Learning refinements to Spatial Template Models based on false-positive and false-negative examples DRAFT ONLY

Steven Woods
University of Waterloo
Student Id 89800462

Abstract

Spatial templates are used to model a spatial concept held in the mind of a domain expert or set of domain experts. These templates consist of objects of varying types that share a set of spatial relationships or constraints that have been recorded as representing the concept in question. These templates define a space of instances that "fit" the template constraints. A situation is a set of objects of varying type and attributes arranged in a finite area. Search algorithms have been applied previously to attempt to locate one or all mappings from a particular situation to a particular template (or set of templates) such that the template constraints are satisfied by the mapped situation objects. In addition, previous work has focused on developing expert-interactive search strategies and representation models for identifying these template instances.

Due to inaccurately specified templates or error in the specification of a situation, it is possible that a complete search method may locate a mapping from a template to a situation that a domain expert may indicate is *not* representative of the concept the template is meant to model. Similarly, it is possible that a complete search method my fail to locate a mapping that a domain expert identifies as a representative instance of the concept the template is modeling. These cases are known as *false-positive* and *false-negative* respectively. Based on these identified error cases, we wish to modify our template model to more accurately represent the ideal spatial concept.

One previously developed approach to dealing with uncertain template definitions and "noisy" situational data is to represent the template as a hierarchy of templates, where an "abstract" template captures the "key factors" of a spatial concept, and subsequently more specific templates capture more, possibly less integral, detail of the concept. Use can be made of this hierarchy in identifying possible template matches in situations even when the mapping between situation and most-specific template is not present. Here we are interested in learning error cases not only within a single level model, but also in determining how refinement at a particular level should or should not propagate to other levels of a template hierarchy.

In this work I will discuss the role and representation of spatial templates as approximations of an ideal spatial concept, as well as outline how the use of expert-identified errors as catalysts for model refinement corresponds to learning improved spatial templates. This work reports ongoing research towards both interactively and automatically learning revisions of spatial templates and in refining spatial templates represented in constraint graphs according to rules for constructing abstract representations of these graphs which have been applied in various domains, and reported in earlier work.

1 The problem

1.1 Introduction

> *Intelligence* means getting better over time ... Real AI means a machine that learns ... AI depends on computers that have real knowledge in them. Thus, the crux of AI is in the representation of this knowledge, the content-based indexing and modification of this knowledge through the exercise of this knowledge (Schank 1992, p. 42,p. 47).

This work is about learning. Specifically it is about learning how to improve models through failures of a model to "fit" or classify new, previously unseen examples. In this section I will introduce a specific domain, Spatial Template Recognition (STR), in which I will be discussing model revision learning, and position the learning aspects of learning model revisions in the broader context of the specific domain problem. In later sections I will further examine what information needs to be learned, and discuss possible methods of integrating this new knowledge into useful and more accurate models. This paper is intended as an exploration of the STR problem with regards to representation and representation revision using machine learning. I will examine current work in the domain itself, approaches to model representation using abstraction, and learning approaches in several related areas.

STR is the problem of finding an instance of some predefined configuration of objects that may be deployed amongst a given situation, or set of object instances. An example of a spatial template in a two-dimensional molecular domain would be a "map" of a typical molecule of some type that describes the molecule in terms of the relative positions, types and orientations of the various atoms of the molecule. An example of a situation might be some two-dimensional sample containing many instances of different types of molecules, and the question formed by STR would be: "Is there an instance of this molecule in this solution?".

Other domains for application of STR are more dynamic, for instance, consider a group of objects on a gameboard being deployed according to some opponent strategy. During a game, a player often would like to identify the plan of his/her opponent. While this determination can depend heavily on the perceived context of the game at a particular moment, the personality characteristics, experience and goals of ones opponent and upon many other factors, a "quick" or "trigger" heuristic that suggests to us the intent of our opponent can be our identification of some representative or typical pattern in either his movements or static board positioning. Basically, given that we understand certain "key" board configurations that our opponent may wish to utilize under certain circumstances, it is possible that we may recognize a particular board configuration and infer that its existence is motivated by a particular plan in our opponent's mind[1]. Thus, this kind of limited pattern recognition can be an integral component of a larger plan recognition system.

As a specific example of this opponent inference, in military deployments, units are positioned according to specific goals and opposition deployments. The "group" or collective deployment patterns are consequently representative of these goals. Typical configurations include such group and activity combinations as "tank regiment in advance", "rifle regiment in attack". So, if we could take a "picture" of the current deployment positions, filter out any "noise" in the form of overlapping or confusing deployments, and identify the elements involved in a particular maneuver, we have essentially completed one part of an intelligence analyst's task in recognition of a local plan.

[1]Of course for the moment ignoring the obvious misdirection possibilities

1.2 Model specifications

In many of the domains of STR, experts use expertise built up over a long period of time to construct a "typical" model of each of many templates. Each of these models may be physically represented or held mentally by individual experts. To varying degrees these models cover a breadth of example instances in their domain, and essentially capture an elaborated domain *concept* as is described formally in the literature of machine learning, one instance of which is Anthony & Biggs (1992). Typically, one standardized generalization is accepted or agreed upon as canonical, and is often referred to as if it were a "definition" of a particular concept for other more naive initiates to the domain. Often these canonical definitions include specific details of the variation allowed within the concept. For instance, a concept in the military domain might define a deployment pattern composed of individual units each related to some subset of the others according to spatial constraints. These constraints could be *rigid* such as "The lead unit must be *ahead of* the trailing unit", or *flexible* as is the case in "lead unit A must be between 5 and 10 km of lead unit B". The later range-constraint is referred to as flexible in this context since it has its tolerance encoded directly. Of course this flexible constraint is still "hard" in the sense that anything (however slightly) outside 10 km or inside 5 km will result in a failed constraint. These arbitrary cut-off points rarely reflect the real-world situation, and we will see subsequently how to handle these more smoothly using abstracted concept definitions.

1.3 Learning as model revision

Given a concept or template definition including both *rigid* and *flexible* constraints, we may attempt to systematically (either manually or automatically) identify these template instances in a specific situation by locating a set of object instances that satisfies all of our template constraints[2]. If a particular template concept "exceeds" in some sense the "true" modeled object concept, then some positive matches may be made which would be classified in the negative by some "oracle" or expert. If the concept expression is unduly restrictive, then some instance groupings that the expert would classify as positive are identified negatively based on the model. The role learning may play in such a framework is to integrate these misclassifications identified by an expert into the current concept model so that these and hopefully other unseen examples are classified correctly. Essentially we are interested in revising a given model to more accurately reflect the expert's belief about the concept in question. This model revision will be the first half of the

[2]Unfortunately, there are an exponential number of possible sets of object instance mappings to template slots.

focus of this paper.

1.4 Abstracting models

The inaccuracy of expert knowledge modeling has been a concern in the design of most expert systems, and the problems of spatial template inaccuracy have been discussed briefly in the work of both Woods (1993*a*) and Lapointe & Prouix (1994). In Lapointe & Prouix (1994), the problem is addressed by matching templates to situations probabilistically based on the existing template definition, thus hopefully avoiding the "model too restrictive" failures by at least classifying the "missed" case as possible in some measure, and avoiding the "model too accepting" problem of too many matches by some ordering based on a measure of the template-situation fit. In contrast, Woods (1993*a*) proposes a template concept representation model where the spatial templates are described not by a single constraint based description, but rather by a connected hierarchy of concept models. This hierarchy defines a concept at ordered levels of specificity, much as people might hierarchically define objects within classes. For instance, the chair on which you are sitting might be considered most abstractly a member of some class (say) *chair*, and more specifically as a member of the a subclass (say) *office chair*, and then most specifically (say) as a member of the subsubclass *computer office chair*. Within this framework we have three levels of expression of the concept "your chair", each conceivably more specific than its predecessor. The intent here is that in absence of complete context, we might benefit by first trying to identify your perch as a "chair" before jumping ahead and trying "office chair". We may consider that the essential "chairness" of the concept is encoded in the most abstract model, and the additional "officeness" is then mapped from chair to office-chair. Thus, we first identify the chairness of an object and then attempt to refine this further according to possible specified mapping(s) in a particular hierarchical concept definition.

This abstracted concept model maps well to the approach of a human expert in the spatial template identification domain. This expert tends to look for key (or obvious) features of a template (like four legs of a chair, for instance) and then attempts to resolve additional detail if possible, essentially "zooming in" with a conceptual magnifying glass. In this way, the expert could be said to be forming an abstract model in their mind of what a "match" looks like in terms of carefully but loosely defined spatial relationships and other key relationships among the objects.

1.5 Inaccuracy in representation and perception

Some of our earlier work (Woods 1993*a*, Woods 1993*b*) has focused on attempting to model these templates in consultation with real experts in an effort to construct an automated tool to assist in the recognition of these templates in object sets that are often very large, and which may quite possibly be full of error and inaccuracy with respect to object instance position, orientation, or other vital characteristics. Thus we see that the STR problem faces inaccuracy at both ends: in the modeling of a particular template concept, and in the perception of the object instance world in which we wish to identify some concept model.

Using a hierarchically specified template concept, a template match may be made at a very abstract level based on only "key" or obvious features. This parallels the concept of viewing an object at a distance where only the outline or "largest" or most "vivid" details stand out. Essentially we get an idea of what an object is at this distance, and then move closer for more detail and refine our belief somewhat. One can imagine in this way trying to analyze a large situation with many different template instances, and examining the situation by focusing in on specific details when examining one possible match, and then essentially taking a step back to view the whole situation at a higher level. This process of "zooming in" to resolve specific objects from the larger palette, and "zooming out" for the larger perspective follows the divide and conquer method that has long been suggested as a sensible way to attack many problems where the complexity of the whole task seems insurmountable. More will be said about this approach later in this paper.

1.6 Learning and hierarchies

Model revision was discussed earlier in this section, but now we are faced with an even more difficult problem, that of learning revision to a hierarchical model. Misclassifications may result in model revisions at a certain abstract level that may (or may not) need to propagate to other levels of the hierarchical concept representation. The examination of some of the various propagation possibilities will form the crux of this paper.

2 Spatial template representation at a single level

2.1 Template models and constraint graphs

A typical model of an expert's beliefs about a particular spatial template as outlined in Section 1.2 may be modeled as a graph of constraints amongst the elements or the template. For instance, a molecule might be composed of atoms which obey positional constraints amongst themselves. The "structure" imposed by these constraints is the essential "templateness" of the definition. Figure 1 details a simple template capture in a constraint graph consisting of three nodes, 3 constraints between pairs of nodes, and four constraints on various nodes. The constraints on a single node represent a constraint that a node or slot maintain a particular attribute: in this case "type" is

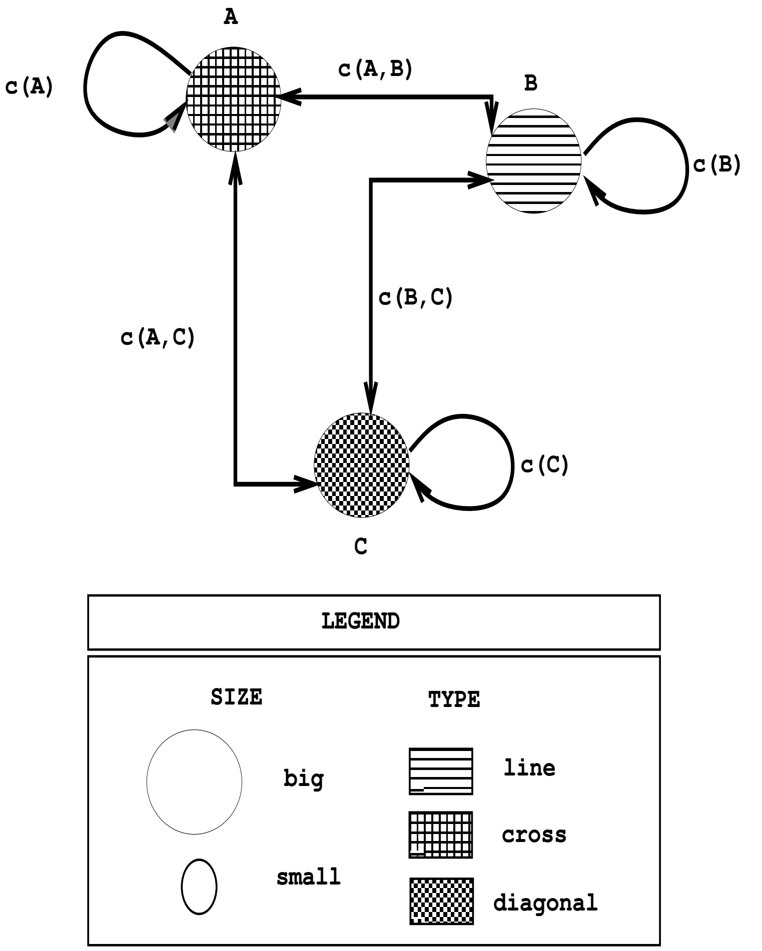

Figure 1: A simple template definition via a constraint graph.

represented by one of three possible colorings of the nodes, and another attribute or attribute set (including for example, "size") might be represented by a reflexive arc on each node. The spatial constraints of the template are represented as constraints among the nodes, for instance, C(A,B) might specify that A and B must be between 1 and 2 km apart perhaps. These constraints need not be binary, and may involve more complex relationships about relative positions. A legend in the figure identifies clearly the possible attribute values for "size" and "type".

2.2 Template instance identification

Given such a template representation we wish now to attempt to locate instances in a given situation, such as that portrayed in Figure 2. In this situation, we have fourteen objects, each either big or small in size, and each "typed" in one of three ways, just as our template definition earlier was typed. Now, if we had specified precisely the constraints C(A,B), C(B,C) and C(C,A) of Figure 1, we could have attempted to locate an assignment of situation objects to template nodes that satisfied the constraint set. Each such satisfying assignment would have constituted a solution to our template model definition. For instance, if we took the constraints in the template of Figure 1 as representing physical distance as drawn, then we could observe that one solution "mapping" or assignment from template node to situation object would be: $(A,B,C) \Rightarrow (4,12,9)$. While the problem of locating all of these mappings is NP-complete[3], the complexity of this problem does not necessarily indicate that in certain domains a search-based approach will not suffice. For a detailed discussion of the search and performance aspects of solving this and other similar Constraint Satisfaction Problems (CSPs), the reader is referred our earlier work in Woods (1993*a*) and that of Prosser (1993) and van Run (1994). A discussion of the complexity of this problem in relation to some other related "hard" problems such as graph and subgraph isomorphism may be found in Woods (1994).

2.3 Template instance misclassification and learning

Section 1.3 motivates the importance of learning in such a matching system. Since our template model is only an approximation of some concept, it must be able to change to accommodate new information about the concept. Since our system is a classifying one, it may err in one of two ways:

- It may identify a mapping between the template and situation objects as a template match that some oracle or expert denies is a "true" instance of the concept to be modeled, i.e. a *false positive example*.
- It may fail to identify some particular mapping of situation objects that an expert identifies as a "true" instance of our template concept, i.e. a *false negative example*.

Figure 3 depicts this system of an automatic Model Matcher identifying a false-positive model match of a template concept in a situation. The Expert Classifier (or oracle) identifies the positive match as false, and the expert can potentially explain this identification in terms of a possible revision. In various domains, one or the other of these classification errors may be thought of as "more important" than the other, however, for the purposes of this paper we will consider them equally bad, and we would like to be able to modify our model to properly classify these examples with the hope that our template will more accurately model the concept, and thus learn from these examples.

2.3.1 Case 1: False negative example model revision

How can we change our model? Well it is immediately apparent that *false positive examples* suggest a "tightening up" of the model so that it is more restrictive, and *false negative examples* suggest a "loosening" of the model to that it is more accepting, at least for the case of the new examples. If we also assume that the model that we have so-far is still valuable and representative in most cases, then we will want to perturb our existing model as little as possible. Specifically, in Figure 4, we see the space of all possible situations that would be categorized positively by the template as opposed to those that "should" be categorized positively according to some "ideal" expert or oracle from which we wish to learn to improve our template. The initial template space exceeds the expert or ideal concept space in certain areas, thus allowing for the existence of false-positive classifications, and the expert space exceeds the template space in some areas, thus allowing for false-negative classifications. Figure 5 shows how we might like to improve on our model so that the identified errors are accommodated. As is made clear from this figure, there is a possibility that changing the space to attempt to accommodate a single false-positive example or false-negative example may result in a space coverage that also excludes or includes other than the intended examples. Essentially we are changing the shape of the space and excluding or including whole ranges of situations, not necessarily only those that we wish to affect. A question arises as to how we may learn our new example(s), and yet minimally change the existing template concept in some sense.

In the case of a false-negative, we will wish to allow our model to be more accommodating. Essentially we wish to "relax" some aspect of our model so that the excluded example is included. Since we are assuming that our initial model is "largely" correct and

[3]For example, consider a reduction to graph coloring, which is a special case of CSP.

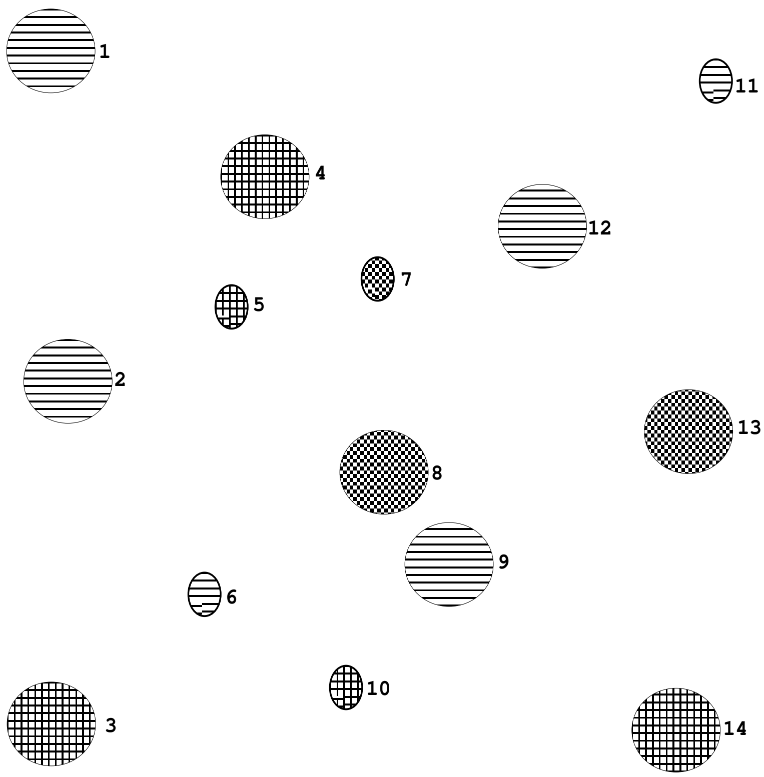

Figure 2: A situation in which template instances may be found.

only needs to be refined, we also assume that a false-negative example, classified as a positive instance by an expert, is a "near-miss" to our model. In terms of CSPs, a "near-miss" essentially means that some subset of the total constraint set failed to be satisfied during the situation matching process andor some nodes failed to be matched to situation objects. These "failures" are at least partially known as a result of the failure of the matching process (i.e. the "reason" for the failure" is the constraint that was determined to not be satisfied, or the node that had no acceptable situation object correspondence).

We would like to relax our model "minimally" in some sense in order to include the given example. In Freuder & Wallace (1992), the concept of Partial Constraint Satisfaction Problems (PCSP) is introduced in order to systematically define the relaxation of constraint satisfaction problems. Our problem of matching the given template (essentially the CSP definition) to the given situation (essentially defining the domain of the variables or nodes in the template) requires that we locate a "minimally" simpler template or CSP definition that will accommodate our new example. In essence, there are three ways in which a CSP or template may be "relaxed" or simplified:

1. The domain of a particular variable can be extended to include more domain values. In our context, this means that the node constraints on size or type may be relaxed to allow for additional possibilities.
2. A variable or constraint may be removed from the original CSP. Essentially, either some node or some constraint in the template is dropped from the concept definition entirely.
3. The domain of a constraint may be enlarged. A constraint on spatial distance, for example, might be changed from "between 1 and 2 kilometres" to "between 1 and 3 kilometres".

In Figure 6 we consider the original CSP as the common root of a graph representing all possible relaxations of that CSP. Each node in the graph represents simpler CSPs obtained by applying a sequence of relaxations. This graph is not a tree since application of relaxation steps in different orders can give the same PCSP as a result. We may make use of this PCSP space by searching in some fashion through it for the "first" relaxed CSP that classifies our example properly. Subject to the approval of some template revision expert, this relaxation could replace the previous template definition. In Yang & Fong (1992), we see an application of this relaxation technique that heuristically searches for relaxed solutions in a scheduling environment, providing solutions that are "closest" to those desired in the sense that a problem is solved that is "minimally changed" from the original problem.

In this fashion, search can yield *many* possible relax-

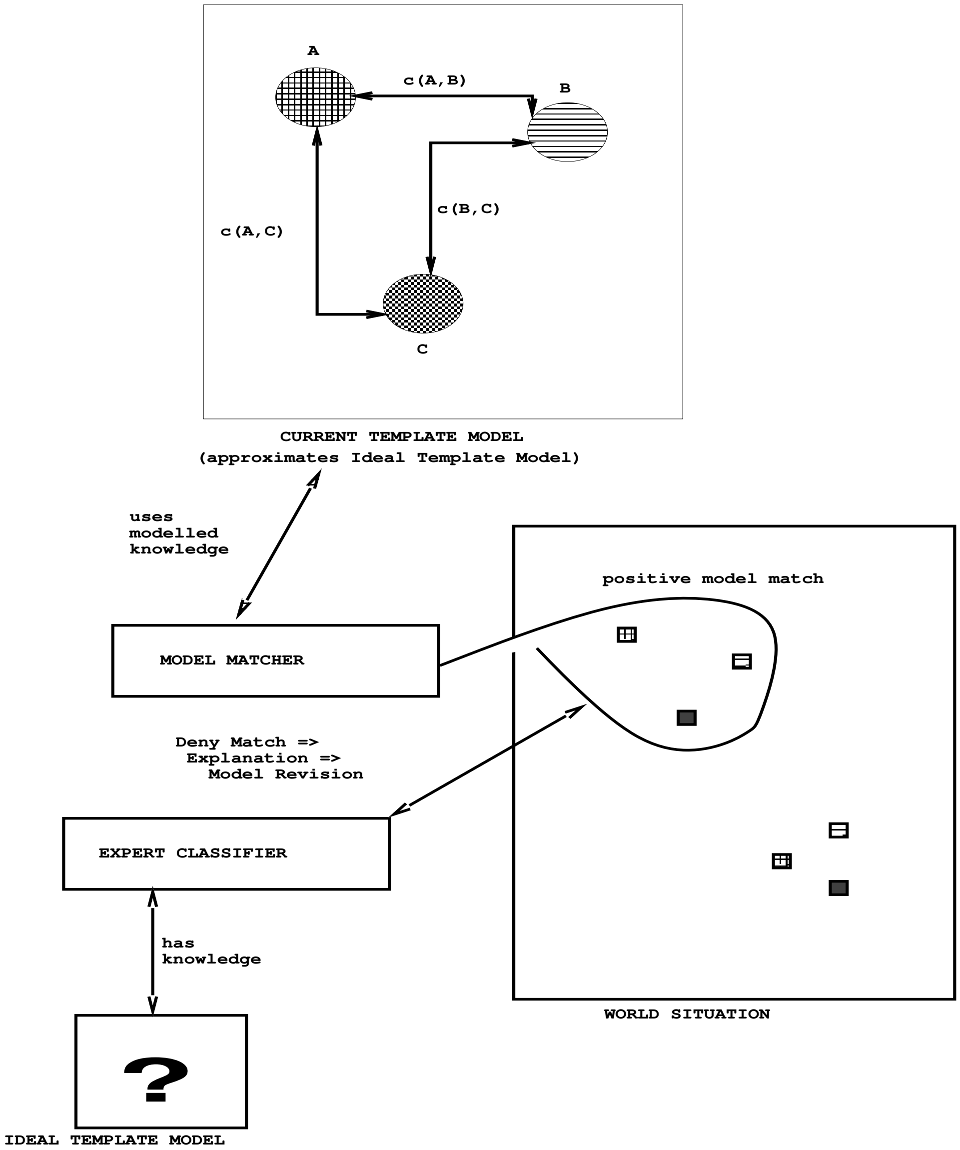

Figure 3: Expert revision of automatic classification.

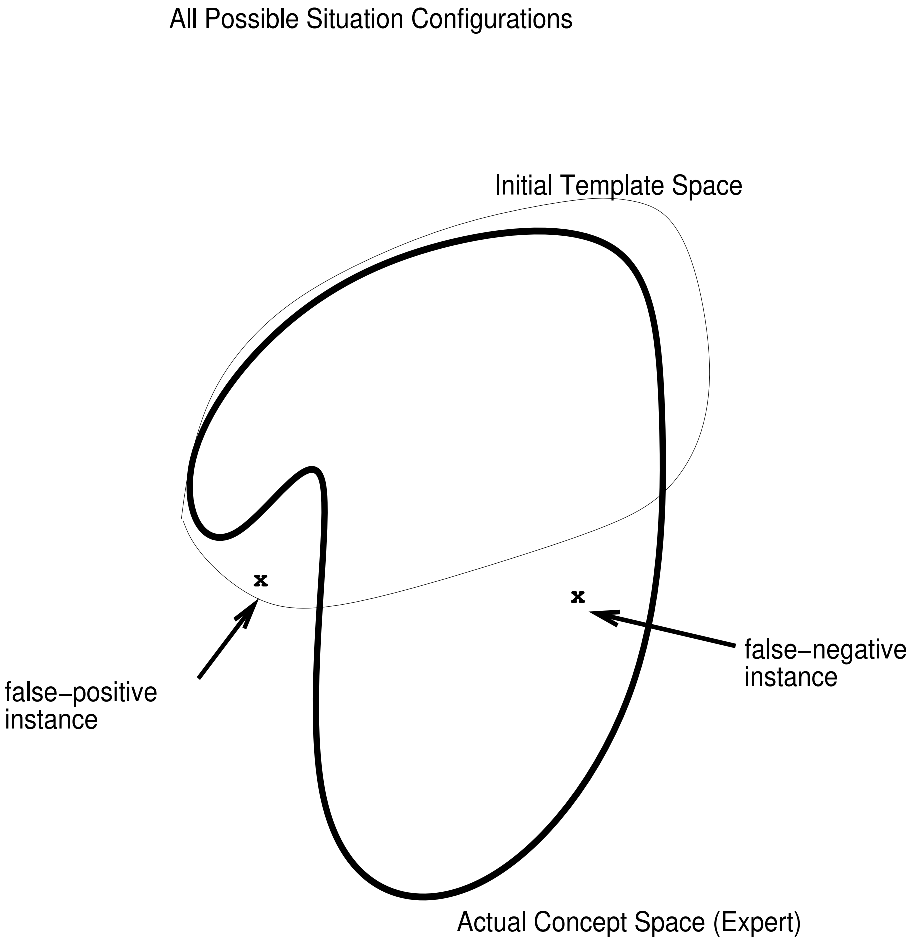

Figure 4: The initial coverage of a template vs actual concept.

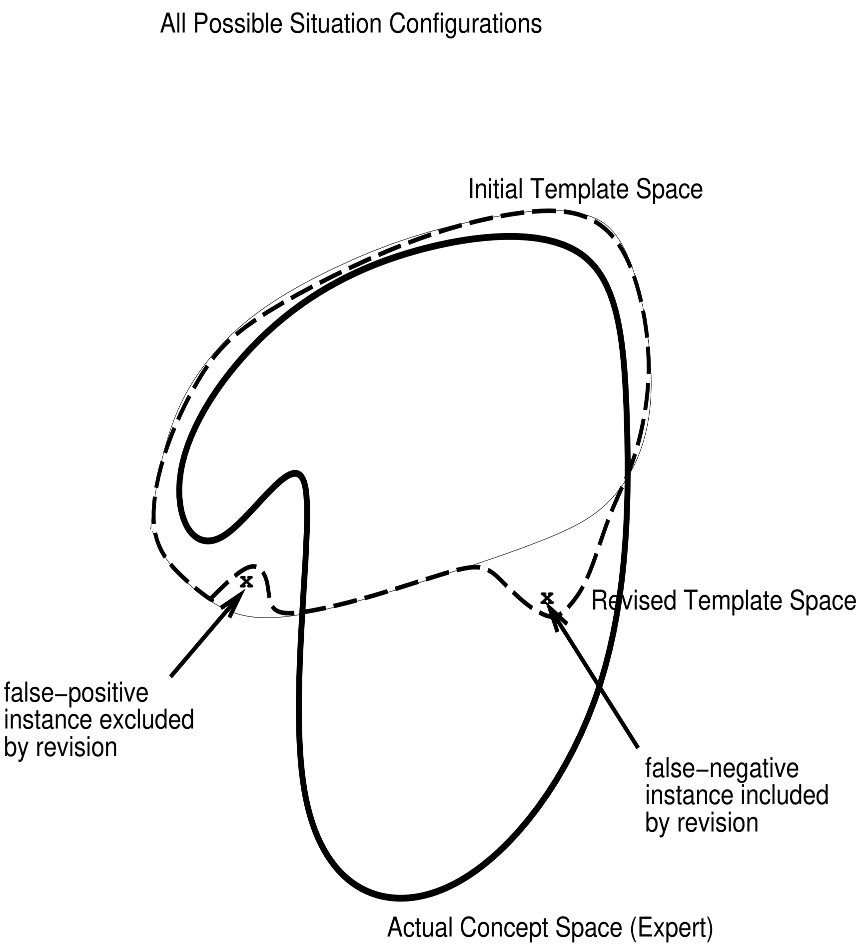

Figure 5: The revised coverage of a template vs actual concept.

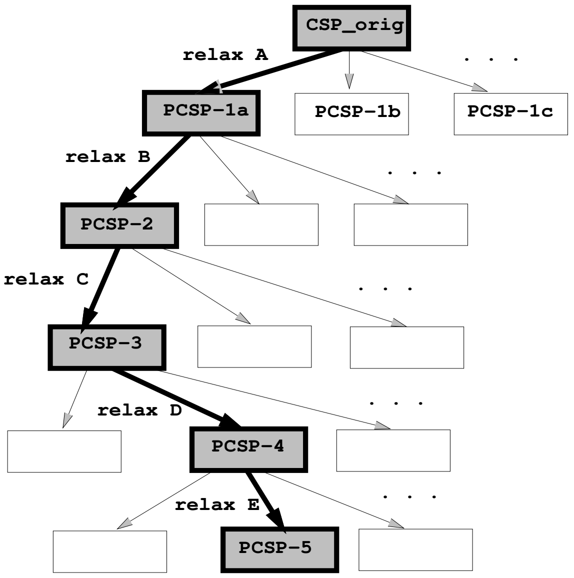

Figure 6: One abstraction hierarchy in a PCSP space.

ations that also satisfy our demand of accommodating the new example. Relaxed CSPs at the same "level" of the PCSP space might be considered "equivalently relaxed" in some sense, or perhaps relaxations of constraints might be preferred over relaxations involving node removal. Further, the definition of what exactly constitutes a "constraint relaxation" will need to be clarified at some length, since spatial constraints are often continuous, and a step by step relaxation suggests some discrete relaxation of a constraint at one step. The decision about which relaxation(s) to "prefer" can be extremely important since the perturbation of the "space" of situation examples made to accommodate the single example may have allowed for many other possible matches, some of which are undesirable. Since we are using the template as a "memory" of previously seen examples and not using the previous examples explicitly we wish to avoid as much as possible changing the space so that previously (correctly) rejected negative examples are now included.

What we have identified with this relaxation technique is a general approach to "minimally" relaxing our CSP (or template) definition so as to learn a new CSP (or template) definition that accommodates a new false positive example. Of concern is that the relaxation used in the refinement "makes sense" with regards to our concept in question. Simply using the minimal relaxation may accommodate the change with little adjustment to the actual model, however, selection of the actual constraint or set of constraints that should be used in this change is best accomplished when approved as sensible by a domain expert or oracle if available. What needs to be done in future work is to further identify the way in which these relaxations are specified in terms of allowable and desirable constraint relaxations.

2.3.2 Case 2: False positive example model revision

In the case that we wish to learn a false positive case into our concept model, we are faced with a problem symmetric to the false negative case in that we wish to create a model that is minimally (in some sense) more *restrictive* as opposed to more *relaxed*. We may apply our PCSP space generation and search approach precisely as before, with the exception that now we will restrict the CSP at each stage of the PCSP space generation. A candidate template for replacing the current template is one that is "minimally restricted" from the original, and yet successfully excludes the false-positive example. From the previous example we can see that there are three ways in which a CSP or template may be "restricted":

1. The domain of a particular variable can be restricted to include less domain values. For instance, the node constraints on size or type may be restricted to allow for fewer possibilities.
2. A variable or constraint may be *added* to the original CSP. This is problematic since without further contextual information, it seems to make little sense to randomly insert a constraint or variable. There are at least two interesting options at this point: we can refer back to the domain expert to suggest a constraint, variable, or set of constraints and variables to add to restrict the template further, or, we can attempt to incorporate some other learning algorithm here that might be able to identify commonalities amongst other positive examples that would exclude this negative example. We will leave the second option for future research and mention it here only as a point of interest.
3. The domain of a constraint might be reduced. A constraint on spatial distance, for example, might be changed from "between 1 and 2 kilometres" to "between 0 and 1 kilometres".

As was the case for the false negative example, the perturbation of a template based on a single example with the intent of accommodating that example affects more than the small part of the situation space indicated by the single example. Excluding a false positive essentially reduces the space covered by the template model and could possibly exclude previously (correctly) identified positive examples. The conception of "minimally" perturbing the model applies equally well in this case, however, there are no guarantees about the space more accurately reflecting the "true" concept that are immediately apparent. The key to the utilization of this PCSP relaxation and restriction approach is that the expert guides the process and is in a position to verify the model frequently.

It is interesting to note that in the false-negative case, we may obtain an explicit machine-generated explanation of the failure of a particular match based on the constraint values when the mapping from template to situation objects is made. This machine-generated (partial) explanation might potentially be used as a starting point for model revision. However, in the false-positive case, the expert is identifying a particular instance where the automated matching process succeeded, consequently the explanation of the model failure must come entirely from the expert, and any automatic restricting process would start without any possible "helpful" starting point.

3 Spatial template representation with multiple levels

To this point we have envisaged the template-based concept approximation at a single level represented by a CSP. We have attempted to show that this single level CSP could be refined through a learning process based on the concept of selecting minimally relaxed or minimally restricted versions of the original CSP in a PCSP space. In practical application, the single level template model we have described as an approximation of the actual template concept has several important failings. First, the initial error in defining the template based on expert information is often inaccurate in practice. In a domain where there are only a few positive examples to work with in refinement, this initial lack of knowledge can serious impair the accuracy of the matching process. Second, the situation that is being matched often contains information that is only partially known (for instance not all objects are typed or sized correctly, if at all), and when it is known, it may be only probable rather than certain. Consequently, the matching process may be attempting to identify poorly defined templates with "noisy" situations. If our matching process was very "rigid" and discounted all but certain matches we would be in trouble since this circumstance might rarely occur. In Section 1.4 the concept of abstracting the template model so as to allow for partial matching was outlined as one way to help alleviate the problem of identifying matches in uncertain circumstances.

The PCSP space introduced in Section 2.3 and detailed in Figure 6 can be utilized as a framework on which to build a hierarchy for a particular template concept. In building a PCSP space, we have constructed an abstraction space in which the least constrained PCSPs are the most abstract, and the most constrained PCSPs are the most specific instances of this CSP. However, there are many possible versions of the least constrained PCSPs. How do we differentiate amongst these in some fashion? What we would wish to do is to arrange these PCSPs into a more systematic hierarchy where some domain knowledge can be used to structure the hierarchy that we will use to define a template concept.

Consider that we could extract the "key" or "critical" information from a CSP, and form a "most abstract" PCSP (call it PCSP-3) which encapsulated this knowledge. Next, we extract several common distinguishing features or identifiers which would reflect the way in which the set of partial solutions is limited. Each of these sets of distinguishing features is applied to PCSP-3 as an inverse instance of the previously described relaxation techniques, specifically we can now add a variable, restrict a variable domain, or restrict a constraint domain. Essentially we now construct the more constrained PCSP-2, then PCSP-1, and finally our complete problem, CSP. Figure 6 on page 10 outlines this process. The number of stages or the groupings of the sets of "making stricter" actions can be dependent on a particular problem, and on the typical features or keys of that domain.

The advantage of this concept definition as a systematic structuring of the hierarchy of PCSPs is that a constraint satisfaction approach which solved PCSP-3 first resulting in a set S1 of solutions, then attempted to solve PCSP-2 for S2 given the initial starting point as each of the solutions in S1, and so on, would essentially have the quality of iteratively refining rough or partial solutions into specific or total ones. The effort spent arriving at the set S1 is not spent again in solving for set S2, as we build on the work done before. We are attempting to deal with the possibly flawed template model and the uncertain situational information by first attempting to identify instances that seems to satisfy the "key aspects" of the template, without additional detail. These "key aspects" may be thought of as the "minimally sufficient" portions of the template that would suggest a possible instance. Thus, "rough" matches are found first based on key indications, and then these rough matches are resolved by attempting to resolve the initial match into a match with the less-abstract template model in the hierarchy. This process can be repeated for as many levels of abstraction that are defined. We then have an iterative-improvement method for matching, where extra work provides more confidence in certain matches over others based on the identification or absence of less-abstract template features in the situation.

The process described above works very much in a top-down manner, solving PCSP-3 giving set S3, then refining S3 in solving PCSP-2 giving S2, finally refining S2 in solving CSP, giving our final solution set. One obvious difficulty with such an approach is that it finds all of the PCSP-3 solutions in S3 before attempting to find any solution elements for S2, for example. Basically each level is completely solved before the subsequent level in a breadth-first type of manner. This approach has the advantage that if search were halted at an arbitrary point in time, all of the "best-so-far" answers are on the same level. It has the disadvantage that time spent searching the less constrained CSPs might have resulted in more concrete solutions. In Woods (1991), a tradeoff search strategy known as 'Left-Wedge" is described for exploring the multiple level of abstraction solution space which progresses downwards towards more concrete solutions, AND from left to right through the solutions at each level at the same time. This strategy is a variation on A* search which can provide us with both concrete solutions and less concrete solutions as more time is spent searching.

To some extent, the selection of an approach for this type of search depends highly on the way in which we would wish to take advantage of interaction with the user during search. It is conceivable in a certain case that a user may wish to view and prioritize or eliminate higher level solutions in order to guide the discovery of concrete solutions.

Figure 7 details one level of an abstract concept definition based on our earlier template example. In this figure we see an abstract or generalized template labelled TA_0, and a more specific or specialized template labelled TA_1. Between these two templates is the "hierarchical glue" that maps both from specialized to general and general to specialized, detailing the exact operations (relaxing operations from specialized to general, and restricting operations from general to specialized) that define a path in the PCSP space that has been identified at initial template definition time as representative of the "key features" and "subsequent evidence" for this template.

We can see that hierarchical specifications improve over single level specification for locating spatial templates in at least the following ways:

- The inaccuracy inherent in the representation of spatial templates and in the representation of situations makes a single-level model very brittle or rigid in terms of searching for mappings with a situation, and consequently particularly subject to errors in classification even in near-miss cases.
- The hierarchical structure provides a degree of "Anytime-ness" to the matching process, and allows us to iteratively improve confidence in solutions with additional effort in search.
- Abstraction support a model of interaction of user and search, particularly with the anytime search-and-refine type of approach. Matching can be done in an "intuitive" top-down fashion from "key factors" to "additional detail", thus providing matches at the greatest degree of detail or confidence possible, but still providing partial (abstract) matches in the absence of either time to search further, or when the template or situation error has prevented more detailed identification.

Our template model now has multiple levels, with the levels mapped by connecting "functions". While this new structure may provide flexibility in recognition in noisy situations and may still provide "high level" matches when a complete match is non-existent, the problem of refining a template model based on expert identified false-positive or false-negative cases has become significantly more complex. In the next section we will outline the cases that present themselves, and suggest a starting point for a method of learning by refinement of our model.

4 Refinement of abstracted template models

Many earlier works in designing abstraction models in other domains including Sacerdoti (1974), Yang & Tenenberg (1990), Christensen (1990), Woods (1991), and Yang, Tenenberg & Woods (1993) often involved specification of abstraction hierarchies with no specific connection between the hierarchical levels. Other work in abstraction and philosophy has suggested that this type of abstraction is a poor reflection of the way people form abstractions. A richer model that is often described, for instance in Holte, Zimmer & MacDonald (1992) and Holte, Zimmer & MacDonald (1993), is one where the "mapping" between layers is at least partially specified. Further, this mapping in describing all or part of the way in which one level of specification can be transformed into another, provides a perfect framework in which changes in one level of a hierarchical model may be reflected in other adjacent levels. Effectively, changing our belief about what constitutes an accurate generalized description of a concept *may* affect what we believe is an accurate specific description of that same concept. Further, a change in belief about a specific description may entail a change in a generalization. Accommodating these changes in belief about an accurate depiction or representation of a domain concept into the representation itself is precisely what we mean as "learning" within a hierarchical spatial template model.

The process of identifying template matches or partial matches in a situation was described in the previous section as a "top-down" method, where the most abstract model is matched to a situation before a less abstract model. Further, any abstract matches are refined according the the mapping to less-abstract detailed in the "hierarchical glue". Rather than a single case covering each of "false positive" or "false negative" matching, we now have the more complicated problem of determining how to modify our entire "hierarchy" based on expert identified errors at various points in the hierarchy. Identified changes to a level of the hierarchy may imply the need to propagate these

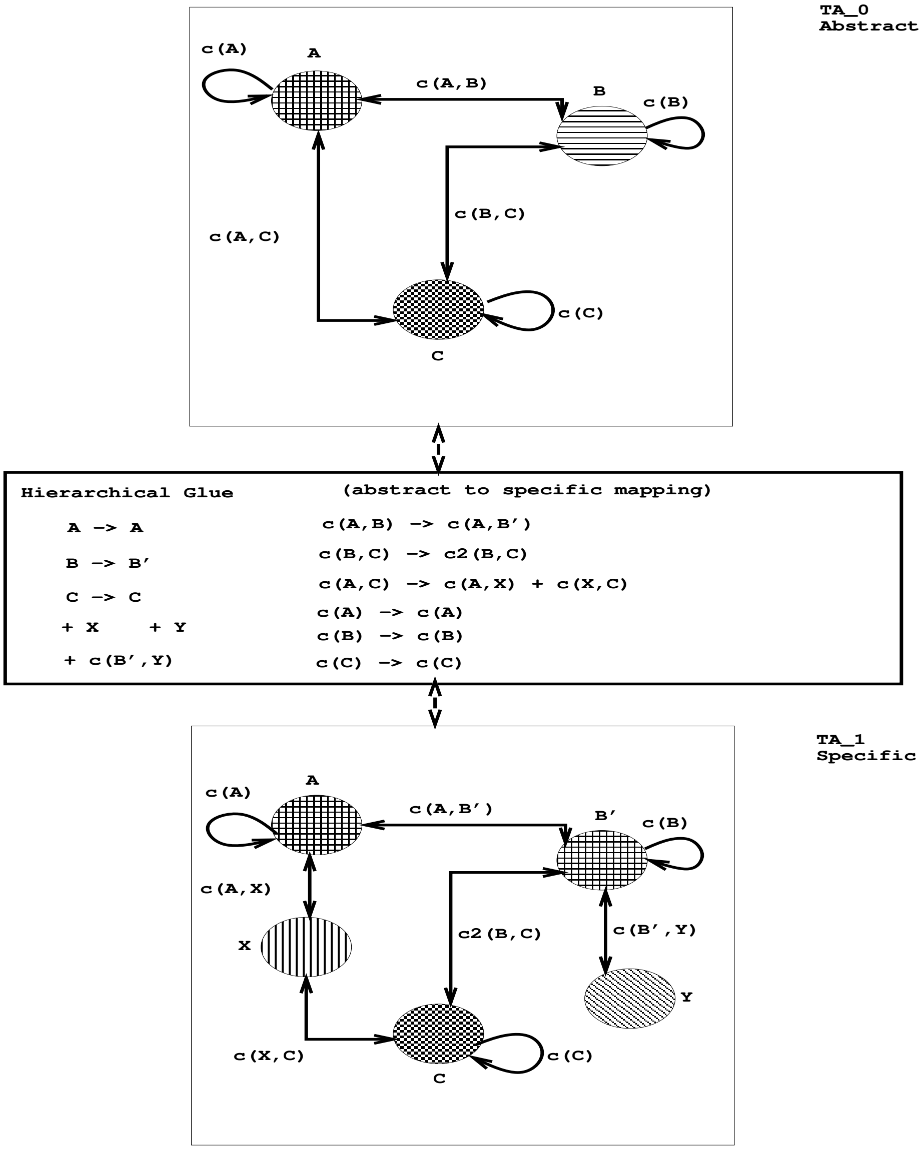

Figure 7: Explicit expression of representation change.

or similar changes upwards or downward, or in some cases, even suggest the creation or deletion of a level of the hierarchy. While these decisions seem to be best made by an expert, we are interested in identifying the most likely or sensible changes. We will discuss a hierarchy that has at least three-levels. In such a hierarchy, it is conceivable that changes to one level may need to propagate up, down, or both up and down in the hierarchy. We will refer to the levels as *abstract*, *middle*, and *specific* for simplicity. The classification problems we will wish to accommodate in our model refinement approach to learning are:

1. False-positive at *abstract*. In this case we are identifying that we may wish to restrict a part of the model that is parent to more specific parts. Essentially, we are stating that the most abstract model specification is too abstract. The question is, do we wish to propagate this "restriction" downwards in our hierarchy, and if so, how far down? Certainly a restriction may have to be reflected in the "hierarchical glue" mapping, even if a restriction is not made at the more specific level. It is possible that a restriction at an "upper" level may render the level change to the subsequent level of little or no use, and the levels may most sensibly be merged.
2. False-negative at *abstract*. The most abstract model specification has failed to accommodate an expert identified instance that should succeed. The abstract model is supposed to encompass the "key indicators" of the model, and the failure to accommodate a positive example indicates that the abstract model must be relaxed further. This further relaxation could potentially be propagated downwards in the hierarchy as other relaxations, or even suggest that a new level in the hierarchy be created above the current level.
3. False-positive at *middle*. As was the case for *abstract*, relaxations may need to propagate downward causing one or more relaxed lower levels. Additionally, a relaxation may indicate that a key-feature of the model that exists at a more abstract level should also be relaxed. In this case, changes would propagate upward via the hierarchical mapping function.
4. False-negative at *middle*. Once again, as was the case for *abstract*, restrictions may need to propagate downward causing one or more restricted lower levels. In addition, a restriction may indicate key-feature of the model that exists at a more abstract level should also be restricted. In this case, changes would propagate upward via the hierarchical mapping function.
5. False-positive at *specific*. There will be no relaxations pass downwards, however, they potentially could pass upwards one or more levels.
6. False-negative at *specific*. There will be no restrictions pass downwards, however, they potentially could pass upwards one or more levels.

Hard and fast rules are not immediately clear regarding how these hierarchies should refine in each situation. It is obvious that in absence of a better understanding about how the "space" of example instances by the model is affecting by changing the model in one direction or other, we must rely primarily on the expert for determining the degree of inter-level change that is supported for a particular model change. However, it would seem that there are two primary cases for inter-level refinement:

1. False-Negative adjustment to a level, implying in-level space expansion
   (a) Upward propagation of restriction, implying that the upper level(s) are similarly (or partially similarly) restricted. Essentially if the more abstract level is functioning as expected, this likely is not necessary, and a misclassification of the specific does *not* seem to imply a necessary misclassification of the general, particularly if the misclassification was noted by the expert during a top-down search.
   (b) Downward propagation of restriction, implying the lower level(s) are similarly restricted. Essentially is would seem to make sense that a restriction of the abstract should restrict the specific in the case that a relevant mapping exists from abstract to specific for the adjust problem aspect or constraint affected. As an example, consider modifying an abstract such as chair to include only those things with four legs, changed from a previous description of three. If the abstract now has four, subsequent more specific versions of this concept such as office-chair should have four also.
2. False-Positive adjustment to a level, implying in-level space shrinkage
   (a) Upward propagation of relaxation, implying that upper level(s) are similarly (or partially similarly) relaxed. Once again, it would appear that if the abstract level is functioning correctly the expert would not wish to reflect these specific changes at that abstract level.
   (b) Downward propagation, implying that lower level(s) are similarly relaxed. Once more, a change at the abstract would seem to imply an appropriate change at the specific if applicable.

The determination of revision in a hierarchy is certainly an open question, and the development of some experimental evidence and further investigation into representation change of a concept which is described at several connected levels of abstraction simultaneously is certainly warranted.

5 Summary

5.1 Why is this work learning?

We are faced in this work with a problem of refining an expert-defined model based upon subsequent examples that fail to fit the current model. Learning new versions of the model when past examples incorporated into the model either no longer are available for direct evaluation, or when these past examples are already represented in an expert's mental model differs dramatically from learning from a set pool of examples, as is the case with Probably Approximately Correct (PAC) learning theory as described in Anthony & Biggs (1992). What is important in this work is refinement that accommodates a given counter-example, while perturbing the existing concept definition minimally, or if possible, perturbing the existing concept definition exactly enough to accommodate the new "class" of problems represented by the given counter-example. In the ideal case, a counter-example identified a single constraint error in some template and a single relaxation or restriction of this constraint is identified and made in the template model. In less ideal cases, we have to somehow select one of many possible relaxations or restrictions that will accommodate the new information. At this point, expert interaction in the process seems advisable if additional domain information or examples are unavailable. We have attempted to outline the major issues and problems in developing an interactive system for refining both single-level and hierarchical template models of spatial concepts.

The general conception of a problem as a graph of constraints and the associated solution set of this graph within a particular situation or domain value set is very powerful. Learning changes to a single-level constraint graph representation from single examples is problematic, especially when considering coverage of previously accommodated examples. Determination of problem aspects for relaxation or restriction is heavily domain dependent, and the specification of explicit candidate problem aspects such as slot removal, specific constraint removal or adjustment and degree of restriction/relaxation needs to be determined by an expert either prior to use of any refinement method, or during refinement. The *addition* of problem aspects such as slots or constraints would appear to be a very difficult problem without many examples present, however, the possible application of a "meta-learning" approach to determining additive components would appear to be an interesting area of future research.

Certainly the definition of a concept approximation such as a spatial template is often intended to "outlive" a particular expert or set of experts, and essentially serve as a definition of the concept to future users or experts. Consequently, model refinements may also be viewed as a transformation from an approximation of a concept held of one user set to an approximation of a (possibly different) concept held by another user set. Similarly, if one spatial template represents a concept A, then if another (known to be similar but slightly different) concept B has no currently known representation, then accepting some spatial template representing A as an approximation of B and then modifying it by example to be closer to an approximation of B seems a sensible approach when specific information about concept B is unavailable.

5.2 Additional Related work

As mentioned, the before-revision template is an approximation of a concept based on experience and previously encountered examples; essentially it is a form of memory of both of these. Wasserman (1989), describes the problem of revising a "memory-based" model based on a single example as the *stability-plasticity dilemma*. The dilemma essentially is whether or not a model can remain plastic enough to incorporate new examples as they arrive and yet retain stability needed to ensure that previous example adjustments are not lost. Incorporation of a new pattern by changing a model may result in the model misclassifying examples it classified correctly previously, and essentially get no closer in approximating the ideal concept. Wasserman (1989) points out that in some domains, neural networks have been shown in such cases of temporal instability to never converge to the desired concept. As a solution, Wasserman (1989) describes Adaptive Resonance Theory (ART) as a way of "distributing" the cumulative or past-experience "memory" so that subsequent examples have only "local" effect and thus do not in a single stroke replace previous knowledge. In a sense the cumulative existing model is preferred of immediacy of a new example. This same type of behavior is essentially reflected in the "minimal" accommodation of templates to the new examples that has been discussed earlier in this work.

A problem with many similarities to learning constraint graphs based on new information is reported in Lam & Bacchus (1993*a*) and Lam & Bacchus (1993*b*). In these works, Bayesian Networks are learned based on the "Minimal Description Length (MDL)" principle. The MDL principle is based on the idea that the best model of a collection of data items is the model that minimizes the sum of:

1. the length of the encoding of the model, and
2. the length of the encoding of the data *given the model*,

both of which can be measured in bits. A good example of this principle is given in Lam & Bacchus (1993*a*), for polynomial learning based on n points in the plane. Essentially, to find a function or model that precisely fits these points, $n + 1$ numbers are required representing the coefficients of the polynomial, and these represent the (first item) encoding of the model. Additionally, the store the data given this polynomial (the second item), n x-coordinates are required so that the n y coordinates could be computed from the model. Consequently the sum of the description lengths would be $2n + 1$ times the number of bits required to store the numbers for some precision. Now, another model might use a lower order polynomial, say of some order $k < n$, and thus only $k + 1$ numbers are required to store the coordinates (first item). Now, the data points can be specified by specifying the x-coordinates, however, this model will not fit the data precisely, and there is going to be some error e_i between the y_i value computed for each x_i and the actual y-coordinate of the i-th data point, y_i. To encode the data points, these error factors would need to be stored along with the x-coordinates (item 2). Clearly it is possible that there may be some polynomial of degree $k < n$ that yields the minimum description length.

Essentially the MDL conception is an alternative to our definition of "minimal" as the least possible number of restrictions or relaxations in the PCSP space. Since we do not have the previous examples stored explicitly in the model as specified, it is difficult to establish how to calculate the MDL for template revision. Lam & Bacchus (1993*a*) defines the two item calculation of MDL for Bayesian Networks, and thus prefers models in a search space of possible learned models accordingly. An interesting future work would be to attempt to define the MDL in terms of constraint graphs.

In Lam & Bacchus (1993*c*), the earlier work is extended to account for refining an existing network based on new data (or examples). The focus in this work is on taking advantage of local revision where possible (since new data may often be localized in the existing Bayesian Network). This work is reflective of the work presented here in that they attempt to preserve as much as possible of the original network as is consistent with the new data. Ideally this is exactly what we would like for template revision. The learned Bayesian structure is desired to be be as simple as possible, to be similar to the original structure, and to accurately represent the distribution of the new data. In the paradigm of this paper, "as simple as possible" reflects into the abstract representation in some sense - perhaps the idea of not adding new constraints, but rather modifying existing ones parallels this concept. The aspect of "similar to the original structure" maps directly to our concept of least relaxed or restricted in the PCSP space, and "accurately represent ..." covers our requirement that the new example be encompassed by the new model. What is lacking in our work to make this kind of mathematical evaluation of a "minimal" change to a model is evaluation functions for change, and a method of representing how individual previous examples are reflected in a new model.

5.3 Conclusion

I have examined here the revision of spatial template represented as constraint graphs as an attempt to learning the "ideal" spatial concept held by one or more domain expert(s). Further, I have attempted to outline the issues in attempting to incorporate these revisions into a more complex hierarchical template model. While this is certainly a work in progress, it is clear that learning revisions to hierarchical models is both important and difficult, particularly when the exact context of examples is not immediately apparent. The presence of an expert can effectively guide the learning process through these "holes", however, the issue of what general and domain knowledge needs to be modeled in order to automatically make these types of decisions is an open question. In essence, we are asking the very hard question "What do we need to know about the domain, knowledge representation, spatial problems, and learning itself to be able to successfully refine our model?". In the absence of all of the pieces of this puzzle, this work is directed towards obtaining an interactive, but intelligently-guided system that supports spatial template model revision and instance identification.

Acknowledgments

This work has been carried out specifically for the requirements of CS786, however, it follows from work undertaken by the author for the Department of National Defense (DND), Defence Research Establishment Valcartier. The author is a PhD candidate supported by an NSERC scholarship and a research contract with DND.

References

Anthony, M. & Biggs, N. (1992), *Computational Learning Theory*, Cambridge University Press.

Christensen, J. (1990), A hierarchical planner that creates its own hieararchies, *in* 'Proceedings of the 8th AAAI', pp. 1004–1009.

Freuder, E. & Wallace, J. (1992), 'Partial constraint satisfaction', *Artificial Intelligence* **58**, 21–70.

Holte, R., Zimmer, R. & MacDonald, A. (1992), When does changing representation improve problem-solving performance ?, *in* 'Proceedings of the Workshop on Change or Representation and Problem Reformulation'. NASA Ames Technical Report FIA-92-06.

Holte, R., Zimmer, R. & MacDonald, A. (1993), A study of the representation-dependency of abstraction techniques, *in* 'Proceedings of the 1993 Workshop on Knowledge Compilation and Speedup Learning'.

Lam, W. & Bacchus, F. (1993*a*), Learning bayesian belief networks: An approach based on the mdl principle, Technical report, University of Waterloo.

Lam, W. & Bacchus, F. (1993*b*), 'Using causal information and local measures to learn bayesian networks', *UAI*.

Lam, W. & Bacchus, F. (1993*c*), Using new data to refine a bayesian network, advance copy.

Lapointe, S. & Prouix, R. (1994), Fuzzy geometric relations to represent hierarchical spatial information, *in* 'Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence'. to appear.

Prosser, P. (1993), 'Hybrid algorithms for the constraint satisfaction problem', *Computational Intelligence* **9**(3), 268–299.

Sacerdoti, E. (1974), 'Planning in a hierarchy of abstraction spaces', *Artificial Intelligence* **5**, 115–135.

Schank, R. (1992), 'Where's the ai?', *Artificial Intelligence Magazine* **12**(4), 38–49.

van Run, P. (1994), Domain independent heuristics in hybrid algorithms for csps, Master's thesis, University of Waterloo.

Wasserman, P. D. (1989), *Neural Computing Theory and Practice*, Van Nostrand Reinhold.

Woods, S. (1993*a*), Interactive recognition of spatially defined model deployment templates, Research memorandum, Combat Intelligence Automation Group, Defence Research Establishment Valcartier, Courcelette, Quebec CANADA. unclassified.

Woods, S. (1993*b*), A method of interactive recognition of spatially defined model deployment templates using abstraction, *in* 'Proceedings of the 1993 DND Workshop on Advanced Technologies in Knowledge Based Systems and Robotics'.

Woods, S. (1994), On the complexity of graph isomorphism : relationship to the spatial template recognition problem, advance copy, technical report.

Woods, S. G. (1991), An implementation and evaluation of a hierarchical non-linear planner, Technical report cs-91-17, Computer Science Department, University of Waterloo.

Yang, Q. & Fong, P. (1992), Solving partial constraint satisfaction problems using local search and abstraction, Technical Report cs-92-50, University of Waterloo.

Yang, Q. & Tenenberg, J. (1990), Abtweak: Abstracting a nonlinear, least commitment planner, *in* 'Proceedings of the 8th AAAI', Boston, MA, pp. 204–209.

Yang, Q., Tenenberg, J. & Woods, S. (1993), Abstraction in nonlinear planning, Was University of Waterloo Technical Report CS-91-65.