Automatic Indexing, Retrieval and Reuse of Topologies in Architectural Layouts

Carl-Helmut Coulon

GMD
German National Research Center for Information Technology
D-53754 Sankt Augustin
GERMANY

Former layouts contain much of the know-how of architects. A generic and automatic way to formalize this know-how in order to use it by a computer would save a lot of effort and money. However, there seems to be no such way. The only access to the Know-how are th.e layouts themselves. Developing a generic software tool to reuse former layouts you cannot consider every part of the architectural domain or things like personal style. Tools used today only consider small parts of the architectural domain. Any personal style is ignored Isn't it possible to build a basic tool which is adjusted by the content of the former layouts, but maybe extended inclemently by modeling as much of the domain as desirable? This paper will describe a reuse tool to perform this task focusing on topological and geometrical binary relations.

Keywords: case based reasoning, graph matching, structural comparison, topology

1 Introduction

The main idea of TOPO is to transfer the knowledge about useful topologies from former layouts to a new problem in order to correct, extend or detail the query. The transfer uses no application specific knowledge besides the information stored in the layouts and the definition of topology. Therefore it is possible to support layout in any kind of application if there exist former layouts. Figure 1 shows one of the former layouts used to support the layout of air-condition, water and lights.

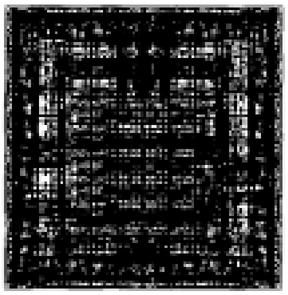

Figure 1: One of a set of former layouts: Subsystems of two floors.

The first part of this paper will describe a scenario followed by the description of the used knowledge representation and the approach itself. Both parts use an example from the domain of office-layout in order to explain the main concepts. Section 5 gives an example of the results of current implementation using the layout of figure 1 to support the layout of supply pipes.

2 Example

The designer marks a situation using his CAD-tool by selecting (Figure 2(1)). He may add objects to be placed. On demand TOPO interprets this situation as a query and analyzes the topological and geometrical relations between the objects considering their attributes. The collection of former designs was analyzed the same way and is now compared with the problem. The retrieval result (Figure 2(2)) is a part of a former design including the maximum number of objects which match the objects of the problem by their relations.

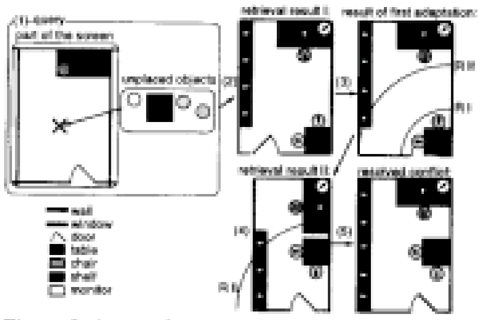

Figure 2: A complete run of TOPO.

TOPO displays the surrounding of the result in order to detail or complete the query. The users elects part of it and the system transfers the objects and relations to the query (smart paste 2(2)). Adaptation (3) cannot transfer all relations, because the table and two chairs cannot be placed between the door and the right wall like in the former design. Additionally the collision of a door with a table and a chair points to a conflict at the lower right corner, because it is unusual for office designs.

Conflicts which are confirmed by the user are automatically interpreted as new queries. Topo searches for the smallest part of a former design including the same objects, but with correct relations (Figure 2(4)) in order to resolve the conflict. The last step transfers the topology of this part to the query.

3 Knowledge and its representation

Layouts are the only necessary knowledge source of TOPO. It extracts two informations from them, a topological representation and a statistics about the frequency of topological relations (compare figure 3).

The topological representation consists of typed 3-dimensional binary relations of various types. The type of a relation is determined by the type of the involved objects and their 3-dimensional relation (Figure 3). The six shown and their opposites. The symbolical representation is like the common representation used in the field of pattern recognition [1]. In order to use them to reconstruct the position of objects some of them are extended by the parameter d which determines the distance marked in the figure.

A 2-dimensional view of an 3-dimensional example is shown in Figure 4. Relations of connected objects are visualized by circles. Additionally all relations are listed. The example shows the relation between a window and a table. The window includes the table in x-dimension, touches it from above in y-dimension arid overlaps it from above in z-dimension. The type of this relation, represented using the symbols used in figure 3, is "window]_xI_y/_ztable".

The definition of relations reveals the only restriction to layouts required by TOPO: The layouts must consist of typed objects. In the actual application TOPO has to deal with several thousands of object types corresponding to the types of elements of a building plan.

The statistics describes the frequency of occurences of relation types for each pair of object types. For example, 95% of air-condition supply zones have a relation to air-condition connection zones, which overlap them in x- and y-dimension and touch them in z -dimension.

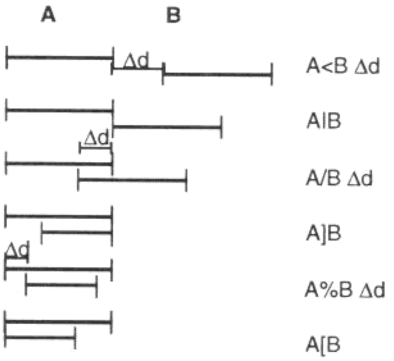

Figure 3: Two objects can be related in 12 different ways for each dimension.

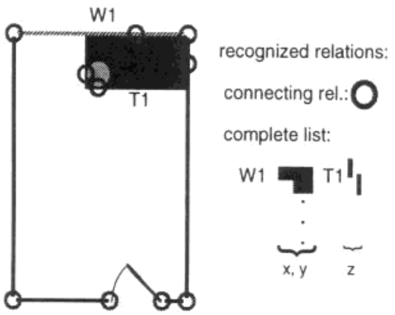

Figure 4: This figure shows the presentation of the query of figure 2 to the user.

Topology is of more than a set of binary relations. In order to retrieve similar situations the context of relations must be considered. Figure 5 gives an example. Situation 1 and situation 2 both include all kinds of the relations in the query. Nevertheless case 2 is more similar to the query, because the relations occur in the same constellation as in the query. By using the context it is also possible to determine which B<A d of case 2 correspond to the B<A d of the query.

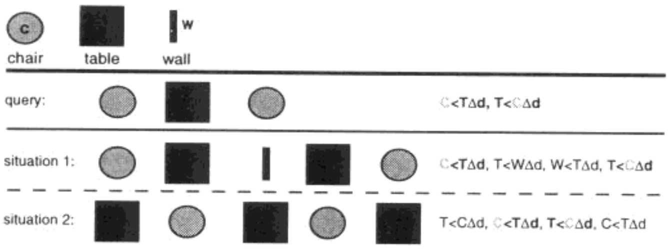

Figure 5: Considering the context, situation 2 is more similar to the query.

TOPO represents the context in a graph as suggested by [2]. Building a graph out of objects and relations, one must decide what should be the nodes and what the edges.

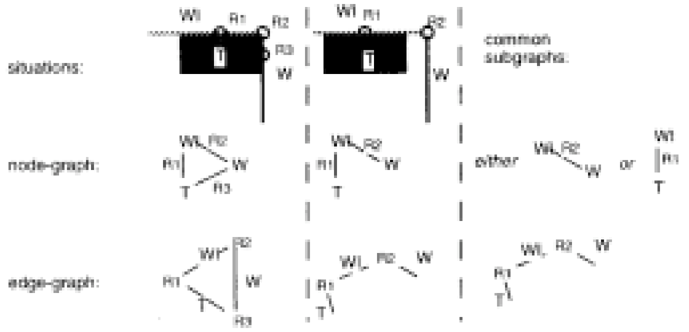

Figure 6: Comparing the node and the edge graphs of two situations leads to different results.

3.1 *Domain-specific extension*

The transfer shown so far uses no domain-specific knowledge besides the types of objects given by the layout and the straightforward definition of the basic relations. Adding knowledge about transformations of structure which do not affect the usability of a case leads to a higher quality of the result. In the domain of building layout, rotation (in steps of 90 degrees) or reflection in the horizontal dimensions of parts of a topological structure are such transformations. Using this knowledge TOPO is able to match the fragments shown in figure 7 (a) completely, although a subpart is reflected. TOPO searches for the largest common subgraph by applying the mentioned transformation to the structure of the query as a whole. It finds the matchings shown in (b). The differences of both results match disjunctive sets of nodes and therefore the two matchings can be combined to (c).

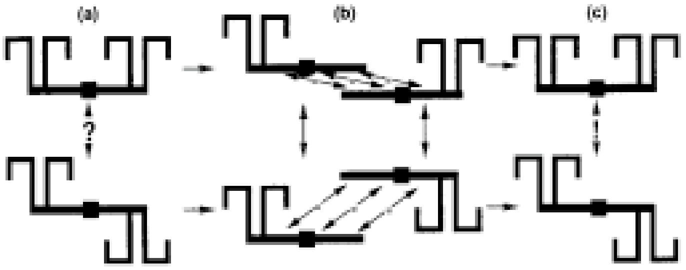

Figure 7: Using domain dependent knowledge, TOPO is able to match the fragments of (a) completely.

4 Approach

As described in the example, TOPO supports several steps during case-based reasoning. These steps of retrieval, transfer and correction are detailed in this section.

4.1 Retrieval

Given a layout of any size, TOPO matches the topology of the layout and the query. This is done by computing an edge graph representation of the layout and of the query and searching for their largest common subgraph. In order to solve a similar problem, the problem of finding a maximum clique of a graph, various algorithms have been developed [3]. A clique is a completely connected subpart of a graph (every node knows every other). Instead of searching for a common subgraph of two graphs one searches for a maximum clique in another graph that represents all possible matchings between the two graphs, called their 'combination graph". A maximum clique is the largest of all cliques of a graph.

Using the transformation described in [4] the nodes of the combination graph represent all matchings of nodes of equal types of the source graphs. Figure 8 shows an example. As mentioned before we decided to match the topological relations instead of the objects themselves, for reasons described in [5]. Therefore the nodes of the combination graph shown in figure 8 represent all matchings between the relations of the source graph of equal type. The source graphs (f) and (g) contain objects of types a and b connected by directed relations. The type of a relation is defined by the types of the source and the target object. Two nodes are connected in the combination graph if and only if the matchings represented by the nodes do not contradict each other. The matchings (R2(ab)<=>R8(ab)) and (R5(bb)<=>R1O(bb)) are connected because both relations occur in both source graphs in the same context. Both are connected by a shared object of type b. (R2(ab)<=>R8(ab)) and (R1(ba)<=>R6(ba)) are not connected because the matched relations share an object of type a in graph f but do not share any object in graph g.

The maximum clique in this combination graph and the corresponding maximum subgraphs are marked in black.

4.1.1 A common maximum clique algorithm

The algorithm of [6] (for further use called "max-clique_BK") finds all cliques in a graph by enumerating and extending all complete subgraphs. It extends complete subgraphs of size k to complete subgraphs of size k1 by adding iteratively nodes which are connected to all nodes of the complete subgraph.

4.1.2 An improvement

The idea of improvement is to search for matchings of connected subgraphs only and to combine those matchings in a second step. This strategy requires to change the transformation and the search algorithm.

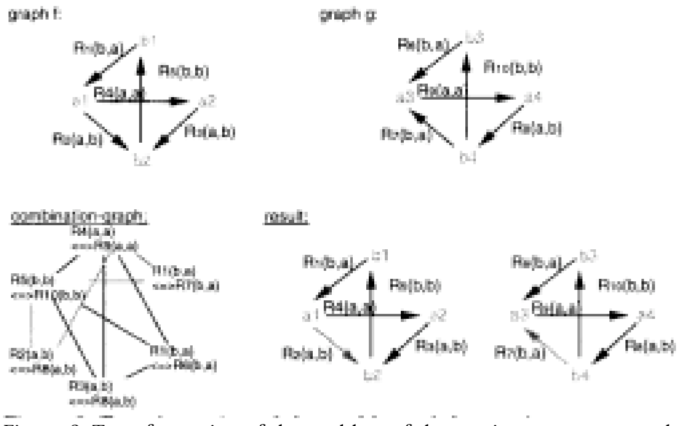

Figure 8: Transformation of the problem of the maximum common subgraph to the problem of finding a maximum clique in a graph: According to the edges of both graphs a combination graph is computed. The maximum clique and the corresponding matching is marked in black.

The transformation must generate two types of edges. One type is called 'expressive edge". A relation of this type is generated if and only if the two matchings connected by the expressive edge match relations which are connected in both source

graphs in the same way. Edges between matching of relations which are not connected in both source graphs get just the normal type 'edge'. Let us consider the combination graph of the comparison between the query and the case of figure 9. The matchings (Al<=>B2) and (B1<=>C2) are connected by an expressive edge because the matched relations Al and Bi share a chair in the same way as B2 and C2. The matchings (Al <=>B2) and (D1 <=>D2) are connected by a "normal" edge because the matched relations do not share any object in both situations. The matchings (Al<=>B2) and (B1<=>D2) are not connected by an any edge because the matched relations Al and Bi share a chair but B2 and D2 do not.

Comparing the query and the case (Figure 9) max-clique_BK takes e.g.two arbitrary chairs of the query and searches for a corresponding group in the case. If these chairs are not related to each other, all not related pairs of chairs of the case might correspond to them. In larger buildings there are thousands of such pairs. Therefore the comparison becomes very costly. Doing it by hand, one searches as far as possible for corresponding relations of connected parts only. Extending the algorithm by this behaviour we get max-clique_BK+.

The new search algorithm must distinguish between both types of edges. It extends complete subgraphs of size k to complete subgraphs of size k+l by adding iteratively nodes which are connected to all nodes of the complete subgraphs and at least one edge must be an expressive edge. Therefore all complete subgraphs found by this algorithm contain a path between any two nodes which consists of expressive edges only. For further use we call these complete subgraphs "complete expressive subgraphs. They represent matchings between connected subgraphs only. The largest common subgraph is the largest combination of connected common subgraphs consisting of disjunctive sets of nodes.

4.2 Transfer

The retrieved situation consists of objects which match part of the objects of the query. There are two kinds of information which might be transferred from the situation to the query. On the one hand the topology of the situation might be used to correct the query. On the other hand the surrounding of the situation might be transferred in order to detail or extend the query (Figure 10).

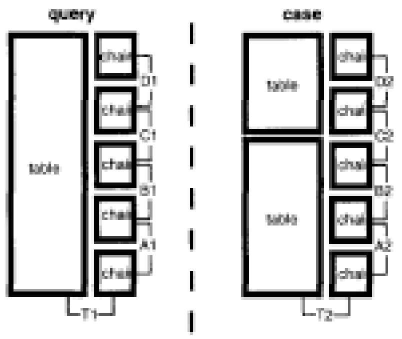

Figure 9: The new algorithm max-clique_BK+ compares connected groups only.

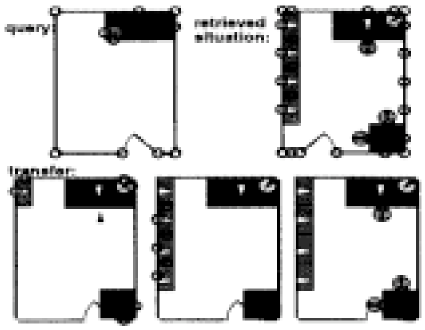

Figure 10: Topology and additional objects of the retrieved situation are transferred to the query.

The topology of the retrieved situation is used in order to correct the topology of the query (the chair is placed next to the table) and to place the unplaced objects of the query. The surrounding of the retrieved situation includes shelfs which do not occur in the query. TOPO suggests to transfer these objects to the query in order to complete the layout.

Some of the objects which might be transferred, have absolute positions in relation to the objects of the query. For example the shelf in the upper left corner. These objects are transferred first. The rest of the objects is transferred later. For example the shelfs below the upper left one have only a relative position in relation to the wall, they should be placed in the middle of the wall. These shelfs are transferred one by one, each one determining the position of the next one. The last step places all objects, which still have relative positions.

The result of the transfer is a new layout, which need not to be correct (Figure 10;lower right corner). Therefore the next step is the correction of the layout.

4.3 Correction

The spatial relations of the result are compared with the relations occurring in the layout. Unusual relations are presented to the user as possible conflicts. Conflicts which are confirmed by the user are automatically interpreted as new queries. TOPO searches for the smallest part of a former layout including the same objects, but with correct relations in order to resolve the conflict.

Figure 11 illustrates how TOPO corrects the result of the transfer. The result (Figure 10) consists of unusual relations in the lower right corner. The door overlaps two chairs and a table. If the user confirms these relations to be incorrect, TOPO initiates a new retrieval. This time it is not useful to search for a situation with the same topological relations, because these relations are incorrect. The task is to find a correct placement of those objects which are involved in incorrect relations.

Additionally the placement must not use more space than used by the wrong placement in the query. Therefore TOPO searches for a correct placement of a door, two chairs and a table within radius I. Because there is no such placement TOPO increases the radius and searches for a larger placement consisting of all objects inside radius II of the query. A situation is found and it is topology and additional objects are transferred to the query.

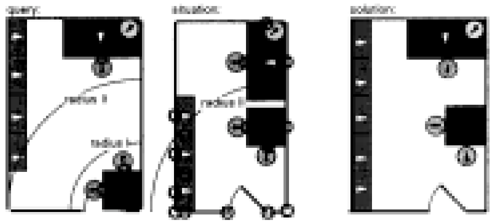

Figure 11: TOPO corrects the result of the transfer.

5 Implementation

This first example is completely artificial, but makes it easy to understand the perspective of TOPO. After implementation of the retrieval step it is now possible to work on realistic examples: Figure 12 shows a sample run of TOPO in order to support the layout of supply pipes using the layout of Figure 1.

Figure 12 shows a query (a) consisting of a trunk line connected to a hook of branch lines and an incorrectly placed hook of twig lines to be compared with the layout of part (b). (b) shows the nearly complete technical layout of two floors. Their topological representations consist of 8 (a) and 395000 (b) instances of relations. As a result of the comparison there are two fragments ((c) and (d)) of the layout (b) which match most of the topology of the query. The only difference is the placement of the hook of twig lines. The disconnected twig lines of (d) are connected to a branch line in the surrounding of (d) (compare part (f)). Both retrieved parts are presented to the user.

If the user chooses part (c) to be transferred, part (e) shows a transfer of the relations of (c) to (a) which corrects the incorrect placement of the twig lines in (a). Part (f) shows the result of a transfer of additional supply-air objects from the surrounding of (c). A transfer of all objects of the surrounding of (c) shown in (g) might be feasible and is supported by TOPO, but must be coordinated with the surroundings of the query.

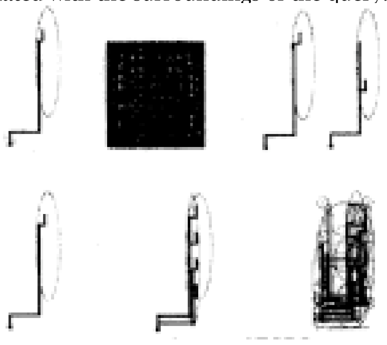
Figure 12: A sample run of TOPO

5.1 Handling exponential complexity for interactive use

Because TOPO wants to support interactive work and uses an algorithm which is practically very efficient, but has still an exponential complexity, it provides in addition several features for controlling the runtime:

- Anytime behaviour:] The algorithm can be interrupted by the user at any time and the quality of the result is always better than or equal to the best earlier result [7].

- Transparency:] In order to help the user to decide if and at which moment he might interrupt the process, TOPO visualizes three indicators during runtime:

- The progress indicator shows how much of the search space has already been searched.

- The quantity indicator shows how many cliques have been found.

- The quality indicator compares the size of the largest clique found so far to the size of the largest clique possible. The size of the largest possible clique depends on the minimum of the number of relations of the case and the query for each type of relation occurring in the query.

- Reflective knowledge: The algorithm estimates its own runtime till completion before and during running [8]. If the runtime is higher than a threshold it asks the user for a strategy in order to reduce the problem.

- Reduction strategy: If the estimated runtime is too high, TOPO suggests to sort the relations of the query by their importance given by the user. By these preferences the amount of complete sub graphs is reduced further. TOPO generates only those complete subgraphs of size k which include matchings for the first k relations of the ordered set of relations of the query. This feature enables the user to easily restrict the search space depending on his own preferences.

5.2 *State of the work on the algorithm and further improvements*

The algorithm max-clique_BK+ is implemented and tested in the domain of technical layout and molecular chemistry providing all features mentioned in section 5.1 besides the reflective knowledge. This knowledge will be acquired and added as described in [8].

For some of the comparisons the measured runtime was exponentially higher than expected. In these cases the query and the case had chains consisting of identical links. An common example is a row of outlets, rooms or larger links like {supply-air branch-line, supply-air branch-line, used-air branch-line}^n. Comparing two such chains, all possible subparts are matched. To avoid this problem, chains could be substituted by abstract elements (chains) with the type of the contained relations and their size. Instead of comparing all possible subparts a future version of our algorithm will match those abstract elements. The used knowledge will be similar to part of the knowledge used in [9]. The difference is that in our approach the content of such abstract chains is not restricted.

6 Comparison to Other Approaches

There are two main features which separate TOPO from former concepts to support the layout.

(a) TOPO uses no domain specific knowledge. It is therefore applicable to all parts of the layout domain. This feature separates TOPO from all concepts, which need some kind of model of the domain in order to give any support [10]. The only precondition is, that the layout must be represented in an object oriented way.

(b) TOPO uses arbitrary layouts for support. It needs no manual modification of former layouts. Other concepts which use former layouts need a collection of small specialized situations, cut out of complete layouts, because they cannot find the matching subpart themselves [10].

The result of TOPO is a layout which is as good as the used former layout. Due to the lack of domain knowledge no further guarantees can be given. Additional domain modeling may be integrated or appended in order to improve the overall behaviour. Integration points are the analysis of relations (which relations are important?), the retrieval step (using e.g. the semantic of floors or rooms) and the evaluation of conflicts(e.g. forbidden relations). Any other approach may be appended in order to assess and repair the result using domain knowledge formalized in constraints or models.

7 Outlook

As mentioned above the transfer and correction steps are not yet implemented. After completing the implementation TOPO will be integrated in the next prototype of the project FABEL [11]. In order to test the general applicability it will be tested in different domains of architecture and building engineering.

8 Acknowledgment

This research was supported by the German Ministry for Research and Technology (BMFT) within the joint project FABEL under contract no. 011W104. Projectpartners in FABEL are German National Research Center for Computer Science (GMD),Sankt Augustin, BSR Consulting GmbH, Muenchen, Technical University of Dresden, HTWK Leipzig, University of Freiburg, and University of Karlsruhe.

9 Bibliography

[1] Lee S.-Y. & Hsu F.-J.: Spatial reasoning and similarity retrieval of images using 2D C -string knowledge representation. Pattern Recognition, 25:305-318, 1992.

[2] Bartsch-Spoerl B. & Tammer E.: Graph-based approach to structural similarity. In A. Voss, editor: Similarity concepts and retrieval methods, pages 45-58. GMD, Sankt Augustin, 1994.

[3] Babel L. & Tinhofer G.: A branch and bound algorithm for the maximum clique problem, ZOR - Methods and Models of Operations-Research, 34:207-217, 1990.

[4] Barrow H.G. & Burstall R. M.,: Subgraph isomorphism relational structures and maximal cliques, Information Processing Letters, 4:83-84, 1976.

[5] Coulon C.-H.: Automatic Indexing, Retrieval and Reuse of Topologies in Complex Designs. In Proceeding International Conference on Computing in Civil and Building Engineering, 1995.

[6] Bron C. & Kerbosch J.: Finding all cliques in an undirected graph.
Communications of the ACM, 16:575-577,1973.

[7] Russell S. & Zilberstein S.: Composing real-time systems. In Proceedings of the 12th International Joint Conference on Artificial Intelligence, Sydney, Australia, volume 1, pages 212 - 217, San Mateo, 1991. Morgan Kaufmann.

[8] Coulon C.-H., van Harmelen F., Karbach W. & Voss A.: Controlling generate and test in any time. Proceedings of GWAI-92, volume 671 of Lecture Notes in Artificial Intelligence, pages 304-306. Springer, 1993.

[9] Boerner K.: Structural similarity as guidance in case-based design. In S. Wess, K.-D. Althoff, & M. M. Richter, (eds): Topics in Case-Based Reasoning: Selected Papers from the First European Workshop on Case-Based Reasoning (EWCBR-93), volume 837 of Lecture Notes in Artificial Intelligence, pages 197-208. Springer, 1994.

[10] Gero J. & Sudweeks F.: AI in Design, pages 75-180, 609-658, Lausanne, Switzerland, 1994.

[11] Voss et al.: Retrieval of similar layouts - about a very hybrid approach in FABEL. In J. Gero & F. Sudweeks, editors: AI in Design94, Kluwer Academic Publishers, pages 625-640, Dordrecht, 1994.