Using Diagrams to Understand Diagrams: A Case–Based Approach to Diagrammatic Reasoning

Dale E. Fish and Robert McCartney
Department of Computer Science and Engineering
University of Connecticut
Storrs, CT 06269–3155
fish@eng2.uconn.edu robert@eng2.uconn.edu

Abstract

We are exploring the possibility of using case–based reasoning as an approach to diagrammatic reasoning. This paper describes a work in progress on a system that 'understands' a diagram of a problem situation by finding correspondences between it and similar diagrams in its case memory. The input to the system is a diagram of the problem situation consisting of some number of simple elements such as circles and lines. The goal of the system is to determine the significance of the diagram and its constituent elements by recognizing the structure of the diagram and structures within it. The approach is to use high level perception (recognition of instances of categories and relations) to build a representation of the problem situation (a diagram), and infer additional information by finding correspondences between this representation and stored cases. The output is the problem situation diagram annotated with descriptions of the elements and relationships between them and any inferences the system may be able to make along with their justifications.

Introduction

There are a number of problem domains where information is more easily and naturally represented using diagrams. While humans are adept at using diagrams, automated reasoning systems typically use pre-distilled codified representations. This means that in domains where diagrammatic representations are the norm, the diagrams must first be interpreted by humans, which (in extreme cases) results in either simplified representations where determinations of relevancy and saliency have already been made, or representations that (attempt to) include all the information present in the original diagrams. The former reduces the system's reasoning to a form of unification and the latter can be very difficult if not impossible given the density and complexity of information inherent even in simple diagrams. An automated system that reasons directly from diagrams both precludes the necessity of spoon feeding the system interpreted representations and retains the original information-rich diagrams. Using the actual diagrams in reasoning is especially important when there is limited domain knowledge (i.e. the first principles of a domain are unknown) or flexibility is required, both of which are characteristics of learning systems.

There has been considerable interest in the area of diagrammatic reasoning (DR) but little work using case-based reasoning (CBR) as an approach to DR. We think this may be a natural fit, reasoning with diagrams about diagrams. This work explores the possible advantages of using a store of case diagrams to interpret new diagrams.

This paper describes a work in progress on a system that 'understands' a diagram of a problem situation by finding correspondences between it and similar diagrams in its case memory. The input to the system is a diagram of the problem situation consisting of some number of simple elements such as circles and lines. The goal of the system is to determine the significance of the diagram and its constituent elements by recognizing the structure of the diagram and structures within it. The approach is to use high level perception (recognition of instances of categories and relations) to build a representation of the problem situation (a diagram), and infer additional information by finding correspondences between this representation and stored cases. The output is the problem situation diagram (or just problem diagram) annotated with descriptions of the elements and relationships between them and any inferences the system may be able to make along with their justifications.

In the next section, we state the problem we are addressing with this work and follow that with a discussion of the approach we are taking. The following section elaborates on the approach by providing an example. We then discuss some other work related to our approach, some problems to be addressed and possible extensions to our work, and we conclude with a brief discussion of the contribution we believe this work provides.

Problem Statement

We are interested in looking at the utility of using diagrams we know and understand to reason about new diagrams. In general terms, the question we are dealing with is can diagrams be useful in understanding other diagrams. For humans working in domains where information is routinely represented diagrammatically, the answer seems to be an obvious yes. Architects represent plans with diagrams and understand these representations because of their experience in working with them. The military uses diagrams to represent tactical scenarios, sports teams use diagrams to represent plays, and directors and actors use

diagrams to represent sets and stage movement. The usefulness of diagrammatic representations in these and other domains is the efficiency with which they convey information, and the efficiency with which the people involved understand the diagrams is due in large part to their experience in working with them.

We would like to develop a CBR system that uses diagrams of situations as cases and reasons from these cases about new diagrammatically represented problem situations. We are assuming that a problem situation can be represented using some number of simple pictorial elements such as basic geometric figures, and that the identity of these elements along with their location and orientation is given, sidestepping the issue of low level recognition. (What this really means is that the system is given instructions on how to draw the problem situation as opposed to giving it a bit map.) What these elements represent at some basic level may also be known given some domain, but the roles the individual elements play in a particular diagram must be inferred by recognizing the structures and relationships inherent in the diagram.

This seems to be a difficult problem. Take a simple relationship like grouping: if an element can be in at most one group, the number of possible groupings is given by the size of the power set of the elements. Even with a small number of elements, a small number of relationships between elements and/or groups, and a small number of possible categories to assign to the elements, the problem space gets large in a hurry. One role of the cases then is to focus attention, sort of like saying look for something like this around here. The structures discovered in the problem diagram may be mapped to the structures in the case or cases that prompted their discovery along with the corresponding elements, allowing inferences to be formed about the elements in the problem diagram (i.e. if this maps to that, then this is like that).

The domain we have chosen is the game of football which is a domain where diagrams are routinely used to represent plans called plays. These play diagrams show the initial locations of the players and their assignments during the play. Figure 1 shows an example of a play diagram. This domain can be quite complicated although the diagrams themselves are very simple, using a small set of symbols, mostly easily recognizable geometric shapes. The simplicity of these diagrams makes it a good domain for initial investigation (since we are more interested in a conceptual analysis of a diagram rather than low level recognition of complex elements within a diagram) while the complexity inherent in the domain should provide fertile ground for interesting extensions. We are of course very interested in proving claims of generality by using this approach on other domains.

For the present we are ignoring the action part of the diagrams (the lines showing player movement during the play) and concentrating on the initial positioning, or alignment, of the players. This is the problem situation the system is to understand. The perspective is that of the defense trying to recognize a novel offensive alignment in terms of offensive plays with which it is familiar. Figure 2 shows a diagram of a problem situation as it would be presented to the system. The diagram consists of 11 circles representing where the players on offense line up relative to one another for the given play.

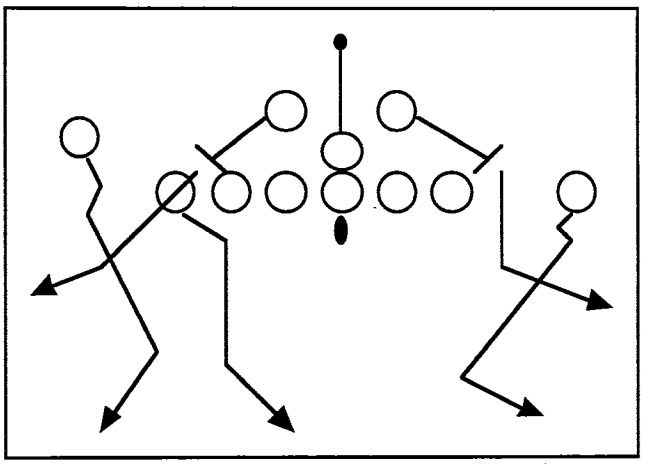

Figure 1: Sample play diagram showing player alignment and routes of receivers.

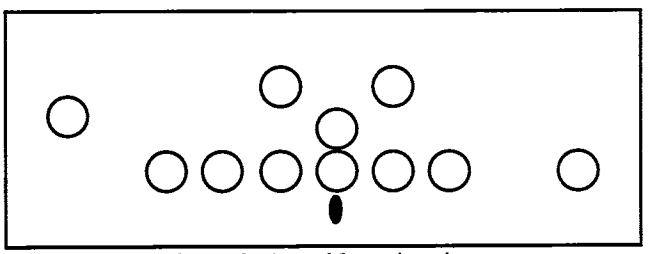

Figure 2: A problem situation.

What we would like to do is take a diagram such as this and make useful inferences to achieve some goal. The goal may be something as nebulous as 'understand the picture' or something as specific as 'identify the primary receiver'. For now, our concern is to be able to make good judgments about similarity between the problem situation and cases and find reasonable correspondences, or mappings, between the elements of two similar diagrams. We believe that this is useful since missing information about an element in the problem diagram may be inferred from the information known about its corresponding element in a similar case. The desired output of the system is the problem diagram annotated with labels identifying the types of the player elements and the discovered structures and relationships; the mappings are included as justifications for the inferences made and to indicate which groups in the problem diagram are inferred to share roles with which groups in the case diagram. In other words, the system's goal is to flesh out the diagram.

Approach

High Level Description

We are taking a CBR approach to DR. We see this as an interesting extension of the CBR paradigm, where cases and problem situations are represented diagrammatically, and as a natural approach to the tough DR problem. The basic goal of this work is to develop a general approach for automated diagram understanding. By 'general' we do not mean to preclude the need for domain knowledge, especially since we are using CBR, but instead that the approach itself is not wedded to a particular domain.

In any CBR system, cases are used to encapsulate specific information. This information may be specific in terms of a given domain, such as recipes in CHEF (Hammond, 1986) or in terms of more abstract goals, such as case adaptation in DIAL (Leake et al., 1997). The utilization of specific knowledge (experiences in human terms) is a strength of CBR. In a real sense, the ability to reason from cases means that a system's domain knowledge requirement is simplified; the system does not need knowledge of first principles which are at best difficult to apprehend and at worst lacking consensus. The other advantage of CBR is that it is naturally a machine learning paradigm; new cases are acquired and reasoning (hopefully) improves as the likelihood of having cases that more closely resemble the problem diagram increases.

Understanding the problem diagram involves finding a consistent set of reasonable structures. (We will use 'structures' as a generic term to refer to descriptions, relationships, and mappings: basically any type of additional information the system attaches to play diagrams.) Consistent simply means that their are no two contradictory structures. For example, an element cannot have two player type descriptions attached to it, or an element in the problem diagram cannot be mapped to more than one element in a particular case. Reasonable is more difficult. For our approach, reasonableness is a measure of the quality of the mappings between structures in the problem diagram and structures in a case (i.e. how much conceptual slippage is required to make a mapping fit). For example, mapping a group of two WIDE-RECEIVER elements to a group of two WIDE-RECEIVER elements is better than mapping a group of two WIDE-RECEIVER elements to a group of five LINEMAN elements.

Our system uses three types of knowledge: general knowledge, general domain knowledge, and specific domain experience. The first two are represented in the system's conceptual network which includes general concepts (e.g. spatial relationships such as BEHIND and NEXT-TO) and domain concepts (e.g. player types such as QUARTERBACK and WIDE-RECEIVER). A network representation is used to encode associations among concepts. These associations enable the system to focus more on structural similarity (structures involving related concepts) and avoid the restricting specificity of superficial similarity by allowing near concepts to map to one another. Specific domain experience is represented as a collection of cases in the case memory where cases are annotated diagrams. The conceptual network and case memory are discussed in more detail later.

An important aspect of the way our approach works is that the various types of processing are interleaved. This means that all types of structure building (describing elements, identifying relations between them, and mapping structures in the problem diagram to structures in the cases) go on simultaneously. There are no separate phases. The rationale behind this (mentioned in the problem statement) is that it is impractical and (we believe) unnecessary to completely describe a situation in order to make useful inferences about it. Better to take a quick look, see what types of things you find, look for more of the sorts of things you are successful at finding, and use the cases to help focus the attention on the problem diagram extensionally (i.e. does the problem diagram have something like *this* over in *that* area?). Another advantage of this approach is scaling: in domains with more concepts, more complex relations, and complicated diagrams, finding a complete consistent description of the problem diagram would be inefficient. This interleaving of recognition processes is an idea borrowed from Tabletop (French, 1995; Hofstadter, 1995) which is discussed briefly in the section on related work.

This interleaving is accomplished by encoding each type of recognition and structure building as a separate chunk, or method, which is added to a code queue. These methods may cause other methods to be added to the queue, either because they were successful (in which case similar methods would be added) or because they require some preprocessing (in which case a different kind of method would be added). These methods are chosen to run based on priority, where priority is a function of dynamic factors, most important of which are the activation levels of the particular concepts involved. All recognition methods involve concepts and when instances of concepts are recognized in the problem diagram, the corresponding nodes in the conceptual network receive some activation which also spreads to neighboring nodes. Methods in the code queue that recognize more activated concepts are chosen over methods whose corresponding nodes are less activated. To begin, the code queue is primed with several methods reflecting domain predispositions.

The system first applies general domain knowledge to the problem diagram in order to begin building descriptions of the elements of the diagram. This involves recognizing elements in terms of the concepts they represent and recognizing relationships between elements or other structures. The problem diagram is annotated with the structures as they are recognized and the concept nodes in the conceptual network representing the types of structures that have been recognized are activated. This activation has two effects: the system tries to recognize more of the activated types of structures (by giving priority to related

methods in the code queue), and cases containing instances of the same (or similar) structures are activated.

This phase of the process does not actually finish since there may be any number of possible structures, few of which would actually be correct and or useful. Instead, the system begins to consider cases that share similarity with the problem diagram in terms of the descriptions and structures as soon as any similarities are identified. The system tries to recognize correspondences between cases and the problem diagram. A correspondence is a structure that indicates a mapping between elements or structures. Each case that is considered will have its own set of correspondences with the problem diagram and for each case, there will be a maximal consistent set of correspondences where maximal is in terms of the quality of the mappings and the extent of the structures involved.

Recognizing new elements of the problem diagram brings new cases to the foreground and causes new structures to be built. Forming correspondences causes the system to look for other elements or structures in order to form similar correspondences.

Cases are ranked on the basis of their maximal consistent sets of correspondences with the problem diagram. These sets change as new structures and correspondences are formed which means cases come in and out of favor. At some point, the system becomes more stable; fewer new structures are found and the ranking of the cases remains the same. When this stability reaches some threshold, the system stops. The 'answer' is the current description of the problem diagram and its correspondences with the highest rated case.

The Conceptual Network

The conceptual network consists of nodes which represent the 'center' of particular concepts connected to one another via directional links which represent the relationships between concepts. There are two main types of links: the ISA and HAS-MEMBER links which represent hierarchical class relationships, and LABELED links which represent relationships described by other concepts in the conceptual network. For example, the TIGHT-END node representing a specific class of player is connected to the ELIGIBLE-RECEIVER node representing a more general class of player by an ISA link, and there is a HAS-MEMBER link connecting the ELIGIBLE-RECEIVER node to the TIGHT-END node; the nodes for the concepts ON-LEFT and ON-RIGHT are connected by a LABELED link where the label is the OPPOSITE node. In this way, a particular concept is represented by the node for that concept and to a lesser extent its neighboring nodes.

The conceptual network serves multiple purposes. The first purpose it serves is providing the methods for recognizing structures in the problem diagram. Each concept node contains a method or methods for recognizing an instance of the concept it represents. These methods may be entirely self contained or may involve the recognition methods of other concepts. The recognition methods for general knowledge concepts are often self contained. For example, the spatial relation concept NEXT-TO has a method for recognizing if two elements are adjacent. Domain knowledge concepts always involve other concepts. For example, recognizing the QUARTERBACK requires recognizing the player element BEHIND the CENTER, where BEHIND and CENTER are concept nodes with their own recognition methods (BEHIND is a spatial relation and CENTER is a player type) . If the system tries to identify the QUARTERBACK, it scans the descriptions of the player objects to see if the CENTER has been identified, meaning that a player element description has been augmented by the system identifying it as the CENTER. If the CENTER has been recognized, then the method for recognizing the BEHIND relationship is used to identify the QUARTERBACK. If the CENTER has not been identified, then the method for recognizing it is added to the code queue along with another copy of the method for recognizing the QUARTERBACK. This is one way that concepts make use of other concepts.

Given a network sort of representation, with related concepts as neighbors, we find a spreading activation approach can be used for a number of purposes. As mentioned in the high level description, nodes are activated by successful recognition methods. The links between nodes provide a measure of the conceptual distance between concepts, the shorter the distance the more similar the concepts. Activation 'flows' along these links so that near nodes receive some fraction of the activation level of the source node. The amount of activation that spreads is a function of the distance between the nodes. For example, if a GROUP structure of THREE elements is recognized, some activation is added to the GROUP concept and THREE concept nodes, and this activation will spread to the TWO and FOUR concept nodes, and so on. Activation levels are an indication of the importance of specific concepts in the problem diagram in terms of what has been discovered up to that point. When the activation level of a node rises above some threshold, a recognition method for that concept is added to the code queue. In this way the spreading activation causes recognition methods similar to successful recognition methods to be added to the code queue. Since we are interested in *similar* things and not just *specific* things, spreading activation is used so that similar concepts are included.

The next purpose the conceptual network serves is that of a kind of index into case memory. This also takes advantage of the spreading activation. Each concept node is connected to occurrences of itself in the individual cases in case memory. Activating concept nodes results in activating cases. The activation level of a particular concept node is in a sense an indication of the relevancy of that concept to the problem diagram. Cases that involve activated concepts are relevant to the problem diagram.

The conceptual network also provides a basis for mapping similar concepts onto one another. When looking for similar cases or forming correspondences between situations, we do not want to restrict the indexing or mapping to exact matches; this would put too much

emphasis on superficial similarity. Instead, we would like to allow for slippage, allowing a particular concept to 'remind' the system of similar concepts and similar concepts to map to each other. Of course, everything else being equal, it seems logical to prefer correspondences involving more similar concepts over less similar concepts. The conceptual distance between concepts provides a mechanism for preferring correspondences between closer matches.

Case Memory

Case memory consists of a number of situations from the problem domain. A case includes a diagram (location and orientation of each player element) and whatever additional structures the system may have formed or discovered during the same process of understanding that the problem diagram is subjected to. These descriptions are not assumed to be complete because the system does not attempt to discover all possible information or form all possible structures when analyzing a situation. Figure 3 shows a sample case diagram of a passing play. The player elements are annotated with labels for the types of players they represent (football is a game of specialization). For example, the element that represents the QUARTERBACK is annotated with the label QB. These labels come from the corresponding concepts in the conceptual network. There are also 4 group structures shown in the case diagram. These are the rectangles marked G1–G4. The primary receiver for this particular play is the SPLIT-END (the element marked SE) and this is indicated by the bold circle. There would be other structures as well, such as the spatial relationships between the groups and between some elements as well as additional information describing group compositions. These are omitted from the figure for simplicity.

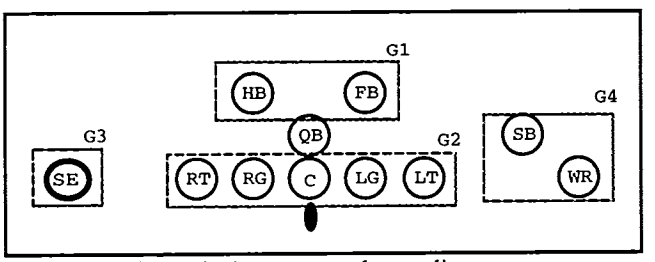

Figure 3: An annotated case diagram.

In considering a case, the system opens a window showing the problem diagram (upper half of the window) and the particular case (lower half of the window). The system chooses structures in the case and adds recognition methods for that type of structure to the code queue. These methods may include an argument for a specific area of the problem diagram. For example, if there is a GROUP structure in the lower right of the case (e.g. G4 in Figure 3), the system may add a method to the code queue for recognizing a group in the lower right of the problem diagram. The system maps elements and structures it finds in the problem diagram to elements and structures in the case. The same element or structure cannot participate in more than one correspondence, but for any element or structure, there may be several reasonable mappings, so there must be a means for preferring one correspondence over another. The quality of correspondences are evaluated according to certain predispositions (e.g. preferring to map groups similar in number and element composition). These mappings work with the conceptual network as a focusing mechanism for the system. The system looks for instances of highly activated concepts and structures involving highly activated relational concepts in the problem diagram. Concept nodes receive additional activation when their concepts participate in correspondences between cases and the problem diagram, so attention on the problem diagram is in a sense directed by its relations to cases.

Cases are scored in order to make a determination as to which cases are most relevant to the problem diagram. The score of a case is a function of the activation levels of the concepts in the case. Activated concepts that participate in structures within a case (as opposed to concepts that occur in isolation) are weighted in order to reflect the system's preference for structural similarity versus superficial similarity.

A Sample Problem

Given a problem situation described by the diagram in Figure 2, the system first attempts to recognize what concepts are represented by the elements of the diagram. The rules of the game of football and certain standardizations regarding the way the game is played are used to identify the classes of players represented by the circles in the problem diagram by recognizing certain spatial relationships between the elements in the diagram. For example, recognizing which circle represents the QUARTERBACK involves identifying which circle stands in the relation BEHIND to the circle representing the CENTER, which in turn is recognized by identifying which circle stands in the relation BEHIND to the element representing the BALL which is recognized by its unique shape, location, etc. in the diagram. As the system recognizes elements, it builds descriptions of them using concepts from the conceptual network. This in turn activates the corresponding concept nodes in the network which has the effect of activating cases in case memory containing occurrences of the activated concepts. The spreading activation in the conceptual network means that cases containing similar concepts will also be activated.

At the same time, the system identifies higher level structures in the problem diagram. These structures consist of groups of elements and relationships between elements and other structures. The system may try a number of different structures, some of which may be inconsistent with one another. For example, in forming groups, the system is predisposed to favor groups containing similar elements (i.e. elements sharing a superclass) and groups containing neighboring elements. Since elements are not

allowed to be in more than one group (unless one group is a proper subset of the other), including one grouping in the description of the problem diagram may prevent the inclusion of another. The system tries to find a consistent set of structures, favoring the set that has the most structure involving the concepts with the most activation.

As the system proceeds with building a description of the problem diagram, it begins to consider cases, preferring cases with similar structures found in the problem diagram. Considering cases means finding correspondences between structures or elements in the case and the problem diagram. These processes are interleaved. This means that in considering a certain case, if a number of good correspondences have been identified but some structure in the case has no corresponding structure in the problem diagram, the system will look for such a structure. The idea is to avoid trying to completely describing the problem diagram and instead let partial mappings with similar cases indicate what to look for.

Figures 4 and 5 are possible results of the system, showing correspondences between the problem diagram (the upper half of each figure) and two cases. For each of the two cases, other correspondences are possible but these may be preferred by the system for a variety of reasons. For example, in Figure 4 the correspondences marked M4 and M5 suggest a degree of diagonal symmetry between the diagrams. The system prefers this mapping over the mirror symmetry mapping (G1 to G3 and G4 to G4) because the correspondences as shown involve more similar groups in terms of the number of elements.

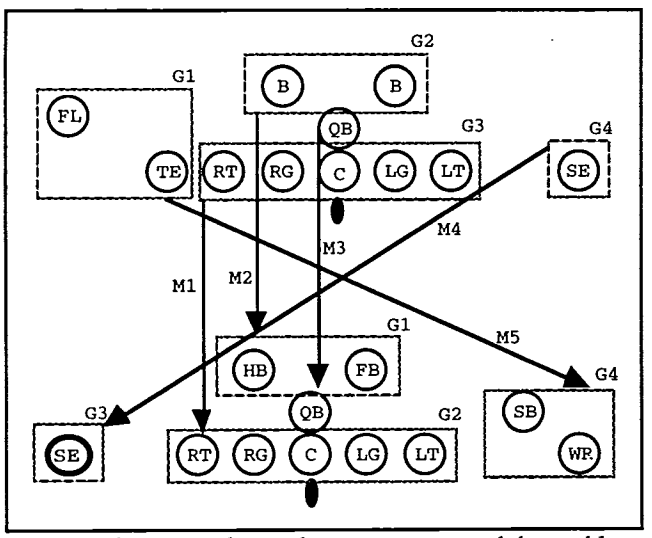

Figure 4: Correspondences between a case and the problem diagram.

As mentioned, building descriptions and structures in the problem diagram is interleaved with forming correspondences with similar cases. The correspondences help focus the system's attention by suggesting where to look and what to look for in the problem diagram. For example, if the grouping G1 in the problem diagram in Figure 4 had not been recognized (and therefore correspondence M5 had not been found), the existence of the unmapped group G4 in the case prompts the system to look for a group of two elements of type SLOT-BACK and WIDE-RECEIVER. Similarly, the correspondence marked M4 prompts the system to look for such a group on the opposite side of the formation by activating the DIAGONAL-SYMMETRY concept which is a particular type of mapping. The group G1 is then formed since it has the right number of elements, the types share a superclass with the elements of group G4 in the case, and it is where it should be. The correspondence M5 can then be recognized.

The system considers multiple cases simultaneously, concentrating on the most promising cases. Rating a case involves scoring the structures within the case and its correspondences with the problem diagram. Structures involving more concepts with more activation are scored higher than structures with fewer concepts having less activation, and correspondences involving structures with higher scores are scored higher than correspondences involving structures with lower scores or correspondences between isolated elements. Also, concepts participating in structures and correspondences receive additional activation. This helps bias the system toward structural similarity as opposed to superficial similarity.

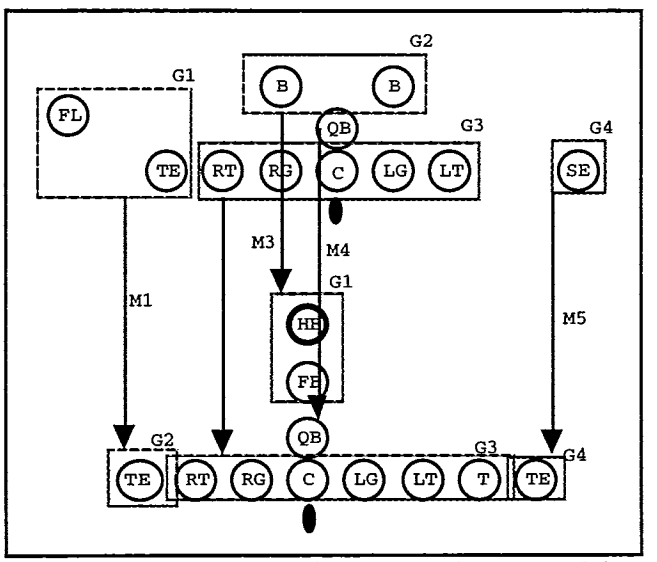

Figure 5: Correspondences between another case and the problem diagram.

The mapping in Figure 4 scores higher than the one in Figure 5. All the groups mapped to each other in Figure 4 have the same number and type of elements. Also there are five elements in the problem diagram that are recognized as representing concepts that are subclasses of the ELIGIBLE-RECEIVER concept. Thus the node for this concept is highly activated which contributes to the high score for the case in Figure 4 since it contains five instances of concepts with ISA relationships to ELIGIBLE-RECEIVER.

Eventually the system begins to stabilize. This means

that fewer new descriptions, relationships, and correspondences are found, the maximal consistent set of descriptions in the problem diagram remains the same, and the ordering of the cases remains the same. When this stabilization reaches some threshold (or activity falls below some threshold) the system is done. The 'answer' is the annotated diagram of the problem diagram and the correspondences with the highest rated case. The descriptions for the problem diagram may include inferences drawn from its correspondences with the best case. For example, the case in Figure 4 may include a description of the play as a PASS-PLAY (versus a RUNNING-PLAY) so the problem diagram would likewise be inferred to represent a PASS-PLAY. Similarly, the element in the case in Figure 4 marked SE (SPLIT-END) may include in its description the designation PRIMARY-RECEIVER, meaning the QUARTERBACK will look to throw the pass first to him, so the element in the problem diagram mapping to the SPLIT-END (also marked SE) is inferred to be the PRIMARY-RECEIVER in the problem diagram.

Related Work

Tabletop

Tabletop is a system developed by French (French, 1995) that models analogical reasoning. It is presented with a source and a target and attempts to identify analogous structures between them. Tabletop's problem is a table set (sometimes haphazardly) for two; an object on one side of the table is 'touched' and Tabletop must decide what the analogous object on the other side of the table would be. It is included here mainly because we borrowed some important aspects of it for our approach, namely using a conceptual network to model associations and focus system resources on activated concepts, and the idea of interleaving building representations and forming correspondences. It is also included here because, although it is not purported to be a diagrammatic reasoning system, it could easily be one, and it is a motivation for our work. We originally thought of our work as an attempt to use CBR to do the sort of high-level perception that Tabletop does, but it soon became obvious that what we were doing was DR. Tabletop's knowledge is represented in its conceptual network or 'slipnet' and it makes no use of a case memory.

Case-Based Reasoning with Diagrams

POLYA (McDougal & Hammond, 1995) is a CBR system that constructs proofs for high school geometry problems. The diagram of the problem POLYA is to solve is used to provide features which are used as indices to plans in case memory. POLYA interleaves the execution of two types of plans: plans that extract more features from the diagram (search plans) and plans that actually write the proof. This is a very interesting piece of work and shares a lot of similarities with our approach, especially the idea that features discovered in the problem diagram are used to prompt the activation of methods to look for certain other features, instead of trying to describe the diagram completely. An important difference is that the search plans are not diagrams. They are methods for finding more features based on the features that have already been identified. Using actual diagrams similar to the problem diagrams to guide processing may make our approach better suited to weak theory domains and it may be more general. But another important difference is that POLYA is working, so we will have to wait to see if these claims are valid.

Anderson (Anderson & McCartney, 1996) describes an efficient use of stored diagrams applicable to a number of domains. A syntax and semantics of inter-diagrammatic reasoning is developed along with operators that allow retrieval of cases using diagrammatic matching. Our approach is different in that retrieval is based on similarity of concepts and structures identified in the diagrams (i.e. what the diagrams represent) as opposed to similarity of planar tessellations.

COACH

COACH (Collins, 1989) is a CBR system that works in the football domain. The emphasis of the work is plan creation. COACH generates new plays by recognizing the bug in a failed play and applying abstract strategies for modifying plans which are indexed by generalized descriptions of plan failures. Cases in COACH are primarily a starting point; planning from scratch is expensive and difficult in weak-theory domains. The differences between COACH and our approach are the system goals (planning versus recognition) and COACH does not use diagrams.

Future Work

We are in the process of implementing a system using the approach described in this paper that applies this approach to limited aspects of the football domain. We have implemented small test versions of the main components of the system but have not yet put the pieces together. The conceptual network consists of a player type hierarchy and several relational concepts. A handful of the recognition methods are implemented and limited annotating of the problem diagram works. The case memory consists of only a few test cases with relatively little similarity between them.

We expect that there are a number of problems that will have to be addressed if our approach is to work. First is how to choose initially which cases to apply to the problem diagram. The problem here is that *all* football play diagrams share substantial superficial similarity, and the main reason behind using cases for recognition was to avoid the difficulty of trying to analyze the deeper structure of the problem diagram without the focus cases provide.

This problem will be exacerbated with a large case base. One approach is to use abstractions of the top level structures in the cases and retrieve cases based on what types of similar structures are found in the problem diagram. Another approach might be that it may not matter that much which cases are initially applied if cases are indexed by the specific structures they contain. For example, a group may be found in response to some case be activated; as the internal description of that group is fleshed out, a case or cases with a similarly described group may be activated.

It will still be difficult to determine exactly how to prefer certain cases over others. There are competing factors such as the number of structures mapped, the quality of the individual mappings, and inter-structure relationships (e.g. symmetry). How we do this will depend on actual results, giving us a better idea of which set of metrics are useful and how they should be weighted.

Another difficulty is knowing when to stop looking. How do you know when the best set of correspondences you have at some point is as good as it is going to get, or that further recognizing and mapping will not yield any significant improvements. One way to handle this may be to think of the approach as an anytime algorithm and the answer is the case-to-problem diagram mappings in order of ranking. This of course is inadequate if the goal is something specific like 'identify the primary receiver' since the primary receiver elements in the various cases being considered may yet to mapped to any element in the problem diagram. The approach we have in mind is to have the system to quit when activity falls below some threshold (e.g. when the frequency of finding new structures drops off) and the rankings have stabilized. When we get some results we will be able to judge how effective this approach is.

There is also the problem of context. Each case that is considered will involve different concepts or similar concepts to different degrees. For example, considering one case may prompt activation of the DIAGONAL-SYMMETRY concept while another may prompt activation of the MIRROR-SYMMETRY concept. We would like to be able to switch contexts without any adverse influence. One approach might be for each case to have its own private concept activation levels. However, it seems that it would still be desirable to be able to have cases respond in some degree to global activation levels so that successes may be repeated. This is a very interesting problem. Note for example that this context switching can occur with a single case: the elements of a group in the problem diagram may map diagonally to the elements of the corresponding group in the case, while the rest of the diagram maps straight on.

An interesting extension to our work would be to consider the dynamic aspects represented in the case diagrams. Since real play diagrams (e.g. Figure 1) represent plans and imply movement of the elements, it would be interesting to consider this movement in matching cases to the problem diagram. For example, differences in the alignment of the players may mean that the route assignment of a player in the case does not transfer well to its corresponding player in the problem diagram, which may mean that the rating of that correspondence should suffer.

The evaluation of the effectiveness of our approach will require that we also test other domains. One interesting extension that would broaden the range of applicable domains would be to consider sequences of diagrams representing movement as the problem situation. Cases would also be sequences where each case represents an episode. The problem would be to use the cases to predict subsequent diagrams in the problem sequence and modify predictions dynamically.

Conclusions

This paper describes an approach to diagrammatic reasoning that uses high level perception to recognize concepts and relationships in diagrams and make inferences by recognizing correspondences with similar diagrams. This work is largely unimplemented at present. Evaluation of the effectiveness of our approach will follow an analysis of the performance of the working system. An assessment of its utility will require application of our approach to a variety of domains.

References

Anderson, M. & McCartney, R. (1996). Diagrammatic Reasoning and Cases. 13th National Conference on Artificial Intelligence, August, 1996, Portland, Oregon. AAAI.

Anderson, M. & McCartney, R. (1995). Inter-Diagrammatic Reasoning. 14th International Joint Conference on Artificial Intelligence, August, 1995, Montreal, Canada. Morgan Kaufmann Publishers, Inc. 878–884.

A. F. C Association. (1995). *Football Coaching Strategies*, Human Kinetics. Champaign, Illinois

Chandrasekaran, B., Glasgow, J. & Narayanan, N. H. eds. (1995). *Diagrammatic Reasoning: Cognitive and Computational Perspectives*, AAAI Press. Menlo Park, California.

Chandrasekaran, B., Narayanan, N. H. & Iwasaki, Y. (1993). Reasoning With Diagrammatic Representations—A Report on the Spring Symposium. In *AI Magazine*, Vol. 14, pp. 49–56.

Collins, G. C. (1989). Plan Creation. In *Inside Case-Based Reasoning*, eds. C. K. Riesbeck & R. C. Schank, Lawrence Erlbaum Associates, Publishers. Hillsdale, N.J., pp. 249–290.

Dreayer, B. (1994). *Teach Me Sports—Football.* General Publishing Group, Inc., Santa Monica, California.

Estes, W. K. (1994). *Classification and Cognition*. Oxford University Press, New York, New York.

French, R. M. (1995). *The Subtlety of Sameness—A Theory and Computer Model of Analogy-Making.* MIT Press, Cambridge, Massachusetts.

Gentner, D. & Stevens, A. L. eds. (1983). *Mental Models.* Lawrence Erlbaum Associates, Inc. Hillsdale, New Jersey.

Goel, V. (1996). *Sketches of Thought.* MIT Press, Cambridge, Massachusetts.

Hammond, K. J. (1986). CHEF: A Model of Case-Based Planning. Proceedings of AAAI-86. AAAI Press/MIT Press.

Hawkes, D. D. (1995). *Football's Best Offensive Playbook,* Human Kinetics. Champaign, Illinois.

Hofstadter, D. (1995). *Fluid Concepts & Creative Analogies—Computer Models of the Fundamental Mechanisms of Thought.* BasicBooks, New York, New York.

Kolodner, J. (1993). *Case-Based Reasoning*. Morgan Kaufmann Publishers, Inc., San Mateo, California.

Kolodner, J. L. (1984). *Retrieval and Organizational Strategies in Conceptual Memory: A Computer Model.* Lawrence Erlbaum Associates, Limited, Hillsdale, New Jersey.

Kolodner, J. L. & Riesbeck, C. K. eds. (1986). *Experience, Memory, and Reasoning.* Lawrence Erlbaum Associates, Limited. Hillsdale, New Jersey.

Leake, D. B., Kinley, A. & Wilson, D. (1997). Case-Based Similarity Assessment: Estimating Adaptability from Experience. Proceedings of the Fourteenth National Conference on Artificial Intelligence, July 1997, Providence, R.I. AAAI Press/MIT Press. 674–679.

Lockhead, G. R. (1992). On Identifying Things: A Case for Context. In *Percepts, Concepts and Categories: The Representation and Processing of Information*, ed. B. Burns, Elsevier Science Publishers B. V. Amsterdam, The Netherlands, pp. 381-410.

McCartney, R. (1993). Episodic Cases and Real-time Performance in a Case-Based Planning System. *Expert Systems with Applications*, 6, pp. 9–22.

McDougal, T. F. & Hammond, K. J. (1995). Using Diagrammatic Features to Index Plans for Geometry Theorem-Proving. In *Diagrammatic Reasoning: Cognitive and Computational Perspectives*, eds. B. Chandrasekaran, J. Glasgow & N. H. Narayanan, AAAI Press. Menlo Park, California, pp. 691–709.

Millikan, R. G. (1984). *Language, Thought, and Other Biological Categories*. MIT Press, Cambridge, Massachusetts.

Mitchell, M. (1993). *Analogy-Making as Perception: A Computer Model.* MIT Press, Cambridge, Massachusetts.

Nahinsky, I. D. (1992). Episodic Components of Concept Learning and Representation. In *Percepts, Concepts and Categories: The Representation and Processing of Information*, ed. B. Burns, Elsevier Science Publishers B. V. Amsterdam, The Netherlands, pp. 381-410.

Novick, L. R. (1988). Analogical transfer: Processes and Individual Differences. In *Analogical Reasoning: Perspectives of Artificial Intelligence, Cognitive Science, and Philosophy*, ed. D. H. Helman, Kluwer Academic Publishers. Dordrecht, The Netherlands, pp. 125–145.