Index Transformation Techniques for Facilitating Creative use of Multiple Cases

Katia P. Sycara and D. Navinchandra
katia@cs.cmu.edu dchandra@csxmu.edu
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Using cases to find innovative solutions to problems is mainly the result of two processes: (1) cross-contextual rem in dings, and (2) composition of multiple cases or case parts. Although the ability to use cases taken from across contextual boundaries is desirable, there is a tension between representing and accessing cases across contexts and in using parts of multiple cases to synthesize a solution. One way of alleviating this difficulty is through index transformation. In this paper, we represent two index transformation techniques that facilitate both cross-contextual remindings and the access of multiple appropriate case parts. The mechanisms are general and principled (based on a qualitative calculus). They are also behavior-preserving, a needed requirement for case synthesis in many domains of interest. The transformation techniques have been implemented in CADET, a case-based problem solver mat operates in the domain of mechanical design.

1. Introduction

Case based reasoning [Kolodner.Simpson.Sycara 85] is the method of using previous cases to guide solving of new problems. Given a new problem solving situation, appropriate previous cases are retrieved from memory, the best is selected, and differences as well as similarities between the previous and current case are identified. These similarities and differences are used to adapt the retrieved case to fit current circumstances. At the end of problem solving, the solved case is stored as a new case to be used in the future.

Using cases to find innovative solutions to problems is mainly the result of two processes: (1) cross-contextual remindings [Gordon 61], and (2) composition of multiple cases or case parts [Koestler 64], Each one of these processes imposes requirements on case representation, memory organization, and indexing. The work presented in this paper focuses on the issue of indexing, in terms both of a domain-independent indexing vocabulary and index transformation techniques to facilitate the access and composition of cases and case pieces. We illustrate our claims in the domain of mechanical design. The abstract index representation and transformation techniques presented in this paper have been implemented in CADET, a case-based design problem solver [Sycara & Navinchandra 89, Navinchandra et al. 91].

A memory that allows retrieval based on similarity is an important component of a case-based system fSchank 82]. The natural way to implement similarity recognition and retrieval based on similarity, is to index memory such that similar cases share common indices and can consequently be retrieved together. But what kind of similarity should this process use? Although surface features have been used successfully to retrieve cases in some CBR systems (e.g., [Simoudis & Miller 90]), it is clearly more useful to index cases in terms of abstract features so that the solution indexed with a particular case is applicable to other cases that share those abstract features, and may share few, if any, surface features. If a case is to have broad applicability, the indexing vocabulary must be at an appropriate level of generality and must reflect some thematic abstraction [Owens 88]. We advocate the use of qualitauve influences as an appropriate vocabulary for expressing thcmatic abstractions.

Qualitauve influences express abstract causal interactions among problem variables. An influence is a qualitative differential (partial or total) *relation* between two variables one of which is a dependent variable and the other an independent variable. We use graphs of these influences to represent the behavior of physical artifacts. Behavioral descriptions are good thematic abstractions that result in interesting and useful cross-contextuaJ remindings. For example, an electronic sensing device could be used to sense water *level* in the design *of a* flush tank. This is a cross-contextual reminding (electronic device used in a hydraulic-domain) that results in an innovative solution.

Retrieving cases from different domains make it difficult

This research is being supported by DARPA and AFOSR under contract number: F49620-90-C-0003.

to access case parts that could be used in the composition. Useful parts may be buried in the case where subpart boundaries may be difficult to identify- One approach for addressing the difficulty of indexing and accessing appropriate case parts is to use index transformation. Our earlier work on the PERSUADER [Sycara 87] and CYCLOPS [Navinchandra 87,Navmchandra 91] used pieces from multiple previous cases to come up with a proposed new solution. Redmond's [Redmond 90] work on diagnosis uses pieces from multiple cases, where conventional subgoaling is used to indicate the appropriate snippets. The snippets arc indexed under their corresponding subgoals.

By contrast, in engineering design, a subgoal decomposition that would facilitate snippet indexing and retrieval cannot be identified a priori. This is a consequence of the fact that there is no one to one correspondence between device components and desired device subfunctions. In addition, since design goals change during problem solving, the cases in memory may not always be compatible with the current goal. Therefore, *it* becomes necessary to transform the goal during problem solving. This is done in CADET through index transformation techniques that arc behavior preserving. In the design domain, a stringent constraint on the retrieved case parts is that their *combined behavior must be equivalent to the original behavioral specification of the desired artifact.* In this paper, we present two index transformation techniques that allow such behavior-preserving retrieval of case parts.

2. CBR in Design

Design is the act of devising an artifact that satisfies a useful need, in other words, performs some function. It is an interesting domain for conducting research in Case Based Problem Solving because: *designs are often synthesized by combining several parts of different design cases taken from different contexts, where each part delivers a portion of the overall behavior of the final artifact. A* design, hence, can contain parts taken from design cases that are, on the surface, functionally dissimilar to the current design problem.

For example, consider the design of a device which controls the flow of water into a flush tank. The behavior can be specified as follows: *as the depth of water (D) in the tank increases, the rate of flow of water into the tank (Q) should decrease.* This specification may be used as a set of indices to find relevant cases in memory. If there are no cases that directly match the specifications, then it would be useful to consider using pans of several cases. In this instance, an analogically relevant case is a hot-cold water faucet shown in Figure 2-1. The faucet is specified as a device that allows for the independent control of the temperature and flow rate of water by appropriately mixing the hot and cold water streams. By extracting portions of the faucet, such as the see-saw part, it is possible to design the flush lank device as shown in in Figure 2-2.

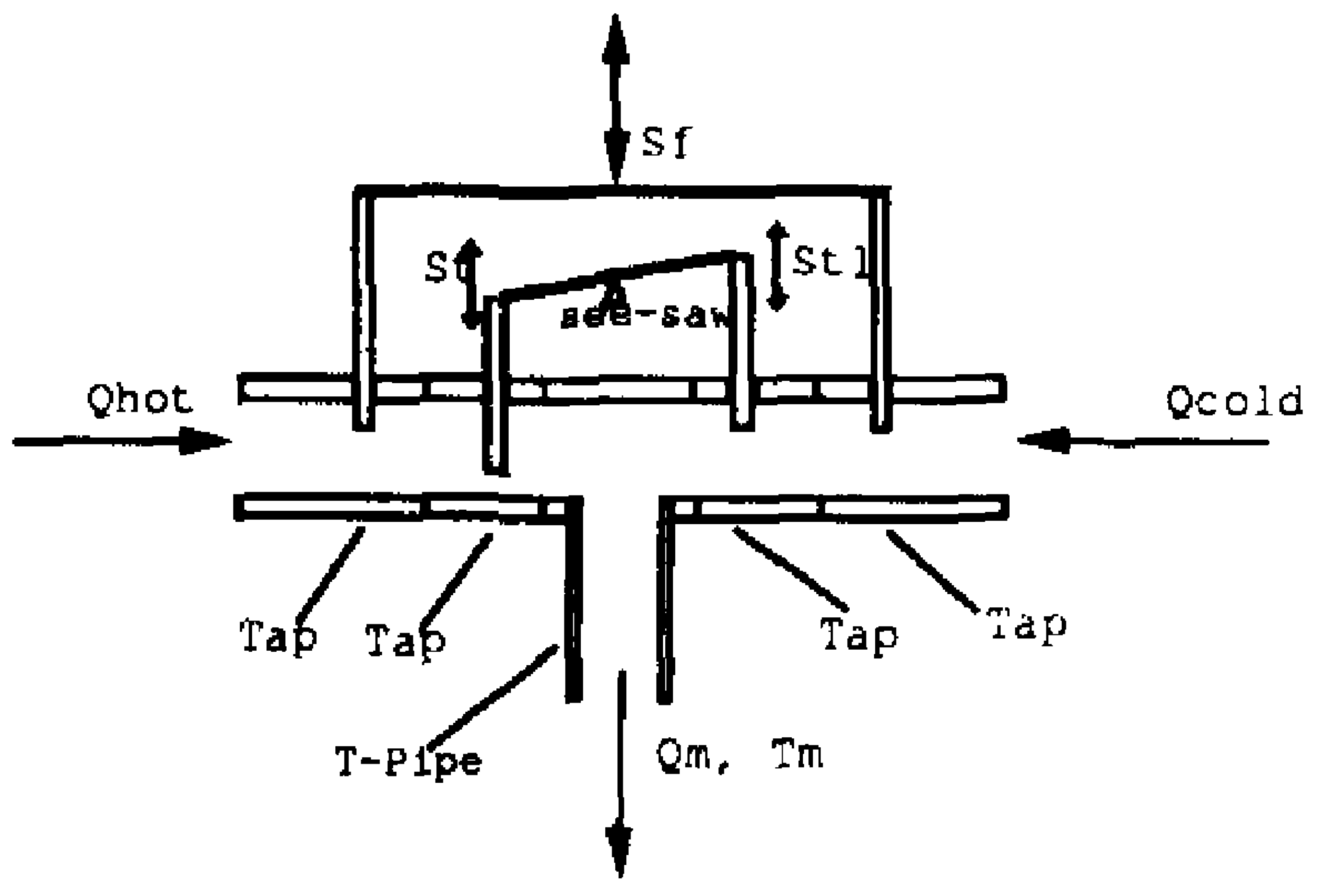


Figure 2-1: A Simple Hot-Cold Water Faucet

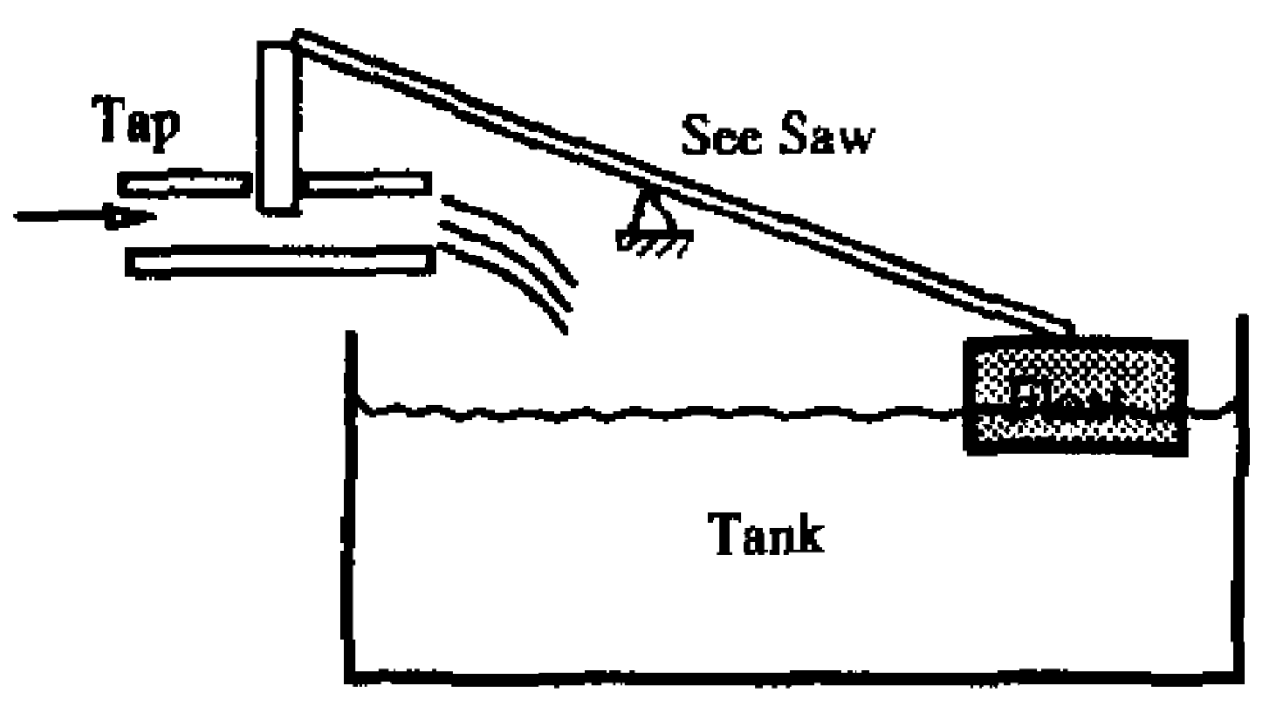


Figure 2-2: A flush tank

The relevance of the faucet case to the design of the flush tank would not have been possible to recognize because: (a) the functional descriptions of the two devices are completely different, and (b) the whole faucet is not relevant to the target problem. The relevance recognition process has two parts: similarity recognition and sub-behavior matching. First, the goal specification is elaborated by applying transformation operators. This process, in essence, generates several alternative behavior descriptions that are equivalent to the original goal. These alternatives are then matched against the cases in memory. The matching process tries to find entire cases or parts of cases that share common "sub-behaviors"[1] with the elaborated goal.

We present an approach to accomplish recognition of shared "sub-behaviors" based on behavior preserving transformations that uses qualitative reasoning methods. The transformation is done in two ways: (a) If it is known what physical laws and principles are going to govern the solution, then the given goal is transformed by relying on the laws to achieve certain sub-behaviors, (b) If, however, the relevant laws are not known *a priori,* sub-behaviors are hypothesized and *the* case memory *is* searched to find ways in which the required behavior may be achieved. This is in contrast to other approaches that assume *a priori* knowledge of the domain laws and models that will be part of the solution [Williams 90], If CADET cannot complete a design because it is not given the relevant physical laws, it hypothesizes new behaviors and looks for cases which embody those behaviors. The approach has the following advantages: (1) the system at each point in the search, is aware of what behavior it is trying to achieve, (2) because cases embody design optimizations, the accessed components correspond to already optimized physical structures, (3) solutions may involve the use of principles outside the current domain, that have been successful in a prior design, (4) the problem solver does not have to re-solve every problem from scratch.

3. Behavior Representation in the Cases

Device behavior is represented as a collection of influences organized in the form of a graph. The notion of influence graphs is a very general one. It applies to any domain in which behavior can be characterized as a set of quantities that relate to one-another. Given such an influence graph, it is possible to predict possible outcomes of given perturbations [Sycara 87].

Consider, for example, a gated water tap that has two inputs: a water source and a signal to regulate the rate of flow of water (Figure 3-1). The flow rate is given by Q and the position of the gate is given by X. The position of the gate controls the flow rate. This behavior is represented as an influence $Q \xleftarrow{+} X$, which is read as follows: "The flow rate (Q) increases (+) monotonically with an increase in the signal (X)". This influence represents the "tap" principle.

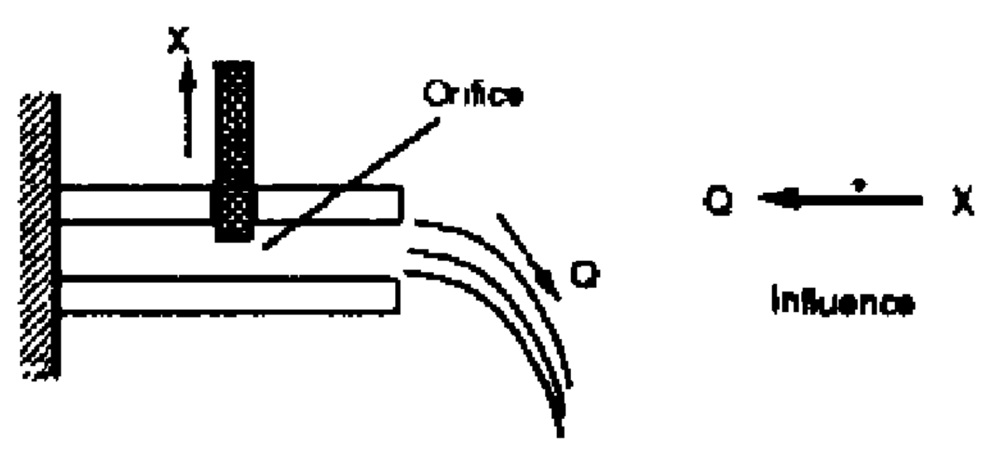


Figure 3-1: A gated-tap

Sets of qualitative influences can be combined to capture the behavior of more complex devices. The see-saw shown in Figure 3-2 has three major behavioral parameters: ft, the angular position of the see-saw and the positions of the two ends of the see-saw *(X1* and *X2),* The main influences are *X2*----------a , *X2* <—*a* and *X2* <—*X1*. These influences form a directed graph.

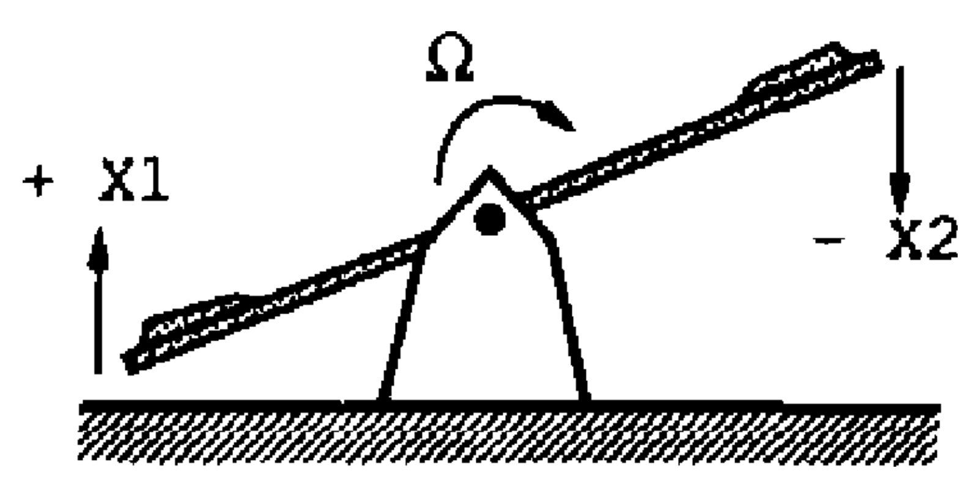


Figure 3-2: A see-saw

3.1, Influence Graphs and Components

The various components of a design work together to deliver its composite behavior. The influence graph that describes the overall behavior has sub-graphs that correspond to individual components. This provides a mapping between behavior and the structure of the case.

Let's re-consider the hot-cold water faucet (Figure 2-1). The behavior of this device can be represented as shown below (Figure 3-3). Two input signals *St* and *Sf* control the mix temperature and the mix flow-rate. When *St* is increased, then the total quantity of hot water *Qh* increases. At the same time, due to the see-saw principle, the signal *St* decreases, causing the total quantity of cold water *Qc* to proportionately decrease.

Note also, that single and multi-influence subgraphs of the faucet's influence graph corresponds to the various components of the faucet. This information serves as handles for snippet (component) identification and extraction.

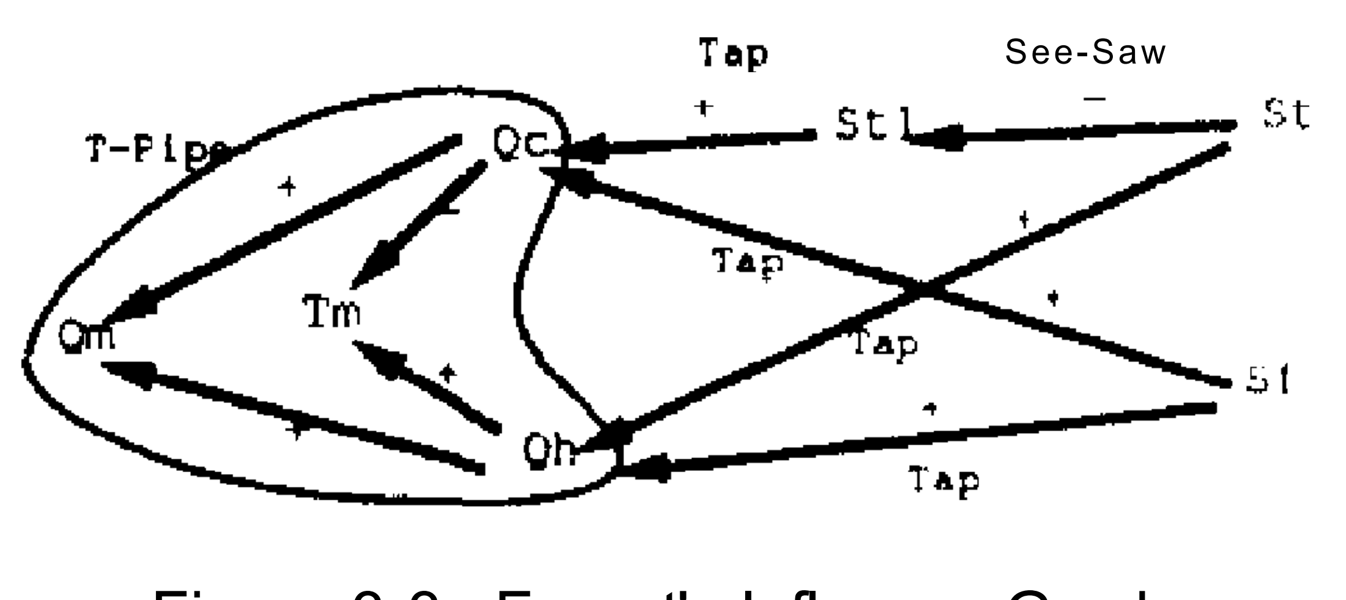


Figure 3-3: Faucet's Influence Graph

When this behavior representation is incorporated in a device case base, it becomes possible to retrieve cases which match given behavior specifications. If retrieval using the design specification fails to retrieve relevant cases, the system should be able to recognize how a combination of component behaviors could produce the required effect. This is done by transforming the indices. In the next section we will cover case indexing and retrieval, followed by a section on index transformation.

4. Case Retrieval

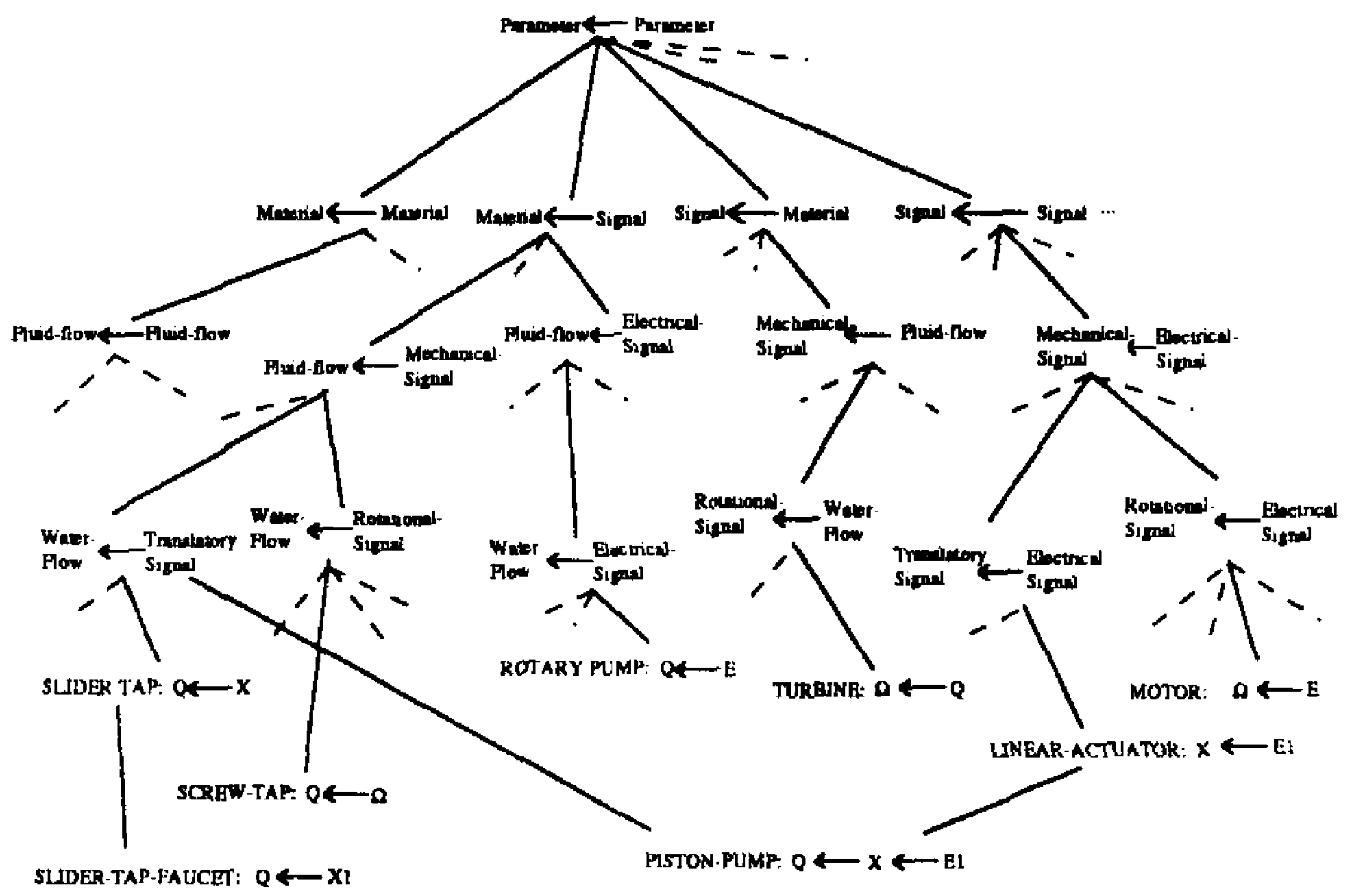
Most CBR systems retrieve cases using indices that match specific attributes about cases. In engineering design, the case matching problem includes matching graphs of influences. Behavior matching for case retrieval is carried out in several ways:

4.1.Abstract Matching

The nodes in the influences are matched using object and concept hierarchies. For example, the simple tap is described by the influence $Q \xleftarrow{+} X$. The node *Q* which is a flow of water is abstracted to be a liquid-flow, a material flow, and finally a design-parameter. The quantity *X* is a translatory-signal, a physical-signal, a signal, and finally a design-parameter. Every design parameter is entered in such a hierarchy.

4.2. Matching a sub-part of a larger case

Design cases are often composed of many other cases (or parts thereof). Each part is indexed separately. For example, the faucet is composed of a see-saw, a rigid-body, four gated-taps, and a right angle t-pipe. All these cases arc indexed in an abstraction hierarchy. The gated-tap's abstractions are: a tap, a flow-control device, a hydromechanical device, and finally a device. In addition to this hierarchy, each influence in the case's behavior graph is indexed in an Influence Hierarchy.



Parameter ← Parameter
Material ← Material
Material ← Signal
Signal ← Material
Signal ← Signal ...
Fluid-flow ← Fluid-flow
Fluid-flow ← Mechanical-Signal
Fluid-flow ← Electrical-Signal
Mechanical-Signal ← Fluid-flow
Mechanical-Signal ← Electrical-Signal
Water-Flow ← Translatory-Signal
Water-Flow ← Rotational-Signal
Water-Flow ← Electrical-Signal
Rotational-Signal ← Water-Flow
Translatory-Signal ← Electrical-Signal
Rotational-Signal ← Electrical-Signal
SLIDER TAP: Q ← X
ROTARY PUMP: Q ← E
TURBINE: Ω ← Q
MOTOR: Ω ← E
LINEAR-ACTUATOR: X ← Ei
SCREW-TAP: Q ← Ω
PISTON-PUMP: Q ← X ← E1
SLIDER-TAP-FAUCET: Q ← Xt

Figure 4-1: Partial Influence Hierarchy

The root of the influence hierarchy is the most generic influence˙. *Parameter* $\longleftarrow$ *Parameter.* The parameters can be cither materials, energy, or signals. This gives us nine possible influences. This hierarchy is expanded all the way down to the influences in the cases, thus providing a direct access *into* the innards of a case. A part of the influence hierarchy is shown below. The lowest level in the hierarchy points to actual cases or components inside cases. For example, the influence $Qh \overset{+}{\longleftarrow} Xt$ corresponds to the case: SLIDER-TAP-FAUCET which is a slider-tap inside the faucet case. Note that this is different from the general SLIDER-TAP case because the tap in the faucet has been modified to interface with other components in the faucet. The behavior may look the same, but the cases are physically different. We will see, later, why this distinction is important to maintain.

When CADET is given an influence as a goal, it can retrieve all cases that contain the goal. This can include all cases that match an abstraction of the goal. For example, the influence $Q \overset{+}{\longleftarrow} Y$, can be abstracted to the influence *Water-flow* $\longleftarrow$*Translatory-Signai* In this way, CADET is able to find components of larger cases that match a given goal specification, making it possible to find all "taps" in the case base, even if they are embedded in larger cases.

4.3. Multiple Influences in the Goal

The goal may consist of several influences arranged in the form of a graph. The aim of case matching is to find cases that have behaviors which correspond to the influences in the given goal

The matching process starts by finding prior cases that contain the nodes (parameters) in the influence graph. This returns a large set of cases for each node in the goal graph. Next, the Influence Hierarchy is used. For each individual influence in the goal statement, the hierarchy is used to identify all the cases that contain that influence. An intersection of the sets of cases that match each influence and each node yields a smaller set of cases that contain the goal influences. This does not, however, guarantee that the topology of the influences in the cases will match that of the goal. The final step involves checking each of the retrieved cases for the goal graph. If the entire graph is matched, then the corresponding components in the case is extracted,

4.4. Matching Multiple cases

When a goal has multiple influences, it may not be possible to find one case that satisfies the entire goal. In this situation, one has to find several components (from different cases) that can be synthesized to yield a final design. The process is as follows: For each influence, we first identify sets of cases that contain that influence. These sets can be viewed as possible colors that can be assigned to the various links in the goal graph. The next step is to select a case for each influence, such that a minimum number of cases are required to cover all the influences in the goal. A heuristic, polynomial-time, algorithm is used to find partitions of the influence graph. The algorithm works as follows: first, the largest contiguous set of influences that share a common case name is identified; next, the set is removed from the graph; and finally, the process is repeated till no unassigned influences are left.

Reconsider the flush tank design problem. The overall goal is to achieve: $Q \longleftarrow D$. Instead of this goal, assume we are given the following elaboration of subgoals: $Q \xleftarrow{+} \mathit{Mechanical\text{–}Signal} \xleftarrow{-} \mathit{Mechanical\text{–}Signal} \xleftarrow{+} D$. Each influence in the graph is used to identify relevant cases. In this instance, the influence: $Q \xleftarrow{+} \mathit{Mechanical\text{–}Signal}$ will match a regular tap, the tap in the faucet, and a pump. The next influence: $\mathit{Mechanical\text{–}Signal} \xleftarrow{-} \mathit{Mechanical\text{–}Signal}$ would match a regular see-saw, the see-saw in the faucet, a pulley, and a wedge. The remaining influence will match a simple float. The only case that is common for the first two influences is the faucet. The resulting design (Figure 2-2) is composed of the tap and see-saw combination taken from the faucet, and a float.

4.5. Exploiting Novel Combinations

A question that is often asked about the Case Based Problem Solving approach is: "If design problems can be solved by finding and synthesizing components such as taps, levers, and gears; then why should be keep larger cases in memory? The component hierarchy should be sufficient, why should the system have to find components embedded in the cases?"

In the mechanical design domain, cases provide useful sub-assembly "packages" of components. These sub-assemblies can reflect good design principles. For example, components are often made to share functions and may incorporate decisions that take advantage of, or compensate for incidental components interactions. The components in a prior design are often coupled efficiently to work as a sub-assembly of the larger design. These components are structurally modified to appropriately mesh with one-another. In addition, special connectors and fixtures may be used to interface the components. When a new design calls for the combination of two components behaviors, the modifications and interfaces need not be re-determined from scratch if relevant cases can be identified.

5, Index Transformation

In the example above, we said that the Hush tank's goal behavior is given by a set of influences. If, however, we were only given the original influence $Q \xleftarrow{-} D$, then it would not have been possible to recognize that the gated-tap, the see-saw, and the float can be combined to solve the problem.

In our domain, it often happens that the goal description docs not correspond to the appropriate cases in memory, A deliberate attempt has to be made to recognize what sub-behaviors can achieve the goal and how these sub-behaviors can be physically realized. We approach this problem in two steps: (1) If the goal description fails to retrieve cases from memory, then the goal is elaborated using certain behavior preserving transformations. (2) The elaborated goal is then used as indices to find relevant cases. Index transformation is a way to change the given salient features of the current problem to match the indices under which previous cases have been stored, thus making accessible to the problem solver previously inaccessible cases. The transformation technique described here is applicable to any domain in which behavior can be modeled as a graph of influences.

We will now examine two rules which arc used to transform given goals into more elaborate sets of influences that are behaviorally equivalent to the goal. The hypothesis is that, if one cannot find a case relevant to a given goal, then it might be possible to find several cases or parts of cases that are relevant to elaborations of the given goal.

Design Rule 1. If the goal is to have *x* influence z, and if it is known *a priori* that *u* influences z, then the goal could be achieved by making *x* influence u.

Design Rule 2. If the goal is to have *x* influence z and if it is known *a priori* that some two quantities *p* and *q* influence z then, the goal could be achieved by making *x* influence p or *q,* or both.

The two design rules transform a given influence into a more detailed set of influences that are behaviorally equivalent to the original influence. In CADET, the transformation is done in two ways: (1) by using domain laws and, (2) by hypothesizing new variables.

5.1. Elaboration Using Domain Laws

The influences implied by domain laws may be used to elaborate given goals. For example, assume it is our goal to achieve the influence; $z \longleftarrow x$, also assume that there arc no known designs that can achieve this effect directly. If, however, there is some domain principle which states that some quantity *u* influences z, then the goal may be achieved by having *x* influence *u.* The goal is hence elaborated to: $z \longleftarrow u \longleftarrow x$. This new influence graph is used as a new index into the case base. If cases or part of cases with influences that match the goal are found, they are retrieved and used.

5.2. Elaboration by Hypothesizing new variables

If the given domain laws are unable to find elaborations that can be realized by the cases in memory, one can try to hypothesize variables. The idea is to hypothesize new influences and then find cases which may be used to achieve those influences. For example, the goal $z \longleftarrow x$ may be elaborated to $z \longleftarrow Var1 \longleftarrow x$ using design rule 1. A new variable *Var* is hypothesized as an intermediary. The elaboration is then used to find cases in memory. This lime however, the system looks for two influences which match the goal and bind the unknown variable *Var1.*

As new variables are introduced, corresponding new influences are hypothesized. In addition, as influences arc all supposed to be based on physical laws or principles, the introduction of new variables implies that laws or principles, unknown to CADET, are being hypothesized. After hypothesizing influences, the case base is used to find prior designs which may embody some physical law or principle that matches the hypothesized influence. With this approach, one often retrieves cases from outside the current design domain that are analogically related to the current design problem. It is for this reason that CADET's solutions are innovative. Through the process of influence hypothesis and matching, the system is able to use physical laws and principles embedded in prior design cases to achieve its current goals.

Let's return to the flush tank example. Another possible elaboration of the original influence for the flush tank is: $Q \xleftarrow{+} Var2 \xleftarrow{+} Var1 \xleftarrow{-} D$. The first influence $Var1 \xleftarrow{-} D$, says that as the water level increases, some quantity *Var1* decreases. An ultrasonic distance measuring device, held over the water surface, could provide this behavior. The output of this device is an electrical signal. Let's call it *Sig* and bind it to *Var1.* The influence $Q \xleftarrow{+} Var2$ matches a basic tap by binding *Var2* to *X* (which is a linear movement). Finally, we are left with the influence $X \xleftarrow{+} Sig$ which says that when an electrical signal increases a body moves linearly in the *X* direction. A positioning device with a linear ratchet and motor can provide this function. The resulting design is shown in Figure 5-1.

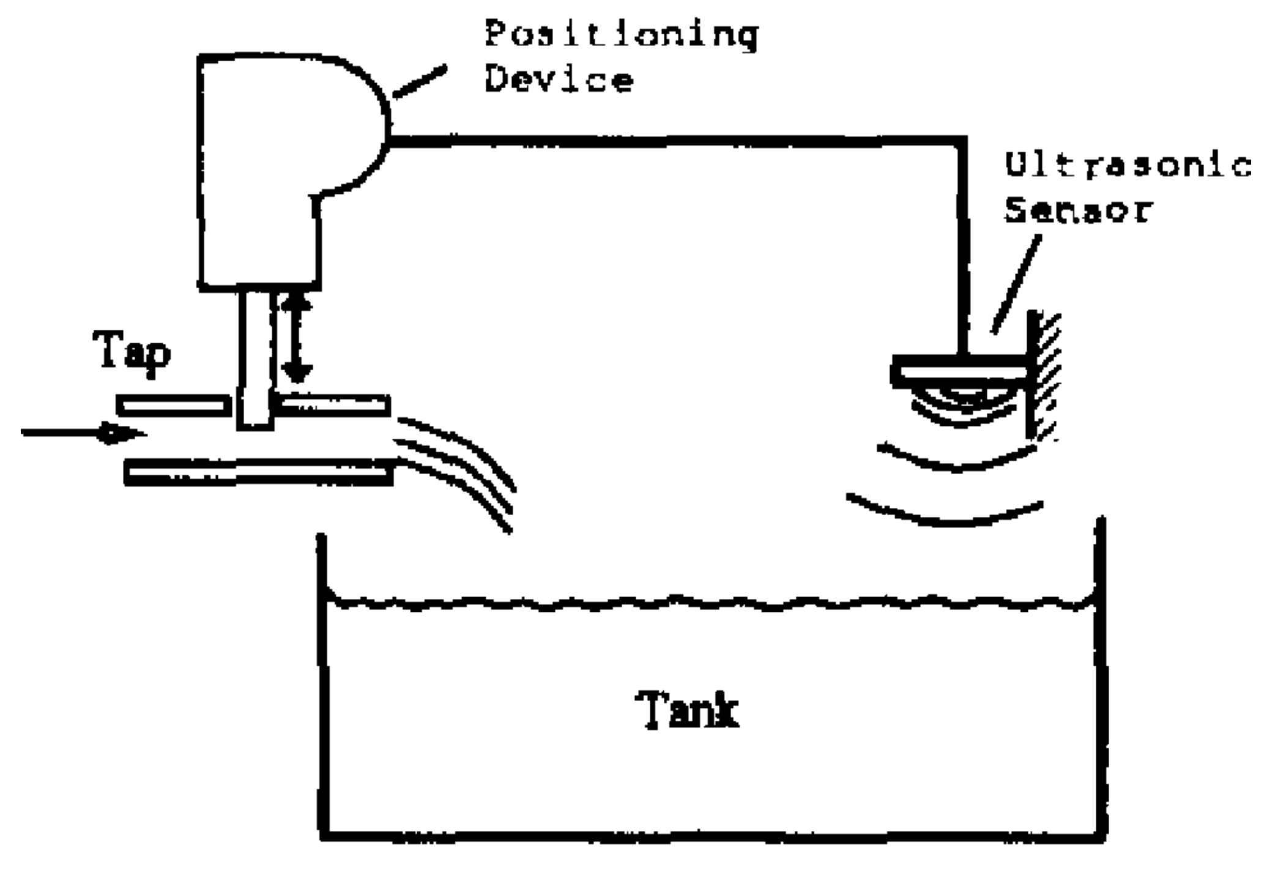


Figure 5-1: A flush tank exploiting extra-domain principle

6. Conclusions

We have presented an approach to the conceptual design of mechanical systems using a case base of previous designs that realize subfuncuons of the desired artifact. The process consists of applying behavior-preserving transformations, based on a qualitative calculus, to an abstract description of the desired behavior until a description is found that closely corresponds to some collection of relevant cases. The major benefits of the approach are: (a) it allows for retrieval of relevant cases and case parts based on behavioral thematic abstractions that enable use of cases from contexts other than hydraulics (e.g., electrical, electronic, chemical), (b) it does not impose a predetermined decomposition of the design, (c) it is a generative approach that utilizes knowledge of design principles, such as simplicity, and behavioral constraints to reason from design goals to possible solution structures, (d) it can identify "missing" cases, necessary for completion of the design, and (e) the resulting transformations are guaranteed to be behaviorally equivalent to the original specification.

References

[Goel & Chandrasekaran 89] Goel A., B. Chandrasekaran, "Integrating Model-based Reasoning and Case-Based Reasoning for Design Problem Solving," *Proceedings of the AAA! Design Workshop*, Expected in 1989.

[Gordon 61] Gordon W.J., *Synectics: The development of Creative Capacity,* Harper & Row, Publishers, NY, 1961.

fKoestler 64] Koestler, A., *The act of creation,* McMillan Publication Co., 1964,

[Kolodner.Simpson.Sycara 85] Kolodner, J.L., Simpson, R.L., and Sycara-Cyranski, K., "A Process Model of Case-Based Reasoning in Problem Solving," *Proceedings of the Ninth Joint International Conference on Artificial Intelligence (IJCAJ-85),* Los Angeles, CA, 1985. pp. 284-290.

[Navinchandra 87] Navinchandra, D., *Exploring for Innovative Designs by Relaxing Criteria and reasoning from Precedent-Based Knowledge,* PhD dissertation, M.I.T., 1987.

[Navinchandra 91] Navinchandra, D„ *Exploration and Innovation in Design: Towards a Computational Model,* Springer-Verlag series on Symbolic Computation, 1991-

[Navinchandra et.al 91] Navinchandra D., K.P. Sycara, S. Narasimhan, "Behavioral Synthesis in CADET, A Case-Based Design Tool," *Proceedings of the Seventh Conference on Artificial Intelligence Applications,* IEEE, Miami, Florida, Feb, 1991 .

[Owens 88] Owens, C., "Domain-Independent Prototype Cases for Planning," *Proceedings of the 1988 Case-Based Reasoning Workshop,* Clearwater, Fla., 1988, pp. 302-311.

[Redmond 90] Redmond, M., "Distributed Cases for Case-Based Reasoning; Facilitating Use of Multiple Cases/' *Proceedings of the Eigth National Conference on Artificial Intelligence, AAAJ-90,* 1990, pp. 304-309.

[Schank 82] Schank, R,C., *Dynamic Memory,* Cambridge University Press, Cambridge, 1982.

[Simoudis & Miller 90] Simoudis, E., J. Miller, "Validated Retrieval in Case-Based Reasoning/* *Proceedings of the Eigth National Conference on Artificial Intelligence, AAAI-90,* 1990, pp. 310-315.

[Sycara 87] Sycara, K., *Resolving Adversarial Conflicts: An Approach Integrating Case-Based and Analytic Methods,* PhD dissertation, School of Information and Computer Science Georgia Institute of Technology, 1987.

[Sycara & Navinchandra 89] Sycara, K, D. Navinchandra, "A Process Model of Case Based Design/* *Proceedings of the Cognitive Science Society Conference,* Ann Arbor, Michigan, 1989.

[Williams 90] Williams, B., "Interaction-Based Invention: Designing Novel Devices from First Principles/' *Proceedings of AAAI-90,* Boston, MA, 1990, pp. 349-356,