Reconstructing Geons in Database Semantics

Roland Hausser

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

Abstract

Database semantics (dbs) was developed originally to model the basic mechanism of natural language communication (Hausser 2001). This important application is extended here to another crucial task of cognitive modelling, namely the task of pattern completion during recognition. It consists in efficiently reconstructing a complex concept from its basic parts.

It is shown that the data structure of dbs, called a word bank, supports recognition by providing for any elementary concept matching part of a complex external object all potential candidates connected to it. In this way recognition is provided with a limited set of concepts to actively try to match against other parts of the complex object. Once a second basic concept has been matched successfully and connected to the first, the database provides a much smaller list of potential candidates for a third basic concept, etc. This algorithm converges very quickly.

The basic concepts are defined as geons, in accordance with the RBC (recognition by components) or geon theory by (Biederman, 1987). Geons are structures of a complexity intermediate between features and templates, e.g., cubes, spheres, or cylinders. The approach works also for the hypothetical reconstruction of the unseen side of a known object, explained by (Barsalu, 1999) on the basis of frames.

1 INTRODUCTION

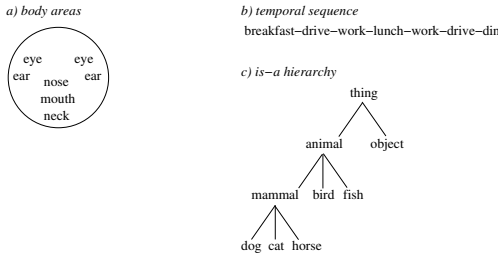
The elementary features provided by the input hardware of a cognitive agent may be combined in many different ways, in different modalities, and at different levels of complexity, creating a huge search space. The software problem of pattern matching is to classify any given input constellation without having to try a potentially very large number of interpreted patterns, as in linear search. In other words, there is the task of finding an *efficient* procedure for going from raw input data to corresponding patterns stored in memory. In addition, the system must be capable of analyzing and storing brand-new constellations of raw data, such that they will be recognized when encountered a second time.

These tasks are treated here with the hand-in-glove combination of an algorithm and a data structure. The system, called Database Semantics (DBS), determines the storage location of any input in terms of properties provided by the input analysis, such that similar analyses – modulo some primary attribute(s) – are stored at adjacent locations, in the order of their arrival (token line). Memory-based pattern completion in DBS proceeds from the partial analysis of the input to the analysis-based retrieval of similar items. It utilizes the fact that the number of relations between items stored in memory is finite and much smaller than the infinite number of relations resulting from their abstract combinatorics.

2 Representing Semantic Fields In DBS

Studies in neurology and psychology show that cognitive agents organize at least some cognitive content in terms of spatial adjacency, which can be measured in the brain and tested in psychological experiments. Such neighborhoods form coherent areas of cognitive content which are called *semantic fields* in traditional linguistics. Consider the following examples:

2.1 SIMPLIFIED EXAMPLES OF SEMANTIC FIELDS



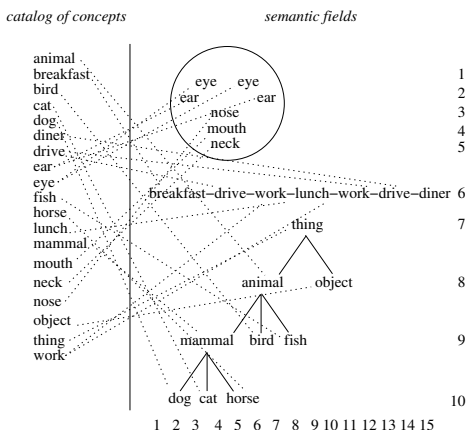
The spatial organization of these different semantic fields does not per se support efficient location and retrieval of a particular item. The situation is similar to a library where the books are stored in accordance with some physical property, for example, all small books in the first shelve, all medium size books in the second shelve, and all large books in the third shelve. Other, similar storage principles are arranging the books in accordance with the color of their covers, the temporal order of their purchase, or their topic. In the worst case the books are stored in no order all. Then we have to remember each book's location based on a mental map or search at random.

The retrieval problem posed by these different storage principles has a solution which is as general as it simple. It corresponds to the building of a library catalog. Each catalog card specifies (i) the name of the author, the title, the topic, and (ii) the location. In this way, the catalog cards may be ordered alphabetically, based for example on the last name of the author, while the books may be physically stored in accordance with any ordering principle of choice, even no principle at all.

Whenever a certain book is looked for, it is simply looked up in the alphabetical catalog and the card in question will tell its exact location. The only costs of this method are (i) building the catalog and (ii) taking care to always bring each book back to its specified location. Today these costs are minimized by using computers, which allow to do storage and retrieval automatically, greatly increasing efficiency.

Adapting the catalog solution to semantic fields results in the following structure:

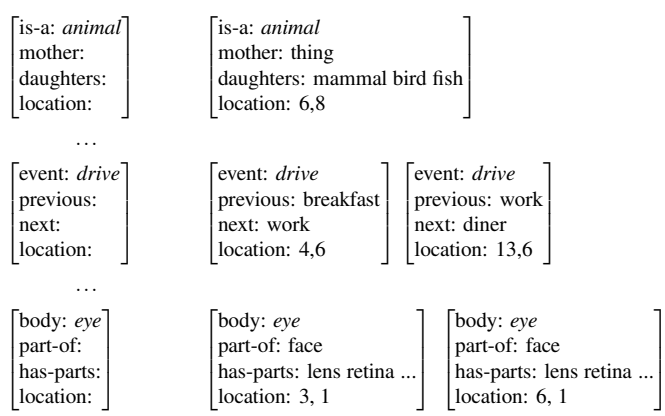
2.2 INDEXING SEMANTIC FIELDS WITH A CATALOG



This example contains the same semantic fields as 2.1. In addition, each concept is related to an ordered list by means of dotted lines. Furthermore, each location of the semantic field is characterized by an ordered pair of numbers. For example, the concept name *animal* is related by a dotted line to the corresponding node in the hierarchy at the bottom, a location also characterized by the ordered pair (6, 8).

In database semantics, the graphical structure shown in 2.2 is recoded equivalently as follows:

2.3 RECONSTRUCTING SEMANTIC FIELDS IN DATABASE SEMANTICS



This greatly simplified, partial example contains proplets from the *is-a* hierarchy (*animal*), the temporal sequence (*drive*), and a body area (*eye*). Using the grid provided by the horizontal and vertical numbering in 2.2, the spatial locations of 2.2 are represend in 2.3 as the values of the location attributes, e.g. location: 6, 8.

Recoding the different semantic fields into the uniform format of database semantics has several advantages. First, the spatial properties of the semantic fields can be reconstructed from the proplets based on the values of the location attributes. Second, being freed from the spatial restrictions of graphical representations, the data structure allows a much more detailed coding of semantic relations than 2.1 or 2.2. Third, the data structure supports an algorithm for navigating through the semantic fields based on the continuation predicates of the proplets, thus activating stored content.[1] Fourth, each item in the various semantic fields can be accessed immediately, irrespective of the spatial layout.

Thereby access is completely independent of any semantic relations or constraints, including modality. For example, the concept sequence apple smell smoooth green hard leads the agent from the modality of seeing to smelling to touching to seeing to touching. Similarly, the concept sequence city flower happy leads the agent from the semantic fields of daily life to botany to a feeling. Such concept sequences may be presented to the agent at the level of context or by means of language.

3 Pattern Completion

The homogeneous coding of different kinds of semantic fields in database semantics is based on connected proplets containing concepts. We turn now to the question of how our prelim-

[1] For example, the connected proplet *animal* with the location value (6, 8) permits upward navigation to *thing* and downward navigation to *mammal, bird*, and *fish*, the connected proplet *drive* with the location value (4, 6) permits forward navigation to *work* and backward navigation to *breakfast*, etc.

inary treatment of simple concepts like *square* (cf. Hausser 1999, pp. 56 f.) or *cylinder* can be scaled up to complex concepts like *cup, pail,* or *watering can*.

Even though the cognitive structure of concepts cannot be seen directly, we may arrive at useful details about their nature by contemplating the following questions:

1. what are the basic parts of a concept?
2. what are the basic parts made of?
3. what are the rules for combining the parts?

There is a fourth question hidden behind these three, namely: at what level of detail should the basic parts be defined?

From an abstract theoretical point of view, we are looking for that level of abstraction, where the necessary properties of the overall system cleanly separate from the accidental properties. From a computational point of view, we are looking for that particular degree of abstraction which happens to result in optimally efficient pattern recognition, indexing, retrieval, etc.

The importance of the fourth question is illustrated by the now somewhat toned down battle between two competing theories in cognitive psychology, namely features versus templates or prototypes, waged on the basis of recognition experiments. Roughly speaking, features take an atomistic position (all concepts are composed from a few elementary parts used many times), while templates take a holistic position (each concept has its own template, composition is best not used at all).

Computationally, these two extremes have inverse, complementary merits. For example, if there are only 50 different kinds of objects to be recognized, a holistic approach using templates would be reasonable: the cost of running through 50 templates in the worst case is compensated by the efficiency of the matching.

However, if there are a thousand different kinds of objects to each be matched as a whole, the cost of checking all templates in the worst case would outweigh any advantage in matching (if there is any left, given the increased complexity[2] of the templates). A compositional approach, in contrast, would do its pattern matching with comparatively few basic patterns, e.g., features, which are used over and over again to form complex concepts from basic parts.

But why choose if we can have the best of both? In cognitive psychology, there is a fairly recent proposal, called RBC (Recognition-by-Components) or geon theory, which makes the most of the inverse merits of the holistic and the atomistic approach by taking the middle of the road. Visual recognition is analyzed in terms of neither features nor templates, but rather in terms of some intermediate structures, called geons – from which all the parts for complex objects are built.

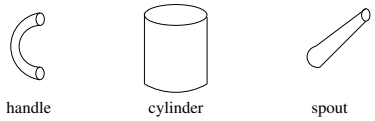
RBC or geon theory is described by Kirkpatrick 2001 as follows:

> The major contribution of RBC is the proposal that the visual system extracts geons (or geometric ions) and uses them to identify objects. Geons are simple volumes such as cubes, spheres, cylinders, and wedges. RBC proposes that representations of objects are stored in the brain as structural descriptions. A structural description contains a specification of the object's geons and their interrelations (e.g., the cube is above the cylinder).
>
> ...
>
> The RBC view of object recognition is analogous to speech perception. A small set of phonemes are combined using organizational rules to produce millions of different words. In RBC, the geons serve as phonemes and the spatial interrelations serve as organizational rules. Biedermann 1987 estimated that as few as 36 geons could produce millions of unique objects.

[2]This increase of complexity follows from the necessity to differentiate the large number of individual templates from each other.

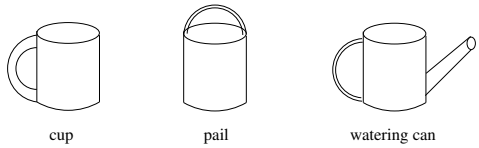
Consider the following examples of geons:

3.1 A SMALL SET OF GEONS



These geons may be assembled into different complex concepts for objects such as the following:

3.2 COMBINING THE GEONS INTO MORE COMPLEX OBJECTS

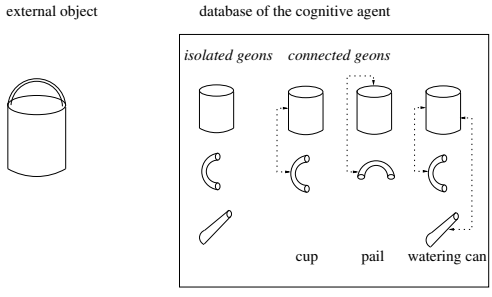


These complex objects raise the following question for database semantics: How should the combinations of geons into concepts for complex objects be stored? For example, should the handle and the cylinder of a pail be represented adjacent, as in the graphical representation 3.2, or should we specify their connection in a more abstract manner – thus opening the way to use geons as the basic key for storage and retrieval in a database?

This question bears on an important application of geons in visual recognition, namely *pattern completion*. If visual recognition is incremental, such that we see some part first and then rapidly reconstruct the rest of the object, how can we get the database to provide relevant content very fast and succinct in order to quickly narrow down the search?

It turns out that the data structure and the associated retrieval algorithm of database semantics provide a highly efficient procedure of pattern completion. As an example, consider recognition of a pail based on a word bank containing the following geons:

3.3 PATTERN COMPLETION DURING RECOGNITION



In this word bank, the isolated geons are the owner records and the connected geons the member records. For intuitive appeal, the connections between the geons are indicated by arrows. The complex objects specified in each colum of connected geons are provided with names, i.e. cup, pail, and watering can.

Given that the external object is a pail, the agent might either first recognize the handle or the cylinder – depending on the conditions of lighting or the orientation of the object. If the cylinder is recognized first, the agent's database will indicate that cylinders are known to be connected in certain ways to handles and/or spouts. This information is used to actively analyze the cylinder's relations to the rest of the external object (pattern completion), checking for the presence or absence of the items suggested by the data base.

Similarly, if the handle is recognized first, the agent's database indicates that handles are known to be connected in certain ways to cylinders to form cups, pails, or watering cans. If the handle-cylinder connection is recognized first, there are two possibilities: pail or watering can. In our example, the system determines that there is no spout and recognizes the object as a pail.

This simple algorithm of pattern completion may be described somewhat more generally as follows: Assume that the agent is faced with a complex object consisting of an open number of geons and that one of these, e.g. B, has been recognized. Then the database will retrieve all B items and list all their connections to other geons. The geons on this list are all tried actively on the external object by checking (i) whether they are present or absent in the external object and, if present, (ii) the nature of their connection to B.

As soon as a second fitting geon, lets call it C, has been found, the algorithm will retrieve all BC connections of the given type from the database, as well as all their connections to other geons. This results in a substantially smaller list. By repeating the process, the algorithm converges very quickly.

For specifying the connections between geons more precisely let us replace the intuitive graphical model illustrated in 3.3 by a full-fledged workbank format based on proplets. The following example expresses the same content as 3.3, but represents geons by names and codes the connections between geons by means of attributes.

3.4 STORING COMPLEX OBJECTS AS GEON PROPLETS IN A WORD BANK

isolated geons *connected geons*

geon: cylinder orientation: attach: onm:	g: cylinder o: vertical a: back handle o: cup	g: cylinder o: vertical a: top handle o: pail	g: cylinder o: vertical a: back handle front spout o: watering can
geon: handle orientation: attach: onm:	g: handle o: vertical a: back cylinder o: cup	g: handle o: horizontal a: top cylinder o: pail	g: handle o: vertical a: back cylinder o: watering can
geon: spout orientation: attach: onm:			g: spout o: diagonal a: front cylinder o: watering can

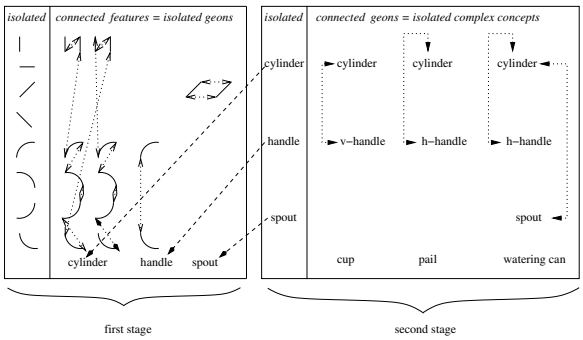
The elementary geons serve here as the values of certain attributes and are represented by words of English, e.g., cylinder, handle, or spout. The orientation of geons, represented graphically in 3.3, is represented in 3.4 by the attribute values of geon proplets, for example, horizontal, vertical, or diagonal. The connection between geon proplets is likewise coded by attributes, namely attach and onm (object name – corresponds to prn).

If the token line of, e.g., *cylinder* is activated in 3.4, the three connected proplets with the onm values *cup, pail*, and *watering can* contained in it specify explicitly which geons to look for next in the object to be recognized. They are *back handle*, *top handle*, and *front spout*.

4 Building Geons from features, Concepts from Geons, etc.

So far we have constructed relations (i) between isolated and connected geons resulting in complex concepts (Section 3), and (ii) between isolated and connected proplets (containing these concepts) resulting in propositions (Hausser 2001). The analogous relation may be constructed also (iii) between isolated and connected features, resulting in geons. Consider the following example:

4.1 CONNECTING FEATURES INTO GEONS AND GEONS INTO CONCEPTS



The isolated features on the left of the first stage are for recognizing lines and arcs of different orientation, and serve as owner records. The associated member records are connected into constellations for a cylinder (first two columns),[3] a handle (third column),[4] and a spout (fourth column).[5]

At the second stage, a constellation of connected features (raw input) is classified by means of isolated geons, here *cylinder, handle*, and *spout*, which are provided by the database. At the third stage (not shown), the constellation of the isolated geons detected is analyzed and classifed by means of isolated concepts provided by the database.[6]

The isolated concepts resulting from classifying constellations of connected geons break the pattern set by features and geons in that they in turn are not connected directly. Instead, the isolated concepts are *embedded* as attribute values into feature structures, and it is those feature structures called proplets on which the connections are defined.[7]

Combining proplets rather than concepts into connected propositions has the advantage that the building up and processing of content can function independently of the nature of the concepts contained in the proplets.[8] Our goal is to describe this independent software module in the form of a word bank, which uses proplets as its elements based on which the operations of recognition, action, and inferencing are defined.

For realizing this system as an artificial cognitive agent (robot), the features (in the sense of cognitive psychology and neurology) must be implemented as hardware for analyzing input and actuating output. Everything else, i.e., the geons, concepts, and proplets, and the functions of cognition built on top of them, is pure software.

This is in concord with the fact that humans are provided with vision, hearing, etc., by nature but have to *learn* the structures of watering cans, mathematical formulas, table manners, etc., during language acquisition, development of reasoning, social behavior, etc. This kind of learning in natural agents has at its counterpart the programming of these structures in artificial agents. Hopefully, machine-learning may soon help to obviate many of these programming chores.

The step by step buildup from isolated features via geons and concepts to concatenated propositions may be shown graphically as follows:

---

[3]The first column of connected features constitutes the left hand side and the bottom circle of the cylinder, while the second column constitutes the right hand side and the top circle of the cylinder.

[4]This column of connected features connects two ninety degree arcs into a half circle, representing a simplified, vertically oriented handle.

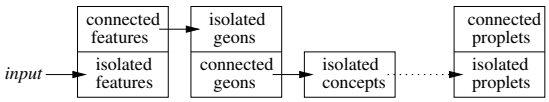
[5]The spout is indicated in a simplified manner as two parallel, diagonal lines.

[6]In other words, connected features = isolated geons, and connected geons = isolated concepts. What is defined at the lower stage is *named* at the next higher stage.

[7]In short, before the next phase of connecting takes place, the concepts are automatically transduced into proplets.

[8]This move is similar to that taken by propositional calculus or predicate calculus, where sentences or predicates are represented by abstract variables or constants.

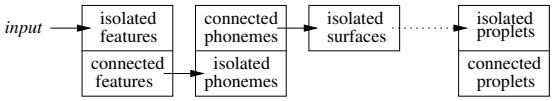
4.2 FEATURES, GEONS, CONCEPTS, AND CONTEXT PROPOSITIONS



This construction of concatenated propositions based on a set of proplets containing concepts, which in turn are based on a smaller set of geons, which in turn are based on an even smaller set of features, may be generalized from the visual modality to the other modalities. For example, the concept of a cup may be composed not only of visual geons for a cylinder and a handle, but also of 'geons' in an extended sense for a certain kind of sound, taste, or touch.

The progression from features to geons to concepts to propositions at the context level is duplicated at the language level as a progression from features to phonemes (graphemes, etc.) to surfaces to propositions. This is in accordance with the original analogy between geons and phonemes (cf. Kirkpatrick 2001 cited above).

4.3 FEATURES, PHONEMES, SURFACES, AND LANGUAGE PROPOSITIONS

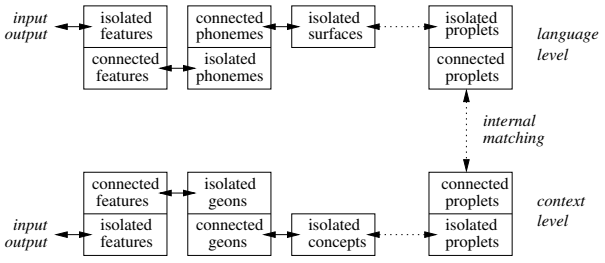


During speech recognition, the input is analyzed by means of isolated features provided by the hardware and connected to the database. Analysis of the relations between the isolated features found in the input results in connected features. These are classifed by isolated phonemes (or graphemes) provided by the database. Then the relations between the phonemes found in the input are analyzed, resulting in connected phonemes. These are classified by isolated surfaces provided by the database. The surfaces are matched with isolated proplets provided by the lexicon. Analysis of the relations between the isolated proplets found in the input, for example by means of parsing, results in connected proplets ready to be matched with the context.

During speech synthesis, the inverse procedure takes place. That is, the content intended to be uttered by the speaker is initially represented by concatenated propositions consisting of connected language proplets containing surfaces. These surfaces are reduced step by step to the kind of features which serve to synthesize the surfaces.

The transitions between features and propositions at the language level (cf. 4.3) and the context level (cf. 4.2) combine as shown in the following schema:

4.4 COMBINING THE CONTEXT AND THE LANGUAGE LEVEL



Note that the structure at the language level is exactly the same as the structure at the context level. It is just that the language level processes surfaces while the context level processes content. Otherwise the progression from features to geons/phonemes to concepts to proplets

during recognition and from proplets to concepts to geons/phonemes to features during action are the same. The clue is that the meeting point between the two levels in 4.4 consists of (i) a connected language proplet and (ii) a connected context proplet ready to be tested for internal matching (providing the relation of *reference*).

5 Outlook

The approach described is suitable also for handling another kind of pattern completion, namely the automatic reconstruction of the unseen side of a known object. For example, when we see a car from one side we have a pretty good idea of what it probably looks like from the other. This phenomenon has been explained on the basis of *frames* (Barsalu, 1999).

Like any complex object, frames may be coded as interrelated proplets (cf. 3.3, 3.4). When recognizing an object from a certain side, the associated frame is activated, filling in the unseen side as a hypothesis.

Bibliography

Barsalu, L. (1999), "Perceptual symbol systems", *Behavioral and Brain Sciences*, Cambridge University Press.

Biedermann, J. (1987) "Recognition-by-components: A theory of human image understanding", *Psychological Review*, APA.

Bresnan, J. (ed.) (1982) *The Mental Representation of Grammatical Relation*, MIT Press, Cambridge, Mass.

Gazdar, G., E. Klein, G. Pullum, & I. Sag (1985) *Generalized Phrase Structure Grammar*, Harvard U. Press, Cambridge, Mass., and Blackwell, Oxford, England.

Hausser, R. (1999) *Foundations of Computational Linguistics, Human-Computer Communication in Natural Language*. 2nd Edition 2001, pp. 578, Berlin, New York: Springer.

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

Kirkpatrick, K. (2001), “Object Recognition", in *Avian Visual Cognition*, cyberbook in cooperation with Comparative Cognitive Press.

Pollard, C. & I. Sag (1987) *Information-Based Syntax and Semantics*, CSLI Lecture Notes 13, Stanford.

Sowa, J.F. (1984) *Conceptual Structures*, Addison-Wesley, Reading, Mass.