Retrieval Issues in Real-World CBR Applications
How far can we go with discrimination-nets?

Pádraig Cunningham*, Barry Smyth**
Donal Finn, Eamonn Cahill****

* Department of Computer Science
Trinity College
College Green
Dublin 2
Ireland
Phone: +353-1-772941
Fax: +353-1-6772204
Padraig.Cunningham@cs.tcd.ie

** Hitachi Dublin Laboratory
Trinity College
College Green
Dublin 2
Ireland
Phone: +353-1-6798911
ecahill@hdl.ie
bsmyth@hdl.ie
dfinn@hdl.ie

Abstract

We propose that analogical reasoning and case based reasoning (CBR) tasks can be usefully characterised as a continuum reflecting the remoteness of the remindings involved. Remindings in CBR are generally between semantically *close* cases while analogical reasoning depends on more abstract remindings. Rather than there being a strict demarcation between CBR and analogical reasoning on these grounds (with analogical reasoning concerned with inter domain remindings and CBR dealing with remindings within one domain) there is a continuum of cognitive tasks that draw on past experience during reasoning. Simpler tasks like diagnosis and classification are located near the CBR end while more complex tasks like creative design are located towards the analogical reasoning end. The question is how far towards the abstract end of the continuum can the index-based retrieval techniques that are effective in CBR be used (eg. discrimination networks). We are considering episode retrieval as a two stage process; the first stage being the initial filtering of the case base, and the second stage selecting the best case from this candidate set. We focus on the base filtering stage and conclude that discrimination networks are adequate for comparatively complex cognitive tasks such as routine design. However, we argue that CBR systems for non-routine design should provide interactive case retrieval and act as CBR *assistants*.

Keywords: Abstract remindings, discrimination networks, interactive CBR

1 Introduction

Case-based reasoning (CBR) and analogical reasoning (AR) both retrieve episodes from memory based on the similarity of that episode to a case or scenario under consideration. Conventionally, analogical reasoning is understood to be concerned with inter-domain remindings where the domain of the analogue may be different to the domain of the target scenario. CBR implementations tend to depend on single domain remindings where the base and target cases come from the same domain. Our assertion is that this demarcation is artificial and the distinction is better characterised as a continuum of abstraction of remindings with CBR towards the concrete end and AR near the more abstract end. This assertion is based on the observation that CBR systems may be required to support remindings between different sub-domains of a 'single' problem domain. Design reuse in routine design might involve using old solutions from the same domain, whereas remindings to support more innovative design might need to be from different domains.

This continuum of design tasks reflects the difficulty of the problem solving task being supported. In design there is a broad acceptance that tasks can be partitioned into three categories (Brown & Chandrasekaran 1985; Gero 1990; Visser 1991); these are as follows:-

Routine Design: This is the simplest category of design task requiring knowledge based problem solving. The design process will follow well known procedures; new designs will be parametric variations of previous designs.

Non-routine Design: This is innovative design that will produce an artefact significantly different from existing ones. Useful remindings in non-routine design are likely to be abstract.

Creative Design: Creative design will produce a new type of artefact. This will create a new state space of designs (see Figure 1). Evidently this is qualitatively different from non-routine design as it expands or shifts the problem space.

This paper focuses on routine and non-routine tasks and emphasises the extra difficulty in supporting remindings for non-routine design. One of our conclusions is that a pragmatic treatment of this difficulty is to work towards interactive CBR systems for non-routine design — case-based design assistants.

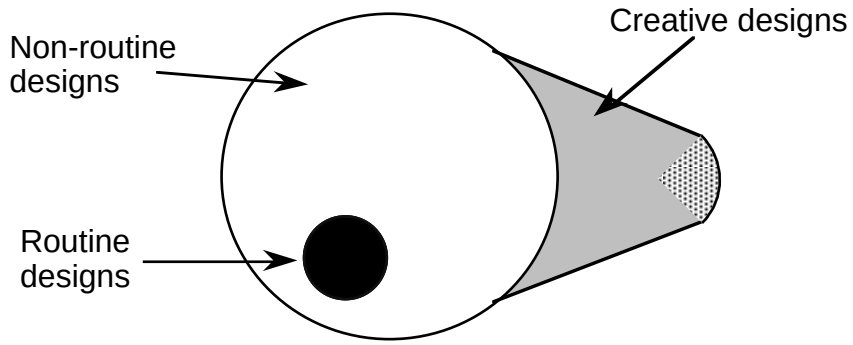

Figure 1. State space of designs (Gero 1990).

In most CBR and AR systems case retrieval is a two stage process. First there is a pre-selection stage, often called **base filtering**, where a small set of candidate cases is selected. Then there is a **mapping** of the target case to these candidates to find which offers the best match. In straightforward CBR systems discrimination networks (D-Nets) are an effective method for organising the case memory to support base filtering. Towards the AR end of the spectrum, where remindings are more abstract, or where the system may need to support remindings at different levels of abstraction, it becomes more difficult to organise the case-base as a D-Net. The argument sometimes presented is that this dependence on indices will preclude remote remindings (Waltz, 1989; Thagard & Holyoak, 1989). In this paper we will look at some practical examples of CBR and assess the extent to which this is true.

We will begin by examining approaches to memory organisation and retrieval in AR and CBR. Some classical analogies will be discussed with emphasis on the difficulties involved in triggering the remindings. We will present D-Nets as a typical example of an abstraction hierarchy and also D-Nets with redundancy which overcome some of the shortcomings of D-Nets. These ideas will be elaborated with a description of a typical CBR application that is adequately implemented as a redundant D-Net. Then we will look at a more complex CBR application in routine software design, requiring more abstract remindings, that causes problems for

memory organisation based on indices. We will consider the use of more abstract indices and also index transformation as solutions to these problems. In section 4 we argue that these indexing problems are more acute in non-routine design and propose that CBR systems for innovative design should be interactive with user involvement in case retrieval.

2 Episode-Based Reasoning

Both analogical reasoning and case-based reasoning methods fall under the general category of episode-based reasoning (EBR) where problem solving knowledge is characterised as a set of episodes, each representing the solution to a specific problem situation. A new problem (the *target*) is solved by retrieving a similar episode (*base episode*) from memory, and its solution is then modified to conform with the target situation. Clearly, the retrieval of an appropriate case is vital to the success of such techniques. Retrieval constitutes a massive search problem which is exacerbated by the fact that we are not concerned with complete matches but partial matches. Conventional methods take a two-stage approach. The initial retrieval stage (called *base filtering*) is responsible for selecting a small number of candidate episodes which are considered contextually similar to the target situation. The second stage (*mapping*) performs a detailed mapping between the target situation and each candidate episode to determine a single *best* episode for modification. The motivation for this two-stage approach is that the computational expense associated with the second stage is lessened because only a small number of candidates are considered for mapping. Before discussing base filtering in more detail we will compare CBR and AR in the context of retrieval.

2.1 A Perspective on Retrieval

Most work on analogy has concentrated on inter-domain remindings where the relationship between the base episode and the analogous new episode is of an abstract or thematic nature (for example Keane 1987). Alternatively, CBR research has focused on single-domain remindings where significant overlap of surface features exists between the base and target episodes. Such differences in the nature of remindings impose different constraints on the organisation of the episode memory and the retrieval process.

In particular, as can be seen from Figure 2, much of the variation between AR and CBR is identified by a relative shift in emphasis from base filtering (in the case of CBR) to mapping (in the case of AR). Essentially, the surface feature based remindings of CBR necessitate simple, shallow mappings whereas the abstract remindings of AR require more complex 'structure mapping' type techniques (see Falkenhainer Forbus & Gentner, 1989).

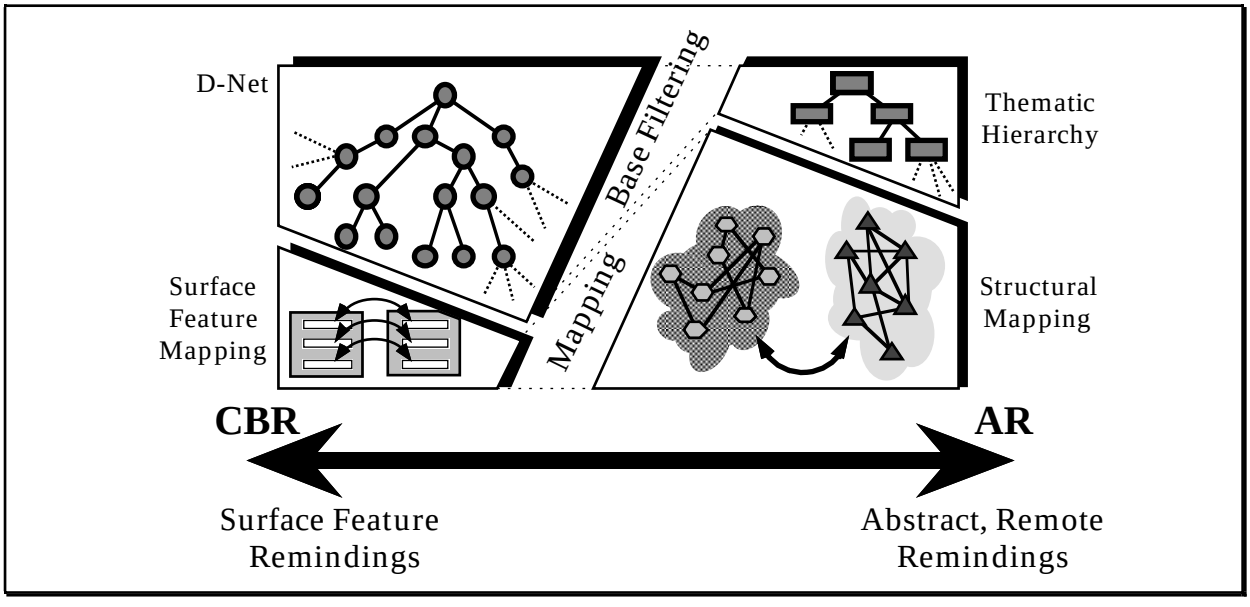

Figure 2. A perspective on retrieval in CBR and AR

We are concerned with the organisation of episode memory from the point of view of base filtering. Episodes must be described in terms of their salient features. These descriptions should be structured not only to allow for efficient retrieval but also to facilitate remindings at the appropriate level of abstraction. Supporting the kind of 'close' reminding that is required for routine design tasks is not difficult. However as we move toward more non-

routine design tasks the remindings need to be more abstract. These issues of indexing and remindings are discussed in the following section.

2.2 Memory Organisation

The organisation of episode memory must serve two objectives in base filtering; (a) to provide for adequate *remindings* so that all suitable episodes are considered and (b) to ensure for efficient *indexing* so as to guarantee fast retrieval of cases. Considering these in more detail:

2.2.1 Indexing

In general in EBR the expedient view is that cases should be indexed in order to support directed search during base filtering. However, it is evident that indexed memory will present difficulties in supporting remote or abstract remindings. The work of Waltz and Stanfill and that of Thagard and Holyoak is explicitly directed at cross domain remindings and memory organisation that supports abstract remindings, (Thagard & Holyoak, 1989; Thagard, Holyoak, Nelson & Gochfeld, 1990; Waltz 1989; Stanfill & Waltz, 1986). However as a consequence, retrieval is only possible when cases are stored without indexing. This requires that memory is content-addressable in that all information about stored cases is matched with the target case and the 'best' match is returned. It is assumed that the exhaustive search implied in this approach is made feasible by parallel hardware. *

2.2.2 Remoteness of Reminding

The other important characterisation of memory organisation is the remoteness of the remindings that are supported. The simplest perspective here is that CBR is concerned with reasoning within one domain where base filtering is done on the basis of surface features. AR involves inter domain remindings where episodes share a structural rather than a semantic similarity and base filtering is based on abstract remindings.

The key issue concerning the role of indexing and reminding in memory organisation is that indexing cases according to their features permits the case base to be organised into an abstraction hierarchy thereby facilitating base filtering. However, abstraction and classification of these features becomes increasingly difficult as the nature of the remindings become more abstract. This issue is best considered if we examine two examples taken for the opposite ends of the EBR spectrum. The first example considers a purely structural analogy between electrical and mechanical systems, where it is difficult to uncover even the most abstract features that the two cases share, so the analogy is valid only on the basis of structural isomorphism between the cases. The second example examines a CBR system for estimating house prices that needs to support only the most superficial type of remindings.

2.2.3 Abstract Remindings: an AR example

Comparison of electrical circuits and mechanical systems provides a rich supply of engineering analogies. Figure 3 illustrates two very different systems that bear a strong structural similarity. Both exhibit damped oscillation as shown in the behavioural graph in the centre of the diagram. If the mass in the mechanical system is displaced it will oscillate about its equilibrium position with constant frequency and decreasing amplitude. If the capacitor in the electrical circuit is discharged its charge q will oscillate and decay to zero.

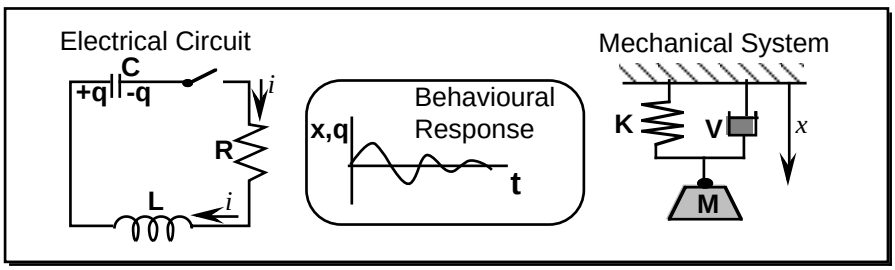

Figure 3. Two systems that exhibit damped oscillation.

* Besides this 'anti inexing' research there are also an 'anti discrimination networks' arguments within the indexing school. The alternatives to retrieval basied onD-nets in indexing are parallel algorithms of spreading activation. (Ownes 1988; Kolodner 1988; Domeshek 1989; Domeshek 1991)

This is quite a fundamental analogy in that both systems are governed by the same differential equation:

$$L\frac{d^2q}{dt^2} + R\frac{dq}{dt} + \frac{1}{C}q = 0 \qquad \text{or} \qquad M\frac{d^2x}{dt^2} + V\frac{dx}{dt} + Kx = 0.$$

This mathematical relationship can be expressed as a set of causal relationships that could be the basis for a structural mapping. However, what is of interest for us in this example is how one of these cases might be retrieved in base filtering as a potential analogue for the other. These cases share no surface features in common so the question is how could we plausibly index these with functional attributes that will capture the similarity? An abstract feature **damped-oscillation** captures the commonality very well. Even if the mechanical example were indexed as **damped-harmonic-motion** we could argue that this would be retrieved as a specialisation of **damped-oscillation.** The analogy would be more perspicuous if the capacitor and spring belonged to an abstract class **energy-reservoir** and the resistor and dash-pot shared an **energy-dissipater** superclass. The issue is, are we stretching credulity by expecting a knowledge-base to have these classifications?

2.2.4 Surface Remindings in a CBR example

Figure 4 shows two example cases from a case-based system called Rachman that can predict the selling value of a house given some details about it. The system contains a large case base of houses and their selling prices and it will retrieve a case or a set of cases describing similar houses and their selling prices. These prices can be adjusted depending on differences between the target and base cases to estimate the price of the target house. This system is comparatively straightforward but is equivalent to a host of potential CBR applications, for example loan risk assessment and help-desk assistants. The complexity of this problem is greatly relieved by having a well populated case base, so good matches can be found and the required adaptation is not difficult.

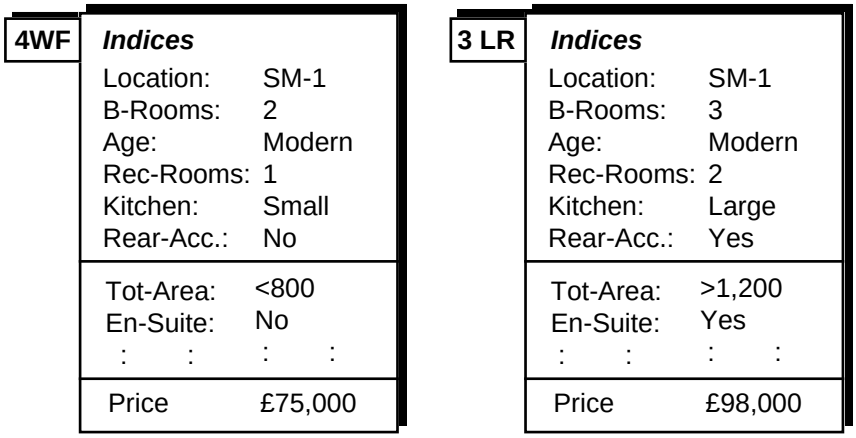

Figure 4. Two sample cases from the Rachman Case-Base

The cases are divided into two sets of features, the index features and the internal features. The index features are the most strongly predictive features and form the basis for the D-Net. The main problem with the D-Net approach is that it forces a strict ordering of the index features, in this example the cases might be organised first under location, then number of bedrooms, etc. However, different users may have different priorities; some, for instance, might consider the number of bedrooms to be more important than location. In addition, it will not be possible to retrieve matches for cases that have missing features as the retrieval process will not be able to search below the level of that feature in the network.

These problems are largely solved by introducing redundancy into the D-Net. This means that the network supports alternative orderings on the index features (see Figure 5). The extent to which redundancy can be introduced into the network is limited because the size of the network grows in proportion to the number of orderings supported. Retrieval in Rachman returns clusters of cases and the cases are ranked according to their frequency of occurrence in these clusters.

The purpose of this example is to show the success of index based retrieval and to illustrate some of the characteristics of an indexed case base. The success of this approach depends on having a case base that can be characterised by a small set of indices that can be determined in advance. It is also important that the case base is well populated so that near or exact matches can be found.

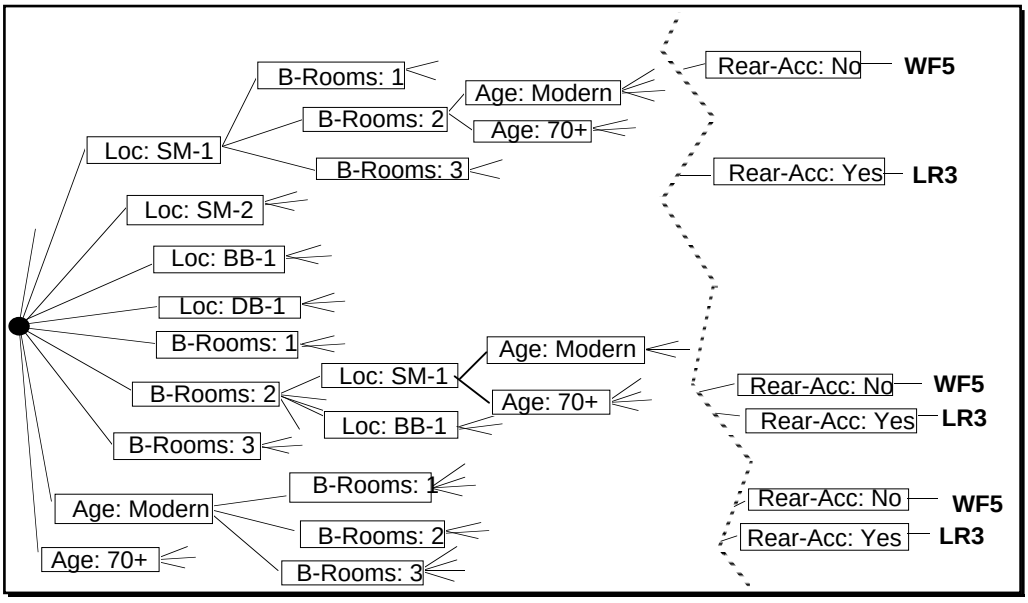

Figure 5. A portion of the Rachman case-base organised as a D-Net with some redundancy.

2.3 Case Indexing for Remote Remindings

The focus of research in AR to date has largely been on adaptation of a base case to solve a target problem rather than on the recognition of potential candidates from within a case base (Mostow, 1989). On the other hand work classed as CBR has tended to concern itself with relatively larger numbers of semantically similar cases from a single domain (e.g. CHEF Hammond, 1989). These cases also tend to embody a significant degree of structure. Consequently the difficulty of adaptation is alleviated somewhat and instead the retrieval process is a more immediate problem than in AR. So CBR systems are generally associated with single domain applications whereas AR is more usually associated with cross-domain tasks such as non-routine design.

As discussed in the preceding sub-sections this division of EBR into AR and CBR based on the notion of intra- or inter-domain remindings is somewhat nebulous. The very concept of a domain is a subjective and ill-defined entity which is open to continual refinement e.g. is the reminding of a proof of a geometry theorem pertaining to angles for the purpose of proving a theorem for line-segments cross-domain or not? Our experience, particularly in research on software design using CBR, is that systems designed to operate within 'one' domain may be required to support mappings between sub-domains of that domain (see Section 3). Therefore, our perspective is that there exists a complete spectrum of remindings at different levels of abstraction.

It has been argued (Thagard, Holyoak, Nelson & Gochfeld, 1990; Waltz 1989) that for cross domain remindings, indexing using abstract remindings is problematic and therefore exhaustive search of the case base with subsequent structural mapping is a preferable approach. However it is our opinion that effective EBR systems will only materialise when this continuum of abstraction is explicitly acknowledged in the retrieval process.

To this end we propose a hierarchical index structure for each case as illustrated in Figure 6. This scheme aims to support remindings across increasingly remote domains and at the same time to reflect the fuzzy nature of the degree of semantic commonality of cases. A more useful concept than "domain" for characterising the remoteness of reminding is that of *semantic distance* between cases. The greater the semantic distance between the cases the more remote the reminding required and consequently the more abstract the perspective (and index) required to achieve a match.

As we progress up the hierarchy there is a transition from data-type indices to more knowledge-type indices which possess richer semantics. The scheme attempts to characterise the human cognitive process of case retrieval using different levels of knowledge and would appear to be more psychologically plausible than a flat index structure. We will explain these indices in the context of the example of the LCR circuit and mechanical spring-dash pot system presented in Section 2.2.

Surface Indices reflect observable features in a case (capacitor, coil and resistor *etc.*) and would be the set of indices used at the strictly CBR end of the spectrum.

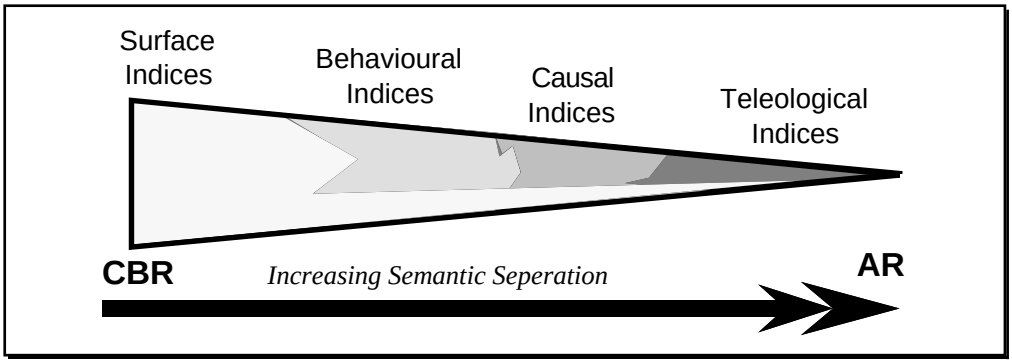

Figure 6. An Index Abstraction Hierarchy

Behavioural Indices embody emergent behaviour of the case stemming from the interaction of the components within the case. They capture the characteristics of the overall system (e.g. damped, oscillatory motion) and it is at this level that matching between the electrical and mechanical systems could be achieved. For planning-type systems a more appropriate notion for this level of index would be the impact of the case on the environment.

Causal Indices capture explanatory knowledge about how the behaviour comes about as a consequence of the interaction of the sub-entities within the case e.g. the inertial effects of the mass (coil) transfers energy between the two energy reservoirs, namely the spring and the mass (capacitor and coil). During the transfer, energy is dissipated by the dash-pot (resistor).

Teleological Indices detail the purpose or goal of the case (to act as a car suspension or to form a tuned radio circuit for example).

This example of analogical reminding illustrates the conceptual and theoretical merits of the proposed indexing scheme for remote remindings. In the next section we will address the practical issues associated with exploiting this scheme in a real-world CBR system involving routine software design.

3 Remindings in Routine Software Design

The Déjà Vu system is a CBR tool for routine design of plant control software involving the reuse of old software designs. The problem domain of Déjà Vu has already been introduced in (Smyth & Cunningham, 1992) and so will only be described in outline here. An example of the type of code is shown in the Solution section in Figure 7. The software is for controlling loading and unloading equipment in a steel mill. The code is expressed in this network representation that is compilable into executable code. This sample case controls the movement of a buggy carrying an empty spool. Buggy*1 is a two speed buggy, so stopping is a two stage process with the buggy switching to its slower speed 200mm from its destination. This case is a sub-component of a complete solution sequence.

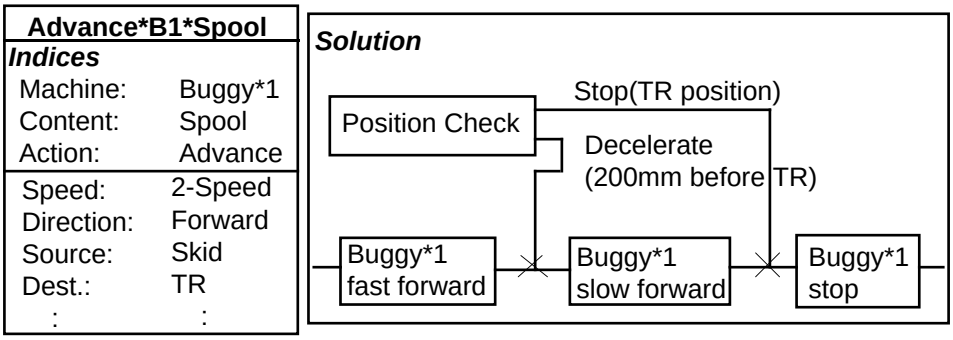

Figure 7. An example case from Déjà Vu.

Describing the scenario in more detail; this buggy is part of a system for carrying coils of steel or empty spools between a storage area (called a skid) and a tension reel where it is mounted on the rolling mill. The load is not mounted directly on the buggy but is carried on a lifting device that is mounted on the buggy. This lifter is used to adjust the height of the load for loading and unloading. The lifter can be a one or two speed device - a component case for a two speed lifter is shown in Figure 8. It can be seen that the solution for the two speed lifter has the same structure as the buggy case described above so it should be possible to reuse the solution from

one in designing the other. This is problematic because these two cases do not share important surface features. So if this lifter case were a target case, the useful Advance*B1*Spool case would not be retrieved using this indexing scheme.

Building a D-Net to characterise a simple domain such as RACHMAN's property domain is a fairly straight forward task. Such a domain can be modelled in terms of its components (an identifiable, finite set), and thus can be completely described. However in more complex domains concerned with reasoning about actions and change this becomes a far more difficult task requiring consideration of more complex behavioural and functional domain features.

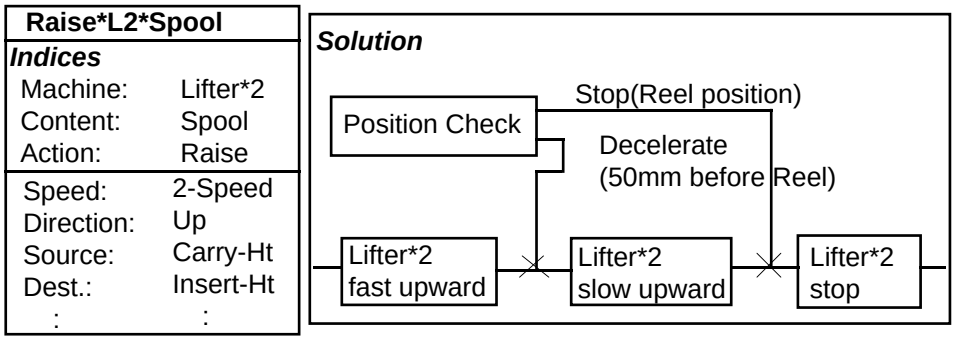

Figure 8. A two-speed lifter case.

The current example from Déjà Vu has attempted to describe two cases in terms of surface features, indicating the ACTION, VEHICLE, and CONTENT of the cases. The problem is that the observed difference in action types causes the cases to be indexed under different routes within the case-base, even though behaviourally and structurally the cases are very similar. There are two approaches to solving this problem, effectively drawing the cases closer together: abstract indexing, and index transformation.*

3.1 Abstract Indices

The problem that we are addressing concerns the fact that even though the two cases differ significantly in terms of their surface features (BUGGY*1 and ADVANCE against LIFTER*1 and RAISE), they exhibit strong behavioural and structural similarities. These similarities are not captured by the chosen indices and so the cases are distant from each other within the case-base.

One solution is to capture this behavioural and structural similarity by adding abstract behavioural indices, such as BEHAVIOUR = MOTION and BEHAVIOUR-TYPE = 2-SPEED. Now the two cases appear as siblings within their appropriate behavioural route. This is illustrated below in Figure 9 where the above behavioural indices are used to capture the inherent similarity between the ADVANCE and RAISE cases. Therefore, in this 'behaviour' section of the net these cases are classified as similar without the need for index processing techniques such as index transformation.

Referring back to the pyramid of index types and the CBR-AR continuum shown in Figure 6, as more abstract remindings are required, so these must be represented as different types of routes within the case-base structure. In this way cases are considered for retrieval at different levels of abstraction. In the Déjà Vu example we are concerned with two levels, namely component and behavioural, and it is at the behavioural level that the inherent similarities between the cases becomes apparent.

Introducing these behavioural features does solve the problem but there are some caveats. This 'closeness' in the network depends on the redundancy of the net supporting just the right ordering of the index features to bring the cases together. In addition 'less relevant' matches based on surface features will also be retrieved by the base filtering. This forces a consideration of what exactly are the requirements on base filtering. One view would be that the onus is on the base filtering process to only produce cases that are truly relevant. This appears to require that the relative importance of indices be context dependant - that dynamic indexing be supported. Introducing redundancy into the D-Net does support different index orderings but it cannot *suppress* orderings that are not relevant in particular contexts.

* It is worth noting that if there were existing two speed lifter cases in the case base then this problem would not arise as they would offer a better match. However, the essential point that there is a similarity that is not being captured continues to be valid.

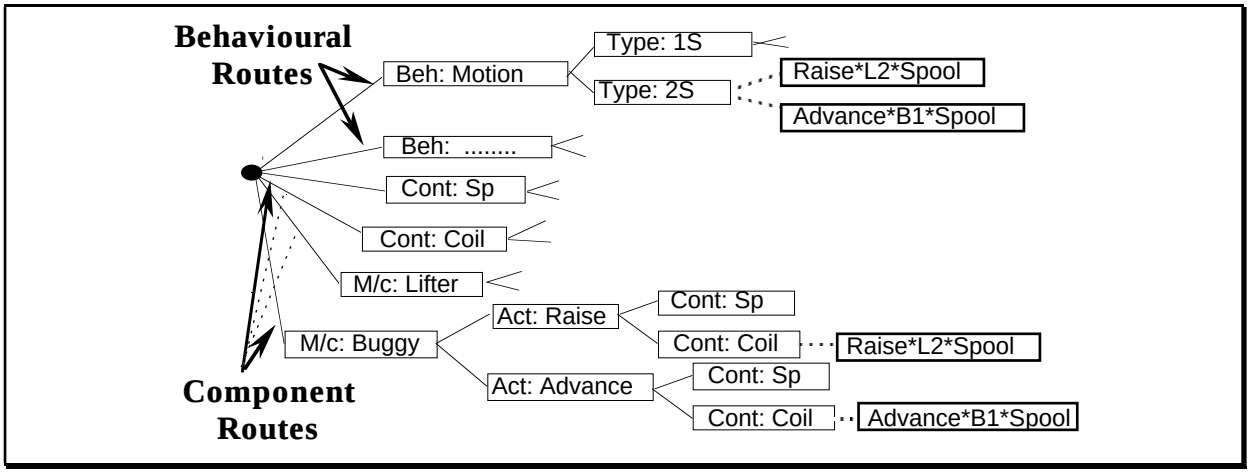

Figure 9. Incorporating behavioural indices facilitates recognition of behavioural similarity.

The stance taken in Déjà Vu is that it is acceptable for the base filtering to retrieve cases that may subsequently prove to be irrelevant: the important point being that the set of cases returned by the base filtering must contain the best case. In the case mapping phase, these cases are examined to determine the one that can most easily be adapted to fit the target problem (Smyth & Keane 1992) for more on adaptation driven case selection).

3.2 Index Transformation

Quoting from Sycara and Navinchandra:- "Index transformation changes given salient features to match the indices under which previous cases have been stored making previously inaccessible cases accessible." (Sycara & Navinchandra, 1991) In the current situation this involves altering one or more of the indices so it is evident that a strong model of the problem domain is required to support transformation. In Déjà Vu the domain is modelled using a frame representation that captures the attributes of the domain concepts and the interactions between them. In this model Buggy and Lifter are sub-classes of Mover and inherit the Speed slot from that class. The appropriate transformation in this situation involves relaxing the Buggy index and seeking cases indexed on siblings of Buggy. In the current system, on retrieval failure, there is no way of knowing in advance which indexes need to be relaxed so all indexes need to be transformed in turn. This greatly increases the retrieval time. This approach has the advantage that it escapes the rigidity of the static indexing by exploiting the available domain model.

In the absence of a close match the system is retrieving a case that is structurally similar but has different surface features to the target case. This means that the adaptation task is more substantial. The very pragmatic argument could be presented that there should be little interest in more abstract remindings in Déjà Vu because the adaptation process would be too complex.

4 Remindings in Non-Routine Software Design

In Section 3, we found that for what is ostensibly a single-domain, routine design problem, the need for abstract remindings did arise, and that surface features alone were often inadequate for retrieving suitable base cases. Furthermore, of the two approaches considered as a means of addressing this problem, *Index Transformation* and *Abstract Indices*, neither was found to provide an ideal solution. Nevertheless for routine design, the problem did prove to be tractable for at least two levels of abstract indices in a D-Net. These conclusions have challenged us to explore how far D-nets can be used to support the more complex remindings that are to be expected with non-routine software design tasks. Therefore, in this section, we examine this issue by attempting to support non-routine design remindings in Déjà Vu.

4.1 Can D-Net based remindings support non-routine design ?

In plant control software generation, we consider non-routine design to be the synthesis of novel programs from a well defined case base of previous software design episodes. What distinguishes it from routine software design, is that although, potential base cases are usually structurally and functionally quite similar to the target case, such cases often appear significantly different from a surface or indexing perspective. Therefore, a base case may require some structural modification and can result in new solutions that lie outside the realm of the existing software routines (Gero 1990).

Our observations from experimenting with Déjà Vu on non-routine software design tasks, is that retrieval is often based on finding similarity between apparently different surface indices through a common abstract ancestor. Therefore, to maintain an effective retrieval mechanism for non routine design there must be support for distant remindings. The question arises as to whether D-Nets can be used as a mechanism for supporting this degree of concept matching. It seems unlikely that introducing multi-level indices into a D-Net will ease the situation, but instead will probably exacerbate the problems associated with the net, primarily in the explosion of the number of possible orderings. Therefore we believe that there is a limit to the applicability of current automatic base filtering techniques. This problem would be alleviated somewhat if the base filtering was based on using disjoint nets for each level of the index hierarchy. The construction of these nets would then be less complicated, and case retrieval within a net much less computationally expensive. However, such an approach would hardly be capable of maintaining the complex relations required for non-routine design.

This has led us to conclude that rather than trying to automatically support the type of novel and abstract remindings typical of non-routine design, it may be more practical to involve the software designer in the retrieval process itself.

4.2 Case Retrieval Assistance

We have suggested that the designer be allowed to interact with, and assist, the retrieval process. It is our belief that index transformation is a suitable point where man-machine interaction can usefully occur. Index transformation allows specifications to be interpreted from different perspectives. The user can provide the context sensitivity that can highlight certain features, de-emphasise others, and expose hidden or implicit goals. While such subjective manipulations are difficult for a computer to perform, they are particularly easy for a human to execute.

There are many problems with index transformation. Firstly, it requires a strong domain model. It is difficult enough to specify the type of transformations which can occur, let alone represent the complex set of interactions which can result due to their application. Secondly, in a given situation it is very difficult to determine which transformation processes should be applied to which indices. One common approach is to try all applicable transformations. While this may be viable for the simpler transformations associated with routine design, the proposal quickly becomes intractable when applied to non-routine design tasks. And thirdly, by its very nature retrieval is extremely sensitive to any changes made to a specification's indices, and since transformation methods can significantly change the indices, it is likely that careless application will cause large bodies of irrelevant cases to be selected by the base filtering process. These problems, and more, are substantially alleviated by allowing interaction between man and machine.

There are a number of different modes of index transformation: *Abstraction*, *specialisation*, *elaboration*, and *condensation*. Each provides general strategies for altering particular aspects of a specification in a certain way.

4.2.1 Abstraction and Specialisation

A common index transformation strategy is to abstract and/or specialise a given index to provide an alternative retrieval context. This approach was used to good effect in section 3.2 where the Buggy index was abstracted to Mover and in turn specialised to Lifter, resulting in the retrieval of cases which contained the sibling index (Lifter). Such simple modifications are indicative of the type of transformations required in routine design; transformations are constrained to only one or two levels of abstraction or specialisation. Unfortunately, this restriction is not satisfactory for non-routine design tasks, where remindings will occur on the basis of more distant similarities. Relaxing this restriction, allowing more far reaching transformations, will facilitate the access of such cases. However, the likely result is that many irrelevant cases will also be selected, thereby reducing the usefulness of base filtering. This problem may be relieved by providing pruning heuristics to narrow the transformation path, but in general the context sensitive nature of the transformation process makes this very difficult.

4.2.2 Elaboration and Condensation

Often specifications are vague and fail to bring about the appropriate remindings due to the lack of certain features. Alternatively, representation complexities may hinder the isolation of useful cases due to excessive remindings. The result is that specifications must be elaborated or condensed. Automatic elaboration and condensation techniques exhibit problems similar to those present in the previous transformation methods; while control heuristics may work well for routine retrievals, the detailed transformations needed for non-routine remindings are often intractable.

Each of these modes of transformation can be enhanced by user interaction. Not only can the user judge which indices should be altered, but he/she can also select the most useful sequence of transformations in a given situation, and determine whether the result is likely to prove useful or not. Furthermore, the user can be assisted in making these decisions by the CBR system itself. Previous retrieval results may suggest suitable candidates for transformation. In addition elaboration, abstraction, and mutation can be guided by providing the user with access to the domain model.

4.3 An Example Transformation Episode

In this example we will briefly demonstrate how the above methods can result in the kind of remindings necessary for non-routine design. The target specification is concerned with controlling a Buggy so that it avoids some stationary obstacle (a Block) which lies in its path. The base case which needs to be retrieved controls the alignment of a load carrying Lifter with a Tension-Reel.

Figure 10 shows the base case's indices and solution. The Lifter's platform is initially positioned below the *Tension-Reel* (the *alignment-point*) * . First, the platform is raised quickly to a position just below the Tension-Reel. Next, the platform position is fine-tuned by slowly lifting it to the exact Tension-Reel alignment height.

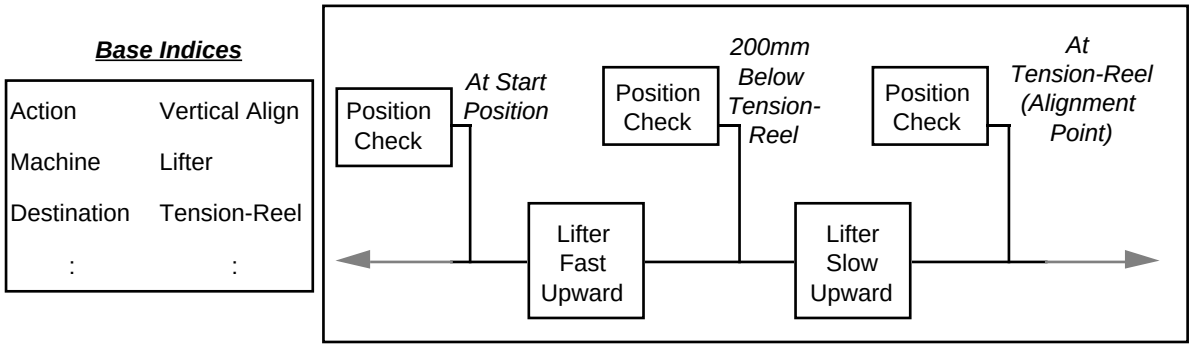

Figure 10. The base case's indices and solution.

Figure 11 show the indices of the target specification and the solution derived from the retrieval and adaptation of the above base case. Briefly, the Buggy, initially heading towards the Block, veers quickly off to the left to a position just right of the *avoidance-point*** . Next, the Buggy position is accurately adjusted to coincide with the avoidance-point, by move slowly to the left. At this point the Buggy can continue moving forward and avoid colliding with the Block.

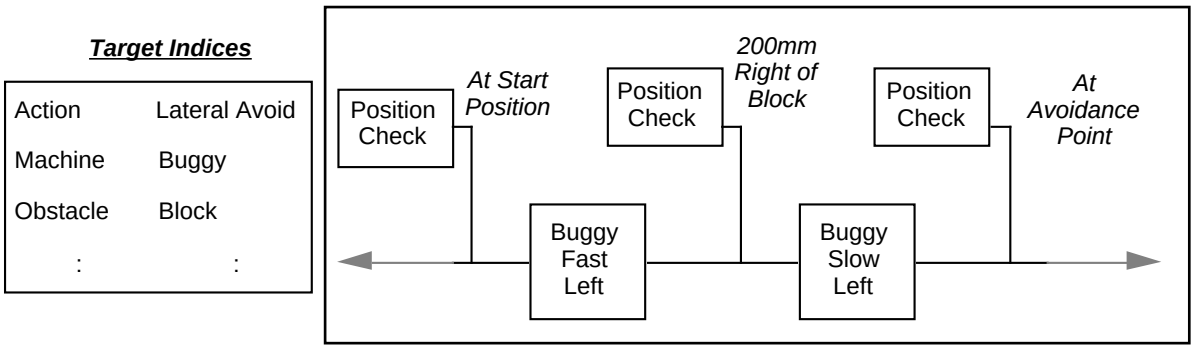

Figure 11. The target indices and resulting solution.

The important point is that the two index sets do not coincide directly and yet the reminding is useful because of an abstract similarity between the alignment and avoidance strategies. By appropriately transforming the target indices we can achieve such a reminding.

---

* The alignment-point is the correct vertical height of the Lifter platform for alignment with the Tension-Reel.

** When the Buggy reaches the avoidance-point it will be able to continue in forward motion and just avoid contact with the obstacle.

First of all, the Buggy index can be transformed into the Lifter index by abstracting it to Vehicle and specialising to Lifter. Because this transformation on its own would result in the retrieval of a large number of cases similar only in terms of the machines they use, we must transform the action indices also. The Avoid index can be elaborated to match with Alignment by recognising that avoiding some obstacle is similar to aligning with the free space beside the obstacle; that is, an implicit avoidance goal is to align with some position free of obstacles.

This is the sort of transformation that we see the user being most helpful in assisting (especially the more complex elaboration processes). It is our experience that many transformations are too subtle and context sensitive to encode as robust transformation rules and that user assistance represents a viable compromise in the move to real-world AI systems.

5 Conclusion

In analysing analogical reasoning and CBR systems an important consideration is the remoteness of the reminding that the system needs to be able to support. The more difficult the problem solving task, the more abstract the remindings that will be useful. In particular, more abstract remindings are needed in non-routine design than in routine design.

In this paper we have argued for the utility of index based retrieval using D-nets in simple CBR tasks such as routine design. However, we have presented examples of the more abstract remindings that are required for non-routine design – a more difficult task. It is difficult to see how these abstract remindings can be effected using D-nets, even using automatic index transformation.

The solution that we propose is that CBR systems for non-routine design should be designed as interactive systems with user involvement in retrieval and index transformation.

Acknowledgements

We would like to acknowledge with thanks the comments of an anonymous reviewer on the submission draft of this paper.

References

Brown D.C., & Chandrasekaran B. (1985). Expert Systems for a class of mechanical design activity. In J.S. Gero (Ed.) Knowledge Engineering in Computer-Aided Design, Amsterdam: North Holland.

Domeshek E.A. (1991). What Abby cares about. In Proceedings of DARPA Case-Based Reasoning Workshop 1991, Washington, D.C., 8-10 May 1991. San Mateo, California: Morgan Kaufmann.

Domeshek E.A. (1989). Parallelism for index generation and reminding. In Proceedings of DARPA Case-Based Reasoning Workshop 1989, Pensacola Beach, Florida, 31 May - 2 June 1989. San Mateo, California: Morgan Kaufmann.

Falkenhainer B., Forbus K.D., & Gentner D., (1989). The Structure Mapping Engine: Algorithm and Examples. Artificial Intelligence, 41, 1-63.

Gero J.S. (1990). Design Prototypes: A knowledge representation schema for design. The AI Magazine, 11, pp. 26-36.

Goel A.J., Kolodner J.L., Pearce M. & Billington R. (1991) Towards a case-based tool for aiding conceptual design problem solving. In Proceedings of DARPA Case-Based Reasoning Workshop 1991, Washington, D.C., 8-10 May 1991. San Mateo, California: Morgan Kaufmann.

Hammond K. J. (1989) Case-Based Planning: Viewing Planning As A Memory Task, Boston: Academic Press.

Keane M. (1987). On Retrieving Analogues When Solving Problems. The Quarterly Journal of Experimental Psychology. 39A, 29-41.

Kolodner J.L. (1988) Retrieving events from a case memory: A parallel implementation. In J. Kolodner (ed.) Proceedings of DARPA Case-Based Reasoning Workshop 1988, Clearwater Beach, Florida, 10-13 May 1988. San Mateo, California: Morgan Kaufmann.

Mostow J. (1989), Design by Derivational Analogy: Issues in the automated replay of design plans, Artificial Intelligence, 40, 119-184.

Owens C. (1988) Domain-independent prototype cases for planning. In J. Kolodner (ed.) Proceedings of DARPA Case-Based Reasoning Workshop 1989, Clearwater Beach, Florida, 10-13 May 1988. San Mateo, California: Morgan Kaufmann.

Smyth B., & Cunningham P. (1992) Déjà Vu: A Hierarchical Case-Based Reasoning System for Software Design. In B. Neumann (Ed.) Proceedings of 10th. European Conference on Artificial Intelligence. Vienna, Austria, 3-7 August 1992. Chicester: Wiley & Son.

Smyth, B. & Keane, M.T. (1992). Adaptation in Retrieval (Technical Report TCD-CS-92-34) Dublin: Trinity College Dublin, Computer Science Department.

Stanfill C., & Waltz D. (1986) Toward memory-based reasoning. Communications of the ACM, 29, 1213-1228.

Sycara K.P., & Navinchandra D. (1991) Index transformation techniques for facilitating creative use of multiple cases. In J.S. Gero & F. Sudweeks (Eds.) Workshop on Artificial Intelligence in Design at Twelfth International Joint Conference on Artificial Intelligence. Sydney, Australia, 25 August 1991, Sydney: University of Sydney.

Thagard P. & Holyoak K.J. (1989), Why indexing is the wrong way to think about analog retrieval. In Proceedings of DARPA Case-Based Reasoning Workshop 1989, Pensacola Beach, Florida, 31 May - 2 June 1989. San Mateo, California: Morgan Kaufmann.

Thagard P., Holyoak K.J., Nelson G. & Gochfeld D. (1990). Analog Retrieval by Constraint Satisfaction. Artificial Intelligence, 46, 259-310.

Visser W. (1991) Planning in routine design: some counterintuitive data from empirical studies. In J.S. Gero & F. Sudweeks (Eds.) Workshop on Artificial Intelligence in Design at Twelfth International Joint Conference on Artificial Intelligence. Sydney, Australia, 25 August 1991, Sydney: University of Sydney.

Waltz D.L. (1989) Is indexing used for Retrieval? In Proceedings of DARPA Case-Based Reasoning Workshop 1989, Pensacola Beach, Florida, 31 May - 2 June 1989. San Mateo, California: Morgan Kaufmann