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

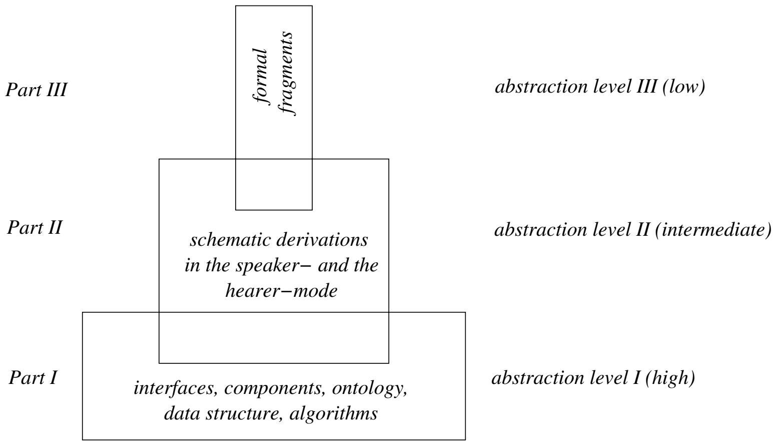
[5] Nativism (Chomsky 1957 et seq.) is a sign-oriented theory of language using substitution-based Phrase Structure Grammar. Nativism aims at characterizing the speaker hearer's innate knowledge of language (competence) – excluding the use of language in communication (performance).

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™ version 5 (1. 5).