Structural Adaptation with TOPO

Angi Voß and Carl-Helmut Coulon
GMD - German National Research Center for Information Technolgy, FIT-KI
D-53754 Sankt Augustin
angi.voss@gmd.de

1 About the approach

The approach of TOPO was originally developed in the FABEL project[1][1] to support architects in designing buildings with complex installations. Supplementing knowledge-based design tools, which are available only for selected subtasks, TOPO aims to cover the whole design process. To that aim, it relies almost exclusively on archived plans. Input to TOPO is a partial plan, and output is an elaborated plan. The input plan constitutes the query case and the archived plans form the case base with the source cases. A plan is a set of design objects. Each design object is defined by some semantic attributes and by its bounding box in a 3-dimensional coordinate system. TOPO supports the elaboration of plans by adding design objects.

The design objects in a plan form structures. TOPO assumes that these structures constitute the major meaning of the plan. The structure of a plan is caused by various requirements and constraints, which can be technical, customer-specific, or stylistic. It is impossible to acquire all this knowledge. Instead, TOPO assumes that the structure itself is meaningful and can be transferred without further rationalization. Based on this hypothesis, TOPO offers a knowledge-poor but general approach to structure adaptation. Esssentially, it transforms the plan into a graph, retrieves similar plans by graph matching, and adapts a plan by transferring paths from the source case to the query case.

2 Knowledge needs

TOPO can operate on arbitrary sets of objects. It needs only three parameters:

1. A set of functions recognizing relations between the objects that will become the edges of the graph (n-ary relations yield hypergraphs).
2. Typing functions mapping design objects and relations onto a set of types (for graph matching only types will be considered).
3. For each relation an object construction function that, given a full list of arguments, completes a partial set of matching arguments (to transfer paths by creating new objects in the query plan).
4. Optionally, heuristics for selecting paths for transfer.

No more knowledge is needed for statistical assessment. If adaptation creates subgraphs that are statistically rare wrt. the case base, they must be unusual. To decide whether they are either wrong or innovative is up to the user.

In general, the relations (1) should be chosen as primitive as possible, in order to minimize knowledge acquisition and to speed up runtime, and they should be broadly applicable to cover a large portion of the design process. In the building domain, the relations were straightforward: a bounding box appears as an interval in each of the three dimensions, and there are 12 possible relations between two intervals, i.e. 12 up to the 3 dimensions. They can directly be computed from the coordinates. The typing function (2) was straightforward, too. Design objects were typed by their semantic attributes. In order to use the relations to reconstruct (3) the position of objects some of them are extended by the parameter d quantifying the distance between the intervals. The only heuristic (4) used so far is taken from a process model of the design process given by architects. It describes dependencies between types of objects. For example, in figure 1 return air pipes depend on shafts and return air outlets. Therefore, if the topology of the shafts and return air outlets match, it might be useful to transfer the return air pipes of the case.

3 Retrieval and indexing

Ideally, retrieval should return cases with large common subgraphs. Additionally, the potentially transferable part of the case should neither be too small (too low a gain) nor too large (too risky). Retrieval could be improved by heuristics that predict gain and risk dependent on the types of objects. Match and prediction are performed by TOPO.

Since graph matching is NP-complete, a fast preselection is performed that only compares the cases with respect to the

[1] This research was supported by the Federal Ministry of Education, Science, Research and Technology (BMBF) within the joint project FABEL under contract no. 01IW104. Project partners in FABEL are German National Research Center for Computer Science (GMD), Sankt Augustin, BSR Consulting GmbH, München, Technical University of Dresden, HTWK Leipzig, University of Freiburg, and University of Karlsruhe.

ECAI 96. *12th European Conference on Artificial Intelligence*
Edited by W. Wahlster
Published in 1996 by John Wiley & Sons, Ltd.

numbers of objects and relations of identical types. Among the preselected cases, the user can choose one, or TOPO can selects one with respect to matching common subgraph, gain and risk prediction.

4 Adaptation

In the domain of the FABEL project, knowledge-based tools are limited in their scopes and cover only islands of the design process. A knowledge-poor, case-based reasoning method should cover the whole process. A pure retrieval method is insufficient. The design space is absolutely too large to suggest an exhaustive case base. Furthermore, retrieving an identical source case would not help, because it offers nothing to elaborate the query case. Therefore an adaptation method is necessary.

TOPO translates plans into graphs by linking design objects with primitive topological and spatial relations (in this application). Given two plans, a source case and a query case, one of their largest common subgraphs is computed via a maximal clique-finding algorithm. This yields a match between the two cases. The common subgraph dynamically splits the source case into a (matching) problem part and (non-matching) solution part. Now paths from the solution part that originate from the problem part are selected (heuristically or by the user) and are transferred from the source case to the query case. Figure 1 gives an example.

TOPO does not simultaneously adapt multiple cases. It can be iteratively applied to its own results, yielding an incrementally growing plan. In principle TOPO could simultaneously adapt multiple cases, and there would be no limitations as to their overlaps. Anyway, TOPO would not recognize any conflicts. All it can do is check a graph for statistically unusual substructures. If the user indicates them as incorrect, TOPO starts another retrieval.

Supplied with heuristics, TOPO could operate on its own. But it is meant as an interactive tool. The user can choose the case to be used for adaptation, and he can choose the parts to be transferred. The user must judge whether unusual substructures are innovative or unacceptable. The prediction of gain and risk can be indicated to the user.

If the matching step takes too long, the user can stop the process. During matching the user is supplied with information about the size of the largest subgraph found so far and how far the search space has been exhausted. After interruption the user can specify a new query case with less objects and relations.

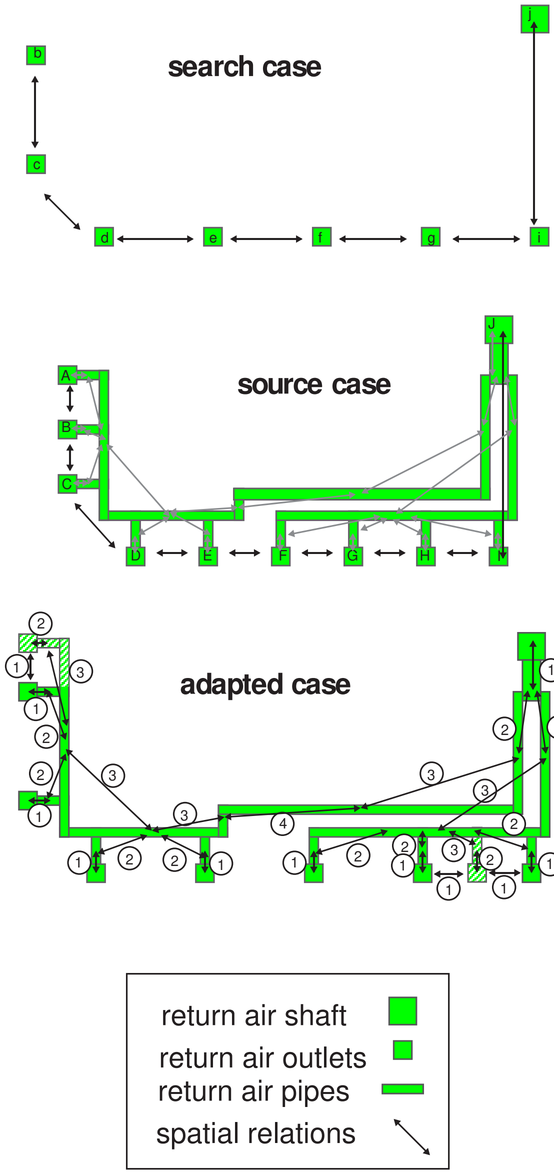

Figure 1. Structure transfer by TOPO. Both, query and source cases contain a shaft for return air and an arrangement of return air outlets. In the source case, they are connected by pipes. There is no unique match, and one is chosen by chance (dark arrows in search and source case). The paths are transferred incrementally as indicated by the numbers in the adapted case. It is not always desirable to transfer all paths, as the ones leading to outlets A and I. Therefore, the designer is asked before transfer, or he can repair the layout, or additional domain-specific heuristics must be added.

5 Strengths, limitations, evaluation

TOPO is poor of knowledge and quite general. It is applicable to cases that contain a set of objects with an implicit or explicit structure. The structure should be made of primitive relations, but there are no restrictions as to the kind of relations. We currently think of molecules, geographic maps, and project plans as other domains.

TOPO is almost purely case-based, and hence as good as the cases and as good as the continuity assumption holds. There is no guaranteed quality or security, but coverage is good, because there is no limit to the size of the common subgraph. Thus, the approach should be used when there are no other methods.

At the time of the workshop, there will be many cases available and TOPO will have been tested empirically with respect to acceptance by an architect and according to an evaluation method described in [3].

The expensive part of TOPO is the graph matching. TOPO will warn the user by providing information about the search progress.

6 Comparison

6.1 Structure adaptation

TOPO needs a structural representation because its major hypothesis is that the relations between a set of objects are most relevant. To improve efficiency of graph matching, the graphs are first compared wrt. their sets of object and relation types. How much to transfer is left to the user or to heuristics. The functions recognizing relations between objects typically perform some abstraction. In the FABEL domain, quantitative distnaces are classified relative to the size of the involved objects. The larger the transferred structure, the more progress can be achieved and the more risky is the result.

RESYN/CBR Here a query case is a molecule and a source case contains a molecule together with a plan to synthesize it. A similar source case is found by applying a set of rewrite rules to the synthesis plan until the molecule to be synthesized is identical to the query molecule. TOPO might solve the problem as follows: For retrieval it would interpret the molecules as graphs and match the query molecule with the molecules synthesized in the source cases. Then it would transfer the synthesis plan from the source case to the query case. Doing so, it must create intermediate objects for each molecule in the plan. For that purpose, TOPO needs a reconstruction function for each transition. Could these functions be defined statically in the domain of RESYN/CBR? Is graph matching too expensive?

The decision support system from Lyon Here a query case is a "complex structure with multi-valued attributes". A source case is a submodel of the conceptual graph of domain knowledge. Matching is performed by comparing event sequences and involved concepts. Similar source cases suggest submodels to explain (solve) the query case. Is the query case a graph or could it be a graph? Then TOPO's graph matching could be used for retrieval. Also, TOPO would transfer relations from the source case to the query case. The problem is which relations should be transferred. TOPO assumes that a source case may contain more relations than are relevant for a particular query case and therefore uses heuristics or leaves the choice to the user. The system from Lyon assumes that a case constitutes a meaningful context or a view. Therefore, all relations from a source case can be transferred.

6.2 Case structure

For design tasks, any partial plan (a spatial plan, not a temporal one) constitutes a query case. Therefore, a source case should contain a single plan, not separate problem and solution plans. This is particularly reasonable when there is no predefined design procedure. Any subplan of the plan in the source case constitutes a possible query case, and its complement a solution. Since there are many subplans, they should be determined dynamically. This is an interpretative approach. Splitting the case into many cases with different problem and solution parts or splitting it into subcases is always a precompilation. It increases the number of cases base and restricts a flexible use of the cases – interpretation is more flexible than precompilation. Since constraints are undirectional, design cases can very well contain a plan enriched by a set of constraints.

Taking plans as cases, there is no notion of a goal. So cases cannot indexed by goals. Similarity must rely on some kind of structure matching and hence may be expensive.

Like design, decision support seems to be an undirected activity. The same situation may be reached from different states. Therefore, the decision support system from Lyon and INRECA (for decison support) do no explicitly distinguish problem and solution parts.

In contrast, diagnosis and classification start with a precise question and deliver a precise answer. Exploiting this knowledge for statically structuring cases and for indexing them may speed up retrieval. What makes it a difference for INRECA when the user indicates solution features? Is the gain in efficiency the only reason?

What about (temporal or sequence) planning as in PARIS or RESYN/CBR? A synthesis plan in RESYN/CBR could be used to synthesize any molecule on its path, couldn't it? Similarly, could we not reuse any partial plan from PARIS?

REFERENCES

[1] C.-H. Coulon, W. Gräther, B. Schmidt-Belz, A. Voß, F. Gebhardt, E. Groß, and J. W. Schaaf, 'Virtual Building Site', in *Artificial Intelligence in Design*, eds., John S. Gero and Fay Sudweeks, Stanford, (1996).

[2] Carl-Helmut Coulon, 'Automatic Indexing, Retrieval and Reuse of Topologies in Architectural Layouts', in *CAAD Futures '95, Proceedings of the Fifth International Conference on Computer-Aided Architectural Design Futures*, Singapore, (1995).

[3] Carl-Helmut Coulon and Friedrich Gebhardt, 'Evaluation of retrieval methods in case-based reasoning', in *EWCBR-94: Second European Workshop on Case-Based Reasoning*, eds., Mark Keane, Jean-Paul Haton, and Michel Manago, pp. 283–291, Chantilly, (1994). AcknoSoft Press, Paris.