A Case-Based Mechanical Redesign System

Francois Daube*
Schlumberger Laboratory for Computer Science
8311 North RR 620
Austin, Texas 78720

Barbara Hayes-Roth
Stanford University
701 Welch Road
Stanford, California 94304

Abstract

We present a system, FIRST, that redesigns structural beams by accessing a case memory of solution plans. FIRST starts by analysing an existing design, using general knowledge about elementary physics. If design constraints are unsatisfied, FIRST searches for similar problem situations in its case memory and retrieves the solution plans associated to those situations. The system performs a transfer by analogy of each plan into the new problem situation and combines the transferred plans and symbolic analysis knowledge into a global redesign plan that is applied to the problem. FIRST is implemented in BB1, a blackboard system that allows the cooperation of problem solving knowledge from different sources.

The system, that includes general methods for transferring a plan by analogy and mapping parts of it into a new problem situation is described through the analysis and redesign of a round cantilever beam.

1 Introduction

Mechanical design is the process of going from a set of specifications to a physical artifact meeting those specifications. It is a very underconstrained process whose complexity has been pointed out in previous research [Howe *et al.*, 1987], [Mittal *et al.*, 1986]. Mechanical design rarely starts from first principles. Rather, an existing artifact is often modified until it meets the design specifications. Problems in which the structure of the redesigned artifact remains fixed throughout the modification process belong to the category of *routine* design problems.

The knowledge to modify a particular design is traditionally encoded as rules or even plans that are instantiated during the solution process. More recent developments such as PROMPT [Murthy and Addanki, 1987] or IstPRINCE [Cagan and Agogino, 1988] have focused on the ability to derive routine and non-routine modifications by performing a symbolic analysis of the behavioral equations describing the design.

Another possibility is to derive the modifications from similar cases. In a designer's memory, these cases represent the knowledge acquired on similar projects and make the difference between the expert and the novice designer. This redesign knowledge is naturally expressed by plans. Plans are attractive because they potentially embody all the knowledge required to successfully modify an artifact.

Our work investigates the potential of a case-based approach to mechanical design. To serve that purpose, we have built a system that uses a case memory of past problems and solution plans to redesign structural beams.

Beam analysis is interesting for various reasons. First, its theory is well established and amenable to some reasoning from first principles, thus providing the ground for comparison with other recent work [Murthy and Addanki, 1987], [Cagan and Agogino, 1988]. Second, it allowed us to focus on a single object or physical subsystem. Finally, the domain still bears significance in Mechanical Engineering projects.

This paper briefly describes the analysis capabilities of FIRST, presents the fundamental assumptions of our case-based reasoning approach and describes the implementation by working through the analysis and redesign of a round cantilever beam.

2 Overview of the System

FIRST solves problems defined by a set of input variables describing a beam, and a set of design constraints to be satisfied. Input variables specify a particular design and constraints specify particular relations between variables.

FIRST is given a preliminary design and its goal is to satisfy the design constraints of that design. The system does not use an objective function to assess the quality of a particular design; hence acceptance is made solely on the satisfaction of the design constraints.

FIRST starts by an analysis phase in order to evaluate the constraints (section 4). Its analysis module embodies general knowledge about elementary physics and allows it to evaluate the effect of modifications applied to the design.

When some of the constraints are violated, FIRST

*The research described in this paper has been supported by a grant of Mrs Gruner-Schlumbergcr to Etudes et Productions Schlumberger, Clamart, France, by the Center for Integrated Facilities Engineering at Stanford, NIH grant 5P41-RR-00785, DARPA contract N00039-86C-OO33

builds a redesign plan from a library of past plans and problems. The system uses a metric to identify the problems that have some similarity (section 6) and transfers their associated redesign plan into the new problem situation (section 7). The system combines each transferred plan into a global redesign plan that is applied to the design under consideration (section 9). The actions suggested by the plan that seem irrelevant to the current problem situation are eliminated by building a dependency graph from the behavioral equations describing the artifact and heuristic information within the taxonomy of domain concepts (section 7). The system stops applying the redesign plan when all design constraints are satisfied. FIRST'S overall flow of control and plan application chart are summarized in figure 1.

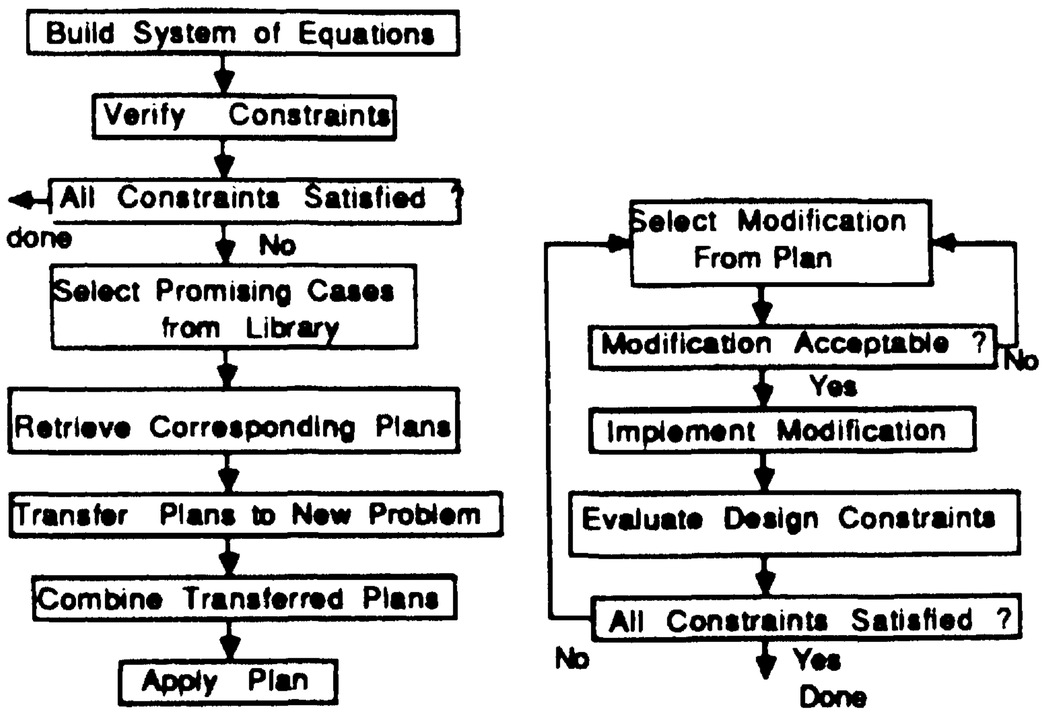

Figure 1. Flow of Control Plan Application

3 Measuring Similarity

The system organizes its knowledge of beam analysis around a taxonomy that defines each of the concepts of the domain. Concepts include the vocabulary to describe beams, loadings, materials, as well as problem types and constraint types. Domain concepts may have particular attributes, that are also defined as concepts in the taxonomy. As we shall see now, the position of a concept and of its attributes in the taxonomy provides the foundation for the case-based reasoning capability of the system.

One basic requirement of our approach is the ability to measure the degree of similarity between concepts included in the taxonomy. Our system defines the distance between two concepts A and B by the number of *is-a links* between those in the type hierarchy and the closest analog of a concept by the one having the smallest distance to it. The instance most similar to the instance diaml in figure 2 would then be either ydiml or zditnl, both located at a distance of 4 links from diaml.

When the information coming from the position of concepts in the taxonomic hierarchy is not adequate or sufficient, our system can refine its approach by measuring the similarity between their attributes, also defined as concepts. We define the distance between two lists of domain concepts by the sum of the minimum distance between individual members of the lists, minimally distant members being removed from the lists at each step. This distance is used to assess the similarity between lists of attributes.

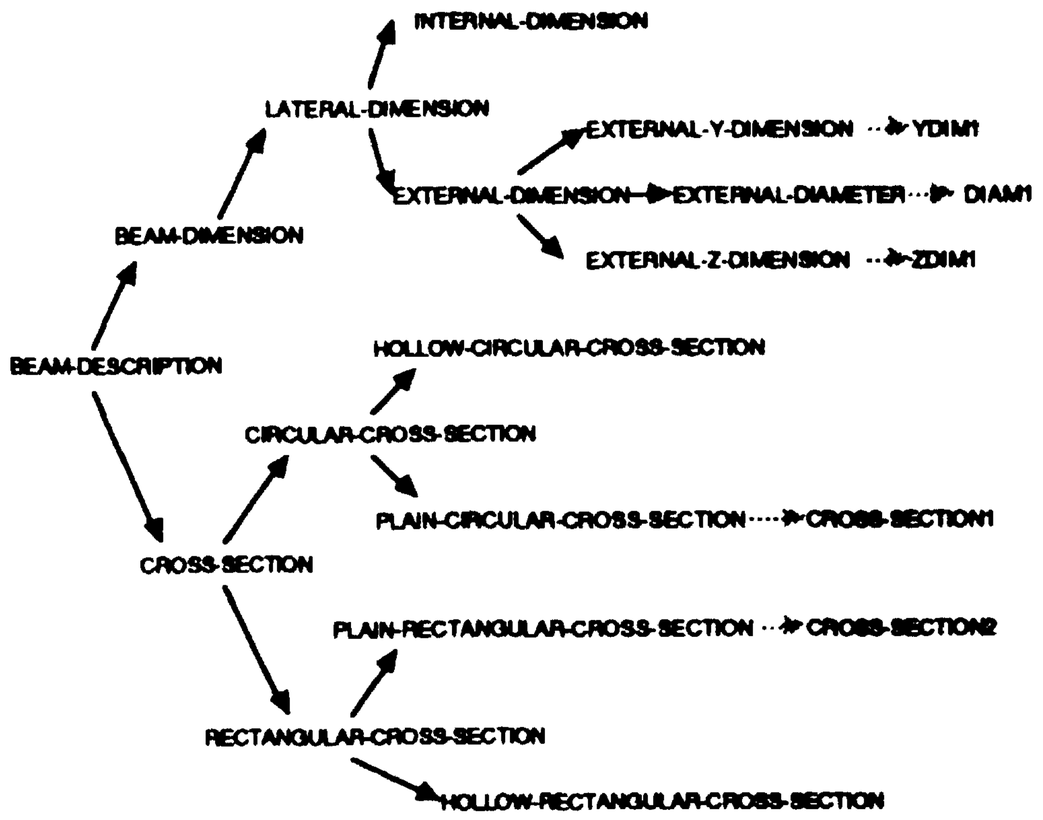

Figure 2. Simplified taxonomic hierarchy.

4 Analysis

The system's initial goal is to check the design constraints. Their validity may depend on variables computed by the equations describing the structure's behavior. The analysis module applies its knowledge about physical principles to derive this set of equations. It is also used to propagate the effect of design modifications to the system of behavioral equations, and instantiate particular domain specific constraints about beam design. The implementation of the analysis module is beyond the scope of this paper which focuses on the case-based reasoning capabilities of the system, but further details can be found in [Daube and Hayes-Roth, 1988],

In our example, FIRST is given the description of the round cantilever beam, presented in figure 3, in terms of its design variables (in typewriter fonts).

Structure: Round-Beam
Cross-section: Cross-sectionl
Material: Materiall
Applied Moment: M
Diameter: Diami
Applied Force: F
Length: L

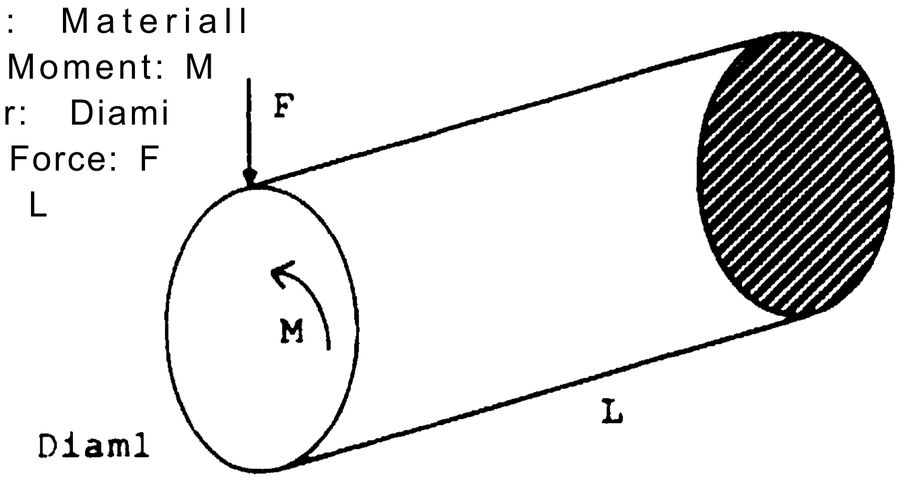

Figure 3. A Cantilever Beam of Circular Cross-Section

FIRST is also told to satisfy the following constraints:

- Weight: W < *w1*
- Diameter: $Diam_1$ < *D*
- Max Stress: MSTRESS < Yield-Strength(Materiali)
- Environment : H_2S *(Corrosive)*

At the end of the analysis phase, FIRST has generated a behavioral model, whose principal equations are :

$$W = \pi(Diam_1)^2 L\rho/4 \qquad I = \pi(Diam_1)^4 \,/\, 64$$
$$I_\rho = \pi(Diam_1)^4 \,/\, 32 \qquad \sigma_{bend} = \mathrm{F\ L}\ Diam_1 \,/\, 2\ \mathrm{I}$$
$$\tau_{tors} = \mathrm{M}Diam_1/2I_\rho \qquad \sigma_{vm} = vonmises(\sigma_{bend}, \tau_{tors})$$

I is the inertia of the cross-section
I_ρ is the polar inertia of the cross-section
σ_{vm} is the equivalent (von-mises) stress within the structure.
After the analysis has been completed, the system checks for the satisfaction of the constraints. In our example, the maximum equivalent stress in the structure is found to exceed the yield strength of the material. In addition, the material, steel, is found unacceptable for the corrosive environment. FIRST then decides to access its memory of past problems in order to modify the structure and eventually satisfy its design constraints.

5 Redesign Options

5.1 Symbolic Analysis

When violated constraints involve simple equations, symbolic analysis provides an efficient way of deriving changes to the structure. We have implemented such a symbolic analysis module to propose qualitative modifications in simple cases of non-conflicting constraints. Too often though, conflicting constraints, evinced by contradicting suggestions from the analysis module prevent such an approach. Our system then relies on its case memory to derive relevant modification steps.

5.2 Redesign Plans - Redesign Language

Case-based reasoning applies earlier experience of similar situations to help solve new problems. Our system focuses on the use of past plans rather than past solutions and in that aspect exemplifies Carbonell's *derivational analogy* [Carbonell, 1983]. A solution plan consists of an ordered set of actions that represent the various steps that lead to a successful solution in a particular case.
Several stages can be distinguished in the generation of a redesign plan using case-based reasoning:
An *access* phase which identifies the cases in memory that present useful similarities with the current problem.
A *transfer* phase where the plans corresponding to the cases selected in the access phase are transferred by analogy in the current problem situation.
A *mapping* stage in which similarities and differences between each case selected from memory and the current problem situation are identified. We shall see in the next sections that this step is performed as the plan is being applied to the structure.

Our decision to work on plans prompted the need to express solution plans in a high level, declarative and simple fashion. To that purpose, we developed a *language framework* that incorporates a type hierarchy of design actions and the translation of the high level actions into low level LISP statements. Using this language, modification steps are concisely expressed by an action verb followed by relevant design variables or domain concepts. Figure 4 provides a view of FIRST'S action hierarchy. For example, the action sentence increase diaml, which tells the system to increase monotonically the variable diaml would result upon application into LISP statements that implement those modifications. Complete plans are then defined by an ordered set of action sentences.

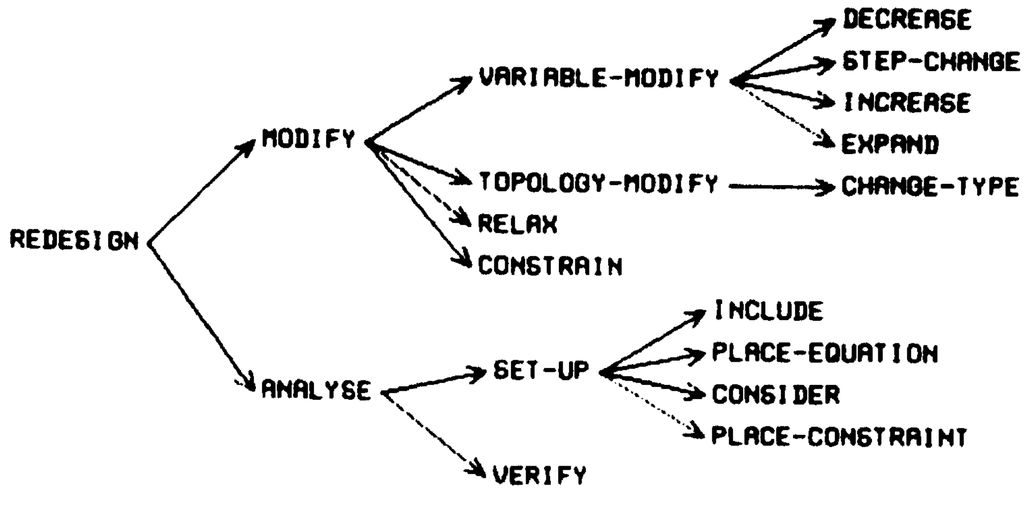

Figure 4. FIRST's action hierachy.

6 Case Access

FIRST has a case memory of solution plans from which particular plans can be retrieved and instantiated into a new problem. The case memory is indexed by the physical nature of the problem and the associated type of constraints and constraint status (satisfied/violated). The system's metric selects analogous cases in memory based on the similarity between the violated and satisfied constraints of the current problem and the violated and satisfied constraints associated to the cases in memory. The result of that phase is a list of useful cases along with the violated constraints of the current problem they address. Note that this is no more than a *goal indexing* method since the system's goal is to satisfy all its violated constraints.

In our example, FIRST'S metric does not find any case addressing both violated constraints, but identifies a rectangular simply-supported beam that includes a maximum stress constraint and a pinned round beam in torsion to address a similar environment constraint. Those structures are presented in figure 5.

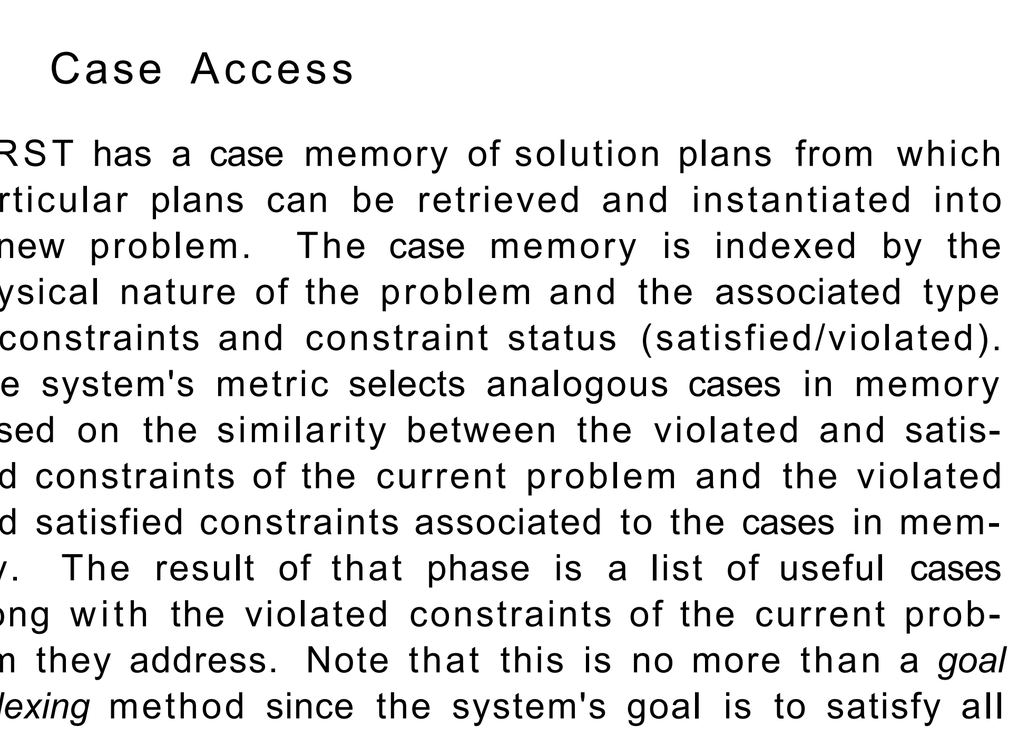

Figure 5. Cases Selected

7 Case Transfer and Mapping

After potentially useful cases (source cases) have been identified in memory, their corresponding solution plans need to be adequately transferred into the new problem domain, while actions irrelevant to the new problem should be eliminated.

An additional problem arises from the fact that if the transfer takes place *before* the start of the redesign process, some design variables referred to by a plan in memory might not have any equivalent yet in the new problem. We shall see that the system addresses this issue by delaying as much as possible the transfer of design variables from one problem to another.

7.1 Plan Transfer

The transfer of a plan occurs by parsing each of its language sentences and transferring each design variable of the problem in memory into its closest analog in the new problem. Section 3 presents the method for transferring variables.

This approach, which does not modify problem-independent concepts, does not suffice for action sentences such as change-type instance-A to type-B, which relate an instance to a class. For such action sentences, the system identifies the transformation leading from concept A to B thru the links of the type hierarchy and characterizes each individual transformation along the links by the list of attributes created or deleted. The closest analog of the action sentence in the new problem will then transform the closest analog of instance-A, instance-A' into type-B' such that the distance between lists of created and deleted attributes at each step in both problems is minimal (section 3).
This way, the sentence : change-type cross-section2 to hollow-rectangular-cross-section
gets translated into : change-type cross-sectionl to hollow-circular-cross-section by identifying the transformation that lead from cross-section2 to hollow-rectangular-cross-section in the case problem and following the most similar one in the type hierarchy of the new problem, starting from cross-sectionl. The concepts referred above are included in the type hierarchy of figure 2.

7.2 Delayed Transfer and Action Elimination

Accessing the case memory takes place at the beginning of the solution process; at that time some of the concepts in a past plan, which might have been created during the solution process may not have proper equivalents in the new problem. We have designed the transfer mechanism so that the transfer of a concept is performed only when the action that refers to it is considered for execution. This late commitment strategy maximizes the number of analogous candidates when a variable is being transfered from a past case into a new problem situation.

Not everything is useful in a past plan; some of the actions suggested by a plan in memory might have nothing to do with the goals of the new problem. FIRST has a mechanism to assess the relevance of each action to the current goals. Recall that case solutions were retrieved from memory with the constraints they addressed. FIRST evaluates the influence between each of the design variables referred to in its action sentences and the violated constraints addressed by the plan. The system scans the variables involved in the violated constraints and builds for each of those a *dependency graph* based on the behavioral equations (derived in the analysis phase). If any of the variables referred to in the action sentence appears in any of the dependency graphs, the action is assumed to be relevant to the case. Thus, the dependency graph is used to decide whether some variable might have some influence on a violated constraint. If no relation can be found between the variables referred to in the action sentence and the violated constraints addressed by the plan, the action is discarded. However, this approach is not sufficient for those design variables that do not appear in any equations, such a cross-section for example. The system then uses particular links in its taxonomy of domain concepts to assess the influence of a particular variable on a set of others. In our domain, the concept cross-section has a named link 'affects' to the concept cross-section-properties, expressing the fact that any change in cross-section will affect the cross-sectional properties. Assessing the relevance of actions is the *mapping* stage of our system, since it establishes the amount of similarity between a case in memory and the current problem situation.

8 Combining Plans: The Blackboard Model

Our implementation needed a way to take into account the cases selected from memory and knowledge coming from its symbolic analysis module.

FIRST is implemented in BB1, a blackboard system that has been applied to the domain of protein structure derivation [Hayes-Roth *et al.*, 1986] and the layout of civil engineering sites [Tommelein *et al.*, 1987]. BB1 supports knowledge-based problem solving by means of a global database (the blackboard) that records the evolving solution and of independent *knowledge sources* that contribute to the solution. Knowledge sources become executable during the solution process, posting action proposals on the system's *agenda.* The system selects among the actions in its agenda according to its current control strategy. A control strategy can roughly be described as a set of functions that rate the actions posted on the agenda. A description of the control capabilities of BB1 is provided in [Hayes-Roth, 1986]. In our blackboard model, a plan is simply expressed by a set of knowledge sources, each of which implements a particular action.

The blackboard model lends itself to the kind *cooperative problem solving* needed to take into account knowledge from symbolic analysis and selected cases in memory. The symbolic module, implemented in a knowledge source posts particular action proposals on the agenda; knowledge sources from the selected cases also post action proposals. The system then selects among those according to its global control strategy, which we irn-

plemented as the summation of the ratings provided by each of the control strategies of the selected cases.

0 Plan Application Example

After the behavioral equations have been built, violated constraints identified and analogous cases selected, we illustrate the redesign of our cantilever beam in figure 6.

The first actions on the agenda are the initial actions from the round rod and rectangular beam plans (rrI, rb1), transferred into the current problem situation and one action coming from symbolic analysis. The action suggested by the round rod plan is considered irrelevant and discarded, by application of the dependency graph technique described in the previous section. Namely, the dependency graphs do not show any relation between the design variable diaml-problem2 and the violated constraint environment that the plan retrieved from the round rod case is addressing (figure 6).

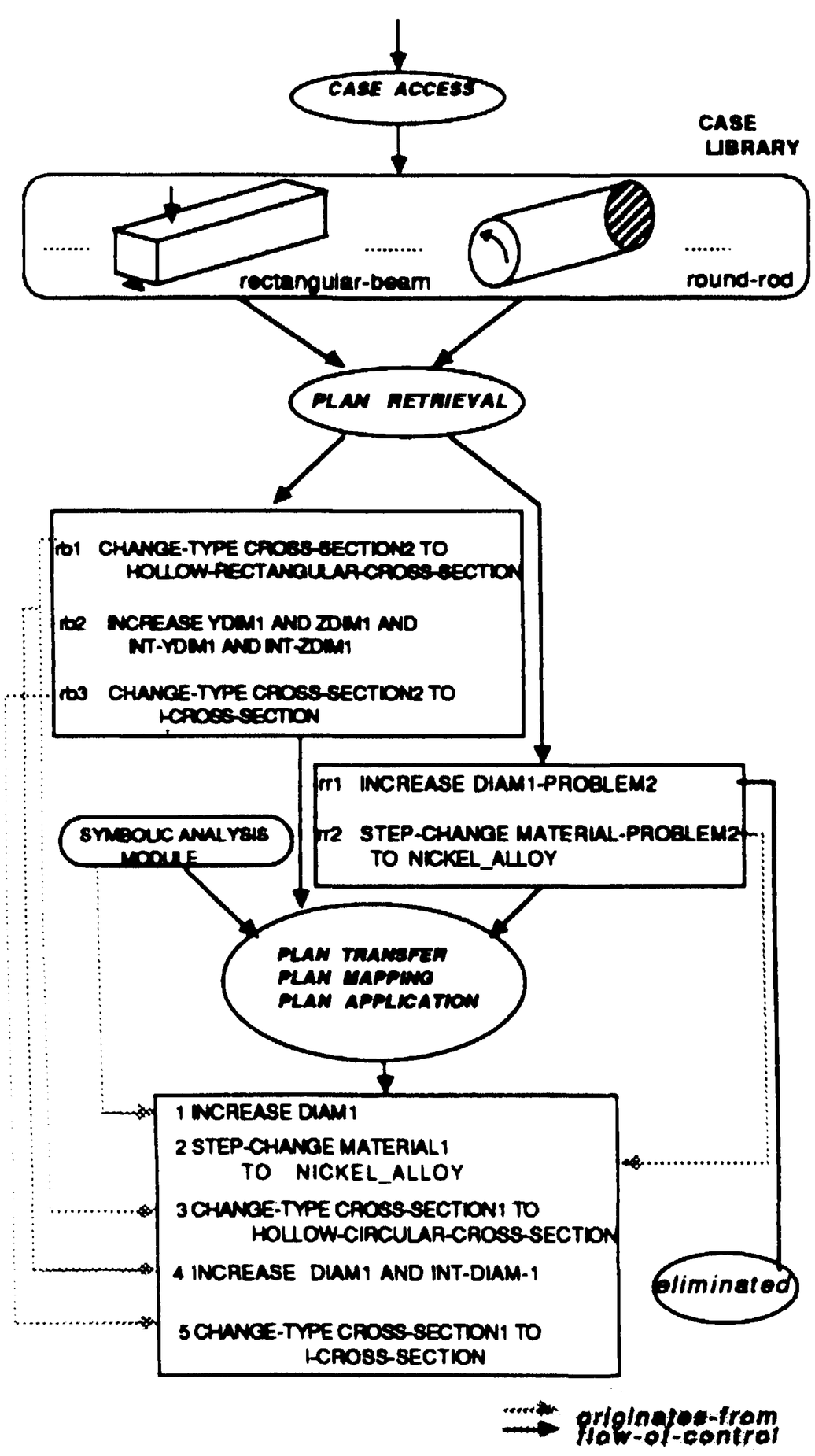

Figure 6: Redesign Summary

The action suggested by the symbolic analysis module is rated higher by the control strategy inherited from both plans because it implements a simple routine modification. It is then selected for execution and the diameter of the beam is increased, until the maximum weight of the beam is reached. Next, the second action transferred from the round rod plan (rr2) is selected (it is also a routine modification) and the system changes the material of the beam from steel to nickel-alloy, satisfying the environment constraint. At this point, there are only conflicting constraints left (maximum stress, maximum weight) and no routine changes can be suggested by the plans or the analysis module. The system then selects a non-routine modification step coming from the rectangular beam plan (rbI) and transforms the circular beam into a hollow circular one, also adding from its domain knowledge of hollow beams a fabrication constraint that limits its external diameter. As a result of this modification in the structure of the beam, the analysis module updates the behavioral equations of the structure (modified inertia, section, weight). The next action selected, derived from the rectangular beam plan (rb2) then suggests to increase both the internal and external diameter of the beam. This step is performed until the fabrication constraint gets violated. Again, there are no routine changes to choose from; the system selects the action coming from the rectangular beam plan (rb3) that calls for converting the circular beam into an I-beam, for which the behavioral model gets appropriately modified (analysis module). At that point, there are no more actions in the plan and the solution process is stopped. Figure 7 describes the modifications of the design as the plan is applied.

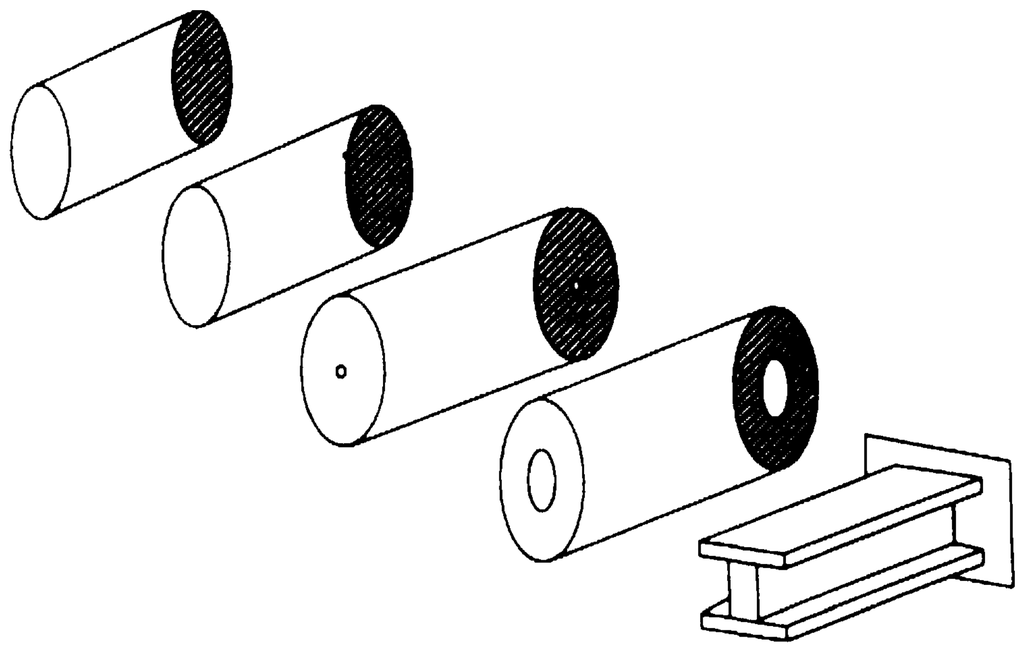

Figure 7. Evolution of the Round Cantilever Beam

10 Status

The analysis module currently includes knowledge about beams of circular cross-section in bending, torsion and compression, rectangular and I-beams in bending and compression. Boundary conditions implemented include cantilever and simply supported at both ends. The case library itself contains a limited number of cases, so that new cases do not have easy matches in memory. Five cases, spanning over the domain of interest are currently included in the library.

11 Limitations

Much of our efforts went into devising transfer and mapping mechanisms for our case-based implementation. Selecting promising problems from the case memory (access phase) is also an important part of a case-based reasoning system. Multiple indexing can provide different viewpoints on the set of problems in memory, especially when dealing with large case memories. We have not explored this field and our metric for selecting problems remains relatively simple.

The algorithm that transfers concepts from one problem situation to another is strictly based on the distance measured on the type hierarchy. This works well when a detailed type hierarchy is available for each problem situation, but otherwise leads to several solutions when looking for the closest analog of a particular concept. In that case, supplementing the distance information with equivalent information coming from the attributes of the concept (that are also concepts) is a promising solution path.

When it comes to deriving modifications from the behavioral equations and applying them to a design, our mechanisms are much simpler than the ones found in [Howe *et a*l., 1987],[Murthy and Addanki, 1987], [Cagan and Agogino, 1988]. Those systems perform a quantitative evaluation of modifications prior to applying them to a design, whereas we proceed by qualitative analysis, followed by iterative changes of design variables.

Perhaps more seriously, plans expressed as an ordered set of actions fail to capture the rationale behind each of those actions, forcing us to rely on weak methods such as the dependency graphs to select relevant actions within a plan.

12 Related Research

Current mechanical design systems do not attempt to utilize past experience to derive their actions. There has been interesting applications of case-based reasoning in other areas though, two of which are described below.

Chef [Hammond, 1986] is a case-based system that works in the domain of Szechwan cooking. As our system, Chef uses a memory of past recipes to build a new one and has a simulation module to assess the quality of new recipes. In addition, Chef has a mechanism to archive and index new plans in the case memory, giving it a learning capability.

Cyclops [Navinchandra, 1988] works in the domain of landscape design. Instead of working on past plans, Cyclops works directly on a memory of past design solutions to extract useful features for its current design problem. We have not addressed this issue in our research.

13 Conclusion

More experiments are needed to see if the ideas presented in this paper can be generalized to the redesign of assemblies, instead of a single artifact and to more complex domains. We believe though that our approach goes further than existing routine design systems [Howe *et al,* 1987] [Brown *et al,* 1986], by giving them the ability to perform non-routine modifications.

Building a new plan from a memory of past ones, our system is not likely to produce innovative designs, but it is more efficient than reasoning from first principles and also takes into account heuristic knowledge that cannot be derived from these first principles.

Although applied to the simple domain of beam design, the architecture we propose is quite general. Given a body of domain knowledge provided by a type hierarchy and a language framework that allows a user to express plans in a concise manner, we present general mechanisms for selecting useful cases, transferring corresponding plans and mapping relevant parts of those into a global redesign plan.

References

[Howe *et al,* 1987] Howe A. et al. Dominic: A Domain Independent Program for Mechanical Engineering Design *International Journal for Al in Engineering,]).* 23-28, January 1987.

[Mittal *et al,* 1986] Mittal S., Dym, C.L., Morjaria, M. PRIDE : An Expert System for the Design of Paper Handling systems. *Computer,* p. 103-114 July 1986.

[Brown *et a*l., 1986] Brown, D.C., Chandrasekaran, B. Knowledge and Control for a Mechanical Design Expert System. *Computer,* p. 93-100, July 1986.

[Mittal *et al.* , 1986] Mittal, S., Araya, A. A Knowledge Based Framework for Design, *Proceedings, AAAI 1986 Conference,* p. 856-865, 1986.

[Murthy and Addanki, 1987] Murthy, S. Addanki, S. PROMPT: An Innovative Design Tool *Proceedings, AAAI 1981 Conference,* p. 637-642, 1987.

[Cagan and Agogino, 1988] Cagan, J., Agogino, A. *Innovative Design of Mechanical Structures from First Principles.* Working Paper, Dept of Mechanical Engineering, U.C. Berkeley, 1988.

[Daube and Hayes-Roth, 1988] Daube, F., Hayes-Roth, B. FIRST, A Case-Based Reasoning System in the BB1 Blackboard Architecture *Proceedings, AAAI Workshop on Case-Based Reasoning* 1988.

[Hayes-Roth, 1986] *BB1 User's Manual* KSL Report 86-61, Stanford University, 1986.

[Hayes-Roth *et al,* 1986] Hayes-Roth, B. et al *PROTEAN: Deriving protein structure from constraints.* Technical Report KSL 86-51, Stanford University, 1986.

[Tommelein *et al.,* 1987] Tommelein, I.D. et al. SIGHT-PLAN, A Blackboard Expert System for Construction Site Layout. *Expert Systems in CAD Conference,* Sydney, 1987.

[Carbonell, 1983] Carbonell, J.G. Derivational Analogy and its Role in Problem Solving. *AAAI 1983 Proceedings* p. 64-69, 1983

[Hammond, 1986] Hammond, K. CHEF, A Model of Case-Based Reasoning. *AAAI 1986 Proceedings*

[Navinchandra, 1988] Navinchandra, D. Case Based Reasoning in CYCLOPS, a Design Problem Solver (working document).