

# Autonomous Control Structure for Artificial Cognitive Agents

Roland Hausser

Universität Erlangen-Nürnberg  
Abteilung Computerlinguistik (CLUE)  
rrh@linguistik.uni-erlangen.de

## Abstract

This paper presents a control structure based on the principle of balance. This principle mediates between the knowledge of the cognitive agent CA and its current situation. CA's knowledge is represented in the form of concatenated propositions. These are individuated into recognition-action-recognition (rac) sequences. Each rac sequence is assigned a need vector indicating whether the associated action raised, lowered, or left unchanged associated physiological or social need parameters. Recognizing a certain combination of concepts, CA activates rac sequences beginning with those concepts, choosing the rac sequence most likely to maintain or regain equilibrium as its model of action in that situation. This general mechanism for keeping the need parameters within normal range may be viewed as a computational implementation of the notion of purpose or intention.

## Overview

Current approaches to semantics may be divided into two basic types: metalanguage-based and procedural. Metalanguage-based semantics are in turn of two kinds, namely truth-conditional (as in model theory) and use-conditional (as in speech act theory). Because metalanguage definitions are not designed for a viable procedural implementation, neither truth- nor use-conditional semantics are suitable for a computational model of natural language communication.<sup>1</sup>

A procedural semantics, on the other hand, uses concepts based on the recognition and action procedures of cognitive agents' as its basic elements. Examples are Lakoff & Johnson 1980, Sowa 1984, Lakoff 1987, Langacker 1987/1991, Fauconnier 1997, and Gärdenfors 2000. This method of analyzing natural – and modeling artificial – cognitive agents provides a viable alternative to truth- and use-conditional semantics.

The advantages of the procedural approach may be endangered, however, by a lack of generality and formal rigor. This may result in computational systems which work only for special cases (the problem of upscaling) and/or do not have a clear declarative specification (as in hacks), or in vague descriptions of an anecdotal nature.

A procedural approach aiming at a systematic model of cognition is database semantics. It uses a special data structure, called a word bank, and a special motor algorithm, called LA-grammar. The data structure and motor algorithm interact in a way that may be described metaphorically as follows: the word bank provides a railroad system and time-linear LA-grammar the locomotive for navigating through it.

---

<sup>1</sup>See Hausser 2001a for a detailed argument with special attention to truth-conditional semantics. A critique of use-conditional semantics may be found in Hausser 1999, p. 83–86.

The task of modeling a cognitive agent CA may in general be divided into two parts. One is to construct the CA's cognitive capabilities. The other is to design a control mechanism for activating these capabilities selectively to optimize the CA's interaction with its current situation. The question is: how do the two parts work together?

In database semantics, the CA's cognitive capabilities are modeled by means of a word bank in combination with LA-grammars for reading propositional content into and out of the word bank, as well as for inferencing. The reading-in comprises external and internal recognition as well as the interpretation of natural language. The reading-out is based on navigating through the propositional content and comprises external and internal action as well as natural language production.

The word bank manages these diverse functions by coding a proposition as a set of proplets. These are feature structures which specify the functor-argument structure and the relations to other propositions by means of attributes. Proplets eliminate the restrictions of graphically-based representations, allowing (i) a time-linear reading-in of propositions (storage), (ii) an activation of propositions by means of time-linear navigation, and (iii) an accessing of propositions using concepts as the key (retrieval).

In addition, a word bank supports a control mechanism based on relating the CA's current recognition and action to physiological and social need parameters. CA's experiences are stored in the word bank as concatenated propositions forming recognition-action-recognition (rac) sequences. A rac sequence describes (i) an internal or external recognition, (ii) the reaction to this recognition, and (iii) the recognition of the result. A rac sequence is evaluated in terms of need vectors which indicate whether it increased, decreased, or left unchanged the values of corresponding need parameters.

When a certain proposition is read into the word bank via recognition, it activates all stored rac sequences beginning with this proposition. Of these, one is chosen as the model for current action. The choice is the rac sequence with a need vector most likely to return the corresponding need parameter value within normal range. The result of this choice is stored again as a rac sequence. In this sense, the control structure presented here is based on the balance principle.

But what about fixed strategies and behavior patterns corresponding to innate behavior? These may be implemented by means of rac sequences as well. The main difference between instinctive and learned behavior is that the former is based on predefined rac sequences while the latter is based on rac sequences resulting from experience.

Because the control structure presented in this paper is based largely on storing and activating propositions, much of this paper is concerned with analyzing their build-up, classifying different types, and describing their functions. This requires an outline of recognition and action (Sections 1 and 2) and language-based reference (Section 3).

Then follows a description of episodic, absolute, and language propositions, their storage in the data structure of a word bank, their activation in terms of navigation (Sections 4–7), and their interaction in terms of partial and complete matching (Section 8). This complex system of propositional content communicates with its environment by means of LA-grammars for recognition, action, and inference (Section 9).

While the LA-grammars for recognition are activated by events, there remains the question of how to activate the LA-grammars for action and inference. The answer is LA-MOTOR. This LA-grammar is not a fixed component of the CA, but a mobile engine for navigating continuously through the concatenated propositions of the word bank. LA-MOTOR's choices between alternative continuations and when to provide the LA-grammars for action and inference with input are guided in part by need parameter values which cause a highlighting of certain propositions. This ensures the selection of rac sequences suitable to maintain overall cognitive balance (Section 10).

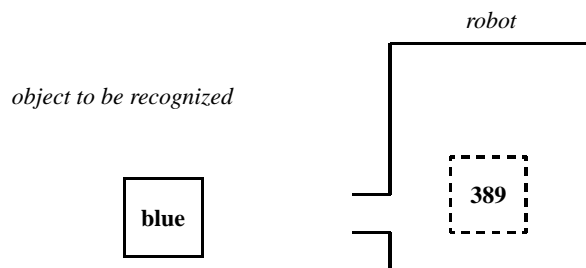
# 1 Recognition and action

A cognitive agent without language interacts with the world in terms of recognition and action. Recognition is the process of transporting structures of the external world into the cognitive agent. Action is the process of transporting structures originating inside the cognitive agent into the world.

The processes of recognition and action may be described at different levels of abstraction. Modeling vision, for example, is complicated by such problems as separating objects from the background, completing occluded portions, perception of depth, handling reflection, changes in lighting, perception of motion, etc.<sup>2</sup>

For purposes of grounding a semantics procedurally, however, a relatively high level of abstraction is appropriate. As an illustration consider a robot without language observing its environment by means of a video camera. The robot's recognition begins with an unanalyzed internal image of the object in question, e.g. a blue square.

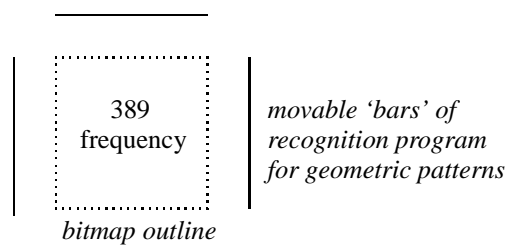
## 1.1 INTERNAL BITMAP REPRESENTATION OF EXTERNAL OBJECT



Inside the robot, the blue square is represented as a bitmap outline and its color as the electromagnetic frequency measured, i.e. 389 nm. Just as an OCR system<sup>3</sup> analyzes bitmap structures to recognize letters, the robot recognizes the form of objects in its task environment by matching their bitmap structures with corresponding patterns.

The recognition of geometric forms may be viewed as a three step process. First, a suitable program approximates the bitmap outline with movable bars resulting in a reconstructed pattern:

## 1.2 ANALYSIS OF AN INTERNAL BITMAP REPRESENTATION



<sup>2</sup>For a summary of natural vision, see Nakayama et al. 1995. A classic treatment of artificial vision is D. Marr 1982. For a summary see J.R. Anderson 1990<sup>2</sup>, p. 36 ff. More recent advances are described in the special issue of *Cognition*, Vol. 67, 1998, edited by M.J. Tarr & H.H.Bülhoff .

<sup>3</sup>OCR or optical character recognition systems are used with scanners to recognize printed text.

Second, the reconstructed pattern is logically analyzed in terms of the number of corners, their angles, the length of the edges, etc. In our example, the logical analysis results in an area enclosed by four lines of equal length forming four right angles.

Third, the logical analysis is classified in terms of abstract concepts, to be discussed in the following section. The classification results in the recognition of the object in question.<sup>4</sup> The basic recognition procedure illustrated above with the example *square* may be extended to other types of geometric objects, to properties like colors, and to relations like A being contained in B.<sup>5</sup>

## 2 Concepts

Logical analyses like 1.2 are classified by concepts defined as abstract types.<sup>6</sup> A type specifies the necessary properties of the structure to be classified as a constellation of parameters and constants, treating accidental properties by means of variables. When a logical analysis is recognized by means of a certain concept type, the analysis is (i) classified by that type and (ii) instantiated as a corresponding concept token.

A concept type is called an M-concept because it can be *matched* with arbitrarily many different logical analyses of the same kind. Consider the following example:

### 2.1 M-CONCEPT OF *square* (TYPE)

$$\left[ \begin{array}{l} \text{edge 1: } \alpha \text{ cm} \\ \text{angle 1/2: } 90^0 \\ \text{edge 2: } \alpha \text{ cm} \\ \text{angle 2/3: } 90^0 \\ \text{edge 3: } \alpha \text{ cm} \\ \text{angle 3/4: } 90^0 \\ \text{edge 4: } \alpha \text{ cm} \\ \text{angle 4/1: } 90^0 \end{array} \right]$$

The accidental property of this M-concept is the edge length, represented by the variable  $\alpha$ .<sup>7</sup> All other properties, such as the number of edges and angles as well as the degree of the angles, are necessary properties. The variables make this M-concept applicable to squares of any size.

### 2.2 DEFINITION: M-CONCEPT

An M-concept is the structural representation of a characteristic logical analysis whereby accidental properties are defined as variables.

When the M-concept 2.1 is matched onto the logical analysis of 1.2, the variables are bound to a particular edge length, here 2cm, resulting in the instantiation of a token, called an I-concept<sub>loc</sub>.

<sup>4</sup>For the sake of conceptual simplicity, the reconstructed pattern, the logical analysis, and the classification are described here as separate phases. In practice, these three aspects may be closely interrelated in an incremental procedure. For example, the analysis system may measure an angle as soon as two edges intersect, the counter for corners may be incremented each time a new corner is found, a hypothesis regarding a possible matching concept may be formed early so that the remainder of the logical analysis is used to verify this hypothesis, etc.

<sup>5</sup>See also Langacker 1987, whose analyses of *above* vs. *below*, p. 219, 467 *before*, p. 222 *enter*, p. 245 *arrive*, p. 247 *rise*, p. 263 *disperse*, p. 268 *under*, p. 280, 289 *go*, p. 283 *pole climber*, p. 311, and *hit*, p. 317, may be regarded as logical analyses in our sense.

### 2.3 I-CONCEPT<sub>loc</sub> OF A *square* (TOKEN)

$$\left. \begin{array}{l} \text{edge 1: 2cm} \\ \text{angle 1/2: } 90^0 \\ \text{edge 2: 2cm} \\ \text{angle 2/3: } 90^0 \\ \text{edge 3: 2cm} \\ \text{angle 3/4: } 90^0 \\ \text{edge 4: 2cm} \\ \text{angle 4/1: } 90^0 \end{array} \right\} \text{loc}$$

The feature *loc* specifies when and where the token was recognized.

### 2.4 DEFINITION: I-CONCEPT<sub>loc</sub>

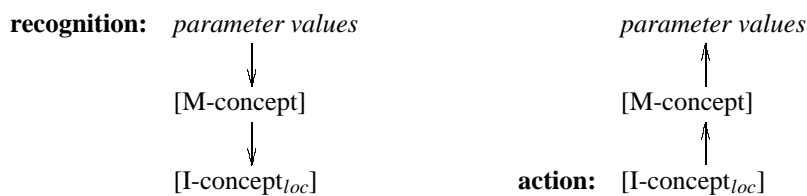
An I-concept<sub>loc</sub> results from successfully matching an M-concept onto a corresponding logical analysis of a parameter constellation at a certain space-time location.

M-concepts are usually defined for subsets of the available parameters and their possible values. For example, the concept 2.1 applies only to a certain constellation of visual parameter values. Other parameters, e.g. color values, are disregarded by the M-concept *square*. It is possible, however, to define elementary concepts which apply to a multitude of different parameters. For example, *situation* could be defined as the elementary M-concept which matches the totality of current parameter values.

Elementary M-concepts may be defined only for those notions for which the corresponding parameters have been implemented. For example, the concepts for *warm* and *cold* may be defined only after (i) the robot has been equipped with suitable sensors for temperature and (ii) the resulting measurements have been integrated into the robot's conceptual structure.

The constellation of parameter values, M-concepts, and I-concepts<sub>loc</sub> provides not only for an abstract characterization of recognition, but also of action. Thereby recognition and action differ in the direction of the respective procedures.

### 2.5 ABSTRACT CHARACTERIZATION OF RECOGNITION AND ACTION



*Recognition* begins with the incoming parameter values which are logically analyzed and classified by means of a matching M-concept. A successful matching binds the

<sup>7</sup>The type-token distinction was introduced by the American philosopher and logician C. S. PEIRCE (1839–1914). An example of a token is the actual occurrence of a sign at a certain time and place, for example the now following capital letter A. The associated type, on the other hand, is the abstract structure underlying all actual and possible occurrences of this letter. Realization-dependent differences between corresponding tokens, such as size, font, place of occurrence, etc., are not part of the associated type.

<sup>7</sup>Instances of the same variable in a concept must all take the same value. Strictly speaking, 2.1 would thus require an operator – for example a quantifier – binding the variables in its scope.

variables to particular values and results in a corresponding I-concept<sub>loc</sub>. In this way constellations of perception parameters are individuated into I-concepts<sub>loc</sub>.

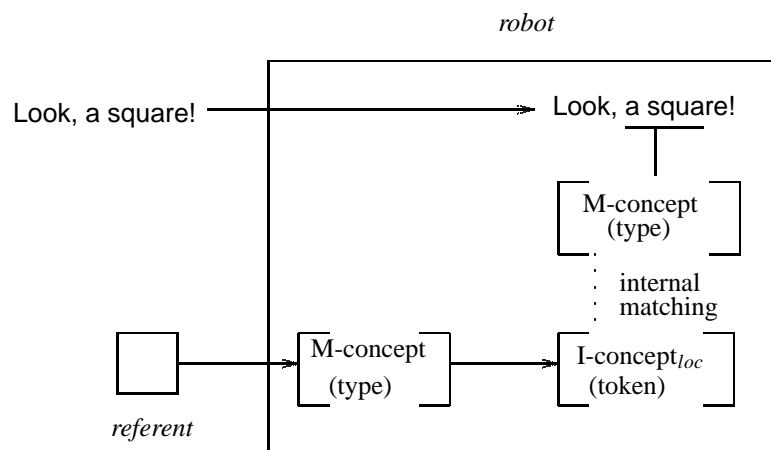
Action begins with a certain I-concept<sub>loc</sub> which is to be realized as a specific outgoing parameter constellation by means of a corresponding M-concept. Consider for example a robot equipped with a gripping device wanting to pick up a glass. For this, a certain I-concept<sub>loc</sub> is realized as a gripping action (token) with the help of the M-concept (general procedure, type). Thereby distance, size, firmness, etc., of the object are determined via the robot's recognition and integrated into the planned action.

The natural evolution of M-concepts can be analyzed as an abstraction over logically similar parameter constellations, whereby the accidental aspects of constellations are represented by variables. Thus the types originate as the formation of classes over sets of similar raw data of perception or unconscious action. Only after the formation of types can individual tokens be instantiated on the basis of these types.

### 3 Internal matching

In the course of evolution,<sup>8</sup> the M-concepts and I-concepts<sub>loc</sub> of recognition and action have acquired additional functions: I-concepts<sub>loc</sub> are used for storing content in the cognitive agent's memory, while M-concepts serve as language meanings. Consider, for example, a robot viewing a geometric object in its task environment and the warden saying Look, a square. In this case, there is a relation of reference between the external word square and the external geometric object which must be reconstructed cognitively inside the robot.

#### 3.1 ANALYSIS OF REFERENCE



<sup>8</sup>Our analysis of language evolution, beginning with non-verbal cognition based on M-concepts and I-concepts<sub>loc</sub>, is of a logical nature. Other approaches to human language evolution are concerned with quite different questions such as

- Why do humans have language while chimpanzees do not?
- Did language arise in 'male' situations such as *Look out, a tiger!*, in 'female' situations of trust-building during grooming, or out of chanting?
- Did language evolve initially from hand signs or was the development of a larynx capable of articulation a precondition?

For further discussion of these questions, which are not addressed here, see J. Hurford et al. 1998.

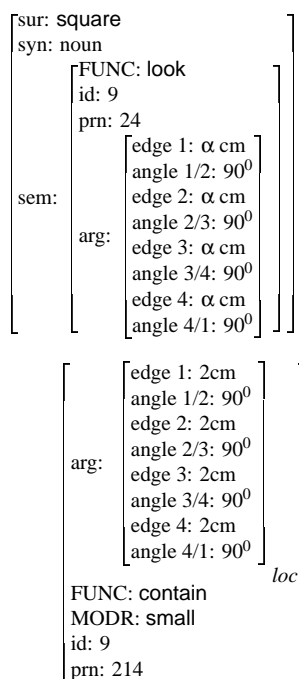
The robot's cognitive structure consists of two levels, the (lower) context level and the (upper) language level. At the context level, the referent is recognized by applying the appropriate M-concept to the incoming parameter values (here a bitmap outline) and instantiating the referent internally as an I-concept<sub>loc</sub>. At the language level, the incoming sign is recognized in a similar way, i.e. an M-form is applied to the parameter values of the external sign and instantiates it as an I-form<sub>loc</sub>. Via lexical look-up, the I-form<sub>loc</sub> (recognized surface) is assigned the M-concept *square* as its literal meaning.

Reference comes about by matching the M-concept of the language level with the I-concept<sub>loc</sub> of the context level. A language meaning defined as an M-concept (type) may be matched with any number of corresponding I-concepts<sub>loc</sub> (tokens) at the context level. Thus, the principle of internal matching models the flexibility of reference which distinguishes the natural languages from the logical and programming languages.

For storing content in memory, I-concepts<sub>loc</sub> are combined into propositions. The combinatorial properties of I-concepts<sub>loc</sub> are characterized by embedding them into feature structures. Similarly, the M-concepts serving as language meanings are embedded into feature structures which specify the associated surface and the syntactic category. These feature structures are generally called proplets and function as the basic elements of concatenated propositions.

Before we turn to the concatenation of proplets into propositions consider the matching relation between a language and a context proplet (depicting reference as in 3.1).

### 3.2 MATCHING BETWEEN THE LANGUAGE AND THE CONTEXT LEVEL



Depending on the cognitive function, the matching between the proplets of the upper and the lower level may apply either only to the concepts involved (partial matching), or to the feature structures as a whole (complete matching).

## 4 Building propositions by means of completion

Proplets using I-concepts<sub>loc</sub> are called episodic proplets. Proplets using M-concepts are of two kinds: one are absolute proplets for expressing general knowledge such as *A square has four corners* in thought rather than language; the other are language proplets. These kinds of proplets differ in their feature structures, as illustrated below. For simplicity, their respective concepts are represented by a name, here *arg: square*.

### 4.1 THREE KINDS OF PROPLETS

EPISODIC PROPLET	ABSOLUTE PROPLET	LANGUAGE PROPLET
$\left[ \begin{array}{l} \text{arg: } \textit{square} \\ \text{FUNC: contain} \\ \text{MODR: small} \\ \text{id: 9} \\ \text{prn: 24} \end{array} \right]$	$\left[ \begin{array}{l} \text{arg: } \textit{square} \\ \text{FUNC: have} \\ \text{MODR:} \\ \text{id: y} \\ \text{prn: abs-3} \\ \text{arg: } \textit{square} \end{array} \right]$	$\left[ \begin{array}{l} \text{sur: } \text{Quadrat} \\ \text{syn: noun} \\ \text{sem:} \left[ \begin{array}{l} \text{FUNC: contain} \\ \text{MODR: small} \\ \text{id: 9} \\ \text{prn: 24} \\ \text{arg: } \textit{square} \end{array} \right] \end{array} \right]$

For purposes of matching, episodic proplets have the attribute containing the I-concept<sub>loc</sub> in first position. Absolute proplets have the attribute containing the M-concept in first and last position. Language proplets have the attribute containing the language surface (here *sur: Quadrat*, where *Quadrat* is the German surface for *square*) in first position and the attribute containing the M-concept in last position. Because the three kinds of proplets have different feature structures, the distinction between I-concepts<sub>loc</sub> and corresponding M-concepts may be left implicit in the attributes containing the concepts.

Propositions built from episodic proplets are called episodic propositions, and similarly for propositions built from absolute and language proplets. A proposition consists of a functor which takes a certain number of arguments. Modifiers are optional and can apply to both arguments and functors. Functors, arguments, and modifiers are defined as certain kinds of proplets which occur in the episodic, absolute, and language variety.

The construction of propositions from proplets is based on a new method of composition based on completing feature structures. The result is a(n unordered) set of proplets which are related to each other solely in terms of attributes. This is in contrast to the usual method of constructing propositions by graphical means, e.g. trees as in linguistics, symbol sequences as in logic, or conceptual graphs as in CG theory.

Completion consists in copying attribute values between proplets in accordance with the relations observed. Assume, for example, that the robot's recognition has produced the I-concepts<sub>loc</sub> *contain*, *field*, *small*, and *square*. These I-concepts are inserted into suitable proplet schemata, as in the following example:

### 4.2 INSERTION OF I-CONCEPTS<sub>loc</sub> INTO PROPLET SCHEMATA

$\left[ \begin{array}{l} \text{func: } \textit{contain} \\ \text{ARG:} \\ \text{MODR:} \\ \text{cnj:} \\ \text{prn:} \end{array} \right]$	$\left[ \begin{array}{l} \text{arg: } \textit{field} \\ \text{FUNC:} \\ \text{MODR:} \\ \text{id:} \\ \text{prn:} \end{array} \right]$	$\left[ \begin{array}{l} \text{arg: } \textit{square} \\ \text{FUNC:} \\ \text{MODR:} \\ \text{id:} \\ \text{prn:} \end{array} \right]$	$\left[ \begin{array}{l} \text{modr: } \textit{small} \\ \text{MODD:} \\ \text{id:} \\ \text{prn:} \end{array} \right]$
---	--	---	---

Such proplets in which only the first attribute has a value are called isolated proplets.

Isolated proplets are combined into a proposition by copying the argument names into the ARG slot of the functor, the functor name into the FUNC slot of the arguments,

the modifier name into the MODR slot of the modified, and the name of the modified into the MODD slot of the modifier. Also, the proplets are provided with a common proposition number prn, and extrapositional relations are established by providing the cnj and id slots with suitable values. Consider the following completion of 4.2:

### 4.3 EPISODIC PROPOSITION *Field contains small square*

[func: <i>contain</i> ARG:field square MODR: cnj: 23 and 24 prn: 24]	[arg: <i>field</i> FUNC: contain MODR: id: 7 prn: 24]	[arg: <i>square</i> FUNC: contain MODR: small id: 9 prn: 24]	[modr: <i>small</i> MODD: square id: 9 prn: 24]
--	---	--	--

This set of proplets is held together by a common proposition number prn (here 24). The first attribute characterizes a proplet as a functor (e.g. func: *contain*), an argument (e.g. arg: *field*), or a modifier (e.g. modr: *small*).

The upper case attributes, called intra-propositional continuation predicates, specify the functor-argument structure of the proposition. For example, the proplet *contain* specifies the arguments *field* and *square*, the arguments *field* and *square* specify the functor *contain*, the argument *square* specifies the modifier *small*, and the modifier *small* specifies the modified *field*. This attribute structure constitutes a bidirectional pointing between the proplets of a proposition.

The intrapositional continuation predicates are followed by the extrapositional continuation predicates, which are conjunctions in functors and identity in arguments. For example, the functor proplet *contain* has the extrapositional continuation predicate [cnj: 23 and 24], which means that the proposition 24 represented in 4.3 is preceded by a proposition 23 and connected to this preceding proposition by the conjunction *and*. Furthermore, the argument proplet *field* has the extrapositional continuation predicate [id: 7], characterizing it as identical to other arguments in the database with the same identity number. The modifier proplet *small*, finally, shares the extrapositional continuation predicate with the modified.

The proposition represented in 4.3 is activated by means of navigating from one proplet to the next based on the respective continuation predicates:

### 4.4 ACTIVATING AN EPISODIC PROPOSITION BY NAVIGATION

[epi: contain] ⇒ [epi: field] ⇒ [epi: square] ⇒ [epi: small]

For simplicity, the episodic proplets of 4.3 are represented in 4.4 as [epi: contain], etc., leaving the continuation predicates implicit. Such a navigation is driven by the LA-grammar defined in 9.2.

## 5 Why is Database Semantics different?

There are at least three reasons why the coding of propositional content in terms of proplets is not equivalent to formalisms based on graphical representations. First, graphical representations like one-dimensional formulas and two-dimensional tree structures or conceptual graphs have restrictions which do not hold for the non-graphical representations of database semantics. Second, graphical representations are motivated by intuitions which database semantics has no cause to replicate. Third, the bidirectional

pointing between the distributed proplets of database semantics offers descriptive possibilities not open to graphical means.

For example, most linguistic analyses are based on trees which are supposed to capture the intuitions of constituent structure. In the well-known case of discontinuous elements, however, these intuitions cannot be expressed by two-dimensional trees, leading to the constituent structure paradox.<sup>9</sup> The coding of concatenated propositions in terms of proplets is based on the more limited intuitions of characterizing (i) the functor-argument structure of elementary propositions and (ii) the nature of their concatenation. Thus the constituent structure paradox is avoided in database semantics.

Similarly, analyses of propositions in predicate calculus use quantifiers binding variables to establish intrapropositional identity relations. Thereby different orders of quantifiers express different meanings. This commitment to one-dimensional order produces problems of (i) creating artificial scope ambiguities and (ii) failing to express readings intuitively required. As an example of the second kind consider the sentences

Every man who loves a woman is happy.

Every man who loves a woman loses her.

and the following attempt to represent their dominant reading in predicate calculus:

$$\forall x [[\text{man}(x) \ \& \ \exists y [\text{woman}(y) \ \& \ \text{love}(x, y)]] \rightarrow [\text{happy}(x)]]$$
$$\forall x [[\text{man}(x) \ \& \ \exists y [\text{woman}(y) \ \& \ \text{love}(x, y)]] \rightarrow [\text{lose}(x, y)]]$$

These parallel formulas are intended to reproduce the subordinate structure of the relative clause. The second formula, however, has the problem that the scope of the quantifier  $\exists y$  does not reach the  $y$  in  $\text{lose}(x,y)$ . An alternative formula with the quantifier  $\exists y$  at the top level would solve the scope problem, but cannot be derived compositionally by translating the words of the sentence into suitable lambda expressions – as required in Montague grammar. This problem does not arise within database semantics, which asserts the identity between proplets both intra- and extrapropositionally by means of attributes rather than by means of quantifiers which bind variables.<sup>10</sup>

The new descriptive possibilities offered by database semantics are based on a strict separation between the representation of content, on the one hand, and its activation, on the other. The representation of content is based on coding all intra- and extrapropositional relations between proplets in a distributed, bidirectional manner by means of attributes. This allows storage and retrieval of proplets in accordance with the principles of a suitable database, using the concepts as the primary key.

The activation of content is based on the principle of navigation. Thereby the content's representation provides a railroad system for navigating through a proposition and from one proposition to the next.<sup>11</sup> For example, given the proplet *field* in 4.3, the system may use the continuation predicate FUNC: contain to proceed to the proplet *contain* with the same proposition number. From there the navigation may continue to *square*, etc. This procedure is based on the retrieval mechanism of the database and does not depend in any way on where and how the two proplets are stored.

As another example consider the two propositions *leave Peter house* and *cross Peter*

---

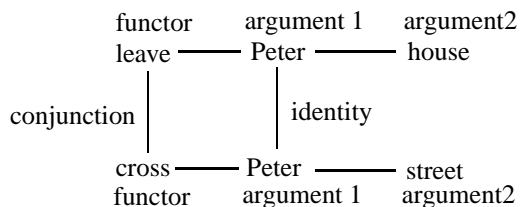
<sup>9</sup>Hausser 1989,p. 24f., 1999, p. 157f.

<sup>10</sup>For a detailed analysis see Hausser 2001c.

<sup>11</sup>In contrast to Fodor's Mentales, e.g. Fodor 1998, neither the representation of content nor its activation constitute a language. Instead, the representation is like the wheels of a mechanical calculator, while the activation is like a calculation based on these wheels.

*street*, concatenated via the conjunction *then* and the identity between the respective names. The corresponding proplet connections may be represented as follows:

### 5.1 'RAILROAD SYSTEM' PROVIDED BY TWO PROPOSITIONS



The content expressed by this railroad system may be activated by a multitude of different navigation types. The most basic distinctions are between intra- and extrapositional forward and backward navigation, and between paratactic (coordinating) and hypotactic (subordinating) navigation, rendering such realizations as the following:<sup>12</sup>

### 5.2 DIFFERENT REALIZATIONS BASED ON DIFFERENT NAVIGATIONS

extrapositional forward navigation:

Peter leaves the house. Then Peter crosses the street.

extrapositional backward navigation:

Peter crosses the street. Before that Peter leaves the house.

intrapositional backward navigation:

The street is crossed by Peter.

subordinating identity-based navigation:

Peter who leaves the house crosses the street.

subordinating conjunction-based forward navigation:

Peter, after leaving the house, crosses the street.

subordinating conjunction-based backward navigation:

Peter, before crossing the street, leaves the house.

These realizations in English reflect different navigations through the propositional content 5.1. The descriptive task of database semantics regarding 5.1 and 5.2 is twofold. One is the coding and decoding task: in the hearer mode, the different surfaces of 5.2 must be mapped into the representation 5.1, and in the speaker mode, the representation 5.1 must be mapped into one of the surfaces of 5.2. The other is the control task: the speaker must choose the navigation most appropriate to the rhetorical task at hand and the hearer must interpret this choice by analyzing the resulting surface.

This paper concentrates on the second task. The goal is to find a general principle for modeling an agent's actions and reactions while interacting with the task environment. Thus, the above examples relating to language interpretation and production merely

<sup>12</sup>Sowa 1984, 2000 seems to come close when he presents a 'linear form' (LF) in addition to a graphical 'display form' (DF). However, his linear form is just a convenient ASCII recoding of the graphical form, and there is no distinction between content representation and activation, no navigation, and no motor algorithm.

serve to illustrate why database semantics is not equivalent to other approaches. The main focus here is on controlling the navigation in general, including non-language-based recognition and action.

## 6 Absolute propositions

Besides episodic propositions there are absolute propositions, built from absolute proplets (cf. 4.1) as in the following example:

### 6.1 ABSOLUTE PROPOSITION *A square has four corners*

func: <i>have</i> ARG:square corners MODR: cnj: abs-3 and abs-4 prn: abs-3 func: <i>have</i>	arg: <i>corner</i> FUNC: have MODR: four id: x prn: abs-3 arg: <i>corner</i>	arg: <i>square</i> FUNC: has MODR: id: y prn: abs-3 arg: <i>square</i>	modr: <i>four</i> MODD: corner id: x prn: abs-3 modr: <i>four</i>
---	---	---	---

Absolute propositions are likewise represented as an unordered set of proplets held together by a common proposition number (here prn: abs-3). For input, this format is suitable for storing absolute propositional content by distributing the proplets in the word bank, using the concepts as the primary key. For output, the content is activated by navigating in accordance with the intra- and extrapropositional continuations.

### 6.2 ACTIVATING AN ABSOLUTE PROPOSITION BY NAVIGATION

[abs: have] ⇒ [abs: square] ⇒ [abs: corner] ⇒ [abs: four]

As in episodic propositions, the navigation serves as the basis (i) of thought in the basic sense of reliving the propositional content in memory, (ii) of inference, and (iii) of language production (conceptualization and serialization).

In database semantics, absolute propositions are not limited to mathematical truth, but represent any kind of knowledge not bound to a particular event. For example, if the robot recognizes three or four times in sequence *Postman comes at 11am*, then these episodic propositions may be replaced by a corresponding absolute proposition.

### 6.3 GENERALIZATION FROM EPISODIC PROPOSITIONS

<i>absolute proposition:</i>	Postman   comes_at   11am
	↑                    ↑                    ↑
<i>episodic propositions:</i>	4. Postman ⇒ comes_at ⇒ 11am
	3. Postman ⇒ comes_at ⇒ 11am
	2. Postman ⇒ comes_at ⇒ 11am
	1. Postman ⇒ comes_at ⇒ 11am

The generalization is based on matching absolute proplets with episodic ones. This type of matching is called complete matching because a whole episodic proposition is matched by a corresponding absolute one. Thereby, not only the concepts but also the feature structures of the proplets involved must be compatible.

One function of absolute propositions is episodic inferencing, i.e. inferencing over episodic propositions using absolute propositions. Consider the following example:

## 6.4 EPISODIC INFERENCE

$$\begin{array}{ccc}
 & 2. [\text{abs: bread}] \Rightarrow [\text{abs:is}] \Rightarrow [\text{abs: food}] & \\
 & \uparrow & \downarrow \\
 1. [\text{epi: John}] \Rightarrow [\text{epi: has}] \Rightarrow [\text{epi: bread}] & & 3. [\text{epi: John}] \Rightarrow [\text{epi: has}] \Rightarrow [\text{epi: food}]
 \end{array}$$

Episodic proposition 1 is the premise, absolute proposition 2 serves as a rule of inference,<sup>13</sup> and episodic proposition 3 is the consequent. The inference is based on matching the M-concept *bread* of the absolute proposition 2 with the corresponding I-concept<sub>loc</sub> of the episodic proposition 1. Using the absolute proposition 2 as a kind of bridge, the episodic proposition 3 is derived as the conclusion.

A second function of absolute propositions is absolute inferencing, i.e. inferencing over absolute propositions. Consider the following example:

## 6.5 ABSOLUTE INFERENCE

$$\begin{array}{ccc}
 & 2. [\text{abs: drink}] \Rightarrow [\text{abs: is}] \Rightarrow [\text{abs: liquid}] & \\
 & \uparrow & \downarrow \\
 1. [\text{abs: milk}] \Rightarrow [\text{abs: is}] \Leftrightarrow [\text{abs: drink}] & & \downarrow \\
 & & \downarrow \\
 & 3. [\text{abs: milk}] \Rightarrow [\text{abs: is}] \Rightarrow [\text{abs: liquid}] &
 \end{array}$$

The matching in episodic as well as absolute inferencing is called partial matching because only part of the premise is matched by the inference proposition and only part of inference proposition is matched by the consequent. Thereby, the compatibility between the proplets involved is limited to their respective concepts.

## 7 Language propositions

The third type of proposition is language propositions. They are also defined as an unordered set of proplets related by intra- and extrapositional continuation predicates and held together by a common proposition number.

### 7.1 LANGUAGE PROPOSITION Feld enthält kleines Quadrat

$$\begin{array}{l}
 \left[ \begin{array}{l} \text{sur: enthalten} \\ \text{syn: verb} \\ \text{ARG:field square} \\ \text{MODR:} \\ \text{sem: cnj: 23 and 24} \\ \text{prn: 24} \\ \text{func: contain} \end{array} \right] \left[ \begin{array}{l} \text{sur: Feld} \\ \text{syn: noun} \\ \text{FUNC: contain} \\ \text{MODR:} \\ \text{sem: id: 7} \\ \text{prn: 24} \\ \text{arg: field} \end{array} \right] \\
 \\
 \left[ \begin{array}{l} \text{sur: Quadrat} \\ \text{syn: noun} \\ \text{FUNC: contain} \\ \text{MODR:} \\ \text{sem: id: 9} \\ \text{prn: 24} \\ \text{arg: square} \end{array} \right] \left[ \begin{array}{l} \text{sur: klein} \\ \text{syn: adn} \\ \text{MODD: square} \\ \text{sem: id: 9} \\ \text{prn: 24} \\ \text{modr: small} \end{array} \right]
 \end{array}$$

This language proposition expresses the content of the episodic proposition 4.3. The sur-attributes contain the German word surfaces { enthalten Feld Quadrat klein }.

<sup>13</sup>A formalized example of such an inference based on an LA-grammar is given in Hausser 1999, p. 494.

Natural language production (speaker mode) is based on navigating through episodic or absolute propositions. Thereby, the proplets traversed are matched with the corresponding sem-attributes of language proplets.

## 7.2 SCHEMA OF LANGUAGE PRODUCTION

<i>language proposition:</i>	blue	truck	belongs_to	milkman.			
	↑	↑	↑	↑			
<i>episodic or absolute proposition:</i>	blue	⇒	truck	⇒	belongs_to	⇒	milkman.

For communicating the propositional content, only the resulting sequence of surfaces is used. It must be adapted to the word order properties of the natural language in question, equipped with proper inflection for tense, number, and agreement, provided with function words, etc.

Natural language interpretation (hearer mode) is based on supplying the sequence of incoming surfaces with language proplet schemata via lexical look up. The analysis of their syntactic composition is interpreted semantically by filling the attributes of the proplet schemata with appropriate values (analogous to the transition from 4.2 to 4.3).

## 7.3 SCHEMA OF LANGUAGE INTERPRETATION

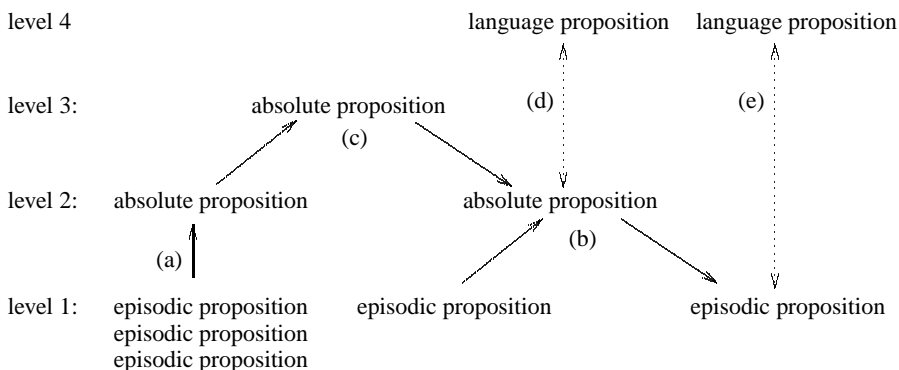
<i>language proposition:</i>	blue	⇒	truck	⇒	belongs_to	⇒	milkman.
	↓		↓		↓		↓
<i>episodic or absolute proposition:</i>	blue		truck		belongs_to		milkman.

For communication, only the sem-attributes of the language proplets are used. They are stored in the word bank using the concepts as the keys.

# 8 Proplet matching

Matching between proplets is always based on an upper and a lower level (cf. 3.1, 3.2, 6.3, 6.4, 6.5, 7.2, 7.3). The different types of matching may be combined, however, into the following four level structure:

## 8.1 MATCHING RELATIONS IN DATABASE SEMANTICS



Episodic propositions appear at level 1, absolute propositions resulting from (a) generalization and used for (b) episodic inference at level 2, absolute propositions used for

(c) absolute inference at level 3, and language propositions used for coding or decoding  
 (d) absolute or (e) episodic propositions at level 4.

The five matching relations are based on the four correlations between (i) absolute and episodic proplets (a, b), (ii) absolute and absolute proplets (c), (iii) language and absolute proplets (d), and (iv) language and episodic proplets (e):

## 8.2 FOUR CORRELATION TYPES BETWEEN PROPLETS

(i) <i>absolute proplet/ episodic proplet</i>	(ii) <i>absolute proplet/ absolute proplet</i>	(iii) <i>language proplet/ episodic proplet</i>	(iv) <i>language proplet/ absolute proplet</i>
$\left[ \begin{array}{l} \text{arg: square} \\ \text{FUNC: have} \\ \text{MODR:} \\ \text{id: y} \\ \text{prn: abs-3} \\ \text{arg: square} \end{array} \right]$	$\left[ \begin{array}{l} \text{arg: square} \\ \text{FUNC: have} \\ \text{MODR:} \\ \text{id: y} \\ \text{prn: abs-3} \\ \text{arg: square} \end{array} \right]$	$\left[ \begin{array}{l} \text{sur: Quadrat} \\ \text{syn: noun} \\ \text{sem:} \\ \left[ \begin{array}{l} \text{FUNC: contain} \\ \text{MODR: small} \\ \text{id: 9} \\ \text{prn: 24} \\ \text{arg: square} \end{array} \right] \end{array} \right]$	$\left[ \begin{array}{l} \text{sur: Quadrat} \\ \text{syn: noun} \\ \text{sem:} \\ \left[ \begin{array}{l} \text{FUNC: contain} \\ \text{MODR: small} \\ \text{id: 9} \\ \text{prn: 24} \\ \text{arg: square} \end{array} \right] \end{array} \right]$
$\left[ \begin{array}{l} \text{arg: square} \\ \text{FUNC: contain} \\ \text{MODR: small} \\ \text{id: 9} \\ \text{prn: 24} \end{array} \right]$	$\left[ \begin{array}{l} \text{arg: square} \\ \text{FUNC: contain} \\ \text{MODR:} \\ \text{id: y} \\ \text{prn: abs-3} \\ \text{arg: square} \end{array} \right]$	$\left[ \begin{array}{l} \text{arg: square} \\ \text{FUNC: contain} \\ \text{MODR: small} \\ \text{id: 9} \\ \text{prn: 24} \end{array} \right]$	$\left[ \begin{array}{l} \text{arg: square} \\ \text{FUNC: have} \\ \text{MODR:} \\ \text{id: y} \\ \text{prn: abs-3} \\ \text{arg: square} \end{array} \right]$

Correlation (i) between an absolute and an episodic proplet constitutes a complete matching in generalization and a partial one in episodic inference. Correlation (ii) between two absolute proplets is limited to absolute inference, where it constitutes a partial matching. Correlations (iii) and (iv) between the sem-attribute of a language proplet and an episodic or absolute proplet constitute a complete matching.

Partial and complete matching equally require that the concepts involved must be compatible. There are only two basic constellations:

## 8.3 MATCHING BETWEEN M-CONCEPTS AND I-CONCEPTS<sub>loc</sub>

(1) M-CONCEPT/I-CONCEPT <sub>loc</sub>	(2) M-CONCEPT/M-CONCEPT
$\left[ \begin{array}{l} \text{edge 1: } \alpha \text{ cm} \\ \text{angle 1/2: } 90^0 \\ \text{edge 2: } \alpha \text{ cm} \\ \text{angle 2/3: } 90^0 \\ \text{edge 3: } \alpha \text{ cm} \\ \text{angle 3/4: } 90^0 \\ \text{edge 4: } \alpha \text{ cm} \\ \text{angle 4/1: } 90^0 \end{array} \right]$	$\left[ \begin{array}{l} \text{edge 1: } \alpha \text{ cm} \\ \text{angle 1/2: } 90^0 \\ \text{edge 2: } \alpha \text{ cm} \\ \text{angle 2/3: } 90^0 \\ \text{edge 3: } \alpha \text{ cm} \\ \text{angle 3/4: } 90^0 \\ \text{edge 4: } \alpha \text{ cm} \\ \text{angle 4/1: } 90^0 \end{array} \right]$
$\left[ \begin{array}{l} \text{edge 1: 2cm} \\ \text{angle 1/2: } 90^0 \\ \text{edge 2: 2cm} \\ \text{angle 2/3: } 90^0 \\ \text{edge 3: 2cm} \\ \text{angle 3/4: } 90^0 \\ \text{edge 4: 2cm} \\ \text{angle 4/1: } 90^0 \end{array} \right]_{loc}$	$\left[ \begin{array}{l} \text{edge 1: } \alpha \text{ cm} \\ \text{angle 1/2: } 90^0 \\ \text{edge 2: } \alpha \text{ cm} \\ \text{angle 2/3: } 90^0 \\ \text{edge 3: } \alpha \text{ cm} \\ \text{angle 3/4: } 90^0 \\ \text{edge 4: } \alpha \text{ cm} \\ \text{angle 4/1: } 90^0 \end{array} \right]$

The matching kinds (i) and (iii) in 8.2 are based on constellation 1 between M-concepts and I-concepts<sub>loc</sub>. The matching kinds (ii) and (iv) are based on constellation 2 between M-concepts and M-concepts.

The I-concepts<sub>loc</sub> and M-concepts in constellation 1 have their origin in basic recognition and action (cf. 2.5). In cognitive agents without language, constellation 1 comes about as generalization (e.g. 6.3), resulting in absolute propositions used for episodic inference (e.g. 6.4). In cognitive agents with language, constellation 1 acquires an additional function in communicating episodic propositional content (e.g. 7.2, 7.3).

Constellation 2 comes about as an abstraction over constellation 1. In cognitive agents without language, this constellation is used for absolute inference (eg. 6.5). In cognitive agents with language, constellation 2 acquires an additional function in the communication of absolute propositional content (eg. 7.2, 7.3).

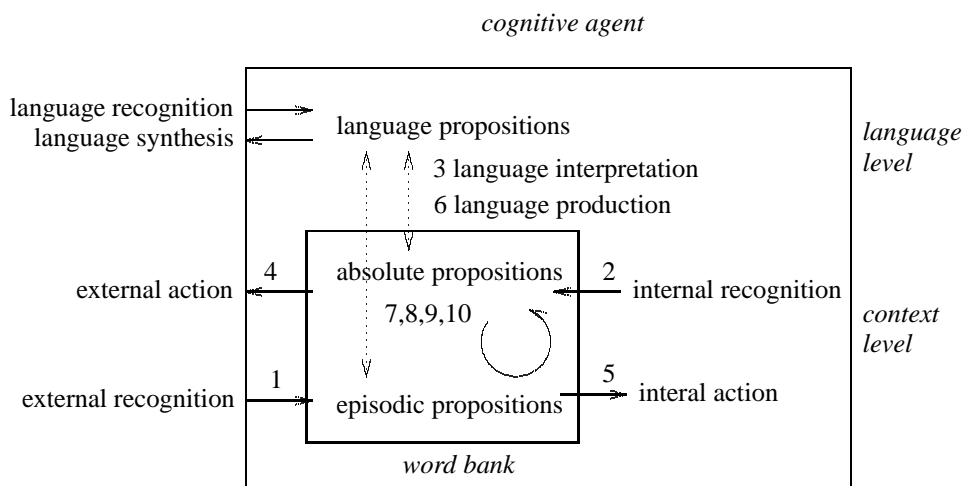
The power of the matching relations in 8.3 is strengthened by the fact that the compatibility between the upper and the lower level is not limited to correlating concepts of the same kind. For example, a familiar object recognized as a table may be classified spontaneously as a platform to better reach a banana (constellation 1). This insight may be turned into a general rule according to which the cognitive agent uses tables routinely as platforms and even stacks them up on top of each other for this purpose (constellation 2). In this way, database semantics mirrors the widely held view<sup>14</sup> that metaphor is not limited to language, but pervades cognition in general.

## 9 Motor-algorithm for powering the navigation

CA's word bank interacts with the internal and external environment by means of LA-grammars. These are of three basic types: (a) recognition, (b) action, and (c) inference.

The LA-grammars for (1) external, (2) internal, and (3) language recognition read propositional content into the word bank. The LA-grammars for action realize certain propositions as (4) external, (5) internal, and (6) language actions. The inference LA-grammars are for (7) generalization as well as (8) episodic and (9) absolute inference.<sup>15</sup>

### 9.1 LA-GRAMMARS FOR RECOGNITION, ACTION, AND INFERENCE



<sup>14</sup>Expressed for example by Lakoff & Johnson 1980. For an overview see Indurkha 1992.

<sup>15</sup>Hausser 1999 presents formally defined LA-grammars for the (3) interpretation and (6) production of a certain sentence, and the (8) derivation of an episodic inference. LA-Q1 and LA-Q2 for answering question (op. cit. p. 492 f.) should be treated as extensions of LA-MOTOR defined below.

The recognition, action, and inference LA-grammars are fixed components and start running when they receive input. In recognition, the input is provided by events in the external and internal environment. But what provides input for action and inference?

The answer is a tenth LA-grammar, called LA-MOTOR (cf. 10 in 9.1). It is special in that it is mobile – like a locomotive. It moves continuously through the word bank’s propositional content, choosing continuations either at random (free association) or by following a highlighting of continuation proplets provided by a control structure. LA-MOTOR is like a greyhound racing after an artificial rabbit provided by highlighting.

LA-MOTOR’s navigation may be one-level or two-level. In one-level navigation, the proplets are simply traversed. In two-level navigation the episodic proplets traversed are matched by corresponding absolute or language proplets. Depending on their content, these matching proplets are passed on to the LA-grammars (4–6) for action and for (7-9) inference, activating them by providing them with input.

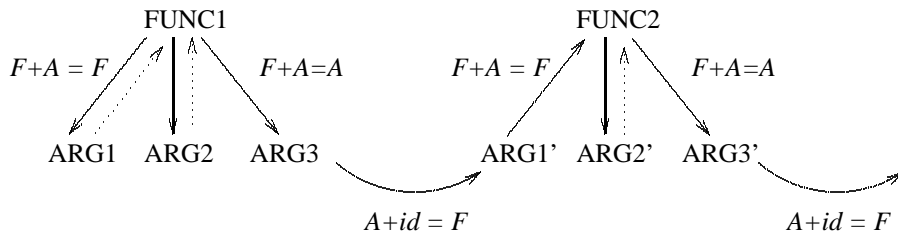
## 9.2 DEFINITION OF LA-MOTOR

$$\begin{aligned}
 ST_S: & \quad \{([\text{func}: \alpha] \{1 \text{ F+A=F}, 2 \text{ F+A=A}\})\} \\
 \text{F+A=F:} & \quad \begin{bmatrix} \text{func: } \alpha \\ \text{ARG: } x \beta y \\ \text{prn: } m \end{bmatrix} \begin{bmatrix} \text{arg: } \beta \\ \text{FUNC: } \alpha \\ \text{prn: } m \end{bmatrix} \Rightarrow [\text{func: } \alpha] \{3 \text{ F+A=F}, 4 \text{ F+A=A}, 5 \text{ F+cnj=F}\} \\
 \text{F+A=A:} & \quad \begin{bmatrix} \text{func: } \alpha \\ \text{ARG: } x \beta y \\ \text{prn: } m \end{bmatrix} \begin{bmatrix} \text{arg: } \beta \\ \text{FUNC: } \alpha \\ \text{prn: } m \end{bmatrix} \Rightarrow [\text{arg: } \beta] \{6 \text{ A+id=F}\} \\
 \text{A+id=F:} & \quad \begin{bmatrix} \text{arg: } \alpha \\ \text{FUNC: } \beta \\ \text{id: } m \\ \text{prn: } k \end{bmatrix} \begin{bmatrix} \text{arg: } \alpha \\ \text{FUNC: } \gamma \\ \text{id: } m \\ \text{prn: } l \end{bmatrix} \Rightarrow \begin{bmatrix} \text{func: } \gamma \\ \text{ARG: } x a y \\ \text{prn: } l \end{bmatrix} \{7 \text{ F+A=F}, 8 \text{ F+A=A}\} \\
 \text{F+cnj=F:} & \quad \begin{bmatrix} \text{func: } \alpha \\ \text{ARG: } x \\ \text{cnj: } m C n \\ \text{prn: } m \end{bmatrix} \begin{bmatrix} \text{func: } \beta \\ \text{ARG: } y \\ \text{cnj: } m C n \\ \text{prn: } n \end{bmatrix} \Rightarrow [\text{func: } \beta] \{9 \text{ F+A=F}, 10 \text{ F+A=A}\} \\
 ST_F: & \quad \{([\text{func: } x] \text{rp}_{\text{F+A=F}})\}
 \end{aligned}$$

The rules of LA-Motor specify patterns which are applied to episodic proplets. The first proplet provides the continuation predicate for moving to the next.<sup>16</sup>

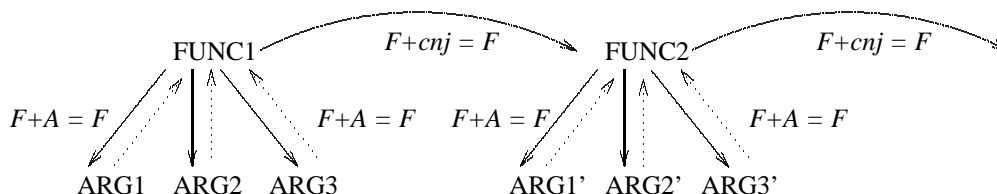
The rule F+A=F is for intrapropositional navigation; it moves from a functor proplet to an argument proplet and back to the functor, using the intrapropositional continuation predicate of the functor to find the argument. F+A=A initiates an extrapropositional navigation based on identity. A+id=F continues this extrapropositional navigation, as indicated by the following continuation structure:

## 9.3 EXTRAPROPOSITIONAL id-NAVIGATION



The fourth rule  $F+cnj=F$  continues an intrapropositional navigation by moving to the next proposition based on the conjunction of the functor, as indicated below:

#### 9.4 EXTRAPROPOSITIONAL *cnj*-NAVIGATION



As an algorithm, LA-MOTOR executes legal transitions from one episodic proplet to the next. However, given that each proplet may provide several possible continuations, there is the question of how to provide the highlighting to guide the navigation.

The first obvious step is to connect LA-MOTOR to the cognitive agent's recognition procedures. When an episodic proposition is read into the word bank by means of LA-grammars for external, internal, or language<sup>17</sup> recognition, the new proplets are highlighted, causing LA-MOTOR to jump to the first proplet and to follow the incoming propositions from there. LA-MOTOR's tracing of incoming propositions is important because it positions the focus point in preparation for subsequent action.

## 10 Autonomous control structure

In order to guide LA-MOTOR's navigation from recognition to action, the concatenated propositions in a word bank are individuated into recognition-action-recognition (rac) sequences. These store past experiences relating to, for example, feeling hungry. When internal recognition reads *CA feels hungry* into the word bank once more, this proposition is used to retrieve and highlight all earlier rac sequences beginning with it:

### 10.1 RECOGNITION HIGHLIGHTING ASSOCIATED RAC SEQUENCES

*CA feels hungry*  
 |  
*CA feels hungry-CA searches at place X-CA finds no food*  
 |  
*CA feels hungry-CA searches at place Y-CA finds a little food*  
 |  
*CA feels hungry-CA searches at place Z-CA finds lots of food*

The retrieval of rac sequences beginning with *CA feels hungry* is based on the word bank's data structure, using the concepts as the key. LA-MOTOR navigates through these rac sequences, jumping throughout the word bank from proplet to proplet by following their continuation predicates, ignoring those which are not highlighted.<sup>18</sup>

<sup>16</sup>For simplicity, the treatment of modifiers is omitted in LA-MOTOR.

<sup>17</sup>While absolute propositions are normally processed by LA-grammars for inference, episodic propositions are normally processed by LA-MOTOR. However, as a special case of natural language interpretation and production, absolute propositions may also be read into and out of the word bank. This requires a straightforward extension of LA-MOTOR.

The question raised by this simplified example is: within which rac sequence's action proposition should LA-MOTOR switch from one-level to two-level navigation, thus activating a suitable LA-grammar for action? The solution requires that the actions initiated by LA-MOTOR have a purpose for the cognitive agent.

For the implementation of purpose it is useful to distinguish between long and short term purposes. Because CA's survival from one situation to the next is a primary concern, the remainder of this paper will be devoted to short term purposes.

In database semantics, short term purposes are driven by need parameters in combination with the balance principle. Examples of need parameters are energy supply and temperature. The current value of the former may leave normal range because of consumption, of the latter because of external changes. Other short term need parameters are rest, sleep, fear, but also shelter, social acceptance, intellectual stimulation, etc.

According to the balance principle, CA strives to maintain the values of the need parameters within normal range. For this, rac sequences originating as a real action are evaluated as to whether they result in raising or lowering any of the need parameters, or leaving them unchanged. The evaluation is expressed by means of need vectors, pointing up or down at various degrees, whereby  $\rightarrow$  expresses 'no change.'

When a stored rac sequence is activated later by an incoming recognition, its need vector is related to the current parameter values. If one of the parameters is out of normal range, the rac sequence with a need vector most likely to regain balance for that parameter is highlighted most. In this way, LA-motor is enticed to (i) choose the rac sequence most appropriate to regain balance, and (ii) to switch to two-level navigation when traversing the action proposition of that rac sequence.

For example, if CA's need parameter for energy supply has dropped a little out of range, the proposition *CA feels hungry* is read into the word bank. Because the state of the energy parameter corresponds best to the second rac sequence in 10.1, it would be highlighted most, causing CA to search at place Y. The first and the third rac sequence in 10.1 would come into play only if CA's search at place Y is unsuccessful.

Similarly, if CA's need parameter for temperature has risen out of range a lot, *CA feels hot* is read into the word bank. This results in highlighting rac sequences beginning with that proposition, e.g. (1) *CA feels hot-CA moves into shade-CA cools down a little*, (2) *CA feels hot-CA moves into the basement-CA cools down a lot*, and (3) *CA feels hot-CA moves into the sun-CA feels hotter*. Here the need vector of the second rac sequence is suited best to regain balance, resulting in the corresponding action.

But what about situations in which there are no rac sequences in the word bank to be activated by the current recognition? In this case, predefined ('innate') open rac sequences provide options for action. Consider the following example:

## 10.2 OPEN RAC SEQUENCES FOR UNKNOWN SITUATIONS

*CA meets unknown animal*  
 |  
*unknown situation-CA approaches- ?*  
 |  
*unknown situation-CA waits- ?*  
 |  
*unknown situation-CA flees- ?*

<sup>18</sup>In memory, the propositions of a rac sequence are naturally connected by subsequent proposition numbers, e.g. 54, 55, 56, reflecting the temporal order of their occurrence (cf. Hausser 2001b). This allows their traversal by means of LA-MOTOR based on the *cjn*-values of the respective functor-proplets.

These rac sequences are open because due to lack of experience the result proposition is represented as unknown ('?'). However, they have predefined need vectors relating to the fear parameter, the first being low, the second medium, and the third high.

Appearance of the unknown animal as small and cuddly will lower the fear parameter, resulting in agreement with the first rac sequence. Appearance as big and ferocious, on the other hand, will increase the fear parameter, resulting in agreement with the third rac sequence. And accordingly if the animal's appearance is uncertain. Provided that CA survives the encounter, a new complete rac sequence evaluating CA's action towards this particular animal will be stored in CA's word bank.

An indirect way to mediate between known (cf. 10.1) and unknown (cf. 10.2) situations is inference. The LA-grammars for generalization (cf. 6.3), episodic inference (cf. 6.4), and absolute inference (cf. 6.5) are activated by incoming absolute proplets resulting from two-level navigation of LA-MOTOR. Thereby, the proplets for generalization are based on complete matching, while those for episodic and absolute inference are based on partial matching. The crucial question is when LA-MOTOR should switch to which kind of two-level navigation to activate an LA-grammar for inference.

Finally, there is language production (cf. 7.2). It is also activated by LA-MOTOR switching to two-level navigation, though the proplets used to match the path traversed are language rather than absolute proplets. As in non-verbal action, the control of language production is based on rac sequences and need parameters. The main difference to the nonverbal examples described above is that the need parameters for language interaction are of a social rather than a physiological nature.

As an example, consider the social rac sequences (1) *CA meets colleague in the morning-CA greets colleague-Colleague greets CA* and (2) *CA meets colleague in the morning-CA looks the other way-Colleague doesn't greet CA*. The need vectors of these rac sequences relate to the need parameter of social acceptance, whereby the first need vector increases the current value of that parameter, while the second one does the opposite.

## Conclusion

The control structure presented in this paper is driven by physiological and social need parameters. They are used to evaluate recognition-action-recognition (rac) sequences in terms of need vectors related to those parameters. A detailed computational mechanism for selecting actions to maintain the cognitive agent's need parameters within normal range is described.

This treatment of control resembles the treatment of coherence (Hausser 1999, p. 473f.) and spatio-temporal indexing (Hausser 2001a) in database semantics: All three are driven by the structure of the external world. However, while coherence and spatio-temporal indexing are merely reflected in the cognitive agent's memory, the control structure uses such memories as models for actions to continuously maintain balance.

The proper modeling of rac sequences in relation to physiological and social need parameters is an empirical matter of considerable interest. In combination with the balance principle, it provides a computational implementation of intention or purpose.

Whether CA's automatic assignment of need vectors to rac sequences has been implemented correctly or not may be verified objectively by testing CA's behavior. The same holds for the derivation of inferences and the implementation of long term purposes. For this methodology to become reality, however, robotics and procedural semantics have to work together much more closely than is currently the case.

## References

- Anderson, J.R. (1990) *Cognitive Psychology and its Implications*, 3rd ed., W.H. Freeman and Company, New York.
- Gärdenfors, P. (2000) *Conceptual Spaces*, MIT Press, Cambridge, Massachusetts.
- Fauconnier, G. (1997) *Mappings in Thought and Language*, Cambridge University Press, Cambridge, England.
- Hausser, R. (1989) *Computation of Language, An Essay on Syntax, Semantics and Pragmatics in Natural Man-Machine Communication*, Symbolic Computation: Artificial Intelligence, Springer-Verlag, Berlin-New York.
- Hausser, R. (1999) *Foundations of Computational Linguistics*, Springer-Verlag, Berlin, New York.
- Hausser, R. (2001a) "The Four Basic Ontologies of Semantic Interpretation," in H. Kangassalo et al. (eds) *Information Modeling and Knowledge Bases XII*, IOS Press Ohmsha, Amsterdam.
- Hausser, R. (2001b) "Spatio-temporal indexing in Database Semantics," in A. Gelbukh (ed.) *Computational Linguistics and Intelligent Text Processing*, Lecture Notes in Computer Science Vol. 2004. Springer-Verlag, Berlin, New York.
- Hausser, R. (2001c) "Reconstructing predicate calculus in database semantics," in preparation.
- Hurford, J., M. Studdert-Kennedy, & C. Knight (1998) *Approaches to the Evolution of Language*, Cambridge University Press, Cambridge, England.
- Indurkha, B. (1992) *Metaphor and Cognition*, Kluwer, Dordrecht.
- Kosslyn, S. & D. Osherson (eds.) (1995) *Visual Cognition*, MIT Press, Cambridge, Massachusetts.
- Langacker, R. (1987/1991) *Foundations of Cognitive Semantics*, Vol. 1/2, Stanford University Press, Stanford, CA.
- Lakoff, G. (1987) *Women, Fire, and Dangerous Things*, University of Chicago Press, Chicago, Illinois.
- Lakoff, G. & M. Johnson (1980) *Metaphors We Live By*, University of Chicago Press, Chicago & London.
- Marr, D. (1982) *Vision*, W.H. Freeman and Company, New York.
- Nakayama, K., H. Zijiang, & Shinsuke Shimojo (1995) "Visual Surface Representation: A Critical Link between Lower-level and Higher-level Vision," in S. Kosslyn & D. Osherson (eds.).
- Peirce, W.S. (1931 – 1958) *Collected Papers of Charles Sanders Peirce*, edited by C. Hartshorne and P. Weiss, 6 vols. Harvard University Press, Cambridge, Massachusetts.
- Sowa, J.F. (1984) *Conceptual Structures*, Addison-Wesley, Reading, Massachusetts.
- Sowa, J. F. (2000) *Conceptual Graph Standard*, revised version of December 6, 2000. <http://www.bestweb.net/sowa/cg/cgstand.htm>
- Tarr, M.J. & H.H. Bülthoff (eds.) (1998) *Image-based object recognition in man, monkey and machine*, special issue of *Cognition*, Vol. 67.1&2:1–208.