12. Conclusion

For natural language communication to be comfortable, the use of the signs must be *flexible* (robust). Therefore, one challenge in building a talking robot is to reproduce the flexibility which is integral to human communication.

In DBS, this flexibility has three aspects. The first is the ability (i) to change levels of abstraction effortlessly and (ii) to use language indirectly as in analogies, metaphors, and so on. This aspect is based on inferences. Derived and applied automatically, the inferences are triggered by the evaluated data stream provided by the agent's cognition.

The second aspect inheres in the nature of pattern matching. This aspect of flexibility is handled by schemata which utilize restricted variables, core values, and the semantic relations of structure, enabling highly differentiated retrieval. Pattern matching is used, for example, in recognition, when raw data are classified by concept types, and in retrieval, when schemata activate relevant contents in the database.

The third aspect is the ability of the hearer to ask for clarification. This is another aspect handled by inferences.

These aspects of flexibility result combined in practically unlimited expressive power, for example, in data mining. In an alert DBS robot, recognition, subactivation, intersection, and inferencing are continuously running and provide autonomous control with the stored world knowledge and the experiences relevant for maintaining the agent in a state of balance.

12.1 Level of Abstraction

The construction of an artificial agent with language may be compared to the construction of an airplane. For both, there is a natural prototype with which they share certain crucial properties and from which they differ in others.

An airplane (artificial model) and a bird (natural prototype) have in common that both stay airborne according to the same principles of the theory of aerodynamics. They differ, however, in that lift and propulsion are combined in the

R. Hausser, *Computational Linguistics and Talking Robots: Processing Content in Database Semantics*,
DOI 10.1007/978-3-642-22432-4_12, © Springer-Verlag Berlin Heidelberg 2011

flapping wings of birds, but separated in planes into the fixed wing for lift and the propeller or jet engine for propulsion.

Correspondingly, an artificial DBS agent and its human prototype have in common that both have a memory connected to external interfaces. An example of their many differences, in contrast, is that DBS uses the equality of core values for sorting proplets into certain token lines of a Word Bank, while the natural counterpart presumably uses a more differentiated similarity metric and different principles of storage and retrieval.

Orthogonal to this analogy between artificial flying and artificial cognition is a crucial difference regarding their interaction with the human user. In an airplane, the method of being airborne is completely separate from its user-friendliness for humans. The latter concerns the size of the doors and the seats, the cabin pressure, the service, etc., while the former concerns the shape of the wings, the manner of propulsion, the technique of takeoff and landing, etc.

For DBS as a computational theory of cognition, in contrast, maximizing the similarity between the artificial agent and its natural prototype amounts directly to maximizing the user-friendliness for humans.[1] This correlation between the artificial agent and its natural prototype has been preformulated as the Equation Principle in NLC'06, 1.3.1: [2]

12.1.1 The Equation Principle of Database Semantics

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

The principle is aimed at long-term upscaling. It applies at a level of abstraction at which it does not matter for completeness of function and data coverage whether cognition is based on natural wetware or electronic hardware.[3] The principle applies to the communication and reasoning aspects[4] of a talking agent which are (i) concretely observable and (ii) impact directly the quality of free human-machine communication in natural language.

[1] Without giving up any of the applications provided by computers already.

[2] The principles presented in Chap. 1 of NLC'06 are (1) the Verification Principle, (2) the Equation Principle (12.1.1), (3) the Objectivation Principle, (4) the Equivalence Principle for Interfaces (2.6.2), and (5) the Equivalence Principle for Input/Output (2.6.3).

[3] As for "true feelings," they are equally inaccessible in natural and artificial agents. In fact, they may be more accessible technically in artificial agents due to their service channel (Chap. 2).

[4] It applies less to other aspects of the artificial agent, such as looks – as shown by C3PO and Ripley.

The flexibility required by the human user depends in part on what the partners in discourse accept as communication. For example, high brow journals will accept only papers which exhibit the correct use of the current buzz words in their domain, some bureaucrats are interested only in a small range of topics, such as last name, first name, date of birth, and place of birth, the customers in a bar, who will accept a stranger only if (s)he is a certain type, and so on. In linguistics, these restrictions of a domain to a certain, well-defined protocol are called *register* (Halliday and Hasan 1976). Register is important for communication, natural or artificial, because it defines the channel of communication from a social point of view.

In this sense, the construction of a talking robot must solve the task of *register adaptation*. The system should be able to smoothly agree with the partner in discourse on a certain level of abstraction, to switch the level of abstraction up or down, accompanied by adjustments of dialect and of intonation, to select between a declarative, interrogative, or imperative sentential mood, to present content in a certain way, etc. This is the most fertile field of Conversation Analysis, founded by Sacks and Schegloff (Schegloff 2007).[5]

The DBS robot requires a computational model of the motivational structures behind the actions of the partners in discourse and of the strategies guiding these actions. For this, the Schegloff corpus provides many authentic examples. A DBS interpretation of the Schegloff corpus would have to translate such founding notions as "pre," "post," "pre-pre," etc., (time-linear!) into DBS inferences with certain goals.

To overcome the inflexibility of current systems. Mohammadzadeh et al. (2005) propose "template guided association," aimed at XML. The DBS approach is similar: the templates are the schemata built from pattern proplets. In addition to a highly effective primary key, DBS provides the option to search for continuation values, morphosyntactic categories, base forms, matching inferences, memorable outcomes, n-grams, frequencies, and so on.

The conceptual backbone of DBS is storing proplets in the order of their arrival[6] in combination with the "no-change" rule of a content-addressable memory. Because the processing of content never modifies what is stored in memory, there is a strict separation between the storage of content and its processing. The storage operations, like the inferences, always write to the *now*

[5] Schegloff's examples are interesting and carefully analyzed, but a computational implementation was not one of Schegloff's goals. The observable facts are well documented, but the difficulties of interpretation or of choosing between several possible interpretations are pointed out repeatedly.

[6] In DBS, time is represented solely by the proplets' relative order of arrival. There are three possibilities: proplet A is earlier than than proplet B, proplet A is later than proplet B, or proplets A and B are simultaneous. This order is reflected by the prn values of the proplets.

front, using the same, simple, transparent procedure (4.1.1) to ensure historical accuracy and to conform to the structure of a content-addressable memory.

For correcting stored content, a comment is written to the *now front*, like a diary entry noting a change in temperature (cf. 4.4.2 for an analogous example). The comment refers to the content by means of addresses, providing instant access. When subactivation lights up a content, any and all addresses pointing at it are activated as well. When subactivation lights up an address, the original and all the other addresses pointing at it are also subactivated. In short, originals and their addresses are systematically *co-subactivated* in DBS.

12.2 RMD Corpus

While the core values, the semantic relations, and the levels of abstraction are agent-internal constructs of cognition, the language data are agent-external objects, collected, for example, as a corpus. Like any contemporary linguistic theory, DBS requires corpora to obtain frequency distributions of various kinds in a standardized framework. For example, when expanding automatic word form recognition, recognition rates will improve best if the most frequent word forms are integrated into the software first, and similarly for parsing syntactic-semantic constructions, and so on.

The frequency information should be obtained from a standardized RMD corpus, i.e., a Reference Monitor corpus structured into Domains. The reference corpus consists of a subcorpus for everyday language, complemented by subcorpora for different domains such as law, medicine, physics, sport, politics (cf. von der Grün 1998), including fiction, e.g., movie scripts. Their sizes may be determined by methods evolved from those used for the Brown corpus (Kučera and Francis 1967, Francis and Kučera 1982).

The reference corpus is continued with monitor corpora following every year (Sinclair 1991, p. 24–26). The annual monitor corpora resemble the reference corpus in every way: overall size, choice of domains, domain sizes, etc. The reference corpus and the monitor corpora use texts from a carefully selected set of *renewable* language data: newspapers for everyday language, established journals for specific domains, and a selection of fiction which appeared in the year in question.

Most of the corpus building and analysis may be done completely automatically. This holds (i) for the collecting of texts for the monitor corpora once the initial set of sources has been settled on, (ii) for the statistical analysis once a useful routine as been established. and (iii) for automatic word form recognition as well as syntactic-semantic parsing. Such automatic corpus processing

replaces markup by hand, ensuring the quality of standardization necessary for meaningful comparisons, and saving the labor of instructing and the cost of remunerating large groups of markup personnel.[7]

A succession of monitor corpora allows a detailed view of how the language and the culture are developing, in different domains and over many decades. Statistical analysis will show, for example, how politics and natural disasters cause a temporary frequency increase of certain words in certain domains.

A carefully built long-term RDM corpus is in the interest of the whole language community and should be entrusted to the care of a national academy. This would secure the necessary long-term funding, though much of the cost could be recovered from commercial use of the continuously upgraded RDM corpus of the natural language in question (Sect. 12.6).

In DBS, the routine of analyzing a corpus begins with running the corpus through automatic word form recognition.[8] The result is a set of proplets called a *content* and stored in the content-addressable database of a Word Bank. Next, semantic relations are established between proplets by means of syntactic-semantic parsing. Finally, LA-think, inferences, and LA-speak are added.

Just as there is no limit to the amount of content stored in a Word Bank, at least in principle, there is no limit to the amount of information that can be added to the content. The information is integrated as a system of footnotes and subfootnotes. The "footnotes" are realized as interpreted pointers to other contents in the Word Bank. This does not increase the number of proplets in the Word Bank, only the number of addresses connecting them.

The user may query the Word Bank content in natural language (provided that the language software is available). Once an LA-think and an LA-speak grammar have been added, the answers may be in the natural language of the query. This method, though developed with carefully constructed input sentences, may eventually be applied to free text such as pages in the Internet. One benefit would be a quality of recall and precision unachievable by a statistical approach (cf. FoCL'99, Sects. 15.4, 15.5) or by manual markup.

12.3 Evolution

A computational model of natural language communication, defined at a level of abstraction which applies to natural and artificial agents alike, need not necessarily include the dimension of evolution. Instead, the software machine could be built as a purely "synchronic" framework of computational function.

[7] See Sect. 8.5 for the use of a corpus for the purpose of search space reduction in DBS.

[8] The usual preprocessing assumed.

Nevertheless, it would shed doubt on DBS as a theory if it turned out to be incompatible with the mechanisms of evolution. By the same token, if DBS can be shown to be analogous to evolution in some relevant process, this may increase interest in DBS for aspects other than synchronic performance alone.

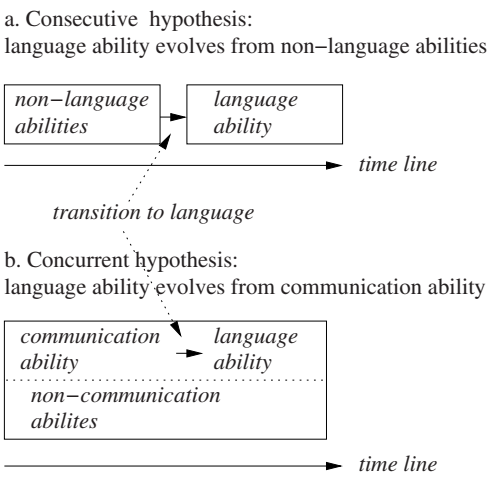
The DBS analysis of learning[9] resembles evolution in that it starts from fixed behavior patterns (FAPs) and reconstructs learning by disassembling the parts and recombining them into adaptive behavior. For such a metamorphosis to be automatic, the steps from one state to the other must (i) be small enough to be driven by the interaction with the environment, (ii) be suitable for both "aggregate states," (iii) and evolve in a meaningful time-linear sequence, parallel branches not excluded. The software would be a kind of genetic algorithm, from which much could be learned for applications.

Another question raised by evolution is whether the language component evolved from (i) a nonlanguage component or from (ii) a non-communication component. This is in part a terminological question. The two dichotomies are language and nonlanguage and communication and non-communication.

For example, a chameleon picking up an insect with its long tongue uses a non-communication as well as a nonlanguage ability. Yet, like all living beings, chameleons also have an ability to communicate with their conspecifics.[10] As far as we know, the chameleon kind of communication does not satisfy criteria 3 and 4 for being a natural language listed in 2.2.3. Thus, the chameleon uses nonlanguage communication abilities, for example, by changing the color of its skin to attract a mate (Karsten et al. 2009).

The two dichotomies may be used to construct two hypotheses about the evolution of natural language, shown graphically as follows:

12.3.1 Consecutive vs. Concurrent Hypothesis



According to the consecutive hypothesis, the language ability evolved from nonlanguage abilities. Theoretically, these could be divided into nonlanguage communication and nonlanguage non-communication abilities. However, because the consecutive hypothesis does not provide for a communication component, language seems to evolve from non-communication abilities, which is wrong.

The concurrent hypothesis, in contrast, assumes that the communication and the non-communication abilities co-evolved from the outset. This provides the two abilities with the same amount of time to evolve. Thus, when language finally arrives as the last stage of the communication ability, it can rely on many nonlanguage communication abilities evolved earlier. It can also rely on a continuous, systematic interaction between the communication and the non-communication (context) components.

Communication ability is necessary to all living beings at least to the degree that reproduction is ensured. In evolution, reproduction is more important than the survival of individual members because reproduction serves the survival of the species as a whole. Communication is at the heart of evolution because reproduction is the motor driving evolution, and communication with conspecifics is at the heart of reproduction.

The concurrent hypothesis is in concord with modern work in ethology (M. D. Hauser 1996). It suggests that the communication and the non-communication abilities of an animal not only evolve in parallel, but also have matching degrees of development. Accordingly, the cognitive abilities of a squirrel to communicate with other squirrels, for example, are just as evolved as its abilities for locomotion and behavior control.[11]

Furthermore, because even nonlanguage communication refers,[12] the communication and the non-communication abilities do not just grow in parallel, but are continuously in close interaction. In DBS, this interaction is modeled as a collaboration between the language and the context components. Natural language signs and contextual data interact in a software-mechanical way during interpretation in the hear mode and production in the speak mode. In short, designing the DBS robot in an evolutionary manner is another way of broadening the empirical base.

The above considerations regarding the evolution of natural language have a direct effect on the software design of DBS: if the communication and the

[9] Sects. 6.1–6.3.

[10] In addition to communication between conspecifics there is also the communication between different species, for example, between a plant and an insect for pollination (symbiosis).

[11] It would be thrilling to know what the little critters all have to communicate about.

[12] For example, a warning call makes reference to a predator.

noncommunication components are in such close functional interaction and at such equal levels of evolutionary development, the simplest approach is to program the two components essentially alike while paying close attention to their interaction with each other and with their external interfaces. Designing, using, and reusing software constructs which are as uniform and simple as possible is also regarded as good programming practice.

12.4 Semantics

Is DBS really a semantics? Let us answer this question by a point by point comparison with the reigning queen of semantics, Symbolic Logic.

The most important property common to Symbolic Logic and Database Semantics is that they are both Aristotelian. At the core of the Aristotelian approach is the use of only two basic semantic relations of structure, namely (i) coordination and (ii) functor-argument.

In Symbolic Logic, this fundamental insight is realized by (i) propositional calculus for extrapropositional coordination and (ii) predicate calculus for intrapropositional functor-argument. These may be extended into (iii) extrapropositional functor-argument [13]and (iv) intrapropositional coordination.[14]

Extrapropositional coordination is represented in Symbolic Log by expressions like ((p∧q)∨r). They are composed by the syntactic-semantic rules of propositional calculus (cf. FoCL'99, 19.3.2).

In DBS, the logical connectives are introduced by function words, lexically analyzed by automatic word form recognition. Called conjunctions, they carry their semantic contribution as a value inside their proplet representation. In DBS, the extrapropositional coordination corresponding to f(a)∧f'(a')∧f"(a") is shown by example in 3.2.1 and abstractly as the schema 3.2.6.

Intrapropositional functor-arguments are represented in Symbolic Logic by expressions like f(a,b), where f is a functor and a and b are arguments. These

[13] However, the Donkey sentence 11.5.5 shows that the extension to subclauses is not always possible. Though a lower level defect, it has been regarded as serious enough to spawn a massive body of literature trying to repair it – within, or almost within, the tradition of predicate calculus.

[14] Extending propositional calculus from extrapropositional to intrapropositional coordination also creates a problem for Symbolic Logic. It arises with examples like All the students gathered; John and Mary are a happy couple; Suzy mixed the flower, the sugar, and the eggs; etc., which intuitively suggest a collective, and not a distributive, reading (see Zweig 2008 for a overview; see also Hausser 1974; Kempson et al. 1981).

The "mix" problem does not arise in extrapropositional coordination, just as the "donkey" problem does not occur in intrapropositional functor-argument. Within Symbolic Logic, the "mix problem" can be solved by proposing new quantifiers and/or connectives, while the "donkey" problem is caused by failing scope and cannot be solved without uprooting the quantifier-based syntax and the bracketing structure of predicate calculus. The DBS solution is based on the use of addresses (11.5.7).

may be viewed as rudimentary parts of speech: f is like a two-place verb, while a and b are like nouns serving as subject and object.[15]

The corresponding constructs in DBS are features, defined as attribute-value pairs, using noun, verb, and adj as the core attributes of proplets. By distinguishing between the attribute and the value(s), there may be features such as [noun: dog] which are more differentiated and intuitive than the individual constants a_1, a_2, a_2, ... of Symbolic Logic, without any loss of generality.[16] In DBS, a functor-argument like f(a,b) of predicate calculus is shown by example in 3.2.5 and represented abstractly as the schema 3.2.6.

In any semantics, there are two kinds of meaningful elements, (i) *logical* and (ii) *contingent*. In predicate calculus, the logical elements are the connectives, the quantifiers, the variables, and a certain bracket syntax, while the contingent elements are letters for constants. In Database Semantics, the logical elements are the attributes and the variables in the proplets, while the non-variable values, i.e., the symbols, indexicals, and names, are contingent.

The traits common to Symbolic Logic and DBS may be increased further by taking the liberty to reinterpret the semantic interpretation of a sign-oriented approach as the *hear mode* of an agent-oriented approach.[17] In this way, the sign-oriented approach may be taken to cover one of the three steps of the natural language communication cycle.

This holds especially for Montague (1974), who shows in PTQ how to semantically interpret surfaces of English by translating them into formulas of predicate calculus. The translation is quasi-mechanical in that the reader can go mentally through the rule applications and the lambda reductions as if going through the steps of a proof. Montague formalizes the translation mechanism as a Categorial Grammar with (i) a cleverly structured set-theoretical interpretation and (ii) the reduction mechanism of typed lambda calculus (lambda reduction).[18]

From the viewpoint of this agent-oriented reinterpretation, DBS may be seen as completing Montague grammar in two ways. One completion is the extension of Montague grammar to the full cycle of natural natural language communication by adding the think and the speak mode. The other completion is

[15] Using typed lambda calculus, Montague (1974) worked hard to formalize functors and arguments set-theoretically as semantic types with corresponding syntactic categories, fitting into the rule schemata of Categorial Grammar. Lambda reduction in a typed lambda calculus may be viewed as a souped-up version of the categorial canceling rules.

[16] If needed, a feature may be represented abstractly as a pattern with the attribute and the values represented by variables. Cf. NLC'06, Sect. 4.1.

[17] This reinterpretation is the founding assumption of SCG'84.

replacing Montague's quasi-mechanical interpretation method with the computational automation provided by efficiently running software.

These two completions, however, have necessitated many solutions different from Montague grammar in particular and Symbolic Logic in general. For example, predicate calculus uses the logical elements to construct a syntactic exoskeleton, while DBS integrates the logical and the contingent elements into flat feature structures with ordered attributes (proplets) which code inter-proplet relations solely by addresses, and are therefore order-free (3.2.4).

The difference between DBS and predicate calculus, in particular regarding the role of quantifiers, may be shown in more detail with the following example from Montague (1974, PTQ):

12.4.1 Predicate calculus analysis of Every man loves a woman.

reading 1: $\forall x[man'(x) \rightarrow \exists y[woman'(y) \,\&\, love'(x, y)]]$

reading 2: $\exists y[woman'(y) \,\&\, [\forall x[man'(x) \rightarrow love'(x, y)]]$

This standard[19] analysis of predicate calculus treats the English surface as syntactically ambiguous. The reason is a scope ambiguity suggested by the creaking hinges of its quasi-mechanical exoskeleton. The contingent elements are man′, woman′, and love′.

In DBS the ambiguity alleged in 12.4.1 is not syntactic-semantic, but at best pragmatic.[20] The content of the sentence is shown as a set of proplets:

12.4.2 DBS analysis of Every man loves a woman.

noun: man	verb: love	woman
cat: snp	cat: decl	cat: snp
sem: exh pl	sem: pres	sem: indef sg
fnc: love	arg: man woman	fnc: love
prn: 23	prn: 23	prn: 23

In DBS, the logical quantifier ∀x is represented alternatively by the sem values exh pl (exhaustive plural) of the *man* proplet, while the quantifier ∃y is

[18] When writing SCG'84 we had to learn the hard way that a typed calculus is not very practical. Even worse, the small fragment we had managed to define in SCG'84 turned out to be unprogrammable. The solution, published as NEWCAT'86, is a strictly time-linear derivation order.

[19] Montague's main contribution is the quasi-mechanical translation of the English surface into formulas of predicate calculus. The number of grammatical constructions is rather small, and motivated in part by concerns of analytic philosophy, such as *de dicto* and *de re*.

[20] The distinction between syntactic, semantic, and pragmatic ambiguities is explained in FoCL'99, Sect. 12.5. Only syntactic ambiguities are of complexity-theoretic relevance. The syntactic ambiguity of the example 12.4.1 seems to originate mostly in classes on predicate calculus.

represented by the sem values indef sg (indef singular) of the *woman* proplet.[21] The verb's cat value decl (for declarative) and its sem value pres (for present tense) complete the picture. The proplets form an order-free set, and are held together by a common prn value, here 23.

In summary, predicate calculus uses (i) variables within formulas representing content and (ii) quantifiers (a) to bind the variables *horizontally* and (b) to replicate determiners at the same time. DBS, in contrast, (i) replaces the binding function of quantifiers by a prn value, (ii) employs variables solely for a *vertical* binding between pattern and content proplets, and (iii) codes the determiner function of quantifiers as values of the cat and sem attributes.

With the elimination of quantifiers, DBS can restrict the use of all kinds of variables to the definition of *pattern* proplets. The pattern proplets are combined into schemata and are used for matching with content by the LA-grammar rules and for retrieval (activation) in the content-addressable memory of a Word Bank. Using restricted variables as proplet values is a simple, powerful method of under-specification,[22] with many uses in DBS.

Up to this point, DBS may be regarded as a modern reconstruction of predicate calculus, though with special emphasis on automatic natural language interpretation, processing of content, and natural language production – rather than on truth, satisfiability, consistency, and completeness. The latter are important, but they are neither intended nor sufficient for building a model of natural language communication.

Finally, let us consider meaning, which is what semantics is all about. Logicians take great care to give their formulas a semantic interpretation. It is a conceptual construction in which "the world" is defined in principle as a set-theoretical model structure, and the surfaces of the variables and constants of a formula are defined by hand to refer to these set-theoretical constructs using a metalanguage. The purpose is inferencing and theorem proving.

Instead of treating concepts in terms of metalanguage definitions, DBS treats concepts as the basic *procedures* of the agent's recognition and action. Declarative representations of these procedures are reused[23] as the meanings of language, with the type/token relation serving in pattern matching (4.3.3). This procedural approach to concepts and meanings demands the switch from a

[21] The values exh, sel, def, indef, sg, and pl are defined set-theoretically in NLC'06, 6.2.9.

[22] Restricted variables are inherently open-ended. When used for matching, variables may be bound tentatively to values not in their restriction set. Also, when new selectional constellations (Chap. 8) have been found, these must be added to the relevant restriction sets.

[23] Cf. Roy 2005 for a similar approach.

sign-oriented [-sense, -constructive] ontology to an agent-oriented [+sense, +constructive] ontology (cf. FoCL'99, Sect. 20.4).[24]

12.5 Autonomous Control

An agent-oriented approach requires an autonomous control for sensible behavior, including natural language production. In DBS, autonomous control is driven by the principle of balance. A state of balance is an absolute like truth, and like truth it provides the fix point necessary for a system of semantic interpretation. But while truth is (at least[25]) bipolar, balance is monopolar. This may be the reason why balance, unlike truth, is a dynamic principle, suitable for driving the cognitive agent trying to survive, and to survive comfortably, in a constantly changing world, short-, mid-, and long-term.

In DBS, the agent's search for balance is implemented as a set of inferences. An inference is triggered by an activated content matching the antecedent. After binding the variables of the antecedent to constants in the input, e.g., core values, the consequent of the inference derives a blueprint for action as output. The output is written to the *now front* of the agent's Word Bank.

This new approach to inferencing is based on the belated insight that the database schema of a Word Bank, designed before FoCL'99, is in fact a *content-addressable* memory.[26] Content-addressable memories happen to be the most efficient for content written once and never changed. By structuring the memory like *sediment*, written once and never changed, all processing for real-time behavior control is restricted to the *now front*.

Stored content which never changes makes practically[27] no processing demands on the system. At the same time, the personal history contained in an agent's static sediment provides an individual notion of what is true and what is right. The agent's current state is defined by the most recent data, stored last (rightmost) in the token lines.

The agent's overall moment to moment behavior may be viewed as regulated by of a basket of weights which represent the current options for maintaining

[24] The binary feature notation without attributes used here, e.g., [+constructive], resembles the "feature bundles" of Chomsky and Halle (1968).

[25] Cf. FoCL, Sect. 20.5.

[26] Originally, the Word Bank had been naively derived from *classic* (i.e., record-based) *network databases* as described in Elmasri and Navathe (1989). Driven by functional concerns, our interest was focused primarily on a running program, working as intended. Over the years, the implementation of a Word Bank has been explored in a sequence of at least four projects at the CLUE.

[27] With the exception of occasional cleanups, cf. Sect. 5.6.

or regaining balance.[28] Which blueprint for action is selected for realization is determined by continuously recalculating the weights associated with the agent's current needs and the consequences of available options. The calculation is based on a cost-benefit analysis, computed in real-time.

Fine-tuning the DBS system to simulate the balance of a robot in a terrain, i.e., in a co-designed changing environment, requires an actual robot. After all, without the robot's external and internal interfaces we would have to recreate every nook and cranny of the changing environment by hand (as in Model Theory, cf. 4.3.1). This would violate a basic principle of nouvelle A.I., namely that *The world is its own best model* (Brooks 1989).

To manage the massive, multiple, parallel search required for operating a DBS robot in real time, retrieval must be based on efficient database operations and provide highly differentiated recall.[29] This task may and must be solved theoretically, i.e., without any need for actual robot hardware. The following aspects may be distinguished: (i) the organization of competing retrieval tasks at any given moment and (ii) the quality of the search mechanism itself.

For the synchronization of competing retrieval tasks, DBS uses the time-linear derivation order (cf. Herlihy and Shavit, in press, for related issues). All parallel derivation strands apply their current step, e.g., a rule application, simultaneously. An example of time-linear derivation strands running in parallel is the simultaneous operation of (i) the hear mode, (ii) subactivation, (iii) intersection, and (iv) inferencing.

In a step, each strand produces a set of retrieval tasks, specified by the DBS rules and database operations to be applied. These jobs may be executed in parallel or sequentially in some suitable order, depending on the hardware and the operating system. This approach to organizing parallel operations is not only efficient but also transparent – which is essential for debugging and optimization of the DBS robot.

The other aspect of optimal retrieval is the quality of the individual search operations. In DBS, it (i) depends on the speed of the retrieval mechanism and (ii) must have the expressive power needed for the kind of queries to be expected. For speed, DBS uses the schema of a content-addressable database (Word Bank) and the use of pointers. For expressive power, DBS utilizes the

[28] As shown in Chaps. 5 and 6, this is based in part on rule-governed and goal-governed inferencing, fixed behavior, and trial and error.

[29] In the hear mode, retrieval is used to determine the correct token line for storing proplets at the *now front*. In the think mode, retrieval must activate a successor proplet somewhere in the Word Bank, using an address. In the speak mode, the content to be realized must be mapped into a sequence of language surfaces, written to the *now front*, which again requires finding the proper token line. Activated content must find inferences with a matching antecedent.

semantic relations of coordination and functor-argument, defined at (a) the content and (b) the schema (rule) levels. Introduced to model the cycle of natural language communication, the semantic relations provide the structural basis also for subactivation, intersection, and inferencing.

Because the content in a Word Bank is structured by the semantic relations of natural language, the system can respond in kind, i.e., it can be as specific or general as formulated in the query or any other search request. This ability to respond to language questions with language answers, developed for natural language dialogue with a talking robot, may also be used for more conventional applications. For example, a database structured as a Word Bank, sitting on a standard computer in some geographically remote warehouse, may be queried, and may answer, in natural language.

12.6 Applications

The DBS approach to practical (commercial) applications of natural language processing is based on solving the most important theoretical question first: *How does the mechanism of natural language communication work?*

To protect against accidentally neglecting some crucial interface, component, or ability, the overall design of a DBS robot aims at functional completeness. By modeling all essential structural aspects of natural language communication by humans it is hoped that there will be no application-motivated requests which cannot be satisfied.

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

Extending the data coverage as a form of upscaling has immediate consequences on commercial applications using the system for their natural language processing needs. Take for example LA-morph, the automatic word form recognition software, running with a certain natural language of choice.

The data coverage of such an instance of LA-morph may be extended by adding to the lexicon and by optimizing the allo- and combi-rules for the natural language at hand. This broadens the base for syntactic-semantic analysis and inferencing. It also provides practical applications with better results for retrieval based on content words.

A second area for completing data coverage is extending the syntactic-semantic analysis. When applied to a new (i.e., previously unanalyzed) natural

language, the LA-hear parser will at first handle only a few constructions. As the language is being studied, more and more constructions (like infinitives, prepositional phrases, relative clauses, etc.) are added to the grammar, tested, and revised. When the LA-hear parser encounters input it cannot yet handle, the passage may be traversed at a lower level of detail until proper parsing can resume (robustness). For this, LA-grammar is especially suitable because it computes possible continuations in a time-linear derivation order.

Expanding syntactic-semantic parsing in the agent's hear mode is more demanding than automatic word form recognition. This effort should not go unrewarded from the application side, however. The coding of functor-argument and coordination extends recall and precision from lexically analyzed word forms to phrases and clauses, and from there to sentences, paragraphs, and text. Technically, this amounts to an extension from matching lexically analyzed content words stored within token lines in the Word Bank, to matching semantic relations between content words defined across token lines.[30]

The think mode is a third area for extending the data coverage. The agent's think mode combines two mechanisms, LA-think and inferencing. The basic mechanism of LA-think is *selective activation* by navigating along the semantic relations in a Word Bank.[31] The navigation is used to activate and report self-contained content.

Inferences are used for deriving the different perspectives of the speaker and the hearer on content,[32] and to compute blueprints for action, including language action. Together with current and stored data, LA-think and inferencing constitute the agent's autonomous control, which has many practical applications, with and without language.

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

[30] In addition, the user may load the proplets in a proprietary database with additional attributes and values as needed for the application. One such application of LA-morph and LA-hear is speech recognition; it could well benefit from the search space reduction resulting from an LA-hear parser computing possible continuations (Sect. 2.4).

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

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

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

While specific applications may benefit selectively from the nurturing of a particular component, all applications will benefit simultaneously from a methodical upscaling of the DBS robot as a whole. An application which does not require certain abilities may be run with a DBS version in which they have been switched off.[33]

The systematic, theory-driven upscaling of a talking robot is of general interest for the following reasons. First, it provides the opportunity to ensure compatibility between the system's components in a declarative manner.[34] Second, the neighboring sciences, for example, psychology, ethology, neurology, philosophy, etc., may use the computational model to test some of their own issues, which may in turn contribute to the long-term effort of upscaling the talking robot.[35] Third, by making regular version updates available to the public, progress in pure research may quasi automatically improve the language processing of participating applications.

The orderly transfer from a continuously improving DBS system to commercial applications of human-machine communication may be illustrated by the following vision. Every year, when the current monitor corpus (Sect. 12.2) has been put through the automatic software grinder of word form recognition, parsing, frequency analysis, and comparison with preceding monitor corpora, the results are used for a software version with improved data coverage.

By making new versions available to paying subscribers for their natural language processing needs, all or most of the research costs may be recovered. For this to work long-term, a new release must not require any labor from the subscriber (e.g., additional personnel training), except for the routine installation. Also, each new version must enhance service directly and noticeably, so that subscribers are attracted and kept in sufficient numbers.

[33] For example, a dialogue system over the phone may omit the ability to read.

[34] In addition, typing (in the sense of computer science) could be used in DBS. It is not really needed, however, because of the simplicity and formal uniformity of proplets and the associated interfaces and algorithm.

[35] For example, a system of control implemented in analogy to the sympaticus/parasympaticus nerve network.

Improvements from one version to the next may be achieved rather easily because there are large fields of empirical data which merely need to be "harvested." The software machine for the systematic collection, analysis, and interpretation of the language data is the DBS robot, originally designed to model the mechanism of natural language communication.

For example, when applied to a new language, the DBS robot's off-the-shelf components for the lexicon, automatic word form recognition, syntactic-semantic parsing, and so on, hold no language-dependent data. As a new language is being analyzed, words are added to the robot's lexicon component, just as compositional structures are added to the LA-Morph, LA-hear, LA-think, and LA-speak grammars in the robot's rule component. Also, culture-dependent content may be added to the Word Bank.

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

For long-term linguistic research, there is no lack of renewable language data, namely (i) the natural changes year to year within the domains of a given language and (ii) a wide, constantly extending range of applications in human-machine communication. In addition, there is (iii) the great number of natural languages not yet charted, or not yet charted completely (including English, in any theory). The harvesting of each of these kinds of data will be of interest to its own group of users.

Charting a new natural language is a standard procedure, but it has to deal with relatively large amounts of data. As more and more languages are analyzed, however, charting is accelerated because software constructs may be reused, based on similarities in lexicalization, in productive syntactic-semantic structures, in collocations, constructions, and idioms, and in inferencing. To better support day-to-day research,[36] these standardized software constructs and their declarative specifications may be stored in system libraries, organized for families of languages.

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