Inferencing in Database Semantics

Roland HAUSSER
Abteilung Computerlinguistik, Universität Erlangen-Nürnberg (CLUE)
Bismarckstr. 6, 91054 Erlangen, Germany
rrh@linguistik.uni-erlangen.de

Abstract. As a computational model of natural language communication, Database Semantics[1] (DBS) includes a hearer mode and a speaker mode. For the content to be mapped into language expressions, the speaker mode requires an autonomous control. The control is driven by the overall task of maintaining the agent in a state of balance by connecting the interfaces for recognition with those for action.

This paper proposes to realize the principle of balance by sequences of inferences which respond to a deviation from the agent's balance (trigger situation) with a suitable blueprint for action (countermeasure). The control system is evaluated in terms of the agent's relative success in comparison other agents and the absolute success in terms of survival, including the adaptation to new situations (learning).

From a software engineering point of view, the central question of an autonomous control is how to structure the content in the agent's memory so that the agent's cognition can precisely select what is relevant and helpful to remedy a current imbalance in real time. Our solution is based on the content-addressable memory of a Word Bank, the data structure of proplets defined as non-recursive feature structures, and the time-linear algorithm of Left-Associative grammar.

Introduction

Designing an autonomous control as a software system requires a functional principle to drive it. Following earlier work such as [Bernard 1865] and [Wiener 1948], DBS control is based on the principle of *balance*, i.e., it is designed to maintain the agent in a steady state (equilibrium, homeostasis) relative to a continuously changing external and internal environment, short-, mid-, and long-term.[2] In this way, changes of the environment are utilized as the main motor activating the agent's cognitive operations.

The balance principle guides behavior towards daily survival in the agent's ecological niche. Behavior driven by instinct and by human desires not directly related to survival, such as power, love, belonging, freedom, and fun, may also be subsumed under the balance principle by treating them as part of the internal environment – like hunger.

The agent's balancing operations provide the foundation for a computational reconstruction of *intention* in DBS, just as the agent's recognition and action procedures provide the foundation for a computational reconstruction of *concepts* and of *meanings*

[1] For an introduction to DBS see [NLC'06]. For a concise summery see [Hausser 2009a].

[2] Though conceptually much different from previous and current approaches to autonomous control, our mechanism is closer in spirit to circular causal systems in ecology [Hutchinson 1948] than to the more recent systems of control with a stratified architecture structured into the levels of organization, coordination, and execution [Antsaklis and Passino 1993].

(cf. [AIJ'01]). This differs from [Grice 1965], who bases his notion of meaning on an elementary (undefined, atomic) notion of intention – which is unsuitable for computation.[3] An autonomous control maintaining a balance by relating recognition to the evaluated outcome of possible reactions is decentralized,[4] in line with [Brooks 1985].

1. Inferences of Database Semantics

Maintaining the agent in a state of balance is based on three kinds of DBS inference, called R(eactor), D(eductor), and E(ffector) inferences.[5] R inferences are initiated by a trigger provided (i) by the agent's current external or internal recognition or (ii) by currently activated memories (subactivation, cf. Sect. 6). D and E inferences, in contrast, are initiated by other already active inferences, resulting in *chaining*. As a first, simple method of chaining, let us assume that the consequent of inference `n` must equal the antecedent of inference `n+1`.

R(eactor) inferences provide a response to actual or potential deviations from the agent's balance (cf. 1.1, 4.1, 12.1). A given trigger automatically initiates exactly those R inferences which contain the trigger concept, e.g., *hot* or *hungry*, in their antecedent.

D(eductor) inferences establish semantic relations of content, and are illustrated by summarizing (cf. 3.2), downward traversal (cf. 10.1), and upward traversal (cf. 10.4). Other kinds of D inferences are *precondition* and *cause and effect*. Triggered initially by an R inference, a D inference may activate another D inference or an E inference.

E(ffector) inferences provide blueprints for the agent's action components.[6] Because E inferences connect central cognition with peripheral cognition, their definition has to be hand-in-glove with the robotic hardware they are intended to control.

The interaction of reactor, deductor, and effector inferences is illustrated by the following chain, using English rather than the formal data structure of proplets[7] for simplicity:

1.1. CHAINING R, D, AND E INFERENCES

1. R: β is hungry **cm** β eats food.
2. D: β eats food **pre** β gets food.
3. D: β gets food $\Downarrow$ β gets α, where $\alpha \in$ {apple, pear, salad, steak}.
4. E: β gets α **exec** β locates α at γ.
5. E: β locates α at γ **exec** β takes α.
6. E: β takes α **exec** β eats α.
7. D: β eats $\alpha \Uparrow \beta$ eats food.

Step 1 is an R inference with the connective **cm** (for countermeasure) and triggered by a sensation of hunger. Step 2 is a D inference with the connective **pre** (for precondition),

[3] Cf. [FoCL'99], Sect. 4.5, Example II.

[4] The cooperative behavior of social animals, e.g., ants in a colony, may also be described in terms of balance.

[5] This terminology is intended to distinguish DBS inferences from the inferences of symbolic logic. For example, while a deductive inference like modus ponens is based on form, the deductor inferences of DBS take content into account.

[6] In robotics, effectors range from legs and wheels to arms and fingers. The E inferences of DBS should also include gaze control.

[7] Proplets are defined as non-recursive (flat) feature structures and serve as the basic elements of propositions. Like the cell in biology, the proplet is a fundamental unit of structure, function, and organization in DBS.

while step 3 is the D inference for downward traversal with the connective ⇓ (cf. 10.1). Steps 4, 5, and 6 are E inferences with the connective **exec** (for execute).

Step 4 may be tried iteratively for the instantiations of food provided by the consequent of step 3 (see the restriction on the variable α). If the agent cannot locate an apple, for example, it tries next to locate a pear, etc. Individual food preferences of the agent may be expressed by the order of the elements in the variable restriction.

Step 7 is based on the D inference for upward traversal with the connective ⇑ (cf. 10.4). This step is called the *completion* of the chain because the consequent of the inference equals the consequent of step 1. The completion indicates the successful execution of the countermeasure to the imbalance indicated by the antecedent of the initial reactor inference.

2. Coreference-by-Address

The implementation of DBS inferences depends on the DBS memory structure. Called Word Bank, it is content-addressable[8] in that it does not require a separate index (inverted file) for the storage and retrieval of proplets. A content-addressable memory is especially suitable for fixed content, i.e., content is written once and never changed. This provides a major speed advantage over the more widely used coordinate-addressable memory (as in a relational database) because internal access may be based on pointers enabling direct access to data.

In DBS, the requirement of fixed content is accommodated by adding content instead of revising it, and by connecting the new content to the old by means of pointers. Consider, for example, a cognitive agent observing at moment t_i that *Julia is sleeping* and at t_j that *Julia is awake*, referring to the same person. Instead of representing this change by revising the first proposition into the second,[9] the second proposition is added as new content, leaving the first proposition unaltered:

2.1. COREFERENTIAL COORDINATION IN A WORD BANK STORING PROPLETS

	member proplets	*owner proplets*
… [noun: Julia, fnc: sleep, prn: 675] …	[noun: (Julia 675), fnc: wake, prn: 702]	… [core: Julia]
…		
… …	[verb: wake, arg: (Julia 675), prn: 702]	… [core: wake]
…		
… [verb: sleep, arg: Julia, prn: 675] …		… [core: sleep]

In a proplet, the part-of-speech attribute, e.g., noun or verb, is called the core attribute and its value is called the core value. A Word Bank stores proplets with equivalent core values in the same token line in the order of their arrival. The occurrence of Julia in the

[8] See [Chisvin and Duckworth 1992] for an overview.

[9] A more application-oriented example would be *fuel level high* at t_i and *fuel level low* at t_j.

second proposition is represented by a proplet with a core attribute containing an address value, i.e., [noun: (Julia 675)], instead of a regular core value, e.g., [noun: Julia].

Coreference-by-address enables a given proplet to code as many semantic relations to other proplets as needed. For example, the proplets representing *Julia* in 2.1 have the fnc value sleep in proposition 675, but wake in proposition 702. The most recent (and thus most up-to-date) content relating to the original proplet is found by searching the relevant token line from right to left, i.e., in the anti-temporal direction.

Coreference-by-address combines with the semantic relations of functor-argument and coordination structure, as in the following example:

2.2. COREFERENCE-BY-ADDRESS CONNECTING NEW TO OLD CONTENT

$$\begin{bmatrix}\text{verb: sleep}\\ \text{arg: Julia}\\ \text{prn: 675}\end{bmatrix} \overset{1}{\leftrightarrow} \begin{bmatrix}\text{noun: Julia}\\ \text{fnc: sleep}\\ \text{prn: 675}\end{bmatrix} \overset{2}{\leftarrow} \begin{bmatrix}\text{noun: (Julia 675)}\\ \text{fnc: wake}\\ \text{prn: 702}\end{bmatrix} \overset{3}{\leftrightarrow} \begin{bmatrix}\text{verb: wake}\\ \text{arg: (Julia 675)}\\ \text{prn: 702}\end{bmatrix}$$

The connections 1 and 3 are intrapropositional and based on the functor-argument relations between *Julia* and *sleep*, and *Julia* and *wake*, respectively. Connection 2 is extrapropositional and based on the coreference between the pointer proplet of proposition 702 and the original *Julia* proplet of proposition 675.[10] One way to realize 2.2 in English would be Julia was asleep. Now she is awake.

3. Inference for Creating Summaries

Coreference-by-address allows not only (i) to revise the fixed information in a content-addressable memory by extending it, as in 2.1, but also (ii) to derive new content from stored content by means of inferencing. One kind of DBS inference is condensing content into a meaningful summary. As an example, consider a short text, derived in detail in Chapts. 13 (hearer mode) and 14 (speaker mode) of [NLC'06]:

> The heavy old car hit a beautiful tree. The car had been speeding. A farmer gave the driver a lift.

A reasonable summary of this content would be *car accident*. This summary may be represented in the agent's Word Bank as follows:

3.1. RELATING SUMMARY TO TEXT

		member proplets		*owner proplets*
...				
		[noun: accident mdr: (car 1) prn: 67]	...	[core: accident]
...				
...	[noun: car fnc: hit prn: 1] [noun: (car 1) fnc: speed prn: 2]	... [noun: (car 1) mdd: accident prn: 67]	...	[core: car]
...				

[10] In its basic form, coreference-by-address is one-directional, from the pointer proplet to the original. The inverse direction may be handled by building an additional index. As usual, the proplets in 2.2 are order-free. During language production, an order is re-introduced by navigating from one proplet to the next.

$$\ldots \begin{bmatrix}\text{verb: hit}\\ \text{arg: car tree}\\ \text{nc: 2 speed}\\ \text{pc:}\\ \text{prn: 1}\end{bmatrix} \ldots \qquad \ldots \big[\text{core: hit}\big]$$

$$\ldots \begin{bmatrix}\text{verb: speed}\\ \text{arg: (car 1)}\\ \text{pc: 1 hit}\\ \text{nc: 3 give}\\ \text{prn: 2}\end{bmatrix} \qquad \ldots \big[\text{core: speed}\big]$$

$\ldots$

Propositions 1 and 2 are connected (i) by adjacency-based coordination coded in the nc (next conjunct) and pc (previous conjunct) attribute values of their verb proplets *hit* and *speed*, and (ii) by coreferential coordination based on the original *car* proplet in proposition 1 and the corresponding pointer proplet in proposition 2.

The summary consists of another *car* pointer proplet and the *accident* proplet, each with the same prn value (here 67) and related to each other by the modifier-modified relation. The connection between the summary and the original text is based on the address value (car 1), which serves as the core value of the rightmost *car* proplet as well as the mdr (modifier) value of the *accident* proplet.

The summary-creating inference deriving the new content with the prn value 67 is formally defined as the following D(eductor) inference rule, shown with the sample input and output of 3.1 at the content level:

3.2. SUMMARY-CREATING D INFERENCE

antecedent *consequent*

rule level

$$\begin{bmatrix}\text{noun: } \alpha\\ \text{fnc: hit}\\ \text{prn: K}\end{bmatrix} \begin{bmatrix}\text{verb: hit}\\ \text{arg: } \alpha\ \beta\\ \text{prn: K}\end{bmatrix} \begin{bmatrix}\text{noun: } \beta\\ \text{fnc: hit}\\ \text{prn: K}\end{bmatrix} \Rightarrow \begin{bmatrix}\text{noun: } (\alpha\ \text{K})\\ \text{mdd: accident}\\ \text{prn: K+M}\end{bmatrix} \begin{bmatrix}\text{noun: accident}\\ \text{mdr: } (\alpha\ \text{K})\\ \text{prn: K+M}\end{bmatrix}$$

where $\alpha \in$ {car, truck, boat, ship, plane, ...} and $\beta \in$ {tree, rock, wall, mountain, ...} $\cup\ \alpha$

matching and binding

content level

$$\begin{bmatrix}\text{noun: car}\\ \text{fnc: hit}\\ \text{prn: 1}\end{bmatrix} \begin{bmatrix}\text{verb: hit}\\ \text{arg: car tree}\\ \text{nc: 2 speed}\\ \text{pc:}\\ \text{prn: 1}\end{bmatrix} \begin{bmatrix}\text{noun: tree}\\ \text{fnc: hit}\\ \text{prn: 1}\end{bmatrix} \quad \begin{bmatrix}\text{noun: (car 1)}\\ \text{mdd: accident}\\ \text{prn: 67}\end{bmatrix} \begin{bmatrix}\text{noun: accident}\\ \text{mdr: (car 1)}\\ \text{prn: 67}\end{bmatrix}$$

input *output*

The rule level shows two sets of pattern proplets, called the antecedent and the consequent, and connected by the operator $\Rightarrow$. Pattern proplets are defined as proplets with variables as values, while the proplets at the content level do not contain any variables. The consequent pattern uses the address (or pointer, cf. Sect. 2) value (α K) to relate to the antecedent and has the new prn value K+M, with M $>$ 0.

In the rule, the possible values which α and β may be bound to during matching are restricted by the co-domains of these variables: the restricted variable α generalizes the summary-creating inference to different kinds of accidents, e.g., *car accident, truck accident*, etc., while the restricted variable β limits the objects to be hit to trees, rocks, etc., as well as cars, trucks, etc. Any content represented by the proplet *hit* with a subject

and an object proplet satisfying the variable restrictions of α and β, respectively, will be automatically (i) summarized as an accident of a certain kind whereby (ii) the summary is related to the summarized by means of an address value, here (car 1), thus fulfilling the condition that the data in a content-addressable memory may not be modified.

By summarizing content into shorter and shorter versions, there emerges a hierarchy which provides retrieval relations for upward or downward traversal (cf. Sect. 10). An upward traversal supplies more and more general notions, which may be used by the agent to access inferences defined at the higher levels. A downward traversal supplies the agent with more and more concrete instantiations.

4. Horizontal and Vertical Aspects of Applying DBS Inferences

DBS inferences are defined as formal rules which are applied to content in the agent's Word Bank by means of pattern matching. As a software operation, such an application may be divided into phases which happen to have horizontal and vertical aspects. The horizontal aspect concerns the relation between the antecedent and the consequent of an inference and the chaining of inferences. The vertical aspect concerns the relation between the rule level and the content level, within an inference and in a chain of inferences.

Consider the formal definition of the first inference in 1.1, applied to a suitable content:

4.1. FORMAL DEFINITION OF THE *hungry-eat* R(EACTOR) INFERENCE

	antecedent			*consequent*		
rule level	noun: β fnc: hungry prn: K	verb: hungry arg: β prn: K	**cm**	noun: (β K) fnc: eat prn: K+M	verb: eat arg: (β K) food prn: K+M where $0 < \mathrm{M} < \theta$	noun: food fnc: eat prn: K+M
	matching and binding					
content level	noun: Julia fnc: hungry prn: 211	verb: hungry arg: Julia prn: 211		noun: (Julia 211) fnc: eat prn: 220	verb: eat arg: (Julia 211) food prn: 220	noun: food fnc: eat prn: 220

The upper bound θ is intended to ensure that the content of the consequent closely follows the content of the antecedent. Furthermore, the inclusion of the antecedent's subject in the consequent by means of the address value (β K) excludes cases in which one agent is hungry and another one eats food – which would fail as an effective countermeasure.

The rule application starts with the vertical grounding of the antecedent in the trigger situation by matching and binding. Next there is the horizontal relation between the grounded antecedent and the consequent, which formalizes a countermeasure (**cm**) connected to the antecedent and its trigger situation. Finally, the patterns of the consequent vertically derive a new content as a (preliminary) blueprint for action which may horizontally activate another inference, as shown in 1.1.

5. Schema Derivation and Intersection

The sets of connected pattern proplets constituting the antecedent and the consequent of an inference like 3.2 or 4.1 are each called a DBS schema. Schemata are used in

general for retrieving (visiting, activating) relevant content in a Word Bank. A schema is derived from a content, represented as a set of proplets, by simultaneously substituting all occurrences of a constant with a restricted variable. Consider the following example of a content:

5.1. PROPLETS CODING THE CONTENT OF Julia knows John.

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

This representation characterizes functor-argument structure in that the *Julia* and *John* proplets[11] specify know as the value of their fnc attributes,[12] and the *know* proplet specifies Julia and John as the values of its arg attribute. The content may be turned into a schema by replacing its prn value 625 with the variable K, restricted to the positive integers. This schema will select all propositions in a Word Bank with a content equivalent to 5.1

The set of proplets matched by a schema is called its *yield*. The yield of a schema relative to a given Word Bank may be controlled precisely by two complementary methods. One is by the choice and number of constants in a content which are replaced by restricted variables. For example, the following schema results from replacing the constants Julia, John, and 625 in content 5.1 with the variables α, β, and K, respectively:

5.2. POSSIBLE SCHEMA RESULTING FROM 5.1

noun: α	verb: know	noun: β
fnc: know	arg: α β	fnc: know
prn: K	prn: K	prn: K

The yield of this schema are all contents in which someone knows someone. However, if only John and 625 in content 5.1 are replaced by variables, the resulting schema has a smaller, more specific yield, namely all contents in which Julia knows someone.

When a schema with several pattern proplets is used as a query, its yield is obtained by "intersecting" the token lines corresponding to the pattern proplets' core values (provided the latter are constants). As an example, consider the schema for *hot potato*:

5.3. SCHEMA FOR *hot potato*

adj: hot	noun: potato
mdd: potato	mdr: hot
prn: K	prn: K

The functor-argument structure of this example (consisting of a modifier and a modified) is a schema because the prn value is the variable K. Applying the schema to the corresponding token lines in the following example results in two intersections:

[11] When we refer to a proplet by its core value, we use Italic, e.g., *John*.

[12] When we refer to an attribute or a value within a proplet, we use Helvetica, e.g., fnc or know.

5.4. INTERSECTING TOKEN LINES FOR *hot* AND *potato*

	member proplets				*owner proplets*
…	**adj: hot** **mdd: potato** **prn: 20**	adj: hot mdd: water prn: 32	**adj: hot** **mdd: potato** **prn: 55**	adj: hot mdd: day prn: 79	[core: **hot**]
…					
…	**noun: potato** **fnc: look_for** **mdr: hot** **prn: 20**	noun: potato fnc: cook mdr: big prn: 35	**noun: potato** **fnc: find** **mdr: hot** **prn: 55**	noun: potato fnc: eat mdd: small prn: 88	[core: **potato**]

The intersections contain the proplets with the prn values 20 and 55. They are selected because the pattern proplets of schema 5.3 match only *hot* proplets with the mdd (modified) value potato and only *potato* proplets with the mdr (modifier) value hot.

The other method to control and adjust the yield of a schema is in terms of the restrictions on the variables. Restrictions may consist in an explicit enumeration of what a variable may be bound to (cf. 3.2). Restrictions may also be specified by constants, like vehicle or obstacle, which lexically provide similar sets as the enumeration method by using a thesaurus, an ontology, WordNet, or the like.

The two methods of fine-tuning a DBS schema result in practically[13] perfect recall and precision. This is crucial for autonomous control because the effective activation of relevant data is essential for the artificial agent to make good decisions.

6. Subactivation (Selective Attention)

In DBS, the selection of content by means of schemata is complemented by the equally powerful method of subactivation: the concepts provided by recognition and inferencing are used as a continuous stream of triggers which select corresponding data in the Word Bank. As an example, consider the following subactivation of a token line:

6.1. TRIGGER CONCEPT SUBACTIVATING A CORRESPONDING TOKEN LINE

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

Subactivation is an automatic mechanism of association,[14] resulting in a mild form of selective attention. It works like a dragnet, pulled by the incoming concepts serving as triggers and accompanying them with corresponding experiences from the agent's past.

Intuitively, subactivation may be viewed as highlighting an area of content at half strength, setting it off against the rest of the Word Bank, but such that exceptional evaluations (cf. Sect. 8) are still visible as brighter spots. In this way, the agent will be alerted to potential threats or opportunities even in current situations which would otherwise seem innocuous – resulting in virtual triggers for suitable inferences.

[13] Recall and precision are defined in terms of subjective user satisfaction. Cf. [Salton 1989].

[14] Like associating a certain place with a happy memory.

The primary subactivation 6.1 may be extended into a secondary and tertiary one by spreading activation[15] [Quillian 1968]. For example, using the semantic relations coded by the left-most proplet in 6.1, the following proposition may be subactivated, based on the continuation and prn values potato 20, look_for 20, and John 20:

6.2. SECONDARY SUBACTIVATION OF A PROPOSITION

noun: John fnc: look_for prn: 20	verb: look_for arg: John, potato pc: cook 19 nc: eat 21 prn: 20	noun: potato fnc: look_for mdr: hot prn: 20	**adj: hot** **mdd: potato** **prn: 20**

While a secondary subactivation utilizes the intrapropositional relations of functor-argument and coordination structure (cf. [NLC'06], Chapts. 6 and 8), a tertiary subactivation is based on the corresponding extrapropositional relations (cf. [NLC'06], Chapts. 7 and 9). For example, using the pc (previous conjunct) and nc (next conjunct) values of the *look_for* proplet in 6.2, the tertiary subactivation may spread from John looked for a hot potato to the predecessor and successor propositions with the verb values cook and eat, and the prn values 19 and 21, respectively.

7. Semantic Relations

Subactivation may spread along any semantic relations between proplets. By coding the semantic relations inside and between propositions solely as proplet-internal values, proplets become order-free and are therefore suitable for efficient storage and retrieval in the content-addressable memory of a Word Bank. Subactivation is made especially efficient by coding the semantic relations as pointers (cf. Sect. 2).

In DBS, the semantic relations are of two kinds, (i) form and (ii) content. The semantic relations of *form* are functor-argument and coordination structure, intra- and extrapropositionally; they are established during recognition and are utilized in the encoding of blueprints for action. In natural language communication, for example, the semantic relations of grammatical form are established in the hearer mode (recognition) and encoded in the speaker mode (action).

The semantic relations of *content* are exemplified by cause and effect, precondition, the semantic hierarchies, etc. Content relations have been used to define associative (or semantic) networks (cf. [Brachman 1979] for an overview). In DBS, semantic relations of content are established by inferences.

The topic of semantic relations in general and of content relations in particular is widely discussed in linguistics, psychology, and philosophy. Content relations in lexicography, for example, are classified in terms of *synonymy, antonymy, hypernymy, hyponymy, meronymy,* and *holonymy*. In philosophy, content relations are viewed from a different perspective, described by [Wiener 1948], p. 133, as follows:

> According to Locke, this [i.e., the subactivation of ideas, R.H.] occurs according to three principles: the principle of contiguity, the principle of similarity, and the principle of cause and

[15] In fiction, our notion of triggering a spreading subactivation is illustrated by the madeleine experience of [Proust 1913], which brings back an almost forgotten area of what he calls "l'édifice immense du souvenir."

> effect. The third of these is reduced by Locke, and even more definitely by Hume, to nothing but constant concomitance, and so is subsumed under the first, contiguity.

Formal examples of semantic relations of content in DBS are the summary inference 3.2, the *hungry-eat* inference 4.1, and the hierarchy inferences for downward traversal 10.1 and for upward traversal 10.4. DBS inferences serve not only to maintain the agent's balance, but also code a kind of knowledge which is different from a content like 5.1.

8. Evaluation of Content

If a cognitive agent were to value all subactivated contents the same, they would provide little guidance towards successful behavior – neither absolute in terms of the agent's survival nor relative in comparison to other agents. Even the path of daily routine, of least resistance, or of following some majority is ultimately the result of choices based on evaluation.

As a general notion, content evaluation has been investigated in philosophy, linguistics, psychology, and neurology. In today's natural language processing, it has reappeared as the *sentiment detection* of data mining [Turney 2002]. In modern psychology, evaluation is analyzed in *emotion theory* [Arnold 1993] and in *appraisal theory* [Lazarus and Lazarus 1994].

For a software model of control, evaluations are not so much a question of how they are expressed or which of them are universal,[16] but how they are assigned internally by individual agents. In DBS, evaluations are assigned when new content is read into the agent's Word Bank – by recognition or by inference. At their lowest level, recognition-based evaluations must be integrated into the agent's hardware (else they would be figments of imagination). For example, hot and cold require a sensor for temperature.

Evaluations have been classified in terms of *joy, sadness, fear,* or *anger*, and are expressed in terms of *good* vs. *bad*, *true* vs. *false*, *excellent* vs. *poor*, *virtuous* vs. *depraved*, *brave* vs. *cowardly*, *generous* vs. *cheap*, *loyal* vs. *treacherous*, *desirable* vs. *undesirable*, *acceptable* vs. *unacceptable*, etc. For guiding the autonomous control of a cognitive agent, DBS uses the features [eval: attract] and [eval: avoid]. They are of a more basic and more neutral nature, and fit into the data structure of proplets. Their values may be scalar and may be set between neutral (0) and the extremes asymptotically approaching -1 or +1.

The overall purpose of DBS evaluation is to record (i) any actual deviation from the agent's state of balance, (ii) any impending threat to the agent's balance, and (iii) any possibility to secure positive aspects of maintaining the agent's balance mid- and long-term. Each is used as a trigger for selecting an inference which provides an appropriate reaction. For example, if it is too hot (evaluation-based trigger), go to where it is cooler (inference-based reaction).

9. Adaptation and Learning

The mechanism of deriving and adjusting DBS schemata (cf. Sect. 5) holds at a level of abstraction which applies to natural and artificial agents alike. Because of the simplicity

[16] Cf. [Darwin 1872], Chapt. XIV, pp. 351–360.

of this mechanism, artificial agents may be designed like natural agents in that they adjust automatically over time. Thereby, the following differences between natural and artificial agents do not stand in the way:

In natural agents, adjusting to a changing environment as well as optimizing come in two varieties, (i) the biological *adaptation* of a species in which physical abilities and cognition are co-evolved, and (ii) the *learning* of individuals which is mostly limited to cognition. Adaptation and learning differ also in that they apply to different ranges of time and different media of storage (gene memory vs. brain memory).

In artificial agents, in contrast, improvement of the hardware is the work of engineers, while development of an automatically adjusting cognition is the work of software designers. Because of this division between hardware and software, the automatic adjustment of artificial agents corresponds more to learning than to adaptation. Fortunately, the absence of natural inheritance in artificial agents may be easily compensated by copying the cognition software (including the artificial agent's experiences and adaptations) from the current hardware model to the next.

The DBS mechanism underlying adaptation as well as learning is based on (i) deriving schemata from sets of content proplets[17] by replacing constants with variables and on (ii) adjusting the restrictions of the variables (cf. Sect. 5). This mechanism may be automated based on the frequency of partially overlapping contents:

9.1. A SET OF CONTENTS WITH PARTIAL OVERLAP

Julia eats an apple
Julia eats a pear
Julia eats a salad
Julia eats a steak

For simplicity, the propositions are presented in English rather than by corresponding sets of proplets.

Because of their partial overlap, the propositions may be automatically summarized as the following schema:

9.2. SUMMARIZING THE SET 9.1 WITH A SCHEMA

Julia eats α, where $\alpha \epsilon$ {apple, pear, salad, steak}

Due to the restriction on the variable α, 9.2 is strictly equivalent to 9.1.

The next step is to replace α by a concept serving as a hypernym, here *food*:

9.3. REPLACING THE RESTRICTED VARIABLE BY A HYPERNYM

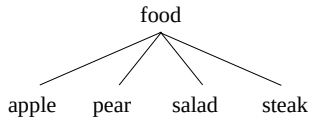
Julia eats food, where food ϵ {apple, pear, salad, steak}

This concept may serve as the literal meaning of the word food in English, aliment in French, Nahrung in German, etc. (cf. [Hausser 2009b]).

Implicit in the content of 9.3 is the following semantic hierarchy:

[17] Content proplets consist of context proplets and language proplets (cf. [NLC'06], Sect. 3.2). Language proplets consist of unconnected lexical proplets (e.g., [NLC'06], 5.6.1) and the connected proplets of language-based propositions (e.g., [NLC'06], 3.2.4).

9.4. REPRESENTING THE SEMANTIC HIERARCHY IMPLICIT IN 9.3 AS A TREE



The automatic derivation of a semantic hierarchy illustrated in 9.1 – 9.3 is empirically adequate if the resulting class containing the instantiations corresponds to that of the surrounding humans. For example, if the artificial agent observes humans to habitually (frequency) eat müsli, the restriction list of α must be adjusted correspondingly.[18] Furthermore, the language surface chosen by the artificial agent for the hypernym concept (cf. 9.3) must correspond to that of the natural language in use.

10. Hierarchy Inferences

An agent looking for food must know that food is instantiated by apples, pairs, salad, or steaks, just as an agent recognizing an apple must know that it can be used as food. In DBS, this knowledge is implemented in terms of inferences for the downward and the upward traversal of semantic hierarchies like 9.4.

For example, if Julia is looking for food, the following downward inference will derive the new content that Julia is looking for an apple, a pear, a salad, or a steak:

10.1. HIERARCHY-INFERENCE FOR DOWNWARD TRAVERSAL

	antecedent		*consequent*
rule level	[noun: food fnc: β prn: K]	$\Downarrow$	[noun: α fnc: (β K) prn: K+M]
	where $\alpha \in$ {apple, pear, salad, steak}		

matching and binding

content level	[noun: Julia fnc: look_for prn: 18]	[verb: look_for arg: Julia food prn: 18]	[noun: food fnc: look_for prn: 18]	[noun: α verb: (look_for 18) prn: 25]

The antecedent consists of a single pattern proplet with the core value food. When this pattern matches a corresponding proplet at the content level, the consequent derives a new content containing the following disjunction[19] of several proplets with core values corresponding to the elements of the restriction set of α:

10.2. OUTPUT DISJUNCTION OF THE DOWNWARD INFERENCE APPLICATION 12.1

[noun: apple *or* fnc: (look_for 18) nc: pear prn: 25]	[noun: pear pc: apple nc: salad prn: 25]	[noun: salad pc: pear nc: steak prn: 25]	[noun: steak pc: salad nc: prn: 25]

[18] This method resembles the establishment of inductive inferences in logic, though based on individual agents.

[19] See [NLC'06], Chapt. 8, for a detailed discussion of intrapropositional coordination such as conjunction and disjunction.

The proplets of the output disjunction are concatenated by the pc (for previous conjunct) and nc (for next conjunct) features, and have the new prn value 25. They are related to the original proposition by the pointer address (look_for 18) serving as the fnc value of the first disjunct. The output disjunction may be completed automatically into the new proposition Julia looks_for apple or pear or salad or steak, represented as follows:

10.3. PROPOSITION RESULTING FROM APPLYING DOWNWARD INFERENCE 12.1

noun: (Julia 18) fnc: (look_for 18) prn: 25	verb: (look_for 18) arg: (Julia 18) apple *or* prn: 25	noun: apple *or* fnc: (look_for 18) nc: pear prn: 25	noun: pear pc: apple nc: salad prn: 25	
noun: salad pc: pear nc: steak prn: 25	noun: steak pc: salad nc: prn: 25			

This new proposition with the prn value 25 is derived from the given proposition with the prn value 18 shown at the content level of 10.1, and related to it by pointer values.

The inverse of downward traversal is the upward traversal of a semantic hierarchy. An upward inference assigns a hypernym like *food* to concepts like *salad* or *steak*. Consider the following definition with an associated sample input and output at the content level:

10.4. HIERARCHY-INFERENCE FOR UPWARD TRAVERSAL

	antecedent				*consequent*
rule level	$\alpha \in$ {apple, pear, salad, steak}	&	noun: α fnc: β prn: K	$\Uparrow$	noun: food fnc: (β k) prn: K+M

matching and binding

content level	noun: Julia fnc: prepare prn: 23	verb: prepare arg: Julia salad prn: 23	noun: salad fnc: prepare prn: 23	noun: food fnc: (prepare 23) prn: 29

Like the downward inference 10.1, the antecedent of the upward inference consists of a single pattern proplet with the restricted variable α as the core value. Due to the use of a pointer address as the fnc value of the output (required anyway by the content-addressable memory of DBS), there is sufficient information to complete the output proplets into the proposition Julia prepares food, with the prn value 29 and pointer proplets for *Julia* and *prepare*.

The limited matching used by the upward and downward inferences has the advantage of generality. The automatic derivation and restriction of schemata (cf. Sect. 9) directly controls the automatic adaptation of the hierarchy inferences. They illustrate how DBS is intended to fulfill the three functions which define an autonomic system: "automatically configure itself in an environment, optimize its performance using the environment and mechanisms for performance, and continually adapt to improve performance and heal itself in a changing environment" [Naphade and Smith 2009].

11. Analogical Models as Blueprints for Action

To obtain a suitable blueprint for an action, the agent may assemble reactor, deductor, and effector inferences creatively into a new chain – which may or may not turn out to be successful. Most of the time, however, it will be easier and safer for the agent to re-use an earlier action sequence, successfully self-performed or observed in others, provided such an analogical model is available in the agent's memory. These earlier models are contained at various levels of detail in the contents subactivated by the initial R inference.

The R inference defined in 4.1, for example, subactivates all contents matching the β is hungry schema (antecedent), the β eats food schema (consequent), as well the token lines of the inference's constants, here hungry, eat, and food. By spreading to secondary and tertiary subactivations (cf. Sect. 6), the initial R inference may subactivate a large set of contents in the agent's Word Bank. These serve to illustrate the trigger situation with a cloud of subactivations (cf. [NLC'06], Sect. 5.6), but their precision is too low as to provide a specific blueprint for practical, goal-directed action.

In order for a content stored in memory to be useful for resolving the agent's current challenge, it must (i) fit the trigger situation as precisely as possible and (ii) have a positively evaluated outcome. For this, our method of choice is DBS intersection (cf. Sect. 5).

Assume that the agent is alone in Mary's house – which serves as a trigger (cf.6.1) subactivating the token line of *Mary* in the agent's Word Bank. Furthermore, the agent is hungry, which triggers the *hungry-eat* inference 4.1. The constant *eat* in the consequent subactivates the corresponding token line, resulting in intersections between the *Mary* and *eat* token lines such as the following:

11.1. EXAMPLE OF TWO *Mary eat* INTERSECTIONS

noun: (Mary 25) fnc: eat prn: 49	verb: eat arg: (Mary 25) **apple** pc: **take 48** prn: 49	noun: (Mary 25) fnc: eat prn: 82	verb: eat arg: (Mary 25) müsli pc: take 81 prn: 82

In other words, the agent remembers Mary once eating an apple and once eating müsli.

The two proplets in each intersection share a prn value, namely 49 and 82, respectively, and are in a grammatical relation, namely functor-argument structure. In both intersections, the verb proplet *eat* provides two continuations. For example, the verb of the first intersection provides the continuation values apple and take 48, which may result in the following secondary and tertiary subactivations (cf. Sect. 6).

11.2. SUBACTIVATION SPREADING FROM *Mary eat* TO *Mary take apple.*

noun: (Mary 25) fnc: eat prn: 49	verb: eat arg: (Mary 25) **apple** pc: **take 48** prn: 49	noun: apple fnc: eat eval: attract prn: 49
noun: (Mary 25) fnc: take prn: 48	verb: take arg: (Mary 25) apple nc: eat 49 pc: **locate 47** prn: 48	noun: apple fnc: take prn: 48

The anti-temporal order corresponds to the spreading direction of the subactivation.

The *apple 49* proplet (secondary subactivation) contains the eval attribute with the value attract. Assuming that the corresponding subactivation for the second intersection happens to evaluate the *müsli 82* proplet as eval: avoid[20] (not shown), the agent would pursue only the tertiary subactivation from the first (and not the second) intersection in 11.1 as a possible candidate for an analogical model for regaining balance.

To get at the information relevant for finding something to eat in Mary's house, the subactivation 11.2 may spread further, based on the pc (for previous conjunct) value locate 47 of the *take 48* proplet. In this way, the subactivation of the earlier eating event may be completed into the following backward sequence of propositions:

11.3. SUBACTIVATED SEQUENCE OF PROPOSITIONS (ANTI-TEMPORAL ORDER)

> Mary eat apple [prn: 49]. Mary take apple [prn: 48]. Mary locate apple in blue cupboard [prn: 47].

The information relevant for the hungry agent is the location from where Mary got the apple, i.e., the blue cupboard.

If the anti-temporal order is reversed, the propositions in 11.3 will match the antecedent of step 5 in Example 1.1 all the way to the consequent of step 7. This completes the chain relative to the consequent of the initial R inference 4.1 at the level of content, obviating steps 1–4 and thus without any assertion that Mary was hungry when she ate the apple.[21]

From the content 11.3 provided by memory via intersection, the agent may obtain an analogical model by (i) reversing the order and (ii) replacing the value Mary with a pointer to the agent, represented as moi:

11.4. RESULTING ANALOGICAL MODEL

> **exec** Moi locate apple in blue cupboard [prn: 102] **exec** Moi take apple [prn: 103] **exec** Moi eat apple [prn: 104] ⇑ Moi eat food [prn: 105]

Whether or not these blueprints for the agent's action components will result in a successful countermeasure depends on whether proposition 102 turns out to hold in the agent's current situation or not.

12. Learning by Imitation

The purposeful subactivation of an earlier content in the Word Bank by means of intersection provides the agent with an analogical model potentially suitable to remedy its current imbalance. For example, instead of looking randomly through Mary's house for something to eat, the agent will begin with searching for an apple in the blue cupboard.

To implement such a system requires an agent with interfaces for recognition and action of a quality not yet available. Therefore, let us consider a simpler example, namely a robot loading its battery at one of several loading stations in its environment. In analogy to 1.1, this behavior may be controlled by the following chain of inferences:

[20] The assumed evaluations reflect the agent's preference of eating apples over eating müsli.

[21] If the agent were to assume (unnecessarily) that Mary must have been hungry, then this would correspond to an abductive inference in logic. The point is that observing Mary eating is sufficient for the purpose at hand.

12.1. AUTONOMOUS CONTROL AS A CHAIN OF R-D-E INFERENCES

1. R: β low battery **cm** β load battery.
2. D: β load battery **pre** β locate station.
3. D: β locate station $\Downarrow$ β locate α, where $\alpha \epsilon$ {1, 2, 3, etc. }.
4. E: β locate α **exec** β attach to α.
5. D: β attach to $\alpha \Uparrow \beta$ attach to station.
6. E: β attach to station **exec** β load battery.

The connectives **cm** (countermeasure), **pre** (precondition), $\Downarrow$ (is instantiated by), $\Uparrow$ (hypernym), and **exec** (execute) are as in 1.1. Steps 3 and 5 show a primitive semantic hierarchy, namely the term station for the instantiations of α. The consequent of step 6 provides completion.

In terms of current technology, each notion used in this software program, e.g., locate, attach, or load, has a rather straightforward procedural counterpart. It is therefore possible even today to build a real robot in a real environment performing this routine.

Instead of programming the robot's operations directly, for example in C or Java, let us use a declarative specification in terms of proplets in a Word Bank. In other words, the robots' recognitions, e.g., locate α, are stored in its Word Bank as sets of proplets and the robot's actions, e.g., attach_to α, are controlled by sequences of proplets.

To simulate learning by imitation, let us use two such robots, called A and B. Initially, each is training in its own environment, whereby A has the loading stations 1 and 2, and B has the loading stations 3, 4, and 5 – with their respective α variables defined accordingly. Once the individual loading routines are well established for both, A is put into the environment of B.

To simplify A's recognition of loading events by B, let us assume that B emits a signal every time it is loading and that A can correctly interpret the signal. In order for A to imitate B, A must follow B, remember the new locations, and adapt A's definition of α to the new environment. The new loading stations may differ in hight, which may cause different efforts of reach, thus inducing preferences (evaluation).

After following B around, A's battery is low. This imbalance triggers step 1 in 12.1. Being in B's environment, A subactivates the token line of B in A's Word Bank, while the consequent of step 1 subactivates the token line of *load*, leading to their intersection – in analogy to 11.1. Spreading results in secondary and tertiary subactivations:

12.2. SUBACTIVATED SEQUENCE OF PROPOSITIONS (ANTI-TEMPORAL ORDER)

B load battery [prn: 69]. B attach to station 3 [prn: 68]. B locate station 3 [prn: 67].

By reversing the spreading order into the temporal order and by replacing B by A, the visiting robot obtains the following blueprints for its action components:

12.3. BLUEPRINTS FOR ACTION

A locate station 3 [prn: 87]. A attach to station 3 [prn: 88]. A load battery [prn: 89].

Except for the replaced subject, these propositions consist of recognition content from A's memory. Therefore, their core values are tokens carrying sensory, motor, and conceptual information which is not provided by the types of the inference chain 12.1, but essential for action blueprints sufficiently detailed to master the situation at hand.

13. Fixed vs. Adaptive Behavior

The behavior of robot A described above is flexible in that it can adapt to different environments of a *known kind*, here two rooms which differ in the number and location of loading stations. In this example, the artificial agents and their artificial environments are co-designed by the engineers.

A more demanding setup is to take a given natural environment and to design a robot able to maintain a balance relative to internal and external changes. This requires (i) analysis of the external environment, (ii) construction of interfaces for the agent's recognition of, and action in, the external environment, and (iii) definition of R(eactor), D(eductor), and E(ffector) inferences for optimal survival.

The ultimate goal, however, is to design a robot with a basic learning software. It should be capable of deriving schemata (cf. Sect. 5) and semantic relations of content (cf. Sect. 7), and of automatically establishing and adapting instantiation classes[22] (cf. Sect. 9). In this way, it should be able to continuously optimize behavior for daily survival in the agent's ecological niche. This may be done in small steps, first testing the artificial agent in artificial environments it was specifically designed for, and then in new environments. By putting the artificial agent into more and more challenging test situations, the control software may be fine-tuned in small steps, by hand and by automatic adaptation.

14. Component Structure and Functional Flow

At any moment in time, the DBS model of a cognitive agent distinguishes three kinds of content: (i) old content stored in the Wordbank, (ii) new content provided by recognition, and (iii) new content provided by inference. Recognition, including language interpretation in the hearer mode, interprets the data stream provided by the external and internal interfaces *non-selectively* and adds the resulting content to the Word Bank.

Inferences, in contrast, are triggered *selectively* by items which match their antecedent. Their derivation of new content is usually based on the subactivation of stored data (cf. Sect.11), and is used as blueprints for action, including language production in the speaker mode. Memories of these actions are added non-selectively[23] to the Word Bank.

The procedures of recognition and of inference are formally based on small sets of connected pattern proplets, called DBS schemata, which operate on corresponding sets of content proplets by means of pattern matching. The matching between individual pattern proplets and content proplets is greatly facilitated by their non-recursive feature structures (cf. [NLC'06], Sect. 3.2). So far, this method has been used for the following cognitive operations:

14.1. COGNITIVE OPERATIONS BASED ON MATCHING

a. *natural language interpretation*:
   matching between LA-hear grammar rules and language proplets (cf. [TCS'92], [NLC'06], Sect. 3.4)

[22] [Steels 1999] presents algorithms for automatically evolving new classes from similar data by abstracting from what they take to be accidental (in the sense of Aristotle).

[23] We are leaving aside the psychological phenomenon of repression (*Unterdrückung*) in natural agents.

b. *navigation*:
matching between LA-think grammar rules and content proplets (cf. [NLC'06], Sect. 3.5, [Hausser 2009a])

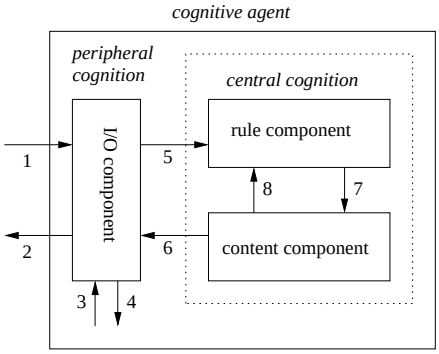
c. *querying*:
matching between query patterns and content proplets (cf. [NLC'06], Sect. 5.1)

d. *inferencing*:
matching between inference rules and content proplets (cf. 3.2, 4.1, 10.1, 10.4).

Navigation (b) and inferencing (d) jointly provide the conceptualization (*what to say?*) and substantial parts of the realization (*how to say it*) for language production.

The different kinds of matching between pattern proplets and content proplets in combination with the agent's cognitive input and output suggest the following component structure:[24]

14.2. COMPONENT STRUCTURE OF A COGNITIVE AGENT



1 = external recognition
2 = external action
3 = internal recognition
4 = internal action
5 = input to rule component
6 = output of content component
7 = rule-content interaction
8 = content-rule interaction

The diagram shows three general components, (i) an I/O (input-output) component for recognition and action, (ii) a rule component for interpretation and production, and (iii) a content component for language and context (or non-language) data.

The separation of patterns and of contents into distinct components provides a uniform structural basis for the rule component to govern the processing of content (7) – with data-driven feedback from the content component (8), including automatic schema derivation (Sect. 9). The rule and the content component are each connected unidirectionally to the I/O component. All recognition output of this I/O component is input to the rule component (5), where it is processed and passed on to the content component (7). All action input to the I/O component comes from the content component (6), derived in frequent (8, 7) interaction with the rule component.

[24]The component structure 14.2 raises the question of how it relates to an earlier proposal, presented in [NLC'06] as diagram 2.4.1. The [NLC'06] diagram models reference in the sense of analytic philosophy and linguistics, namely as a vertical relation between a horizontal language level and a horizontal context level – which is helpful for explaining the Seven Principles of Pragmatics (see [NLC'06], Sect. 2.6, for a summary). In diagram 14.2, this earlier component structure is embedded into the content component.

Technically, the [NLC'06] diagram is integrated into 14.2 by changing to a different view: instead of viewing content proplets as sets with a common prn value (propositions), and separated into a language and a context level, the same proplets are viewed as items to be sorted into token lines according to their core value.

Treating the [NLC'06] diagram as part of the content component in 14.2 serves to explain the separate

Conclusion

Language production in the speaker mode of a cognitive agent raises the question of where the content to be realized should come from. The cycle of natural language communication modeled in DBS answers this question by providing two sources: (i) content provided by recognition, either current or stored in the agent's memory, and (ii) blueprints for action derived on-the-fly by the agent to maintain a state of balance (equilibrium, homeostasis) vis-à-vis a constantly changing external and internal environment.

So far, work on the speaker mode in DBS has concentrated on a systematic description of (i), i.e., production from recognition content (cf. [NLC'06], [Hausser 2009b]). This paper, in contrast, explores the foundations of (ii), i.e., a general solution to providing blue-prints for meaningful actions by the agent, including natural language production. As a consequence, our focus here is on the *what to say* aspect of natural language production (conceptualization) rather than the *how to say it* aspect (realization).

A conceptualization based on a cognitive agent with a memory and interfaces to the external and internal environment is in a principled contrast to a language production for weather reports or query answering for ship locations, train schedules, and the like. The latter are *agentless* applications; they are popular in the research literature because they allow to fudge the absence of an autonomous control. Their disadvantage, however, is that they cannot be extended to agent-based applications such as free dialog [Schegloff 2007], whereas the inverse direction from an agent-based to an agentless application is comparatively easy.

Proceeding on the assumption that a sound theoretical solution to natural language production must be agent-based, this paper shows how an autonomous control based on the principle of balance may be embedded into the cycle of natural language communication as formally modeled and computationally verified in DBS [NLC'06]. Founded technically on a content-addressable memory and coreference-by-address (pointers), this extension of the existing system requires a number of new procedures, such as automatic schema derivation, the subactivation and evaluation of content, adaptation and learning, the definition and chaining of inferences for deriving action blueprints, etc. The resulting conceptual model of a cognitive agent is summarized by showing the basic components and the functional flow connecting the interfaces for recognition with those for action.

To bring across the basic ideas, the presentation tries to be as intuitive as possible. Nevertheless, the formal illustrations of contents, patterns, rules, intersections, etc., provide the outline of a declarative specification for a straightforward transfer into efficiently running code.

Acknowledgements

This paper benefitted from comments by Johannes Handl, Thomas Proisl, Besim Kabashi, and Carsten Weber, research and teaching associates at the Abteilung für Computer-Linguistik Uni Erlangen (CLUE).

input-output channels for the language and the context component in the earlier diagram: The I/O component of 14.2 provides the rule component with a (usually clear) distinction between language and non-language surfaces, resulting in a distinction between language proplets and context proplets during lexical lookup [Handl et al. 2009]. Therefore, the input channel to the content component 7 and the output channel 8 may each be divided into a part for language proplets and a part for context proplets.

References

[AIJ'01] Hausser, R. (2001). Database Semantics for natural language, *Artificial Intelligence*, 130.1:27–74, Elsevier. Available online at http://www.linguistik.uni-erlangen.de/clue/de/publikationen.html

[Anderson 1983] Anderson, J. R. (1983). A spreading activation theory of memory, *Journal of Verbal Learning and Verbal Behavior*, 22:261-295

[Antsaklis and Passino 1993] Antsaklis, P.J., and K. M. Passino, eds. (1993). *An Introduction to Intelligent and Autonomous Control*, Dordrecht: Kluwer Academic

[Arnold 1993] Arnold, M. B. (1984). *Memory and the Brain*, Hillsdale, NJ: Erlbaum

[Bernard 1865] Bernard, C. (1865). *Introduction à l'étude de la médecine expérimentale*, first English translation by Henry Copley Greene, published by Macmillan, 1927; reprinted in 1949

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

[Brooks 1985] Brooks, R. (1985). A Robust Layered Control System for a Mobile Robot Cambridge, MA: *MIT AI Lab Memo 864*, 227–270

[Chisvin and Duckworth 1992] Chisvin, L., and R. J. Duckworth (1992). Content-Addressable and Associative Memory In M.C. Yovits (ed.) *Advances in Computer Science, 2nd ed.* pp. 159–235, Academic Press

[Darwin 1872] Darwin, C. (1872/1998). *The Expression of the Emotions in Man and Animals. 3rd edition*. London: Harper Collins

[FoCL'99] Hausser, R. (1999). *Foundations of Computational Linguistics, 2nd ed.*. Heidelberg Berlin New York: Springer

[Grice 1965] Grice, P. (1965). Utterer's meaning, sentence meaning, and word meaning, *Foundations of Language*, 4:1–18

[Handl et al. 2009] Handl, J., B. Kabashi, T. Proisl, and C. Weber (2009). JSLIM - Computational morphology in the framework of the SLIM theory of language, in C. Mahlow and M. Piotrowski (eds.) *State of the Art in Computational Morphology*, Berlin Heidelberg New York: Springer

[Hausser 2009a] Hausser, R. (2009). Modeling Natural Language Communication in Database Semantics, *Proceedings of the APCCM 2009*, Australian Comp. Sci. Inc., CIPRIT, Vol. 96. Available online at http://www.linguistik.uni-erlangen.de/clue/de/publikationen.html

[Hausser 2009b] Hausser, R. (2009). From Word Form Surfaces to Communication, in T. Tokuda et al. (eds.) *Information Modelling and Knowledge Bases XXI*, Amsterdam: IOS Press Ohmsha. Available online at http://www.linguistik.uni-erlangen.de/clue/de/publikationen.html

[Hutchinson 1948] Hutchinson, G.E. (1948). Circular Causal Systems in Ecology, Ann. New York Acad. Science 50:221-246

[Lazarus and Lazarus 1994] Lazarus, R., and B. Lazarus (1994). *Passion and Reason: Making Sense of Our Emotions*, New York: Oxford University Press

[Naphade and Smith 2009] Naphade, M.R., and J. R. Smith (2009.) Computer program product and system for autonomous classification, Patent Application #:20090037358 - Class: 706 46 (USPTO)

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

[Proust 1913] Proust, M. (1913). *Du côté de chez Swann*, ed. by Jean-Yves Tadie et al., Bibliotheque de la Pleiade, Paris: Gallimard,1987-89

[Quillian 1968] Quillian, M. (1968). Semantic memory in M. Minsky (ed.), *Semantic Information Processing*, 227–270, Cambridge, MA: MIT Press

[Salton 1989] Salton, G. (1989). *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*, Reading, Mass.: Addison-Wesley

[Schegloff 2007] Schegloff, E. (2007). *Sequence Organization in Interaction*, New York: CUP

[Steels 1999] Steels, L. (1999). *The Talking Heads Experiment*. Antwerp: limited pre-edition for the Laboratorium exhibition

[TCS'92] Hausser, R. (1992). Complexity in Left-Associative Grammar. *Theoretical Computer Science* 106.2:283-308, Elsevier. Available online at http://www.linguistik.uni-erlangen.de/clue/de/publikationen.html

[Turney 2002] Turney, P. (2002). Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews, *Association for Computational Linguistics (ACL)*, 417-424

[Wiener 1948] Wiener, N. (1948). *Cybernetics: Or the Control and Communication in the Animal and the Machine*, Cambridge, MA: MIT Press