

Modeling Natural Language Communication in Database Semantics

Roland Hausser¹

¹ Abteilung Computerlinguistik
Universität Erlangen-Nürnberg (CLUE)
Bismarckstr. 6 & 12, D-91054 Erlangen, Germany
Email: rrh@linguistik.uni-erlangen.de

Abstract

Database Semantics (DBS) is a computational model of how communicating with natural language works. Defined at a level of abstraction which may be applied to natural and artificial agents alike, such a model is a precondition for achieving free human-machine communication in natural language, and thus has great potential for a wide range of practical applications.

The basic functionality of DBS is a cognitive agent's turn taking between the *speaker mode* (mapping content stored in memory into language surfaces) and the *hearer mode* (mapping surfaces into content for storage in memory). DBS is defined in terms of (i) the data structure of flat feature structures, (ii) the algorithm of time-linear LA-grammar, and (iii) the database schema of a classic network database.

Keywords: language production; language interpretation; turn taking; data structure; algorithm; database schema; parts of speech; semantic relations; functor-argument structure; coordination; elementary, phrasal, and clausal level; hearer mode; speaker mode; proplets; LA-grammar; Word Bank

1 Linguistic Background

The analysis of natural languages and of natural language communication has long been an interdisciplinary enterprise involving linguistics, philosophy, psychology, physiology, neurology, sociology, mathematics, and computer science. As a consequence, we are faced today with a vast patchwork of different theories, topics, and empirical analyses. Rather than attempting an overview of current work, which by necessity would have to be selective, let us begin with those aspects of traditional¹ language analysis which are (i) uncontroversial and (ii) helpful for understanding the remainder of this paper.

1.1 Compositionality

The sentences of natural language are built from word forms. For example, the sentence *Julia knows John.* is built from the word forms *Julia*, *knows*, *John*, and the full stop. Each word form has a surface and a lexical meaning.² By ordering word form surfaces of

a given natural language into a grammatically well-formed sequence, the associated lexical meanings are combined into a sentence meaning.

Lexical meanings are analyzed in the linguistic field of *lexical semantics*, while the derivation of sentence meanings is analyzed in *compositional semantics*. The relation between lexical semantics and compositional semantics is described by the Fregean Principle, named after Gottlob Frege (1848–1925):

1.1.1 FREGEAN PRINCIPLE

The meaning of a complex expression is a function of the meaning of its parts and their mode of composition.

For example, the two sentences

The dog bites the man.
The man bites the dog.

consist of exactly the same parts (English word form surfaces), but differ in their word order (mode of composition) – which is one reason why two sentences may turn out to have different sentence meanings.

Furthermore, the two sentences

The dog is chasing a cat.
The dog is chasing a squirrel.

have the same word order (mode of composition), but differ in one of their parts (namely the respective last word form) – which is the other reason why two sentences may turn out to have different sentence meanings.³

1.2 Parts of Speech

The words of a natural language are divided into *content words*, for example, *table*, *write*, or *beautiful*, and *function words*, for example, *the*, *and*, or *before*. The number of content words in a natural language is several ten thousands, while the number of function words is only a few hundred. However, while a function word like *the* is used very often, comprising 6% of the running word forms in a corpus, a content word like *table* has a comparatively much lower frequency.

The most basic way of classifying a word is according to its *part of speech*. The basic parts of speech are the *nouns*, the *verbs* and the *adjectives*.⁴

Consider the following examples:

of meaning in an agent-oriented approach see FoCL'99, Sects. 3.2–3.4, 4.2, 4.3, 6.1, 6.2, 19.1; NLC'06, Sects. 4.2–4.4, 5.4–5.6.

³In order for the Fregean Principle to hold it is essential to distinguish (i) between the literal meaning of language expressions (meaning₁) and the speaker meaning of utterances (meaning₂) and (ii) between the unanalyzed and the analyzed surface. As shown in FoCL'99, Sects. 4.2–4.5, the Fregean Principle applies solely to the meaning₁ of analyzed surfaces.

⁴Additional parts of speech used in the classic grammar of Latin are *adverb*, *pronoun*, *preposition*, *conjunction*, and *interjection*.

Copyright ©2009, Australian Computer Society, Inc. This paper appeared at the Sixth Asia-Pacific Conference on Conceptual Modelling (APCCM 2009), Wellington, New Zealand, January 2009. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 96, Markus Kirchberg and Sebastian Link, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

¹For a representative contemporary example see Benner 2008.

²For example, the meaning of the English surface *tree* has the surface *arbre* in French and *Baum* in German. For a detailed analysis

1.2.1 PARTS OF SPEECH AT ELEMENTARY LEVEL

1. *noun*: Examples are Julia (proper name) and table (common noun).
2. *verb*: Examples are sleep (one-place), write (two-place), and give (three-place).
3. *adjective*: Examples are brief (adnominal) and briefly (adverbial).

In the linguistic school of structuralism, the parts of speech are justified by means of substitution and movement tests.⁵ For example, Julia and John have the same part of speech because substituting John for Julia in the sentence *Julia slept briefly* results in another well-formed sentence with a similar meaning. In contrast, substituting Julia with wrote would result in an ungrammatical expression. However, substituting wrote in *John wrote a letter* with typed renders another well-formed sentence, indicating that wrote and typed are of the same part of speech.

In addition to the classification of a word in terms of its part of speech, there is the distinction between the different *word forms* of a word. For example, book, book's, books, and books' are different forms of the noun *book*. Similarly, learn, learns, learned, and learning are different forms of the verb *learn*. And quick, quickly, quicker, and quickest are different forms of the adjective *quick*. Strictly speaking, sentences, texts, and dialogues of natural language consist of word forms rather than words.⁶

Forms of elementary content and function words may be combined into complex parts of speech at the *phrasal* and the *clausal* level:

1.2.2 PARTS OF SPEECH AT THE PHRASAL LEVEL

1. *noun*: An example is the pretty young girl, which consists of a determiner (function word), two elementary adjectives, and an elementary noun, the latter being content words.
2. *verb*: An example is could have been sleeping, which consists of (forms of) a modal verb, two auxiliaries, and a main verb.
3. *adjective*: An example is for five minutes, which can be used adnominally as well as adverbially and consists of a preposition, a determiner, and a noun, the latter being a content word.

Again, the functional equivalence between elementary and clausal parts of speech may be shown by substituting one for the other. For example, the sentence *Julia slept briefly* consists of an elementary noun, an elementary verb, and an elementary adjective. Replacing Julia with *The pretty young girl* results in a similar sentence with a phrasal noun, namely the pretty young girl slept briefly. By replacing the elementary verb and adjective with phrasal counterparts as well, we might get *the pretty young girl could have been sleeping for five minutes*. – which has the same grammatical structure at the phrasal level as *Julia slept briefly*. has at the elementary level.

At the highest level, elementary and phrasal parts of speech are combined into clausal parts of speech, which serve as nouns and as adjectives. The part of speech *verb* at the clausal level is equivalent to a complete clause or sentence.

Adverbs are treated here as the adverbial use of adjectives (in contradistinction to the adnominal use); pronouns like you or he are treated as nouns; prepositions are treated as function words which make phrasal adjectives, conjunctions are treated as function words which make coordinations (parataxis) and clausal nouns or adjectives (hypotaxis). A treatment of interjections is omitted.

⁵Cf. FoCL'99, Sect. 8.4.

⁶For a more detailed discussion see FoCL'99, Chapt. 13.

1.2.3 PARTS OF SPEECH AT THE CLAUSAL LEVEL

1. *noun*: An example is that Julia read a book, as in *That Julia read a book pleased her mother*. – which has a similar grammatical structure as *Julia pleased her mother*. In other words, *that Julia read a book* may serve the same grammatical function as the elementary noun *Julia* or the phrasal noun *the pretty young girl*.
2. *adnominal adjective*: An example is which Mary saw, as in *The dog which Mary saw barked*. – which has a similar grammatical structure as *The black dog barked*. In other words, the adnominal clause *which Mary saw* has the same grammatical function as the elementary adnominal *black* or the phrasal adnominal *with the black fur*.
3. *adverbial adjective*: An example is When Fido barked, as in *When Fido barked Mary smiled* – which has a similar grammatical structure as *Recently Mary smiled*. In other words, the adverbial clause *When Fido barked* has the same grammatical function as the elementary adverb *recently* or the phrasal adverb *after her nap*.

1.3 Semantic Relations

In natural language, word forms are assembled into well-formed, meaningful expressions by means of only two kinds of semantic relations, namely (i) *functor-argument* and (ii) *coordination* structure. These semantic relations are based to a large extent on the parts of speech of the words and expressions, and are the topic of the linguistic disciplines of syntax and compositional semantics.

Functor-argument structure is used to build the different sentential moods, such as declarative, interrogative, and imperative in English. In a proposition (sentence), there is the obligatory relation between the functor (verb) and its arguments (nouns), and the optional relation between a verb or noun and its modifiers (adjectives).

For example, the functor-argument structure of *Julia knows John* is based on the lexical analysis of the verb form *knows* as a *two-place functor* and of the nouns *Julia* and *John* as *arguments*. Similarly, the functor-argument structure of *Julia slept* is based on the lexical analysis of the verb form *slept* as a *one-place functor* and *Julia* as the argument. And the functor-argument structure of *John gave Julia a flower* is based on the lexical analysis of *gave* as a *three-place functor* and the arguments *John*, *Julia*, and *a flower*.

Depending on the lexical valency structure of the verb, its arguments are distinguished with respect to their grammatical role. For example, in the sentence *Julia slept*., the argument *Julia* serves as the *subject*; in *Julia knows John*., *Julia* serves as the subject and *John* as the *object*; and in *John gave Julia a flower*., *John* serves as the subject, *Julia* as the *indirect object*, and *a flower* as the *direct object*. These distinctions apply also to arguments at the phrasal and the clausal level.

In order for a sentence to be complete, it must have as many arguments as required by its verb. It is in this sense that the arguments are *obligatory*. The modifiers, in contrast, are *optional*. For example, the sentence *Julia slept briefly* is still complete without the adverbial modifier *briefly*. Similarly, the sentence *The black dog barked* is still complete without the adnominal modifier *black*. The optional nature of modifiers applies also at the phrasal and clausal level.

While functor-argument structure assembles expressions of different parts of speech, *coordination* assembles word forms, phrases, and clauses of the same part of speech. For example, *Julia, Susanne,*

and John is an elementary noun coordination, bought, cooked, and ate is an elementary verb coordination, and quickly, tastefully, and professionally is an elementary adjective coordination in adverbial use.

Clausal coordination may be *intrasentential*, as in Julia slept, John sang, and Susanne dreamed. or *extrasentential*, as in Julia slept. John sang. Susanne dreamed. An intrasentential form are gapping constructions, such as Bob ate an apple, walked his dog, and read the paper (subject gap), Bob ate an apple, Jim a pear, and Bill a peach (verb gap), and Bob bought, Jim peeled, and Bill ate the peach. (object gap).⁷

The combination of functor-argument and coordination structure at the elementary, the phrasal and the clausal levels may result in constructions of arbitrary complexity and length: intrasententially, functor-argument structure provides for subclause constructions of arbitrary complexity in the different sentential moods; extrasententially, coordination provides for text or dialogue of arbitrary length.

1.4 Word Order, Word Forms, Agreement

While the functor-argument and coordination structure are based semantically on the parts of speech, their coding depends on *word order*, *word forms*, and *agreement*. The role of these structural surface properties varies widely between different natural languages and is thus highly language-dependent.

For example, in English the grammatical role of the arguments is coded by means of word order: in Julia knows John., the first argument is the subject and the second argument the object. Changing the order into John knows Julia. changes the subject and the object. Furthermore, in declarative sentences of English, the verb (functor) must be in post-nominative position.

In Russian, in contrast, the word order is free and the case role of the arguments is coded morphologically in the endings (suffixes) of the word forms. Thus the content of English Julia knows John has six surfaces in Russian, corresponding in word order to Julia_N knows John_A, Julia_N John_A knows, Knows Julia_N John_A, Knows John_A Julia_N, John_A knows Julia_N, and John_A Julia_N knows., while the content of English John gave Julia a flower has a total of twenty-four corresponding surfaces in Russian.

Because of the fixed word order of English, the number of different word forms may be comparatively small. In common nouns, there is the distinction between singular and plural, e.g., book vs. books and between unmarked and genitive case, e.g., book vs. book's and books vs. books'. In finite verbs, there is the distinction between 3. person singular present tense, and the other persons and numbers (numeri) in present tense, e.g., sings vs. sing, and the distinction between present tense and past tense, e.g., sing and sang. In adjectives, there is the distinction between adnominal and adverbial use, e.g., beautiful and beautifully, and the distinction between the positive, comparative, and superlative, as in fast, faster, fastest.

The choice of a particular form of a word is partially semantic, as between Julia knows John. and Julia knew John., and partially syntactic, as in every girl (singular) and all girls (plural). In the latter case, the correct choice of word forms is a matter of grammatical *agreement*. For example, in English there is agreement between the subject and the finite verb in the present tense, and between determiners like a(n), every, all, one, two, three, etc., and their nouns.⁸

⁷Even though gapping constructions turn out to be of very low frequency in corpora, we know of no natural language in which they (i) don't occur and (ii) don't have strong grammatical intuitions with the native speakers. For a detailed analysis of coordination including gapping constructions see NLC'06, Chaps. 8 and 9.

⁸For a DBS analysis of "quantifiers" see NLC'06, Sect. 6.2.

1.5 Automatic Word Form Recognition

For purposes of computational linguistics, the traditional distinctions of grammar at the level of elementary word forms may be represented as flat feature structures. A feature structure is defined as a set of attribute value pairs or avp. In DBS, a flat⁹ feature structure representing word forms is called a *proplet*: just as a droplet is the smallest element in a sea of water, a proplet is the smallest element in a sea of propositions. Consider the following examples of lexical proplets, which illustrate the basic parts of speech with forms of content words:

1.5.1 EXAMPLES OF LEXICAL PROPLETS

[sur: books noun: <i>book</i> cat: pl sem: count fnc: mdr: nc: pc: prn:]	[sur: swam verb: <i>swim</i> cat: s3' v sem: past arg: mdr: nc: pc: prn:]	[sur: slowly adj: <i>slow</i> cat: adv sem: pos mdd: nc: pc: prn:]
---	--	--

The first attribute *sur* (surface) has the language-dependent surface of the word as its value. The second attribute is called the *core attribute* of the proplet. It indicates the part of speech and has a language-independent meaning as its value, called the *core value*. Core values may be of the different kinds of sign *symbol*, *indexical*, and *name*.¹⁰ For simplicity, the meanings are represented by corresponding words of English written in italics,¹¹ used here as place holders or names of the meaning in question.

The third attribute *cat* specifies grammatical properties relevant for the combinatorics of the word form, such as singular vs. plural in nouns, the valency positions of verbs, and the adnominal versus adverbial use in adjectives. The fourth attribute *sem* specifies morphologically coded aspects of meaning which are not directly relevant for the combinatorics of the word form, such as tense.

The remaining attributes are non-lexical and therefore have no values yet. The attributes *fnc* (functor), *mdr* (modifier), *nc* (next conjunct), and *pc* (previous conjunct) in nouns are called the *continuation attributes* which code the functor-argument and coordination relations of the proplet. Verbs and adjectives differ from nouns in that verbs have *arg* (argument) and adjectives *mdd* (modified) as their obligatory continuation attribute.¹²

In addition to the continuation attributes there is the *bookkeeping attribute* *prn* (proposition number) in nouns, verbs, and adjectives. Other bookkeeping attributes used only in the software are *trc* (transition counter) and *wrn* (word number). While lexical attributes have their values defined in the on-line lexicon and continuation attributes receive their values by the rules of syntactic-semantic parsing, bookkeeping attributes have numerical values assigned by the parsing engine.

The on-line definition of lexical proplets is the basis of *automatic word form recognition*. It is needed because for the computer an unanalyzed surface like *books* is merely a sequence of letters, with no essential difference between *books* and *skoob*, for example. By matching the unanalyzed surface with the *sur* value of

⁹A feature structure is called flat or non-recursive if it does not allow that values may themselves be feature structures.

¹⁰Cf. FoCL'99, Sect. 6.1.

¹¹Basic meanings originate as the patterns (types) needed for the agent's recognition and action procedures, cf. FoCL'99, Sect. 3.2; NLC'06, Sects. 4.2–4.4.

¹²For a more detailed discussion see NLC'06, Sect. 4.1.

a lexical proplet, all the information contained in the proplet definition becomes available to the parser.

The main tasks of automatic word form recognition are *categorization* and *lemmatization*. Categorization is provided by the *cat* value of a proplet, and needed for controlling the combinatorial interaction within the functor-argument and the coordination structure. Lemmatization is provided by the *core* value, i.e., the value of the *noun*, *verb*, or *adj* attribute, and needed for access to the words' core meaning.¹³

2 Formal Systems of Grammatical Analysis

Based on automatic word form recognition, complex expressions of natural language may be grammatically analyzed by a computer in accordance with the Fregean Principle (cf. 1.1.1). For computational linguistics, this procedure has two aspects: (i) the formal grammatical analysis of complex expressions and (ii) the construction of an automatic analysis system, called parser, which uses the formal grammatical analysis as a *declarative specification*.

A declarative specification is important for software writing in general because it formulates the abstract solution to the task at hand, specifying the *necessary* properties of the software, in contrast to accidental properties such as the choice of programming language or idiosyncrasies of the programmer.¹⁴ However, while using a formal grammar as a declarative specification is a necessary condition for implementing a transparent natural language parser, it is not sufficient to guarantee longterm success.

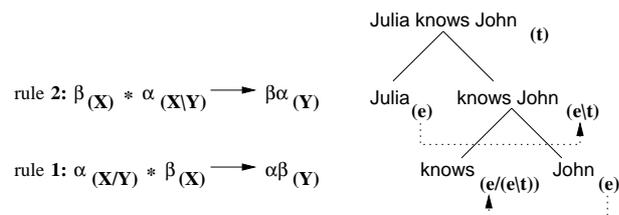
In addition to being formally explicit, the grammar must be of *low mathematical complexity* to run in real time. Furthermore, the grammar must turn out to be empirically adequate by supporting *fast upscaling* from a small system to one with an ever wider data coverage. Finally, the formal system must be *functionally complete* to support different applications of human-computer communication, whereby building a freely talking robot is the ultimate challenge.

2.1 Categorial Grammar

Designing a formal grammar raises the questions of (i) how exactly the word forms in a sentence should be combined and (ii) what the purpose of the result should be. To get an idea of how these questions have been approached in the past, let us briefly consider the most distinctive approaches proposed so far.

The historically first approach to a formal grammatical analysis is *Categorial grammar* (C-grammar) as proposed by Bar Hillel (1953), based on a formal system by Leśniewski (1929) and Ajdukiewicz (1935). Consider the following example:

2.1.1 BOTTOM-UP CATEGORIAL ANALYSIS OF Julia knows John.



The grammatical analysis is shown as the tree on the right, while the combination rules applying at the input levels are shown on the left. The derivation

starts with (i) categorized words like $\text{knows}_{(e/(e|t))}$ and $\text{John}_{(e)}$. The rules consist of the variables α and β , which match the surface of a categorized word, e.g., John , and the variables X and Y , which match a category or part of a category, e.g., $(e|t)$. Categorized words or expressions may be combined if their categories can be matched by one of the two rules. The purpose is a semantic interpretation of functor-argument structure using set theory.

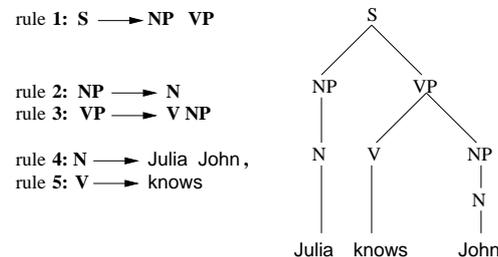
For example, $\alpha_{X/Y}$ matches the categorized word $\text{knows}_{(e/(e|t))}$, while β_X matches the categorized word $\text{John}_{(e)}$. Therefore, according to rule 1, $\text{knows}_{(e/(e|t))} * \text{John}_{(e)}$ may be combined into the expression $\text{knows John}_{(e|t)}$. The category $(e|t)$ results from the category $(e/(e|t))$ of know by canceling the first e with the category (e) of John (see dotted arrow on the right of 2.1.1).

Next, $\text{Julia}_{(e)}$ and $\text{knows John}_{(e|t)}$ are combined by rule 2 into $\text{Julia knows John}_{(t)}$, this time canceling the e in the category $(e|t)$ of knows John . The two rules differ only in that the expression with the complex category (the functor) precedes the expression with the canceling category (the argument) in rule 1, while in rule 2 the order is reversed.

2.2 Phrase Structure Grammar

The historically second approach to formal grammar is *Phrase Structure grammar* (PS-grammar), as proposed by Chomsky (1957, 1965), based on a formal system by Post (1936). Its purpose is to characterize the native speakers' innate language capability and to explain language acquisition by the child.

2.2.1 TOP-DOWN PHRASE-STRUCTURE ANALYSIS OF Julia knows John.



The analysis uses rewrite rules of the form $A \rightarrow B C$. There are non-terminal symbols like S (for start or sentence), NP (for noun phrase), VP (for verb phrase), N (for noun), and V (for verb), and terminal symbols like Julia , John , and knows . The derivation begins with the S , which rule 1 substitutes with $NP VP$. Then rule 2 substitutes NP with N , and rule 3 substitutes VP with $V NP$. Then rule 2 applies again, substituting the second NP with N . Finally the terminal rules 4 and 5 apply, replacing the nodes N and V with terminal symbols, i.e., with word surfaces.

C-grammar and PS-grammar share the assumption that the grammatical relations in natural language constitute *hierarchies*, called constituent structure (cf. FoCL'99, Sects. 8.4, 8.5). These hierarchies are based on the principle of *possible substitutions* and reflected by the associated trees in terms of the *dominance* and *precedence* relations between the nodes.

The hierarchy assumption is far-reaching because it affects derivation order: in Categorial grammar, the phrasal and clausal parts of speech must be assembled before they can be combined with each other like elementary parts of speech; in Phrase Structure grammar, the phrasal and clausal parts of speech must be derived from corresponding elementary ones, represented abstractly by non-terminal nodes.

¹³An overview of different methods of automatic word form recognition is provided in FoCL'99, Chapt. 14.

¹⁴For a more detailed discussion see NLC'06, Sect. 1.2.

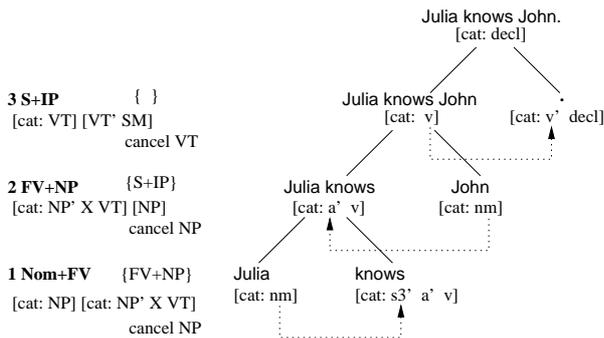
2.3 Left-Associative Grammar

According to constituent structure, it would not be correct to postulate a Phrase Structure rule like $S \rightarrow N V$ because $N V$ or the corresponding *Julia knows* is neither an elementary nor a phrasal nor a clausal part of speech. For the speaker, however, a sentence like *Julia knows John* begins by combining *Julia* and *knows* into *Julia knows*. Then *Julia knows* is combined with *John* into *Julia knows John*. Similarly for the hearer, who doesn't have to wait for the end of the sentence before interpretation can begin.

In other words, the derivation order used by the speaker and the hearer is *time-linear*, i.e., linear like time and in the direction of time, word by word, from beginning to end. This raises the question of whether there isn't a better way to express the grammatical relations than dominance and precedence. How could functor-argument and coordination structure at the elementary, phrasal, and clausal levels be integrated into a strictly time-linear derivation order?

The time-linear derivation order corresponds to the *left-associative* bracketing structure $((((a\ b)\ c)\ d)\ e)$ known from logic. It is used by Left-Associative grammar (LA-grammar), which analyzes the combination of elementary, phrasal, and clausal parts of speech by computing *possible continuations* rather than possible substitutions. LA-grammar was first developed in the context of the NEWCAT'86 parser.

2.3.1 TIME-LINEAR NEWCAT DERIVATION OF *Julia knows John*.



The bottom-up left-associative derivation always combines a sentence start with a next word into a new sentence start, until no next word is available. The completely regular structure of the tree is used to express the derivation order, not the grammatical relations – which are coded instead into the categories shown in the tree on the right and into the categorial operations of the rules on the left.

The combination of *Julia* and *knows* is based on canceling the subject valency position $s3'$ in the category of the next word. The combination of *Julia knows* and *John* is based on canceling the object valency position a' in the sentence start. And similarly for the third combination (as indicated by the dotted arrows).

Each rule consists of (i) a rule name, e.g., *Nom+FV* (for nominative plus finite verb), (ii) a rule package, e.g. $\{FV+NP\}$, containing the set of rules to be tried after a successful application of the current rule, (iii) a pattern for the sentence start, e.g., $[cat: NP]$, (iv) a pattern for the next word, e.g., $[cat: NP' X VT]$, and a set of operations, e.g., *cancel NP*. The patterns are based on variables like *NP* and *NP'*, where *NP* is restricted to noun valency fillers and *NP'* is restricted to noun valency positions.¹⁵

LA-grammar differs from Categorial grammar as shown in 2.1.1 in that (i) an LA-grammar is not

¹⁵In addition to the definition of the rules, an LA-grammar requires the definition of a set of start states, a set of final states, a set of variables and their restrictions, and a set of lexical proplets.

restricted to two rules, (ii) the LA-grammar rules have rule packages, (iii) the rule patterns indicate valency structures as lists rather than binary bracketings, (iv) there is an open number of variables with suitable restrictions and agreement conditions, and (v) the derivation has a clearly defined starting point, namely the first word. LA-grammar differs from Phrase Structure grammar in that it has a complexity hierarchy which is orthogonal to the Chomsky hierarchy (cf. TCS'92), parsing many context-free and context-sensitive languages in linear time.

The NEWCAT approach illustrated in 2.3.1 has been applied to all major functor-argument and coordination constructions of English and German. Due to its time-linear derivation structure, it proved to be well-suited for rapid upscaling. An algebraic definition of LA-grammar, constructed with the help of Dana Scott, provides a foundation based on the formal metalanguage of set theory (cf. CoL'89).

2.4 Database Semantics: Hearer Mode

Despite their differences, C-grammar 2.1.1, PS-grammar 2.2.1, and NEWCAT 2.3.1 have in common that they are *sign-oriented*. Sign-oriented approaches analyze expressions of natural language as isolated grammatical structures, but leave unanswered the central question of *What to do with them?*

For human speaker-hearers the answer is obvious: expressions of natural language are used for communication. To arrive at a similar answer in linguistics, the analysis of language expressions (theory of grammar) must be embedded into a functional model of how communicating with natural language works (theory of language).¹⁶ Today, the most straightforward scientific way to do this is the construction of an artificial cognitive agent, i.e., a talking robot with a real body¹⁷ acting in the real world (AIJ'01).

This leads from a sign-oriented to an *agent-oriented* approach, which constrains the grammatical analysis with much needed functional requirements. The most basic of these is the computational reconstruction of turn taking,¹⁸ i.e., the agent's ability to switch between the *hearer mode* and the *speaker mode*. In the hearer mode signs of natural language are interpreted as content which is stored in the agent's memory. In the speaker mode, content in the agent's memory is selectively activated and realized as corresponding surfaces in the natural language of choice.

Leaving important matters¹⁹ like the external interfaces for recognition and action, the auto- and the service channel of artificial cognitive agents, and the language and the context level aside, let us turn to three theoretical issues which any computational agent-oriented approach has to address from the start:

2.4.1 COMPUTATIONAL DESIGN DECISIONS

1. What should be the *data structure* (abstract data type) of the content items stored in memory?
2. What should be the *algorithm* to read items of content into (hearer mode) and out of (speaker mode) the agent's memory?
3. What should be the *data base schema* according to which the content items are stored in and retrieved from memory?

¹⁶Cf. NLC'06, Sect. 2.4.

¹⁷The importance of agents with a real body (instead of virtual agents) has been emphasized by emergentism (MacWhinney 2008).

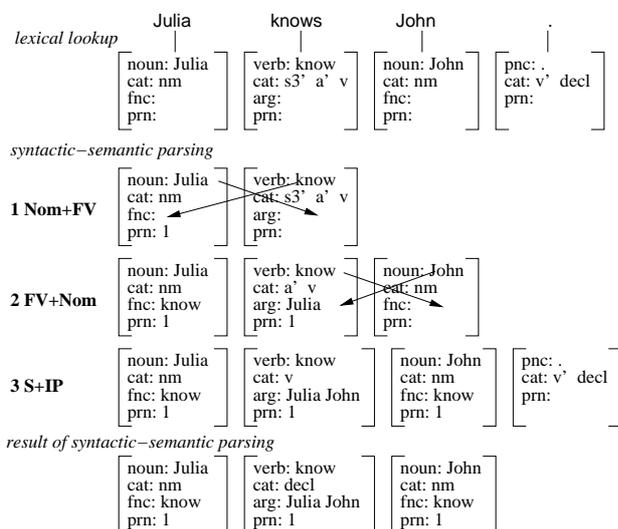
¹⁸Cf. NLC'06, Sect. 1.1, Schegloff (2007).

¹⁹For a detailed discussion see NLC', Chaps. 1 and 2.

These three must be co-designed for maximal support of the functional flow in the hearer mode, the think mode, and the speaker mode.

In Database Semantics, the data structure are proplets, i.e., flat (non-recursive) feature structures representing word forms; the algorithm is time-linear LA-grammar; and the data base schema is that of a classic network database (Elmasri and Navate 1989). As an example of the hearer mode, consider the following derivation, using the same sentence as before:

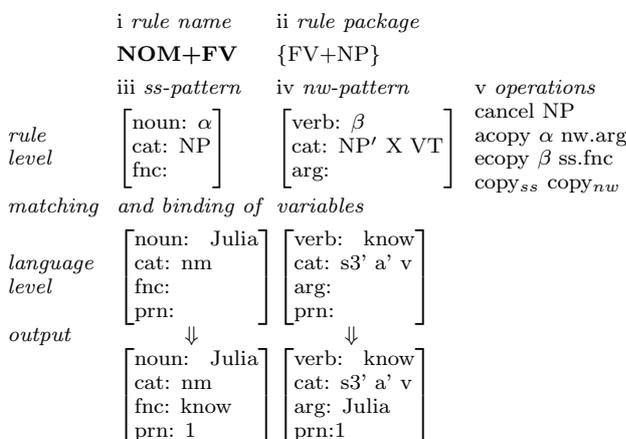
2.4.2 HEARER-MODE DERIVATION IN DBS



The derivation begins with lexical lookup of the incoming unanalyzed surfaces. The functor-argument structure is coded by copying values between proplets (indicated by the diagonal arrows). The derivation is strictly time-linear, as indicated by the stair-like addition of next word proplets.²⁰ The result of the derivation is an order-free set of proplets ready to be sorted into the database. The case assigned to the arguments is indicated in the resulting verb proplet by the order of the arg values (here arg: Julia John).

Connecting the lexical proplets in accordance with the grammatical functor-argument (and, if given, coordination) structure of an input is provided by syntactic-semantic parsing, based on a variant of LA-grammar, called *LA-hear*. Consider, for example, the first rule application in 2.4.2 (explanations in italics):

2.4.3 EXAMPLE OF LA-HEAR RULE APPLICATION



The LA-hear rule resembles the corresponding NEW-CAT rule (cf. 2.3.1) in that it consists of (i) a rule

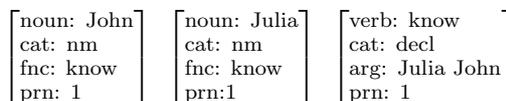
²⁰In contradistinction to the bottom-up NEWCAT derivation 2.3.1, the DBS hearer mode derivation is not represented as a tree and must be read from top to bottom.

name, (ii) a rule package, (iii) an ss-pattern, (iv) an nw-pattern, and (v) a set of operations. Moreover, the NEWCAT and the LA-hear versions of Nom+FV share the same cat features and the associated operation cancel NP.

The two versions differ, however, in that the LA-hear version has the additional features [noun: α] and [fnc:] in the ss-pattern, and [verb: β] and [arg:] in the nw-pattern. Vertical binding of the variable α to the value Julia and of β to know at the language level enables the operations acopy α nw.arg and ecopy β ss.fnc. Consequently, the output proplet *Julia* specifies its functor as know and the output proplet *know* specifies one of its arguments as *Julia*.

Furthermore, while the result of the NEWCAT derivation 2.3.1 is the whole tree (which raises the same problems for storage in and retrieval from a database as the other sign-oriented representations), the result of the LA-hear derivation 2.4.2 is an *order-free* set of proplets. This set, shown below in a sequence using the alphabetical order of the core values, represents the content coded by the natural surface:

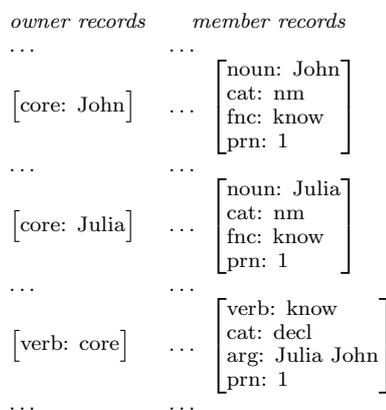
2.4.4 CONTENT OF Julia knows John.



The proplets are order-free because the grammatical relations between them are coded solely by attribute-value pairs (and not in terms of dominance and precedence in a hierarchy²¹). Therefore, they are suitable for storage and retrieval in accordance with the methods of one's database.

Given that DBS uses the database schema of a classic network database, the proplets of 2.4.4 are stored as follows:

2.4.5 STORAGE OF PROPLETS IN A WORD BANK



A network database containing proplets is called a Word Bank. The database schema consists of owner records in alphabetical vertical order, and member records stored horizontally behind their owner record. A horizontal sequence of an owner record followed by an arbitrary number of member records with the same core value is called a *token line*. The horizontal order of proplets in a token line reflects the order of their arrival, like sediment, because new incoming proplets are always stored at the end of their token line.

This method of storage is complemented by an equally simple method of retrieval: any core value provides access to all member records with the same core value by selecting the corresponding token line. Furthermore, given a core value and a prn (proposition number) value, the associated proplet may be

²¹For a detailed discussion see Hausser (2007)

retrieved by (i) going to the token line of the core value and (ii) searching through the token line for the proplet with the corresponding prn value.²²

2.5 Database Semantics: Think Mode

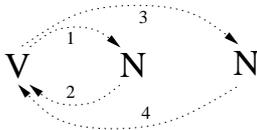
Next let us see how (i) the data structure of proplets, (ii) the algorithm of LA-grammar, and (iii) the database schema of a network database (Word Bank) serve to realize language production in the speaker mode. The task of language production is traditionally divided into *what to say* and *how to say it*.

In Database Semantics, these two aspects are viewed as (i) the *activation* of content in the agent's Word Bank and (ii) the *realization* of activated content in a natural language. The selective, autonomous activation of content is treated as a kind of thinking and based on a structural property of a Word Bank which goes beyond a classic network database.

This property is the concatenation of proplets in terms of the semantic relations of functor-argument and coordination structure. It constitutes something like a railroad system which supports a *navigation* along the semantic relations between proplets, thus selectively activating content and re-introducing a time-linear order.

For example, the *know* proplet in 2.4.4 specifies *Julia* as its first argument. Using the retrieval mechanism of the database, the navigation moves an imaginary focus point from *know* to *Julia*, then back to *know* and on to *John*, back to *know* and on to the next proposition.²³ This navigation may be shown schematically as follows:

2.5.1 NAVIGATING THROUGH A CONSTELLATION



The schema shows the constellation VNN (with V for verb and N for noun) and a navigation through the constellation, indicated by numbered arrows.

Given that any written representation of an order-free constellation of concatenated proplets must use some order, the V is written first because it specifies the connections to previous and following propositions (extrapropositional relations) as well as to the nominal arguments (intrapropositional relations). The first N is interpreted here as the subject and the second N as the object.

Navigating through a constellation of concatenated proplets is constrained by the following conditions:

2.5.2 CONDITIONS ON A NAVIGATION

1. A navigation is a *shortest* route to traverse
2. *all* proplets in the constellation such that
3. each successor proplet is *accessible* from the current proplet.

For example, after navigating from the V to the first N in a VNN constellation, it is necessary to return

²²Please note that the structuring of a Word Bank indicated in 2.4.4 is purely conceptual and does in no way restrict the storage locations in the actual implementation of the database.

²³Because the concatenated propositions in a Word Bank usually offer alternative continuations, there is the need to build an autonomous control structure for choosing between alternatives. This is a major task (cf. Hausser 2002, Hausser in print) which exceeds the limits of this paper. We concentrate here on the question of how a given navigation should be realized in a natural language.

to the V to get the continuation values for accessing the second N (object noun). After navigating to the second N, it is necessary to return to the V in order to get the continuation values for accessing the next proposition.

A navigation through a constellation is driven by the rules of an LA-think grammar. For example, traversal 1 in the constellation 2.5.1 is based on the following application of the rule VNs to proplets in the Word Bank 2.4.4 (explanations in italics):

2.5.3 EXAMPLE OF LA-THINK RULE APPLICATION

	i rule name VNs	ii rule package {NVs}	v operations
	iii ss pattern	iv nw pattern	
rule level	$\left[\begin{array}{l} \text{verb: } \beta \\ \text{arg: !X } \alpha \text{ Y} \\ \text{prn: k} \end{array} \right]$	$\left[\begin{array}{l} \text{noun: } \alpha \\ \text{mdr: Z} \\ \text{fnc: } \beta \\ \text{prn: k} \end{array} \right]$	output position nw
<i>matching and binding variables</i>			
Word Bank level	$\left[\begin{array}{l} \text{verb: know} \\ \text{cat: decl} \\ \text{arg: Julia John} \\ \text{prn: 1} \end{array} \right]$	$\left[\begin{array}{l} \text{noun: Julia} \\ \text{cat: nm} \\ \text{fnc: know} \\ \text{prn: 1} \end{array} \right]$	
		↓	
output		$\left[\begin{array}{l} \text{noun: Julia} \\ \text{cat: nm} \\ \text{fnc: know} \\ \text{prn: 1} \end{array} \right]$	

The explanations i–v show that this LA-think rule consists of the same components as the NEWCAT rules 2.3.1 and the LA-hear rule 2.4.3. The rule name VNs indicates that the rule moves the navigation from a V to an N and stays there.

By vertically binding the variable β in the ss-pattern to the value *know*, the variable α to the value *Julia*, and the variable k to the value 1 at the Word Bank level, the retrieval mechanism of the database has the information needed for navigating to (touch, activate, traverse) the nw-proplet *Julia*. This output serves as the ss-proplet of the next LA-think rule application. Navigation step 2 in 2.5.1 returns to the V, based on the rule NVs (specified in the rule package of VNs). NVs resembles VNs except that the patterns for ss and nw are reversed.

The basic principle of navigating through the content of a Word Bank, using the semantic relations between proplets as a railroad system, is as simple as it is efficient.²⁴ Yet it has enormous descriptive potential because the rules may have more operations than the one in 2.5.3. For language production, for example, the LA-think rules may be turned into LA-speak rules by adding operations like *lex-n* [noun: α], which serve to realize specific surfaces. Other operations are used for inferences, as in the reconstruction of *modus ponens* in NLC'06, Sect. 5.3, and the answering of questions, as shown in NLC'06, Sect. 5.1.

2.6 Database Semantics: Speaker Mode

Having outlined the *what to say* aspect of language production, let us turn to the aspect of *how to say it*. The time-linear sequence of proplets activated by a navigation like 2.5.1 contributes the following structural properties to language production:

2.6.1 CONTRIBUTIONS OF NAVIGATION

1. core values
2. parts of speech

²⁴The complexity of a navigation is linear as long as it does not branch, the number of operations per rule is below a finite upper bound, and the operations are simple.

3. semantic relations
4. traversal sequence
5. ingredients of perspective

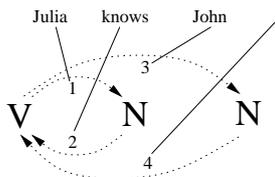
In order to map these structural details into a corresponding language-dependent surface, the LA-think rules activating the constellation must be turned into LA-speak rules by adding operations for handling the following properties of the language-dependent surface:

2.6.2 CONTRIBUTIONS OF GRAMMAR

1. language-dependent word order (defined on top of the traversal sequence)
2. language-dependent function word precipitation (utilizing proplet features)
3. selection of word forms (based on proplet features and rules of agreement)
4. lexical selection (driven by the core values of the proplets traversed)

These language-dependent properties of grammar are tightly interconnected, riding piggyback on the LA-think navigation rules.²⁵ For example, the activation of constellation 2.5.1 may be used to produce the English surface *Julia knows John.* as follows:

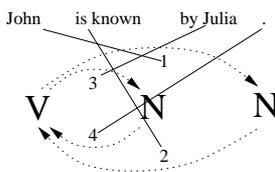
2.6.3 REALIZING A TRANSITIVE SURFACE



Transition 1 is used for realizing *Julia*, 2 for *knows*, 3 for *John*, and 4 for the full stop.

An alternative way to express the same content is passive.²⁶ It is based on an alternative activation order which is in concord with the conditions 2.5.2:

2.6.4 REALIZING A PASSIVE SURFACE



As indicated by the traversal numbers, the navigation first traverses the object and then the subject.²⁷ The realization of the passive surface in English provides a change of word order, but it also requires word forms different from the active as well as function word support.

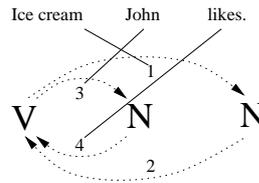
²⁵In earlier work, LA-speak was treated as a separate component, which required a constant switching between LA-think and LA-speak. As shown by NLC'06, Chapt. 13, this turned out to be rather complicated. The current approach narrows the gap between LA-think and LA-speak by defining LA-speak as a variant of LA-think. Now the only difference between the two are additional operations for lexical realization in LA-speak, which provides for a much simpler solution.

²⁶As pointed out by Givón (1997), one of the main functions of passive is the possibility of leaving the deep subject (agens) unspecified, as in *The car had been stolen.*

²⁷SVO languages like English or German and SOV languages like Korean or Japanese use the navigation order of 2.6.3 as the unmarked case, whereas VOS languages like Fijian or Malagasi and OSV languages like Xavante or Warao use 2.6.4.

Other word orders in English are illustrated by *Ice cream John likes.*, which may be realized from the navigation of 2.6.4 using the realization steps 134:

2.6.5 REALIZING A TOPICALIZED OBJECT



Furthermore, in spoken language and in poetry the word order restrictions of a language are often handled rather freely. In Database Semantics, all such phenomena relating to word order and function word precipitation are accommodated by (i) the choice between different ways to traverse the constellation in question (for example, 2.6.3 vs. 2.6.4) and (ii) the choice of when to realize a surface during the traversal (for example, 2.6.4 vs. 2.6.5).

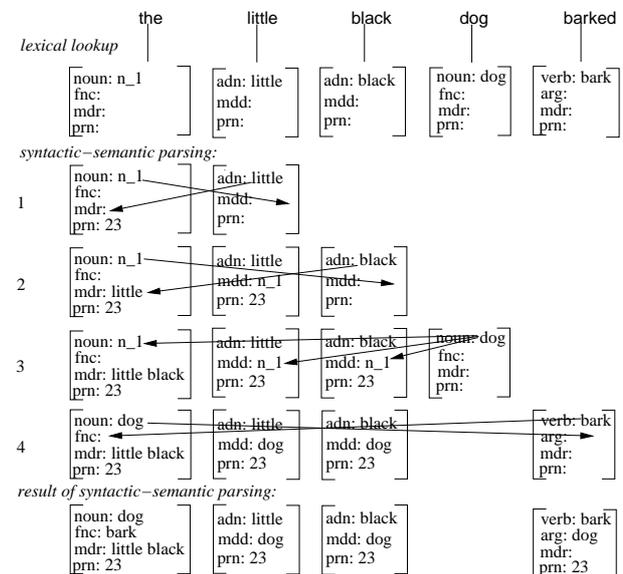
3 Treating the Phrasal and Clausal Levels

So far, the examples used have been functor-argument structures at the elementary level, without any modifiers and without an explicit treatment of function words. As an example at the phrasal level let us consider the sentence *The little black dog barked.*

3.1 Interpretation in the Hearer Mode

At the phrasal level, the hearer mode mapping of natural language surfaces into content involves *function word absorption* and *modification*, as illustrated by the following derivation:

3.1.1 ADNOMINAL MODIFICATION (HEARER M.)



The determiner *the* is treated lexically as a function word with the core value *n_1*, called a substitution value. In line 1 of syntactic-semantic parsing, the value *little* is copied into the *mdr* (modifier) slot of *the* and the value *n_1* is copied into the *mdd* (modified) slot of *little*. Similarly in line 2, in which *black* and *n_1* are cross-copied, as indicated by the diagonal arrows.

In line 3, all instances of *n_1* are simultaneously replaced by the core value of the *dog* proplet, which is then discarded (as indicated by the gap in line 4). In other words, the determiner and the noun proplets are being fused by absorbing the core value of the

noun into the determiner proplet,²⁸ and providing the adnominal modifiers with suitable *mdd* values.

The result of the hearer mode derivation is an order-free set of proplets, shown below using the order of a VNAA constellation (cf. 3.2.1 below):

3.1.2 CONTENT OF The little black dog barked

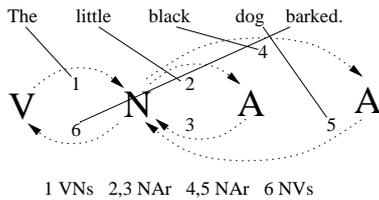
$\left[\begin{array}{l} \text{verb: bark} \\ \text{cat: decl} \\ \text{arg: dog} \\ \text{prn: 23} \end{array} \right]$	$\left[\begin{array}{l} \text{noun: dog} \\ \text{cat: def sg} \\ \text{fnc: bark} \\ \text{mdr: little black} \\ \text{prn: 23} \end{array} \right]$	$\left[\begin{array}{l} \text{adj: little} \\ \text{cat: adn} \\ \text{mdd: dog} \\ \text{prn: 23} \end{array} \right]$	$\left[\begin{array}{l} \text{adj: black} \\ \text{cat: adn} \\ \text{mdd: dog} \\ \text{prn: 23} \end{array} \right]$
--	--	--	---

In DBS, the phrasal nature of the noun **the little black dog** is not expressed by a single node dominating other nodes, as it would be in PS-grammar (cf. 2.2.1) or C-grammar (cf. 2.1.1). Instead of a hierarchy, it is treated as a constellation of bidirectionally concatenated proplets representing the modified and its modifiers. The contribution of the function word **the** appears as the *cat*²⁹ value *def* in the *dog* proplet.

3.2 Production in the Speaker Mode

Producing the English sentence **the little black dog barked.** from a constellation requires (i) realization of the determiner absorbed during interpretation and (ii) the correct positioning of the elementary adnominal modifiers between the determiner and their modified. This is shown by the following navigation with associated realization:

3.2.1 ADNOMINAL MODIFICATION (SPEAKER M.)



In this VNAA constellation, the V precedes the N for the reasons explained in connection with 2.5.1. The N precedes the As because they are accessible only from the N (cf. transitions 2,3 and 4,5).

The English surfaces are realized from the goal proplet of a transition by means of lex functions such as *lex-d* (for determiners), *lex-nn* (for elementary common nouns) and *lex-n* (for determiner noun sequences without intervening adnominal adjectives), as defined below for the regular cases:

3.2.2 LEXICALIZATION FUNCTIONS

1. *lex-d*

If one of the following patterns matches an N proplet, then *lex-d* applied to this proplet produces the associated surface:

<i>pattern</i>	<i>surface</i>	<i>pattern</i>	<i>surface</i>
$\left[\begin{array}{l} \text{noun: } \alpha \\ \text{sem: indef sg} \end{array} \right]$	<i>a(n)</i>	$\left[\begin{array}{l} \text{noun: } \alpha \\ \text{cat: snp} \\ \text{sem: pl exh} \end{array} \right]$	<i>every</i>
$\left[\begin{array}{l} \text{noun: } \alpha \\ \text{sem: sel} \end{array} \right]$	<i>some</i>	$\left[\begin{array}{l} \text{noun: } \alpha \\ \text{cat: pnp} \\ \text{sem: pl exh} \end{array} \right]$	<i>all</i>
$\left[\begin{array}{l} \text{noun: } \alpha \\ \text{sem: def X} \end{array} \right]$	<i>the</i>		

²⁸The relevant content of a function word may also be absorbed into a content word. An example are punctuation signs, which are absorbed into the verb proplet, as illustrated in 2.4.2.

²⁹In 3.1.1, the *cat* features as well as the *sem*, *nc*, and *pc*, features (cf. 1.5.1) are omitted for simplicity. For a complete definition of the LA-hear grammar for elementary and phrasal intrapositional functor-argument structure and extrapositional coordination see NLC'06, Chapt. 13.

2. *lex-nn*

If $\left[\begin{array}{l} \text{noun: } \alpha \\ \text{cat: snp} \end{array} \right]$ matches an N proplet, then *lex-nn*[noun: α] = α .

If $\left[\begin{array}{l} \text{noun: } \alpha \\ \text{cat: pnp} \end{array} \right]$ matches an N proplet, then *lex-nn* [noun: α] = $\alpha+s$.

3. *lex-n*

If one of the following patterns matches an N proplet, then *lex-n* applied to this proplet produces the associated surface:

<i>pattern</i>	<i>surface</i>	<i>pattern</i>	<i>surface</i>
$\left[\begin{array}{l} \text{noun: } \alpha \\ \text{cat: snp} \\ \text{sem: indef sg} \end{array} \right]$	<i>a(n) \alpha</i>	$\left[\begin{array}{l} \text{noun: } \alpha \\ \text{cat: snp} \\ \text{sem: pl exh} \end{array} \right]$	<i>every \alpha</i>
$\left[\begin{array}{l} \text{noun: } \alpha \\ \text{cat: snp} \\ \text{sem: sel} \end{array} \right]$	<i>some \alpha</i>	$\left[\begin{array}{l} \text{noun: } \alpha \\ \text{cat: pnp} \\ \text{sem: sel} \end{array} \right]$	<i>some \alpha+s</i>
$\left[\begin{array}{l} \text{noun: } \alpha \\ \text{cat: pnp} \\ \text{sem: pl exh} \end{array} \right]$	<i>all \alpha+s</i>	$\left[\begin{array}{l} \text{noun: } \alpha \\ \text{cat: snp} \\ \text{sem: def X} \end{array} \right]$	<i>the \alpha</i>
$\left[\begin{array}{l} \text{noun: } \alpha \\ \text{cat: pnp} \\ \text{sem: def X} \end{array} \right]$	<i>the \alpha+s</i>		

Other lex functions are *lex-v* for the realization of a finite verb form, *lex-adn* for adnominal adjectives, and *lex-p* for the punctuation signs.

The lex functions are called by the rules of an LA-speak grammar. Consider the following definition, which replaces the definition of LA-speak.2 in NLC'06, Sect. 14.2:

3.2.3 FORMAL DEFINITION OF LA-SPEAK.E2

$\mathbf{ST}_S =_{def} \{ \{ \text{verb: } \alpha \} \{ 1 \text{ VN's} \} \}$

VNs {2 NVs, 3 NAr}

$\left[\begin{array}{l} \text{verb: } \beta \\ \text{arg: !X } \alpha \text{ Y} \\ \text{prn: i} \end{array} \right]$ $\left[\begin{array}{l} \text{noun: } \alpha \\ \text{mdr: Z} \\ \text{fnc: } \beta \\ \text{prn: i} \end{array} \right]$ if *adn* $\notin Z$, *lex-n* [noun: α]
if *adn* $\in Z$, *lex-d* [noun: α]
(where *adn* is an elementary adnominal)

NVs {4 VN's, 5 VV's}

$\left[\begin{array}{l} \text{noun: } \alpha \\ \text{fnc: } \beta \\ \text{prn: i} \end{array} \right]$ $\left[\begin{array}{l} \text{verb: } \beta \\ \text{arg: X! } \alpha \text{ Y} \\ \text{cat: VT} \\ \text{prn: i} \end{array} \right]$ mark α in β -arg
if X = NIL, *lex-fv* [verb: β]
if Y = NIL, *lex-p* [verb: β]

NAr {6 NAr, 7 NVs}

$\left[\begin{array}{l} \text{noun: } \alpha \\ \text{mdr: !X } \beta \text{ Y} \\ \text{prn: i} \end{array} \right]$ $\left[\begin{array}{l} \text{adj: } \beta \\ \text{mdd: } \alpha \\ \text{prn: i} \end{array} \right]$ mark β in α -mdr
lex-a [adj: β]
if *adn* $\notin Y$, *lex-nn* [noun: α]

VVs {8 VN's}

$\left[\begin{array}{l} \text{verb: } \alpha \\ \text{nc: j } \beta \\ \text{prn: i} \end{array} \right]$ $\left[\begin{array}{l} \text{verb: } \beta \\ \text{pc: i } \alpha \\ \text{prn: j} \end{array} \right]$

$\mathbf{ST}_F =_{def} \{ (\left[\begin{array}{l} \text{verb: } \beta \\ \text{arg: X!} \end{array} \right] \text{rPNVs}) \}$

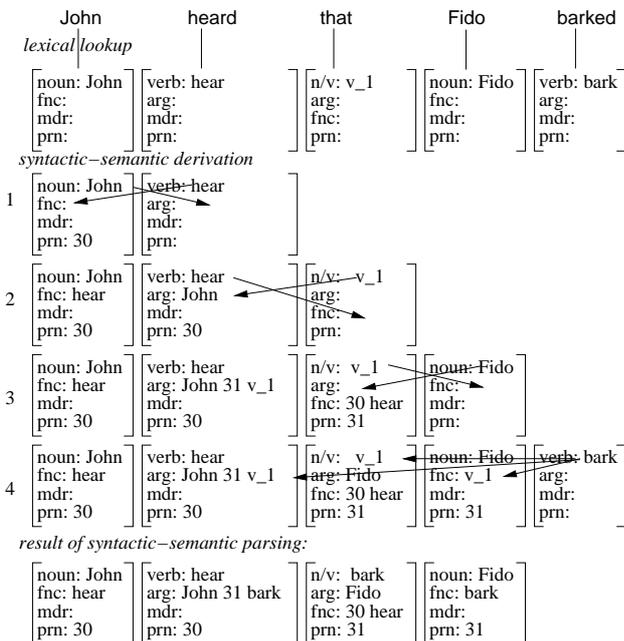
The rules of this grammar have two kinds of suffixes, *s* (for staying at the second proplet) and *r* (for returning to the first proplet). This way of specifying the output position in the rule name makes the use of the corresponding operations redundant (compare 2.5.3).

As indicated at the bottom of 3.2.1, rule VN's executes transition 1, thereby realizing the determiner using *lex-d*. Rule NAr executes transition 2, realizes the adnominal *little* with *lex-a*, and returns to the N (transition 3). Then NAr applies again (cf. rule package), executes transition 4, realizes *black*, and returns to the N (transition 5), thereby realizing *dog* with *lex-nn* (because *adn* $\notin Y$). Finally, NVs executes transition 6 and realizes *barked.* using *lex-v* and *lex-p*.

3.3 Interpretation at the Clausal Level

It remains to show the handling of clausal constructions. As an example, consider the following derivation of John heard that Fido barked.

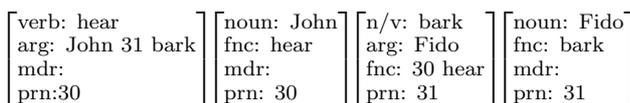
3.3.1 OBJECT CLAUSE (HEARER MODE)



In the result, the proplets of the main clause and of the subordinate clause are distinguished by different prn values, 30 and 31. The connection between the main verb and its sentential object is coded by the arg value 31 bark of *hear*. The inverse connection is coded by the fnc value 30 hear of *bark*.

The result of the hearer mode derivation is an order-free set of proplets, shown below in the order of a VNVⁿN constellation (cf. 3.4.1 below):

3.3.2 CONTENT OF John heard that Fido barked.

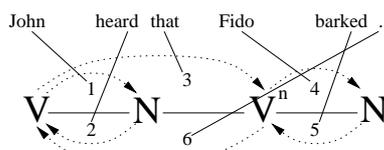


The double function of the *bark* proplet as the functor of the subordinate clause and as the argument of the main clause is indicated by its core attribute n/v (for nominal argument in the form of a subordinate verb).

3.4 Production at the Clausal Level

Turning to production, consider a schematic characterization of the proplet sequence 3.1.2 as a VNVⁿN constellation with an associated realization; Vⁿ represents the proplet with the n/v core attribute.

3.4.1 OBJECT CLAUSE (SPEAKER MODE)



1 VNs 2 NVs 3 VV^{ns} 4 VⁿNs 5 NV^{ns} 6 VⁿVs

The LA-speak.3 grammar for realizing English functor-argument constructions with clausal arguments and clausal modifiers (presented systematically in NLC'06, Chapt. 7) has a total of 15 rules (including the 4 rules of LA-speak.2 defined in 3.2.3).

4 Conclusion

Is it really necessary to analyze the grammatical relations as a hierarchy, such that a functor (verb) must dominate its arguments (nouns), for example? Apart from well-known problems with discontinuous elements and the handling of coordination, the assumption of a hierarchical structure stands in the way of a time-linear interpretation in the hearer mode, a time-linear navigation in the think mode, and a time-linear production in the speaker mode.

For a computational model of these operations, Database Semantics treats the semantic relations of functor-argument and coordination structures instead as constellations of order-free proplets connected by attribute-value pairs. It has been shown here that this formal method easily accommodates a traditional approach to grammar based on nouns, verbs, and adjectives at the elementary, phrasal, and clausal level.

5 References

AIJ'01 = Hausser, R. (2001) "Database Semantics for natural language," *Artificial Intelligence*, 130.1:27-74

Ajdukiewicz, K. (1935) "Die syntaktische Konnexität," *Studia Philosophica*, 1:1-27

Bar Hillel, J. (1953) "A quasi-arithmetical notation for syntactic description," *Language*, 29.1:47-63

Benner, M. L. (2008) *Towson University online Writing Support*, <http://www.towson.edu/ows/>

Chomsky, N. (1957) *Syntactic Structures*, The Hague: Mouton

Chomsky, N. (1965) *Aspects of the Theory of Syntax*, Cambridge, Mass.: MIT Press

CoL'89 = Hausser, R. (1989) *Computation of Language, Symbolic Computation: Artificial Intelligence*, Berlin Heidelberg New York: Springer

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

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

Givón, T. (1997) *Grammatical Relations: A Functionalist Perspective*. Amsterdam: John Benjamins

Hausser, R. (2002) "Autonomous Control Structure for Artificial Cognitive Agents," in H. Kangassalo et al. (eds) *Information Modeling and Knowledge Bases XIII*, Amsterdam: IOS Press Ohmsha

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

Leśniewski, S. (1929) "Grundzüge eines neuen Systems der Grundlagen der Mathematik," Warsaw: *Fundamenta Mathematicae* 14:1-81

MacWhinney, B. (2008) "How Mental Models Encode Embodied Linguistic Perspective," in L. Klatzky et al. (eds.) *Embodiment, Ego-Space, and Action*, New York: Psychology Press

NEWCAT'86 = Hausser, R. (1986) *NEWCAT: Parsing Natural Language Using Left-Associative Grammar*, Lecture Notes in Computer Science 231, Berlin Heidelberg New York: Springer

NLC'06 = Hausser, R. (2006) *A Computational Model of Natural Language Communication*, Berlin Heidelberg New York: Springer

Post, E. (1936) "Finite Combinatory Processes — Formulation I," *Journal of Symbolic Logic*, 1:103-105

Schegloff, E. (2007) *Sequence Organization in Interaction*, New York: Cambridge Univ. Press

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