Conceptual Analogy

Katy Börner
University of Freiburg, Centre for Cognitive Science
Institute for Computer Science and Social Research
79098 Freiburg, Germany
e-mail: katy@cognition.iig.uni-freiburg.de
phone: +49 761 203 4937
fax: +49 761 203 4938

Abstract

Conceptual analogy (CA) is an approach, that integrates conceptualization based on prior experiences, i.e. case memory organization, and analogical reasoning (Börner and Janetzko 1995, Börner and Wode 1995). It was prototypically implemented and tested to support the design process in building engineering (Börner 1994a). This paper reviews the development of CA. It provides the basic assumptions and surveys the psychological results which influenced the development of CA. It argues for the combination and integration of different proposals to memory organization, case-based reasoning and analogical reasoning to reduce the complexity of design support.

Introduction

Building engineering is one of the keystones to economic competitiveness. As a consequence, computational models for design are important research topics. Case-based design (CBD) has been suggested as an appropriate problem solving method (Goel 1989, Kolodner 1993, Domeshek and Kolodner 1992, Hua and Faltings 1993). Here, prior CAD layouts are retrieved and adapted to solve actual design problems. Several problems, however, arise. First, retrieval and adaptation of CAD layouts require to consider not only the geometric attribute values of designed objects, but most of all their topological relations. Complex case representations are needed. This increases the computational expense to retrieve, match, and adapt cases. On the other hand, short response times are crucial for the acceptance and usage of CBD systems. Second, adaptation of prior layouts mainly corresponds to adding, eliminating or substituting objects and their relations. Because of the variety and the possible combinations of these modifications, adaptation knowledge is hard to acquire. Third, graphical user interaction has been identified as a desirable feature of design support systems (Pearce, Goel, Kolodner, Zimring, Sentosa and Billington 1992). However, graphical interfaces restrict the input and output of CBD systems to CAD layouts. As far as we know, there is no approach available, which automatically extracts the knowledge needed for CBD (i.e. complex case representations, the relevance of object features and relations, and proper adaptations) from attribute-value representations of prior layouts.

This paper argues for conceptual analogy, an approach that uses huge amounts of prior layouts to support innovative design tasks. Automatic memory organization directly tailored to analogical reasoning enables computationally effective structural retrieval of adaptable layouts. The paper is organized as follows. Section 2 starts out by introducing the basic functionality CA wants to model. It motivates the need for an integration of conceptualization and analogical reasoning as well as the grounding of both processes. Our view on memory structure and their derivation during conceptualization is introduced in section 4. Section 5 introduces the notion of similarity and applies it to exploit memory structures for analogical reasoning. The paper concludes with a discussion of CA.

Outline of the Approach

This section introduces and exemplifies the desired functionality of CA. It motivates the integration of conceptualization and analogical reasoning as well as the grounding of both processes on concrete experience.

Functionality

The development of CA was motivated by the desire to support the design of complex installation infrastructures for industrial buildings. Here, the main problem is how to layout subsystems for fresh and return air, electrical circuits, computer networks, phone cables, etc. Such a design involves thousands of often incompatible objects in different stages of elaboration, at different levels of abstraction, planned at different places and by different engineers. Due to its complexity, there is hardly any operational information available on how to support design steps. Architects frequently use prior CAD layouts to inspire and guide their work. This points to case-based reasoning (CBR) (Kolodner 1993) as the predominant problem solving method.

A typical subtask occurring in building engineering is the design of interconnections between supply accesses and main accesses. Representing outlets by small squares and the main access by a square of larger size, Fig. 1 (left) depicts a simple access pattern. Different tasks (e.g. the connection of fresh air, return air, or electricity accesses) require different connection patterns. Return air supply accesses for example are connected using the shortest way. Contrary, fresh air connections take curved tracks to reduce the noise caused by the flowing air etc. Representing pipes by line segments Fig. 1 (right) illustrates three task-dependent solutions.

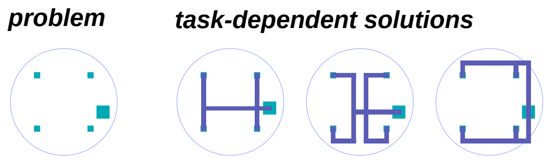

Figure 1: A problem and task-dependent solutions

To reduce the complexity of design decisions we employ a highly standardized method of construction, which was developed by architects at the University of Karlsruhe, Germany and at Solothurn, Switzerland. This method provides a standardized catalogue of parts, a standard planning grid, and a huge amount of examples of how to use it in design (Haller 1985, Hovestadt 1993b). The corresponding data representation scheme A4 (Hovestadt 1993a) represents every designed object by its placement and extension in three dimensions together with its type (e.g. room, door, furniture, lamp, etc.). Furthermore, we employ knowledge about the sequence of tasks to be tackled during the design of a building (e.g. accesses have to be designed before they are connected etc.). These predecessor relations between object types are represented by a semantic network, named *task structure*.

Retrieval and adaptation of CAD layouts, however, require to consider not only the geometrical attribute values of single objects (e.g. accesses, pipes, etc.), but most of all their topological relations (e.g. which pipe connects which accesses). Uniform topological representations of identical layouts as well as the consideration of geometrical transformations (such as reflection or rotation) are important. The computational complexity of relational comparisons and the short response time required by real world applications make a preprocessing of concrete experiences necessary. Efficient analogical reasoning requires the definition of similarity in terms of adaptability. That is, similarity should not only depend on the new problem and prior layouts but also on the adaptation knowledge available. But, how to efficiently organize CAD layouts to support design? How to ground memory organization and analogical reasoning on attribute-value input data?

Integrating Conceptualization and Analogical Reasoning

Computational analogy integrates *conceptualization* (i.e. the bottom-up formation of memory structures based on input data) and *analogical reasoning* (i.e. the top-down exploitation of conceptualizations to handle new situations). Following (Ram 1993), CA uses two processes for the bottom-up *conceptualization*. First, the incremental *construction* of memory structures from input data. Second, the *extrapolation*, i.e. extension or adaptation of existing memory structures in response to novel and unfamiliar situations.

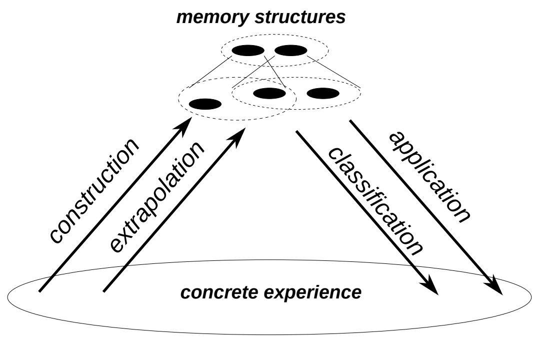

Figure 2: Conceptualization and analogical reasoning

During *analogical reasoning*, the memory structures have a top-down influence on the *classification*[1] of new situations and on the *application* (i.e. the transfer and adaptation) of prior solutions. Existing conceptualizations focus the attention to certain features of new situations and state their importance. They constitute the basis for analogical inferences and their evaluation. Figure 2 depicts these interacting processes.

Grounding

CA conforms with the idea that is be impossible to foresee and handcode all the memory structures (i.e. cases, similarity relations, and adaptation knowledge) that might be needed to handle different design tasks (Brooks and Stein 1993). Consequently, the computation of memory structures is based on the exchange of attribute-value representations of concrete experiences and the tasks of the system. Similarly, the treatment of a new problem situation depends on the focussed task and the interactions between the memory structure and the given problem.

[1]Usually, analogical reasoning is handled as a mapping from a known *source* into a novel *target*. Here we are confronted with a huge amount of prior *sources*. Thus we do not retrieve and match one *source* but classify a new problem (target) into the appropriate memory structure.

Conceptualization

Memory organization proceeds via task-oriented grouping, re-representation, and generalization over classes of similar cases. Aiming at a task-oriented user support of design tasks (Janetzko, Börner, Jäschke and Strube 1994), cases are no longer given by sets of attribute values and corresponding concepts. Instead, the task-structure may be employed to connect objects which influence the solution of a task (i.e. the *problem*) to the objects which constitute the outworked task (i.e. the *solution*) by means of a *case*. Cases which support the same task are *grouped* into a task-dependent case base. To *re-represent* geometrical layouts (cases) in terms of their topology, we use an algebraic representation (e.g. terms without variables), which was inspired by (O'Hara and Indurkhya 1994). Given layouts as depicted in Fig. 1 re-representation starts at the main access. It then continues to walk through the layout, stopping at each object and describing what is to the north of, east of, south of, west of this object, until it has visited all objects. To derive unique representations of reflected or rotated versions of a layout, we incorporate knowledge about geometrical transformations. This background knowledge is represented by a set of term rewriting rules. Output is a set of structural case representations called case class (CC).

Memory Structure

Memory structures in CA represent each case class by a *prototype* and the set of *weighted modifications* which lead to the prototype when applied to the instances.

Corresponding to psychological work, the term *prototype* refers either to a best instance of a case class (Rosch 1975), or to an abstract description of a CC that is more appropriate to some members than it is to others (Smith, Osherson, Rips and Keane 1988). The former holds if background knowledge (e.g. geometrical transformations) is available, which reduces every case into one unique instance of the CC. The latter is used if generalization or abstraction is applied to derive the common structure, i.e. prototype of a case class. Combinations of generalization and abstraction and/or geometrical transformations are possible.

Weighted modifications are employed to denote the ease to learn the case class[2], its size[3], the variability of its instances[4], and the diagnosticity of certain case attributes[5]. We distinguish two kinds of modifications, namely conceptualization rules (*c_rules*) which replace attributes and their relations by variables and analogy or adaptation rules (*a_rules*) which instantiate variables occurring in prototypes in a proper way. As we will see, this allows for automatical acquisition of memory structures as well as for their efficient exploitation during analogical reasoning.

Construction

Algebraic case representations (e.g. terms without variables) are the basis to inductively determine the prototype and corresponding modifications of every case class. As proposed by (Gero 1990), the prototype represents common features by constants or functions. Distinctive features are represented by variables. The *c_rules* and the *a_rules* define the replacement of subterms by variables and variable instantiations, respectively. They state which attributes and relations may be represented by variables and how variables may be instantiated. Weights for modification are used to denote how often a real subterm, i.e. one that is not empty, was replaced by a variable. Weights for instantiations represent how often the same term has been replaced by the same variable during the computation of the prototype[6].

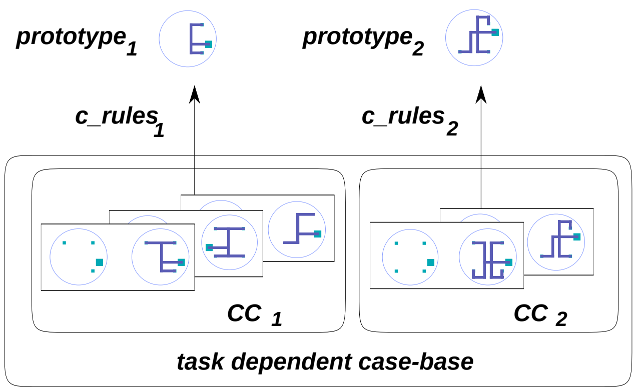

Figure 3: Memory organization

[2]Generalizations for example should be easier to learn than abstractions. In CA the algebraic knowledge representation guarantees that placement, size, type of objects or their distance to each other may be generalized (i.e. constancies are replaced by variables). Contrary, changes in the number of objects or their relations should require abstraction (i.e. function symbols are replaced by variables).

[3]The frequency of occurrence of some attribute or relation is represented by weights for modifications. Thus the prototype of a CC is independent of the number of multiple instances.

[4]In CA the variance of an attribute or relation represented by a prototype corresponds directly to the range of variable instantiation provided by the corresponding modifications.

[5]In CA the salience of attribute values is covered by weights for specific variable instantiations.

[6]The definition of a prototype is similar to the definition of *least general antiunifier (au)* (Muggleton 1992), or *most specific generalization (msg)* of a set of terms (Plotkin 1970). The only difference is, that no variable used for generalization will appear more than once within a prototype, whereas the *au* and *msg* will have the same variable at different places, if the generalized subterms at these places were equal.

Extrapolation

Extrapolation extends existing memory structures to cover novel cases. Characteristic is a reframing of what properties and relations are considered and most relevant. New cases may lead to entire reorganizations. Depending on how well a prototype matches a new case: (1) fully; (2) partially; (3) contradictory we distinguish three kinds of extrapolation. (1) the weights on modifications need to be updated, (2) a new prototype needs to be derived out of the former prototype and the new case or (3) the case class needs to be reorganized and the prototypes and modifications of the newly established case classes have to be determined.

Figure 3 sketches memory organization illustrated with cases taken from geometric layout design. Here, cases belonging to one task-dependent case base are organized in two case classes. Both, CC_1 and CC_2 are represented by their prototype and the corresponding modifications.

Analogical Reasoning

Analogical reasoning proceeds via reformulation and classification of the new *problem*, followed by transfer and adaptation of the prototypical solution. In conceptual analogy similarity is the key to recognize the case class to which a new problem belongs. Furthermore, similarity provides guidance to solution transfer and adaptation during the *application* process. So we provide the definition of similarity first.

Similarity Assessment

In CA similarity assessment proceeds via complex case representations. To handle this in a computationally effective way, a new situation is mapped against prototypes (representing case classes) instead against single cases. To be more precise, the set of modifications (*c_rules*) which lead to the prototype of a case class are applied to the new situation. Weights on modifications induce some ordering of the application of *c_rules*. Similarity is defined by *identity* or *subsumption* of the modified problem situation and the prototype (see also (Börner 1994b)).

As for the definition of similarity, the following aspects have to be considered: the size of the part of the problem which can be represented by the prototype or part of it; the size of the part of the problem which can not be represented by the prototype; and the weights for modifications, because they hint at the possibilities for adaptation. The function returns the prototype of highest similarity, i.e. the CC to classify the problem.

Classification

The classification of a new problem proceeds in two steps. First, the type values are used to select the appropriate case base. Second, the new problem is reformulated in terms of the prototypes representing the case classes of the selected case base. Therefore, the geometric attributes of the problem are matched with the corresponding objects of a prototype. Geometrical transformations are considered. Objects are connected according to the prototype, which results in a structural description of the problem. As for classification, similarity determines the prototype (i.e. the right case class out of the selected case base), which is most similar to the new problem. The most similar prototype determines the appropriate conceptual level for solution transfer and the proper modifications (variable instantiations) to be used for adaptation.

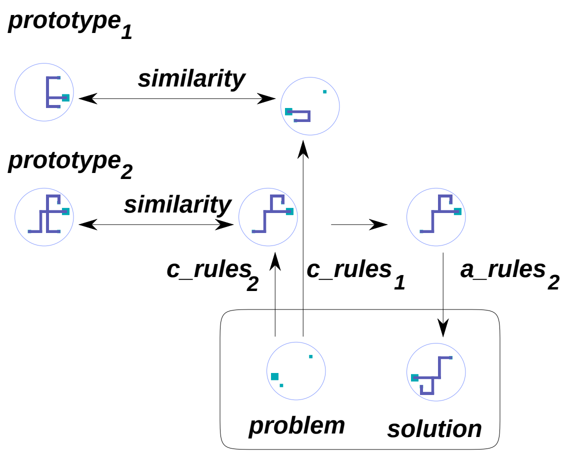

Figure 4: Analogical reasoning

Application

Given the appropriate case class a new problem situation belongs to, its prototypical solution is transferred. In the selected domain, prototypical solutions may represent general connectivity patterns of supply accesses. They are algebraically represented by terms containing variables instead of concrete placements and extensions of supply accesses. The *c_rules* applied to transform the concrete problem into the prototype are applied inversely (i.e. as *a_rules*) to instantiate the prototypical solution. Here weights for instantiations may place an ordering over the tried *a_rules*. Afterwards the solution is transformed into its attribute-value representation and graphically presented. Now the user is in the position to either accept or modify the solution or, if the solution is incomplete, to either complete or reject it. The new *case* may be added to the corresponding case class and memory organization may start again.

Figure 4 sketches analogical reasoning. We assume prior cases to be organized into two case classes (see Fig. 3). These case classes are re-represented by their prototypes and corresponding weighted modifications. Given a new problem, it is reformulated in terms of these prototypes. Because of its similarity to *prototype*$_2$ it is classified into CC_2. The prototypical solution is adapted by applying the corresponding set of *a_rules*$_2$.

Discussion

The paper sketched the basic assumptions and ideas underlying CA. Real world demands and psychological results forced the integration of conceptualization and analogical reasoning. Both processes are grounded to exploit databases of prior geometrical layouts and to provide graphical interfaces. Memory organization is directly tailored to analogical reasoning. This reduces the computational complexity associated with complex case representations. Response times which are realistic for real-world applications become possible.

CA is similar to hierarchical problem solving in that it automatically derives more general knowledge representations, i.e. prototypes for each case class. During analogical reasoning a new problem is compared to these prototypes at a more general level in which matching is less expensive than at the concrete level. Furthermore, the prototypical solution is used to guide problem solving at the concrete level. CA differs, in that it exclusively uses one case class dependent general level (i.e. the level in which the prototype of the case class is represented) for classification and application.

It should be mentioned that most work in case-based design and case-based planning follows a *solution path perspective*. For example, derivational analogy (Carbonell and Veloso 1988) constructs cases from derivational traces of planning episodes. Knowledge about how to derive solutions for problems is employed to support reasoning. This works well, if operational knowledge is actually available. In our domain prior layouts are the main knowledge source. Here CA proposes a *(problem and solution) state oriented perspective* to support design decisions.

Acknowledgements

We thank Bipin Indurkhya, Scott O'Hara, Robert Goldstone, and Gerhard Strube for their many critical comments that helped shape this research. Nonetheless the paper reflects our personal view. This research was supported by the Federal Ministry of Education, Science, Research and Technology (BMFT) within the joint project FABEL under contract no. 413-4001-01IW104. Project partners in FABEL are German National Research Center of Computer Science (GMD), Sankt Augustin, BSR Consulting GmbH, München, Technical University of Dresden, HTWK Leipzig, University of Freiburg, and University of Karlsruhe.

References

Börner, K. (1994a). Structural similarity as guidance in case-based design, *in* Wess, Althoff and Richter (1994), pp. 197–208.

Börner, K. (1994b). Towards formalizations in case-based reasoning for synthesis, *in* D. W. Aha (ed.), *AAAI-94 Workshop on Case-Based Reasoning*, pp. 177–181.

Börner, K. and Janetzko, D. (1995). System architecture for computer-aided building engineering, *6th International Conference on Computing in Civil and Building Engineering*, Berlin, Germany.

Börner, K. and Wode, H. (1995). Conceptual analogy: Integrating and automating memory organization and analogical reasoning, *submitted to the International Conference on Case-based Reasoning*, Sesimbra, Portugal.

Brooks, R. A. and Stein, L. A. (1993). Building brains for bodies, *A.I. Memo No. 1439.*

Carbonell, J. G. and Veloso, M. (1988). Integrating derivational analogy into a general problem solving architecture, *DARPA Workshop on Case-Based Reasoning*, Morgan Kaufmann Publishers, pp. 104–124.

Domeshek, E. A. and Kolodner, J. L. (1992). A case-based design aid for architecture, *Proc. Second International Conference on Artificial Intelligence in Design*, Kluwer Academic Publishers, pp. 497–516.

Gero, J. S. (1990). Design prototypes: A knowledge representation schema for design, *AI Magazine* **11**(4): 26–36.

Goel, A. K. (1989). *Integration of case-based reasoning and model-based reasoning for adaptive design problem solving*, PhD thesis, Ohio State University.

Haller, F. (1985). *ARMILLA ein Installationsmodell: Instrumentarium zur Planung von Leitungssystemen in hochinstallierten Gebäuden*, IFIB, University of Karlsruhe.

Hovestadt, L. (1993a). A4 – digital building – extensive computer support for the design, construction, and management of buildings, *CAAD Futures '93, Proceedings of the Fifth International Conference on Computer-Aided Architectural Design Futures*, North-Holland, pp. 405–422.

Hovestadt, L. (1993b). *A4 - digitales Bauen: Ein Modell für die weitgehende Computerunterstützung von Entwurf, Konstruktion und Betrieb von Gebäuden*, PhD thesis, IFIB, University of Karlsruhe.

Hua, K. and Faltings, B. (1993). Exploring case-based building design – CADRE, *AI EDAM* **7**(2): 135–144.

Janetzko, D., Börner, K., Jäschke, O. and Strube, G. (1994). Task-oriented knowledge acquisition and reasoning for design support systems, *in* R. Bisdorff (ed.), *First European Conference on Cognitive Science in Industry*, pp. 153–184.

Kolodner, J. L. (1993). *Case-based reasoning*, Morgan Kaufmann Publishers, San Mateo, CA.

Muggleton, S. (1992). *Inductive logic programming*, Academic Press.

O'Hara, S. E. and Indurkhya, B. (1994). Incorporating (re)-interpretation in case-based reasoning, *in* Wess et al. (1994), pp. 246–260.

Pearce, M., Goel, A. K., Kolodner, J. L., Zimring, C., Sentosa, L. and Billington, R. (1992). Case-based design support: A case study in architectural design, *IEEE Expert* pp. 14–20.

Plotkin, G. D. (1970). A note on inductive generalization, *in* B. Meltzer and D. Michie (eds), *Machine Intelligence 5*, American Elsevier, pp. 153–163.

Ram, A. (1993). Creative conceptual change, *in* W. Kintsch (ed.), *Fifteenth Annual Conference of the Cognitive Science Society*, Lawrence Erlbaum, pp. 17–26. University of Colorado-Boulder, June 18 to 21.

Rosch, E. (1975). Cognitive reference points, *Cognitive Psychology* **7**: 531–547.

Smith, E. E., Osherson, D. N., Rips, L. J. and Keane, M. (1988). Combining prototypes: A selective modification model, *Cognitive Science* **12**: 485–527.

Wess, S., Althoff, K.-D. and Richter, M. M. (eds) (1994). *Topics in Case-Based Reasoning – Selected Papers from the First European Workshop on Case-Based Reasoning (EWCBR-93)*, Vol. 837 of *Lecture Notes in Artificial Intelligence*, Springer.