Preface

In the preface to the first edition of *The Principia*,[1] Sir Isaac Newton distinguished two aspects of mechanics: theoretical and practical. The theoretical aspect, called rational by Newton, consists in accurate demonstration. The practical aspect includes all the manual arts. What would be the result of applying a corresponding distinction to the current state of linguistics?

Instead of first praising the importance of our field – as Newton would – let us go straight to the questions at hand: What is theoretical linguistics and what is practical linguistics? Practical linguistics is instantiated by such tasks as speech recognition, desktop publishing, word processing, machine translation, content extraction, classification, querying the Internet, automatic tutoring, dialogue systems, and all the other applications involving natural language. They have generated a huge demand for practical linguistic methods.

Compared to the users' needs and expectations, however, the results leave much to be desired. Today, the most successful applications in practical linguistics are based on the methods of statistics and metadata mark-up. These are *smart solutions*,[2] which try to get by without a general theory of how communicating with natural language works. Instead they aim to maximally exploit the special properties and natural limitations of each application or kind of application.

Now consider practical mechanics: It is instantiated by tasks ranging from accurately predicting the tides, to predicting the future positions of the planets, to aiming cannon balls, to landing on the moon. These applications have created an equally large demand for applied methods as in linguistics.

In contrast to linguistics, however, the field of mechanics was able to satisfy any such demands far beyond the users' imagination. This was possible because Newton's general theory can be translated into the specific applications while maintaining compatibility with traditional craft skills. Each translation is hard work and requires theoretical knowledge as well as practical experience, but the results are nothing but spectacular.

The example of Newton's mechanics leads naturally to the question: Can we do the same in linguistics? Can we conceive a new framework suitable to fulfill all the

[1] Orginal Latin title: *Philosophiæ Naturalis Principia Mathematica* (1687), complete English title *The Principia: Mathematical Principles of Natural Philosophy.*

[2] Cf. FoCL, Section 2.3. The alternative is a solid solution.

wide-ranging demands of the users by simply translating the linguistic theory into the limited and specialized contexts of various practical applications? This question may be taken as a worthy challenge to our basic research.

As a first step towards a complete, general linguistic framework, let us reconstruct the cognitive 'mechanics' of natural language communication between humans. The theory, called *Database Semantics*[3] (DBS) is presented here as the declarative specification of a talking robot. It is in the nature of our project that its potential to improve practical applications correlates directly with its relative success in adequately modeling human cognition.[4]

The declarative specification of a talking robot must be designed as a *functional* model, which effectively realizes the mechanism of natural language communication. To ensure *completeness*, the model must take the language-based interaction between humans as its prototype. The model's functionality and data coverage must be *verified* automatically by an efficient implementation as a running computer program. This combination of functionality, completeness, and verifiability constitutes the best scientific basis for the long-term success of *upscaling* the model.

The resulting system is able to serve in all the practical applications involving natural language communication. In most cases it is sufficient to simply reduce the functionality and the data coverage of the model to fit the demands of the application at hand. For example, when using the cognition of the talking robot for building an automatic dialogue system used over the phone, there is no need for artificial vision, manipulation, or locomotion.[5]

Other applications, notably machine translation, not only allow a reduction, but also require an extension of the theory. For such extensions, however, Database Semantics provides a solid basis, given that it models monolingual communication, including monolingual language understanding.

Furthermore, any application-independent (theoretical) improvements regarding the data coverage of the lexicon, of automatic word form recognition, of syntactic–semantic parsing, of absolute and episodic world knowledge, of inferencing, etc., may directly benefit existing practical applications simply by routinely replacing their components with improved versions. This is possible because the theory provides functionally motivated modules with clearly defined interfaces.

The following pages aim at presenting Database Semantics as directly and simply as possible. Intended audiences are graduate students, researchers, and software engi-

---

[3] As the name of a specific scientific theory, the term Database Semantics is written with initial capital letters. This use is distinct from referrring to generic issues, for example semantic constraints on databases (cf. Bertossi, Katona, Schewe, and Thalheim (eds.) 2003).

[4] Like any basic science with practical ramifications, a computational reconstruction of natural language communcation raises the threat of possible misuse. This must be curtailed by developing responsible guidelines for clearly defined laws to protect privacy and intellectual property while maintaining academic liberty, access to information, and freedom of discourse.

[5] Similar reductions apply to such applications as automatic grammar checking, content extraction, indexing to improve recall and precision of Internet querying, or supporting automatic speech recognition.

neers in linguistics and natural language processing. The text may also be of interest to scholars in philosophy of language, cognitive psychology, and artificial intelligence.

As background literature for readers who are new to computational linguistics in general and Database Semantics in particular, *Foundations of Computational Linguistics* (1999, 2nd ed. 2001) is recommended. FoCL'99 is a textbook that systematically describes the traditional components of grammar, compares a wide range of different linguistic approaches in their historical settings, and develops the SLIM theory of language, which is also used here.

A complementary effort in cognitive psychology is the ACT-R theory by Anderson (cf. Anderson and Lebiere 1998). Like Database Semantics, ACT-R is essentially symbol-based rather than statistical, and uses computational modeling as the method of verification. However, ACT-R focusses on memory, learning, and problem solving, while Database Semantics concentrates on modeling the speaker and the hearer mode in natural language communication.

ACKNOWLEDGEMENTS

This book evolved in the context of the laboratory of Computational Linguistics of the University Erlangen–Nürnberg (CLUE). I am grateful to the members of my team (in alphabetical order), Matthias Bethke, Johannes Handl, Besim Kabashi, and Jörg Kapfer, who discussed the technical and conceptual issues of the theory and the implementation at great length and in great depth, and contributed many ideas. I am also grateful to my students, especially Arkadius Kycia, who programmed the first Java™ implementation of DBS.1 and DBS.2 in the speaker, the think, and the hearer mode. I would like to thank Brian MacWhinney (Carnegie Mellon University, Pittsburgh) and Haitao Liu (Communications University of China, Bejing) for providing detailed comments on an earlier stage of the manuscript. Mike Daly (Dallas) proofread the manuscript and made many valuable suggestions. I am indebted to Marie Hučinova (Charles University, Prague), Vladimir Petroff (Northeastern University, Boston), Kiyong Lee (Korea University, Seoul), and a team of copy editors at Springer for numerous improvements in the final phase of the manuscript. All remaining mistakes are mine.

February 2006 Roland Hausser
Erlangen–Nürnberg

ABBREVIATIONS REFERRING TO PREVIOUS WORK

SCG'84	= Hausser, R. (1984) *Surface Compositional Grammar*, pp. 274, München: Wilhelm Fink Verlag
NEWCAT'86	= Hausser, R. (1986) *NEWCAT: Parsing Natural Language Using Left-Associative Grammar*, Lecture Notes in Computer Science 231, pp. 540, Berlin Heidelberg New York: Springer
CoL'89	= Hausser, R. (1989) *Computation of Language, An Essay on Syntax, Semantics, and Pragmatics in Natural Man-Machine Communication*, Symbolic Computation: Artificial Intelligence, pp. 425, Berlin Heidelberg New York: Springer
TCS'92	= Hausser, R. (1992) "Complexity in Left-Associative Grammar," *Theoretical Computer Science*, 106.2:283-308, Amsterdam: Elsevier
FoCL'99	= Hausser, R. (1999/2001) *Foundations of Computational Linguistics, Human–Computer Communication in Natural Language, 2nd ed.*, pp. 578, Berlin Heidelberg New York: Springer
AIJ'01	= Hausser, R. (2001) "Database Semantics for natural language." *Artificial Intelligence*, 130.1:27–74, Amsterdam: Elsevier
L&I'05	= Hausser, R. (2005) "Memory-Based pattern completion in Database Semantics," *Language and Information*, 9.1:69–92, Seoul: Korean Society for Language and Information

Contents

Appendices

Introduction

I. Basic assumptions

A computational model of natural language communication cannot be limited to the grammatical analysis of the language signs. Instead it must start with the general recognition and action procedures of the cognitive agents, treating language interpretation and production as special cases.

Recognition and action are based on the external interfaces of the cognitive agent's body, which contains a database for storing content. Agents without language have only one level of cognition, called the context level. Agents with language have two levels of cognition: the context level and the language level. The connection between language and the world, i.e., reference, is established solely by the cognitive procedures of the agent. Reference is based (i) on the external interfaces, and (ii) on relating the cognitive levels of language and context using pattern matching.[1]

Database Semantics (DBS) models the behavior of natural agents, including language communication, by automatically (a) reading propositional content resulting from recognition into the agent's database and (b) reading content out of the agent's database resulting in action. Recognition and action are (c) related by a control structure based on reasoning which results in sensible (meaningful, rational, successful) conduct.

II. Components of a cognitive agent

At the most abstract level, cognitive agents consist of three basic components. These are (i) the external interfaces, (ii) the database, and (iii) the algorithm.[2] They use a common format, called the data structure,[3] for representing and processing content.

[1] Autonomy from the metalanguage. See FoCL'99, pp. 382–383.

[2] These components correspond roughly to those of a von Neumann machine (vNm): The external interfaces represent the vNm input-output device, the database corresponds to the vNm memory, and the algorithm performs the functions of the vNm arithmetic-logic unit. For a comparison of standard computers, robots, and virtual reality machines see FoCL'99, p. 16.

[3] The term "data structure" is closely related in meaning to the term "data type." Even though there has been some argument that the format in question should be called an abstract data type rather than a data structure, the latter term is preferred here to avoid confusion with the classic type/token distinction (cf. Sect. 4.2). It is for the same reason that we use the term "kind of sign" rather than "sign type" (cf. Sect. 2.6), "kind of sentence" rather than "sentence type," "kind of word" rather than "word type," "kind of coordination" rather than "coordination type," etc.

The external interfaces are needed by the agent for recognition and action. Recognition is based, for example, on eyes to see, and ears to hear. Action is based, for example, on a mouth to talk, hands to manipulate, and legs to walk. Without them the agent would not be able to tell us what it perceives and to do what we tell it to do.

The agent's database is needed for the storage and retrieval of content provided by the interfaces. Without it, the agent would not be able to determine whether or not it has seen an object before, it could not remember the words of language and their meaning, and it would be limited to reflexes connecting input and output directly.

The algorithm is needed to connect the interfaces and the database (i) for reading content provided by recognition into the database, and (ii) for reading content out of the database into action. Also, the algorithm must (iii) process the content in the database for determining goals, planning actions, and deriving generalizations.

In the cognition of natural agents, the external interfaces, the data structure, and the algorithm interact very closely. Therefore, in a computational model of natural cognition they must be codesigned within a joint functional cycle. The three basic components may be simple initially, but they must be *general* and *functionally integrated* into a coherent framework from the outset.

III. Treating natural language

A model of natural language communication requires the traditional components of grammar, i.e., the language-specific *lexicon* and the language-specific rules of *morphology*, *syntax*, and *semantics*. During communication, these components must cooperate in (i) the hearer mode, (ii) the think mode, and (iii) the speaker mode.

In the hearer mode, the external interfaces provide the input, consisting of language signs. The algorithm parses the signs into a representation of content which is stored in the database. The parsing of the signs is based on a system of automatic word form recognition and a system of automatic syntactic–semantic analysis.

In the think mode, the algorithm is used for autonomously navigating through the database, thus selectively activating content. This general method of navigation is also used for deriving inferences which relate the current input and the content stored in the database to derive action.

In the speaker mode, the activation of content and the derivation of inferences is used for the conceptualization of language production, i.e., choosing what to say. The production of language from activated content requires the selection of language-dependent word form surfaces, and the handling of word order and agreement.

IV. Different degrees of detail

In the following chapters, some components of DBS are worked out in great detail, while others are only sketched in terms of their input, their function, and their output. This is unavoidable because of the magnitude of the task, its interdisciplinary nature, and the fact that some technologies are more easily available than others.

For example, realizing Database Semantics as the prototype of an actual robot with external interfaces for recognition and action, i.e., artificial vision, speech recognition, robotic manipulation, and robotic locomotion, was practically out of reach. This is regrettable because the content in the database is built from concepts which are "perceptually grounded" in the agents' recognition and action procedures (Roy 2003).

While the external interfaces of the artificial agent are described here at a high level of abstraction, the algorithm and the data structure are worked out not only in principle, but are developed as "fragments," that model the hearer, the think, and the speaker mode using concrete examples. These fragments are defined as explicit rule systems and are verified by means of a concomitant implementation in Java™.

V. Available systems and approaches

Today many kinds of parsers are available. Some are based on statistical methods, such as the Chunk Parser (Abney 1991; Déjean 1998; Vergne and Giguet 1998), the Brill Tagger and Parser (Brill 1993, 1994), and the Head-Driven Parser (Collins 1999; Charniak 2001). Others are based on the rules of a Phrase Structure Grammar such as the Earley Algorithm (Earley 1970), the Chart Parser (Kay 1980; Pereira and Shieber 1987), the CYK Parser (Cocke and Schwartz 1970; Younger 1967; Kasami 1965), and the Tomita Parser (Tomita 1986).

Also, there are many theories of syntax. Some are based on Categorial Grammar (Leśniewski 1929; Ajdukiewicz 1935; Bar-Hillel 1964). Related to Categorial Grammar is the approach of Valency Theory (Tesnière 1959; Herbst 1999; Ágel 2000; Herbst et al. 2004) and Dependency Grammar (Mel'čuk 1988; Hudson 1991; Hellwig 2003). Others are based on Phrase Structure Grammar (Post 1936; Chomsky 1957), for example Generalized Phrase Structure Grammar (GPSG, Gazdar et al. 1985), Lexical Functional Grammar (LFG, Bresnan 1982, 2001), Head-Driven Phrase Structure Grammar (HPSG, Pollard and Sag 1987, 1994), and Construction Grammar (Östman and Fried 2004; Fillmore et al. forthcoming).

Similarly, there are many approaches to semantic analysis. Some are based on Model Theory (Tarski 1935, 1944; Montague 1974), others on Speech Act Theory (Austin 1962; Grice 1957, 1965; Searle 1969), or Semantic Networks (Quillian 1968; Sowa 1984, 2000). In addition, there is Rhetorical Structure Theory (RST, Mann and Thompson 1993), Text Linguistics (Halliday and Hasan 1976; Beaugrande and Dressler 1981) as well as different approaches to the definition of concepts in cognitive psychology, such as the schema, the template, the prototype, and the geon approach (cf. Sect. 4.2).

This list of partial systems may be continued by pointing to efforts at providing a more general theory of machine translation (Dorr 1993), finding a universal set of semantic primitives (Schank and Abelson 1977; Wierzbicka 1991), application-oriented systems of language production (Reiter and Dale 1997), as well as efforts to improve indexing and retrieval on the Internet by means of metadata mark-up based on XML, RDF, and OWL (Berners-Lee, Hendler, and Lassila 2001). This raises the question: Which of the partial systems should be chosen to serve as components of a general, complete, coherent, computational model of natural language communication?

On the one hand, there is little interest in reinventing a component that is already available. On the other hand, reusing partial theories by integrating them into a general system of natural language communication comes at a considerable cost: Given that the available theories have originated in different traditions and for different purposes, much time and effort would have to be spent on making them compatible.

Apart from the time-consuming labor of integrating partial theories there is the more general question of which of them could be suitable in principle to be part of a coherent, functional theory of how natural language communication works. This question has been investigated in FoCL'99 for the majority of the systems listed above.[4]

As a result, Database Semantics was developed from scratch. Thereby many of the ideas and methods of the above systems have been absorbed. The most basic ideas are the notions of a proposition, as formulated by Aristotle, and of the time-linear structure of language, as emphasized by de Saussure.

While our grammatical analysis is very traditional in many respects, it does not adopt the commonly practiced separation between syntax (combinatorics) and semantics (interpretation). Instead, syntactic and semantic composition are derived simultaneously (cf. Tugwell 1998) in a time-linear order. Thus, the only difference between a purely syntactic and a syntactic–semantic grammar is that the former defines (i) fewer lexical properties of the parts, and (ii) fewer relations between the parts, than the latter.

VI. FORMAL FOUNDATIONS

Database Semantics is the first and so far the only rule system in which natural language interpretation and production are reconstructed as *turn-taking*, i.e., the cognitive agent's ability to switch between the speaker and the hearer mode. The reconstruction of the communication cycle in Database Semantics is founded on two innovations:

> The *algorithm* of Left-Associative Grammar (LA-grammar, TCS'92):
> LA-grammar is based on the principle of possible continuations. This is in contrast to the algorithms commonly used in today's linguistics, namely Phrase Structure Grammar (PSG) and Categorial Grammar (CG), which are based on the principle of possible substitutions. Computing possible continuations models the time-linear structure of natural language and permits us to handle turn-taking as the interaction of three kinds of LA-grammar, namely LA-hear, LA-think, and LA-speak.

[4] These analyses are conducted at a high level of abstraction. For example, rather than discussing in detail how Situation Semantics might differ from Discourse Semantics, FoCL'99 concentrates on the more basic question of whether or not a metalanguage-based truth-conditional approach could in principle be suitable for a computational model. Similarly, rather than comparing GPSG, LFG, HPSG, and GB, FoCL'99 concentrates on the question of whether or not the algorithm of substitution-based Phrase Structure Grammar could in principle be suitable for modeling the speaker and the hearer mode.

The *data structure* of a Word Bank (AIJ'01):
A Word Bank stores propositional content in the form of flat (nonrecursive) feature structures called *proplets*. While the substitution-based approaches embed, for example, the feature structure of the subject into the feature structure of the verb (unification, cf. 3.4.5), no such embedding is allowed in Database Semantics. Instead, the individual proplets code the grammatical relations between them in terms of features (i.e., attribute-value pairs) only. As a consequence, content represented as a set of proplets is well-suited for (i) storage and retrieval in a database, and for (ii) pattern matching, as needed to relate (iia) the levels of grammar rules and language (cf. 3.4.3 and 3.5.1) and (iib) the levels of context and language (cf. 3.3.1).

The algorithm of LA-grammar and the data structure of a Word Bank together provide the basis for an autonomous navigation through propositional content, utilizing the grammatical relations between proplets as a kind of railroad system and LA-grammar as a kind of locomotive which moves a unique focus point along the rails. This new way of combining a data structure and an algorithm serves as our basic model of thought. It may be used for merely activating content selectively in the Word Bank (free association), but may also be extended into a control structure which relates the agent's recognition and action using stored knowledge and inferences.

VII. Scope of the linguistic analysis

Our linguistic analysis aims at a systematic development of the major constructions of natural language. These are (i) functor–argument structure, (ii) coordination, and (iii) coreference. They occur intra- and extrapropositionally, and may be freely mixed.

The major constructions are analyzed in a strictly time-linear derivation order, in the hearer mode and in the speaker mode. It is shown that the much greater functional completeness of Database Semantics as compared to the sign-oriented approaches is no obstacle to a straightforward, linguistically well-motivated, homogeneous analysis, which provides for a highly efficient computational implementation.

The analyses include constructions which have eluded a generally accepted treatment within Nativism.[5] These are the gapping constructions (cf. Chaps. 8 and 9), especially "right-node-raising", and coreference (cf. Chap. 10) in the "donkey" and the "Bach–Peters" sentence.

VIII. Structure of the book

The content of this book is presented in three parts. Part I presents the general framework of the SLIM Theory of Language (FoCL'99) in terms of the cognitive agent's external interfaces, data structure, and algorithm. This part addresses many questions

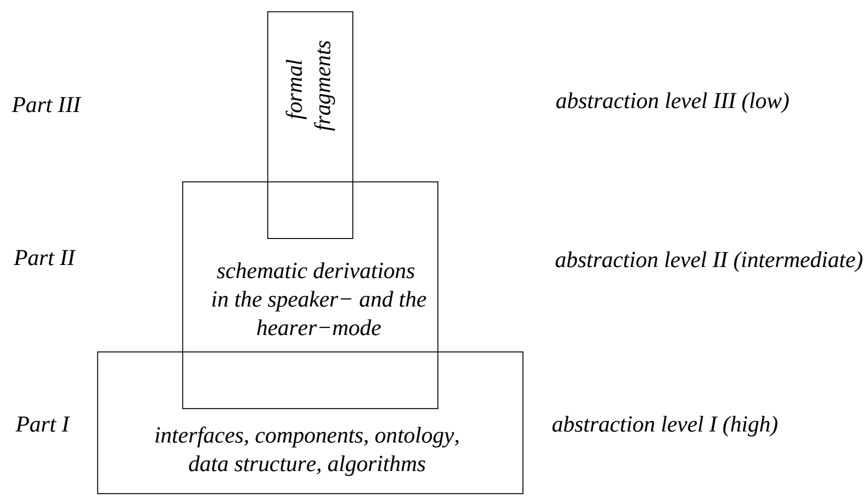
[5] Nativism (Chomsky 1957 et seq.) is a sign-oriented theory of language using substitution-based Phrase Structure Grammar. Nativism aims at characterizing the speaker hearer's innate knowledge of language (competence) – excluding the use of language in communication (performance).

which are crucial for the overall system, but cannot be pursued in further detail. Examples are the nature of concepts and their role in recognition and action, the reference mechanisms of the different sign kinds, and the formal structure of the context level.

Part II systematically analyzes the major constructions of natural language, presented as schematic derivations of English examples in the hearer and the speaker mode. The hearer mode analyses show the strictly time-linear coding of functor–argument structure and coordination into sets of proplets, treating coreference as a secondary relation based on inferencing. The speaker mode analyses show the retrieval-based navigation through a Word Bank (conceptualization), as well as the language-dependent sequencing of word forms and the precipitation of function words.

Part III presents fragments of English. Expanding on Montague's use of this term, a "fragment" refers to a system of natural language communication which is functionally complete but has limited coverage. The fragments show the interpretation and the production of small sample texts in complete detail, explicitly defining the lexicon and the LA-hear, LA-think, and LA-speak grammars required.

The different scope and the different degrees of abstraction characterizing the three parts may be summarized schematically as follows:



A high degree of abstraction corresponds to a low degree of linguistic and technical detail, and vice versa.

The general framework outlined in Part I is built upon in Part II. The methods of analysis presented in Part II are built upon in Part III. The analyses and definitions of Part II and III have served as the declarative specification of an implementation called JSLIM (Kycia 2004), which is currently being reimplemented by Jörg Kapfer and Johannes Handl using Java^{TM} version 5 (1. 5).

Part I

The Communication Mechanism of Cognition

1. Matters of Method

In science, the method of verification is the outermost line of defense against error. It may be crude as long as it can be made objective. Designed for a particular theory (or kind of theories), the verification method should interact with its theory in such a way that there constantly arise new questions of a kind (i) which can be decided more or less conclusively by the method of verification, and (ii) the answers to which are relevant to the theory's further development.

In natural science, verification consists in experiments which are (i) specified exactly in quantitative terms and (ii) which can be repeated by anybody anywhere. This requires that the notions and structures of the theory are so precise that they are suitable for the scientific setup of experiments. For the grammatical analysis of language, however, the quantitative verification method happens to be unsuitable.

The method we propose instead consists in building a functional model of natural language communication. This requires (i) a declarative specification in combination with an efficiently running implementation (prototype of a talking robot), (ii) establishing objective channels of observation, and (iii) equating the adequacy of the robot's behavior with the correctness of the theory – which means that the robot must have (iv) the same kinds of external interfaces as humans, and process language in a way which is (v) input/output-equivalent with the language processing of humans.

1.1 Sign- or Agent-Oriented Analysis of Language?

A natural language manifests itself in the form of signs, the structures of which have evolved as conventions within a language community. Produced by cognitive agents in the speaker mode and interpreted by agents in the hearer mode, these signs are used for the transfer of content from the speaker to the hearer. Depending on whether the scientific analysis concentrates on the isolated signs or on the communicating agents, we may distinguish between *sign-oriented* and *agent-oriented* approaches.[1]

Sign-oriented approaches like Generative Grammar, Truth-Conditional Semantics, and Text Linguistics analyze expressions of natural language as objects, fixed on paper, magnetic tape, or by electronic means. They abstract away from the aspect of communication and are therefore neither intended nor suitable to model the speaker

[1] Clark (1996) distinguishes between the *language-as-product* and *language-as-action* traditions.

and the hearer mode. Instead, linguistic examples, isolated from the communicating agents, are analyzed as hierarchical structures which are formally based on the principle of *possible substitutions*.

The agent-oriented approach of Database Semantics (DBS), in contrast, analyzes signs as the result of the speaker's language production and as the starting point of the hearer's language interpretation. Inclusion of the agents' production and interpretation procedures requires a time-linear analysis which is formally based on the principle of *possible continuations*.

The goal of Database Semantics is a theory of natural language communication which is complete with respect to function and data coverage, of low mathematical complexity, and is suitable for an efficient implementation on the computer. The central question of Database Semantics is:

How does communicating with natural language work?

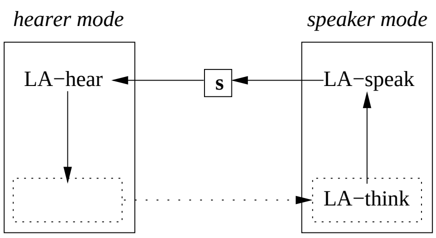
In the most simple form, this question is answered as follows.

Natural language communication takes place between cognitive agents. They have real bodies "out there" in the world with external interfaces for nonverbal recognition and action at the context level, and verbal recognition and action at the language level. Each agent contains a database in which contents are stored. These contents consist of the agent's knowledge, its memories, current recognition, intentions, plans, etc.

The cognitive agents can switch between the speaker and the hearer mode (turn-taking).[2] In a communication procedure, an agent in the speaker mode codes content from its database into signs of language which are realized externally via the language output interface. These signs are recognized by another agent in the hearer mode via the language input interface, their content is decoded, and is then stored in the second agent's database. This procedure is successful if the content coded by the speaker is decoded and stored equivalently by the hearer.

In Database Semantics, the modeling of turn-taking is based on a special data structure in combination with the time-linear algorithm of Left-Associative Grammar (LA-grammar).[3] The algorithm is used in three variants, called LA-hear, LA-think, and LA-speak. In communication, these three LA-grammars cooperate as follows:

1.1.1 THE BASIC MODEL OF TURN-TAKING



[2] For a study of turn-taking see Thórisson (2002).
[3] For the formal definition and complexity analysis of LA-grammar as well as a detailed comparison with Phrase Structure Grammar and Categorial Grammar see FoCL'99, Part II.

In the agent shown on the right (speaker mode), LA-think selectively activates content stored in the agent's database. The activated content is mapped into surfaces of a natural language by LA-speak, which are realized as external signs (represented by the small box containing **s**). In the agent shown on the left (hearer mode), LA-hear interprets the signs, which are stored in the agent's database.

The representation of turn-taking shown in 1.1.1 may be interpreted in two ways:

1.1.2 TWO VIEWS OF TURN-TAKING

1. *Viewed from the outside:*
   Two communicating agents are observed as they are taking turns. This is represented by 1.1.1 when the two boxes are taken to be two different agents, one in the hearer and the other in the speaker mode.
2. *Viewed from the inside:*
   One communicating agent is observed as it switches between being the speaker and the hearer. This is represented by 1.1.1 when the two boxes are taken to be the same agent switching between the speaker and the hearer mode (with the dotted right-hand arrow indicating the switch).

In DBS, turn-taking is regarded as a well-defined, well-motivated computational problem, which is central to the linguistic analysis of natural language: all syntactic and semantic analysis must be integrated into turn-taking as the most basic mechanism of communication. Without it, there is only one-sided monologue as the limiting case.

1.2 Verification Principle

Our theory of natural language communication is developed as a functional model, presented as a *declarative specification* for an efficient computer program with associated hardware. A declarative specification describes the necessary properties of a software, such as the external interfaces, the data structure, and the algorithm. Thereby, the accidental[4] properties of an implementation, such as the choice of programming language or the stylistic idiosyncrasies of the programmer, are abstracted away from.

In contrast to an algebraic definition[5] in logic, a declarative specification is not based purely on set theory. Instead, it takes a procedural point of view, specifying the general architecture in terms of components with input and output conditions as well as the functional flow through the system. A declarative specification must be general enough to provide a solid mathematical foundation and structure, and detailed enough to permit easy programming in different environments.

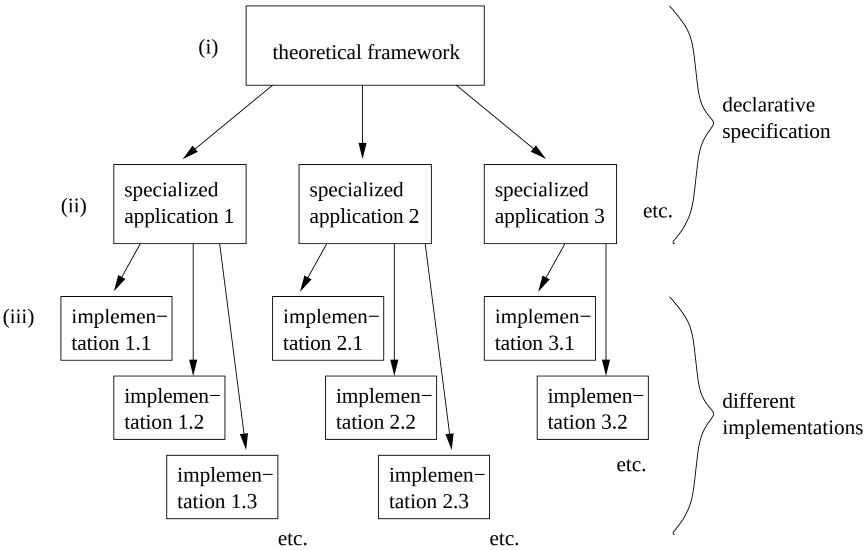
A declarative specification is needed because machine code is not easily read by humans. Even programs written in a higher level programming language such as Lisp

[4] The term accidental is used here in the philosophical tradition of Aristotle, who distinguishes between the necessary and the accidental (or incidental – kata sumbebêkos).
[5] The algebraic definition of LA-grammar in CoL'89 benefited greatly from help by Dana Scott.

are meaningful only to experts. What one would like to see in a piece of software is the *abstract functional solution* to the task it was designed to perform.

The declarative specification for a certain application consists of two levels: (i) a general theoretical framework (e.g., a functional system of natural language communication) and (ii) a specialization of the general framework to a specific application (e.g., English, German, Korean, or any other natural language). The theoretical framework in combination with a specialized application may in turn be realized (iii) in various different implementations, written in Lisp, C, or Java, for example.

1.2.1 Correlation of declarative specification and implementations



A declarative specification may have many different implementations which are equivalent with respect to the necessary properties. In Database Semantics, the evolving declarative specification must always be accompanied by at least one up-to-date implementation in order to automatically demonstrate the functioning of the theory in its current stage, and to test it with respect to an ever-widening range of various tasks. In this way, errors, incompletenesses, and other weaknesses of the current stage may be determined (explicit hypothesis formation, cf. FoCL'99, 7.2.3), which is a precondition for developing the next improved stage of the declarative specification.

The cycle of theory development and automatic testing is the *verification method* of Database Semantics. It differs from the quantitative methods of the natural sciences (repeatability of experiments) as well as the logical-axiomatic methods of mathematics (proof of consistency), though it is compatible with them.

The verification method[6] of Database Semantics is important for the following reasons. First, the signs of natural language are based on conventions which are not sus-

[6] See also FoCL'99, Introduction VIII–X.

ceptible to the quantitative methods of the natural sciences. Second, the analysis of the natural languages in linguistics and neighboring fields such as the philosophy of language is fragmented into a very large number of different schools and subschools, which raises the question of their comparative evaluation.

1.3 Equation Principle

Database Semantics aims at modeling the language communication of artificial agents as naturally as possible for two reasons. First, maximal *user-friendliness* should be provided in practical applications. User-friendliness in man–machine communication means that the human and the robot can understand each other (i) correctly and (ii) without the human having to adapt to the machine.[7]

Second, long-term *upscalability* in theory development should be ensured. Upscaling in the construction of a talking robot means that one can proceed without difficulty from the current prototype to one of greater functional completeness and/or data coverage.[8] In the history of science, difficulties in upscaling have practically always indicated a fundamental problem with the theory in question.[9]

To ensure user-friendliness and upscalability in the long run, Database Semantics must strive to approximate at the various levels of abstraction what has been called "psychological reality." For this purpose, we propose the following principle, which equates the correctness of the theoretical description with the behavioral adequacy of the electronic model (prototype of a talking robot).

1.3.1 The equation principle of database semantics

1. The more realistic the reconstruction of cognition, the better the functioning of the model.

[7] For this, the robot must be designed to have procedural counterparts of human notions. For example, in order to understand the word red, the robot must be capable of physically selecting the red objects from a set; in order to understand the notion of being happily surprised, the robot must be capable of experiencing this emotion itself; etc.

Given that the technical preconditions for this kind of user-friendliness will not become available for some time, Liu (2001) proposes to integrate current robotic capabilities with practical tasks guided by humans. This is a positive example of a smart solution, like the use of restricted language in machine translation (cf. FoCL'99, p. 47).

[8] For example, functional completeness requires the ability of automatic word form recognition in principle. Extending the data coverage means that more and more word forms of the language can be recognized; similarly, functional completeness requires the ability of contextual action in principle. Extending the data coverage means that more and more contextual action types such as different kinds of locomotion, manipulation, etc., become available.

[9] Problems with upscaling in Truth-Conditional Semantics arise in the attempts to handle the Epimenides Paradox (cf. FoCL'99, Sect. 19.5), propositional attitudes (cf. ibid, Sect. 20.3), and vagueness (ibid, Sect. 20.5). Problems with upscaling in Generative Grammar arise in the attempts to handle the constituent structure paradox (ibid, Sect. 8.5) and gapping constructions (cf. Chaps. 8 and 9 below).

2. The better the functioning of the model, the more realistic the reconstruction of cognition.

The first part of the Equation Principle looks for support from and convergence with the neighboring sciences in order to improve the performance of the prototype. This means, for example, that we avoid conflicts with established facts or strong conjectures regarding the phylogenetical and the ontogenetical development as provided by ethology and developmental psychology, include the functional explanations of anatomy and physiology, and take seriously the results of mathematical complexity theory (no undecidable or exponential algorithms).

The second part of the equation principle provides a heuristic strategy in light of the fact that the "real" software structures of cognition (at their various levels of abstraction) are not accessable to direct observation.[10] Our strategy tries to achieve a realistic reconstruction indirectly by aiming for functional completeness and completeness of data coverage in the incremental upscaling of an artificial cognitive agent.

1.4 Objectivation Principle

For a functional reconstruction of cognition in general and natural language communication in particular, different kinds of data are available. The differences stem in part from alternative constellations in which the data originate, and in part from alternative channels which are used in the respective constellations.

The constellations regard the interaction between (i) the user, (ii) the scientist, and (iii) the electronic model (robot). They are distinguished as follows:

1.4.1 Constellations providing different kinds of data

1. Interaction between (i) the user and (iii) the robot
2. Interaction between (i) the user and (ii) the scientist
3. Interaction between (ii) the scientist and (iii) the robot

Depending on the constellation, data can be transmitted via the following channels:

1.4.2 Data channels of communicative interaction

1. The *auto-channel* processes input automatically and produces output autonomously, at the context as well as the language level. In natural cognitive agents, i.e., the user and the scientist, the auto-channel is present from the very beginning in its

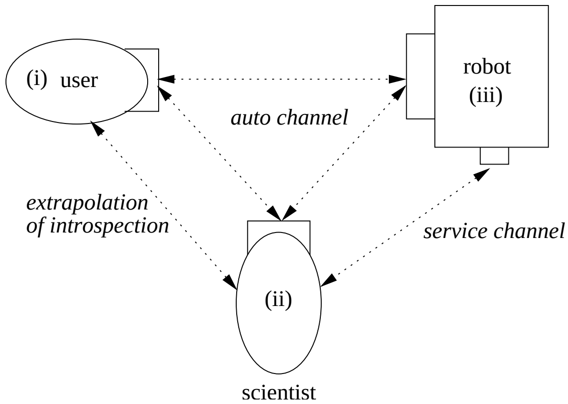
[10] A notable exception is the direct study of central cognition in neurology, especially fMRI or functional magnetic resonance imaging (cf. Matthews et al. 2003; Jezzard et al. 2001). Currently, however, these data leave room for widely differing interpretations, and are used to support conflicting theories.

full functionality. In artificial agents, in contrast, the auto-channel must be reconstructed – and it is the goal of Database Semantics to reconstruct it as realistically as possible.

2. The *extrapolation of introspection* is a specialization of the auto-channel and results from the scientists' effort to improve man–machine communication by taking the view of the human user. This is possible because the scientist and the user are natural agents.
3. The *service channel* is designed by the scientist for the observation and control of the artificial agent. It allows direct access to the robot's cognition because its cognitive architecture and functioning is a construct which in principle may be understood completely by the scientist.

The three constellations and the role of the three data channels in the interaction between user, scientist, and robot may be summarized graphically as follows:

1.4.3 INTERACTION BETWEEN USER, ROBOT, AND SCIENTIST



The scientist observes the external behavior of the user and the robot via the auto-channel, i.e., the scientist sees what they do and can also interview them about it. In addition, the scientist observes the cognitive states of (a) the user indirectly via a scientifically founded extrapolation of introspection and (b) the robot directly via the service channel. For the scientist, the user and the robot are equally real agents "out there" in the world, and their cognitive states have the same ontological status.

Of the three channels, the auto-channel is available to the user, the robot, and the scientist. It is the channel used most, but it is also most prone to error: At the level of context there are the visual illusions, for example, and at the level of language there are the misunderstandings. In addition, one has to take into account the possibility that the partner of discourse might deviate from the truth, either consciously or unconsciously.

As long as everyday access to the partner of discourse is restricted to the auto-channel, we can never be completely certain whether what we said was really understood as intended by us, or whether we really understood what was intended by the other, or whether what was said was really true. In philosophy, this is the much discussed problem known as *solipsism* (Wittgenstein 1921).

For a scientific analysis of natural language communication, however, there are the priviledged accesses of (i) the extrapolation of introspection and (ii) the service channel. In the extrapolation of introspection, the discourse between the scientist and the user is restricted to the domain of user–robot interaction. Therefore, misunderstandings between the scientist and the user are much less likely than in free communication, though they are still possible. The direct access to the robot via the service channel, furthermore, allows the scientist to determine objectively whether or not the cognition of the artificial agent is functioning properly. Thus, artificial cognitive agents are special insofar as they are not subject to the problem of solipsism.

1.5 Equivalence Principles for Interfaces and for Input/Output

The methodological principles of Database Semantics presented so far, namely

1. the *Verification Principle*
   i.e., the development of the theory in the form of a declarative specification which is continuously verified by means of an implemented prototype (cf. Sect. 1.2),
2. the *Equation Principle*
   i.e., the equating of theoretical correctness with the behavioral adequacy of the prototype during long-term upscaling (cf. Sect. 1.3), and
3. the *Objectivation Principle*
   i.e., the establishing of objective channels for observing language communication between natural and artificial agents (cf. Sect. 1.4),

are constrained by

4. the *Interface Equivalence Principle*, and
5. the *Input/Output Equivalence Principle.*

According to the Principle of Interface Equivalence (4), the artificial surrogate must be equipped with the same interfaces to the external world as the natural original. At the highest level of abstraction, this requires the external interfaces of recognition and action at the context and the language level (cf. 2.1.3). At lower levels of abstraction, the interfaces in question split up into the different modalities (cf. Sect. 2.2) of vision, audio, tactile, etc., for recognition, and locomotion, manipulation, etc., for action.

The Interface Equivalence between the model and the natural original is crucial for the automatic reconstruction of reference, i.e., the relation between language and the world. For example, if the robot cannot perceive, it cannot understand the human's reference to a new object in their joint task environment. The Interface Equivalence

Principle has fundamental consequences on the theory of semantics for natural language, especially the ontological foundations (cf. 2.3.1).

The Principle of Input/Output Equivalence (5) presupposes Interface Equivalence (4). Input/Output Equivalence requires that the artificial agent (i) takes the same input and produces the same output as the natural original, (ii) disassembles input and output in the same way into parts, and (iii) orders the parts in the same way during intake and discharge. The input and output data, like the external interfaces, are concretely given and therefore are susceptible to an objective structural analysis.

The Input/Output Equivalence between the model and the natural original is especially relevant for the automatic interpretation and production of the signs used in natural language communication. Therefore, this principle has fundamental consequences on the theory of grammar for natural language.

The two Equivalence Principles constitute a *minimal requirement* for any scientific reconstruction of cognition in general and the mechanism of natural language communication in particular. The reason is as follows: If we had direct access to the architecture and the functioning of cognition, comparable to the investigation of the physical structures and functions of the bodily organs in the natural sciences (anatomy, physiology, chemistry, physics), the resulting model would certainly have to satisfy the Principles of Interface Equivalence and Input/Output Equivalence.

If, due to the absence of direct access, the nature of the cognitive system must be inferred indirectly, namely in an incremental process of upscaling the functional completeness and the data coverage of an artificial surrogate, this does not diminish the importance of the external interfaces and the input/output data. On the contrary, as concretely given, directly observable structures they constitute the external fixpoints for any reconstruction of the internal cognition procedures which is scientifically well-founded.

1.6 Surface Compositionality and Time-Linearity

The general principles of Interface Equivalence and Input/Output Equivalence require a careful analysis and reconstruction (i) of the natural agent's recognition and action components and (ii) of the data being passed through these components. One important kind of data are the expressions of natural language produced in the speaker mode and interpreted in the hearer mode.

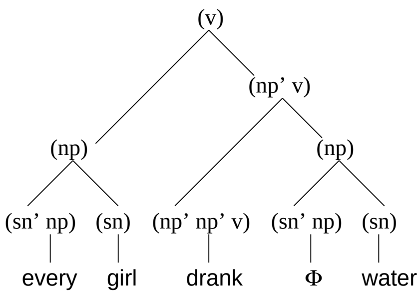
Externally, these data are objects in a certain medium, represented by sounds, handwritten or printed letters, or gestures of a sign language, which can be recorded on film, tape, or disc, and measured and described with the methods of the natural sciences. Given that these objects are concretely given, they constitute the empirical basis which linguistic analysis should neither add to nor subtract from. This elementary methodological principle is known as Surface Compositionality (SCG'84):

1.6.1 SURFACE COMPOSITIONALITY

> A grammatical analysis is surface compositional if it uses only the concrete word forms as the building blocks of composition, such that all syntactic and semantic properties of a complex expression derive systematically from the syntactic category and the literal meaning of the lexical items.

Surface Compositionality is best illustrated by examples which violate it, such as the following grammatical analysis:

1.6.2 ANALYSIS VIOLATING SURFACE COMPOSITIONALITY



In order to treat the noun phrases every girl and water alike, this analysis postulates the zero element Φ. The presumed “linguistic generalization” is illegitimate, however, because the postulated determiner Φ of water is not concretely given in the surface.

Nevertheless, the categories of 1.6.2 are well-motivated and defined as follows:

1.6.3 THE CATEGORIES OF 1.6.2

(sn’ np) = *determiner*, takes a singular noun sn’ and makes a noun phrase np.
(sn) = *singular noun*, fills a valency position sn’ in the determiner.
(np’ np’ v)= *transitive verb*, takes a noun phrase np and makes an intransitive verb (np’ v).
(np) = *noun phrase*, fills a valency position np’ in the verb.
(np’ v) = *intransitive verb*, takes a noun phrase np and makes a (v).
(v) = *verb* with no open valency positions (sentence).

The rules generating Example 1.6.2 are based on the principle of possible substitutions, and are defined as follows:

1.6.4 RULES COMPUTING POSSIBLE SUBSTITUTIONS FOR DERIVING 1.6.2

(v) → (np) (np’ v)
(np) → (sn’ np) (sn)
(np’ v) → (np’ np’ v) (np)
(sn’ np) → every, Φ
(sn) → girl, water
(np’ np’ v) → drank

Each rule replaces the category on the left-hand side of the arrow by the categories on the right-hand side (top-down derivation). It is also conceivable to replace the categories on the right-hand side by the one on the left-hand side (bottom-up derivation).

Without the zero determiner postulated in 1.6.2, at least one additional rule would have to be defined. However, according to the Principle of Surface Compositionality, it is methodologically unsound to simply postulate the existence of something that is absent, but considered necessary or desirable.[11] Failure to maintain Surface Compositionality leads directly to high mathematical complexity and computational intractability.

Having determined the basic elements of linguistic analysis, i.e., the surfaces in the concretely given sign and their standard lexical analysis, let us turn to the proper grammatical relations between these basic items. The most elementary relation between the words in a sentence is their time-linear order. Time-linear means linear like time and in the direction of time (cf. Sect. 3.4).

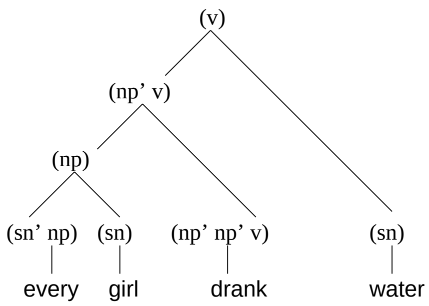
The time-linear structure of natural language is so fundamental that a speaker cannot but utter a text sentence by sentence, and a sentence word form by word form. Thereby the time-linear principle suffuses the process of utterance to such a degree that the speaker may decide in the middle of a sentence on how to continue.

Correspondingly, the hearer need not wait until the utterance of a text or sentence has been finished before his or her interpretation can begin. Instead the hearer will interpret the beginning of the sentence without having to know how it will be continued.

Example 1.6.2 violates not only Surface Compositionality, but also Time-Linearity. The grammatical analysis is not time-linear because it fails to combine every girl with drank directly. Instead, based on the principle of possible substitutions, the complex expression drank water must be derived first.

A time-linear analysis, in contrast, is based on the principle of possible continuations. As an example, consider the following time-linear derivation, which uses the same categories (cf. 1.6.3) as the non-time-linear derivation 1.6.2:

1.6.5 SATISFYING SURFACE COMPOSITIONALITY AND TIME-LINEARITY



This bottom-up derivation always combines a sentence start with a next word into a new sentence start, using the following (simplified) rules of Left-Associative Grammar:

[11] The inverse kind of violating Surface Compositionality consists in treating words which are concretely given in the surface as if they weren’t there, simply because they are considered unnecessary or undesirable for one’s “linguistic generalization.” For a more detailed discussion see SCG’84 and FoCL’99, Sects. 4.5, 17.2, 18.2, and 21.3.

1.6.6 Rules computing the possible continuations for deriving 1.6.5

(VAR' X) (VAR) ⇒ (X)
(VAR) (VAR' X) ⇒ (X)

Each rule consists of three patterns. The patterns are built from the variables VAR, VAR', and X.[12]

The first pattern of a rule, e.g., (VAR' X), represents the sentence start *ss*, the second pattern, e.g., (VAR), the next word *nw*, and the third pattern, e.g., (X) the resulting sentence start *ss'*. The variables VAR and VAR' are restricted to a single category segment, while X is a variable for a sequence of category segments consisting of zero or more elements.

Rules computing possible continuations are based on matching their patterns with the input expressions, thereby binding their variables:

1.6.7 Application of a rule computing a possible continuation

	ss		*nw*		*ss'*	
rule patterns	(VAR'	X)	(VAR)	⇒	(X)	
	|	|	|		|	*matching and binding*
categories	(sn'	np)	(sn)		(np)	
surfaces	every		girl		every girl	

During matching, the variable VAR' is "vertically" bound to sn', the variable X to np, and the variable VAR to sn. In the result, the valency position sn' of the determiner category (sn' np) has been filled (or canceled), producing the *ss'* category (np), and the input surfaces every and girl are concatenated into every girl.

To handle the combination between a verb and object nouns with or without a determiner, e.g., ...drank + a coke versus ...drank + water, in a surface compositional manner, the possible values of the variables VAR and VAR' are restricted[13] and correlated as follows:

1.6.8 Variable definition of the time-linear rules for deriving 1.6.5

If VAR' is sn', then VAR is sn. (*identity-based agreement*)
If VAR' is np', then VAR is np, sn, or pn. (*definition-based agreement*)

The formalism of a time-linear derivation sketched in 1.6.5–1.6.8 is of a preliminary kind. It was used in NEWCAT'86 for the automatic time-linear analysis of 221 syntactic constructions of German and 114 of English, complete with LISP source code. It was also used in CoL'89 for 421 syntactic–semantic constructions of English with a sign-oriented, hierarchical semantic analysis.

[12] There is a convention in Database Semantics that constants are written in lowercase Roman letters, while variables are written in uppercase Roman letters or in lowercase Greek letters. Cf. Appendix C, Sect. C.3.

[13] The variable restrictions for handling agreement in English are summarized in the Appendix C, C.3.4.