Content-Based Retrieval in Database Semantics
A Theoretical Foundation for Practical NLP

Roland Hausser

Abteilung Computerlinguistik
Universität Erlangen-Nürnberg (CLUE)
Bismarckstr. 6, 91054 Erlangen, Germany
email: rrh@linguistik.uni-erlangen.de
phone: +49 (0) 911 55 68 79
fax: +49 (0) 9131 852 9251

Abstract Database Semantics (DBS) approaches practical (commercial) applications of natural language processing by solving the most important theoretical question first: *How does the mechanism of natural language communication work?* A model of natural language communication requires completeness (i) of function, (ii) of data coverage in different languages, and (iii) computational efficiency for processing in real time. This paper shows how the practical retrieval from online texts may benefit from realizing the theoretical goals of DBS.

Key words:
cycle of natural language communication; time-linear algorithm; non-recursive feature structures; content-addressable database

DBS models the human prototype computationally as a talking robot. In this sense, DBS is *agent-oriented* – in contradistinction to the phrase structure analysis of generative grammar, the truth-conditions of Montague grammar, the markup and statistical tagging of corpora, and other systems of past and current linguistic analysis and natural language processing, which are all *sign-oriented*.

An agent-oriented approach to natural language models the cycle of communication by designing computational agents with bodies in the real world. The agents have interfaces for recognition and action, an internal memory, an algorithm for reading content in and out, etc. The agents communicate with each other by using unanalyzed external language surfaces, and by switching between the speak, think, and hear modes.

A sign-oriented approach, in contrast, tries to get by without the design of a cognitive agent. There is no distinction between the agent-external real world and an agent-internal cognition, there are no interfaces for recognition and action, there is no individual agent-internal memory, etc. Without differentiating between the speak and hear mode, sign-oriented generative grammars generate all language expressions either from the same "start symbol", e.g., S (top down generation) or combine categorized word forms into the same "result symbol", e.g., t for truth (bottom up amalgamation).[1] Corpus linguistics is also sign-oriented in that it applies only to recorded language data.

[1] For a more detailed discussion see FoCL'99, Sect. 10.1.

1 Four Levels of Abstraction for Representing Language Data

The empirical base of an agent-oriented approach is much broader than that of a sign-oriented approach. Consequently, attempts to extend a sign-oriented into an agent-oriented approach have not been successful.[2] The agent-oriented approach of DBS, in contrast, can simply use its hear mode for any sign-oriented applications.

One of these is the analysis of online text for purposes of retrieval. In the electronic representation of written text the following levels of abstraction may be distinguished:

1.1 The four levels of abstraction for representing text

- Level-one: *Representation as bitmap*
  Pages are scanned into the computer as bitmaps. This preserves the appearance of the page (which may be important, as in a medieval manuscript), but does not allow any letter-based text processing.
- Level-two: *Digital representation*
  The bitmap representation is transferred automatically into a digital representation (e.g., ASCII or Unicode) by means of an OCR software or the letters are written online to begin with. The result allows text processing, such as automatic search based on letter sequences, simultaneous substitution, and the movement of paragraphs.
- Level-three: *Representation with metadata markup*
  The digital letter sequences are enriched with a markup, for example in XML (preferably in stand-off), which characterizes chapter and/or section headings, the paragraph structure, name and address of the author, bibliography, etc., depending on the kind of text, e.g., newspaper article, novel, play, or dictionary.[3] As a result, the text may be printed in different styles while maintaining the encoded text structure. Furthermore, the markup may be extended to a semantic characterization of content, for example the text's domain, thus supporting retrieval.
- Level-four: *Representation as content*
  The content is derived automatically from the text's letter sequence by means of a rule-based syntactic-semantic parser. The resulting output depends on the underlying linguistic theory.

In the DBS hear mode, natural language surfaces are coded as level-four content in a format suitable for efficient storage in and retrieval from a content-addressable database.

2 Tagging in Natural Language Processing

Written text differs from other online data such as photographs, videos, spoken language, or music in that it consists primarily of letters. For the computer, all level-two

[2] An early attempt to turn the truth-conditional approach of Montague Grammar into an agent-oriented system was SCG'84. Proposals to enrich sign-oriented systems with "performative clauses" or "constraints," attached to the analyzed signs and supposed to serve as the context of use, are not sufficient for building a talking robot (cf. Hausser 2011, Sect. 10).

[3] Metadata markup originated as the cataloging work of the library/information sciences and the manuscript annotation in print shops.

letter sequences are created equal such that English learns, for example, is treated no different from the inverted letter sequence, i.e., snreal. The amazing retrieval power of search engines like Google or Yahoo is based on matching letter sequences in a text regardless of whether they happen to represent a frequent or an infrequent word form, a neologism, an acronym, an expression of a foreign language, or simply nonsense.

For the same reason, level-two letter sequences do not provide any grammatical information. For example, swimming and swam are not counted as forms of the same word, i.e., swim. Furthermore, the grammatical distinctions between noun, verbs, and adjectives, singular and plural, the syntactic and verbal moods, the tenses, etc., are not provided. Without them, grammatical analysis is not possible.

In natural language processing (NLP), the first attempt at introducing grammatical distinctions was TAGGIT by Francis 1980, a pattern-based system of categorization which required a lot of post-editing. Building from there,[4] Garside, Leech, and Sampson 1987 developed the CLAWS1-system. It tries to induce the categorization from the statistical distribution of word forms in texts. This *tagging* was developed in part for getting better and quicker retrieval results from large corpora than letter-based level-two pattern matching alone.

Tagging is based on a manual level-three markup of a small part of a corpus, called the *core corpus*. The categories used for the classification are called *tags* or *labels*. Their total is called the *tagset*. After hand-tagging the core corpus, the probabilities of the transitions from one word form to the next are computed, usually by means of *Hidden Markov Models* (HMMs).[5] Then the transition probabilities of the hand-tagged core corpus are transferred to the whole corpus using a simplified tagset. These steps may be summarized as follows:

2.1 MANUAL LEVEL-THREE MARKUP IN STATISTICAL TAGGING

1. manual markup of a core corpus
2. extending the result of step 1 to the whole corpus using statistical methods (HMM)
3. manual post-editing of the step 2 markup extension

Once the whole corpus has been tagged, frequency counts may be based on tagged word forms rather than letter sequences.

Unfortunately, this approach is caught on the horns of a dilemma, namely (i) low accuracy and (ii) a difficulty to correct. The dilemma has been concealed by misleading claims of excellent accuracy. For example, Leech 1995 asserts an error rate of 1.7% for the CLAWS 4 tagger of the BNC. This corresponds to a recognition rate of 98.3%, which seems to be very good at first glance. It is important to realize, however, that these numbers apply to the word form *tokens* and not to the *types* (as in a lexicon).

As shown in FoCL'99, 15.5.2, the most frequent nine word forms of the BNC, namely the, of, and, to, a, in, is, that, and was, amount to 0.001368% of the types, but cover

[4] Cf. Marshall 1987, pp. 43-45.

[5] The use of HMMs for the grammatical tagging of corpora is described in, e.g., Leech, Garside and Atwell 1983, Marshall 1983, DeRose 1988, Sharman 1990, Brown, Della Pietra, et al. 1991. See also Church and Mercer 1993.

21.895% of the word form tokens. At the other extreme are the word forms which occur only once (hapaxlegomena). Even though they comprise 52.807% of the types in the BNC, they cover only 0.388% of the tokens. In other words, not recognizing more than half of the word form types in the BNC would fit 4.38 times into the 1.7% error rate asserted by Leech 1995. This distressing lack of accuracy[6] cannot be taken lightly because the correct and complete analysis of the word forms is the foundation for all further work, for example, syntactic analysis.

The other horn of the dilemma is that specific errors cannot be corrected by improving the tagger. At a certain point, the only way to improve the error rate[7] is manual post-editing. Today, the actual results of the 1994 CLAWS 4 BNC tagging (Burnard et al. 1995) are buried under 12 years of massive manual post-editing (BNC XML edition, Burnard et al. 2007).

In summary, the apparent attraction of statistical tagging for NLP, namely working automatically for any amount of data and for any language, fails on two counts: (i) the hand-tagging of the core corpus and (ii) the manual post-editing of the results. The training, supervising, and remunerating the large numbers of personnel required for these tasks is cumbersome and costly. Worse, the method is limited to *fixed theoretical data* such as the BNC; it is unfit for the use on *expanding practical data* because it could never keep up with today's fast moving world.

3 Word Form Recognition in Computational Linguistics

The attraction of statistical tagging would be understandable if no better option were available. In fact, however, there is the level-four alternative of *automatic word form recognition*, based on traditional dictionaries and rule-based morphological analysis. It takes an unanalyzed surface, e.g., a letter sequence like learns, as input and provides the computer with the information needed for syntactic-semantic processing. Words not in the online dictionary or word forms with an inflectional/agglutinational, derivational, or compositional structure not handled by the morphological rules produce an error. Errors may be found systematically by parsing word form lists derived from corpora.

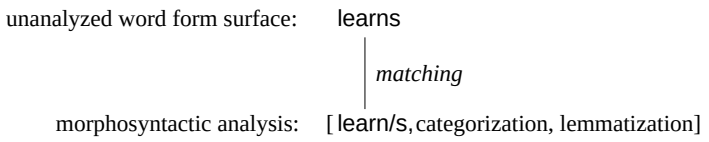
Systems of automatic word form recognition must provide (i) *categorization* and (ii) *lemmatization*. Categorization specifies the grammatical properties, which in the case of learns would be something like "verb, third person singular, present tense." Lemmatization specifies the base form, here learn, which is used to look up the meaning common to all the word forms of the paradigm, i.e., learn, learns, learned, and learning.

[6] It is also at the root of a 20 year stagnation in speech recognition (CLaTR'11, Sect. 2.4). After all, low frequency doesn't make expressions like megazostrodon (paleontology), free solo (rock climbing), cold opening (tv series), or Costa Rica dome (oceanography) less relevant. On the contrary, the lower the frequency of a word or a phrase relative to everyday language, the more relevant it is semantically (Zipf 1949) and the more valuable it is for a practically single-handed characterization of the domain.

[7] The error rate of the CLAWS 4 BNC tagging has been analyzed in detail in FoCL'99, Sect. 15.5. As far as we know, this analysis has never been challenged. On the contrary, it is indirectly confirmed by Sampson's 1999 sullen complaint that " the sole purpose [of the BNC tagging critique in FoCL'99] seems to be to argue that statistical word-tagging algorithms are not 100 per cent successful."

The recognition algorithm in its most primitive form consists of matching the surface of the unknown letter sequence with the corresponding surface (key) in a full-form lexicon,[8] thus providing access to the relevant lexical description:

3.1 MATCHING AN UNANALYZED SURFACE ONTO A KEY



Automatic word form recognition in DBS is based on the allomorph method, which is capable of recognizing neologisms. Its algorithm consists of (i) segmenting the letter sequence of a surface into known but unanalyzed parts, called allomorphs, (ii) lexical lookup of the corresponding analyzed allomorphs in a trie structure,[9] and (iii) their composition into well-formed analyzed word forms (cf. FoCL'99, Chap. 14). This requires (i) an online lexicon for base forms (morphemes), e.g., wolf, (ii) allo-rules for deriving variants (allomorphs) of the base form, e.g., wolf and wolv-, before runtime, and (iii) combi-rules for combining the analyzed allomorphs during runtime, e.g., wolv/es, and providing categorization and lemmatization for the complete word form.

Building such a system for any given natural language is not particularly difficult, even for writing systems based on characters, e.g., Chinese and Japanese, rather than letters. Given (i) a traditional dictionary of the natural language of choice, (ii) a suitable off-the-shelf software framework, and (iii) a properly trained computational linguist, an initial system can be completed in less than six months. It will provide accurate, highly detailed lexical analyses of about 90% of the word form types in a corpus.

Increasing the recognition rate of the word form types to approximately 100% is merely a matter of additional work. It consists of adding missing entries to the online lexicon, and improving the rules for allomorphy and for inflection/agglutination, derivation, and composition. In contrast to the post-editing of a tagged corpus, the improvements of an automatic word form recognition system apply to the natural language in question as a whole, and not just to an isolated, fixed corpus.

4 Using Different Core Values in the Same Proplet Shell

Automatic word form recognition raises the theoretical question of how the properties of a word form should be formally represented. For modeling the cycle of natural language communication in DBS, this format should be suitable for the hear mode, the think mode, and the speak mode of a cognitive agent (talking robot). Furthermore, the format should be suitable for modeling *reference* as a pattern matching between the agent-internal levels of language and the context of use.

[8] Full-form lookup is one of the three basic methods of automatic word form recognition, the others being the morpheme method and the allomorph method. For further explanation see FoCL'99, Chaps. 13–15.

[9] See Fredkin 1960; Knuth 1998, pp. 495–512; FoCL'99, 14.3.3.

From a computational point of view, this question of format amounts to the definition of a *data structure* suitable for an algorithm reading language content into (hear mode) and out of (speak mode) an agent-internal database, the processing of content in the database (think mode) and the pattern matching between language content and context content (reference). After several decades of development, DBS has settled on the format of *proplets*, defined as flat (non-recursive) feature structures[10] with a finite set of ordered attributes and double-ended queues as values.

Proplets turn out to be versatile in that they maintain their format and their formal properties in a multitude of different functions. The most basic distinction is between proplet shells and constant proplets. Proplet shells are a kind of pattern proplet; they use variables as values or as attributes, in contradistinction to constant proplets which may not contain any variables. Constant proplets arise as lexical or as connected content proplets; they differ in that lexical proplets have empty continuation (fnc, arg, mdr, mdd, pc, nc) and book-keeping (prn) attributes, while connected content proplets do not. Content proplets arise as language or as context proplets; they differ in that context proplets have empty sur(face) attributes, while language proplets do not.

As an example, consider the following relation between a proplet shell and several related lexical proplets. As shown by their empty sur(face) attribute, these content proplets happen to be context rather than language proplets.:

4.1 PROPLET SHELL MATCHING DIFFERENT LEXICAL CONTEXT PROPLETS

proplet shell		*context proplets*				
sur:		sur:	sur:	sur:	sur:	sur:
noun: α		noun: dog	noun: book	noun: child	noun: apple	
cat: pn		cat: pn	cat: pn	cat: pn	cat: pn	
sem: count pl		sem: count pl	sem: count pl	sem: count pl	sem: count pl	
fnc:	$\Rightarrow$	fnc:	fnc:	fnc:	fnc:	
mdr:		mdr:	mdr:	mdr:	mdr:	
nc:		nc:	nc:	nc:	nc:	
pc:		pc:	pc:	pc:	pc:	
prn:		prn:	prn:	prn:	prn:	

DBS uses the *core* attributes noun, verb, and adj for the parts of speech; the *language* attributes sur for surface, cat for category, and sem for semantics; the *continuation* attributes fnc for functor, arg for argument, mdr for modifier, mdd for modified, nc for next conjunct, and pc for previous conjunct; and the *book-keeping* attribute prn for proposition number. The proplets in 4.1 differ only in the value of the noun attribute.

Context proplets may be turned into language proplets by inserting the appropriate sur values, as in the following example for English:

[10] A precursor of feature structures in linguistics is a list of binary values without attributes, called a "feature bundle", e.g., $\begin{bmatrix}\text{+vocalic}\\ \text{+high}\end{bmatrix}$, used by Chomsky and Halle 1968 for purposes of morphophonology. The introduction of feature structures with attribute-value pairs is credited to Minsky 1975. Called "frames" by Minsky, their origin may be traced back to the legendary Dartmouth workshop of 1956, attended also by McCarthy, Newell, Rochester, Shannon, and Simon.

4.2 PROPLET SHELL MATCHING DIFFERENT LEXICAL LANGUAGE PROPLETS

proplet shell		*language proplets*				
sur: α'+x		sur: dog+s	sur: book+s	sur: child+ren	sur: apple+s
noun: α		noun: dog	noun: book	noun: child	noun: apple
cat: pn		cat: pn	cat: pn	cat: pn	cat: pn
sem: count pl		sem: count pl	sem: count pl	sem: count pl	sem: count pl
fnc:	$\Rightarrow$	fnc:	fnc:	fnc:	fnc:
mdr:		mdr:	mdr:	mdr:	mdr:
nc:		nc:	nc:	nc:	nc:
pc:		pc:	pc:	pc:	pc:
prn:		prn:	prn:	prn:	prn:

Assuming that the context proplets in 4.1 have been already acquired, learning the associated language proplets in 4.2 involves only a single value, namely that of the sur attribute, which facilitates learning.

Once the proplets have been acquired for one language, they may be reused for another, provided the lexicalization (Pustejovsky 1995) is similar. The following example shows proplets for the concept *dog* with English, French, German, and Italian surfaces:

4.3 TAKING SUR VALUES FROM DIFFERENT LANGUAGES

proplet shell		*language proplets*				
sur: α'		sur: dog	sur: chien	sur: Hund	sur: cane
noun: α		noun: dog	noun: dog	noun: dog	noun: dog
cat: sn		cat: sn	cat: sn	cat: sn	cat: sn
sem: count sg		sem: count sg	sem: count sg	sem: count sg	sem: count sg
fnc:	$\Rightarrow$	fnc:	fnc:	fnc:	fnc:
mdr:		mdr:	mdr:	mdr:	mdr:
nc:		nc:	nc:	nc:	nc:
pc:		pc:	pc:	pc:	pc:
prn:		prn:	prn:	prn:	prn:

For syntactic-semantic parsing, the French, German, and Italian proplet versions will have to be complemented with the additional cat value m (for the grammatical gender masculine). This language-dependent information may be obtained from the traditional dictionaries for these languages. In addition, corpus-based information, such as domain-dependent frequency, LA-hear predecessors and successors ordered according to frequency (n-grams), semantic relations, etc., may be added (CLaTR'11, Sect. 8.5).

5 Using the Same Core Value in Different Proplet Shells

The previous section has shown an orthogonal relation between proplet shells and core values in the sense that a given proplet shell may take different core values. Let us turn now to a second orthogonal relation between proplet shells and core values, namely the embedding of a given core value into different proplet shells. The latter is a simple but effective method to enhance the expressive power of the lexicon of a natural language without having to acquire additional core values. For example, the core value *book* may be used as a noun, a verb, or an adj:

5.1 EXAMPLES USING *book* IN DIFFERENT PARTS OF SPEECH

Mary loves a good *book* (noun).
Mary *book*ed (verb) a flight to Paris.
Mary is a rather *book*ish (adj) girl.

The lexical *book* proplets used in these contents are defined as follows:

5.2 CORE VALUE *book* IN LEXICAL NOUN, VERB, AND ADJ PROPLETS

$$book \Longrightarrow \begin{bmatrix} \text{sur: book} \\ \text{noun: book} \\ \text{cat: sn} \\ \text{sem: count sg} \\ \text{fnc:} \\ \text{mdr:} \\ \text{nc:} \\ \text{pc:} \\ \text{prn:} \end{bmatrix} \begin{bmatrix} \text{sur: booked} \\ \text{verb: book} \\ \text{cat: n' a' v} \\ \text{sem: past} \\ \text{arg:} \\ \text{mdr:} \\ \text{nc:} \\ \text{pc:} \\ \text{prn:} \end{bmatrix} \begin{bmatrix} \text{sur: bookish} \\ \text{adj: book} \\ \text{cat: adn} \\ \text{sem: psv} \\ \text{mdd:} \\ \text{nc:} \\ \text{pc:} \\ \text{prn:} \end{bmatrix}$$

Similar examples are *red* and *square*, which may also be used as the core values of a noun, a verb, and an adj, as in the following contents:

5.3 EXAMPLES USING *red* AND *square* IN DIFFERENT PARTS OF SPEECH

Mary preferred the other *red* (noun).
The rising sun *red*dened (verb) the sky.
Mary drank *red* (adj) wine.

Mary's house faces a *square* (noun).
Mary *square*d (verb) her account.
Mary bought a *square* (adj) table.

The lexical methods of using (i) different core (4.1) and surface (4.2, 4.3) values in the same proplet shell and (ii) the same core value in different proplet shells (5.2, 5.3) are complemented by (iii) the compositional method of syntax and semantics, resulting in an enormous increase in expressive power. For example, embedding the core values *book*, *square*, and *red* into V(erb), N(oun), and A(djective) proplet shells allows the formal construction of the following expressions in DBS:

5.4 CORE VALUES IN SYNTACTIC-SEMANTIC COMPOSITION

book_V the red_A square_N
book_V the square_N red_A
book_V the square_A red_N
square_V the red_A book_N
square_V the book_N red_A
square_V the book_A red_N
redden_V the square_A book_N

redden$_{\mathrm{V}}$ the book$_{\mathrm{N}}$ square$_{\mathrm{A}}$
redden$_{\mathrm{V}}$ the book$_{\mathrm{N}}$ square$_{\mathrm{N}}$
etc.

The examples are grammatically well-formed and their core values[11] have straightforward procedural implementations as the recognition patterns of a talking robot.

The dadaistic absurdity of the *literal meanings*$_1$ of the expressions in 5.4 highlights the cognitive mechanism of the compositional semantics in natural language. It demonstrates the need to distinguish the literal meaning$_1$ of language expressions from the speaker's *meaning*$_2$ of utterances. The latter, defined in Pop-1 (FoCL'99, 4.3.3) as the use of the former relative to a context of interpretation, seem mostly to fail for 5.4. Which of the meanings$_1$ can be used literally or non-literally, or not be used sensibly at all, depends on the context of interpretation, for humans and talking robots alike.

In summary, linguistic examples as isolated signs do not have any concrete context of interpretation. Therefore, evaluating the utterance meaning$_2$ of the examples 5.4 amounts methodologically to evaluating how easily they can be supplied with a virtual context of interpretation. Especially for non-literal uses, the result depends on the imagination of the agent doing the evaluation.

6 Representing Level-Four Content

The proplets presented in Sects. 4 and 5 are lexical proplets because their continuation attributes fnc, arg, mdr, mdd, nc, and pc as well as their bookkeeping attribute prn have no values yet. In order to represent a propositional content, the lexical proplets of DBS have to be connected[12] by means of compositional semantic relations. In natural language, these are *functor-argument* and *coordination*, intra- and extrapropositionally. As an example, consider a content consisting of an intrapropositional functor-argument, represented as a set of linked proplets:

6.1 INTRAPROPOSITIONAL FUNCTOR-ARGUMENT: Julia knows John.

noun: Julia	verb: know	noun: John
fnc: know	arg: Julia John	fnc: know
prn: 625	prn: 625	prn: 625

These proplets are simplified to essentials. For example, the distinction between language and context proplets is omitted (no sur attribute). They are recognizable as content rather than lexical proplets, however, because their continuation attributes fnc and arg as well as their book-keeping attribute prn have non-empty values (in contradistinction to the lexical proplets in 4.1 – 4.3 and 5.2). The three proplets are part of the

[11] The definition of basic concepts (core values) as elementary recognition and action procedures constitutes a fundamental difference between the agent-oriented approach of DBS and the sign-oriented approach of truth-conditional semantics (CLaTR'11, Sect. 12.4).

[12] This is in contradistinction to statistically-based tagging, in which the lexical analysis of a word form and its syntactic-semantic role in a sentence are inextricably fused. As a result, the number of CLAWS 4 tagged word forms in the BNC is 37.5% greater than the number of unanalyzed surfaces (FoCL'99, Sect. 15.5.), constituting an opaque form of lexical ambiguity.

same proposition because they are held together by a common prn value (here 625).[13] The functor-argument relation is coded in terms of attribute values, serving as addresses (CLaTR'11, Sect. 4.4). For example, the *Julia* and *John* proplets specify their functor as know, while the *know* proplet specifies Julia and John as its arguments.[14]

A content like 6.1 may be turned into a schema by replacing each occurrence of a constant with a variable (simultaneous substitution):

6.2 TURNING 6.1 INTO A SCHEMA

$$\begin{bmatrix}\text{noun: } \alpha\\ \text{fnc: } \beta\\ \text{prn: K}\end{bmatrix}\ \begin{bmatrix}\text{verb: } \beta\\ \text{arg: } \alpha\ \gamma\\ \text{prn: K}\end{bmatrix}\ \begin{bmatrix}\text{noun: } \gamma\\ \text{fnc: } \beta\\ \text{prn: K}\end{bmatrix}$$

The schema 6.2 defines the same semantic relations between *pattern proplets* as does the content 6.1 between *content proplets*. A schema matches the content from which it has been derived as well as an open number of similar contents. A DBS schema is not just a *l'art pour l'art* linguistic generalization, but allows using detailed syntactic and semantic properties for efficient high-resolution retrieval. The matching between a schema and a content is illustrated below using the schema 6.2 and the content 6.1:

6.3 PATTERN MATCHING BETWEEN SCHEMA 6.2 AND CONTENT 6.1

$$\begin{array}{lccc}
\textit{schema level} & \begin{bmatrix}\text{noun: } \alpha\\ \text{fnc: } \beta\\ \text{prn: K}\end{bmatrix} & \begin{bmatrix}\text{verb: } \beta\\ \text{arg: } \alpha\ \gamma\\ \text{prn: K}\end{bmatrix} & \begin{bmatrix}\text{noun: } \gamma\\ \text{fnc: } \beta\\ \text{prn: K}\end{bmatrix}\\
\multicolumn{4}{c}{\textit{internal matching}}\\
\textit{content level} & \begin{bmatrix}\text{noun: Julia}\\ \text{fnc: know}\\ \text{prn: 625}\end{bmatrix} & \begin{bmatrix}\text{verb: know}\\ \text{arg: Julia John}\\ \text{prn: 625}\end{bmatrix} & \begin{bmatrix}\text{noun: John}\\ \text{fnc: know}\\ \text{prn: 625}\end{bmatrix}
\end{array}$$

The matching between the schema and the content is successful because (i) the pattern proplets have the same[15] attributes in the same order as the corresponding content proplets and (ii) the variables used at the schema level match the constants at the content level.

The coding method illustrated in 6.1–6.3 with an example of intrapropositional functor-argument works equally well for extrapropositional coordination or any other construction of natural language:

[13] In DBS, the "quantifiers" of Symbolic Logic are absent. Their binding function is taken by the prn values and the use of addresses, while the determiner functions of ∃x and ∀x are coded as values of the cat and sem attributes of nominal proplets (CLaTR'11, Sect. 11.5).

[14] When we refer to a proplet by its core value, we use italics, e.g., *John*, whereas for reference to an attribute or a value within a proplet, we use helvetica, e.g., fnc or know.

[15] As defined in NLC'06, 3.2.3, it is sufficient for successful matching if the attributes of the pattern proplet are a *subset* of the attributes of the content proplet.

6.4 EXTRAPROPOSITIONAL COORDINATION: Julia sang. Sue slept. John read.

noun: Julia fnc: sing prn: 10	verb: sing arg: Julia nc: (sleep 11) pc: prn: 10	noun: Sue fnc: sleep prn: 11	verb: sleep arg: Sue nc: (read 12) pc: (sing 10) prn: 11	noun: John fnc: read prn: 12	verb: read arg: John nc: pc: (sleep 11) prn: 12		

The propositions with the prn values 10, 11, and 12 are concatenated by the pc (for previous conjunct) and nc (for next conjunct) values of the respective verbs. For example, the nc value of the second proplet *sing* is (sleep 11), while the pc value of the fourth proplet *sleep* is (sing 10).

The proplets in the extrapropositional coordination 6.4 may also be turned into a schema by replacing the constants with variables:

6.5 TURNING 6.4 INTO A SCHEMA

noun: α fnc: β prn: K	verb: β arg: α nc: (δ K+1) pc: prn: K	noun: γ fnc: δ prn: K+1	verb: δ arg: γ nc: (ψ K+2) pc: (β K) prn: K+1	noun: ϕ fnc: ψ prn: K+2	verb: ψ arg: ϕ nc: pc: (δ K+1) prn: K+2		

The schema matches the content 6.4 from which it was derived as well as an open number of similar contents:

6.6 PATTERN MATCHING BETWEEN SCHEMA 6.5 AND CONTENT 6.4

schema level	noun: α fnc: β prn: K	verb: β arg: α nc: (δ K+1) pc: prn: K	noun: γ fnc: δ prn: K+1	verb: δ arg: γ nc: (ψ K+2) pc: (β K) prn: K+1	noun: ϕ fnc: ψ prn: K+2	verb: ψ arg: ϕ nc: pc: (δ K+1) prn: K+2		
internal matching								
content level	noun: Julia fnc: sing prn: 10	verb: sing arg: Julia nc: (sleep 11) pc: prn: 10	noun: Sue fnc: sleep prn: 11	verb: sleep arg: Sue nc: (read 12) pc: (sing 10) prn: 11	noun: John fnc: read prn: 12	verb: read arg: John nc: pc: (sleep 11) prn: 12		

The computational simplicity and efficiency of the DBS matching procedure depends crucially on the definition of proplets as non-recursive feature structures, with the order of attributes fixed within proplets.[16]

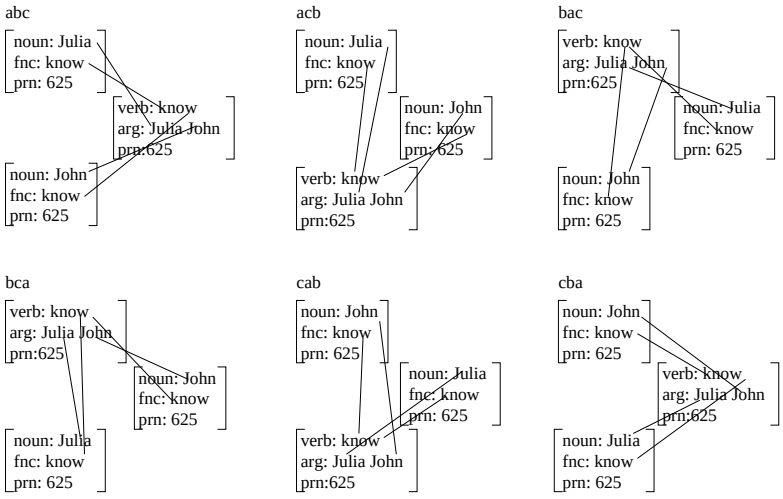
The proplet set of a content is *order-free*, however, in the sense that the storage location of the proplets does not affect the semantic relations defined between them.[17] This is because the semantic relations between proplets are coded by *address* rather than by

[16] For a systematic analysis of functor-argument and coordination relations at the elementary, the phrasal, and the clausal level see Hausser 2009.

[17] This is in contradistinction to a logical formula such as $\exists x[man(x) \wedge walk(x)]$, a phrase structure tree, or a recursive feature structure with unification as shown in NLC'06, 3.4.5, which change

embedding (as in recursive feature structures). For example, the three proplets in 6.1 may be represented in the order abc, acb, bac, bca, cab, and cba, and yet maintain their semantic relations intact, as shown below:

6.7 MAINTAINING SEMANTIC RELATIONS REGARDLESS OF ORDER



The six possible proplet orders are arranged vertically. The bidirectional semantic relations within each triple are indicated by lines, which are like rubber bands adjusting to the varying arrangements of the proplets. For a better drawing of these lines, the proplet in the middle is moved one step to the right.

In summary, a proplet is defined as a *list* of features (internal order), whereby a feature is defined as an attribute-value pair (avp). The proplets representing a complex content, in contrast, are a *set* (no external order), which is essential for the storage of content in a database. Compared to the recursive feature structures used in nativism (with unordered attributes, but an order of embedding[18]), proplets have the following advantages:

6.8 ADVANTAGES OF PROPLETS

1. Flat ordered feature structures are easier to read and computationally more efficient than recursive feature structures with unordered attributes.
2. Flat ordered feature structures provide for easy schema derivation and for easy pattern matching.

---

their meaning or lose their well-formedness if the order of their parts is changed. For the same reason, they fail to provide a natural primary key for storage and retrieval in a database (cf. Hausser 2007).

[18] The reason why nativist approaches such as GB, LFG, GPSG, and HPSG use recursive feature structures is the dominance relation in their constituent structure trees, which is modeled in the corresponding feature structures as an embedding. The empirical deficiency of constituent structures as defined in context-free phrase structure grammar is known since Bar Hillel 1953. Cf. FoCL'99, Sects. 8.4, 8.5.

3. The combination of a proplet's core and prn value provides a natural primary key for storage in and retrieval from memory.
4. Coding the semantic relations between proplets as addresses makes proplets order-free and therefore amenable to the needs of one's database.
5. The semantic relations between proplets enable a time-linear navigation along those relations, reintroducing order and serving as the selective activation of content, as needed in language production and inferencing.

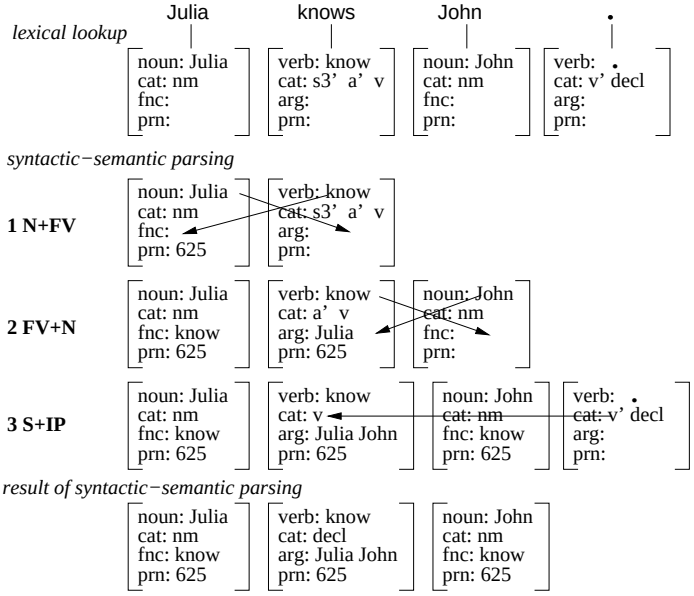
The data structure of proplets has been used in the LA-hear, LA-think, and LA-speak grammars defined in FoCL'99 and NLC'06. They map time-linear sequences of natural language surfaces into sets of proplets (hear mode), and sets of proplets into corresponding time-linear sequences of natural language surfaces (speak mode).

7 Hear, Think, and Speak Modes

The algorithm of DBS is Left-Associative Grammar. LA-grammar always combines the current "sentence start" with the current "next word" into a "new sentence start." This amounts to interpreting a sequence like a+b+c+d... left-associatively as ...(((a b) c) d)... (cf. Hopcroft and Ullman 1977, p. 47). DBS uses the left-associative derivation order to model the time-linear structure of natural language.

In preparation of the formal rules of LA-hear, LA-think, and LA-speak illustrated in the following Sect. 8, let us show the cycle of natural language communication in a user-friendly conceptual format. The format represents the basic units as proplets, but indicates the derivational operations graphically, i.e., by means of arrows. We begin with a time-linear surface compositional hear mode derivation:

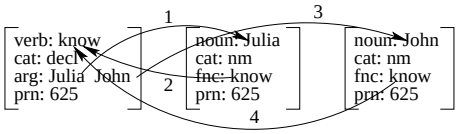
7.1 DBS HEAR MODE DERIVATION OF Julia knows John.



The analysis is surface compositional in that each surface is analyzed as a lexical proplet. The derivation is time-linear, as shown by the stair-like addition of one lexical proplet in each new line. Each line represents a derivation step, based on the application of the specified LA-hear grammar rule, e.g., **1 N+FV** (defined in 8.1). The rules establish grammatical relations by copying values, as indicated by the diagonal arrows. The result of the derivation is the order-free set of proplets 6.1, ready to be stored in the agent's content-addressable memory (9.1).

Based on the grammatical relations between the proplets stored in the agent's memory, the second step in the cycle of natural language communication is a *selective activation* of content by navigating from one proplet to the next. The following example is based on the content 6.1, derived in 7.1:

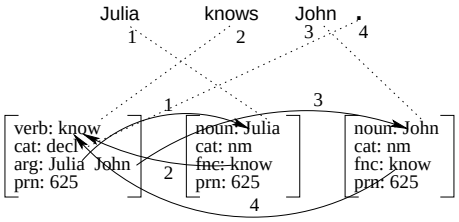
7.2 DBS THINK MODE NAVIGATION



The navigation is driven by an LA-think grammar which uses the grammatical relations between proplets like a railroad system. By constructing proplet addresses from the arg, fnc, and prn values, the navigation proceeds from the verb to the subject noun (arrow 1), back to the verb (arrow 2), to the object noun (arrow 3), and back to the verb (arrow 4).

Such a think mode navigation provides the *what to say* for language production from stored content, while the third step in the cycle of communication, i.e., the speak mode, provides the *how to say it* (McKeown 1985) in the natural language of choice. Consider the following example of a speak mode derivation, resulting in a surface realization:

7.3 DBS SPEAK MODE REALIZATION



The derivation is based on the same navigation as 7.2, whereby the surfaces are realized from the *goal proplet* of each navigation step, using mainly the core value. In NLC'06, the DBS cycle of communication has been worked out in detail for more than 100 constructions of English.

8 Algorithm of LA-Grammar

Having shown informally how the data structure of proplets may be used (i) for mapping surfaces into level-four content (hear mode, 7.1), (ii) for activating level-four content

selectively (think mode, 7.2), and (iii) for mapping activated level-four content into surfaces (speak mode, 7.3) let us turn to the formal rules performing these procedures. Given that the three phases of the cycle of natural language communication are each time-linear, i.e., linear like time and in the direction of time, they may all three be handled by the same algorithm, namely time-linear LA-grammar.[19]

The LA-grammar rules of DBS use pattern proplets at the rule level and content proplets at the input level (cf. 6.3, 6.6). By matching the pattern proplets of the sentence start and of the next word with the content proplets of the input, the variables of patterns are bound to constants. The bound variables are used by the rule's output pattern to derive the output content.

Consider, for example, the definition of **N+FV** (for noun plus finite verb). The rule is shown as it applies to a matching language input, corresponding to the first derivation step of 7.1 (explanations in italics):

8.1 LA-HEAR RULE APPLICATION

i. *rule name*	ii. *rule package*
N+FV	{FV+N}

iii. *ss-pattern*	iv. *nw-pattern*	v. *resulting ss′-pattern*

rule level

$$\begin{bmatrix}\text{noun: } \alpha\\ \text{cat: NP}\\ \text{fnc:}\end{bmatrix}\quad\begin{bmatrix}\text{verb: } \beta\\ \text{cat: NP}'\text{ X VT}\\ \text{arg:}\end{bmatrix}\Rightarrow\quad\begin{bmatrix}\text{noun: } \alpha\\ \text{cat: NP}\\ \text{fnc: } \beta\end{bmatrix}\quad\begin{bmatrix}\text{verb: } \beta\\ \text{cat: X VT}\\ \text{arg: } \alpha\end{bmatrix}$$

matching and binding of variables ⇑ *output* ⇓

language level

$$\begin{bmatrix}\text{noun: Julia}\\ \text{cat: nm}\\ \text{fnc:}\\ \text{prn: 1}\end{bmatrix}\quad\begin{bmatrix}\text{verb: know}\\ \text{cat: s3' a' v}\\ \text{arg:}\\ \text{prn:}\end{bmatrix}\qquad\begin{bmatrix}\text{noun: Julia}\\ \text{cat: nm}\\ \text{fnc: know}\\ \text{prn: 1}\end{bmatrix}\quad\begin{bmatrix}\text{verb: know}\\ \text{cat: a' v}\\ \text{arg: Julia}\\ \text{prn:1}\end{bmatrix}$$

An LA-hear grammar rule consists of (i) a rule name, (ii) a rule package, (iii) a pattern for an *ss* (sentence start), (iv) a pattern for an *nw* (next word), and (v) a pattern for the *ss′* (resulting/next sentence start). A pattern at the rule level matches a content at the language level if (a) the attributes of the pattern are a subset of the attributes of the content and (b) the values of the pattern are compatible with the values of the content.

Vertical binding of the variables in the input patterns (rule level) to the corresponding constants in the language input (language level) enables computing an output. The new content is built by replacing the variables in the output patterns with the constants matching the input part of the rule. The operation is performed at the language level.

For example, by binding the variable α of the input pattern to the corresponding constant *Julia* (language level), the [arg: α] feature of the output pattern (rule level) provides the value Julia to the arg attribute of the output at the language level. If a

[19] LA-grammar provides the first and so far the only complexity hierarchy which is orthogonal to the Chomsky hierarchy of Phrase Structure Grammar (TCS'92, Handl 2011). This is empirically relevant insofar as the lowest class of C1-LAGs (linear complexity) includes many context sensitive languages and handles all the structures found in the natural languages (cf. FoCL'99, 12.5.7, CoNSyx hypothesis).

current rule application is successful, the resulting sentence start is provided with the proplet of a next word, if available, by automatic word form recognition. This results in a new input pair to which the rules of the current rule package are applied.

A variant of the method illustrated in 8.1 is used by LA-think. An LA-think rule takes a current proplet as input and computes a next proplet as output (successor). As an example, consider the LA-think rule **VNs** (from verb to noun). It is shown as it executes navigation step 1 from the first to the second proplet in 7.2:

8.2 LA-THINK RULE APPLICATION

	i. *rule name* **VNs**	ii. *rule package* {NVs}
	iii. *current proplet*	iv. *next proplet*
rule level	[verb: β arg: X α Y prn: K] $\Rightarrow$	[noun: α fnc: β prn: K]
matching and binding of variables	$\Uparrow$	$\Downarrow$
Word Bank level	[verb: know cat: decl arg: Julia John prn: 625]	[noun: Julia cat: nm fnc: know prn: 625]

By using the same variables, α, β, and K, in the patterns for the current and the next proplet, and by binding them to the values know, Julia, and 625 of the input proplet *know*, the pattern for the next proplet provides the information required for visiting the successor proplet, here *Julia*.

Finally consider a rule of an LA-speak grammar. It is like an LA-think rule, except that it is extended to produce appropriate word form surfaces by using the core value as well as the morphosyntactic information of the cat and sem attributes. The following example shows the LA-speak rule application underlying transition 2 in 7.3, which navigates from the noun *Julia* back to the verb *know*, mapping the core value of the goal proplet into the appropriate surface know+s.

8.3 LA-SPEAK RULE APPLICATION

	i. *rule name* **NVs**	ii. *rule package* {VNs}	
	iii. *current proplet*	iv. *next proplet*	*output*
rule level	[noun: α cat: γ fnc: β prn: K] $\Rightarrow$	[verb: β sem: δ arg: α Y prn: K]	know+s $\Uparrow$ $\Rightarrow$ lex(β γ δ)
matching and binding of variables	$\Uparrow$	$\Updownarrow$	
Word Bank level	[noun: Julia cat: nm fnc: know prn: 625]	[verb: know sem: pres arg: Julia John prn: 625]	

As in an LA-think grammar, the next proplet (here *know*) serves as input for the next rule application(s). The difference between an LA-think and an LA-speak rule is that the latter also produces a surface (here know+s), using a variant of the lex function defined in NLC'06, 14.3.4. The resulting agent-internal, modality-free surface is used as a blueprint (template) for one of the agent's language synthesis components.

9 Database Schema of a Word Bank

In addition to designing (i) the basic items (proplets in Sects. 4, 5), (ii) the nature of their semantic connections (Sect. 6), and (iii) the algorithm modeling the hear, think, and speak modes (Sects. 7, 8), a level-four representation of language content requires (iv) the definition of a suitable database schema. When faced with the choice of a database, the most basic alternative is between a *coordinate*-addressable and a *content*-addressable approach (cf. Chisvin and Duckworth 1992 for an overview). Though peppered with patents, the content-addressable approach is less widely used than the coordinate-addressable approach. A content-addressable memory is suited best for the superfast retrieval of content which is written once and never changed.

A coordinate-addressable memory, e.g., an RDBMS, resembles a modern public library in which a book can be stored wherever there is space (random access) and retrieved using a separate index (inverted file) relating a primary key (e.g., author, title, year) to its location of storage (e.g., 1365). A content-addressable memory, in contrast, is like a private library in which books with certain properties are grouped together on certain shelves, ready to be browsed without the help of a separate index. For example, at Oxford University the 2 500 volumes of Sir Thomas Bodley's library from the year 1598 are still organized according to the century and the country of their origin.

In an initial response to a content-addressable approach, mainstream database scientists pointed out that it can be simulated by the coordinate-addressable approach (Fischer 2002), using well-established relational databases. The issue here, however, is whether or not the formal intuitions of the content-addressable approach can be refined naturally into a model of cognition.

Our point of departure is the data structure of proplets. For purposes of storage and retrieval, a proplet is specified uniquely[20] by its core and prn values (primary key). This suggests a two-dimensional database schema, as in a classic network database (cf. Elmasri and Navathe 1989/2010). A column of owner records is in the alphabetical order of their core values. Each owner record is preceded by a list of member records, distinguished in terms of their prn values. However, instead of using member and owner *records* we use equivalent member and owner *proplets*. The result is called a Word Bank.

As an example, consider storing the proplets of the content 6.1:

[20] Propositions containing two or more proplets with the same values, as in Suzy loves Suzy, require extra attention. They constitute a special case which (i) occurs very rarely and (ii) is disregarded here because it is easily handled.

9.1 STORING THE PROPLETS OF 6.1 IN A WORD BANK

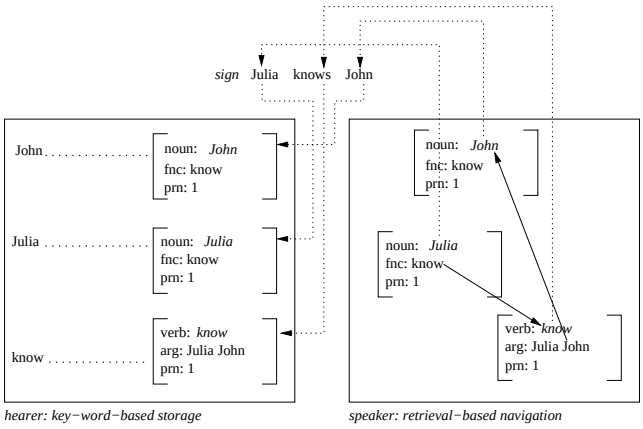
	member proplets	*now front*	*owner proplets*
	…		…
…	noun: John cat: nm fnc: ... prn: 610	noun: John cat: nm fnc: know prn: 625	core: John
	…		…
…	noun: Julia cat: nm fnc: ... prn: 605	noun: Julia cat: nm fnc: know prn: 625	core: Julia
	…		…
…	verb: know cat: decl arg: ... prn: 608	verb: know cat: decl arg: Julia John prn: 625	core: know
	…		…

An owner proplet and the preceding member proplets form a *token line*. The proplets in a token line all have the same core value and are in the temporal[21] order of their arrival, reflected by their prn values. In contrast to the task of designing a practical schema for arranging the books in a private library, the sorting of proplets into a Word Bank is simple and mechanical. The letter sequence of a proplet's core value completely determines its token line for storage: the storage location for any new arrival is the penultimate position in the corresponding token line, called the *now front*.

By storing content like *sediment*, the stored data are never modified and the need for checking consistency (Schewe and Thalheim 1994) is obviated. Changes of fact are written to the now front, like diary entries recording changes of temperature. New states are related to old ones by means of addresses, implemented as pointers.

The interaction between the algorithms of LA-speak and LA-hear and the content-addressable database of a Word Bank is illustrated by the following simplified example:

9.2 DBS MECHANISM OF TRANSFERRING CONTENT FROM SPEAKER TO HEARER



The Word Bank of the speaker contains the content 6.1, sorted into token lines. Using the algorithm of LA-speak (cf. 8.3), the navigation follows the compositional semantic relations of intrapropositional functor-argument (simplified compared to 7.3). The sur values of the proplets traversed are passed to the agent's action component and realized as unanalyzed modality-dependent external surface tokens (CLaTR'11, Chap. 2).

The hearer's recognition component uses the incoming external surfaces (*sign*) for lexical lookup (3.1). The lexical proplets are connected by the time-linear LA-hear derivation 7.1, and sorted into the hearer's Word Bank 9.1 at the end of the corresponding token lines. The communication is successful insofar as the hearer reconstructed the speaker's content using nothing but the time-linear sequence of unanalyzed modality-dependent external word form surfaces.

A Word Bank is content-addressable because no separate index (inverted file) is required. Furthermore, a Word Bank is scalable (a property absent or problematic in some other content-addressable systems). The cost of insertion is constant, independent of the size of the stored data, and the cost of retrieving a specified proplet grows only logarithmically with the data size (external access) or is constant (internal access). External access to a proplet requires (i) its core and (ii) its prn value, e.g., know 625. Most cognitive operations, however, require internal access based on addresses (pointers).

Compared to the classic 1969 CODASYL network database, a Word Bank is highly constrained. First, the members in a token line must share the owner's core value (no multiple owners). Second, the member proplets belonging to an owner proplet are listed in the temporal order of their arrival. Third, the only connections between proplets across token lines are the semantic relations of functor-argument and coordination. Fourth, like the relations between owners and members, the semantic connections are 1:n relations: one functor – several possible arguments; one first conjunct – several possible successors; one original – several possible coreferent address proplets.

A Word Bank is a kind of navigational database because it supports the navigation from one proplet to the next, using the semantic relations between proplets (7.2) and along token lines (10.2) like a railroad system, with the algorithm of LA-grammar (Sect. 8) as the locomotive. However, while the navigational databases of the past (Bachman 1973) and the present (XPath, Kay 2004) are intended to be driven by external human users, the system presented here is located inside an artificial cognitive agent, serving as the container and structural foundation of autonomous control (CLaTR'11, Chap. 5).

10 Retrieving Answers to Questions

Because the proplets derived in the hear mode (7.1) have a core value, they are suitable for (i) storage in a Word Bank (9.1). Because they also have a prn value, stored proplets support the operation of (ii) navigating from a given proplet to a successor proplet across token lines (7.2) in one of the two basic kinds of think mode. Moreover, because

[21] The token line for any core value is found by using a trie structure. The search for a proplet within a token line may use the prn value of the address in relation to the strictly linear increasing prn values. Technically, this may be based on binary search, in time *O(log(n))* (Cormen et al. 2009), or interpolation, in time *O(log(log(n)))* (Weiss 2005), where *n* is the length of the token line.

there is a speak mode which is riding piggyback on the think mode (7.3), the proplets in a Word Bank are suitable (iii) for language production from stored content as well.

Another operation enabled by a level-four content in a Word Bank is (iv) retrieving answers to questions. This operation is based on moving a query pattern along a token line until matching between the pattern proplet of the query and a content proplet is successful. A query pattern is defined as a proplet with at least one variable as a value.

Consider an agent thinking about girls. This means activating the corresponding token line, as in the following example:

10.1 EXAMPLE OF A TOKEN LINE

member proplets				*now front*	*owner proplet*
noun: girl fnc: walk mdr: young prn: 10	noun: girl fnc: sleep mdr: blond prn: 12	noun: girl fnc: eat mdr: small prn: 15	noun: girl fnc: read mdr: smart prn: 19		core: *girl*

As indicated by the fnc and mdr values of the member proplets, the agent happened to observe or hear about a young girl walking, a blonde girl sleeping, a small girl eating, and a smart girl reading.

For retrieval, the member proplets of a token line may be checked systematically by using a pattern proplet as the query. The following example shows the pattern proplet representing the query Which girl walked? as it applies to the token line 10.1:

10.2 APPLYING A QUERY PATTERN

			query pattern	noun:*girl* fnc: walk mdr: σ prn: K	
			matching (?)		
noun: girl fnc: walk mdr: young prn: 10	noun: girl fnc: sleep mdr: blonde prn: 12	noun: girl fnc: eat mdr: small prn: 15	noun: girl fnc: read mdr: smart prn: 19		core: *girl*

The attempt at matching indicated by *(?)* fails because the fnc values of the pattern proplet (i.e., walk) and of the member proplet (i.e., read) are incompatible. The same holds after moving the pattern proplet one member proplet to the left. Only after reaching the leftmost member proplet is the matching successful. Now the variable σ is bound to the value young and the variable K to the value 10. Accordingly, the answer provided to the question Which girl walked? is The young girl (walked).[22] A powerful extension of this method is connecting pattern proplets into schemata (e.g., 6.2, 6.5).

[22] For a more detailed presentation including yes/no questions, see NLC'06, Sect. 5.1.

11 Subactivation and Intersection in Database Semantics

The retrieval mechanism of DBS, based on (i) the content-addressable database schema of a Word Bank, (ii) pattern proplets, (iii) content proplets, and (iv) the semantic relations between proplets coded as addresses, supports a new, fully automatic (autonomous) database operation not available in coordinate addressable databases. This operation, called *subactivation*, is a kind of guided association which continuously accompanies the agent's current cognition with corresponding content stored in the agent's memory.

Subactivation works like a dragnet, pulled by the concepts activated by the agent's current recognition and inferencing, and providing them with relevant experiences and knowledge from the agent's past. As a form of association,[23] subactivation results in a mild form of selective attention.

Intuitively, subactivation may be viewed as highlighting an area of content at half-strength, setting it off against the rest of the Word Bank, but such that exceptional evaluations are still visible as brighter spots. In this way, the agent will be alerted to potential threats or opportunities even in current situations which would otherwise seem innocuous.

An elementary subactivation consists of three steps and may be of a a primary and a secondary degree. The first step is subactivating the token line which corresponds to a trigger concept provided by the agent's current situation. Consider the following example:

11.1 TRIGGER CONCEPT SUBACTIVATING CORRESPONDING TOKEN LINE

member proplets				*owner proplet*	*trigger concept*
adj: hot mdd: potato prn: 20	adj: hot mdd: water prn: 32	adj: hot mdd: potato prn: 55	adj: hot mdd: day prn: 79	... core: hot	⇐ *hot*

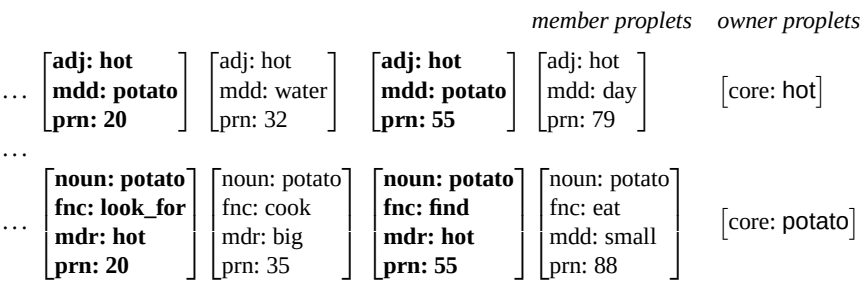
The trigger concept *hot* provided by the agent's external recognition matches the core value hot of an owner record in the agent's Word Bank. This subactivates the associated token line.

However, if a continuous sequence of trigger concepts were to always subactivate complete token lines, the resulting amount of data would be too large to be useful. Therefore, the second step is to use the semantic relations of functor-argument and coordination connecting the incoming concepts for restricting subactivation via *intersection* of the relevant token lines. In this way, the more semantically connected the concepts coming in, the more narrow and specific the subactivated data (search space reduction).

For example, if the agent's current recognition relates *hot* and *potato* as modifier-modified, the token lines of *hot* and *potato* might contain the following intersections, indicated typographically by bold face:

[23] Such as associating a certain place with a happy or an unhappy memory.

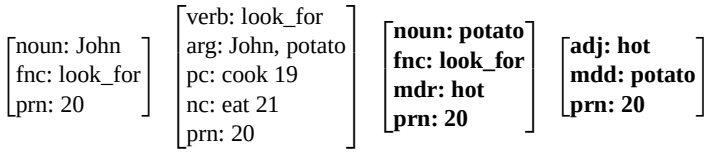
11.2 INTERSECTING TOKEN LINES FOR *hot* AND *potato*



The example contains two intersections, each consisting of two proplets sharing (i) a prn value and (ii) the modifier-modified relation between *hot* and *potato*. The intersections differ from each other in their respective prn values, 20 and 55, and the fnc values of the nouns, look_for and find.

The third step of a primary subactivation consists of completing an intersection into a full proposition. It is based on subactivating the intrapropositional semantic relations (spreading activation, Quillian 1968). For example, using the functor-argument coded by the leftmost proplets in 11.2, the intersection may be completed into a proposition:

11.3 COMPLETING AN INTERSECTION BY SPREADING ACTIVATION



This primary subactivation is based on the addresses potato 20 in the *hot* proplet, look_for 20 in the *potato* proplet, and John 20 in the *look_for* proplet.

While a primary subactivation utilizes the intrapropositional relations of functor-argument and coordination (cf. NLC'06, Chaps. 6 and 8), a secondary subactivation is based on the corresponding extrapropositional relations (cf. NLC'06, Chaps. 7 and 9). For example, using the pc (previous conjunct) and nc (next conjunct) values of the *look_for* proplet in 11.3, a secondary subactivation may spread from John looked for a hot potato to the predecessor and successor propositions with the verb values cook and eat and the prn values 19 and 21, respectively.[24]

In a cognitive agent, the degree of subactivation that is automatically selected at any current moment, including no subactivation at all, depends on the computational resources available. These depend on the agent's interests and current preoccupation. Compare, for example, running for one's life and a leisurely walk, both in the same park: the very same triggers will be completely ignored in the first case, but may result in rich subactivations in the second.

[24] Thus, the movie title All the President's Men (Pakula 1976) will likely activate Couldn't put Humpty Dumpty Together Again as a continuation, referring to R. M. Nixon. In fiction, our notion of triggering a spreading subactivation is illustrated by the madeleine experience of Proust 1913, which brings back an almost forgotten area of what he calls "l'édifice immense du souvenir."

Subactivation may be used for an application in natural language processing known as text mining (Feldman and Sanger 2006). As in the other uses of DBS for NLP applications, such as automatic word form recognition and syntactic-semantic parsing in the hear mode, the use of subactivation for NLP consists in simplifying the function it has been originally designed for.

More specifically, instead of using subactivation for the autonomous control of a talking robot (CLaTR'11, Sect. 5.4), its NLP application is limited to the querying of a textual database. Thereby, only the concepts of the user's query are used for subactivating only content in a Word Bank selected by the user. In this way, the already very effective level-two retrieval based on letter sequences may be enhanced with a level-four analysis based on the semantic relations of DBS content.

12 RMD Corpus

While the core values, the semantic relations, and the levels of abstraction are agent-internal constructs of cognition, recorded language data are agent-external objects. A classic example is the Brown Corpus (Kučera and Francis 1967, Francis and Kučera 1982), designed as an electronically stored monolingual synchronic corpus. Its scientific purpose is to provide an accurate snapshot of American English in the year 1961.

Such a corpus should contain the vocabulary, constructions, collocations, idioms, and frequency distributions characteristic of different domains in the language and the time interval selected. This may have practical ramifications. For example, when expanding automatic word form recognition, token recognition rates will improve best if the most frequent word forms are integrated into the software first, and similarly for parsing syntactic-semantic constructions, etc.

A well-designed[25] corpus should be *representative* and *balanced* (FoCL'99, Sect. 15.3). However, no corpus can be proven to fulfill these desiderata (Oostdijk 1988).[26] Therefore, instead of building a fixed, one-shot corpus like the BNC, an empirically more broad-based, more differentiated, more interesting, and more long-term approach is the construction of a standardized RMD corpus, i.e., a Reference-Monitor corpus structured into Domains.

The reference corpus consists of a subcorpus for everyday language, complemented by subcorpora for such domains as anthropology, architecture, astronomy, biology, chemistry, ecology, entomology, etology, fiction, history, law, medicine, music, philosophy, physics, politics, religion, and sport (von der Grün 1998).Grün, A. von der The domains may be established using the cataloging work of the library/information sciences. Their sizes may be determined by methods evolved from those used for the Brown corpus.

The reference corpus is continued with monitor corpora (Sinclair 1991), following every year. The annual monitor corpora must resemble their reference corpus in ev-

[25] A well-designed corpus is in contradistinction to some random, all-you-can-get collection of language texts, euphemistically called an "opportunistic corpus" and offered as a "resource."

[26] Just consider quantifying the word form frequency distributions of a newspaper article or a radio address. The problem is the asymmetry between the speak and the hear mode: how should the word forms be ranked if they are produced once, but interpreted by millions?

ery way: overall size, choice of domains, domain sizes, etc. This is easily achieved by building the reference corpus and the monitor corpora with texts from the same carefully selected basket of *renewable* language data: newspapers for everyday language, established journals for specific domains, and a selection of fiction, e.g., movie scripts.

The monitor corpora generated every year require linguistic analysis. Statistical tagging, however, cannot handle such a continuous flow of new data because of its need for massive manual markup (Sect. 2). The DBS solution is automation.

This applies to (i) the collecting of texts for the monitor corpora once the initial set of renewable language sources has been settled on, (ii) their statistical analysis once a useful routine as been established, and (iii) automatic word form recognition and syntactic-semantic parsing. Replacing manual markup with automatic processing ensures the quality of standardization necessary for meaningful comparisons between monitor corpora of different years. It also saves the instruction and supervision of large numbers of markup personnel, not to mention the cost, and obviates the follow-up labor required for standardization and for ensuring interoperability.

A succession of monitor corpora allows a detailed view of how the language and the culture are developing, in different domains and over many decades. Statistical analysis will show, for example, how politics and natural disasters cause a temporary frequency increase of certain words in certain domains. A carefully built RMD corpus is in the interest of the whole language community and should be entrusted to the care of a national academy.

13 Applications

If a functional framework of natural language communication works properly at all levels of abstraction, though initially with small (and highly relevant) data coverage only, then all that remains to be done is to increase the data coverage. For natural language communication, this is a mammoth project, though nothing compared to projects in physics (CERN) or biology (human genome project), for example.

Extending the data coverage as a form of upscaling may be used to directly improve commercial applications which use the system for their natural language processing needs. Take for example LA-morph, the automatic word form recognition software, running with a certain natural language of choice. Its data coverage may be extended by adding missing entries to the lexicon and by optimizing the allo- and combi-rules for the natural language at hand (Sect. 3). This broadens the base for syntactic-semantic analysis, which depends on high quality automatic word form recognition. It also provides practical applications with better results for retrieval based on content words.

A second area for completing data coverage is extending the syntactic-semantic analysis. When applied to a new (i.e., previously unanalyzed) natural language, the LA-hear parser will at first handle only a few constructions. As the language is being studied, more and more constructions (like infinitives, prepositional phrases, relative clauses, etc.) are added to the grammar, tested, and improved. When the LA-hear parser encounters input it cannot yet handle, the passage may be traversed at a lower level of

detail until proper parsing can resume (robustness). For this, LA-grammar is especially suitable because it computes possible *continuations*[27] in a time-linear derivation order.

Expanding syntactic-semantic parsing in the agent's hear mode is more demanding than automatic word form recognition. This effort should not go un-rewarded from the application side, however. The coding of functor-argument and coordination extends recall and precision from lexically analyzed word forms to phrases and clauses, and from there to sentences, paragraphs, and text. Technically, this amounts to an extension from matching lexically analyzed content words stored within token lines in the Word Bank to matching semantic relations between content words defined across token lines.[28]

A third area for extending the data coverage is the think mode, which combines (i) LA-think and (ii) inferencing (CLaTR'11, Chap. 5). The basic mechanism of LA-think is *selective activation* by navigating along the semantic relations in a Word Bank.[29] The navigation is used to activate and report self-contained content. Inferencing is used for deriving the different perspectives of the speaker and the hearer on content,[30] and to compute blueprints for action, including language action. Together with current and stored data, LA-think and inferencing constitute the agent's autonomous control, which has many practical applications, with and without language.

Finally, consider LA-speak. It takes content as input and produces corresponding surfaces as output. If the content has already been serialized by the navigation along the semantic relations in the Word Bank, the task of LA-speak is confined to adjusting to the word order of the language and to providing proper lexicalization with proper perspective (e.g., tense) and proper morphosyntactic adjustments (e.g., agreement).

This work will not go unrewarded from the application side either. The obvious application is *query answering* in natural language. Thereby the LA-speak part is only the tip of the iceberg. Prior to answering, the query is converted automatically into several schemata which are used to subactivate corresponding contents (Sect. 11). These data are processed into the content for the query answer by means of intersection and inferencing. Once the resulting answer content has been derived, it is passed to LA-speak for realization as an unanalyzed external surface in the modality of choice.

While specific applications may benefit selectively from nurturing a particular component, all applications will benefit simultaneously from a methodical upscaling of the DBS robot as a whole. An application which does not require certain abilities may be run with a DBS version in which they have been switched off.[31]

---

[27] This is in contradistinction to the grammars of the sign-oriented approaches, such as phrase structure grammar and categorial grammar, which compute possible *substitutions* (FoCL'99, Sect. 10.1).

[28] In addition, the user may load the proplets in a proprietary database with additional attributes and values as needed for the application. One such application of LA-morph and LA-hear is speech recognition; it could well benefit from the search space reduction resulting from an LA-hear parser computing possible continuations.

[29] A Word Bank may be viewed as a syntactic-semantic network. For some questions and results of linguistic networks, see Liu 2011, Solé et al. 2010, Sowa 1987/1992, Brachman 1979, and others.

[30] The interaction between LA-think, LA-speak, and inferencing is shown in CLaTR'11, Chap. 10, with an example of dialogue.

[31] For example, a dialogue system over the phone may switch off the ability to read.

14 Unifying Linguistic Theory and Practice

The systematic transfer from a continuously improving DBS system to commercial applications of natural language processing and human-machine communication may be illustrated by the following vision: Every year, when the current monitor corpus (Sect. 12) has been put through the software grinder of word form recognition, syntactic-semantic parsing, frequency analysis, and comparison with preceding monitor corpora, the results are used for a software version with improved (updated) data coverage.

By making new versions available to paying subscribers for their natural language processing needs, all or most of the research costs may be recovered. For this to work long-term, a new release must not require any labor (e.g., additional personnel training) from the subscriber.[32] Also, each new version must enhance service directly and noticeably, so that paying subscribers are attracted and kept in sufficient numbers. Improvements from one version to the next may be achieved rather easily because there are large fields of empirical data which merely need to be "harvested."

The harvester for collecting and analyzing new language data as well as testing and correcting is the DBS software machine. Originally designed to model the language communication mechanism as a talking robot, its off-the-shelf components for the lexicon, word form recognition, syntactic-semantic parsing, and so on, may be used also for storing the language-dependent data of a new (i.e., not yet analyzed) language: words are added to the robot's lexicon component, just as compositional structures are added to the LA-Morph, LA-hear, LA-think, and LA-speak grammars in the robot's rule component. Also, culture-, domain-, and application-dependent content may be added to the Word Bank.

Storing the analysis of a natural language in the DBS robot makes the harvest directly available for computational testing by the scientists and for computational applications by the users. This works not only for the hear mode, as in testing on a corpus, but for the full cycle of natural language communication. The testing is designed (i) to automatically enhance the robots performance by learning (CLaTR'11, Chap. 6), and (ii) to provide the scientists with insights for improving the robot's learning abilities.

For long-term linguistic research, there is no lack of renewable language data, namely (i) the natural changes year to year within the domains of a given language and (ii) a wide, constantly extending range of applications in human-machine communication and natural language processing. In addition, there is (iii) the great number of natural languages not yet charted, or not yet charted completely, including English.[33]

Charting a new natural language in DBS is a standard procedure, but it has to deal with relatively large amounts of data. As more and more languages are analyzed, however, charting is accelerated because software constructs may be reused, based on similarities in lexicalization, in productive syntactic-semantic structures, in collocations, constructions, and idioms, and in inferencing. To better support day-to-day research,[34] these

[32] Except for routine installation. Automatic updates (patch computing) are also possible.

[33] This raises a question for the sign-oriented schools of linguistics, which have been well-funded for more than five decades, employing thousands of researchers.

[34] For example, in work on typology or on expanding a given language to new constructions.

standardized software constructs and their declarative specifications may be stored in system libraries, organized for families of languages.

Conclusion

Modeling the cycle of natural language communication as an efficient computer program has long been overdue. Yet its mechanism turned out to be simple and straightforward, as shown in Sects. 4–10, esp. 9.2. It is based on a **S**urface Compositional analysis of natural language expressions (methodological principle), a time-**L**inear derivation order (empirical principle), and an agent-**I**nternal (ontological principle) pattern **M**atching (functional principle) between the language and the context level inside the head of a cognitive agent.

Though not part of current mainstream linguistics and philosophy, these principles are well-motivated. Together, they structure and constrain the broadened empirical base of our agent-oriented approach. They combine into the SLIM theory of language (FoCL'99, NLC'06, CLaTR'11), which is designed to process any natural language in linear time with completeness of function and of data coverage. As the theoretical foundation of DBS, SLIM unifies theoretical and practical issues of language analysis and processing, respectively, such that they cross-fertilize each other, similar to the natural sciences.

Our example of a practical application benefiting from the theoretical development of DBS is an improvement of retrieval based on the automatic level-four derivation of content (1.1). The DBS hear mode coding of content from text amounts to a standardized markup which happens to use primarily (i) traditional notions of natural language grammar and (ii) the base forms of content words. The main grammatical notions are the parts of speech noun, verb, and adj, coded as the core *attributes* of proplets, and the compositional semantic relations of functor-argument and coordination, intra- and extrapropositional, coded as the *values* of the continuation attributes fnc, arg, mdr, mdd, nc and pc, and as the numerical value of the book-keeping attribute prn.

The automatic DBS hear mode coding of text into level-four content does not interfer with letter-based level-two retrieval because letter sequences are still available as the core and continuation values of proplets. Letter-based retrieval is enhanced, however, because content provides (i) the compositional semantic relations between proplets which can be (ii) recognized by the pattern-based retrieval mechanism of the content-addressable database of a Word Bank (Sects. 10, 11).

Level-three metadata markup may also be utilized in DBS: the markup notions designed by scientists to extend and standardize the metadata are embedded into the automatic word form recognition, the syntactic-semantic parsing, the inferencing, and the frequency counts of DBS in the form of additional proplet attributes and values. This results in an automatic insertion of the proper metadata during the processing of new text, making them available for classification, standardization, and retrieval, but without the current need for manual labor.

References

Bachman, C. W. (1973) "The Programmer as Navigator," 1973 ACM Turing Award Lecture, *Comm. ACM*, Vol. 16:11.653–658

Bar-Hillel, Y. (1953) "Some Linguistic Problems Connected with Machine Translation," *Philosophy of Science*, Vol. 20:217–225

Brachman, R. J. (1979) "On the Epistemological Status of Semantic Networks," in N. Findler (ed.), *Associative Networks*, Academic Press, pp. 3–50

Brown, P., S. Della Pietra, V. Della Pietra, and R. Mercer (1991) "Word Sense Disambiguation Using Statistical Methods," Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics, Berkeley, CA, June 1991, 264-270

Burnard, L. (ed.) (1995) *Users Reference Guide British National Corpus Version 1.0*, Oxford U. Computing Services, England: Oxford

Burnard, L. (ed.) (2007) *Reference Guide for the British National Corpus (XML Edition)*, Oxford U. Computing Services, England: Oxford

Church, K., and R.L. Mercer (1993) "Introduction to the Special Issue on Computational Linguistics Using Large Corpora," *Computational Linguistics*, Vol. 19.1:1–24

CLaTR'11 = Hausser, R. (2011) *Computational Linguistics and Talking Robots*, Berlin, New York: Springer

Cormen, T. H., C. E. Leiserson, R.L. Rivest, and C. Stein (2009) *Introduction to Algorithms, 3rd ed.*, Cambridge, MA: MIT Press

DeRose, S. (1988) "Grammatical Category Disambiguation by Statistical Optimization," *Computational Linguistics*, 14.1:31–39

Elmasri, R. and S. B. Navathe (1989/2010) *Fundamentals of Database Systems, 6th ed.*, Redwood City, CA: Benjamin-Cummings

Feldman, R., and J. Sanger (2006) *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*, Cambridge University Press

Fischer, W. (2002) *Implementing Database Semantics as an RDMS* (in German), Studienarbeit am Institut für Informatik der Universität Erlangen-Nürnberg (Prof. Meyer-Wegener), published as CLUE-Arbeitsbericht 7 (2004), available at http://www.linguistik.uni-erlangen.de/clue/de/arbeiten/arbeitsberichte.html

FoCL'99 = Hausser, R. (1999/2001) *Foundations of Computational Linguistics, Human-Computer Communication in Natural Language, 2nd ed.*, Berlin, New York: Springer

Francis, W.N. (1980) "A tagged corpus: Problems and Prospects," in S. Greenbaum, G. Leech, and J. Svartvik (eds.), 192–209

Francis, W. N. and H. Kučera (1982) *Frequency Analysis of English Usage: Lexicon and Grammar*, Boston: Houghton Mifflin

Fredkin, E. (1960) "Trie Memory," *Commun. ACM*, Vol. 3.9:490–499

Garside, R., G. Leech, and G. Sampson (1987) *The Computational Analysis of English*, London: Longman

Grün, A. von der (1998) *Wort-, Morphem- und Allomorphhäufigkeit in domänenspezifischen Korpora des Deutschen*, M.A. thesis, CLUE

Handl, J. (2011) *Inkrementelle oberflächenkompositionale Analyse und Generierung von natürlicher Sprache*, Ph.D. thesis, CLUE

Hausser, R. (2007) "Comparing the Use of Feature Structures in Nativism and in Database Semantics," in Jaakkola, H., Y. Kiyoki, and T. Tokuda (eds.) *Information Modelling and Knowledge Bases XIX*, Amsterdam: IOS Press Ohmsha

Hausser, R. (2009) "Modeling Natural Language Communication in Database Semantics," in M. Kirchberg and S. Link (eds.), *Proceedings of the APCCM 2009*, Australian Computer Science Inc., CRPIT, Vol. 96, 17-26

Hausser, R. (2011) "Corpus Linguistics, Generative Grammar, and Database Semantics", in *The Phraseological View of Language - A Tribute to John Sinclair*, T. Herbst, S. Faulhaber and P. Uhrig (eds.), Berlin, New York: De Gruyter Mouton

Jaggers, M., and K. Richards (1967) *Between the Buttons*, London: Decca

Kay, M. (2004) *XPath 2.0 Programmer's Reference (Programmer to Programmer)*, Wrox Press

Knuth, D. E. (1998) *The Art of Computer Programming. Vol. 3, 2nd ed.* Boston: Addison-Wesley

Kučera, H. and W. N. Francis (1967) *Computational Analysis of Present-Day English*. Providence, Rhode Island: Brown U. Press

Leech, G. (1995) "A Brief User's Guide to the Grammatical Tagging of the British National Corpus," web site

Leech, G., R. Garside, and E. Atwell (1983) "The Automatic Grammatical Tagging of the LOB Corpus," *ICAME Journal* 7, 13–33

Liu, Haitao (2011) "Linguistic Networks: Metaphor or Tool?" *Journal of Zhejiang University*, (Humanities and Social Science) Vol. 41.2:169–180.

Marshall, I. (1987) "Tag Selection Using Probabilistic Methods," in R. Garside, G. Leech, and G. Sampson *The Computational Analysis of English*, London: Longman

McKeown, K. (1985) *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text*, Cambridge: CUP

NLC'06 = Hausser, R. (2006) *A Computational Model of Natural Language Communication: Interpretation, Inference, and Production in Database Semantics*, Berlin, New York: Springer

Oostdijk, N. (1988) "A Corpus Linguistic Approach to Linguistic Variation," in G. Dixon (ed.): *Literary and Linguistic Computing*, Vol. 3.1

Pakula, A. J. (1976) *All the President's Men*, Warner Brothers

Proust, M. (1913) *Du côté de chez Swann*, ed. by Jean-Yves Tadie et al., Bibliothèque de la Pléiade, Paris: Gallimard,1987–1989

Pustejovsky, J. (1995) *The Generative Lexicon*, Cambridge, MA: MIT Press

Quillian, M. (1968) "Semantic Memory," in M. Minsky (ed.), *Semantic Information Processing*, pp. 227–270, Cambridge, MA: MIT Press

Sampson, G. (1999) "Book Review Roland Hausser, Foundations of Computational Linguistics: Man-Machine Communication in Natural Language," University of Sussex elsnet Dec 1999

Schewe, K.D., and B. Thalheim (1994) "Achieving Consistency in Active Databases," in S. Chakravarthy and J. Widom (eds.) Proc. IEEE RIDE-ADS, Houston

SCG'84 = Hausser, R. (1984) *Surface Compositional Grammar*, Munich: Wilhelm Fink Verlag

Sharman, R. (1990) *Hidden Markov model methods for word tagging*, Report 214, IBM UK Scientific Centre, Winchester

Sinclair, J. (1991) *Corpus Concordance Collocation*, Oxford: Oxford U. Press

Solé, R. V., B. Corominas-Murtra, S. Valverde, and L. Steels (2010) "Language networks: their structure, function, and evolution," *Complexity*, Vol. 15.6:20–26

Sowa, J. (1987/1992) "Semantic networks," *Encyclopedia of Artificial Intelligence*, edited by S. C. Shapiro; revised and extended for the second edition, 1992. New York: Wiley
Available at http://www.jfsowa.com/pubs/semnet.htm

TCS'92 = Hausser, R. (1992) "Complexity in Left-Associative Grammar," *Theoretical Computer Science*, Vol. 106.2:283–308

Weiss, M.A. (2005) *Data Structures and Problem Solving Using Java. 3rd ed.*, Upper Saddle River, NJ: Pearson Addison-Wesley

Zipf, G. K. (1949) *Human Behavior and the Principle of Least Effort*, Cambridge, MA: Addison-Wesley