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, i.e., memory organization based on prior experiences and analogical reasoning (Börner 1994a). It was implemented prototypically and tested to support the design process in building engineering (Börner and Janetzko 1995, Börner 1995). There are a number of features that distinguish CA from standard approaches to CBR and AR. First of all, CA automatically extracts the knowledge needed to support design tasks (i.e., complex case representations, the relevance of object features and relations, and proper adaptations) from attribute-value representations of prior layouts. Secondly, it effectively determines the similarity of complex case representations in terms of adaptability. Thirdly, implemented and integrated into a highly interactive and adaptive system architecture it allows for incremental knowledge acquisition and user support. This paper surveys the basic assumptions and the psychological results which influenced the development of CA. It sketches the knowledge representation formalisms employed and characterizes the sub-processes needed to integrate memory organization and analogical reasoning.

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). Prior CAD layouts are retrieved and adapted to solve actual design problems. Due to the characteristics of the domain several problems arise. First of all, graphical user interaction has been identified as a desirable feature of design support systems (Pearce, Goel, Kolodner, Zimring, Sentosa and Billington 1992). This restricts the input and output of CBD systems to CAD layouts, i.e., to sets of designed objects each represented by its attribute values. The retrieval and adaptation of cases (CAD layouts), however, require the consideration of not only of the geometric attribute values of designed objects, but most of all of the topological relations among objects. Complex case representations are needed to represent cases in terms of their topology. Likewise, a new problem is represented by the attribute values of a set of graphically selected objects. These "raw data" have to be reformulated in topological terms to make them comparable to past experience. The complex case representations increase the computational expense in retrieving, matching, and adapting cases. However, short response times are crucial for the acceptance and usage of CBD systems. Efficient memory organization directly tailored to analogical reasoning becomes essential. Another problem concerns the additional knowledge needed for design support. Architects are not able to give any definition of what it means for layouts to be similar. Hardly any information about the relevance of single attributes or objects is available. The 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 by hand. 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 extract the knowledge needed to support innovative design tasks. The approach provides automatic memory organization directly tailored to analogical reasoning thus enabling 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 provides the notion of similarity and applies it to exploit memory structures for analogical reasoning. The paper concludes with a discussion of conceptual analogy.

Outline of the Approach

This section introduces and exemplifies the desired functionality of conceptual analogy. It promotes 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. The main problem in this area is how to layout subsystems for supply air, 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, Aamodt and Plaza 1994) as the predominant problem solving method.

A typical subtask occurring in building engineering is the design of interconnections between accesses and outlets. Fig. 1 (left) depicts a collection of accesses (represented by small squares) and one outlet (indicated by a larger square). Different tasks (e.g., the connection of supply air, return air, or electricity accesses) require different connection patterns. Return air supply accesses, for example, are connected in the shortest admissible way. Supply air connections, on the contrary, take curved tracks to reduce the noise caused by the flowing air etc. Thus different tasks require different interpretations of and different solutions to solve the attribute-value description of the problem. Figure 1 (right) depicts three task-dependent solutions that properly connect the outlet to the single accesses (pipes are represented by line segments).

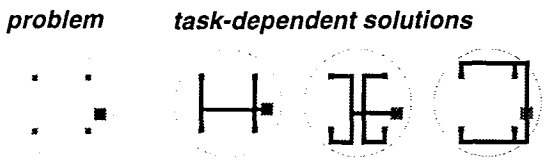

Figure 1: Problem and task-dependent solutions

As the basic knowledge representation we employ A4 (Hovestadt 1993), developed at the Institute of Industrial Building Design, University of Karlsruhe, Germany. It represents every designed object by its *geometric* attribute values (i.e., placement and extension in three dimensions) and its *type* attribute values (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 requires the consideration of not only the geometric attribute values of single objects (e.g., accesses, pipes, etc.), but most of all of 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 can CAD layouts be efficiently represented and organized to support design? How can memory organization and analogical reasoning be grounded on attribute-value input data?

Integrating Conceptualization and Analogical Reasoning

Conceptual 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 in handling new situations). Following (Ram 1993), CA uses two processes for the bottom-up *conceptualization*. Firstly, the incremental *construction* of memory structures from input data and secondly, the *extrapolation*, i.e., the extension of existing memory structures in response to novel and unfamiliar situations.

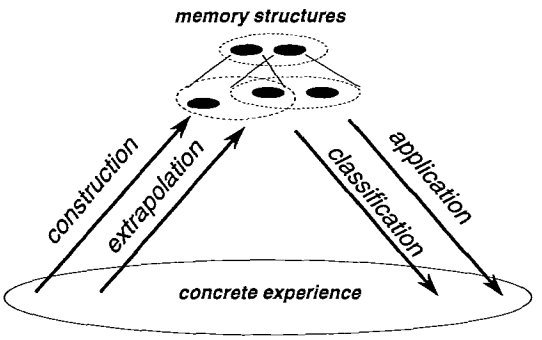

Figure 2: Conceptualization and analogical reasoning

During *analogical reasoning*, the memory structures have a top-down influence on the *classification*[1] of new

[1] Usually, analogical reasoning is handled as a mapping from a known *source* into a novel *target*. Here we are con-

situations and on the *application* (i.e., the transfer and adaptation) of prior solutions. Existing conceptualizations focus attention on certain features of new situations and state their importance. They constitute the basis for analogical inferences and their evaluation. Figure 2 depicts these four strongly interacting processes.

Grounding

Conceptual analogy conforms with the assumption that it is 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 needs to be grounded on the exchange of attribute-value representations of concrete experiences and the tasks of the system. Similarly, the treatment of a new problem situation will depend on the focussed task and the interactions between the memory structure and the given problem.

Conceptualization

As for *conceptualization*, pragmatic and structural features force the bottom-up acquisition and organization of past experience.

Aiming at the support of design tasks there are no natural units that may define the granularity of cases[2]. The methodology of task-oriented knowledge acquisition (Janetzko, Börner, Jäschke and Strube 1994) may be applied here. It provides a way to employ the knowledge encoded in the *task structure* to connect objects which influence the solution of a task (i.e., the *problem*) to the objects which constitute the completed task (i.e., the *solution*) by means of a *case*. Note that there is no information about solution paths (i.e., how to derive the set of solution objects from the problem objects) available. Cases represent exclusively collections of objects where each object is represented by its geometric and type attribute values.

Cases which support an identical task are *grouped* into a task-dependent case base. Thus, case-bases provide uniform comparison and reasoning context essential in providing design support. To *re-represent* geometric layouts (cases) in terms of their topology, we use an algebraic representation (terms without variables) inspired by (O'Hara and Indurkhya 1994). Given the attribute value representation of layouts (cases) as depicted in Fig. 1, re-representation starts at the main access and continues its way through the layout, stopping at each object to describe what is north, east, south, and west of it, 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. Structural features are used to organize the set of structural case representations into *case classes* showing similar structure.

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 cases. 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, cf. (Börner 1994b).

Weighted modifications are employed to denote the degree of difficulty in learning the case class[3], its size[4], the variability of its instances[5], and the diagnosticity of certain case attributes[6]. 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 memory structure allows for its automatic acquisition as well as for their efficient exploitation during analogical reasoning.

Construction

Construction determines memory structures from sets of cases. At the beginning, all cases may belong to one

fronted with a huge amount of prior *sources*. Therefore, we do not retrieve and match one *source* but classify a new problem (target) to the appropriate memory structure.

[2]For analytic tasks the observations, symptoms, and diagnoses are quite ready at hand to be converted into the standard representation of cases.

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

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

[5]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.

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

case class. During *extrapolation* case classes are further divided to better reflect the common structure of cases. As for construction, algebraic case representations (e.g., terms without variables) are the basis for inductive determination of the prototype and corresponding weighted modifications for 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 with variables and how variables may be instantiated, respectively. 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[7].

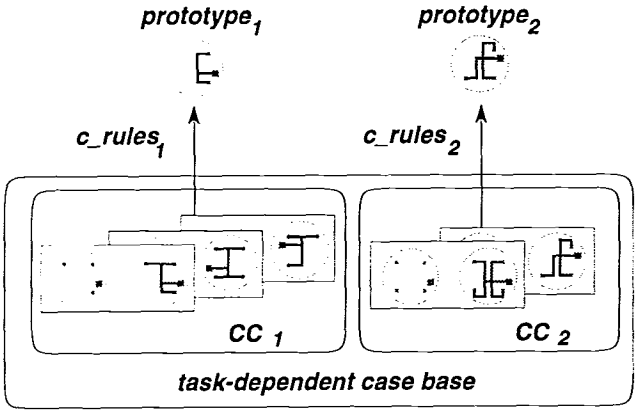

Figure 3: Memory organization

Extrapolation

Extrapolation describes the conceptual clustering techniques applied to partition a set of structurally represented cases into case classes showing similar structure. The quality of a case class correlates to the quality of the prototype that represents it. The quality of a prototype is good if it contains most of the information stored in the cases of a case class. That is, the *size* of the prototype has to be somewhat close to the average size of the terms in the case class. A fixed *quality_threshold* value determines the number of case classes representing a set of cases. With a *quality_threshold* = 1, the size of the prototype is required to equal the average size of terms the case class represents. That is, only identical cases are classified in one case class and the case class is represented by a prototype equaling these cases. On the other hand, with a *quality_threshold* = 0 the entire set of cases is treated as one case class. It is represented by one prototype that may at the very worst be a variable. The *quality_threshold* value may be provided handcoded or may be computed from the response time available to derive a solution for an actual problem and the confidence required for the solution.

Two kinds of extrapolation can be distinguished. Firstly, in an initial phase it may be advantageous to extrapolate a huge set of prior cases. An agglomerative method treats each case as a separate class at the beginning. These small, specific classes are repeatedly combined to form larger, more general classes that still fit the required *quality_threshold* value. Secondly, during the systems usage new cases provided by the user or suggested by the system need to be incrementally incorporated into existing memory structures. Depending on how well the existing case classes match a new case, the new case may be added into an existing case class or the case class may need to be divided to meet the required *quality_threshold*. Subsequently, the prototype and modifications of the modified case classes are constructed.

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, CC_1 and CC_2. Both case classes are represented by their prototype and the corresponding sets of modifications, c_rules_1 and c_rules_2.

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 in recognizing the case class to which a new problem belongs. Furthermore, similarity provides guidance to solution transfer and adaptation during the *application* process. We provide the definition of similarity first.

Similarity Assessment

Similarity assessment proceeds in CA via complex case representations. To handle this in a computationally efficient way, a new situation is mapped against prototypes (representing case classes) instead against single cases. As for the definition of similarity, the following aspects have to be considered: the size of the problem which can be represented by the prototype or part of it, and the weights for modifications, because they hint at the possibilities for adaptation. The function returns the prototype of highest similarity and the corresponding CC to classify the problem.

[7]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.

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, cf. (Börner 1994b).

The applied definition of structural similarity provides not only the most appropriate category it also mediates and guides solution transfer and adaptation. Similarity depends on the relevance of case features and the adaptation knowledge available. It guarantees the retrieval and matching of not only similar, but also useful experience. The selected case class provides a prototypical solution together with proper adaptations to solve the new problem analogously. At the same time, the prototype provides the appropriate level of solution transfer.

Classification

The classification of a new problem proceeds in two steps. Firstly, the type values are used to select the appropriate case base. Secondly, the new problem is reformulated in terms of the prototypes representing the case classes of the selected case base. The essential process is that problem objects are matched with the objects of a prototype and are connected accordingly. Geometrical transformations are considered. Usually, a partial structural description of the problem results. In terms of 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.

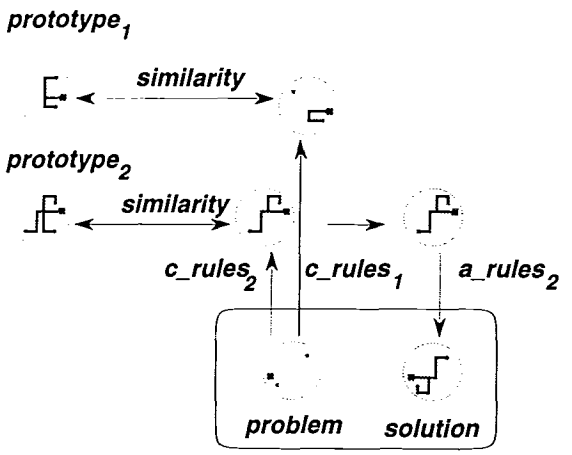

Figure 4: Analogical reasoning

Application

The case class which a new problem situation belongs to provides a prototypical solution and a set of weighted *a_rules* (remember that this kind of knowledge representing the case class has been automatically derived during memory organization).

Prototypical solutions may represent general connectivity patterns of supply accesses in the selected domain. They are algebraically represented by terms containing variables instead of concrete placements or extensions of supply accesses. The set of *a_rules* fixes the space of proper instantiations traversed in adapting the prototype to the actual requirements. Weights for instantiations may place an ordering over the attempted *a_rules*. The complete solution is transformed into its attribute-value representation and presented graphically. Now the user is in the position to either accept, modify, or reject the proposed solution. The new problem and its solution is subsequently added to the corresponding case class and memory organization starts again. If an actual problem cannot be reformulated, classified, or solved, its user-provided solution is incorporated into the existing memory structure and may be advantageously applied the next time.

Figure 4 sketches analogical reasoning. We assume the selected task-dependent case base to be partitioned into two case classes (see Fig. 3). These case classes are represented by their prototypes and corresponding weighted modifications. A new problem is reformulated in terms of these prototypes. While $prototype_1$ allows only for a partial reformulation of the problem, i.e., one access is not covered at all, part of $prototype_2$ may be transferred to completely reformulate the problem description. Because of its similarity to $prototype_2$, the reformulated problem is classified into CC_2. The prototypical solution is adapted by applying the corresponding set of a_rules_2. In the selected example the reformulated problem already equals the complete solution. Its rotation about 180 degrees results in the correct solution.

Discussion

The paper argued for *conceptual analogy*, an approach that integrates and automates conceptualization and analogical reasoning on the basis of attribute-value data. The approach provides a number of features that improve the applicability of CBR techniques to support real-world design tasks. Problem formulation and solution presentation proceed graphically via CAD layouts. All knowledge needed to support innovative design tasks (i.e., the relevance of objects and relations, proper adaptations) are derived from prior cases representing layouts. Algebraic case representations allow the exploitation of powerful mechanisms like term rewriting techniques and antiunification. The organization of the knowledge is directly tailored to analogical reasoning. Prototypes representing case classes effectively short cut much of the memory retrieval effort that would be necessary to check each past case separately for structural similarity. The complexity of structural mappings can be handled in an economical

manner. Additionally, weights are employed to improve the accuracy and speed of analogical reasoning. All together increases the overall problem solving efficacy by decreased computational complexity of both processes. Response times which are realistic for highly interactive support systems become possible.

Overall, CA tightly integrates short-term and long-term adaptation. As for short term adaptation, CA handles the current situation in terms of its memory structures and in long term adaptation, new experiences are added to the memory, affecting not only the knowledge stored there but also the organization of this knowledge. The memory structures used to guide reasoning can be best viewed as the result of compiling past experience. As new memory structures are learned from experience, the way the reasoner performs short-term adaptation is affected.

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. Cases represent knowledge about how to derive solutions for problems and are employed to support reasoning. This works well if operational knowledge is actually available. In our domain prior layouts are the main knowledge source. Here the approach of conceptual analogy proposes a *(problem and solution) state oriented perspective* to support design decisions.

Acknowledgements

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

References

Aamodt, A. and Plaza, E. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches, *AICOM* **7**: 39–59.

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. (1995). Interactive, adaptive, computer aided design, *accepted at International Conference CAAD Futures 95.*

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.

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.

Hovestadt, L. (1993). 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.

Hua, K. and Faltings, B. (1993). Exploring case-based building design – CADRE, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* **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.), *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, Lawrence Erlbaum Associates, pp. 17–26.

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 *LNAI*, Springer Verlag.

Appendix: Describing the contribution

A. Reasoning framework:

1. Reasoning framework: pragmatic and structurally sensitive memory organization and and analogical reasoning (conceptual clustering, structure mapping).
2. Benefits from adaptation: Building engineering is far to complex to support reasoning from scratch. General design solutions may be extracted and applied to guide analogical problem solving.
3. Specific benefits and limitations: CA automatically extracts the knowledge needed to support design tasks (i.e., complex case representations, the relevance of object features and relations, and proper adaptations) from attribute-value representations of prior layouts. Memory organization directly tailored to analogical reasoning reduces the complexity of analogical reasoning.
   CA is restricted to innovative design tasks. No merging of cases is possible. It was applied to geometric layout design. Other tasks may require different case representations and background knowledge.
4. Roles of adaptation, knowledge, and reuse in your approach?
   Knowledge: Cases: geometric features and topology of CAD layouts; background knowledge: uniform representations of identical layouts, geometrical transformations; task structure: the sequence of objects to be designed
   Knowledge representation: cases: attribute value pairs and terms; background knowledge: term rewriting rules; task structure: semantic network
   Usage: explained in the paper - sorry, no space here
   Acquisition: automatically extracted from CAD layouts
   Adaptation: What: position, orientation, and topology of geometric layouts (prior cases) *Why:* to correctly solve the actual problem *Properties:* cases represented by ground terms, background knowledge represented by term rewriting rules, prototypes represented by terms.
   Reuse What: the problem is classified into a case class that provides the most similar prototype inclusive its proper modifications (adaptations) *Properties:* modifications which lead to the prototype of the selected case class may be useful in adapting the prototypical solution to the new problem.

B. Task:

1. Task and domain: innovative design of geometric layouts in building engineering.
2. Inputs: CAD layouts represented by attribute-value pairs; background knowledge about uniform representations of layout topologies and geometrical transformations.
3. Outputs: Adapted CAD layouts, i.e., objects designed conform to prior layouts.
4. Constraints on the outputs: Output corresponds to the input knowledge available.
5. Characteristics of the domain that the method relies on: Time and functional knowledge is partially represented by the task structure and is employed organize and use knowledge in a pragmatic way. Geometric features and topological relations are used for structurally sensitive memory organization and analogical reasoning.

C. Evaluation:

1. What type of evaluation? If empirical: What independent and dependent variables? If mathematical: What model?
   The evaluation of the used memory organization and analogical reasoning is not presented in this paper.
2. What comparisons were made with other methods?
   We compared CA to approaches which also use complex but operational case representations to support CBD. The memory organization and reasoning used by CA is similar to hierarchical problem solving.
3. What are the primary contributions of your research?
   Conceptual analogy is a general approach that integrates and grounds memory organization and analogical reasoning on prior experience. It therefore identifies the sub-processes that are needed and proposes general knowledge representation formalisms, knowledge acquisition techniques, and reasoning techniques (instead of domain specific heuristics).