24. Recognizing Structures: Some Problems in Reasoning with Drawings

Richard White

EdCAAD
Department of Architecture
University of Edinburgh
Edinburgh, United Kingdom

This paper describes work on our current project aimed at developing a generalized system for performing automated reasoning tasks in various domains, using information extracted from drawings. It briefly describes the MOLE representation system, aframe-like formalism which can be used to build both description and inheritance hierarchies. The use of MOLE for representing graphical objects as well as the objects they represent is also described. The paper goes on to discuss some of the problems faced in the development of systems which can perform reasoning tasks on such representations. In particular, problems arising from the need to map the structures required by the application domain to the drawing description are outlined and a model which adapts existing Artificial Intelligence (Al) techniques to solve these problems is proposed.

Introduction

Our current project (SERC Research Project GR/F61868) is concerned with issues surrounding the extraction of knowledge from drawings, with the aim of allowing this knowledge to be used by applications to perform reasoning tasks. In particular we are considering how such knowledge can be used for assessing the fire safety of buildings, but our aim is that the system should be more general, and capable of supporting applications in any domain.

This paper describes our approach, using the MOLE formalism for representing objects and drawings (Tweed & Bijl 1988, White & Ramscar 1991). The basic knowledge representation scheme is described, along with a method of using it for describing drawings using logical relationships between segments. We propose a model which embodies a schema driven process of drawing analysis, with the schemata upon which a particular interpretation is based being chosen by their relevance to the current task as defined by the user. Mapping between the high level structures of the task domain and the users design description serve to prune the possible search space further.

Knowledge Representation in MOLE

The MOLE knowledge representation system has been more fully described elsewhere (Tweed & Bijl 1988), as has its relationship with a system following the classic frame convention (White 1990). This section will outline the basic knowledge representation system with a view to going on to discuss the description of drawings using this formalism.

Kinds, Slots and Fillers

The basic entities of representation are the *kind, slot* and *filler*. A kind (as in "any-kind-of-thing") is defined by a unique name, and a series of slots which make up its description. The description is completed by giving the slots values via their fillers. Thus the description of a cylinder head within a car engine might include:

```
cylinder-head:
    material <- alloy
    part-number <- 61842#45
```

where the kind *cylinder-head* has slots *material* and *part-number* which are filled by the values *alloy* and 61842#45 respectively the <- operator being the "assignment" operator for filling slots with values.

It is important to note that the slots represent decomposition of the *description* of an object rather than the object itself. Bearing this in mind, the relation between the kind and slot is "has-part". Thus the description of the kind *cylinder-head* has the parts *material* and *part-number*.

As well as atomic values, slots can be filled by other kinds, allowing hierarchical descriptions of complex objects to be built up. For example the description of a cylinder head may form part of a larger representation of an engine, which in turn may contribute to the description of a car:

```
car:
    body <-...
    engine <- #syskind00l
    chassis <-...

syskind#O01:
    bottom-end <-...
    cylinder-head <- #syskind002
    gearbox <-..

syskind#002:
    valves <-..
    piston <- #syskind003
    gasket <-...
```

syskind#003:
material <- alloy
part-number <- 61842#45

Elements of the resulting tree structure can be accessed by specifying the path that is traversed in reaching them:

car:engine:cylinder-head:piston:material <- alloy
car:engine:cylinder-head:piston:part-number <- 61842#45

These path specifications provide the means by which MOLE descriptions are defined and edited (White & Ramscar 1991).

It is therefore possible to build up structured descriptions of large and complex objects, expressing many levels of granularity within the hierarchy. MOLE does not impose any system of decomposition upon the user. They are free to describe an object to whatever level they require, using whatever method of breaking down the description they think appropriate. As we shall see this will lead to some profound difficulties when it comes to the development of systems for extracting meaning from drawings.

Instances

Another way of describing an object, or part of an object, is as a specific instance of some more general category of things. This is especially the case if we need to describe a number of things which can be described in a similar manner, with only minor differences in their representations.

MOLE allows the user to define one kind as an instance of another kind. The following structure might represent part of a description of a tenement block:

tenement:
location <- guthrie-street
walls <- stone
roof <- slate
number-of-flats <- 6

This is not a description of any particular tenement, but describes an imaginary "typical' block in Guthrie Street. If we want to describe a specific tenement then we can define it as an instance of this typical one:

number 5 ' tenement

where A is the "*instance-of*" operator.

Thus the structure describing *numberl5* inherits all the slots and values describing a typical Guthrie Street block, with the exception of *roof* which is over-ridden by the local value:

numberl5:
location <- guthrie-street
walls <- stone
roof <- slate
number-of-flats <-6
roof <- felt

Note that the inherited slots, in italics, are not stored as part of the *numberl5* structure, but are available for evaluation in the same way as if they were. As with the decomposition of objects, MOLE places no restriction on the user as to how to organize inheritance with the exception that a kind cannot inherit from more than one other kind.

Counterparts

Finally, and recently, we have introduced the *counterpart* relation (Ramscar 1991). This enables structure sharing where two kinds share the same slots and fillers, and changes to one result in changes to the other. To use a simple example, if a rectangular room was specified, then the opposite sides would be defined as counterparts:

room: wall1:length <-10
wall2:length <- 5
wall3:length = wall1:length
wall4:length = wall2:length

where = is the counterpart operator.

Thus whenever *wail :length* or *wall3 :length* has their value changed the value of their counterpart will be changed also. This relation provides the means for users to include their own constraints on values within a design description.

With decomposition, inheritance and counterparts, MOLE provides a flexible means of representing any type of object. The following section describes how this formalism can be used for the representation of drawings.

Drawing in MOLE:

The Basic Ingredients

The current method of representing drawings in MOLE was developed several years ago in EdCAAD (Steel & Szalapaj 1983) and was based upon a desire to have a representation system which could combine graphic and abstract descriptions of objects within MOLE, and which would treat lines symbolically rather than mathematically according to a particular co-

ordinate system. The method described here is based upon the metaphor of constructing drawings from pencil and ink lines as practiced by designers at a drawing board. However the achievement of the original aims is not inextricably linked to the use of this metaphor.

Part of the MOLE system (as distinct from the MOLE representation language) is a *drawing machine,* which takes graphical input from a designer and generates MOLE descriptions corresponding to the resulting graphical objects. The first step in the production of a drawing is for the designer to call up a top level kind, say *house-plan* by which the resulting descriptions can be accessed.

In line with the pencil and ink metaphor, the designer first lays down some 'pencillines'. These will later be "inked in" to produce the actual drawing. The pencil lines, or *construction lines,* act as carriers for the final ink lines, or *segments,* and have no physical dimensions other than an angle, relative to some pre-defined origin. The intersection of two construction lines defines a *construction point,* and two construction points, in turn, define the endpoints of a segment. Figure 1 shows a simple example of this process.

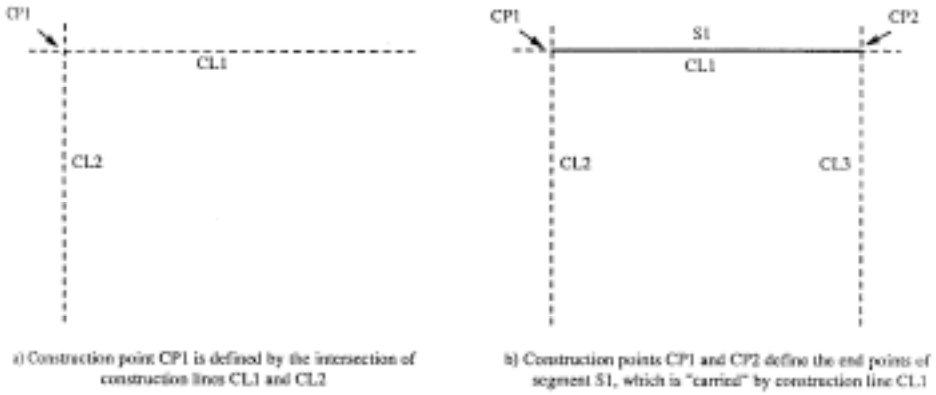

Figure 1. Drawing using construction lines.

As well as containing information about the construction lines which define them, construction point structures contain attachment information which defines how segment intersections will react as the drawing is manipulated graphically (see Steel & Szalapaj 1983 for a description of some of the problems of attachment). The segment structure contains slots for defining its endpoints, length and bearer, the latter pointing to a construction line which defines the angle of the segment. Further information referring to the style properties of the segment is also included. Figure 2 provides an example of segment and construction line descriptions for a rectangle.

While this description may appear over-complex for the description of a simple rectangle, the construction points and lines will be used as part of the definition of other shapes without the need for repetition. A single construction line can form the basis for more than one geometric form, even if they are non-congruent, and this feature allows separate objects within a drawing to be easily orientated with respect to one another. In effect, the user is

defining their own grid to be used for the positioning and orientation of objects within the drawing. An advantage of this system is that lines on this grid need only be defined when needed (i.e. will not cover the entire drawing space) and are not limited to orthogonal orientations.

As the graphical elements are represented as MOLE expressions, it is possible for the designer to edit these expressions directly, and the drawing machine will reflect these changes in the drawing. Additionally, and more importantly, it allows for the addition of higher level structures to the drawing representation.

```
mole:[richard:[
    conline1:[angle<- 0],
    conline2:[angle<- 90],
    conline3:[angle<- 90],
    conline4:[angle<- 0],

    conpt1:[first_conline<- conline2,
            second_conline<- conline4],
    conpt2:[first_conline<- conline3,
            second_conline<- conline4],
    conpt3:[first_conline<- conline1,
            second_conline<- conline3],
    conpt4:[first_conline <- conline1,
            second_conline<- conline2],

    seg1:[      first_conpt<- conpt1,
            second_conpt<- conpt2,
            bearer<- conline4,
            length<- 3060,
            style<- "solid"],
    seg2:[      first_conpt<- conpt2,
            second_conpt<- conpt3,
            bearer<- conline3,
            length<- 2640,
            style<- "solid"],
    seg3:[      first_conpt<- conpt3,
            second_conpt<- conpt4,
            ...
            style<- "solid"],
    seg4:[      first_conpt<- conpt4,
            second_conpt<- conpt1,
            ...
            style<- "solid"]].
```

Figure 2. A rectangle defined in MOLE.

Building Larger Things

If we take the description of a simplified house plan that might be produced by the drawing machine (figure 3), it will consist of a large number of segment definitions, along with the construction line and point definitions supporting them. MOLE enables these bottom level descriptions to be linked together into higher level structure, for example into squares or rectangles, perhaps using the inheritance mechanism to produce generalized template descriptions.

The top level plan could then be described in terms of these higher level structures rather than the lowest level segment descriptions. This would convert a single one-layer description (figure 4a) into a two-layer description (figure 4b), and other layers could be added equally easily if required. The process of producing extra layers of description can be seen as *compositional* if carried out in a bottom-up manner as described here, with simple elements being combined to form more complex ones. Similar effects can be obtained using a *decompositional* or top-down manner, where description starts at the highest level and each level is split into lower level constituent parts until the basic level of representation is reached. MOLE supports both methods of structuring complex object and the criteria used for combining or decomposing elements to form new ones are solely the responsibility of the designer. Similarly the system does not limit the description hierarchy to binary trees, but allows any type of graph structures, including cyclic, to be built.

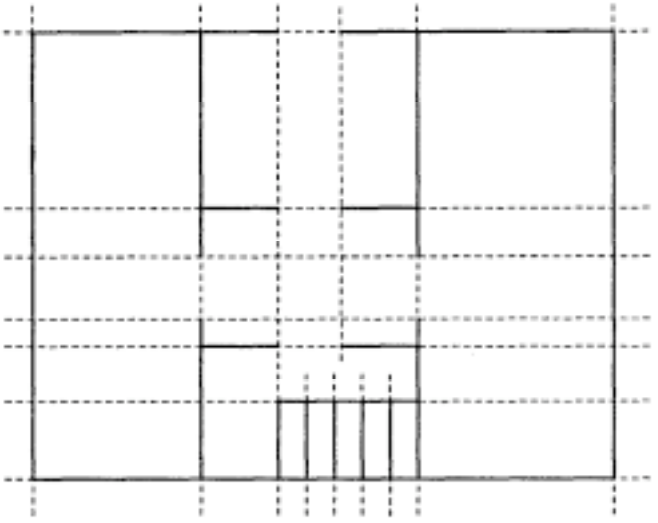

Figure 3. Simple plan of a house - ground floor.

What is to be Gained from Adding Extra Layers of Description

From the designers' point of view, drawings can be manipulated at levels higher than the basic segment descriptions, say at the level of the square. A designer can group a series of lines into a square, either graphically or by the direct editing of MOLE expressions, and then move the lines together as a square, rather than individually. This will be quicker, more efficient and lead to fewer errors.

Additionally it will be possible for the designer to view the design description at any one of a number of different granularities. The granularity of a description is an expression of how much detail is present. At high levels of description, there is less detail than at low levels. The granularity required of a particular view of a description will depend on the use to which that view is to be put. For example if the description of a building is to be used for town-planning, then nothing more might be required than a very high level description giving information about such things as its overall shape, size, and elevation. If the building was to be marketed to clients then they would want a lower level view of the description, with details of the number of rooms, their sizes etc. Finally, if the design description was to be used as a basis for the building process itself, then a still lower level view giving details of materials and quantities would be required. The careful structuring of the design representation could allow all these viewpoints to be accessed from one description.

Structuring the design description also leads to advantages from the point of view of building systems to manipulate this description automatically. Any such system will, at some stage, have to perform search operations on the plan description, to access certain drawing elements. In a flat, single layer, representation this effectively becomes a sequential search through all the elements of the drawing. This can be likened to searching for the definition of a single word in a random list of definitions - a long and tedious process. If intermediate levels of description are introduced, producing a tree structure, then the search is made easier and more efficient.

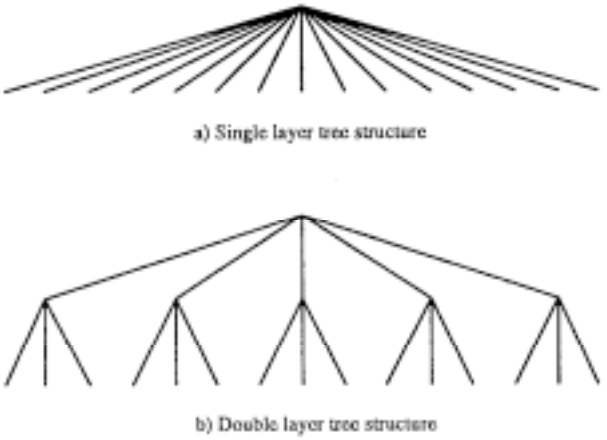

Figure 4. Tree diagrams showing addition of intermediate structures.

If we apply this to our previous problem of searching for word definitions, we might choose to put our definitions in alphabetical order of the first letter of each word. This can be imagined as a tree structure with a row of nodes representing each letter of the alphabet with a corresponding list of words under each. Thus if we know that the first letter of the word we are looking for is 'f then we only have to search the sub-tree under the 'f node. Further levels of organization can be introduced by ordering words within each sub-tree according to their subsequent letters until we have a dictionary! A more complete discussion of the problems of searching trees can be found in Korf (1987).

What Does the Designer Mean?

We have considered how large structures can be built, adding extra layers of description to a design. At this point it is useful to distinguish between *graphical* and *non-graphical* information. Graphical structures will primarily be created by the MOLE drawing machine and will consist of construction line, construction point and segment information.

Non-graphical information will consist of any structures representing facts about the object depicted by the drawing, rather than the drawing itself. These structures may be linked to elements of graphical information. Thus a designer may build a description of a house, consisting of a number of rooms, which in turn consist of a number of walls. As well as information about materials and dimensions, the structure may also point to the elements in the drawing which correspond to each of the walls, thus linking non-graphical and graphical information.

MOLE gives the designer the absolute freedom to link these two types of information in any way they wish, including the choice of adding no extra information to the drawing at all. If the designer were the only person to use the resulting knowledge structures then this would be no problem. However, certain complications arise when we try to perform some automated reasoning task upon the design description produced by the designer. The nature of such tasks will be varied, as it is one aim of our current research to show how a common formalism can be used as the basis of many applications. There may be different goals, one application may be giving feedback to the designer concerning how well their design conforms to certain regulations, another may be to assist the designer in complex calculations regarding the energy efficiency of a particular design.

Whatever the application, each will need to map different non-graphical concepts to the basic graphical objects in the drawing description. For example, our current project includes development of an application to judge how well a design conforms to fire safety regulations, and such an application will need to deal in terms of internal spaces, fire barriers and escape routes etc. It is unlikely that the non-graphical structures used by a designer will bear much relation to these. Even if the designer did have details of the structures used by a particular application, and structured her design description accordingly, problems would arise if the same design were to be used, perhaps at a later date, as data for a different one.

The precise nature of our problem, that of mapping structures from one domain to another will depend, to a large extent, on how much non-graphical information is supplied by the designer.

The Designer Doesn't Say - Analysis of Graphical Information

In some cases the designer may not have provided any information in addition to that produced by the graphics machine. Thus any description that exists refers purely to the drawing, and not to the object which the drawing is intended to depict. In some respect this problem is relatively simple and consists of imposing structure onto a structureless description. Thus the application might search through the drawing description searching for patterns which would fit the particular higher level descriptions it required.

This problem is similar to the high level pattern matching required of vision systems, and the models used to either explain human vision, or the implementation of computer vision systems may be of some help to us. In vision systems a mapping is sought between the representation of external stimuli produced by the low level vision system and a representation of the corresponding object in memory. In our domain, the graphical information, defined in terms of segments, can be likened to representations of the external stimulus and the high level, non-graphical structures the object representation in memory.

Amongst the many methods used for such processing (see Spoehr & Lehmkuhle (1982)- chapter 3 - for an overview) the *template* or *schema* based approach seems particularly suitable for our purposes. In this, memory contains a series of schemata representing prototypical examples of the objects which might be represented. A particular schema would contain information about those features shared by each member of the class. The external stimulus will be an instance of one of these schemata, and is matched by abstracting its representation until its class type is found.

A particularly pertinent example of schema driven image analysis is the ANON system (Pridmore & Joseph 1990). Designed for analyzing drawings of mechanical engineering drawings, it uses high level schemata to control all aspects of the interpretation process. A high level schema, describes general classes of objects within the drawing such as solid lines, junctions or outlines, and guides the interpretation process by specifying a set of rules for recognizing and combining lower level graphical entities. It is designed to deal with information at the pixel level, but the same approach is equally valid when dealing with analysis of MOLE descriptions.

In some respects our problem is easier than that faced by ANONs designers in that we do not have to try and decide which schema is the most appropriate to use. Our analysis will be goal driven, with the schema or schemata to be used in the interpretation of the graphical information depending on the task which the user has requested of the particular application. Associated with each task will be set of schemata, in the form of MOLE kinds, which describe the concepts which form the basis of that application along with some information about the geometry of these structures. For example, if one of the basic concept is a room, and associated with this is the information that rooms are rectangular, then when a user invokes an application task which refers to rooms, the analysis process will use the schema for rectangles in interpreting the graphical information in the design.

The use of such goal driven techniques will help to cut down the search space of the analysis process, ruling out a great many possible interpretations of the drawing that are irrelevant to the current task. However the process may still be slow, and subject to the ambiguities of interpretation that arise in all such systems. The process can be made more effi-

cient, and less error prone, by using any non-graphical information that the user has supplied in the design description.

The Designer Does Say - Mapping of Non-graphical Structures

There is no reason why the approach described above should not be used even if the designer has specified some non-graphical information, with the application simply ignoring it, solely using the drawing machine information and its own schemata. However the efficiency and accuracy of the interpretation process as a whole will be improved if the explicit links between graphical and non-graphical information that the designer has provided can be used. This shifts the problem of determining mappings between graphical elements in the users design description and non-graphical structures in the application domain to that of determining mappings between two sets of non-graphical structures.

Such problems have been studied extensively in the field of analogical reasoning (Gentner 1988, Keane 1988, Holyoak & Thagard 1989). In this type of reasoning, a particular domain, known as the *base is* understood, and acts as a source of information which can be used to help understand a second, poorly understood domain, known as the *target. A* classic example is that of the use of the solar system to explain the Rutherford model of sub-atomic structure, where the sun corresponds to the nucleus of the atom and the planets, in orbit around the sun, correspond to the electrons in orbit around the nucleus.

Despite many different approaches, all models include some way of specifying the semantic similarity between elements of the two domains. There may be differences in the level at which this applies. For example in Gentner's Structure Mapping' model (Gentner 1983, Gentner 1988) any similarities between elements at the lowest level are ignored, with only functions or predicates playing an active part in the mapping process. In contrast Holyoak and Thagard (Holyoak & Thagard, 1989) propose a model based on parallel constraint satisfaction which allows semantic information between any level of element to be taken into account.

The problem with this, as far as we are concerned, is that it assumes at least a partial knowledge of the target domain, and the relationship between its elements and those in the base. What we need to do is to determine whether we have this type of information available to us when mapping MOLE structures, and at what level this information might exist.
Unfortunately we can trust very little of what the designer tells us. Even if they build a structure that has the same name as one used by the application, there is no reason to suppose that they refer to the same thing. For example a structure that the designer considers to be two spaces divided by a wall might be considered a single space by a fire safety application if the dividing wall offers no resistance to the progress of a fire.

So our problem, when attempting to map non-graphical structures, is determining at what level can we ground two structures, and say that they represent the same thing. *As* a MOLE kind is defined by its constituent slots and fillers, we are going to have to start at this level. A starting point might be to say that two structures can be considered equal if they share a number of slots with common fillers. This general principle will need to be adapted to deal with task definitions, and further information will be needed to allow it to work efficiently. For example, some slots are more important than others in defining the characteristics of an object. In a kind describing a cat, the slots specifying dentition and

skeletal structure would be more characteristic and therefore more important than those specifying the animals color and fur length. Any mechanism for matching kinds by their constituent slots will need some way of weighting the slots so that matches between important slots are more highly regarded than matches between less important ones.

Pragmatic factors are also commonly considered in analogical mapping processes (e.g Holyoak & Thagard 1989). Thus features of a domain that are considered central to the behavior of that domain are more likely to be mapped than those which are not. It is likely that similar factors will be useful in guiding our mapping process. In the domain of fire safety rules, for example, escape routes, such as staircases of fire escapes, are particularly important structures and will be mapped preferentially to other less important ones.

It is important to point out that our concept of importance, both at the slot and structure level, only extend to the application domain. We have no idea of the relative contributions of the various parts of the user's design descriptions. Nevertheless, these principles, form the field of analogical reasoning will prove to be important and useful in our task of determining potential mappings between designer and task non-graphical information.

What Do We Do if it All Goes Wrong?

Associated with attempts to map application structures to the drawing description produced by the designer is the problem of knowing what to do if the process fails. The application may be unable to find, or impose, any organization on the description to match the structures it uses. In this case it is likely that the application will simply fail.

Alternatively, matching structures may be found, but in a way that does not correspond to the organization of the drawing originally envisaged by the designer. This is more likely to be the case if no high level structures have been defined, but may occur even if they have. The misinterpretation of drawings is likely to lead to the designer becoming confused, and frustrated with the system.

A simple method of solving these problems would be to force the designer to organize her drawings using the structures required by the application, thus removing the need for the application to perform any mapping at all. This however implies a level of prescription which is completely at odds with the philosophy behind the development of MOLE (Tweed & Bijl 1988). It also implies that a different structuring would need to be imposed on a drawing for each application was to be used for. Whilst MOLE could support such a structure it would lead to an over-complex representation. Further problems would also arise if a description was to be used for applications other than those intended by the designer.

Many of these problems will be alleviated, by introducing some degree of dialogue between the designer and the application. The use of dialogue in explanation is a topic that is currently attracting much attention, (e.g. Cawsey 1989) and it is likely that a similar approach will be useful in the process of determining the structure of a designer's drawing. Whilst the precise approach will depend on how we eventually decide to structure the designer/application interface, dialogue will help us to get extra information if the search for napkins cannot proceed; to provide the designer with feedback as to what interpretations are being applied; and to give the designer the final choice in ambiguity resolution. Some consideration of the nature of the dialogue will be essential. For example, getting a balance between too much interruption in the design process and getting too little information for the

mapping to be found will be particularly difficult. However, despite this, and other problems, a dialogue based approach will help considerably in the process of developing systems for automated reasoning with drawings.

Conclusion

It is important that the provision of design support tools within the MOLE environment does not impinge upon the flexibility of representation provided by that environment, despite the problems that this creates. We propose a system, based upon appropriate Al techniques, which will interpret design descriptions in any number of contexts, with the minimum of guidance from the designer themselves.

We suggest that the principles and problems at the heart of this system go beyond the interpretation of drawings by a single application and may be applicable to the wider problem of multi-user design systems. In any situation where more than one person is working on a design the problem of communication between them arises. In the case of pure pencil and paper designers, then there are conventions and symbols for specifying what different parts of a drawing mean. To an extent the problem is solved outside the design system itself. In developing MOLE for such situations we could take the same approach, insisting that designers communicate about their meanings outside the MOLE environment itself. However we can apply the principles used in mapping between designer and application structures to the case where we want to determine mappings between structures developed by two designers. The problem will be a lot less constrained, in that we have little knowledge about what either of the designers mean, and thus more complicated. However it would make an interesting and worthwhile extension to our current work.

Acknowledgments

The author is supported by SERC research grant GR/F61868.

References

Cawsey, A. J. 1989. 'Explanatory Dialogues." Interacting with Computers, no. 1: 69-92.

Gentner, D. 1983. "Structure-mapping: a theoretical framework for analogy." Cognitive Science, no. 7: 155-170.

Gentner, D. 1988. "Analogical Inference and Analogical Access." In A. Prieditis (ed.). Analogica. London: Pitman.

Holyoak, K. J., and P. R. Thagard. 1989. "Analogical Mapping by Constraint Satisfaction." Cognitive Science, no. 13: 295-355.

Keane, M. 1988. "Analogical Mechanisms." Artificial Intelligence Review, no. 2: 229-250.

Korf, R. 1988. "Search: a survey of recent results." In H. Shrobe (ed). Exploring Artificial Intelligence: survey talks from the national conferences on artificial intelligence. Morgan Kaufmann.

Pridmore, T. P., and S. H. Joseph. 1990. "Integrating Visual Search with Visual Memory in a Knowledge Directed Image Interpretation System." In Proceedings of the British Machine Vision Conference 1990.

Ramscar, M. J. A. 1991. Counterpart relationships in MOLE. Technical Report. EdCAAD, Dept. of Architecture, University of Edinburgh.

Spoehr, K. T., and *S*. W. Lehmkuhle. 1982. *Visual Information Processing*. San Fransisco: W. H. Freeman & Co.

Steel S. W. D., and P. J. Szalapaj. 1983. Pictures *without numbers: graphical realization of logical models*.

Technical Report, EdCAAD, Dept. of Architecture, University of Edinburgh.

Tweed, C., and A. BijI. 1988. 'MOLE: a reasonable logic for design." In P. J. W. ten Hagen, T. Tomiyama and V. Akman (eds). *Intelligent CAD Systems 2*. Berlin: Springer-Verlag.

White, R. H. 1990. *MOLE or ART - which way forward?* Technical Report, EdCAAD, Dept. of Architecture,

University of Edinburgh.

White, R. H., and M. J. A. Ramscar. 1990. A Quick *Guide to Mole2*. Technical Report, EdCAAD, Dept. of Architecture, University of Edinburgh.