Output[1]"\\\\")
      }

' |

##############################################################################
# Restore punctuation marks
sed "s/ \([.,;:?!]\)/\1/g
     s/^ //g
     s/ $//g" >> $INPUT.conc.tex

done

##############################################################################

# Format with Latex and PS to make sure fonts are available and 12h
echo =================================================================
echo Format with Latex and PS to make sure fonts are available and 12h
latex $LATEXFILE
latex $LATEXFILE
dvips -o $INPUT.ps $INPUT.dvi

echo =================================================================
echo Patch to force 12h to use text [C] for text cross-references
echo by replacing section.subsection reference numbering
echo
cat ${INPUT}lex.aux |

gawk '

$0 ~ newlabel {
  gsub(/{{}[^\.]*\.[^}]*/,"{\\textbf\{[C]\}")
}

$0 ~ numberline {
  gsub(/{[^{}\.]*\.[^}]*/,"{\\textbf{[C]\}")
}

{ print }

' > tmp.4

mv -f tmp.4 ${INPUT}lex.aux
```

```
echo =================================================================
# latex2html -html_version 3.0 -split 4 -link 1 -t "$TITLE" $NAME.tex
latex2html -html_version 3.0 -split 4 -link 1 -t "$NAME" $LATEXFILE

echo =================================================================
echo Colouring hypertext output ...
cd ${INPUT}lex
htmlcolor "fffff0 text=882200"

echo =================================================================
echo Changing l2h icon references to local directory ...
icons2local
echo Copying l2h icons to local directory gifs/ ...
mkdir gifs
cp -f $WWWHOME/icons/latex2html/* gifs/
cp -f ${INPUT}lex.html index.html
cd ..
```

5.8 Interaction on the Web: dynamic concordances

The access routine for elementary databases which has already been described in a previous chapter may be given a Web interface and used interactively. The Web interface has three sections:

1. A HTML page with form menu.
2. A CGI (Common Gateway Interface) UNIX script to interface with the concordance search programme.
3. A UNIX script to perform the required search or other action and return the required result.

5.8.1 HTML form menu: the VERBMOBIL lexical database

The HTML form provides a wide range of selection and search facilities, as well as context–sensitive help for database attributes and technical report documentation. The Web appearance of the menu is shown in Figure 5.3, and the HTML code (which was automatically generated from the database itself with a UNIX script) is given below.

```
<html>
<header>
<title>VM-HyprLex bielefeld.lexdb.v3.3</title>
<body bgcolor=#ffffe0>
<center>
<h3>VM-HyprLex Interface 3</h3>
<b>bielefeld.lexdb.v3.3, Mar 18 1996<br>
8081 data records, 35 attributes)</b>
</center>
<form method=post action=/cgi-bin/cgi_hyprlexx>
<input type=hidden name=lexdb value=fp3>
<input type=hidden name=attrdef
value="BIorth BIorthseg BImorpro BIorthstem BIphonstem BIflex BIlemma BIspell
 BIproper BIcompsem BICD1 BICDall BIpercent BIrank BIortherror BLAUBEU DemoWL
 RQH-WL BIhitlist FPWL3 KIcanon KIfreq IMSlem IMSpos IMSfreq SIEMENSorth
 SIEMENScats SIHUBval BIgloss IBMorth IBMmorph IBMHUBsyn TUBsem TUEBcomp
 IMSrule">
<center><table border=0>
<tr><td align=right>
<b><select name=keytype>
<option selected>String
<option>Substring
<option>MorLemma
<option>SemLemma
<option>Global
<option>A-V-Pairs
</select></b>
<td><input type=text name=word size=25 value=Terminabsprache>
<td><b><a href=hyprhelp.html#input>KEY</a> type and string </b>
```

```
<tr>
<td align=right>
<select name=atype>
<option selected>Key
<option>SubDB
<option>Corpus-WL
<option>Demo-WL
<option>Blau-WL
<option>RQH-WL
<option>Hitlist
<option>FP-WL
</select>
<td>
<b><a href="hyprhelp.html#search">KEY / SubDB SEARCH</a></b>
<td align=center rowspan=2>
<hr>
<input type=reset value=" Defaults ">
<input type=submit value=" Consult lexicon ">
<hr>
<a href=bielefeld.lexdb.v3.3.sta><b>Coverage</b></a>
&nbsp; &nbsp;
<a href=hyprhelp.html#operate><b>Operation</b></a>
<hr>
<tr>
<td align=right>
<select name=qtype>
<option selected>Marked
<option>All
</select>
<td><b><a href="hyprhelp.html#display">ATTRIBUTE DISPLAY</a></b>
</table>
</center>
<center>
<table border=0>

<tr>
<tr><td colspan=10>
<tr><td colspan=10><b>Morphology, Morphophonology, Morphosemantics</b>
<tr>
<td align=right>
<input type=checkbox name=i value=BIorth>
<td>
<a href=hyprhelp.html#BIorth><b>BIorth</b></a>
<td align=right>
<input type=checkbox name=ii value=BIorthseg>
<td>
<a href=hyprhelp.html#BIorthseg><b>BIorthseg</b></a>
<td align=right>
<input type=checkbox name=iii value=BImorpro>
<td>
<a href=hyprhelp.html#BImorpro><b>BImorpro</b></a>
<td align=right>
<input type=checkbox name=iv value=BIorthstem>
<td>
<a href=hyprhelp.html#BIorthstem><b>BIorthstem</b></a>
<td align=right>
<input type=checkbox name=v value=BIphonstem>
<td>
<a href=hyprhelp.html#BIphonstem><b>BIphonstem</b></a>

<tr>
<td align=right>
```

```
<input type=checkbox name=vi value=BIflex>
<td>
<a href=hyprhelp.html#BIflex><b>BIflex</b></a>
<td align=right>
<input type=checkbox name=vii value=BIlemma>
<td>
<a href=hyprhelp.html#BIlemma><b>BIlemma</b></a>
<td align=right>
<input type=checkbox name=viii value=BIspell>
<td>
<a href=hyprhelp.html#BIspell><b>BIspell</b></a>
<td align=right>
<input type=checkbox name=ix value=BIproper>
<td>
<a href=hyprhelp.html#BIproper><b>BIproper</b></a>
<td align=right>
<input type=checkbox name=x value=BIcompsem>
<td>
<a href=hyprhelp.html#BIcompsem><b>BIcompsem</b></a>

<tr><td colspan=10>
<tr><td colspan=10><b>Corpus distribution, selection, tagging</b>
<tr>
<td align=right>
<input type=checkbox name=xi value=BICD1>
<td>
<a href=hyprhelp.html#BICD1><b>BICD1</b></a>
<td align=right>
<input type=checkbox name=xii value=BICDall>
<td>
<a href=hyprhelp.html#BICDall><b>BICDall</b></a>
<td align=right>
<input type=checkbox name=xiii value=BIpercent>
<td>
<a href=hyprhelp.html#BIpercent><b>BIpercent</b></a>
<td align=right>
<input type=checkbox name=xiv value=BIrank>
<td>
<a href=hyprhelp.html#BIrank><b>BIrank</b></a>
<td align=right>
<input type=checkbox name=xv value=BIortherror>
<td>
<a href=hyprhelp.html#BIortherror><b>BIortherror</b></a>

<tr>
<td align=right>
<input type=checkbox name=xvi value=BLAUBEU>
<td>
<a href=hyprhelp.html#BLAUBEU><b>BLAUBEU</b></a>
<td align=right>
<input type=checkbox name=xvii value=DemoWL>
<td>
<a href=hyprhelp.html#DemoWL><b>DemoWL</b></a>
<td align=right>
<input type=checkbox name=xviii value=RQH-WL>
<td>
<a href=hyprhelp.html#RQH-WL><b>RQH-WL</b></a>
<td align=right>
<input type=checkbox name=xix value=BIhitlist>
<td>
<a href=hyprhelp.html#BIhitlist><b>BIhitlist</b></a>
<td align=right>
```

```
<input type=checkbox name=xx value=FPWL3>
<td>
<a href=hyprhelp.html#FPWL3><b>FPWL3</b></a>

<tr>
<td align=right>
<input type=checkbox name=xxi value=KIcanon>
<td>
<a href=hyprhelp.html#KIcanon><b>KIcanon</b></a>
<td align=right>
<input type=checkbox name=xxii value=KIfreq>
<td>
<a href=hyprhelp.html#KIfreq><b>KIfreq</b></a>
<td align=right>
<input type=checkbox name=xxiii value=IMSlem>
<td>
<a href=hyprhelp.html#IMSlem><b>IMSlem</b></a>
<td align=right>
<input type=checkbox name=xxiv value=IMSpos>
<td>
<a href=hyprhelp.html#IMSpos><b>IMSpos</b></a>
<td align=right>
<input type=checkbox name=xxv value=IMSfreq>
<td>
<a href=hyprhelp.html#IMSfreq><b>IMSfreq</b></a>

<tr><td colspan=10>
<tr><td colspan=10><b>Syntax, Semantics, Transfer, Dialogue, Glossary</b>
<tr>
<td align=right>
<input type=checkbox name=xxvi value=SIEMENSorth>
<td>
<a href=hyprhelp.html#SIEMENSorth><b>SIEMENSorth</b></a>
<td align=right>
<input type=checkbox name=xxvii value=SIEMENScats>
<td>
<a href=hyprhelp.html#SIEMENScats><b>SIEMENScats</b></a>
<td align=right>
<input type=checkbox name=xxviii value=SIHUBval>
<td>
<a href=hyprhelp.html#SIHUBval><b>SIHUBval</b></a>
<td align=right>
<input type=checkbox name=xxix value=BIgloss>
<td>
<a href=hyprhelp.html#BIgloss><b>BIgloss</b></a>

<tr>
<td align=right>
<input type=checkbox name=xxx value=IBMorth>
<td>
<a href=hyprhelp.html#IBMorth><b>IBMorth</b></a>
<td align=right>
<input type=checkbox name=xxxi value=IBMmorph>
<td>
<a href=hyprhelp.html#IBMmorph><b>IBMmorph</b></a>
<td align=right>
<input type=checkbox name=xxxii value=IBMHUBsyn>
<td>
<a href=hyprhelp.html#IBMHUBsyn><b>IBMHUBsyn</b></a>

<tr>
<td align=right>
```

```
<input type=checkbox name=xxxiii value=TUBsem>
<td>
<a href=hyprhelp.html#TUBsem><b>TUBsem</b></a>
<td align=right>
<input type=checkbox name=xxxiv value=TUEBcomp>
<td>
<a href=hyprhelp.html#TUEBcomp><b>TUEBcomp</b></a>
<td align=right>
<input type=checkbox name=xxxv value=IMSrule>
<td>
<a href=hyprhelp.html#IMSrule><b>IMSrule</b></a>
</table></center>
</form>
<hr>
<center>
<a href=hyprhelp.html#changes><b>Changes</b></a> -
<a href=demolex2.doc.html><b>Reference</b></a> -
<a href=hyprhelp.html#faq><b>FAQ</b></a> -
<a href=hyprhelp.html><b>Help doc</b></a> -
<a href=demolex6b.html><b>Concordance</b></a> -
<a href=.><b>MAIN MENU</b></a>
</center>
<hr><address>
<a href=mailto:gibbon@spectrum.uni-bielefeld.de><b>VM-HyprLex service:</b></a>
Mapped from bielefeld.lexdb.v3.3 with cfg2hl on Mar 18 1996</address>
</body>
/html>
```

5.8.2 CGI interface

The CGI interface script depends on an appropriately configured server; great care should be taken with CGI scripts to ensure that no operations can be triggered via the Web which may endanger the system, either by performing undesirable file operations, or by buggy programmes or scripts.
The script shown here was been used in the VERBMOBIL Hyperlex interface for several years.

```
#!/bin/sh
# cgi_hyprlexx
# D. Gibbon
# 10 July 1995
# 29 Dec 95

PATH=$WWWBIN/cgi-bin:$WWWBIN/bin:$GNU/bin:$PATH

export PATH

QUERY_STRING=‘cgiparse -read‘
export QUERY_STRING
eval ‘cgiparse -form‘

# Fields known as $FORM_<name>
#____________________________________________________________

# Output file:
OUTFILE=tmp.$$.html
OUT=$WWWHOME/VM-HyprLex/$OUTFILE
LOG=$WWWHOME/VM-HyprLex/wwwcgi.log
FORM_lexdb2=$WWWHOME/VM-HyprLex/Datenbanken/bielefeld.lexdb.v2.1

FORM_lexdb3=$WWWDBHOME/bielefeld.lexdb.v3.1
FORM_lexdb3a=$WWWDBHOME/bielefeld.lexdb.v3.1
FORM_lexdb3b=$WWWDBHOME/bielefeld.lexdb.v3.2
FORM_lexdb3c=$WWWDBHOME/bielefeld.lexdb.v3.3
FORM_fpwl=$WWWDBHOME/bielefeld.lexdb.v3.3.fpwl
```

```
FORM_cd=$WWWDBHOME/bielefeld.lexdb.v3.3.cd
FORM_dbsem=$WWWDBHOME/Datenbanken/dbs-0.5.6

FORM_discdb=$WWWDBHOME/bielefeld.discpartdb.v1.0
FORM_discpartdb=$WWWDBHOME/bielefeld.discpartdb.v1.0
FORM_phonsimdb=$WWWDBHOME/bielefeld.phonsimdb.v1.0

FORM_prog1=wwwdbview
FORM_prog2=wwwdbviewr
FORM_prog3=wwwdbselect
FORM_prog4=wwwdbviewx

if [ "$FORM_lexdb" = "demo" ]
   then
   FORM_lexdb=$FORM_lexdb2
   elif [ "$FORM_lexdb" = "fp" ]
        then
        FORM_lexdb=$FORM_lexdb3a
   elif [ "$FORM_lexdb" = "fp2" ]
        then
        FORM_lexdb=$FORM_lexdb3b
   elif [ "$FORM_lexdb" = "fp3" ]
        then
        FORM_lexdb=$FORM_lexdb3c
   elif [ "$FORM_lexdb" = "fpwl" ]
        then
        FORM_lexdb=$FORM_fpwl
   elif [ "$FORM_lexdb" = "cd" ]
        then
        FORM_lexdb=$FORM_cd
   elif [ "$FORM_lexdb" = "disc" ]
        then
        FORM_lexdb=$FORM_discdb
   elif [ "$FORM_lexdb" = "discpart" ]
        then
        FORM_lexdb=$FORM_discpartdb
   elif [ "$FORM_lexdb" = "dbsem" ]
        then
        FORM_lexdb=$FORM_dbsem
   elif [ "$FORM_lexdb" = "phonsim" ]
        then
        FORM_lexdb=$FORM_phonsimdb

   else FORM_lexdb=$FORM_lexdb3
fi

#___________________________________________________________
# Write HTML document

echo "<HTML>"                                       > $OUT
echo "<HEAD>"                                      >> $OUT
echo "<TITLE>HyprLex results</TITLE>"             >> $OUT
echo "</HEAD>"                                     >> $OUT
#___________________________________________________________
echo "<body>" >> $OUT

echo "<h3>VM-HyprLex results</h3>" >> $OUT
echo "<b>Server:</b> ‘env | grep SERVER_NAME | sed "s/.*=//g"‘
         (via $OUTFILE)<br>" >> $OUT
echo "<b>Date:</b> ‘date‘<br>" >> $OUT
echo "<b>Specification:</b> <em>$FORM_keytype / $FORM_atype / $FORM_qtype /
         ‘echo $FORM_lexdb | sed "s/.*\///g"‘</em><br>" >> $OUT
```

```
if [ $FORM_qtype = Marked ]
 then
ATTRIBUTES="$FORM_i $FORM_ii $FORM_iii $FORM_iv $FORM_v $FORM_vi $FORM_vii
      $FORM_viii $FORM_ix $FORM_x $FORM_xi $FORM_xii $FORM_xiii $FORM_xiv
      $FORM_xv $FORM_xvi $FORM_xvii $FORM_xviii $FORM_xix $FORM_xx $FORM_xxi
      $FORM_xxii $FORM_xxiii $FORM_xxiv $FORM_xxv $FORM_xxvi $FORM_xxvii
      $FORM_xxviii $FORM_xxix $FORM_xxx $FORM_xxxi $FORM_xxxii $FORM_xxxiii
      $FORM_xxxiv $FORM_xxxv $FORM_xxxvi"

SWITCH="$FORM_i$FORM_ii$FORM_iii$FORM_iv$FORM_v$FORM_vi$FORM_vii$FORM_viii
        $FORM_ix$FORM_x$FORM_xi$FORM_xii$FORM_xiii$FORM_xiv$FORM_xv$FORM_xvi
        $FORM_xvii$FORM_xviii$FORM_xix$FORM_xx"

 elif [ $FORM_qtype = All ]
 then
 ATTRIBUTES=$FORM_attrdef
 SWITCH=$FORM_attrdef

 elif [ $FORM_qtype = Best ]
 then
 SWITCH=$FORM_attrbest

 else echo "Unknown selection."                                 >> $OUT

fi

# Safety filter for regexp
# FORM_word="‘echo $FORM_word | sed ’s/[\.,?\*]//g’‘"


if [ $FORM_atype = Key ]
 then
 echo "<pre>"                                                   >> $OUT
 $FORM_prog4 $FORM_keytype $FORM_lexdb $ATTRIBUTES $FORM_word   >> $OUT
 echo "</pre>"                                                  >> $OUT

 elif [ $FORM_atype = SubDB ]
 then
 echo "<b>Sub-database (with all records, marked attributes only) of
       bielefeld.lexdb.v3.1</b><br>"             >> $OUT
 echo "<b>No. of data records = " ‘tail +3 $FORM_lexdb | wc -l‘ "</b><br><xmp>"
         >> $OUT
 $FORM_prog3 $FORM_lexdb $ATTRIBUTES                             >> $OUT
 echo "</xmp>"                                                   >> $OUT

elif [ $FORM_atype = Corpus-WL ]
 then
 echo "<b>Sub-database (with marked attributes) for VM CDROM
      Corpus Wordlist</b><br>" >> $OUT
 head -2 $FORM_lexdb                                              > $OUT.a
# grep " cd.=" $FORM_lexdb                                       >> $OUT.a
 FORM_word=" cd.="
 cat $FORM_lexdb |
 gawk ’
  BEGIN {crit=ARGV[1]; ARGV[1]=""}
  $0~crit {print $0} ’ $FORM_word                                >> $OUT.a
 echo "<b>No. of data records = " ‘tail +3 $OUT.a | wc -l‘ "</b><br><xmp>"
    >> $OUT
 $FORM_prog3 $OUT.a $ATTRIBUTES                                  >> $OUT
 echo "</xmp>"                                                   >> $OUT
 elif [ $FORM_atype = Demo-WL ]
 then
 echo "<b>Sub-database (with marked attriutes) for VM Demonstrator
       Wordlist</b><br>" >> $OUT
 head -2 $FORM_lexdb                                              > $OUT.a
# grep demo-wl $FORM_lexdb                                       >> $OUT.a
 FORM_word="demo-wl"
 cat $FORM_lexdb |
 gawk ’
  BEGIN {crit=ARGV[1]; ARGV[1]=""}
  $0~crit {print $0} ’ $FORM_word                                >> $OUT.a
 echo "<b>No. of data records = " ‘tail +3 $OUT.a | wc -l‘ "</b><br><xmp>"
  >> $OUT
 $FORM_prog3 $OUT.a $ATTRIBUTES                                  >> $OUT
 echo "</xmp>"                                                   >> $OUT

 elif [ $FORM_atype = Blau-WL ]
 then
 echo "<b>Sub-database (with marked attributes) for VM Blaubeuren Dialogue
    Wordlist</b><br>" >> $OUT
 head -2 $FORM_lexdb                                              > $OUT.a
# grep blau-wl $FORM_lexdb                                       >> $OUT.a
 FORM_word="blau-wl"
 cat $FORM_lexdb |
 gawk ’
  BEGIN {crit=ARGV[1]; ARGV[1]=""}
  $0~crit {print $0} ’ $FORM_word                                >> $OUT.a

 echo "<b>No. of data records = " ‘tail +3 $OUT.a | wc -l‘ "</b><br><xmp>"
   >> $OUT
 $FORM_prog3 $OUT.a $ATTRIBUTES                                  >> $OUT
 echo "</xmp>"                                                   >> $OUT

 elif [ $FORM_atype = RQH-WL ]
 then
 echo "<b>Sub-database (with marked attributes) for Reithinger/Quantz/Herweg
 Wordlist</b><br>" >> $OUT
 head -2 $FORM_lexdb                                              > $OUT.a
# grep rqh-wl $FORM_lexdb                                        >> $OUT.a
 FORM_word="rqh-wl"
 cat $FORM_lexdb |
 gawk ’
  BEGIN {crit=ARGV[1]; ARGV[1]=""}
 NR>2&& $0~crit {print $0} ’ $FORM_word                          >> $OUT.a
 echo "<b>No. of data records = " ‘tail +3 $OUT.a | wc -l‘ "</b><br><xmp>"
     >> $OUT
 $FORM_prog3 $OUT.a $ATTRIBUTES                                  >> $OUT
 echo "</xmp>"                                                   >> $OUT

 elif [ $FORM_atype = Hitlist ]
 then
 echo "<b>Sub-database (with marked attributes) for VM Top 1000 Hits
  Wordlist</b><br>" >> $OUT
 head -2 $FORM_lexdb                                              > $OUT.a
# grep hitlist $FORM_lexdb                                       >> $OUT.a
 FORM_word="hit"
 cat $FORM_lexdb |
 gawk ’
  BEGIN {crit=ARGV[1]; ARGV[1]=""}
  NR>2&&$0~crit {print $0} ’ $FORM_word >> $OUT.a
 echo "<b>No. of data records = " ‘tail +3 $OUT.a | wc -l‘ "</b><br><xmp>"
```

```
     >> $OUT
 $FORM_prog3 $OUT.a $ATTRIBUTES                                  >> $OUT
 echo "</xmp>"                                                   >> $OUT

 elif [ $FORM_atype = FP-WL ]
 then
 echo "<b>Sub-database (with marked attributes) for Research Prototype
      Wordlist</b><br>" >> $OUT
 head -2 $FORM_lexdb                                              > $OUT.a
# grep fp-wl $FORM_lexdb                                         >> $OUT.a
 FORM_word="fpwl"
 cat $FORM_lexdb |
 gawk '
  BEGIN {crit=ARGV[1]; ARGV[1]=""}
  NR>2&&$0~crit {print $0} ' $FORM_word                          >> $OUT.a
 echo "<b>No. of data records = " ‘tail +3 $OUT.a | wc -l‘ "</b><br><xmp>"
     >> $OUT
 $FORM_prog3 $OUT.a $ATTRIBUTES                                  >> $OUT
 echo "</xmp>"                                                   >> $OUT


 elif [ $FORM_atype = SemDB ]
 then
 echo "<b>Sub-database (with marked attributes) for Research Prototype
      SemDB</b><br><pre>" >> $OUT
 echo "<b>Attributes selected:</b> $ATTRIBUTES<br>"               >> $OUT
 $FORM_prog3 $FORM_dbsem $ATTRIBUTES >> $OUT
 echo "</pre>"                                                   >> $OUT

 else
 echo Unknown action.                                            >> $OUT
fi

echo "</body></html>"                                            >> $OUT

# Log file
echo $0 $FORM_atype | sed "s/\/.*\///g" >> $LOG
ls -al $OUT | gawk '{print $5,$6,$7,$8,$9}' | sed "s/\/.*\///g"  >> $LOG

# Relative location of script in web space:
echo "Location:/VM-HyprLex/$OUTFILE"
echo
```

5.8.3 Concordance routine

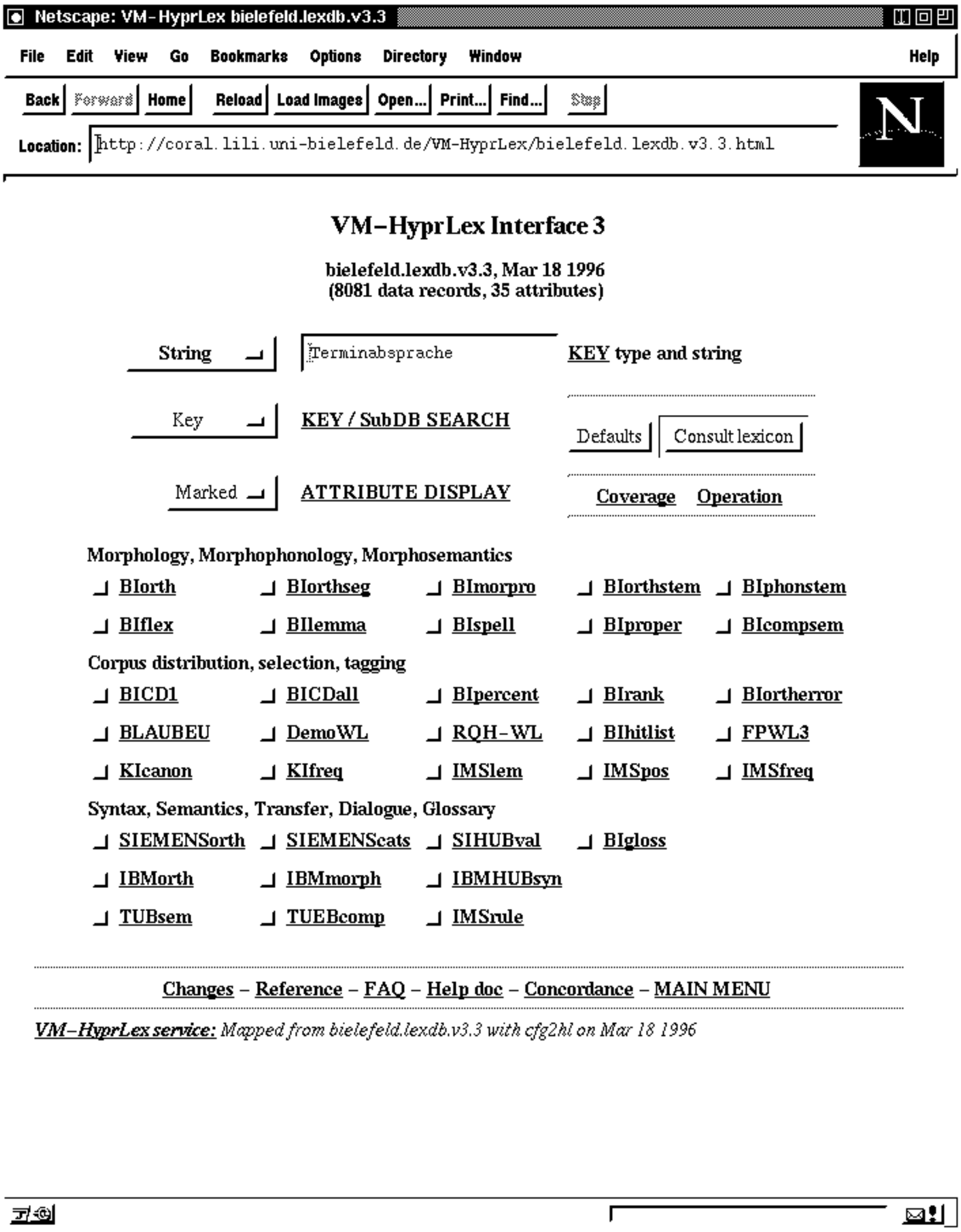
The concordance search routine script has already been listed and commented, but is repeated here for convenience.

```
#!/bin/sh
# dbviewr
# D.Gibbon, 20.11.1994
# Prettyprint of single entries
# and attributes in lexicon database,
# with equality match.

Firstline="bielefeld.lexdb.v2.1 VM-TP5.9 31 Jan 1995 DG UBI"
Attrib="BIorth BIorthseg BImorpro BIlemma BIorthstem BImorprostem BIflex BICD1 BICD2 BICD3 BICDall KICanon BIdi

if [ $# -lt 3 ]
 then
 echo "Error: Possibly no parameters were defined for the \"SELECT Marked\" option."
 exit
fi
```

```
if [ $1 = help -a -f $2 ]
 then
 echo "Internal error. Please mail bug report to administrator."
 echo $Firstline
 echo $Attrib
 exit
fi

if [ ! -f $1 ]
 then
 echo "Internal error. Please mail report bug to administrator."
 echo File $1 does not exist.
 exit
fi

#DBhead=`head -1 $1`
#DBattrib=`head -2 $1 | tail +2`

#if [ "$Firstline" != "$DBhead" -a "$Attrib" != "$DBattrib" ]
# then
# echo DB version should be \"$Firstline\".
# exit
#fi

gawk '

BEGIN {keyword = ARGV[ARGC-1]}

NR == 2 {{for (i=2 ; i < ARGC ; i++)
{for (j=1 ; j <= NF ; j++)
if (ARGV[i] == $j) {attrib[j] = "yes"; attname[j] = $j}}}
{for (i = 2 ; i < ARGC ; i++)
ARGV[i]=""}}

NR > 2 && $1 ~ keyword {print "\n<b>Entry " NR-2 " contains substring <em>" keyword "</em>: </b>"
        {for (i=1 ; i <= NF ; i++)
        if (attrib[i] ~ "yes") {print "  " attname[i] ":\t" $i}}}

 ' $* |

sed -e "s/;;/\//g" |
sed -e "s/;/&                /g" |
sed -e "s/=>/&;                 /g" |
tr ";" "\012" | tee tmp.0 |
gawk '
BEGIN {sum=0}
$1~/Entry/ {sum++}
END{print "<br><b>Number of matches =",sum,"</b>"}
    '
cat tmp.0
```

5.8.3.1 Output of concordance routine

The output of a query for all attributes to the VM-HyprLex dynamic concordance is shown in Figure 5.4.



Location: http://coral.lili.uni-bielefeld.de/VM-HyprLex/bielefeld.lexdb.v3.3.html

VM–HyprLex Interface 3

bielefeld.lexdb.v3.3, Mar 18 1996
(8081 data records, 35 attributes)

String	Terminabsprache	KEY type and string
Key	KEY / SubDB SEARCH	Defaults Consult lexicon
Marked	ATTRIBUTE DISPLAY	Coverage Operation

Morphology, Morphophonology, Morphosemantics

☐ BIorth	☐ BIorthseg	☐ BImorpro	☐ BIorthstem	☐ BIphonstem
☐ BIflex	☐ BIlemma	☐ BIspell	☐ BIproper	☐ BIcompsem

Corpus distribution, selection, tagging

☐ BICD1	☐ BICDall	☐ BIpercent	☐ BIrank	☐ BIortherror
☐ BLAUBEU	☐ DemoWL	☐ RQH–WL	☐ BIhitlist	☐ FPWL3
☐ KIcanon	☐ KIfreq	☐ IMSlem	☐ IMSpos	☐ IMSfreq

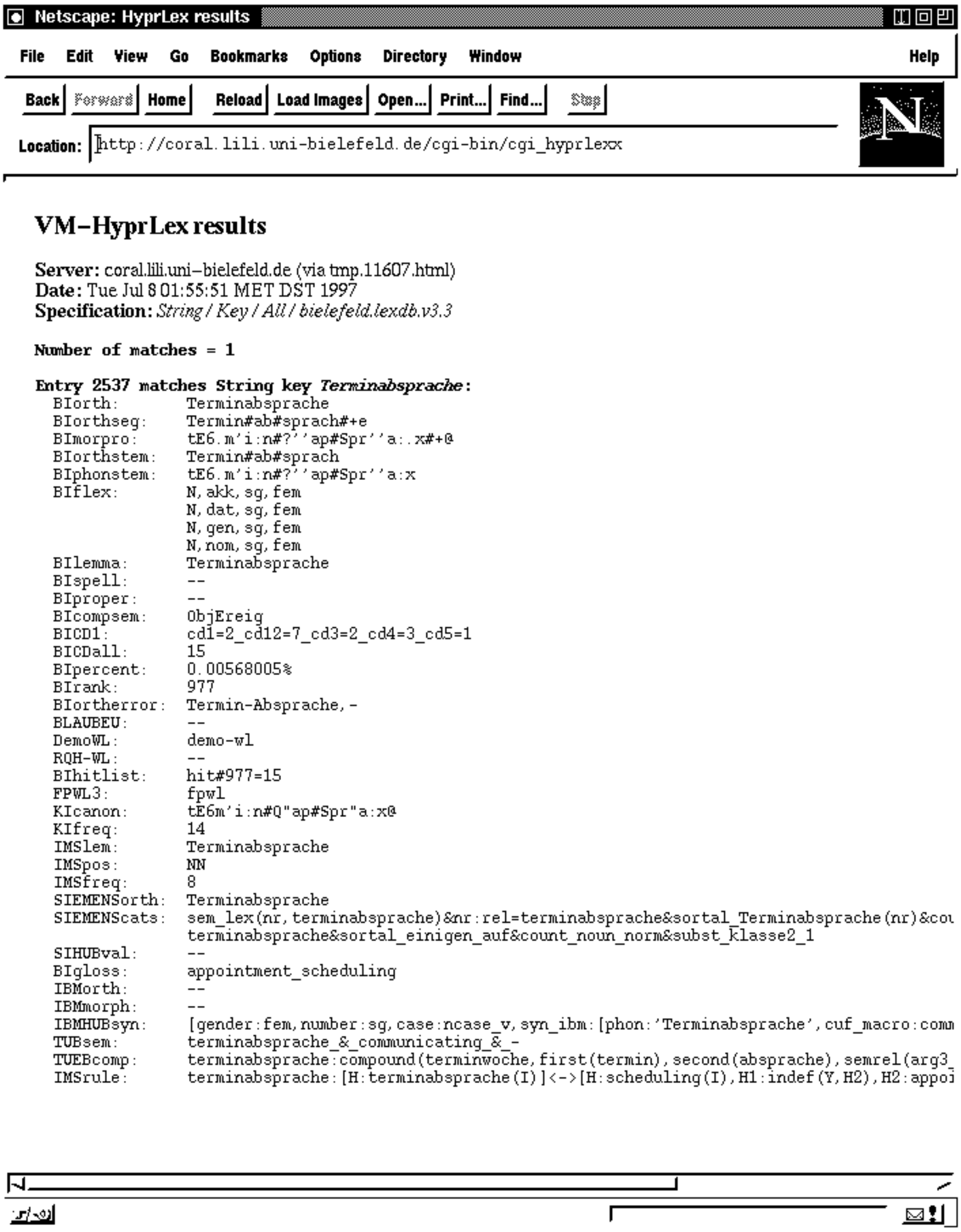
Syntax, Semantics, Transfer, Dialogue, Glossary

☐ SIEMENSorth	☐ SIEMENScats	☐ SIHUBval	☐ BIgloss
☐ IBMorth	☐ IBMmorph	☐ IBMHUBsyn	
☐ TUBsem	☐ TUEBcomp	☐ IMSrule	

Changes – Reference – FAQ – Help doc – Concordance – MAIN MENU

VM–HyprLex service: *Mapped from bielefeld.lexdb.v3.3 with cfg2hl on Mar 18 1996*

Figure 5.3:



VM–HyprLex results

Server: coral.lili.uni–bielefeld.de (via tmp.11607.html)
Date: Tue Jul 8 01:55:51 MET DST 1997
Specification: *String / Key / All / bielefeld.lexdb.v3.3*

```
Number of matches = 1

Entry 2537 matches String key Terminabsprache:
  BIorth:        Terminabsprache
  BIorthseg:     Termin#ab#sprach#+e
  BImorpro:      tE6.m'i:n#?''ap#Spr''a:.x#+@
  BIorthstem:    Termin#ab#sprach
  BIphonstem:    tE6.m'i:n#?''ap#Spr''a:x
  BIflex:        N,akk,sg,fem
                 N,dat,sg,fem
                 N,gen,sg,fem
                 N,nom,sg,fem
  BIlemma:       Terminabsprache
  BIspell:       --
  BIproper:      --
  BIcompsem:     ObjEreig
  BICD1:         cd1=2_cd12=7_cd3=2_cd4=3_cd5=1
  BICDall:       15
  BIpercent:     0.00568005%
  BIrank:        977
  BIortherror:   Termin-Absprache,-
  BLAUBEU:       --
  DemoWL:        demo-wl
  RQH-WL:        --
  BIhitlist:     hit#977=15
  FPWL3:         fpwl
  KIcanon:       tE6m'i:n#Q"ap#Spr"a:x@
  KIfreq:        14
  IMSlem:        Terminabsprache
  IMSpos:        NN
  IMSfreq:       8
  SIEMENSorth:   Terminabsprache
  SIEMENScats:   sem_lex(nr,terminabsprache)&nr:rel=terminabsprache&sortal_Terminabsprache(nr)&cou
                 terminabsprache&sortal_einigen_auf&count_noun_norm&subst_klasse2_1
  SIHUBval:      --
  BIgloss:       appointment_scheduling
  IBMorth:       --
  IBMmorph:      --
  IBMHUBsyn:     [gender:fem,number:sg,case:ncase_v,syn_ibm:[phon:'Terminabsprache',cuf_macro:comm
  TUBsem:        terminabsprache_&_communicating_&_-
  TUEBcomp:      terminabsprache:compound(terminwoche,first(termin),second(absprache),semrel(arg3_
  IMSrule:       terminabsprache:[H:terminabsprache(I)]<->[H:scheduling(I),H1:indef(Y,H2),H2:appoi
```

Figure 5.4: Result of dynamic concordance consultation.

Bibliographical references

References

A. Aho, B. Kernighan and P. Weinberger (1987). *The AWK programming language.* Addison-Wesley Publishing Company, Reading, Mass., etc.

A. Akmajian (1984). *Linguistics: An introduction to language and communication.* The MIT Press, Cambridge, Massachusetts, 2nd edition.

G. Allen (1988). The PHONASCII system. *Journal of the International Phonetic Association* 18(1): 9–25.

J. Allen, M. Hunnicutt and D. Klatt (1987). *From text to speech: The MITalk system.* Cambridge University Press, Cambridge.

F. Althoff, G. Drexel, H. Lüngen, M. Pampel and C. Schillo (1996). The treatment of compounds in a morphological component for speech recognition. In: D. Gibbon, ed., *Natural language processing and speech technology. Results of the 3rd KONVENS Conference, Bielefeld, October 1996*, pp. 71–76. Mouton de Gruyter, Berlin, New York.

F. Andry, S. McGlashan, N. Youd, N. Fraser and S. Thornton (1992). Making DATR work for speech: Lexicon compilation in SUNDIAL. *Computational Linguistics* 18(3): 245–267.

V. Aubergé (1992). Developing a structured lexicon for synthesis of prosody. In: G. Bailly, C. Benoît and T. Sawallis, eds., *Talking machines: Theories, models and designs*, pp. 307–321. North-Holland, Amsterdam.

D. Autesserre, G. Pérennou and M. Rossi (1989). Methodology for the transcription and labeling of a speech corpus. *Journal of the International Phonetic Association* 19(1): 2–15.

A. Averbuch, L. Bahl and R. Bakis (1987). Experiments with the TANGORA 20000 word speech recognizer. In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, pp. 701–704.

A. Averbuch, L. Bahl, R. Bakis, P. Brown, A. Cole, G. Daggett, S. Das, K. Davies, S. De Gennaro, P. De Souza, E. Epstein, D. Fraleigh, F. Jelinek, S. Katz, B. Lewis, R. Mercer, A. Nadas, D. Nahamoo, M. Picheny, G. Shichman and P. Spinelli (1986). An IBM PC-based large-vocabulary isolated-utterance speech recognizer. In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, pp. 53–56.

H. Baayen (1991). De CELEX lexicale databank. *Forum der Letteren* 32(3): 221–231.

G. Bailly (1994). Rule compilers and text-to-speech systems. *Les Cahiers de l'ICP* 3: 87–91.

G. Bailly and C. Benoît, eds. (1992). *Talking machines: Theories, models and designs.* North-Holland, Elsevier Science Publishers, Amsterdam.

J. Baker (1975a). The DRAGON system – An overview. *IEEE Transactions on Acoustics, Speech and Signal Processing, ASSP-23* pp. 24–29.

J. Baker (1975b). Stochastic modeling for automatic speech understanding. In: D. Reddy, ed., *Speech recognition*, pp. 521–541. Academic Press, New York, N.Y. Also in: A. Waibel, K.-F. Lee, eds. (1990), Readings in speech recognition, Morgan Kaufmann Publishers, San Mateo, California, 297–307.

J. Baker (1989). Dragondictate-30k: Natural language speech recognition with 30000 words. In: *Proceedings of the European Conference on Speech Technology*, volume 2, pp. 161–163.

J. Baker, P. Bamberg, K. Bishop, L. Gillick, V. Helman, Z. Huang, Y. Ito, S. Lowe, B. Peskin, R. Roth and F. Scattone (1992). Large vocabulary recognition of Wall Street Journal sentences at Dragon systems. In: *Speech and Natural Language Workshop*, pp. 387–392, Harriman, New York, 23–26 February.

M. Ball (1991). Computer coding of the IPA: Extensions to the IPA. *Journal of the International Phonetic Association* 21(1): 36–41.

D. Bleiching (1992a). Prosodisches Wissen im Lexikon. In: G. Görz, ed., *KONVENS 92, 1. Konferenz "Verarbeitung natürlicher Sprache", Nürnberg, 7.–9. Oktober 1992*, pp. 59–68. Springer-Verlag, Berlin.

D. Bleiching (1992b). Prosodisches wissen im lexikon. In: G. Görz, ed., *KONVENS 92*, pp. 59–68. Springer–Verlag, Berlin.

D. Bleiching (1994). Integration von morphologie und prosodie in ein hierarchisches lexikon. In: H. Trost, ed., *KONVENS 94*, pp. 32–41. Springer–Verlag, Berlin.

D. Bleiching, G. Drexel and D. Gibbon (1996a). Ein synkretismusmodell für die deutsche morphologie. In: D. Gibbon, ed., *Natural language processing and speech technology. Results of the 3rd KONVENS Conference, Bielefeld, October 1996*, pp. 237–248. Mouton de Gruyter, Berlin, New York.

D. Bleiching, G. Drexel and D. Gibbon (1996b). Ein synkretismusmodell für die deutsche morphologie. In: D. Gibbon, ed., *Natural Language PRocessing and Speech Technology: Results of the 3rd KONVENS Conference, Bielefeld 1996*, pp. 237–248. Mouton de Gruyter, Berlin.

D. Bleiching and D. Gibbon (1994). Handbuch zur Demonstrator-Wortliste. V1.1. May 1994, Bielefeld University, Bielefeld, Germany.

D. Bobrow and T. Winograd (1977). An overview of KRL, a knowledge representation language. *Cognitive Science* 1: 3–46.

B. Boguraev, J. Carroll, S. Pulman, G. Russell, G. Ritchie, A. Black, E. Briscoe and C. Grover (1988). The lexical component of a natural language toolkit. In: D. Walker, A. Zampolli and N. Calzolari, eds., *Automating the lexicon: Research and practice in a multilingual environment.* Cambridge University Press, Cambridge.

R. Brachman and H. Levesque (1985). *Readings in knowledge representation.* Morgan Kaufmann Publishers, Inc., Los Altos, California.

A. Brietzmann, H. Hein, H. Niemann and P. Regel (1983). The Erlangen system for understanding continuous German speech. In: *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, pp. 304–307, Boston.

C. Browman (1980). Rules for demisyllable synthesis using Lingua, a language interpreter. In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, pp. 561–564, Denver.

G. Bruce (1989). Report from the IPA Working Group on suprasegmental categories. *Working Papers 35, Lund University, Department of Linguistics, Lund* pp. 25–40.

H. Bunt, R.-J. Beun, F. Dols, J. von der Linden and G. thoe Schwartzenberg (1985). The TENDUM dialogue system and its theoretical basis. *IPO Annual Progress Report* 19: 105–113.

L. Cahill (1993a). Morphonology in the lexicon. In: *Sixth Conference ofthe European Chapter of the Association for Computational Linguistics, Utrecht*, pp. 87–96.

L. Cahill (1993b). Morphonology in the lexicon. In: *Proceedings of the Sixth Conference of the European Chapter of the Association for Computational Linguistics*, pp. 87–96, Utrecht.

L. Cahill and R. Evans (1990). An application of DATR: The TIC lexicon. In: R. Evans and G. Gazdar, eds., *The DATR Papers*, pp. 31–39. School of Cognitive and Computing Science, University of Sussex, Brighton, 2nd edition.

N. Carbonell and J. Pierrel (1986). Architecture and knowledge sources in a human computer oral dialog system. In: *Proceedings of the NATO workshop: Structure of multimodal dialogues including voice*, Corsica, France.

J. Carson-Berndsen (1993a). Time map phonology and the projection problem in spoken language recognition. Doctoral dissertation, University of Bielefeld, Bielefeld, Germany.

J. Carson-Berndsen (1993b). *Time Map Phonology and the Projection Problem in Spoken Language Recognition.* Ph.D. thesis, U Bielefeld.

E. Charniak and D. McDermott (1985). *Introduction to Artificial Intelligence.* Addison-Wesley, Reading, Massachusetts.

N. Chomsky (1965). *Aspects of the Theory of Syntax.* MIT Press, Cambridge, Mass.

N. Chomsky and M. Halle (1968). *The sound pattern of English.* Harper and Row, New York, Evanston, London.

K. Church (1987a). Phonological parsing and lexical retrieval. *Cognition* 25: 53–69.

K. Church (1987b). *Phonological parsing in speech recognition.* Kluwer Academic Publishers, Boston, Dordrecht, Lancaster.

A. Content, P. Mousty and M. Radeau (1990). Brulex, une base de données lexicales informatise pour le français écrit et parlé. *L'Annèe Psychologique* 90: 551–566.

G. G. Corbett and N. M. Fraser (1995). Network morphology: a datr account of russian nominal inflection. *Journal of Linguistics* 29.

M. Cresswell (1973). *Logics and Languages.* Methuen, London.

D. Cruse (1986). *Lexical semantics.* CUP, Cambridge.

D. Crystal (1985). *A dictionary of linguistics and phonetics.* Basil Blackwell, Oxford, UK.

W. Daelemans (1987). *Studies in Language Technology: An Object–Oriented Compter Model ofMorphophonological Aspects of Dutch.* Ph.D. thesis, U Leuven.

R. De Mori, M. Gilloux, G. Mercier, M. Simon, C. Tarrides and J. Vaissiere (1984). Integration of acoustic, phonetic, prosodic and lexical knowledge in an expert system for speech understanding. In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP.* 42.9.1–42.9.4.

J.-M. Dolmazon, J.-C. Caërou and W. Barry (1990). Initial development of SAM standard workstation. SAM–UCL–022, December, Appendix Se.10, University College London, London.

D. Dougherty (1990). *sed & awk.* O'Reilly & Associates Inc., Sebastopol, CA.

G. Dreckschmidt (1987). The linguistic component in the speech understanding system SPICOS. In: H. Tillmann and G. Willée, eds., *Analyse und Synthese gesprochener Sprache, Jahrestagung der Gesellschaft für Linguistische Datenverarbeitung, Bonn*, pp. 96–101. Olms, Hildesheim.

L. Erman (1977). A functional description of the HEARSAY-II speech understanding system. In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, Hartford.

L. Erman and F. Hayes-Roth (1981). The HEARSAY-II speech understanding system: Integrating knowledge to resolve uncertainty. In: B. Webber and N. Nilsson, eds., *Readings in Artificial Intelligence*, pp. 349–389. Tioga, Palo Alto, CA.

L. Erman and V. Lesser (1980). The HEARSAY-II speech understanding system: A tutorial. In: W. Lea, ed., *Trends in speech recognition*, pp. 361–381. Prentice Hall, Englewood Cliffs, NJ. Also in: A. Waibel and K.-F. Lee, eds. (1990), Readings in speech recognition, Morgan Kaufmann Publishers, San Mateo, California, 235–245.

J. Esling (1988). 7.1 Computer coding of IPA symbols and 7.3 detailed phonetic representation of computer data bases. *Journal of the International Phonetic Association* 18(2): 99–106.

J. Esling (1990). Computer coding of the IPA: Supplementary report. *Journal of the International Phonetic Association* 20(1): 22–26.

R. Evans and G. Gazdar (1989). The DATR papers. Research Report: May 1989, School of Cognitive and Computing Science, University of Sussex, School of Cognitive and Computing Science, University of Sussex, Brighton.

R. Evans and G. Gazdar (1990). The DATR papers. Research Report: February 1990, School of Cognitive and Computing Science, University of Sussex, School of Cognitive and Computing Science, University of Sussex, Brighton.

R. Evans and G. Gazdar (1996). Datr: A language for lexical knowledge representation. *Computational Linguistics* 22:2.

I. Ferrané, M. De Calmès, D. Cotto, J.-M. Pécatte and G. Pérennou (1992). Statistiques lexicales sur le corpus de textes utilisés dans le projet BREF: Questions de couverture lexicale. In: *Proceedings Communication Homme–Machine, Séminaire LEXIQUE*, pp. 217–226, 21–22 January 1992, IRIT-UPS, Toulouse.

D. Flickinger (1987). *Lexical Rules in the Hierarchical Lexicon.* Ph.D. thesis, Stanford University.

P. Geutner (1995). *Using morphology towards better large-vocabulary speech recognition systems.* Interactive Systems Laboratories, University of Karlsruhe, Karlsruhe, Germany.

D. Gibbon (1990). Prosodic association by template inheritance. In: W. Daelemans and G. Gazdar, eds., *Proceedings of the Workshop on Inheritance in natural Language Processing*, pp. 65–81. Institute for Language Technology, Tilburg.

D. Gibbon (1991). Lexical signs and lexicon structure: Phonology and prosody in the ASL-lexicon. Research Report ASL–MEMO–20–91/UBI, University of Bielefeld, Bielefeld, Germany.

D. Gibbon (1992a). ILEX: A linguistic approach to computational lexica. In: U. Klenk, ed., *Computatio linguae. Aufsätze zur algorithmischen und quantitativen Analyse der Sprache*, pp. 32–51. Franz Steiner Verlag, Stuttgart.

D. Gibbon (1992b). Ilex: A linguistic approach to computational lexicology. In: U. Klenk, ed., *Computatio Linguae, Beihefte zur Zeitschrift für Dialektologie und Linguistik*, pp. 32–53. Steiner, Stuttgart.

D. Gibbon (1993). Generalized DATR for flexible lexical access: PROLOG specification. VERBMOBIL Report 2, October 1993, Bielefeld University, Bielefeld, Germany.

D. Gibbon (1995). The VERBMOBIL lexicon: Bielefeld lexicon database V2.1. VERBMOBIL Technisches Dokument 21, 31 January 1995, Bielefeld University, Bielefeld, Germany.

D. Gibbon and U. Ehrlich (1995). Spezifikationen für ein VERBMOBIL-Lexikondatenbankkonzept. VERBMOBIL Memo 69, Bielefeld University & Daimler Benz AG, Bielefeld, Ulm.

D. Gibbon, S.-C. Tseng and K. Folikpo (????). Prosodic inheritance and phonetic interpretation: lexical tone. to appear.

J. Goldsmith (1990). *Autosegmental and metrical phonology.* Indiana University Linguistics Club, Bloomington, Indiana.

L. Goorfin (1989). Electronic dictionary pronounces over 83,000 words. *Speech Technology* 4(4): 49–51.

S. Hertz, J. Kadin and K. Karplus (1985). The DELTA rule development system for speech synthesis from text. In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, pp. 1589–1601.

G. Heyer, K. Waldhur and H. Khatchadourian (1991). Motivation, goals and milestones of ESPRIT II MULTILEX. In: *Génie Linguistique 91*, volume 1, Versailles, France, 16–17 January.

H. Höge, E. Marschall, O. Schmidbauer and R. Sommer (1985). Worthypothesengenerierung im Projekt SPICOS. In: H. Niemann, ed., *Mustererkennung 85, 7. DAGM-Symposium Erlangen, Informatik-Fachberichte, vol. 107*, pp. 175–179. Springer-Verlag, Berlin.

W. Jassem and P. Łobacz (1989). IPA phonemic transcription using an IBM PC and compatibles. *Journal of the International Phonetic Association* 19(1): 16–23.

F. Jelinek (1985). A real-time, isolated-word, speech recognition system for dictation transcription. In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, pp. 858–861.

L. Karttunen (1983). KIMMO: A general morphological processor. *Texas Linguistic Forum* 22: 165–186.

R. Kasper and W. Rounds (1986). A logical semantics for feature structures. In: *Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics*, Morristown. ACL.

K. Kirchhoff (1996). Phonologisch strukturierte hmms zur automatischen spracherkennung. In: D. Gibbon, ed., *Natural language processing and speech technology. Results of the 3rd KONVENS Conference, Bielefeld, October 1996*, pp. 55–63. Mouton de Gruyter, Berlin, New York.

D. Klatt (1982). The KLATTalk text-to-speech conversion system. In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, pp. 1589–1592.

D. Klatt (1987). Review of text-to-speech conversion in English. *Journal of the Acoustical Society of America* 82: 737–793.

D. Knuth (1973). *The art of computer programming 3: Sorting and searching.* Addison-Wesley, Reading, Massachusetts.

A. Kornai (1991). Formal phonology. Doctoral dissertation, Stanford University, Stanford.

K. Koskenniemi (1983a). *Two–level Morphology.* Ph.D. thesis, U Helsinki.

K. Koskenniemi (1983b). *Two-level morphology: A general computational model for word-form recognition and production.* University of Helsinki, Department of General Linguistics, Helsinki, Finland.

R. Lacouture and Y. Normandin (1993). Efficient lexical access strategies. In: *Proceedings of the European Conference on Speech Technology.*

H. Langer and D. Gibbon (1992). DATR as a graph representation language for ILEX speech oriented lexica. Research Report, March 1992, ASL–TR–43–92/UBI, University of Bielefeld, Bielefeld, Germany.

K.-F. Lee, H.-W. Hon and R. Reddy (1990). An overview of the SHPINX speech recognition system. In: A. Waibel and K.-F. Lee, eds., *Readings in speech recognition*, pp. 600–610. Morgan Kaufmann Publishers, San Mateo, California.

V. Lesser, R. Fennell, L. Erman and D. Reddy (1975). Organization of the HEARSAY-II speech understanding system. In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP-23*, pp. 11–23.

J. Llisterri and J. Mariño (1993). Spanish adaptation of SAMPA and automatic phonetic transcription. In: ESPRIT Project 6819 (SAM-A), ed., *Speech technology assessment in multilingual applications, Year 1, 1 April 1993–30 September 1993*, pp. 1–9. London. SAM-A periodic progress report, Document No: SAM-A/UPC/001/V1.

J. Lyons (1977). *Semantics. Volumes I and II.* Cambridge University Press, Cambridge.

H. Ney, D. Mergel, A. Noll and A. Paeseler (1988). Overview of speech recognition in the SPICOS system. In: H. Niemann, M. Lang and G. Sagerer, eds., *Recent advances in speech understanding and dialog systems*, volume 46 of *NATO ASI Series F*, pp. 305–310. Springer-Verlag, Berlin.

H. Niemann, A. Brietzmann, R. Mühlfeld, P. Regel and G. Schukat (1985). The speech understanding and dialog system EVAR. In: R. De Mori and C. Suen, eds., *New systems and architectures for automatic speech recognition and synthesis, NATO ASI Series F, vol. 16*, pp. 271–302. Springer-Verlag, Berlin.

H. Niemann, E. Nöth, M. Mast and E. Schukat-Talamazzini (1992). Ein Lexikon für ein natürlich-sprachliches Dialogsystem. In: *Beiträge des ASL-Lexikonworkshops*, pp. 15–18, Wandlitz, 26–27 November. ASL–TR–40–92/ZSB.

M. Nossin (1991). Le projet GENELEX: EUREKA pour les dictionnaires génériques. *Génie Linguistique 91,* volume 1. Versailles, France, 16–17 January 1991.

G. Pérennou, D. Cotto, M. De Calmès, I. Ferrané, J. Pécatte and J. Tihoni (1991). Composantes phonologique et orthographique de BDLEX. In: *Deuxièmes Journées Nationales du GRECO-PRC Communication Homme–Machine*, pp. 351–362, Toulouse, 29–30 January.

G. Pérennou, D. Cotto, M. De Calmès, I. Ferrané and J.-M. Pécatte (1992). Le projet BDLEX de base de données lexicales du Français écrit et parlé. In: *Proceedings Communication Homme-Machine, Séminaire LEXIQUE*, pp. 153–171, 21–22 January 1992, IRIT-UPS Toulouse.

G. Pérennou and M. De Calmès (1987). BDLEX lexical data and knowledge base of spoken and written French. In: *European Conference on Speech Technology*, volume 1, pp. 393–396, Edinburgh.

G. Pérennou and J. Tihoni (1992). Lexique et phonologie en reconnaissance de la parole. In: *Proceedings Communication Homme–Machine, Séminaire LEXIQUE*, pp. 41–57, 21–22 January 1992, IRIT-UPS Toulouse.

M. Plenat (1991). Vers d'une phonémisation des sigles. In: *Deuxièmes journées du GDR-PRC Communication Homme–Machine, EC2 Editeur*, pp. 363–371, Toulouse, 29–30 January.

C. Pollard and I. Sag (1987). *Information–based Syntax and Semantics, Volume I.* CSLI, Stanford.

C. Pollard and I. Sag (1994). *Head–Driven Phrase Structure Grammar.* U Chicago Press, Chicago.

J. Pustejovsky (1996). *The Generative Lexicon.* MIT Press, Cambridge.

A. Radford (1988). *Transformational grammar: A first course.* CUP, Cambridge.

M. Rayner, H. Alshawi, I. Breton, D. Carter, V. Digalakis, B. Gamback, J. Kaja, J. Karlgren, B. Lyberg, S. Pulman, P. Price and C. Samuelsson (1993). A speech to speech translation system built from standard components. In: *Proceedings of a Workshop: Human Language Technology*, pp. 217–222, Princeton, NJ, 21–24 March.

S. Reinhard and D. Gibbon (1981). Prosodic association and template inheritance. In: *Fifth Conference of the European Chapter of the Association for Computational Linguistics, Berlin*, pp. 131–136.

G. Ritchie, A. Black, G. Russell and S. Pulman (1992). *Computational morphology.* The MIT Press, Cambridge, Massachusetts and London.

E. Rosch (1978). Principles of categorization. In: E. Rosch and B. Lloyd, eds., *Cognition and Categorization*, pp. 27–48. Erlbaum Associates, Hillsdale, N.J.

A. Rudnicky, L. Baumeister, K. De Graff and E. Lehmann (1987). The lexical access component of the CMU continuous speech recognition system. In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP.*

G. Ruske (1985). Demisyllables as processing units for automatic speech recognition and lexical access. In: R. De Mori and C. Suen, eds., *New systems and architectures for automatic speech recognition and synthesis*, volume 16 of *NATO ASI Series F*, pp. 593–611. Springer-Verlag, Berlin.

G. Ruske and T. Schotola (1981). The efficiency of demisyllable segmentation in the recognition of spoken words. In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, pp. 971–974, Atlanta.

G. Sagerer (1990). *Automatisches Verstehen gesprochener Sprache*, volume 74 of *Reihe Informatik*. Bibliographisches Institut, Mannheim.

C. Schillo (1996). *A DATR implementation in C: ZDATR Manual Version 1.0.* Technical Report, U Bielefeld.

S. Schröder, G. Sagerer and H. Niemann (1987). Wissensakquisition mit semantischen Netzwerken. In: E. Paulus, ed., *Mustererkennung 87, 9. DAGM-Symposium Braunschweig, Informatik-Fachberichte*, pp. 305–309. Springer-Verlag, Berlin.

E. Schukat-Talamazzini (1993). Automatische Spracherkennung. Habilitationsschrift, Erlangen University, Erlangen, Germany.

P. Sells (1985). *Lectures on contemporary syntactic theories: An introduction to Government-Binding theory, Generalized Phrase Structure Grammar, and Lexical-Functional Grammar.* CSLI Center for the Study of Language and Information, Stanford, California.

S. Shieber (1986). *An Introduction to Unification–based Approaches to Grammar.* CSLI, Stanford.

G. Thurmair (1986). Linguistische Analyse im Projekt SPICOS. *Kleinheubacher Berichte* 29.

J.-P. Tubach and L.-J. Bok (1985). *ZUT – Petit dictionnaire français.* Institut de Phonitique de Grenoble, avec le concours du CNRS (GRECO Comm. Parlie), Grenoble.

B. Van Coile (1989). The DEPES development system for text-to-speech synthesis. In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, pp. 250–253.

J. Van Hemert, U. Adriaens-Porzig and L. Adriaens (1987). Speech synthesis in the SPICOS-project. In: H. Tillmann and G. Willée, eds., *Analyse und Synthese gesprochener Sprache: Vorträge im Rahmen der Jahrestagung 1987 der Gesellschaft für Linguistische Datenverarbeitung e.V., Bonn, 4–6 March*, pp. 34–39. Olms, Hildesheim.

A. Waibel (1988). Prosody and speech recognition. Research notes in artificial intelligence, Pitman Publishing, London.

A. Waibel and K.-F. Lee, eds. (1990). *Readings in speech recognition.* Morgan Kaufmann Publishers, San Mateo, California.

L. Wall and R. Schwartz (1991). *Programming perl.* O'Reilly & Associates Inc., Sebastopol, CA.

J. Wells (1987). Computer-coded phonetic transcription. *Journal of the International Phonetic Association* 17(2): 94–114.

J. Wells (1989a). Computer-coded phonemic notation of individual languages of the European Community. *Journal of the International Phonetic Association* 19(1): 31–54.

J. Wells (1993a). Applying SAM-PA to Spanish, Portuguese, and Greek: A preliminary discussion document. In: ESPRIT Project 6819 (SAM-A), ed., *Speech technology assessment in multilingual applications.* London. Document No: SAM-A/D1-Appendix B, SAM-A periodic progress report, Year 1, 1 April 1993–30 September 1993.

J. Wells (1993b). An update on SAMPA. In: ESPRIT Project 6819 (SAM-A), ed., *Speech technology assessment in multilingual applications*, pp. 1–6. London. Document No: SAM-A/D1-Appendix A, SAM-A periodic progress report, Year 1, 1 April 1993–30 September 1993.

J. C. Wells (1989b). Computer–coded phonemicnotation of individual languages of the european community. *Journal of the IPA* 19:1.

W. Woods and V. Zue (1976). Dictionary expansion via phonological rules for a speech understanding system. In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, pp. 561–564, Philadelphia.

M. Woszczyna, N. Coccaro, A. Eisele, A. Lavie, A. McNair, T. Polzin, I. Rogina, C. Rose, T. Sloboda, M. Tomita, J. Tsutsumi, N. Waibel and W. Ward (1993). Recent advances in Janus: A speech translation system. In: *Proceedings of a Workshop: Human Language Technology*, pp. 211–216, 21–24 March, Princeton, NJ.

S. Young, A. Hauptmann, W. Ward, E. Smith and P. Werner (1989). High level knowledge sources in usable speech recognition systems. *Communications of the ACM* 32(2): 183–194. Also in: A. Waibel and K.-F. Lee, eds., (1990), Readings in speech recognition, Morgan Kaufmann Publishers, San Mateo, California, 538–549.

A. Zwicky (1993). Heads, bases, and functors. In: G. G. Corbett, N. M. Fraser and S. McGlashan, eds., *Heads in Grammatical Theory*, pp. 292–315. Cambridge University Press, Cambridge.