A Theory of Signs for Database Semantics

Roland Hausser

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

Abstract

The goal of this paper is to build a bridge from a certain intuitive conception of natural language communication, called the SLIM theory of language, to a technical approach, called database semantics (cf. Hausser 1999/2001). The intuitive approach proceeds from the interfaces of a cognitive agent's recognition and action, the levels of language and context, the nature of concepts, and the reconstruction of reference based on internal matching between the two levels, including an intuitive theory of signs. The technical software approach begins with a data structure suitable for the indexing and retrieval of content, an algorithm operating on the data structure, and a procedure for modeling language interpretation, inferencing, and production based on this algorithm.

Both approaches have coexisted for years, illuminating each other in suggestive ways, but without a systematic transition from one to the other. When it comes to the details of handling individual words in language interpretation and production, however, the intuitive theory of signs must be reconciled with the structural details required by the technical approach. The solution presented in this paper reconstructs the language and context components of the intuitive approach by refining the data structure of database semantics. For this, the different sign types are integrated into the basic information units of proplets as the values of certain attributes. Furthermore, different language proplets and context proplets are distinguished, and stored in different areas of the database.

1 Interfaces and Basic Components of a Cognitive Agent

Up to now, scientific analyses of natural language have been mostly limited to structural objects, fixed on paper or magnetic tape. Such objects are exemplified by a single word form, a sentence, or a text. By concentrating on the structure of the signs, one has attempted to abstract away from the aspect of communication.

The purpose of producing and interpreting language in the first place, however, is interaction between cognitive agents. Therefore, a scientific analysis of natural language cannot fail to be inadequate if it does not include the production and interpretation procedures inside the cognitive agents. The question should not be what a sign of language *is*, but rather what it *does*, and how it does it by virtue of the way it is.

In order to have artificial agents which can tell what they see and do what we tell them, they must be designed to have vision, hearing, articulation, locomotion, and an arm to handle things. And they must have a central control which integrates cognitive input and output as well as physical interaction with the world into a smooth routine.

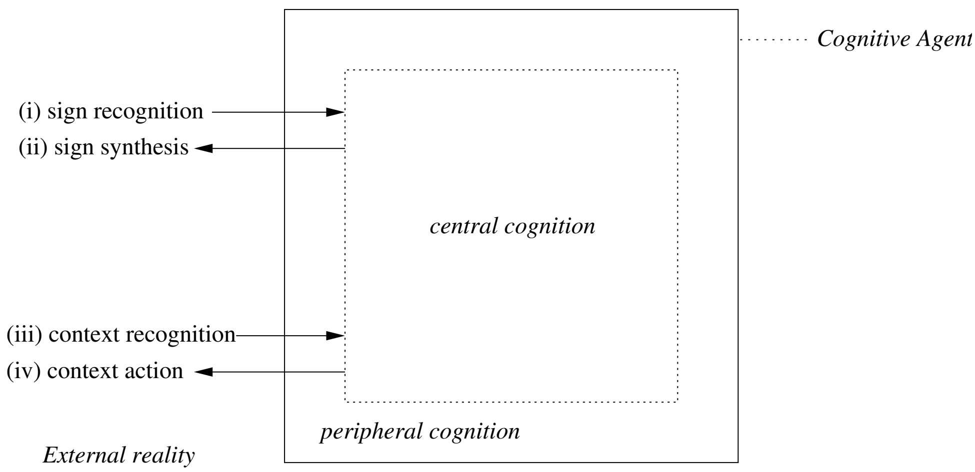
Such a functional model must be verified in the form of a hardware realization which interacts with the real world. This is an engineering task consisting of a hardware part and a software part. In order for the various parts to interact smoothly, there should be a *declarative specification*. A declarative specification may be written as a design plan before starting to program or as a design explanation after the implementation. Often the two interact in a sequence of development cycles.

The crucial aspect of a declarative description is its level of abstraction. The need for such a level is shown by computer source code, which runs perfectly but is very hard to read. Moreover, source code fails to specify which properties are accidental (e.g., programming style including choice of programming language) and which are necessary. Therefore, a declarative specification is needed not only to guide the implementation of the system, but also to provide a wider public with a clear understanding of the theoretical solution instantiated by the implementation.

A declarative specification should be like an algebraic definition in logic: it should list the basic elements and specify the rules of combination. However, unlike an algebraic definition, a declarative specification is not restricted to set theory. Also unlike an algebraic definition, a declarative specification must take care of many additional aspects related to computational modeling such as the input-output conditions, the procedures of recognition and action, the data structure, the method of indexing and retrieval, inferencing, control structure, spatio-temporal orientation, interpretation and production of language, etc.

Let us begin the design of the declarative specification of our functional model by determining the input-output conditions of a talking cognitive agent. They are based on (i) sign recognition and synthesis and (ii) context recognition and action.

1.1 INTERFACES OF A COGNITIVE AGENT



Peripheral cognition handles the agent's interaction with the external world. This interaction may be viewed as the processes of transporting agent-external aspects of the world into agent-internal cognition (recognition and language interpretation) and of transporting agent-internal aspects of cognition into the world (action and language production).

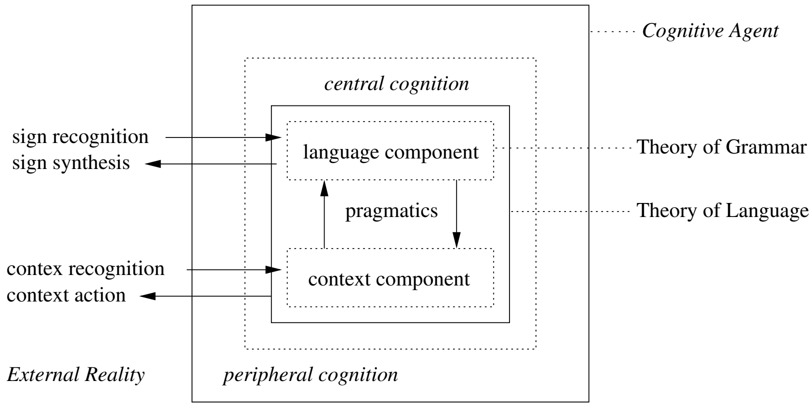
Central cognition handles storage and retrieval of content provided originally by peripheral cognition. We may assume that the processing of this content in central cognition is based on a homogenous coding, such as neural activity in humans or electronic operations in a computer. This coding may be described in an abstract manner which is independent of its realization in brain or computer.

Next we turn to the basic functional principles of cognition. The first, obvious requirement is that these functions combine to connect the input and the output of the agent.

That is, sign recognition must be connected to context action and context recognition must be connected to sign synthesis. These connections are not just a reflex between a stimulus and a response, but based on a memory with associated inferencing. This is the basis for sign and context recognition with processing but without external action, for context interaction without language, and for language interaction without input-output activity at the context level.

These structural properties of communication and cognition are best handled by complementing the distinction between peripheral and central cognition, indicated in 1.1 above, with the distinction between the levels of language and context, shown below:

1.2 COMPONENTS OF CENTRAL COGNITION



The distinction between the levels of language and context is motivated by the evolutionary fact that there are cognitive agents which resemble humans in many respect, yet don't have the capability of language. Thus, language may be viewed as an additional component, built upon the earlier developed context component.

The distinction between the two levels is also motivated functionally in that the context provides the content which is expressed in language by the speaker and stored in the context by the hearer. The context component stores what the agent recognizes, and provides algorithms for drawing inferences on this content and for deriving actions. The language level, in contrast, has the task of encoding a certain content of the context level (speaker mode) and of decoding a given language sign for storage in the contextual database (hearer mode).

The interaction between the two components is handled by the rules of language pragmatics. Language pragmatics may be analyzed as a phylo- and ontogenetic specialization of nonlinguistic pragmatics, just as language recognition and synthesis may be analyzed as phylo- and ontogenetic specializations of contextual recognition and action, respectively.

2 A First Step: Designing a Context Component

The structural analyses presented in 1.1 and 1.2 raise the question of whether they could be agreed upon by a wider range of sciences. After all, the basic structure of cognition is being studied in several fields, each taking its own point of view. Consider the following examples.

Evolutionary psychology studies higher primates in order to construct a model of communication which explains how human communication evolved from this higher primates. Thereby it finds that these animals have skill teaching, highly evolved social structures, and some surprisingly language-like forms of communication (Marc D. Hauser 1997).

Cognitive psychology wrestles with the question of how to present the concept of, e.g., a car, given that we can see only one side of a car but are able to imagine the whole shape, and even add structural and functional information such as where the engine is and what it does (Barsalou 1999). Others study the emergence of language from embodiment (MacWhinney 1999), which has also become a topic in philosophy, as witnessed by a recent boom in Merleau-Ponty (Dreyfus 2002).

Artificial intelligences is spearheaded by the MIT robotic lab. Its wellknown effort is the building of COG (Brooks et al. 1998), a metallic robot resembling a human from the waiste up, with a face consisting of video cameras, but also eye brows to signal emotions, with arms and hands to perform acts of holding and moving, and an elaborate mechanism to model 'shared attention', i.e., the following of the other's gaze during communication, especially important in early child-mother interaction.

As the cognitive models mentioned above do not form a coherent, explicit theoretical whole, we do not know whether our model of input-output conditions, peripheral and central cognition, and language and context level would be acceptable to them. To enhance compatibility, however, we pay heed to the other sciences' credos. Thus, we would certainly want our model to be in concord with the basic principles of evolution. It should also be compatible with the findings of cognitive psychology. And it should be verified in the form of computational hard ware models functioning in the real world.

Given the difficulty of our task, we would like to begin with the most basic component in its most basic form. This component is what we call the *context*. In animals without language, the context is the only component of cognition. In humans, it is one of two cognitive components, the other being the language component.

The construction of a cognitive agent with a context but without language, as exemplified by a dog, is a good starting point[1] for the following reasons. First, when we come to modeling language interpretation and production, we will need a context component anyway. Second, if we assume that there is a natural relation between the cognition of dogs and humans, there is a good chance that there is a natural upscaling from a model of a dog's cognition to that of a human. Third, the comparison of what the cognition of a dog and a human have in common, i.e. the context component, provides heuristics for finding abstract representations behind a multitude of variants.[2]

3 Level of Abstraction

To increase the chances of success, we must simplify the design of a context. For this, we take a lesson from modern ecology by controlling the robot's environment – and thus the conditions of its recognition. For the purpose of modeling recognition and action in principle, we use the simplified world of colored geometric objects such as triangles, squares, etc.

The cognitive building blocks of the context are called *concepts*. They are used to classify bitmap outlines for recognizing squares and triangles as well as intervalls of the color spectrum to recognize red, green, etc. They are also used for storing content in memory, as the data on which inferences are run, as the input to context action and language production, etc.

Concepts come in two varieties, types and tokens. This distinction, introduced by Peirce, is illustrated in the following example of the type and the token of the concept *square*:

[1]We are using dogs as our example because higher primates are currently subject of a debate as to whether or not they exhibit rudimentary language and culture.

[2]For example, we take the liberty to omit smell in our model, even though it is important for dogs (and for non-verbal communication in humans).

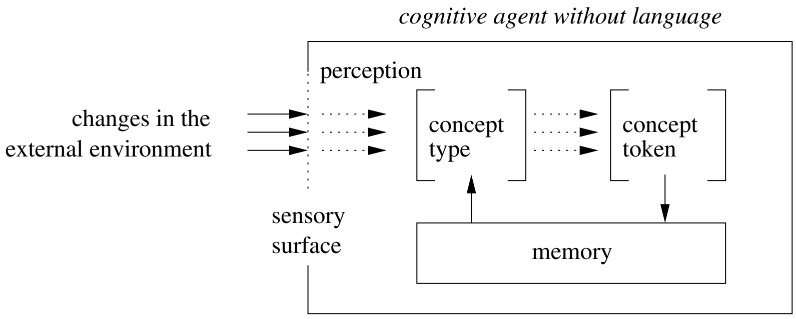
3.1 THE TYPE AND A TOKEN OF THE CONCEPT *square*

$$
\begin{array}{ll}
\textit{type} & \qquad\qquad \textit{token} \\
\begin{bmatrix}
\text{edge 1: } \alpha \\
\text{angle 1/2: } 90^o \\
\text{edge 2: } \alpha \\
\text{angle 2/3: } 90^o \\
\text{edge 3: } \alpha \\
\text{angle 3/4: } 90^o \\
\text{edge 4: } \alpha \\
\text{angle 4/1: } 90^o
\end{bmatrix}
& \qquad\qquad
\begin{bmatrix}
\text{edge 1: 2cm} \\
\text{angle 1/2: } 90^o \\
\text{edge 2: 2cm} \\
\text{angle 2/3: } 90^o \\
\text{edge 3: 2cm} \\
\text{angle 3/4: } 90^o \\
\text{edge 4: 2cm} \\
\text{angle 4/1: } 90^o
\end{bmatrix}
\end{array}
$$

The type defines the necessary properties of a concept by means of constants and the accidental properties by means of variables. In the above example, the necessary properties of the concept type *square* are four angles of 90 degrees and four edges of equal length. The accidental property is the edge length, represented by the variable α. The variable makes the concept type applicable to squares of any size.

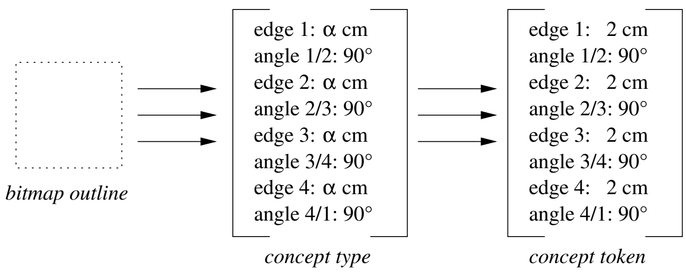
In recognition, concept types and concept tokens function together as follows:

3.2 CONCEPT TYPES AND TOKENS IN CONTEXTUAL RECOGNITION



The incoming parameter values are matched by a concept type, which is instantiated as concept token: The concept type is provided by memory. The resulting context token is stored in memory. The matching process involving a concept type and and a concept token is illustrated below with the recognition of a square:

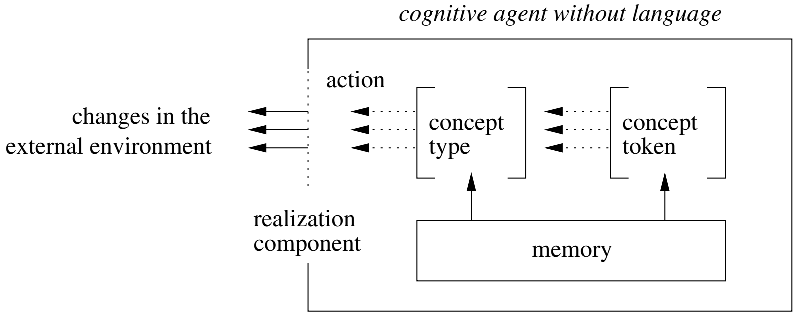
3.3 CONCEPT TYPES AND CONCEPT TOKENS IN RECOGNIZING A SQUARE



It would be no problem to build a machine that can recognize colored geometric objects using concepts like those illustrated above. This is because the notions used in the definitions have procedural counterparts which can be implemented relatively easily. Incidentally, they also correspond to the line, edge, and angle detectors discovered by Hubel and Wiesel 19?? in the visual cortex of cats.

Next consider the inverse direction. The inverse of perception and recognition is intention and action. Intention is the process of developing an action cognitively, while action is the mechanism of realizing an intention by changing the external environment.

3.4 CONCEPT TYPES AND CONCEPT TOKENS IN ACTION

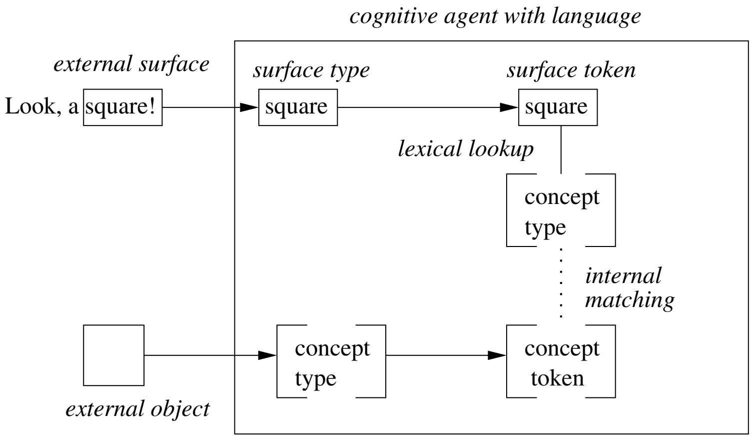


Intentions are represented as constellations of concept tokens and realized as actions by means of corresponding types. The formation of intentions is based on the agent's control structure, current situation, and inferences over content stored in memory.

4 Adding Language: Basic Reference

In cognitive agents with language, the concept types acquire a secondary function as the meanings of words. This is the basis of reference, which comes about by matching the concept types of language meaning with the concept tokens at the level of context:

4.1 SYMBOLIC REFERENCE BASED ON INTERNAL MATCHING

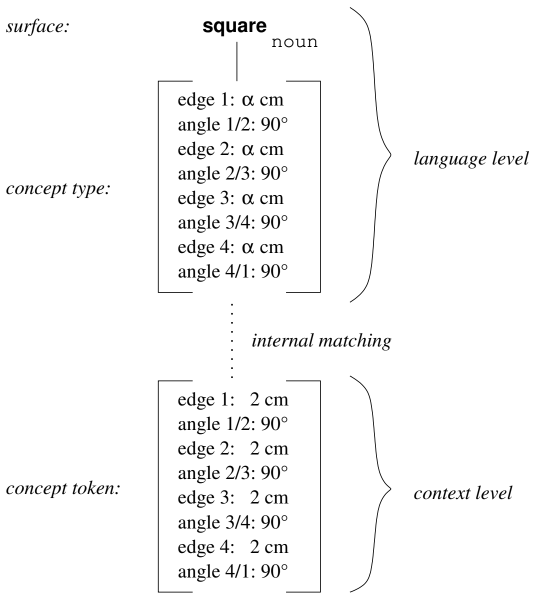


In this example, the cognitive agent recognizes a square at the level of context and the sign Look, a square! at the level of language (immediate reference).

Sign recognition and synthesis at the level of language are analyzed as specializations of context recognition and action, respectively. Just as the recognition of a square at the level context is based on parameter values in a certain modality (here vision) which are matched by a concept type and instantiated as a concept token, the recognition of the word surface square is based on parameter values in a certain modality (e.g., hearing) which are matched by a surface type and instantiated as a surface token. And similarly for contextual action (cf. 3.4) and sign synthesis.

The recognized surface of the sign is passed to lexical lookup, which assigns a literal meaning defined as a concept type. Reference comes about by matching the language concept type with the context concept token, as illustrated below:

4.2 INTERNAL MATCHING BASED ON THE TYPE-TOKEN CORRELATION



In this example, the concept type used in 3.3 for contextual recognition is reused in a secondary function as a literal meaning which is lexically attached to the English surface square. Furthermore, the type/token relation used in 3.3 for recognition is reused in the secondary function of *internal matching*.

The principle of internal matching between concept types and concept tokens models the flexibility of reference which distinguishes the natural languages from the logical and programming languages. It allows use of the same analyzed sign to refer to an open number of different referents at the level of context – in language interpretation and language production, and in immediate reference as well as mediated reference.

5 Reference Mechanisms of the Different Kinds of Signs

The mechanism of reference illustrated above with the word square is characteristic for a certain kind of sign, called *symbol*. Other kinds of natural language signs are *proper names* like John or R2D2, and *indexicals* like here, now, I, you, or this. In addition, there is the sign type of *icons*, which is marginal for synchronic natural language communication,[3] but important for explaining the evolution of symbols.

In modern times, the theory of signs was founded by Peirce, who analyzes the sign kinds symbol, indexical, and icon, but omits names. Symbols are defined as follows:

> A symbol is a sign which would lose the character which renders it a sign if there were no interpretant. Such is any utterance of speech which signifies what it does only by virtue of its being understood to have signification.
>
> Peirce 1940, p. 104.

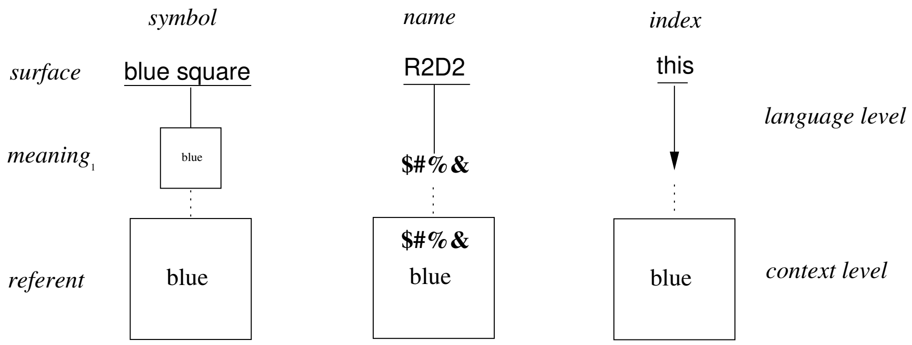
[3] As pointed out by de Saussure 1967, pp. 81, 82. See Hausser 1999/2001, pp. 114 f.

Similarly, an index is defined by Peirce as a sign which would lose the character which renders it a sign as soon as the object it is pointing at is removed; an icon is defined as a sign which retains its character as a sign even if there is no object to refer to, and no interpretant.

The disadvantage of Peirce's definitions is that they are unsuitable for computational implementation. For the purpose of modeling natural language communication computationally, the functioning of the different sign types, i.e., their respective mechanisms of reference, must be explained in terms of their meaning structures instead.

The different reference mechanisms of symbols, indexicals, and names, based on their different kinds of meanings, is illustrated in the following schematic comparison of the three sign types, used to refer to the same contextual object, i.e., a blue square.

5.1 COMPARING ICONIC, NAME-BASED, AND INDEXICAL REFERENCE



At the language level, each sign type consists of a surface and a lexically attached meaning$_1$. At the context level, the referential object is a blue square in a simplified representation of a concept token.

The meaning$_1$ of a *symbol* is a concept type. As explained in the previous section, reference with the sign type of symbol is based on matching the concept type at the level of language with a corresponding concept token at the level of context.

In the sign type *name*, the role of the meaning$_1$ is taken up by private identity markers. They originate at the level of context in order to indicate that different appearances of the referential object are being recognized as the same individual.[4]

In cognitive agents with language, (copies of) the private markers are reused by attaching them to public surfaces in an act of naming. Thereafter, reference with a name uses the public surface to call up each agent's private marker to match the corresponding marker in each agent's cognitive representation of the referential object.

Acts of naming may be explicit, as in a ceremony of baptism, or implicit, as in the following example: Agent A observes an unfamiliar dog running around, appearing and disappearing in the bushes. For continuity, the private identity marker $#%& is inserted into the cognitive structures representing the different appearances of the dog in A's context. Later, the owner calls the dog Fido. Agent A adopts the name by attaching $#%& to the public surface Fido. Henceforth, the name Fido refers for A to the dog in question by matching the private marker attached to the name with the corresponding marker inserted into (the cognitive representations of) the referent.

The sign type *indexical*, finally, has a meaning$_1$ defined as one of two characteristic pointers. The first points into the agent's context and is called the context pointer or C. The second points at the agent and is called the agent pointer or A. Consider the following examples:

[4]The marker used in 5.1 is $#%&, because the cognitive ability of recognizing identity may be present in agents without language and without a competence of numbering.

5.2 INDEXICALS AS NOUNS AND ADJECTIVES WITH A AND C POINTERS

noun A	*noun C*	*adj A*	*adj C*
I, we	you	here	there
	he, she, it	now	then
	this, they		

Indexicals sharing the same pointer differ in terms of additional symbolic-grammatical distinctions. For example, the context pointer of the word this is restricted to single, non-animate referential objects, while they is restricted to a plurality of referential objects which may be animate or inanimate. Similarly, you is restricted to referential objects of any number and gender which are potential partners of communication, while he is restricted to a single referential object of male gender which is currently is not regarded as a partner of communication.

All three mechanisms of reference must be analyzed as internal, cognitive procedures. This is because it would be ontologically unjustifiable to locate the fixed connections between surface and meaning$_1$ in the external reality.

The distinction between the different *kinds of signs*, i.e., symbol, name, and indexical is orthogonal to the distinction between the main *parts of speech*, i.e., noun, verb, and adjective, as well as to the corresponding distinction between the basic *elements of propositions*, i.e., argument, functor, and modifier. *Symbols* occur as noun, verb, and adjective. *Indexicals* occur as noun and adjective. *Name* occur only as noun.

The orthogonal correlation between the kinds of signs and the parts of speech may be represented graphically as follows:

5.3 RELATION BETWEEN THE KINDS OF SIGN AND THE PARTS OF SPEECH

name	Fido		
indexical	this	here	
symbol	dog	black	see
	noun	*adj.*	*verb*

The kind of sign which is the most general with respect to the parts of speech is the symbol, while the name is the most restricted. Conversely, the part of speech (and, correspondingly, the propositional element) which is the most general with respect to different kinds of sign is the noun (object), while the verb (relation) is the most restricted.

6 Database Metaphor of Successful Communication

The theory of language outlined above is modeled computationally in database semantics (dbs). Dbs is based on the assumption that the knowledge of the speaker and the hearer are represented in the form of databases. Furthermore, communication is successful if the speaker encodes a certain part of his or her database into language, and the hearer reconstructs this part *analogously* in his or her database – in terms of (i) a correct decoding and (ii) a correct storage at a corresponding location.

It is a crucial property of natural language that a sign's storage location in the speaker's database can be reconstructed by the hearer without any need for special indexing information. Achieving this kind of automatic indexing and retrieval is not trivial.

Consider robot A coding current observations in the following form:

Field contains triangle. Field contains square.

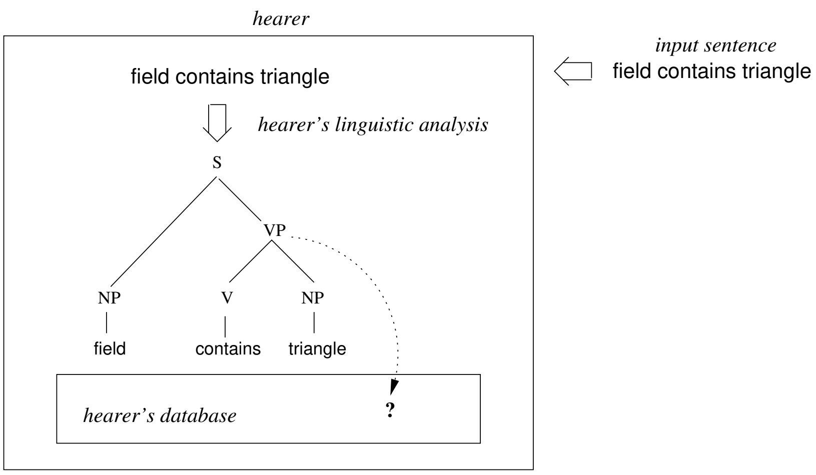
These are two elementary propositions[5] concatenated in terms of their adjacent position in the sequence. The elements of the propositions are the words.

Intrapropositonally, the words are related by a functor-argument structure. The functors are two instances of *contain*, which have the arguments *field* and *triangle* in the first proposition, and the arguments *field* and *square* in the second.

Extrapropositionally, the two propositions are related by their concatenation in terms of adjacency. They are also related in terms of the identity between the referents of the two occurrences of the noun *field* – assuming that the triangle and the square are observed in the same field.

Robot A's reporting his findings to robot B using a language raises the question of where robot B should store the propositions in its database:

6.1 ILLUSTRATION OF THE HEARER'S STORAGE PROBLEM



The external sign (input sentence) is represented inside the hearer and linguistically analyzed. For the sake of the argument, the analysis is a familiar phrase structure tree.

Storing this analysis in the hearer's memory (database) requires a *primary key*. Which property of the tree should be selected? The S node is not suitable as a primary key because it is shared by all phrase structure trees. The structure of the tree[6] is not be suitable either because it is shared by many different contents.

Furthermore, when the hearer turns into a speaker, the hearer's problem of storage turns into the speaker's problem of retrieval. Assume, for example, that the speaker has just the uttered the proposition field contained triangle. This raises the question: Which key should allow the robot to retrieve the closely related content of field contained square in order to arrive at a coherent sequence of utterances?

7 Data Structure of a Word Bank

The solution of database semantics to the indexing and retrieval problem of successful communication is based on following principles:

[5]For an explanation of the classic notion of a proposition see Hausser 1999/2001, Section 3.4, pp. 61–63.

[6]Assuming that the tree structure is coded into a compressed form, e.g. (S(NP VP (V NP))).

7.1 Basic principles of database semantics

1. The primary key is the content words. Content words are the nouns (including names), verbs, and adjectives, in contradistinction to function words, which include determiners, auxiliaries, and prepositions.

2. Content words may be connected by two kinds of relations: intrapropositional and extrapropositional. The intrapropositional relations are the functor-argument structure between verbs, nouns and adjectives within the same elementary proposition. The extrapropositional relations are the identity between nouns and the conjunction between verbs of different propositions.

3. The intra- and extrapropositional relations are not represented by graphical means. Instead, the content words are analyzed as feature structures called *proplets*[7] such that all intra- and extrapropositional relations are coded by means of attributes.

These principles are realized as the data structure of a word bank, consisting of proplet types and proplet tokens.

7.2 Data structure of a word bank

type_a	token_{a1} token_{a2} token_{a3} token_{a4}, etc.
type_b	token_{b1} token_{b2} token_{b3} token_{b4}, etc.
type_c	token_{c1} token_{c2} token_{c3} token_{c4}, etc.
etc.	

The proplet types are like the *owner records*, and the proplet tokens are like the *member records* of a classic *network database*. Like a network database, a word bank can be simulated in a relational database system.

In a word bank, the two concatenated propositions of 6 are represented as follows:

7.3 Representating the propositions of 1.2.1 in a word bank

PROPLET TYPES PROPLET TOKENS

$$
\begin{bmatrix} \text{verb: } \textit{contain} \\ \text{arg:} \\ \text{ctn:} \\ \text{ctp:} \\ \text{prn:} \end{bmatrix}
\qquad
\begin{bmatrix} \text{verb: } \textit{contain} \\ \text{arg: field triangle} \\ \text{ctn: 2 contain} \\ \text{ctp:} \\ \text{prn: 1} \end{bmatrix}
\begin{bmatrix} \text{verb: } \textit{contain} \\ \text{arg: field square} \\ \text{ctn:} \\ \text{ctp: 1 contain} \\ \text{prn: 2} \end{bmatrix}
$$

$$
\begin{bmatrix} \text{noun: } \textit{field} \\ \text{fnc:} \\ \text{idy:} \\ \text{prn:} \end{bmatrix}
\qquad
\begin{bmatrix} \text{noun: } \textit{field} \\ \text{fnc: contain} \\ \text{idy: 1} \\ \text{prn: 1} \end{bmatrix}
\begin{bmatrix} \text{noun: } \textit{field} \\ \text{fnc: contain} \\ \text{idy:1} \\ \text{prn: 2} \end{bmatrix}
$$

$$
\begin{bmatrix} \text{noun: } \textit{square} \\ \text{fnc:} \\ \text{idy:} \\ \text{prn:} \end{bmatrix}
\qquad
\begin{bmatrix} \text{noun: } \textit{square} \\ \text{fnc: contain} \\ \text{idy: 2} \\ \text{prn: 2} \end{bmatrix}
$$

[7]The term *proplet* is coined by analogy to *droplet* and refers to a basic part of an elementary proposition.

$$\begin{bmatrix}\text{noun: }\textit{triangle}\\ \text{fnc:}\\ \text{idy:}\\ \text{prn:}\end{bmatrix} \qquad \begin{bmatrix}\text{noun: }\textit{triangle}\\ \text{fnc: }\mathsf{contain}\\ \text{idy: 3}\\ \text{prn: 1}\end{bmatrix}$$

This word bank is based on two kinds of proplets, verbs and nouns, the attributes of which are defined as follows:

7.4 DEFINITION OF ATTRIBUTES IN VERBAL AND NOMINAL PROPLETS

verb:	= part of speech
arg:	= argument(s) of a verb
ctn:	= connection to next
ctp:	= connection to previous
prn:	= proposition number

noun:	= part of speech
fnc:	= functor of noun
idy:	= identity number
prn:	= proposition number

In proplet types, all attributes except for the part of speech (pos) have the value NIL (represented by empty space). As shown in 7.3, the types are ordered alphabetically in a column. Each type is followed by a *token line*. All proplets in a token line have the same pos value. The number of proplets in a token line is open.

Proplet tokens belonging to the same proposition share a common proposition number prn. The arg attribute in verbs and the fnc attributes in nouns are the intrapropositional continuation predicates. They code the functor- argument structure. For example, the proplet[8]

$$\begin{bmatrix}\text{verb: }\textit{contain}\\ \text{arg:}\mathsf{field\ triangle}\\ \text{ctn: 2 }\mathsf{contain}\\ \text{ctp:}\\ \text{prn: 1}\end{bmatrix}$$

of proposition 1 specifies field and triangle as its arguments. The proplets

$$\begin{bmatrix}\text{noun: }\textit{field}\\ \text{fnc: }\mathsf{contain}\\ \text{idy: 1}\\ \text{prn: 1}\end{bmatrix} \text{ and } \begin{bmatrix}\text{noun: }\textit{triangle}\\ \text{fnc: }\mathsf{contain}\\ \text{idy: 3}\\ \text{prn: 1}\end{bmatrix},$$

also belonging to proposition 1, specify contain as their functor.

The extraprositional relation of *conjunction* is coded in the ctn and ctp attributes of the two verb proplets in the word bank 7.3 :

$$\begin{bmatrix}\text{verb: }\textit{contain}\\ \text{arg:}\mathsf{field\ triangle}\\ \text{ctn: 2 }\mathsf{contain}\\ \text{ctp:}\\ \text{prn: 1}\end{bmatrix} \qquad \begin{bmatrix}\text{verb: }\textit{contain}\\ \text{arg:}\mathsf{field\ square}\\ \text{ctn:}\\ \text{ctp: 1 }\mathsf{contain}\\ \text{prn: 2}\end{bmatrix}$$

As shown by their different proposition numbers, these two proplets belong to two different propositions. However, the ctn attribute of the verb of proposition 1 specifies the proplet

[8]Superficially, proplets may seem to resemble the feature structures of HPSG. The latter, however, are intended to be part of a phrase structure tree rather than a database, do not encode the functor-argument structure in terms of *bidirectional* pointers, do not concatenate propositions in terms of extrapropositional relations, and thus do not provide a suitable basis for time-linear navigation.

contain of proposition 2 as its successor, while the ctp attribute of the verb of proposition 2 specifies the proplet contain of proposition 1 as its predecessor.

The extrapropositional relation of *identity* is coded in the idy attributes of the following two noun proplets in the word bank 7.3:

$$\begin{bmatrix}\text{noun: } \textit{field}\\ \text{fnc: contain}\\ \text{idy: 1}\\ \text{prn: 1}\end{bmatrix} \qquad \begin{bmatrix}\text{noun: } \textit{field}\\ \text{fnc: contain}\\ \text{idy:1}\\ \text{prn: 2}\end{bmatrix}$$

Again, the different proposition numbers indicate that these two proplets belong to different propositions. They are asserted to be coreferential, however, as indicated by their idy attributes having the same value, namely 1.

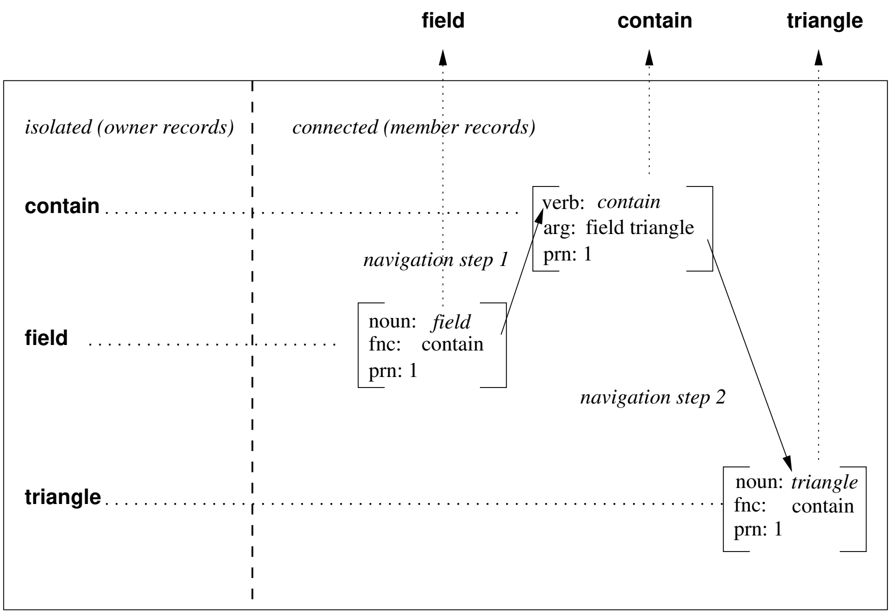
8 Autonomous Navigation

By coding all intra- and extrapropositional relations of concatenated propositions into the proplets of their content words, database semantics achieves complete freedom from the constraints of graphical representations. This provides for easy storage and powerful retrieval.

The continuous retrieval of successor or predecessor proplets in a word bank constitutes a kind of operation which conventional databases do not provide, namely an autonomous time-linear navigation through the concatenated propositions of the database. Navigation plays a crucial role in the modeling of successful communication in database semantics: it is used for the *conceptualization* and basic *serialization* of language production by the speaker.

Consider the following example of language production from a word bank:

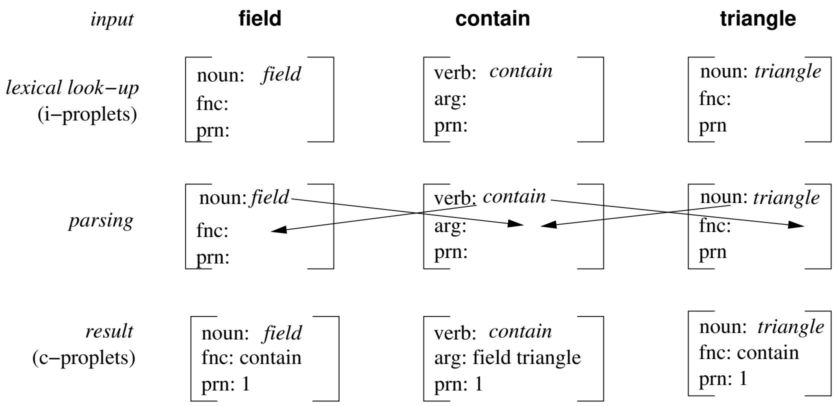
8.1 PRODUCTION OF field contains triangle



The navigation happens to begin at the proplet *field* with the proposition number 1. Based on the proplet's continuation predicate, i.e., [fnc: contain], the navigation algorithm looks for the token line of contain and proceeds from the owner record to the member record with the prn value 1. The proplet *contain* has the continuation predicate [arg: field triangle]. The first value, field, confirms the previous navigation. The second value triangle is the new 'next' proplet. Again, it is found by going to the token line of triangle and proceeding from the owner record to the member record with the prn value 1.

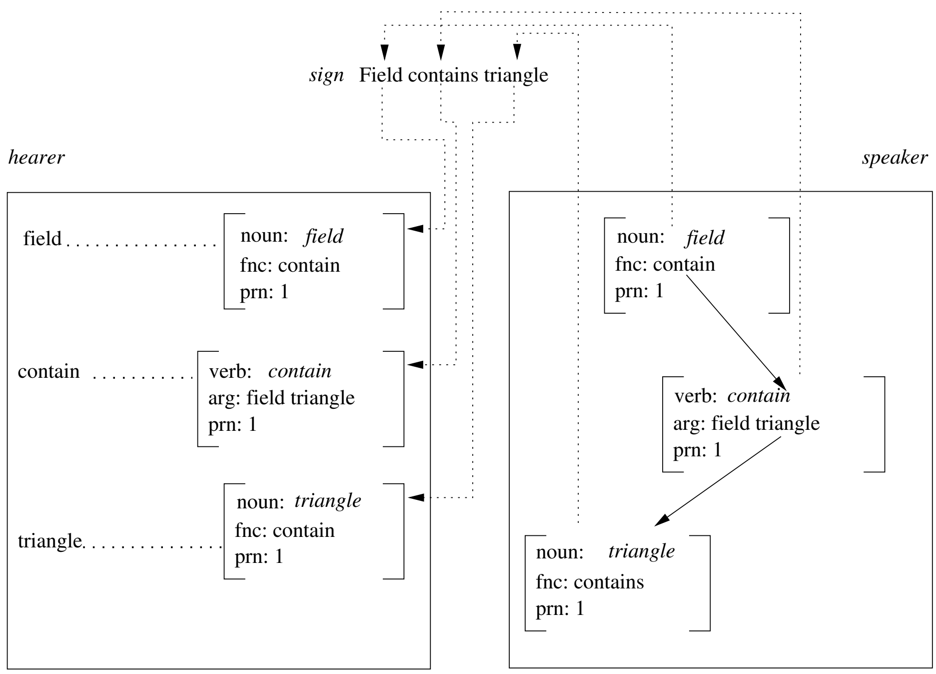
The content traversed may be read out automatically by matching the value of its pos attributes (here *field, contain*, and *triangle*) with corresponding language-dependent surfaces – as indicated in the top line of 8.1. The hearer receives the surfaces and assigns suitable language proplets to them via lexical look-up, resulting in a sequence of lexical types. Time-linear parsing of this sequence completes the lexical types into proplet tokens.

8.2 INTERPRETATION OF field contains triangle



The completion consists in copying the pos value of one proplet into a suitable continuation attribute of another, and vice versa. The result is a(n unordered) set of the three co-indexed proplets *field*, *contain*, and *triangle*. The hearer may store the resulting proplets in accordance with the principles of the data structure in question, eliminating the time-linear order:

8.3 TRANSMISSION OF CONTENT BY MEANS OF NATURAL LANGUAGE



When the hearer turns into the speaker, however, the time-linear structure of language is reintroduced by means of navigation.

9 Reconstructing Internal Matching in a Word Bank

Next, the theory of signs presented informally in Section 5 must be reconciled with the data structure and communication mechanism of a word bank. This is accomplished by means of the following extensions of the data structure:

9.1 EXTENSIONS OF WORD BANK TO ACCOMMODATE THEORY OF SIGNS

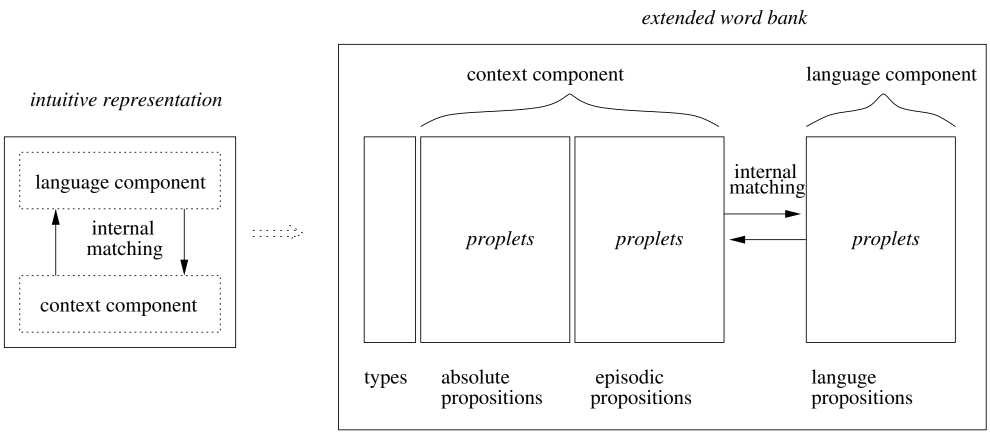
1. The levels of language and context must be distinguished in the data structure of a word bank.

2. The different kinds of signs must be integrated into the proplets representing the language level.

The first requirement is fulfilled by introducing a general distinction between language proplets and context proplets in a word bank. This distinction is realized (i) by introducing characteristic differences between proplets and (ii) by storing different kinds of proplets in different areas in the word bank.

The distinction between areas in a word bank is shown in the following example in comparison to our intuitive correlation of the language and the context level:

9.2 ADAPTING INTUITIVE APPROACH TO WORD BANK DATA STRUCTURE



In the intuitive representation, the border line between the language and the context component is horizontal, such that internal matching occurs in a vertical direction. In the word bank, in contrast, the vertical direction is already occupied by the column of types (cf 7.3 and 8.1). Therefore, internal matching occurs here in a horizontal direction, whereby the borderline between the language and the context component is vertical.

The horizontal axis of the word bank in 9.2 is differentiated further by dividing the context into the areas for absolute and episodic proplets, used for absolute and episodic propositions, respectively. Absolute propositions express content which is independent of any specific spatio-temporal event, like *A square has four corners*. Episodic propositions, in contrast, express content representing a specific observation or action, like *Field contains a square*.

In addition, the horizontal axis is utilized for the spatio-temporal indexing of episodic propositions. This is shown by the following example of a word bank, containing an absolute, two episodic, and one language proposition.

context type	*absolute context token*	*episodic context token*			*language token*
sur: noun: *apple* cat: NP sem: mdr: fnc: idy: prn:	sur: noun: *apple* cat: NP sem: sg def mdr: fnc: be idy: 2 prn: a4	sur: noun: *apple* cat: NP sem: sg def mdr: fnc: buy idy: 2 prn: e14	sur: noun: *apple* cat: NP sem: sg def mdr: fnc: eat idy: 2 prn: e15	⇒	sur: this noun: C cat: SNP sem: -animate sg mdr: fnc: eat idy: 1 prn: 1
sur: verb: *be* cat: V sem: mdr: fnc: ctn: ctp: prn:	sur: verb: *be* cat: V sem: pres mdr: arg: apple fruit ctn: ctp: prn: a4				
sur: verb: *buy* cat: V sem: mdr: fnc: ctn: ctp: prn:		sur: verb: *buy* cat: V sem: pres mdr: arg: person^z apple ctn: e15 eat ctp: prn: e14			
sur: verb: *eat* cat: V sem: mdr: fnc: ctn: ctp: prn:			sur: verb: *eat* cat: V sem: pres mdr: arg: person^z apple ctn: ctp: e14 buy prn: e15	⇒	sur: ate verb: *eat* cat: V sem: pres mdr: arg: person^z C ctn: ctp: prn: 1
sur: noun: *fruit* cat: NP sem: mdr: fnc: idy: prn:	sur: noun: *fruit* cat: NP sem: sg def mdr: fnc: be idy: 2 prn: a4				
sur: noun: *person*^{z} cat: NM sem: sg def mdr: fnc: idy: *&%# prn:		sur: noun: *person*^{z} cat: NM sem: sg def mdr: fnc: buy idy: *&%# prn: e14	sur: noun: *person*^{z} cat: NM sem: sg def mdr: fnc: eat idy: *&%# prn: e15	⇒	sur: John noun: *person*^{z} cat: NM sem: sg def mdr: fnc: eat idy: *&%# prn: 1

The first column shows the proplet types serving as owner records. The second column presents the absolute proposition *an apple is a fruit* with the proposition number a4. The third and fourth column present the episodic propositions *John buys an apple* and *John eats the apple* with the proposition numbers e14 and e15, respectively. The fifth and last column shows the language proposition John ate this with the prn 1, containing the symbol ate, the name John and the indexical this. It is produced from proposition e15 by matching its episodic proplets with corresponding language proplets.

The spatio-temporal indexing is based on the linear sequence of content coming into (recognition) and going out of (action) the agent's database, illustrated by the propositions e14 and e15 in 9.3. The sequence is established by adding episodic propositions representing the agent's incoming recognitions and outgoing actions in the natural order of their occurrence.

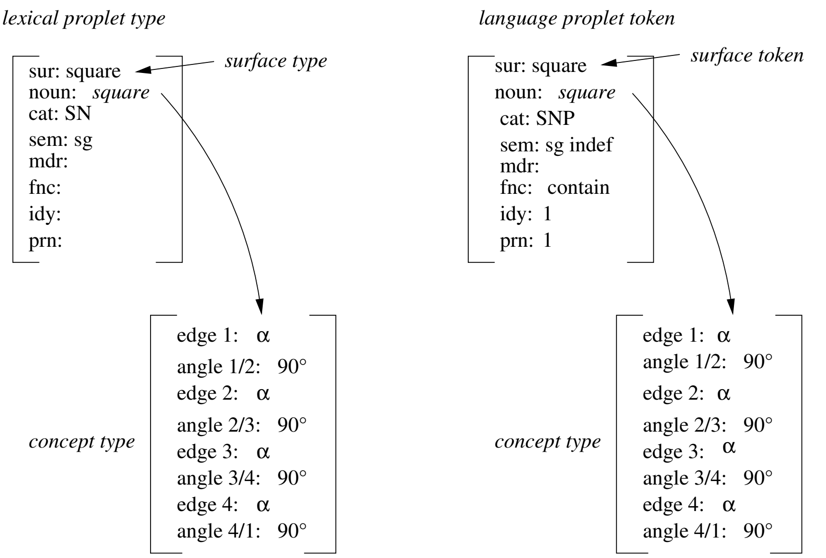
For spatio-temporal orientation, the agent uses propositions which relate to temporal or spatial *landmarks*. Temporal landmarks are cyclical events such as the change of day and night, the phases of the moon, the changing seasons, the daily round of the milkman, etc.[9] Spatial landmarks are familiar objects fixed in the agent's environment.[10]

10 Integrating Different Kinds of Signs into Proplets

To complete the reconstruction of our intuitive theory of signs in database semantics, it remains to realize the second requirement of 9.1. It is fulfilled by replacing the values of the noun, verb, and adj attributes in language proplets – represented so far by English words for simplicity – by corresponding concepts in the case of symbols, identity markers in the case of names, and pointers in the case of indexicals.

As an example consider the following language proplets representing a symbol:

10.1 LEXICAL PROPLET TYPE AND LANGUAGE PROPLET TOKEN OF square



In language interpretation, the parameter values of the incoming sign surface, here square, are matched by the surface type[11] of a matching lexical proplet. Thereby, the surface is in-

[9]Cyclical events are also used for structuring the future, as when telling a child: *You have to sleep three more nights and then it will be your birthday.* Applying temporal inferencing learned from analyzing the past, the child can extrapolate into the future.

[10]Our approach to spatio-temporal indexing corresponds to the A-series of MacTaggert 1908. His distinction between the A- and the B-series reflects two basic approaches found from antiquity to present, called here the subjective and the objective approach. Simply speaking, we say that on the subjective approach time and space are moving through the agent, while on the objective approach the agent is moving through time and space.

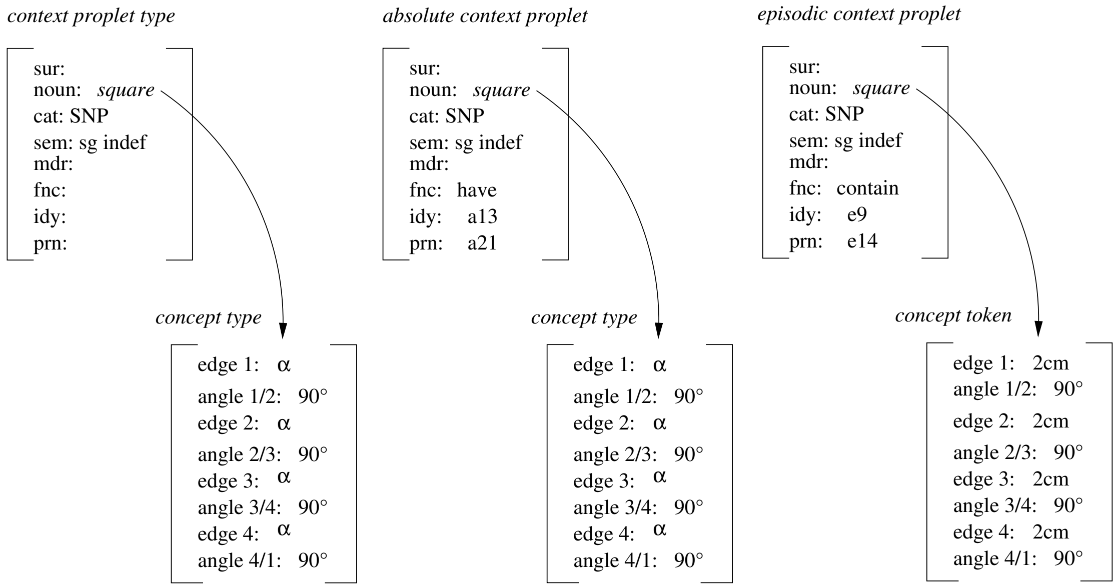
[11]The difference between surface types and tokens resembles that between concept types and tokens in that surface types specify accidental properties by means of variables while surface tokens specify them by means of constants.

stantiated as a token and provided with a lexical analysis (lexical lookup). Syntactic-semantic parsing of the preceding and following proplets (cf. 8.2) results in a language token of square by supplying values to various attributes. Both kinds of language proplets use lexically provided concept *types* as the value of their noun attribute.

The corresponding context proplets differ from language proplets in that the value of their sur attribute is NIL. There are context proplet *types*, which serve as the owner records of the word bank, and context proplet *tokens*. The tokens are divided further into absolute and episodic proplet tokens for representing absolute and episodic propositions, respectively.

The three kinds of proplets at the context level are illustrated below with explicit concepts:

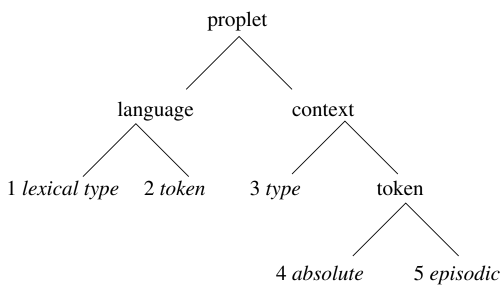
10.2 CONTEXT TYPE AND ABSOLUTE AND EPISODIC CONTEXT TOKENS



Context types and absolute context tokens take concept types as the value of their part of speech attribute (here noun), while episodic context tokens take concept tokens. Absolute and episodic proplets are further distinguished in terms of their identity and propositions numbers, which are prefixed by a and e, respectively (cf. 9.3). Context proplet types are like absolute context proplet tokens except that most of their attributes have the value NIL.

The five kinds of proplets used in dbs may be represented as the following hierarchy:

10.3 HIERARCHY OF THE FIVE KINDS OF PROPLETS USED IN DBS



The different proplet kinds have the same attributes for nouns, verbs, and adjectives, respectively. They differ only in whether certain values are types, tokens, or NIL.

Just as the sign type of symbols is based on the prior existence of concept types[12] as their literal meanings, the sign type of *names* is based on the prior existence of private markers. Consider the following illustration of the five kinds proplets of a proper name, corresponding to those of a symbol in 10.1 and 10.2:

10.4 FIVE KINDS OF PROPLETS RELATED TO A NAME

lexical type	*language token*	*context type*	*absolute token*	*episodic token*
sur: Fido	sur: Fido	sur:	sur:	sur:
noun: *dog*^{z}	noun: *dog*^{z}	noun: *dog*^{z}	noun: *dog*^{z}	noun: *dog*^{z}
cat: NM	cat: NM	cat: NP	cat: NP	cat: NP
sem: sg def	sem: sg def	sem:	sem: sg def	sem: sg def
mdr:	mdr: black	mdr:	mdr:	mdr:
fnc:	fnc: run	fnc:	fnc: have	fnc: run
idy: *&%#	idy: *&%#	idy: *&%#	idy: *&%#	idy: *&%#
prn:	prn: 32	prn:	prn: a21	prn: e14

The identity marker characteristic of a proper name is the value *&%# of the regular idy attribute of nouns. For simplicity, we will use numbers prefixed by n, e.g., n1, n2, n3, etc., to represent the idy value of names.

In addition to the idy value, the referent of a name is characterized by a concept, here *dog*^{z}, which serves as the value of the noun attribute. In the name proplet of a person like Aristotle, the concept would be *person*^{y}, in the name proplet of ship like Missouri, the concept would be *ship*^{x}, and similarly with cities, countries, cars, pet animals, etc.[13]

The concept *dog*^{z} in the lexical type, the context type, and the absolute token of 10.4 is the type of a particular dog, which is instantiated by corresponding concept tokens in the proplets of language tokens and episodic tokens. For example, the concept type of Fido may characterize it as a dog of medium size, black shaggy hair, etc., while a corresponding concept token may instantiate the hair at a particular moment as wet and dirty.

The general concept *dog* of a symbolic context type and the particular concept *dog*^{z} of a named context type may interact as follows: first, the agent recognizes a dog, using a symbolic context proplet type; then the agent recognizes the dog as Fido using a named context proplet type, and instantiating it as a named episodic proplet token. Assuming that Fido is familiar, the agent has a lexical type as well as a context type already available, both containing the same specialized dog^z concept type and the same identity marker *&%#.

A symbolic language proplet can refer to a symbolic token as well as a named token. For example, the name-based context token *Fido* may be referred to with the language proplet dog. A name-based language proplet, however, cannot refer to a symbolic context token. For example, one cannot normally use Fido to refer to a generic dog. This is because the particular concept *dog*^{z} in the name-based language proplet is a more specific type than the concept *dog* in the corresponding symbolic episodic token.

Finally consider the reconstruction of *indexicals* in the word bank example 9.3: the language proplet this is positioned opposite the episodic proplet apple serving as referent, the characteristic pointer into the context is represented by the value C, and the values SNP (singular noun phrase), -animate and eat provide additional referential restrictions.

[12]In the sense that the concept types evolve already in agents without language. Cf. 3.2 and 3.4.

[13]On the on hand, our analysis follows Frege 1896 in that name proplets contain a concept which may be regarded as providing a sense ('Kennzeichnung'). On the other hand, our analysis differs from Frege in that the reference of names depends not only on the concept, but also on the private identity marker.

Thus, the formal reconstruction of indexicals as proplets provides at least as many structural details as the intuitive approach illustrated in 5.1. A systematic analysis, however, requires more space than that remaining in this paper.

Conclusion

This paper reconstructed the language and context components of an intuitive approach to natural language communication by refining the data structure of a word bank. The motivation for this refinement was the introduction of different kinds of signs, i.e., symbols, names, and indexicals, into database semantics.

This leads naturally to the question of how exactly the mechanisms of reference characteristic of each kind of sign function in language interpretation and production. In the case of symbols and names, the storage of language and context proplets in the same token line, and the compatibility or incompatibility of their attribute values, provides for a fairly straightforward implementation, at least for literal uses. The inferences[14] needed for non-literal uses, based in part on absolute propositions, and the spatio-temporal indexing[15] needed for indexicals, however, lead into areas beyond the scope of our present topic. Apparently, the questions raised by a paper are sometimes more interesting than those it set out to answer.

References

Brooks, R., C. Breazeal, M. Marjanovic, B. Scassellati, & M. Williamson (1998) "The Cog Project: Building a Humanoid Robot," in C. Nehaniv, ed., *Computation for Metaphors, Analogy and Agents*, Vol. 1562 of Springer Lecture Notes in Artificial Intelligence, Springer-Verlag.

Dreyfus, H. (2002) "Intelligence without representation – Merleau Ponty's critique of mental representation," in *Phenomenology and the Cognitive Sciences*, Vol. 1: 367–383, Kluwer.

Frege, G. (1967) *Kleine Schriften*, ed. by Ignacio Angelelli, Wiss. Buchgesellschaft, Darmstadt.

Hauser, M.D. (1997) *The Evolution of Communication*, MIT Press.

Hausser, R. (1999/2001) *Foundations of Computational Linguistics: Human-Computer Communication in Natural Language*, Springer-Verlag, Berlin-New York.

Hausser, Roland (2001a): "Spatio-Temporal Indexing in Database Semantics," in A. Gelbukh (ed). *Computational Linguistics and Intelligent Text Processing*, Vol. 2004 of Springer Lecture Notes in Computer Science, Springer-Verlag.

Hausser, R. (2001b) "Database Semantics for Natural Language," *Artificial Intelligence*, Vol. 130.1:27–74, Elsevier, Dordrecht.

MacWhinney, B. (1999) "The Emergence of Language from Embodiment,´´ in B. MacWhinney (ed.).

MacWhinney, B. (ed.) (1999) *The Emergence of Language*. Lawrence Erlbaum, Mahwah, NJ.

Peirce, W.S. (1931 – 1958) *Collected Papers of Charles Sanders Peirce*, edited by C. Hartshorne and P. Weiss, 6 vols. Harvard U. Press, Cambridge, Mass.

Saussure, F. de (1972) *Cours de linguistique générale*, Édition critique préparée par Tullio de Mauro, Éditions Payot, Paris.

---

[14] Cf. Hausser 2001b.

[15] For a more detailed analysis of spatio-temporal indexing in database semantics see Hausser 2001a as well as work in progress.