Visual Abstraction in Analogical Problem Solving: A Dissertation Proposal

Jim Davies

Georgia Institute of Technology, College of Computing, Atlanta, Georgia.
jim@jimdavies.org

Abstract. Analogical problem solving with visual representations is important in many situations. I propose to develop a computational theory of visual analogy that will 1) support the notion that visual abstractions are useful for analogical transfer in cases where there are symbolic mismatches at the non-visual level, and 2) provide a language of primitives that will prove useful for visually representing problems and problem solving procedures. Support for these endeavors will come from how the theory predicts and explains test data, comparison with published experimental participant data, comparison to non-visual accounts, implementation of the theory in a computer program, and experiments with it.

1 Introduction and Problem Statement

Visual representations are often important for problem solving (Schrager, 1990; Farah, 1988; Casakin & Goldschmidt 1999; Monaghan & Clement, 1999). For example, problem solving can be facilitated by animations (Pedone et al., 2001), and visually evocative phrases in stimuli (Gick & Holyoak, 1980; Beveridge & Parkins, 1987). There is also anecdotal and documentary evidence for visual thinking in science (Miller, 1984; Nersessian, 1984; 1992; Gooding, 1994; Shepard, 1988; Thagard & Hardy, 1992; Boden, 1990). But the details of how people use visual resources in problem solving is hardly understood.

Some visual reasoning involves analogical problem solving, which is gaining knowledge about some *target analog* by transferring it from a base or *source analog*. By *visual analogy* I mean analogical reasoning with visual knowledge. This work will focus visual properties such as spatial relationships between objects, shapes and sizes, rather than on textures and colors.

This dissertation will deal with two main problems. The first is that we do not know the conditions under which visual analogy is useful. The second is that we don' t know *how* to represent visual information, or exactly *what* visual information to represent for computational purposes. What kinds of visual symbols are useful for analogical problem solving? Which symbols should be used?

A sub-problem of analogical problem solving is the symbol mismatch problem. A symbolic mismatch is when processing is hindered because the symbols representing two things are not the same.

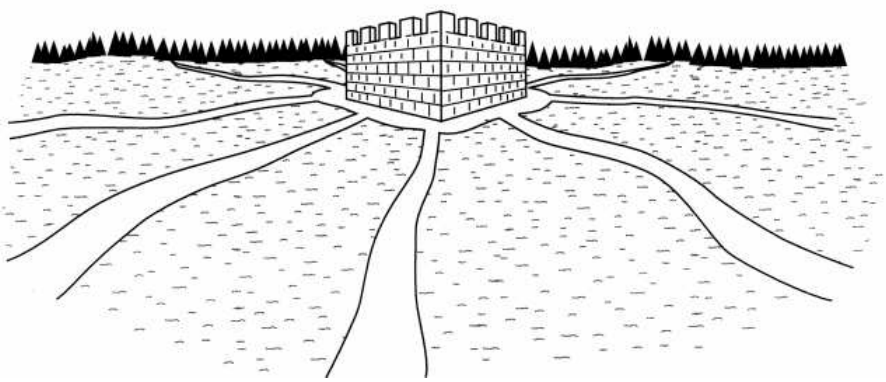

Fig. 1. This figure shows the Fortress. Experimental participants first read a story about a problem-solving situation: A general with a large army wants to overthrow a dictator who lives in a fortress. All roads to the fortress are armed with mines that will go off if many people are on them at the same time. To solve this problem the general breaks up his army into small groups and has them take different roads. The groups arrive at the same time and take the fortress.

Let us examine Duncker's classic fortress/tumor problem as an example situation where symbolic mismatches could cause an analogical problem solving agent to fail

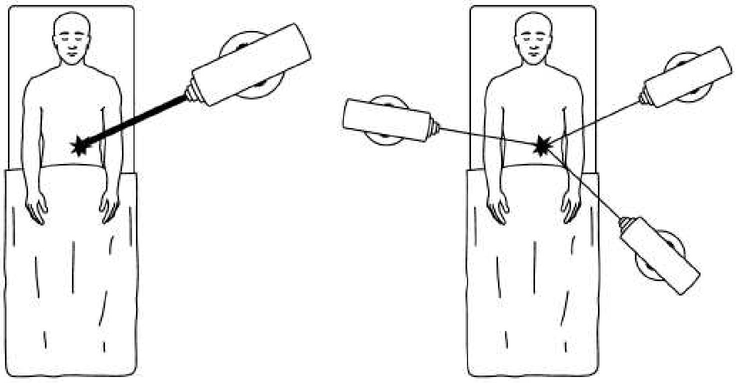

Fig. 2. Participants are given a new problem: A patient needs radiation treatment on a tumor inside the body, but the radiation will harm the healthy tissue it reaches on the way in. Finally, the participants are asked to solve the tumor problem. The analogous solution is to target the tumor with low-level rays coming from different directions, and have them converge on the tumor.

(Duncker, 1926). Imagine the agent knows of a solved problem which involves breaking up an army into smaller groups. The army is, quite reasonably, represented as a group of constituent soldiers. The target problem involves a ray of radiation which must be turned into a number of rays with less intensity (see Figure 2). The ray might be represented as energy, with a number associated with its intensity, a representation chosen to serve a different function (e.g. so that numeric intensities can be added). Not having anticipated that the ray and army might need to be aligned, they could have been encoded with incompatible representations.[1]

Symbolic mismatches can be encountered during analogical retrieval, mapping, or transfer. In this example we have two symbolic mismatches. First, without some similarity (perhaps through their relational structure), the ray and the army, being different symbols, cannot be aligned. They are, semantically, rather distant. But even if this alignment problem is somehow overcome, the agent would still have a problem with transferring the solution strategy. The transformation applied to the army will not work on the ray because the the representation of the ray, in this example, does not have constituent parts. Breaking something into parts is different from dispersing energy.

The point of this example is to show that a reasonable non-visual representation can fail for analogical problem solving. It is possible to represent this problem with no symbolic mismatches (Holyoak & Thagard (1989) do so), but symbolic mismatches are bound to occur in any large knowledge base (Lenat & Guha, 1990).

I propose to develop a theory of visual analogy that addresses these problems, and a representation-level model to show how it could work. In the next two sections I will describe the work done so far in this regard, and lay out a plan for its completion. In my

[1] This problem has been identified by Yarlett and Ramscar (2000), whose system takes two different symbols and evaluates their similarity using Latent Semantic Analysis (Landaur, 1998). The analogical mapper treats as identical any symbols that are evaluated as sufficiently similar.

theory evaluation section I will describe the planned computer implementation, along with other planned evaluations.

2 Theory

This dissertation is based on the general idea that visual representations provide a level of abstraction at which two otherwise dissimilar domains may be more alike. There are many theories that also resolve symbolic mismatches by finding similarities at a higher level of abstraction. For example, in conceptual dependency theory (Schank 1972), verbs are categorized into ACTs, which are abstractions of actions. Bhatta and Goel's Generic Teleological Mechanisms (1997) cover different instantiations of mechanisms that perform the same function. Falkenhainer's Minimal Ascension (1988) rule uses a generalization hierarchy to determine the distance between concepts. This dissertation is also research of this kind. I will show that visual abstraction too is a useful mechanism for analogical problem solving.

The first problem is this: under what conditions are visual analogies useful? This dissertation will argue that one of the ways visual representations are useful is in resolving symbolic mismatches in analogical problem solving. Symbolic mismatches encountered in non-visual representations are resolved by providing a level of visual abstraction at which two different symbols are similar.

With respect to the second problem, that of representing visual information for computational purposes, this research introduces *Covlan*, a language of visual primitives, and will show that it is effective for representing problem-solving episodes involving physical systems, to facilitate analogy. It will consist not only of a language of visual primitives, but also rules for turning non-visual representations into visual ones.

The scope of this work is problems involving physical systems, which I define as those whose solutions involve physical changes to the state of the system in question. For example, solving a problem by changing ownership of something, or by changing a power relationship among people is not a physical change, but physically moving someone's chair so that they sit somewhere else is. *System* in this work refers to all the physical things associated with a problem situation, the relationships between them, or the spatial representation of those things.

This scope was chosen because it was large enough to be interesting but small enough such that my theory could generalize within it. This is not to say that visual representations are not useful for problem solving in non-physical systems, just that this work will not show evidence for it.

To summarize, at the highest level my hypotheses are:

1. Analogical problem solving using analogs that are representations of physical systems can be facilitated by represention of those systems in terms of visual information.
2. In particular, transformation into visual information (visual instantiation) is useful when there is a symbolic mismatch at the non-visual level. This occurs for both object representations and the transformations that are applied to them.
3. My visual representation language offers an explicit way to transform systems into representations using visual information in a way that enables analogical transfer.

To make comparisons between visual and non-visual representations, it is important to be precise about what I mean by "visual." I will use the terms "visual representation" and "visual knowledge" interchangeably, and by them I mean any representation that encodes *visual information*. Visual *information* are the visual properties of something, and visual *knowledge* is visual information encoded for use by an intelligent agent with some representation language. Specifically, this dissertation deals with the following kinds of visual information: shapes, their sizes, locations, motions, and spatial relationships between shapes (e.g. connections, overlaps).

This dissertation will use symbolic *descriptive* representations, which are structured descriptions of visual information. This is differentiated from *depictive* representations, or bitmaps. A depictive representation "specifies the locations and values of points in space" (Kosslyn, 1994, p.5). There is widespread agreement that visual reasoning, particularly in problem solving and analogy, is a symbolic process. Not surprisingly, all previous computational visual analogy programs also use symbols to represent visual information (Thagard et al., 1992; McGraw et al., 1993; Ferguson, 1994; Croft & Thagard, forthcoming). A more detailed discussion of visual representations will follow in the discussion section.

2.1 Resolving symbolic mismatches

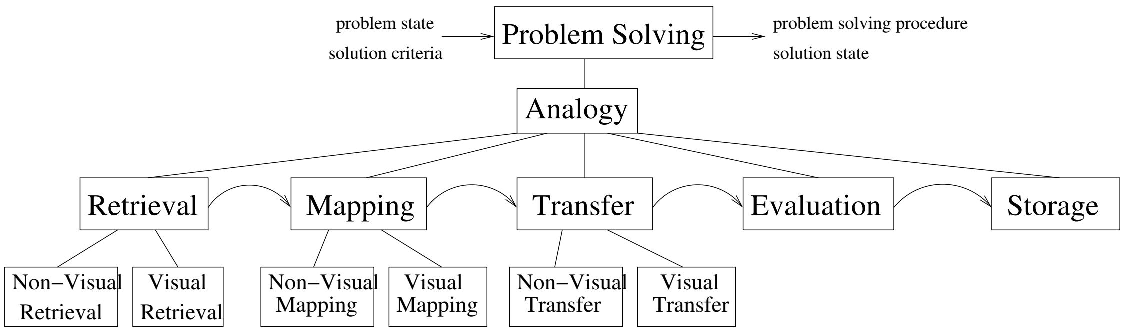

Fig. 3. This figure shows a high level description of my theory of the role of visual reasoning in analogical problem solving. Straight horizontal arrows represent input (arrows entering a box) and output (arrows exiting a box.) Boxes represent complex actions to be taken by the agent. Curved arrows represent an ordering relation. A series of boxes connected with curved lines represent a series of ordered subtasks of the higher task, connected with a vertical line. The order is from left to right. Boxes below a task that are unconnected to each other are not subtasks but alternative methods for achieving the task in the box above it. Realistically, there is looping in the analogical process; this will be detailed in the model section of this proposal.

Analogy is one among many ways to find a problem solution. For example, if an identical problem has been encountered before, that solution might be retrieved directly.

In Figure 3, analogy is a method for a problem solving task. Analogy consists of several steps: *retrieval* of a candidate source analog in memory; *mapping* the components of the analogs; *transfer* of knowledge from source to target; *evaluation;* and *storage* of the target in memory, perhaps to be used as a source analog later. Much of this dissertation will involve changing the representations of the analogs, which is a non-essential but often useful process to prepare the analogs for one of the above core steps.

Retrieval and Mapping. Visual information can be used to retrieve memories. Since visual cueing can occur, it is reasonable to think that some memories are encoded in terms of perceptual information. This is so well accepted that there is a theory that *all* memories are encoded in terms of perceptual information (Barsalou, 1999).

In analogical problem solving, a visual representation of the target can be used to retrieve visual representations of potential sources. An analog can be represented in terms of process, causality, uses, etc.; the visual representation is one of many. The mind will use these connected, different representations for retrieval when relevant.

Visual representations can be generated to aid in retrieval. For example, if many people are making demands of you, the sensory experience is multi-modal and complicated. However, in trying to understand the situation you might generate a visual abstraction to represent it. For example, the demands might be represented as converging red lines on a circle, representing you, perhaps triggering other memories regarding convergence.

Mapping two analogs involves aligning their elements. Since retrieval is based on matching, to some extent, it is reasonable to suppose that the processing done to retrieve an analog can be used to guide mapping (Holyoak & Thagard, 1997; Falkenhainer et al., 1990).

Since I am dealing with problem solving, and the analogical transfer of problem solving solutions, retrieval queries are based on the initial problem state and the solution constraints. See Figure 4.

Mapping visual analogs may differ from mapping of non-visual analogs in two ways. First, visual analogy can use visual knowledge, such as an abstraction hierarchy (e.g. a square is a kind of rectangle). Such knowledge can be used to find matches based on similarity. Second, the agent's perceptual system can be brought to bear to inform the mapping. Seeing a truck may have something to do with matching its parts to the parts of some episode of truck experience in memory. Likewise, one might help guide mapping by using perception on generated mental images (Kosslyn 1994).

Transfer. As in some other theories of analogical problem solving, in my theory problem solving strategies are transferred. Other theories of *visual* analogy do not do this.

As shown in Figure 4, the possible source analogs are represented as solution procedures connecting knowledge states. In analogical problem solving, transfer is using a source analogy's solution strategy for the target. This can be done with both visual and non-visual representations.

Transfer works as follows: In the mapping stage, a mapping is found between the source and target initial problem states. The manipulation that connects the first to the

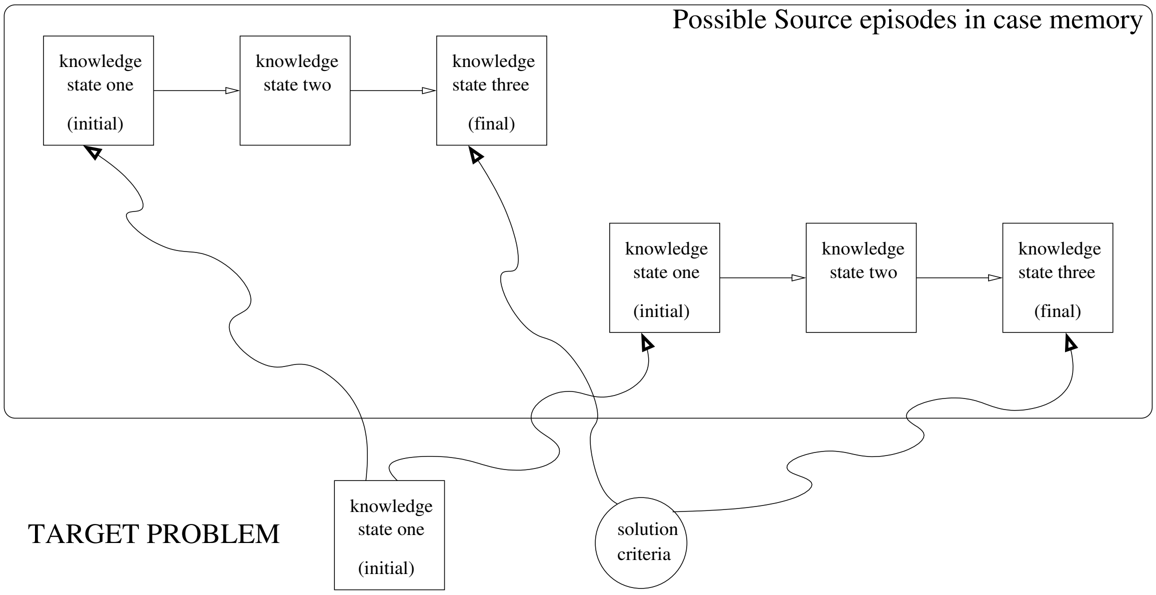

Fig. 4. This figure shows a target analog problem and how it relates to the potential analogs in the case memory. The items in the case memory are represented here as a series of knowledge states (represented by boxes) connected with manipulations, which are changes to the knowledge state (represented by straight arrows pointing right.) The last box in a series is the solution state, and the arrows, in order, represent the solution procedure. The target only has a single knowledge state because there is no solution procedure yet. There is also no solution state, but rather a set of criteria that must be fulfilled. The initial target knowledge state is compared to the initial states of the cases in memory for similarity (shown as wiggly arrows coming from the target knowledge state). Also, the solution criteria is compared to the solution states in memory to see if they fulfill the criteria (shown as wiggly arrows coming from the target solution criteria). Based on these measures cases are retrieved.

next knowledge state in the source is transferred to the target. The parts of the target that the manipulation affects are those analogous parts of what get affected in the source.

When this is done with visual representations, transfer of manipulations can work because they are sufficiently abstract such that they can apply equally well to many different elements. For example, a manipulation that *moves* something can apply to lines as well as circles. This means that the same *move* manipulation that worked in the source with a circle can work with the line in the target.

This process can repeat unhindered for the entire sequence, transferring the manipulation from the source and generating new knowledge states in the target. Sometimes, however, there can be problems with symbolic mismatches. For example, as discussed above with the fortress/tumor example, trying to transfer the *break-up* manipulation to the ray will not work because the ray does not have constituent parts.

Visual representations can be used as an intermediate level of abstraction to do plan *adaptation*. To follow the example, imagine the advancing army gets visually instantiated as a *line*, and the ray does as well. The manipulation, too, gets visually instantiated as the *decompose* visual transformation, which applies fairly broadly to visual elements. In the generated visual representation, the transfer of the manipulation occurs without a problem, as *decompose* can apply equally well to both lines. I call this visual representation an intermediate step because it must be turned back into the non-visual again, because, I assume, "solving" the problem in the visual abstraction is really not informative of actions that must be taken in the real world (more on this in the solution evaluation section). When the visual transformation is turned into a manipulation for the ray in the non-visual representation, it becomes a different manipulation: *disperse-energy*. Because transformations can specify into different actions, strategies can be adapted to new instances, and steps in the problem solving process do not need to be transferred literally.

Solution Evaluation. In my theory the target problem has only a single knowledge state. The "goal state" is not represented at the start. I am dealing with insight problems, for which I assume a clear picture of the goal state is practically all one needs to solve the problem. Most of the work is in finding out what the goal state is, as opposed to how to get there. Rather, the "solution" is represented in terms of criteria that determine whether the problem is solved.

An agent cannot tell if the problem is solved by examining the visual representation. An agent needs to turn it back into a non-visual representation and run a simulation to determine the effectiveness of the manipulations made. For example, moving the weaker rays around and pointing them toward the tumor cannot be identified as an adequate solution to the problem unless the agent understands that the result of this would be that the tumor is destroyed while leaving the healthy tissue unharmed. Then the goal criteria can be applied. Once the knowledge state is non-visual again, its workings need to be simulated to be able to test it against the goal criteria. The agent maintains correspondences between elements of the knowledge states throughout the transfer process, and is able to use that information to return the final solution state back into a non-visual representation. Simulation means predicting the behavior of a system given knowledge of how it works.

Thus to simulate, the agent needs causal knowledge. Neither this causal knowledge, nor the goal criteria can be represented with only visual information, as causality consists of more than visual relationships between things.

By causal I mean knowledge of how things in a system change as they interact. Pre- and post-conditions are a straightforward way to represent this, but it is difficult to imagine what "visual" pre- and post-conditions might look like. For the reasons above I hypothesize that visual representations alone cannot enable evaluation of the solution.

Solution Storage. Newly created knowledge state series are stored just like source analogs so they can be used as such in the future.

3 Model

The previous section described my overall theory about analogical problem solving and the role of visual analogy. In this section I will flesh out the parts of the theory that I will evaluate.

According to my theory, episodes of problem solving can be represented as a sequence of *knowledge states* in the problem solving procedure, connected by manipulations between them. *Manipulations* are operations the agent can take on the system to change it. These are distinguished from *simulation events,* which occur in the system on their own. For example, the workings of a clock are simulation events, and winding the clock is a manipulation. Applying a particular manipulation to a knowledge state results in another, changed knowledge state. They are states in the problem solving process.

Problems and solution procedures can be represented non-visually and visually. In the non-visual representation, the knowledge states are called *nv-states*, and the manipulations are called *actions*. Both actions and nv-states can be transformed into visual representations. This is done using the *Cognitive Visual Language*, Covlan, which provides the vocabulary and processes for turning non-visual physical system representations into visual ones. Covlan knowledge states and manipulations are called *s-images* and *transformations*, respectively.[2]

Unsolved target problems are represented as single nv-states or s-images.

Table 1

	Knowledge State	Entity	Manipulation
non-visual	nv-state	object	action
Covlan	s-image	element	transformation

First I will describe the representation language, then the inference and control of the theory.

[2] The terms "entity," "element," "object," "manipulation," "action," and "transformation" are used merely to differentiate the visual, non-visual, and super-ordinate counterparts of things and operators. Their common sense meanings in English do not have anything to do with which term gets used with which meaning.

3.1 Knowledge Representation: Covlan

Medin and Ross (1990) report that experts *qualitatively* represent large knowledge bases in memory. Doing this for visual information requires an ontology of visual primitives. I am designing Covlan, a language to describe visual analogs. It can represent diagram-like images, relationships between them, and changes to them. I will also design a non-visual language, based on the SBF modeling language, but the details of this non-visual representation will not be important for my theoretical claims.

I will make choices of what to put into Covlan based on the following constraints:

1. I have some data from experiments run by Dr. David Craig. The details of these data can be found in appendix A. Diagrams made by experimental participants in the Craig study provide information about what abstractions are useful for representing the systems in question. I will say more about these data in the theory evaluation section.
2. Primitives from research that suggest a visual vocabulary such as geon theory (Beiderman & Cooper, 1991). As this is a cognitive theory, psychological research will also constrain it.
3. Certain choices in what will go into the theory will be determined by what is needed to get the program to function correctly. For example, to represent the fortress/tumor problem's solution, I needed to break the solution procedure into steps, so that they could be transferred. The steps chosen should be general enough such that they might be useful for other problems. Thus the choices of transformations, for example, were constrained by what was needed. I will make similar decisions with the examples from the Craig data.

Covlan consists of the following kinds of entities: *S-images, transformations, elements (primitive* and *complex), miscellaneous slot values,* and *relations.* I will describe these in turn.

S-images: Symbolic Images. Knowledge States in Covlan are symbolic images, or *s-images*, which are visual descriptions containing visual elements, miscellaneous slot values, and relations between them. Problem solutions are represented by a series of s-images, connected with transformations. Figure 5 is a diagram of an s-image for the fortress problem.

Transformations. S-images in the sequence are connected to other s-images before and after it with transformations. Transformations, like functions, take arguments to specify their behavior. Making topological changes of this kind to imagined physical systems has been shown in earlier work to be useful in problem solving. (Griffith et al., 2000; 1996). Table 1 shows some examples of transformations.

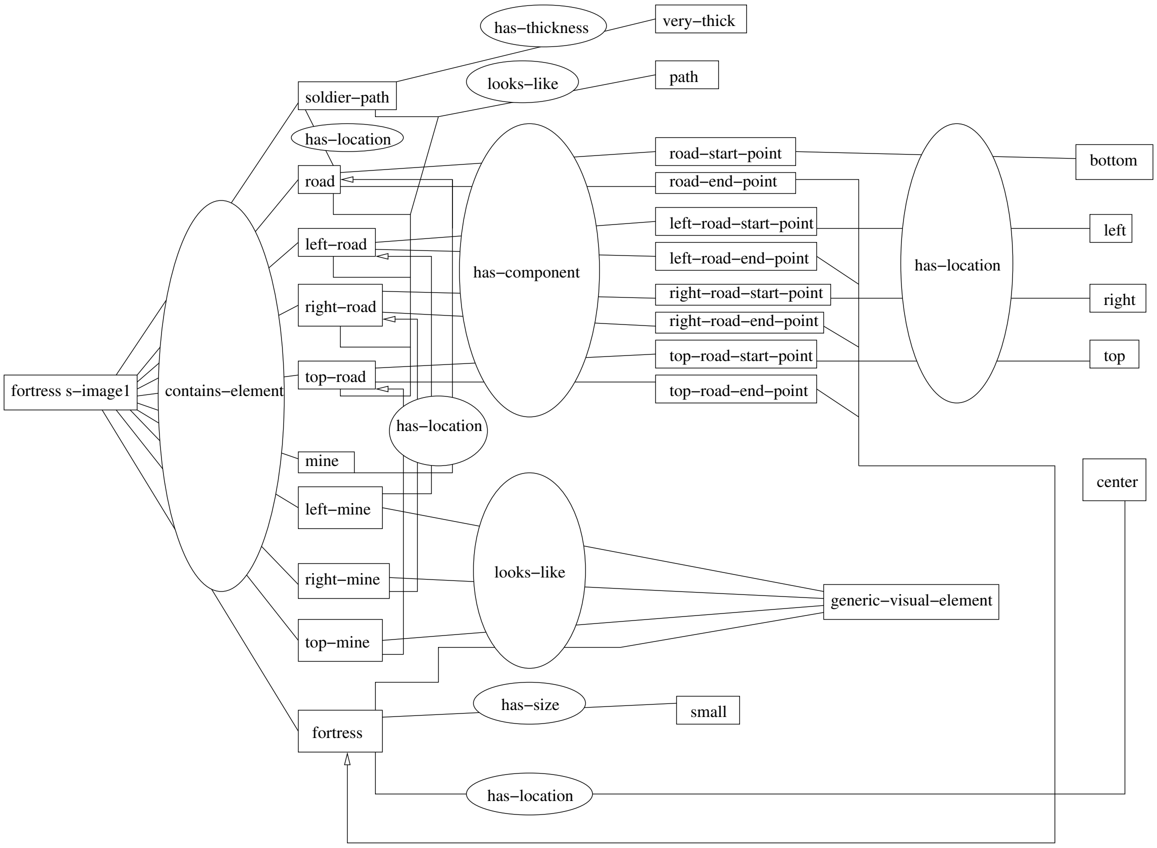

Fig. 5. An s-image of the problem state of the fortress problem. Things in ovals are relations. All connections are directional, going from left to right unless indicated differently by an arrow.

Table 2	
Transformation name	**arguments**
move-to-location	object, new-location
move-to-touch	object, object2, new-location
move-above	object, object2
move-to-right-of	object, object2
move-below	object, object2
move-to-left-of	object, object2
move-in-front-of	object, object2
move-off-s-image	object, location
move-to-set	object, object2
rotate	object, direction
start-rotating	object, direction
stop-rotating	object
start-translation	object, direction
stop-translation	object
set-size	object, new-size
add-element	object, location (optional)
remove-element	object
decompose	object, number-of-resultants, type
scale	object, new-size

These transformations control normal graphics transformations such as translation (*move-to-location*, *move-to-touch*, *move-above*, *move-to-right-of*, *move-to-left-of*, *move-below*), rotation (*rotate*), and scaling (*set-size*). In addition there are transformations for adding and removing elements from the s-image (*add-element*, *remove-element*). Certain transformations (*start-rotating*, *stop-rotating*, *start-translation*, *stop-translation*) are changes to the dynamic behavior of the system under simulation. For example, *rotate* changes the initial orientation of an element. In contrast *start-rotating* sets an element in motion. In simulation, if something is touching it, there would be friction.

Glasgow and Papadias (1998) also came up with primitive visual functions for spatial reasoning. They are: retrieve, put, find, delete, move, turn, focus, unfocus, store, and adjacent (p. 186). My choice of transformations will be constrained by the findings of this work and others like it (Thagard & Hardy, 1992 added the primitive “surround” to this ontology.)

Primitive Elements. S-images contain both primitive and complex elements. *Primitive elements* are the smallest units of a picture. All visual representations are made of combinations of these elements. *Visual Instantiation* is turning a non-visual representation into a visual one.

Two ideas should be associated with a visual abstraction only if the inferences the agent can make and actions the agent can do with the abstraction can also be done to the referents of the ideas themselves. Said another way, the abstraction captures visual similarities of the their referents that are relevant to simulation and manipulation.

Table 3	
Primitive Element name	**attributes**
polygon	location, size
rectangle	location, size, height, width, orientation
triangle	location, size, height, width, orientation
ellipse	location, size, height, width, orientation
circle	location, size, height
arrow	location, length, start-point, end-point, thickness
line	location, length, start-point, end-point, thickness
point	location
curve	location, start-point, mid-point, end-point, thickness
text	location, length, letters

Table 3 shows a list of the primitive elements.

The primitive elements are organized into a two-tiered hierarchy. Rectangle and triangle are subclasses of polygon; circle is a subclass of ellipse.

Complex Elements. Complex elements are composed of two or more visual elements (primitive or complex). Like primitive elements, complex elements have attributes.

Primitive Elements are intended to be be common to all visual agents. Unlike primitive elements, many complex elements can be domain-specific, and differ from agent to agent.

The following is a first pass at what the complex elements might be based on an examination of Craig's diagram data.

1. Multiple-pass line
2. Dual line arrow
3. Semi-circle
4. arrowhead-isosoles-triangle
5. arrow-curve
6. arrow-line
7. fringe
8. two-line line
9. curly-cue
10. shading
11. dotted line
12. wiggly line
13. circle
14. dotted circle
15. dotted curve
16. curved fringe
17. box
18. curly braces
19. zig-zag
20. cylinder

21. box door closed
22. box door open
23. line door closed
24. line door open
25. triangle
26. text alphabet

Miscellaneous Slot Values. These are symbols that can give a value to element attributes or a transformation argument. They can be broken down into the following types: *locations*[3], *sizes*, *thicknesses, numbers*, *speeds*, *directions*, and *lengths.*

The *locations* are absolute locations in an s-image to anchor where things are. I expect there will be two levels of abstraction: a high qualitative level with symbols like top, bottom, center, etc. and a lower finer-grained level that can be zoomed into when necessary. The lower level of locations will be based on an estimate of the resolution of the visual cortex, based on results from perception and brain studies.

Table 4: Classifications of Miscellaneous slot values

locations	bottom	top	right	center	off-bottom	off-top	off-right	off-left
sizes	small	medium	large					
thicknesses	thin	thick	very-thick					
speeds	slow	medium	fast					
directions	left	right	up	down				
lengths	short	medium	long					

Primitive Visual Relations. This class of symbols describe how certain visual elements relate to each other and miscellaneous slot values.

Visual Relations

1. **touching.**
2. **above-below.**
3. **right-of-left-of.**
4. **in-front-of-behind.**
5. **off-s-image** specifies that a part of the element is not visible due to being partially off the image.

Motion Relations

1. **rotation** has the arguments speed and direction.
2. **translation** has the arguments speed, direction, and destination.

[3] Relative locations are classified under *primitive visual relations.*

Analogy Representations. Knowledge states (both s-images and nv-states) can have *analogies* between them. Each analogy can have any number of analogical *mappings* associated with it (determining which mapping is the best is the *mapping problem*.) Each alignment in a given mapping is called a *map*.[4]

Similarly knowledge states next to each other in sequences have *transform-connections.* These are necessary so the agent can track how visual elements in a previous knowledge state change in the next. A difference between analogies and transform connections are that there can be multiple analogical mappings for an analogy, but only one mapping for a transform connection.

Transformations are attached, in fact, to a map between two components of sequential knowledge states. So if a rectangle changes into a circle, the agent knows which rectangle turns into which circle.

3.2 Inference and Control

A general hypothesis of this work is that visual information will prove to be useful in resolving symbolic mismatches. In this section I will specify this hypothesis: Visual information will help resolve two different kinds of symbolic mismatches in analogical problem solving. First, it can help with the generation of mappings. Take, for instance, where the agent needs to find an alignment between the army and the ray of radiation. For the sake of simplicity, let's say the army is represented with the token *army* and the radiation is represented with the token *ray*. The tokens are not identical, and the system cannot align them. The situation is depicted in Figure 6.

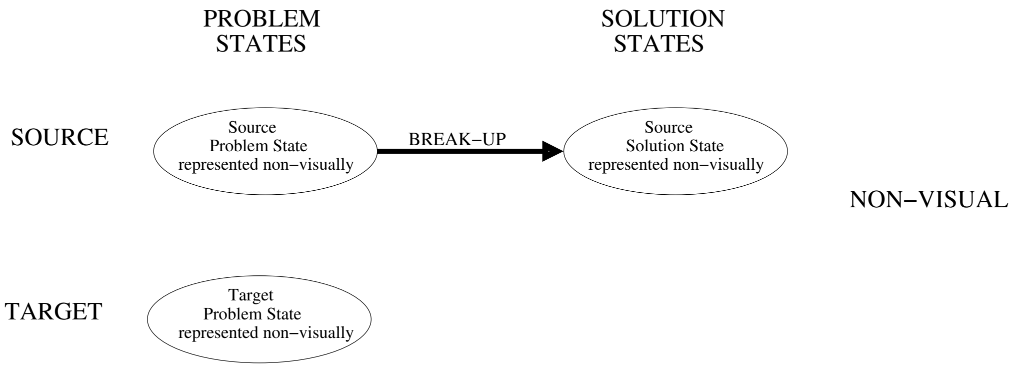

Fig. 6. The ovals along the top represent the solved problem in memory. For the sake of simplicity, imagine that the solution to the problem involves only a single transformation. The top left oval represents the start nv-state. The action *break-up*, splits the army up into smaller groups, resulting in many smaller armies. Every oval in this figure represents a non-visually represented s-image. The problem state on the bottom is the tumor problem.

[4] A map is called a *match hypothesis* in the SME literature.

Now imagine the agent has knowledge that both the ray and the army share the visual abstraction *line*, because a ray is shaped like a line, and the army's motion can be abstracted into a line-shaped path. The agent generates a visual representation of both systems and finds that there are lines in each, and makes the alignment as a result of this found similarity. This alignment is brought back to the non-visual representation and the analogical problem solving process continues in the non-visual representation.

The second use I predict is in the visual abstraction of actions. Let's say that to solve the problem the agent needs to break up the army into smaller armies, and the action it uses to do this is *break-up*, which takes a set of things as an argument and outputs smaller groups. Further, suppose break-up works by finding the constituent parts of the idea, and dividing them into n groups. If the ray of radiation is represented such that it does not have constituent parts, then the action *break-up* will not work on it. You can see the state of the agent's knowledge at this point in Figure 7.

The agent can visually instantiate the transformation into one that can be applied to lines. Let's suppose it visually instantiates into the visual *decompose* transformation, which will turn a line into several thinner lines. This visual transformation can be applied to both source and target s-images.

Next comes *specification*, which is translating the transformation back into an action in the non-visual representation, which I assume is a more realistic representation. That is, if you can't translate it back from the visual abstraction, then you won't really know what to do to the radiation, just a line representing it.[5]

But if you can trivially translate back and forth, then there is no need for the visual abstraction. What makes the visual transformation useful is that it does not specify back into *break-up*. The *decompose* transformation specifies into *break-up* when dealing with, perhaps, entities with constituent parts, but when dealing with something like energy, whose intensity might be represented by a number, it specifies into a different action. Let's call this new transformation *distribute*, which divides an intensity level by some number m and allocates the intensity to several sources of energy. This final nv-state is depicted in Figure 9.

In summary, according to my theory the visual abstractions can provide an intermediate representation through which otherwise dissimilar entities and manipulations can be aligned.

See Figure 10 for a diagram of the problem solving in the general case, when everything can be substituted directly. If there is a failure in this direct substitution, due to a symbolic mismatch, then the agent can use visual instantiation to generate visual representations to attempt to resolve the mismatch. The top ovals represent the source analog problem, where the top left oval is the initial state and the top right oval is the solution state. The bottom left oval represents the input target problem state. From the

[5] It may well be that you cannot understand how to solve a problem in a physical system without some perceptual representation of it. The visual representations I'm dealing with in this dissertation are more abstract than a full-blown, pictorial image. You can imagine how to decompose a ray of radiation quite realistically, and solve a problem with this in mind, but the visual level I'm talking about is more akin to a sketched diagram than an instructional video. They are so abstract that they are often ambiguous as to what they represent (e.g. a circle representing a person).

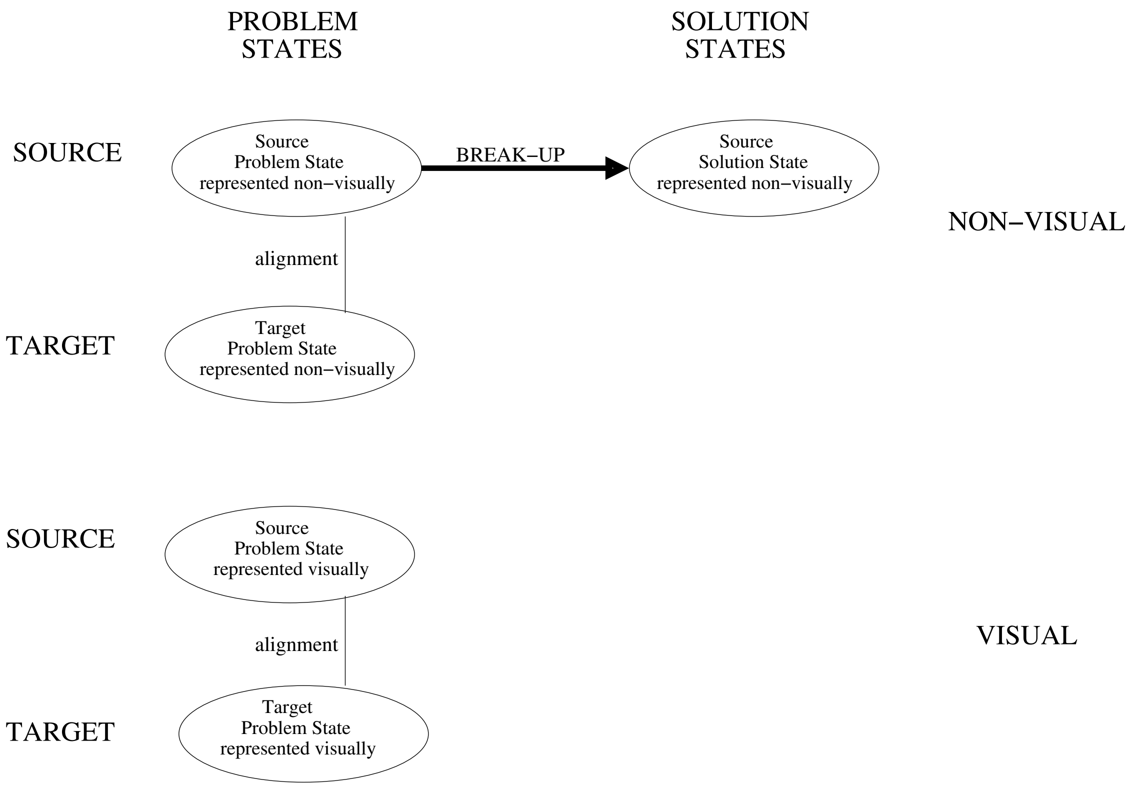

Fig. 7. The bottom two ovals are the problem states of the target and source in terms of their visual information. As a result of their visual similarity, an alignment, or mapping, can be found. This allows the agent to hypothesize the analogous alignment in the non-visual representation. From here, the agent can attempt the transfer process in the non-visual representation. It will fail because the action *break-up* cannot be applied to the ray of radiation.

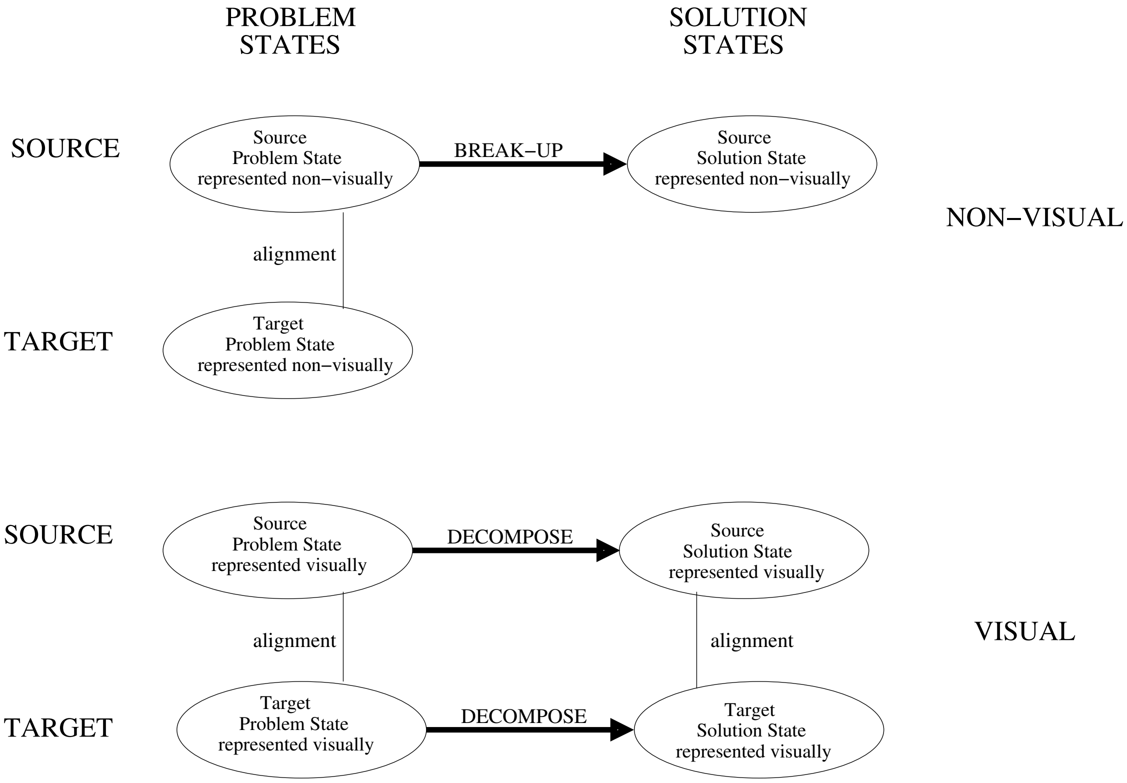

Fig. 8. To resolve this transfer of strategy problem, the agent continues fleshing out the visual representation, visually instantiating the *break-up* action into the visual *decompose* transformation. Since the ray and the soldier path are abstracted as lines, the decompose function can be transferred from the source to the target in the visual. The visual solution state can be generated. But solving the problem in a visual abstraction does not mean solving it in the more realistic non-visual representation. But if *decompose* specifies to *break-up*, then the visual representation does no good, because we could have just substituted *break-up* in the non-visual to begin with.

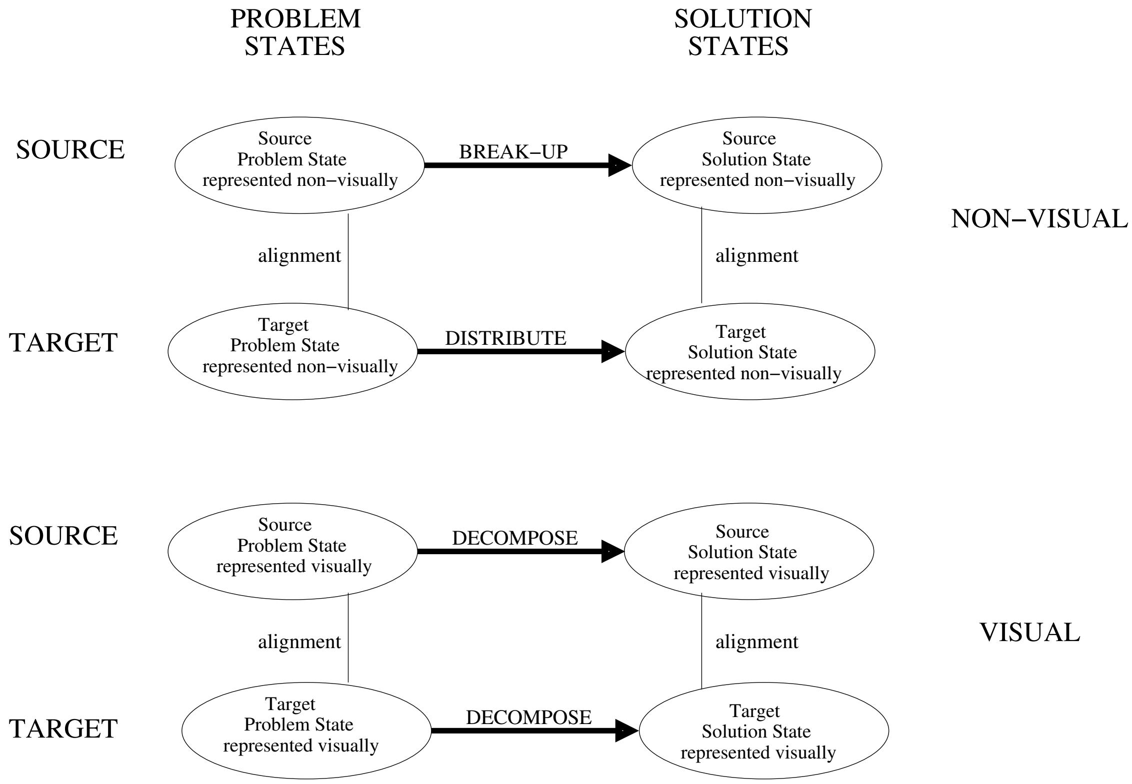

Fig. 9. The *decompose* function specifies to more than one non-visual transformation. The agent chooses the correct specification based on the kind of object it is modifying. In this case, it's energy, so it chooses the more appropriate *distribute* transformation. It decomposes the ray correctly. The target solution state represented non-visually is generated and evaluated. The problem is solved.

mapping stage, there is an analogical mapping between the source s-image 1 and the target knowledge state 1.

The same transformation that brings the source s-image 1 to the source s-image 2 is applied to the target s-image 1, leading to the generation of the target s-image 2, shown as the oval at the bottom middle. Then the next transformation is transfered, until all the transformations from the source have been applied to the target, resulting in the solution state of the target problem.

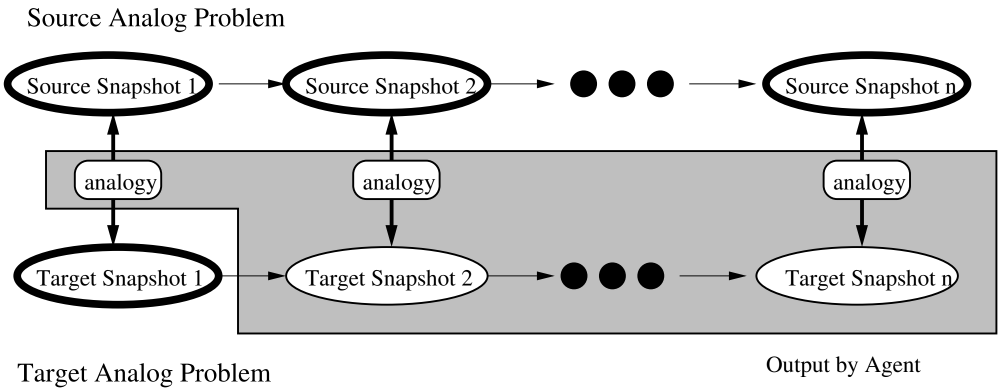

Fig. 10. This Figure shows the structure of the procedure transfer in the general case. The things outside the shaded box are given to the agent: a complete source problem an incomplete target problem. The system completes the analogical transfer and stores the new s-image sequence for the target problem.

Here is the main algorithm. It takes as input at least the following items: A memory of potential source series, success conditions, and a series consisting only of a single problem nv-state (the target problem). Words in bold represent functions that will be described in more detail later.

1. **Evaluate.** Run **evaluate**, where the knowledge-state argument of evaluate is the last nv-state currently in the target problem, and the input success conditions are the specification conditions. If the goal conditions are met, exit, and the problem is solved. If not, set the target nv-state to current-target-knowledge-state, then go on.
2. **Choose problem solving strategy.** Choose a problem solving strategy from those that have not failed for this target problem. If all have failed, exit and fail. If analogy is chosen then go on.
3. **Retrieve.** This is the first step of *non-visual analogical problem solving*. Run **retrieve**, where the single argument is the input target nv-state. If retrieve fails, mark analogy as having failed for this problem and return to 2 and choose another strategy.
4. **Find mapping.** Run **find-mapping** with the following arguments: the current-target-knowledge-state and current-source-knowledge-state. There may be an input suggestion from a visual mapping that was found. If find-mapping fails to find a

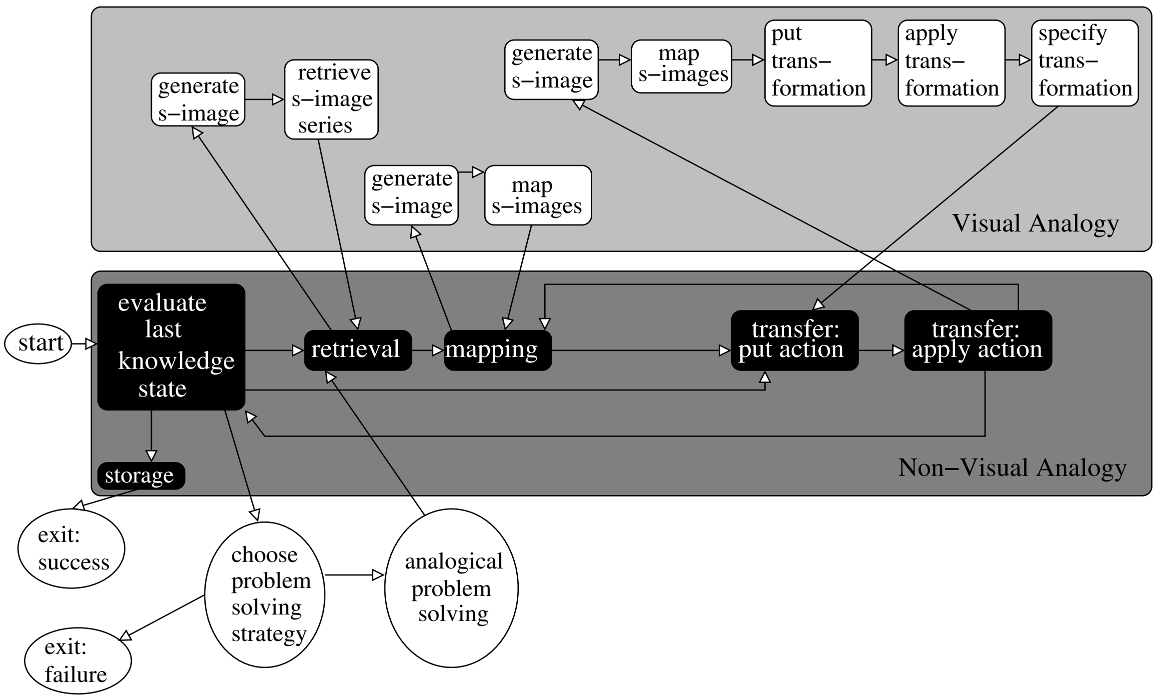

Fig. 11. Flow diagram demonstrating control in theory.

mapping, and visual knowledge state abstraction has not failed, go on. If it succeeds, Go to 8 to try to transfer the actions. If it fails because there are no more states to evaluate, or because visual knowledge state abstraction has failed, go to step 3 and try another retrieval.

5. **Generate new s-images.** This is the first step of *knowledge state visual instantiation*. There are multiple ways to visualize something. Each time this step is reached, search through the space of visual instantiations for the source and target, creating new visual instantiations by running: **generate-s-image** where its argument is the current-target-knowledge-state. It will generate the current-target-s-image. Then run **generate-s-image** where its argument is the current-source-knowledge-state. It will generate the current-source-s-image. If there are no more new visual instantiations to be made, mark knowledge state visual instantiation as having failed for this mapping and go to 4.
6. **Find visual mapping.** Map the s-images by running **find-mapping** with the following arguments: The current-target-s-image and the current-source-s-image. If you fail, go to 5 and try to get a different visual instantiation to try. If you succeed, go on.
7. **Transfer mapping.** Run **transfer-mapping** with the following arguments: the new visual mapping and its s-images, and their corresponding nv-states. It will generate a suggested mapping for the nv-states. Use this suggestion, going back to 4, the mapping stage. This is the last step in knowledge state visual instantiation.
8. **Transfer actions.** Run **transfer-manipulation**, attempting to transfer the current source nv-state's action to the target. Go on to try to apply it.
9. **Apply action.** Run the associated action on the current target knowledge state. If it works, generating a new nv-state, go to 1 and evaluate. If it does not work, go on to try manipulation visual instantiation at 10. If visual instantiation has been marked as failed for this mapping, go back and try another mapping at 4.
10. **Generate s-images.** Generate new s-images as in step 5, if 1) there are no s-images for the current nv-states, or 2) the current s-images have been marked as having failed.
11. **Generate transformation.** Run **generate-transformation** and find the visual analog of the action of the current source nv-state. If this function fails, mark visual manipulation abstraction as having failed for this source and target nv-state series and go back to apply action at 9.
12. **Apply-manipulation.** Apply this transformation to the corresponding element in the current source visual s-image. This makes a new s-image.
13. **Transfer manipulations.** Run **transfer-manipulation**, attempting to transfer the current source s-image's transformation to the target. If the transformation cannot be applied, go back to 5, marking this visual instantiation as failed. Else go on.
14. **Apply transformation.** Apply this transformation to the current target s-image. This makes a new s-image. Run **apply-manipulation** to do this.
15. **Generate action.** Run **generate-action** to specify the abstract transformation into an action that can be taken on the current target nv-state. If no unused specifications remain, mark visual manipulation abstraction as having failed for this source and target s-images. Go back to 10 and generate new s-images. If an action is generated, associate that action with the nv-state and go to 9 to apply it.

Now I will describe the functions referred to above in more detail.

- **Name:** Evaluate
- **Input:** Success Conditions, nv-state (nv-state)
- **Output:** [success — failure]
- **Process:**
  1. Evaluate will run a simple simulation of the system as it stands in the input nv-state. The details of how this will work will be written up in the dissertation, but are not a part of my theoretical claims.

- **Name:** Retrieve
- **Input:** knowledge state (knowledge state)
- **Output:** knowledge state-series (knowledge state-series)
- **Process:**
  1. If the input knowledge state is an s-image, then this function will return series represented in Covlan. If the input knowledge state is a nv-state, it will return nv-state series.
  2. The function will reject any series that has been marked as failed for the input knowledge state's series.
  3. If there is a conflict, the best matching series is returned. The details of how retrieval happens will be fleshed out over the course of the dissertation and is not important to my theoretical claims.
  4. If all potential analogs have been marked as failed, mark analogical problem solving has having failed for this source nv-state series and exit.
  5. If it succeeds, set the retrieval problem state to current-source-knowledge-state.

- **Name:** find-mapping
- **Input:** knowledge state1 (knowledge state), knowledge state2 (knowledge state)
- **Output:** a mapping
- **Process:**
  1. The details of the mapping process will be determined in the process of the dissertation and are not important to my theory.

- **Name:** generate-s-image
- **Input:** knowledge-state (nv-state)
- **Output:** an s-image, visual instantiation connections between the nv-state and the s-image
- **Process:**
  1. The agent has knowledge of what each object looks like. This means that each object is associated with an element or complex of elements and relations. Default values (such as where something will be placed in an s-image) will be determined by the relations of objects with other objects in the nv-state. There is psychological data (Richardson et al., 2001) showing how actions are associated with image placement I will use to constrain how this works. This module needs considerable fleshing out.

- **Name:** transfer-mapping

- **Input:** mapping (mapping), knowledge state1 (knowledge state), knowledge state2 (knowledge state)
- **Output:** mapping or failure
- **Process:**
  1. Generate a new symbol for the mapping.
  2. Associate with the new mapping new versions of all the maps.
  3. Change the referents of all the maps to what is connected to them with the visual instantiation connections.

- **Name:** transfer-manipulation
- **Input:** knowledge state1 (knowledge state), manipulation (manipulation), knowledge state2 (knowledge state)
- **Output:**
- **Process:**
  1. Take the manipulation connected to knowledge state1 and connect it to knowledge state 2. Specifically, connect the manipulation to the analogous entity or entities in knowledge state2.
  2. Transfer all manipulation arguments to the new manipulation. If an argument has an analog in snapshot2, use that. If it it does not, transfer it literally.

- **Name:** apply-manipulation
- **Input:** knowledge state1 (knowledge state), manipulation (manipulation)
- **Output:** another link in knowledge state1's series
- **Process:**
  1. Generate a new knowledge state, connected in series to knowledge state1. This new knowledge state is like knowledge state1 except it has the manipulation applied to it. If this cannot be done, exit and fail. Else exit with success.

- **Name:** generate-action
- **Input:** transformation1 (transformation), knowledge-state1 (knowledge-state)
- **Output:** an action associated with knowledge-state1
- **Process:**
  1. Retrieve an unused candidate action from the specifications of transformation1. Take into account the transformation, and what it will be applied to in knowledge-state1.

- **Name:** generate-transformation
- **Input:** knowledge-state1 (knowledge-state), action1 (action)
- **Output:** transformation, and possibly s-images.
- **Process:**
  1. If there is no s-image associated with knowledge-state1, make one.
  2. If there is no s-image associated with knowledge-state1's target knowledge-state, make one.
  3. Abstract action1 into a visual transformation appropriate to the visual abstraction in the s-image.

4 Theory Evaluation

My evaluation methods will reflect that this is a theory of human visual analogy as well as a contribution to artificial intelligence in general.

I will use several converging sources of evidence to test my hypotheses. They are 1) explaining the Craig test data, 2) making psychological predictions, 3) comparison to non-visual theories, 4) implementation of the theory into a working computer program, and 5) experimentation with that agent.

4.1 Explaining the Craig Test Data

Part of the Craig data have been set aside for evaluation. These are the test data. A disinterested third party will choose examples from the test data that can be used to appropriately test my hypotheses.

In particular I predict that Covlan will account for all the visual elements determined in the test data, where the elements found in the test data will be found using the same reliability methods used to get elements from the training data. This may seem at first a claim more about the consistency of the data than the coverage of my theory, but recall that analysis of the training data is not the only constraint going into the creation of Covlan. Realistically, I don't expect it will account for all of them, but the success of the theory on this measure is a function of its coverage.

If the primitives in the theory are used to represent multiple problems such that problem solving can occur, this will support the hypothesis that the primitives chosen are not case specific, and provide a general language for problem representation. I have four examples to model: The fortress/tumor problem, the Maxwell case, the furnace/factory problem, and the lab/weed-trimmer problem (the last two being from the Craig data). These four cases are very different from each other in terms of content, and constitute a representative sample of the physical systems domain that I am making claims about. Thus, my results should generalize to other physical problems.

4.2 Psychological Experimentation

My theory will make psychological predictions, and it will be possible to test the theory using psychological experimentation. I will probably not run these experiments, as my focus is on computer science aspects, but I will at least describe possible experiments that could be run to test my theory. These descriptions will help demonstrate my theory is falsifiable and making testable, non-vacuous claims.

I will describe one possible experiment as an example. One prediction of my theory is that visual representations will be used, primarily, when causal representations fail. Typically these analogy experiments involve the participants reading about problems, solved and unsolved. My theory predicts that if symbolic mismatches in the text description (regarding objects and their causal properties) participants, when solving the problem analogically, will make more diagrams and experience more mental imagery.

4.3 Comparison to Non-Visual Agents

I will compare my theory to non-visual schemes: Structure-Behavior-Function, PI, ACME, and SME. I will respond to all theoretically, and to Structure-Behavior-Function (SBF) experimentally and computationally by implementing an SBF model into my agent. The comparison of the non-visual to the visual will be a part of how the program will run– when the non-visual fails, the visual will attempt to help, showing that such representations are useful.

It's important for me to compare my theory to these other theories because it will show evidence for when visual representations give some benefit. If, for example, these non-visual schemes work just as well in all conditions, my argument for the usefulness of visual representations will not be supported.

Structure-Behavior-Function. A theme of model-based analogy (Bhatta & Goel, 1997c) is creating ontologies of useful abstractions by making claims about what kinds of inferences are needed and what kinds of knowledge are required to draw the needed inferences. This functional, top-down approach contrasts with more bottom-up architectural approaches to knowledge representation. For example, a bottom-up theory might specify that knowledge is represented as chunks or as productions. In my work, I postulate specific kinds of knowledge that need to be encoded to enable particular kinds of inferences. For example, the KRITIK system (Goel 1991a; 1991b; Goel et al., 1997) represented knowledge of the functioning of physical devices in terms of structure, behavior and function models (Chandrasekaran et al., 1993; Prabhakar & Goel, 1996). The primitives of the SBF language enabled the inferences needed to retrieve and adapt previous design cases to solve new design problems. Similarly, the IDEAL system, (Bhatta & Goel, 1997a; 1997b) used a language of generic physical principles and generic teleological mechanisms, which are useful units of analogical transfer in creative device design. Generic teleological mechanisms provide a taxonomy of functional and causal transformations to physical devices. In contrast, the ToRQUE system (Griffith et al., 1996; 2000) used a taxonomy of generic structural transformations that could be applied to physical systems. These transformations were found to be useful in modeling a protocol of a human subject solving a problem dealing with spring systems.

How can the wall be mapped to the the weed-trimmer? At first blush the two ideas have little in common. The trimmer is moving and encounters a barrier; the wall is not moving and is a barrier itself. One systematic way to characterize and represent these objects is with the Structure-Behavior-Function language. To represent devices in this language, one describes the functions first, then the relevant behaviors and structures needed to enable those functions.

I will also compare the SBF to Covlan as languages. KRITIK (Goel et al. 1997) has a vocabulary of changes which I can compare to transformations:

1. substance substitution (generalization and specification)
2. component modification (replacement, modality change, component-modality change, component-parameter adjustment)
3. relation modification (series-to-parallel and parallel-to-series)
4. substructure deletion (e.g. component deletion)

5. substructure insertion (e.g. substructure replication)

The SBF language does not, at this point, possess the power to express the geometrical relations necessary to represent the problems in the Craig data. For example, in SBF there is no sense of sides, or how they connect. The closest SBF concept to such a thing would be the start and end point of a wire in a circuit design domain, or the connecting points of a component in the KRITIK system.

In this dissertation I will expand SBF so that it can represent these problems, and compare the resultant representation to the visual one.

ACME: Analogical Constraint Mapping Engine. ACME is a mapping engine based on the theory that mapping is a result of structural, semantic, and pragmatic constraints (Holyoak & Thagard 1989b). Structure, in this sense, is does not necessarily mean a physical makeup, but the nature of the representation: elements are structurally similar if they share the same relational structure with other elements. Semantic similarity means elements are either identical symbols or share predicates (e.g. a common super ordinate). Pragmatic constraints involve relative importance of some propositions in the representation given the goals of the agent. The mapping is generated as a result of a constraint-satisfaction spreading activation network. Transfer in ACME involves transferring relations and postulating new elements from the source analog, but it does not have a mechanism for the transfer of a solution procedure. That is, it is made to transfer facts, not instructions.

SME: Structure Mapping Engine. SME is an agent based on the Structure-Mapping Theory (Gentner, 1983). It constrains the mapping problem with the empirically validated systematicity principle (Falkenhainer et. al., 1990) which means that high order relational similarities are preferred. SME finds many possible mappings, then evaluates them according to the map rules. Similarity is based on analogy (described above), literal similarity (where both relational and object predicates are mapped), mere-appearance (where primarily only the object descriptions are mapped), and abstraction mapping (where the entities in the base domain are variables rather than objects). These correspond to different match rules that can be used with SME.

Process of Induction. The process of Induction (PI) model (Holyoak & Thagard, 1989) is the only implemented computational model, other than my own, of the fortress/tumor problem that actually transfers the solution steps.

PI uses spreading activation to retrieve a source analog. The similarities between the analogs that result in the retrieval are used start the mapping. Like ACME, further mapping occurs through constraint-satisfaction.

4.4 Implementation

I will show evidence that my theory's generality by by implementing it in a running computer program and making it work on several examples. The theory, as described in the section above, is a more complete description of how agents both humans and artificial agents might solve problems. The implementation will focus on a subset of these claims. For example, the agent will choose a problem solving strategy, but analogy will be the only one available.

I have already implemented some of the theory in an agent called *Galatea.* The fact that I already have a program working with two examples (the fortress/tumor and the Maxwell case) provides initial evidence for all three of my high-level hypotheses (Davies & Goel, 2001; Davies, Nersessian & Goel, 2002). In incorporating more examples from the Craig data, the expanded program will further support them, if indeed it works.

It has no non-visual representations and cannot do analogical mapping. It has the machinery for analogical problem solving for visually represented source and target problems, and much of this will be used for implementing the non-visual as well. The expanded program I will make for my dissertation evaluation will be called *Proteus*. First I will describe Galatea, which works for the fortress/tumor problem and the Maxwell case, and then I will describe what more Proteus will do.

The knowledge representation, at an architecture level, consists of propositions: connections of two ideas or propositions with a relation. The substance of the theory is Covlan, in the higher level visual knowledge types.

Galatea currently takes as input 1. a solved source problem, 2. an unsolved target problem (both represented visually), 3. an analogical mappings between the s-images, and 4. criteria for an adequate problem solution. When instructed to solve the target using the source, it analogically transfers the solution procedure. As can be seen in Figure 10, it outputs a series of s-images for the target problem, and checks to see if the solution transferred indeed solves the problem constraints.

The agent Proteus will be an extension of Galatea. It will be expanded as needed to model the Craig data problems, and make the models of the fortress/tumor and Maxwell case problems theoretically consistent. It will also implement much of the non-visual problem solving and representation as outlined in the theory section.

4.5 Agent Experimentation

To understand why the agent works and under what conditions, I will run experiments with the agent under different internal conditions. For example, in an ablation experiment, a function of representation is removed from the program. The agent's behavior is then observed to help understand the effect of that missing piece. Exactly what experiments to run will be clearer as I develop the theory and agent.

Through experimentation I plan to evaluate these more specific hypotheses:

1. Visual instantiation will prove to be useful for two kinds of symbolic mismatches that can occur in non-visual representations.
2. Visual instantiations can provide an intermediate representation through which otherwise dissimilar entities and manipulations can be aligned.
3. Visual representations alone cannot enable evaluation of the solution.

5 Expected Results

As a result of modeling the Craig training data, I expect that Covlan will be able to represent all of the Craig test data, both in terms of elements and transformations. I expect that my theoretical comparison with the non-visual will show that, as a category, they would benefit from visual abstraction. I expect my implementation to work, and to show computationally the usefulness of visual abstraction. Experiments with the agent will support my hypotheses as well.

6 Discussion

In this section I will describe other analogy and visual analogy models and compare my work to them. Then I will discuss in more detail the nature of visual representations.

6.1 Other Analogy Models

The landscape of analogy research can be cut in different ways. I will discuss some major theories in the field with respect to 1) their approach to matching and similarity, 2) their structure versus content emphasis, and 3) their emphasis on particular parts of the analogical process. The models I will discuss are my own, ACME (Holyoak & Thagard, 1989b), MBA (Bhatta & Goel, 1997c), SME (Falkenhainer et. al., 1990; Gentner 1983), derivational analogy (Veloso & Carbonell, 1993; Carbonell, 1986), case-based reasoning (Kolodner, 1993; Hammond, 1990), Copycat (Hofstadter & Mitchell, 1995) and HACKER (Sussman, 1975).

Matching and similarity issues concern how analogs are mapped, retrieved and processed in transfer. Approaches can be classified into five themes. First, attributes of analogs can be encoded according to numeric values. Similarity can then be defined as some distance measure in a multi-dimensional space (described in Winston, 1992, p24). Second, attributes can be represented in terms of non-numeric features to be matched exactly. Proteus, SME, ACME, Copycat, derivational analogy, case-based reasoning, and HACKER fall into this category. Third, structure-based similarity has a focus on the relational structure of the analogs. SME and ACME both rely heavily on both features and relational structure. Some systems guide similarity evaluation with the agent's goals, the fourth theme. MBA relies heavily on the goals, as well as different methods for the same task. ACME has a heuristic for favoring elements and propositions relevant to the goal in retrieval and mapping. In contrast, SME's mapper has no notion of the goal. Proteus does not emphasize the agent's goals. Fifth, Some systems use stored abstractions to guide similarity. Certain MBA systems do, and Proteus uses visual abstractions to guide similarity. SME uses very small abstractions in the form of variablized functions.

The next cut is that of content-based theories and process-based theories. Proteus, Copycat, and MBA (and its SBF and TMK representation languages) rely on knowledge but also make theoretical claims about how knowledge is structured. Content-based theories create typologies of content. SME and ACME, for example, do not do this. For example, an SME model might represent the sun as having the property *hot*, but there is nothing in SME's theory that determines that it must be represented this way. The emphasis is on the processing *whatever* the content might be. In contrast, my theory has a limited language of primitives of which visual representations are composed. Enforcing this language for every example used and tying it to the process theory counters the argument that the systems works because the representation was example-specific.

Some theories of analogy, such as CBR, endeavor to explain similarities of within-domain examples, and others focus on cross-domain similarity. In CBR, the case library is assumed to be large enough such that a case similar to the one you are working with can be retrieved. My theory, MBA, SME, and ACME all focus on cross-domain analogy. As a result mapping and transfer are difficult problems, and these systems focus on them.

HACKER represents problems and solution states. CBR represents problems, solution states, and an evaluation of the outcome. SME, ACME, and MBA focus on the problem, the solution, and a model of the systems in question. As opposed to a more traditional approach to problem solving, these systems do explanation and model construction. Derivational analogy represents the problem, the solution state, and the trace of the problem-solving procedure. My theory is closest to this, in that the problem, solution procedure, intermediate states, and the solution state are represented. Like Derivational Analogy, my theory focuses on transfer.

Traces, called *derivations*, are scripts of the steps of problem solving, along with justifications for why the steps were chosen over others. One way my work differentiates itself is that in derivational analogy, the nv-states are not saved, as such, only the record of the changes made to them. This means that the nv-states can be inferred, but are not explicitly present in memory. Functionally, this means that retrieval based on those states is not possible, where in my theory, it is possible to retrieve a subsection of a problem solving solution sequence.

There are computational theories of analogical problem solving that use non-visual representations (dealing with, for example, actions, events and causality), but there are none that use visual representations.

6.2 Other Visual Analogy models

In the previous section I discussed other analogy models. This section narrows the focus to visual analogy models.

ANALOGY is an early visual analogy program (Evans 1968). It solved multiple choice analogy of the kind found on intelligence tests (e.g. A:B::C:?). It does this by describing how to turn A into B, then how C turns into all the choices. It matches the A to B transformation semantic net to the nets of the choices. The best match is determines ANALOGY's choice for the answer. Like my theory, ANALOGY had a visual language consisting of primitives (e.g. dot, circle, square, rectangle, triangle), relations (above, left-of, inside) and transformations (rotate, reflect, expand, contract, add, delete). My theory's ontology has considerable overlap with ANALOGY's.

ANALOGY has many differences with my theory. ANALOGY has no sense of absolute location in its visual representation. It describes only meaningless images, without any tie to what they represent (indeed, the domain is intentionally non-representational).

It can only describe transformations that occur in a single step. That is, it cannot represent a series of transformations that must be done in order. It has no sense of transfer–transformations are transferred to other analogs. Because of the domain, it also does not deal with retrieval issues.

GeoRep (Ferguson & Forbus, 2000) takes in line drawings and outputs the visual relations in it with the LLRD (low-level relational describer). Its visual primitives are: line segments, circular arcs, circles, ellipses, splines, and text strings. It finds relations of the following kinds: grouping, proximity detection, reference frame relations, parallel lines, connection relations, polygon and polyline detection, interval relations, and boundary descriptions. Then the HLRD (high-level relational describer) finds higher-level, more domain-specific primitives and relations. GeoRep's content theory is at the low level–the higher level primitives are left up to the modeler. My theory includes GeoRep's primitives, except for splines, which would be modeled in my theory with connected lines and curves.

Like my theory, LetterSpirit is a model of analogical transfer (McGraw & Hofstadter 1993). It takes a stylized seed letter as input and outputs an entire font that has the same style. It does this by determining what letter is presented, determining how the components are drawn, and then drawing the same components of other letters the same way. Like Galatea, the analogies between letters are already in the system: the vertical bar part of the letter *d* maps to the vertical bar in the letter *b,* for example. A mapping is created for the input character. For example, the seed letter may be interpreted as an *f* with the cross-bar suppressed. When the system makes a lower-case *t,* by analogy, it suppresses the crossbar. This is only theoretical– LetterSpirit never worked as a computer program.

LetterSpirit transfers single transformations/attributes (e.g. crossbar-suppressed) and therefore cannot make analogical transfer of procedures (e.g. moving something, then resizing it) like my theory can. In contrast, one can see how Galatea might be applied to the font domain: The stylistic guidelines in LetterSpirit, such as “crossbar suppressed” are like the visual transformations in my theory: it would be a transformation of removing an element from the image, where that element was the crossbar and the image was a prototype letter *f.* Then the transformation could be applied to the other letters one by one. In this way my theory has more generality than LetterSpirit.

Galatea does not generate the analogical mapping, but other systems, that create mappings with visual information, show that it can be done. The VAMP systems are analogical mappers (Thagard et al., 1992). VAMP.1 uses a hierarchically organized symbol/pixel representation. It superimposes two images, and reports which components have overlapping pixels. VAMP.2 represented images as agents with local knowledge. Mapping is done using ACME/ARCS (Holyoak & Thagard, 1997). The radiation problem mapping was one of the examples to which VAMP.2 was applied.

MAGI (Ferguson, 1994), uses mechanisms from SME. It takes visual representations and uses SME to find examples of symmetry and repetition in a single image. JUXTA (Ferguson & Forbus, 1998) uses MAGI in its processing of a diagram of two parts, and a representation of the caption. It outputs a description of what aligns with what, distracting differences, and important differences. It models how humans understand repetition diagrams.

Like my theory, MAGI, JUXTA, and the VAMPs use visual knowledge. But unlike my theory their focus is on the creation of the mapping rather than on transfer of a solution procedure. MAGI's and my theory are compatible: a MAGI-like system might be used to create the mappings that my theory uses to transfer knowledge. The theory behind the VAMPs is incompatible because they use a different level of representation for the images.

6.3 Visual Representation

Representations consist of content and format. The content is what is being represented, and the format is nature of the representation. Format can be thought of at two levels: physical and informational. For example, representing temperature with the number 8 can be thought of as ink on paper in the physical sense, or as a number in an informational sense. The informational sense is more appropriate for cognitive representations. Physically, mental representation consists of neural anatomy and the chemical and electrical states of neurons. Debates about mental representations are not at this level of abstraction. The representational debates occur over informational notions: propositions, images, bitmaps, etc.

Thus a representation can be considered propositional or image-like as a result of how it interacts informationally with the representing agent. What, then, is a *visual representation*?

I distinguish two types of visual representation. The first is a representation that is interpreted by a perceptual visual system, but need not have visual information as its content. For example, the level of mercury in a thermometer is a visual representation of temperature, even though temperature is not inherently visual. This is not the sense of visual representation I am using in this dissertation. Mental representations are neurons in the dark; no agent is using light to see them.

The sense I use in this dissertation is of visual representations as representing visual information, which is, roughly speaking, the information a visual perceptual system extracts from a scene. More formally, visual information consists of shapes, their sizes, locations, motions, and spatial relationships between shapes (e.g. connections, overlaps).

This sense of visual representation includes all represented visual information regardless of format (propositions, bitmaps, neural network weights, etc.) (Glasgow et al., 1995). Since format differences are *informationally* distinct rather than *physically* distinct, the format of a representation is determined by how the agent interprets that information. For example, there are no physical propositions in the head, but the way the mind accesses some information may reveal the representational format as propositional.[6]

One of Covlan's primitive elements is the point. Entire images could be created out of points, which would result in a bitmap image. This dissertation does not explore the use of bitmaps, though they can be used to resolve symbolic mismatches at the visual symbol level (such as the letter “o” and and ellipse).

[6] Barsalou argues against the existence of purely non-perceptual symbols, but even his theory is propositional: “Because perceptual symbol systems have the same potential to implement propositions, they too are propositional systems” (Barsalou 1999, p. 26).

7 Conclusion

I propose a cognitive theory as a model of the use of visual representations in analogical problem solving.

Participants have mentioned that the description of the roads in the fortress problem "radiating like the spokes of a wheel" was helpful in solving the tumor problem (Gick & Holyoak 1980). Our theory explains why this is the case: The fortress scene, from above, looks like rays hitting a tumor, facilitating several steps of analogy. Other accounts of the fortress/tumor problem cannot explain these data.

My theory shows that visual knowledge alone, with no amodal knowledge, is sufficient for enabling analogical transfer. This supports the central hypothesis of my work. My theory suggests a computational model of analogy based on dynamic visual knowledge that complements traditional models based on amodal knowledge.

Although Galatea does not yet address the issues of retrieval and mapping, put together with other work described in the previous section, we can now more confidently conjecture that visual knowledge alone can enable retrieval, mapping and transfer in analogy.

My theory represents visual knowledge symbolically, in the form of symbolic images made of visual elements and transformations. The symbolic representation provides the standard benefits of discreteness, abstraction, ordering, and composition. Although sequences of lower-level *bitmap* representations also capture the notion of ordering, they, by themselves, neither capture abstractions that enable noticing visual similarity nor enable transformations on the images. My theory provides additional evidence that symbolic representations of visual images are necessary for analogy. This is finding is important because visual reasoning is often thought to be a sub-symbolic process, but if my theory is correct, in analogical transfer even visual reasoning is symbolic.

I will expand on my previous work in this dissertation. I will further flesh out the role of visual information in analogical problem solving by creating a theory and implemented agent that demonstrates how problems in non-visual representations can be aided by visual instantiation and reasoning visually. The fact that the visual and non-visual systems will use the same analogical problem solving machinery will in some sense control for differences in processing.

I will show more evidence that visual information is useful for problem solving, that it is especially useful when there are symbolic mismatches for entities and manipulations at the non-visual level, and flesh out a visual language, as well as a theory of visual instantiation that will show how to get a visual representation from a non-visual one. I will show that my theory works for a number of examples, two of which have data from multiple experimental participants. I will evaluate my theory by explaining these data, making psychological predictions, comparing it to non-visual theories, implementing the theory in a program, and experimenting with that program.

The expected results of this proposed work follow. I will have developed a language of visual primitives for representing physical domains, and processes for using them to analogically solve problems. The community will have a better idea of the conditions under which visual representations are useful for solving problems, especially in contrast with SBF non-visual representations.

References

1. Anderson, J. R., & Libiere, C. (1998) *The Atomic Components of Thought.* Lawrence Erlbaum Associates.
2. Barsalou, L.W. (1999). Perceptual symbol systems. *Behavioral and Brain Sciences*, 22, 577-609.
3. Beiderman I., & Cooper E. (1991). Priming contour deleted images. *Cognitive Psychology*, 23, 393–419.
4. Beveridge, M. & Parkins, E. (1987). Visual representation in analogical problem solving. *Memory & Cognition.* 15(3), 230–237.
5. Bhatta, S. R. & A. K. Goel (1997a). A functional theory of design patterns. In the *Proceedings of IJCAI-97. pp294–300*
6. Bhatta, S. and Goel, A. K. (1997b) Learning Generic Mechanisms for Innovative Design Adaptation. *Journal of Learning Sciences,* 6(4):367-394.
7. Bhatta, S. R. & Goel, A. K. (1997). Design patterns: A computational theory of analogical design. In *the Proceedings of IJCAI-97 workshop on "Using Abstraction and Reformulation in Analogy."*
8. Boden, M. (1990) *The Creative Mind: Myths and Mechanisms.* Basic Books: London.
9. Carbonell, J. (1986). Derivational analogy: A theory of reconstructive problem solving and expertise acquisition. In Michalski, R., Carbonell, J., & Mitchell, T. (Eds.) *Machine Learning: An Artificial Intelligence Approach.* Morgan Kaufman Publishers: San Mateo, CA.
10. Casakin, H, & Goldschmidt, G. (1999). Expertise and the use of visual analogy: Implications for design education. *Design Studies*, 20:153–175.
11. Chandrasekaran, B., Goel, A. & Iwasaki, Y. (1993). Functional Representation as a Basis for Design Rationale. *IEEE Computer*, 26(1):48–56.
12. Craig, D. L., Nersessian, N. J., & Catrambone, R. (in press). Perceptual simulation in analogical problem solving. To appear in: *Model-Based Reasoning: Science, Technology, & Values (2002).* Kluwer Academic: Plenum Publishers, New York.
13. Croft, D., & Thagard, P. (forthcoming). Dynamic imagery: A computational model of motion and visual analogy. In L. Magnani (Ed.), Model-based reasoning: Scientific discovery, technological innovation, values. New York: Kluwer/Plenum.
14. Davies, J., & Goel, A. K. (2001). Visual analogy in problem solving. *Proceedings of the International Joint Conference on Artificial Intelligence 2001.* pp 377–382. Morgan Kaufmann publishers.
15. Davies, J., Nersessian, N. J., & Goel, A. K. (2002). Visual models in analogical problem solving. To appear in *Foundations of Science 2002, special issue on Model-Based Reasoning: Visual, Analogical, Simulative.* Magnani, L. & Nersessian, N. J., Eds.
16. Duncker, K. (1926). A qualitative (experimental and theoretical) study of productive thinking (solving of comprehensible problems). *Journal of Genetic Psychology*. 33:642–708.
17. Evans, T. G. (1968). A heuristic program to solve geometric analogy problems. In Semantic Information Processing edited by Minsky, M. MIT Press, Cambridge, MA.
18. Falkenhainer, B. (1988). Learning from physical analogies. Department of Computer Science, University of Illinois at Urbana-Champaign technical report UIUCDCS-R-88-1479.
19. Falkenhainer, B., K. D. Forbus, & D. Gentner (1990). The Structure mapping engine: algorithm and examples. *Artificial Intelligence* (41) 1–63.
20. Farah, M. J. (1988) The neuropsychology of mental imagery: Converging evidence from brain-damaged and normal subjects. In J. Stiles-Davis, M. Kritchevsky, and U. Bellugi (Eds.) *Spatial Cognition– Brain bases and development*. 33–59. Hillsdale, NJ. Erlbaum.
21. Ferguson, R. W. (1994). MAGI: Analogy-based encoding using regularity and symmetry. In Ram, A. & Eiselt, K. (Eds.), *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society* Atlanta, GA: Lawrence Erlbaum Associates, 283–288.

22. Ferguson, R. W. & Forbus, K. D. (1998) Telling juxtapositions: Using repetition and alignable difference in diagram understanding. In Holyoak, K., Gentner, D., & Kokinov, B. (Eds.) *Advances in Analogy Research*, 109–117. Sofia: New Bulgarian University.
23. Ferguson, R. W., & Forbus, K. D. (2000). GeoRep: A flexible tool for spatial representation of line drawings, *Proceedings of the 18th National Conference on Artificial Intelligence.* Austin, Texas: AAAI Press.
24. Gentner, D. (1983). Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, 7, pp 155-170.
25. Gick, M. L. & K. J. Holyoak. (1980). Analogical problem solving. *Cognitive Psychology*. 12, 306–355.
26. Gick, M. L. & K. J. Holyoak. (1996). LISA: A computational model of analogical inference and schema induction. In G. W. Cottrell (Ed.), *Proceedings of the Eighteenth Annual Conference of the Cognitive Science Society* (pp. 352–357). Atlanta, GA: Lawrence Erlbaum Associates.
27. Glasgow, J. & Papadias, D. (1998). Computational imagery. In Thagard, P. *Mind Readings*. Cambridge, MA: MIT Press.
28. Glasgow, J., Narayanan, N. H., Chandrasekaran, B. (1995). *Diagrammatic Reasoning: Cognitive and Computational Perspectives.* AAAI Press/MIT Press: Cambridge, MA.
29. Goel, A. K. (1991a) Model Revision: A Theory of Incremental Model Learning. *Proc. Eighth International Conference on Machine Learning (ICML-91)*, Chicago, June 1991, Los Altos, CA: Morgan Kaufmann, pp. 605–609.
30. Goel, A. K. (1991) A Model-Based Approach to Case Adaptation. *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*. Chicago, August 1991, Hillsdale, NJ: Lawrence Erlbaum, pp. 143–148.
31. Goel, A., Bhatta, S. & Stroulia, E. (1997). KRITIK: An early case-based design system. In: *Issues and Applications of Case-Based Design.* Maher, M. L. and Perl, P., eds., Erlbaum, Hillsdale, NJ., pp. 87–132.
32. Gooding, D. C. (1994) *Experiment and the Making of Meaning.* Dordrecht, Kluwer Academic Publishers.
33. Griffith, T. W., Nersessian, N. J. & Goel, A. K. (1996). The role of generic models in conceptual change. In *Proceedings of the Eighteenth Annual Conference of the Cognitive Science Society*, Lawrence Erlbaum, Mahwah, NJ.
34. Griffith, T. W., Nersessian, N. J., & Goel, A. K. (2000). Function-follows-form transformations in scientific problem solving. *Proceedings of the Twenty-Second Annual Conference of the Cognitive Science Society.* Lawrence Erlbaum, Mahwah, NJ.
35. Hammond, K. J. (1990). Case-Based Planning: A Framework for Planning from Experience. *Cognitive Science,* 14(4):385–443.
36. Hayes, J. R. (1989). *The Complete Problem solver.* 2nd ed. Hillsdale, NJ:Erlbaum.
37. Hofstadter, D. R. & Mitchell, M. (1995). The copycat project: A model of mental fluidity and analogy-making. In Hofstadter, D. and the Fluid Analogies Research group, *Fluid Concepts and Creative Analogies.* Basic Books. Chapter 5: 205–267.
38. Holyoak, K. J., & Thagard, P. (1989). A computational model of analogical problem solving. In S. Vosniadou & A. Ortony (Eds.), *Similarity and analogical reasoning*. Cambridge: Cambridge University Press. 242–266.
39. Holyoak, K. J., & Thagard, P. (1989). Analogical mapping by constraint satisfaction. *Cognitive Science* 13, 295–355.
40. Holyoak, K. J. & Thagard, P. (1997). *The analogical mind. American Psychologist*.52(1)35-44.
41. Kellman, P. J. & Arterberry, M.E. (1998). Chapter 5: Object Perception. In The cradle of knowledge: Development of perception in infancy, edited by P. J. Kellman & M. E. Arterberry. Cambridge: M.I.T. Press.

42. Kolodner, J. L. (1993). *Case-Based Reasoning*, Morgan Kaufmann Publishers, San Mateo, CA.
43. Kosslyn, S. M. (1994) *Image and Brain: The Resolution of the Imagery Debate.* MIT Press, Cambridge, MA.
44. Kriz, S. (2002). Understanding Simultaneity and Causality in Static Diagrams versus Animation. Poster Session: Cognitive Aspects of Diagrammatic Representation and Reasoning, Diagrams 2002.
45. Landaur, T. K. (1998). Learning and Representing verbal meaning: The latent semantic analysis theory. *Current Directions is Psychological Science*, 7 (5) pp 161-164.
46. Lenat, D. & Guha, R. (1990). *Building Large Knowledge Based Systems: Representation and Inference in the Cyc Project.* Addison-Wesley Publishing. Reading, MA.
47. Maxwell, J. C. (1861-2). On physical lines of force. In *The Scientific Papers of J. C. Maxwell*, Niven, D. ed. Cambridge: Cambridge University Press, 1890. Reprinted in 1952, New York: Dover Publications Vol. 1, pp. 451–513.
48. McGraw, G. & Hofstadter, D. R. (1993) Perception and Creation of Alphabetic Style. In *Artificial Intelligence and Creativity: Papers from the 1993 Spring Symposium*, AAAI Technical Report SS-93-01, AAAI Press.
49. Medin, D. & Ross, B. (1990) *Cognitive Psychology*. Harcourt Brace: New York.
50. Meuller, E. (1998). Panel discussion: ”Evaluating Representations of Common Sense”. *Fifteenth National Conference on Artificial Intelligence (AAAI 1998)*. July 30. Organizer: Douglas B. Lenat.
51. Miller, A. I. (1984. *Imagery in Scientific Thought: Creating Twentieth Century Physics.* Boston: Birkhauser.
52. Monaghan, J. M. & Clement, J. (1999). Use of computer simulation to develop mental simulations for understanding relative motion concepts. *International Journal of Science Education* 21(9), 921–944.
53. Nersessian, N. J. (1984) *Faraday to Einstein: Constructing Meaning in Scientific Theories.* Kluwer, Dordrecht, pp. 68-93.
54. Nersessian, N. J. (1992). How do scientists think? Capturing the dynamics of conceptual change in science. In Giere, R. N. (ed.)*Cognitive Models of Science.* University of Minnesota Press. Minneapolis, MN.
55. Nersessian, N. J. (1994a). Opening the black box: Cognitive science and the history of science. In: *Constructing Knowledge in the History of Science.* Osiris, Thackray, A., ed., 10:194–214.
56. Nersessian, N. J. (1994b). Abstraction via generic modeling in concept formation in science. Georgia Institute of Technology Cognitive Science Technical Report 94/22. To appear in: *Idealization and Abstraction in Science.* Jones, M. R. and Cartwright, N., eds., Amsterdam: Editions Rodopi, in press.
57. Nersessian, N. J. (2001). Maxwell and 'the method of physical analogy': Model-based reasoning, generic abstraction, and conceptual change. In: *Reading Philosophy of Nature: Essays in the history and philosophy of science and mathematics to honor Howard Stein on his 70th Birthday.* D. Malamet, ed., LaSalle, IL: Open Court, in press.
58. Pedone, R., Hummel, J. E., & Holyoak, K. J. (2001). The use of diagrams in analogical problem solving. *Memory & Cognition*, 29, 214–221.
59. Prabhakar, S. & Goel, A. (1996) Learning about novel operating environments: Designing by adaptive modelling. *Artificial Intelligence in Engineering Design, Analysis and Manufacturing, Special Issue on Machine Learning*, 10:136-142.
60. Pylyshyn, Z. W. (1978). Imagery and artificial intelligence. from Savage, C. W. (Ed.), *Perception and Cognition. Issues in the Foundations of Psychology, Minnesota Studies in the Philosophy of Science,* vol. 9, Minneapolis: University of Minnesota Press) 19–55.

61. Richardson, D. C., Spivey, M. J., Edelman, S., & Naples, A. J. (2001). “Language is spatial”: Experiemental evidence for image schemas of concrete and abstract verbs. in *Proceedings of the Twenty-third Annual Meeting of the Cognitive Science Society.* 873–878, Erlbaum: Mahwah, NJ.
62. Schank, R. C. (1972). Conceptual Dependency: A Theory of Natural Language Understanding, *Cognitive Psychology*, (3)4, 532-631.
63. Schrager, J. (1990). Commonsense perception and the psychology of theory formation. In Shrager, J. & Langley, P. (Eds.) *Computational Models of Scientific Discovery and Theory Formation.* Morgan Kaufman, San Mateo, CA. 437–470.
64. Shepard, R. & Cooper, L. (1982). *Mental Images and their Transformations* Cambridge, MA: MIT Press.
65. Sussman, G. J. (1975). *A Computational Model of Skill Acquisition*. American Elsevier, New York.
66. Thagard, P. & Hardy, S. (1992) Visual thinking and the development of Dalton's atomic theory. *Proceedings of the Ninth Canadian Conference on Artificial Intelligence.* Vancouver. 30–37.
67. Thagard, P., Gochfeld, D., & Hardy, S. (1992). Visual analogical mapping. In *proceedings of the 14th Annual Conference of the Cognitive Science Society*. Hillsdale, Erlbaum. 522–527.
68. Veloso, M. M. & Carbonell, J. G.(1993). Derivational Analogy in PRODIGY: Automating Case Acquisition, Storage, and Utilization. *Machine Learning*, 10(3):249-278.
69. Winston, P. H. (1992). *Artificial Intelligence*. Addison-Wesley Publishing, Reading, Massachusetts.
70. Yarlett, D. & Ramscar, M. (2000). Structure-mapping theory and lexico-semantic information. In Gleitman, L. R., and Joshi, A. K. (Eds.) *Proceedings of the Twenty-Second Annual Conference of the Cognitive Science Society.* Lawrence Erlbaum. Mahwah, NJ. pp.571–576.

8 Appendices

A The Craig Data

This research I selected a data set that fulfilled the following criteria: 1. It is data collected from human experimental participants, 2. the problems use examples of visual analogical problem solving, and 3. they provide constraints for the visual abstraction language.

In David Craig's experiments (Craig, Nersessian, & Catrambone in press) participants solved problems similar to the fortress/tumor problem. They saw diagrams showing the solution to the source problem and were asked to draw diagrams describing their solutions to the target problem. This data set fulfills the above criteria in that it is human participant data, it's an example of analogy with both visual input and output, and the diagrams drawn provide information about what to put in the visual abstraction language. I assume that the diagrams and descriptions the participants wrote down reflect, to some degree, the thinking process that went on when they solved the problem.

The first example is the lab/weed-trimmer problem.

This is the solved source problem given to the participants: “Please read the two problems below. At the bottom of the page, please try to solve Problem 2. Draw a diagram to show what you're thinking. The solution to Problem 1 may be helpful in solving Problem 2. Problem 1: A computer chip manufacturer has designed a special

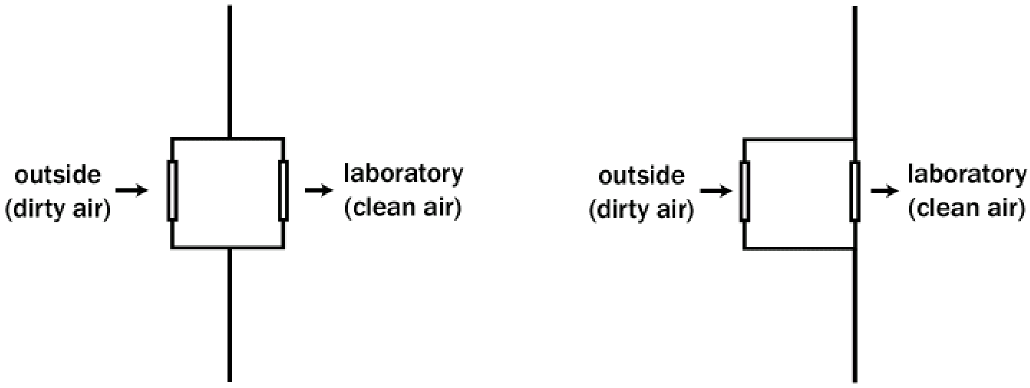

Fig. 12. This Figure shows the lab door problem.

lab for manufacturing microscopic devices. They have taken great care to seal off the lab from the surrounding environment in order to keep the air inside the lab free of dust and undesirable gases. The problem, though, is that whenever lab workers enter or leave the room, the seal is broken and contaminated air is allowed in. The company is trying to design a door that will allow workers to enter and leave the lab easily, while minimizing the amount of contaminated air that is let in. Solution: Have workers enter a vestibule space before entering the lab." A diagram of the intermediate room is shown. It is a plan (a roofless top view), where the walls are represented with lines and the doors as unfilled rectangles.

The unsolved target problem is: "Problem 2: In order to trim the weeds that grow along the side of the road, the department of transportation has designed a weed trimmer that attaches to the end of a long pole sticking off the side of a truck. As the truck drives down the highway, the trimmer is extended about 6 feet to the right, perfectly positioned to trim the weeds at the side of the road. The problem is that the 6-foot pole is obstructed by sign posts that are positioned at the curb in certain parts of the city. The weed-trimmer pole, in fact, is exactly 2 feet too long to clear the sign posts. Although the wee-trimmer pole could be retracted or lifted out the way to clear the sign posts, this would interfere with the weed trimming. And although the pole could bend over the top of the sign posts, this would be impractical since in some areas the signs are 15 feet tall. The Department of Transportation is trying to design a pole that can pass *through* the sign posts without stopping or changing the position of the trimmer. *In the space below, try to design a weed-trimmer pole that can pass through sign posts. Draw a diagram to illustrate what you're thinking.*"

The solution to the first problem is to make a set of redundant doors so that air can't go through both at the same time because one is always closed. Some of the participants came up with the analogous solution for the weed-trimmer problem, which is to have a redundant support mechanism for the trimmer. That is, have double door-like support mechanisms, one of which is always closed, holding the trimmer together.

The second analogy problem given is the furnace/factory problem.

“Problem 1: A blast furnace at a steel mill requires a steady stream of fresh air to keep the furnace burning at a suitably high temperature. The problem, though, is that as fresh, oxygen-rich air enters the furnace, stale air must exit, taking a considerable amount of heat with it. The fresh air comes from the surrounding environment and this is several hundred degrees cooler than the furnace. Although the engineers at the steel mill have considered heating the incoming air before it reaches the furnace, this would be very expensive and wouldn't solve the problem that valuable heat is lost when stale air leaves the furnace. The engineers are trying to design a system that will solve the heat-loss problem. Solution: Run the input and output streams close to each other so that the hot output stream warms up the cool input stream.” A diagram is shown of the air input and output shafts up against one another, connected to a square furnace at the bottom. Arrows indicate the direction of air flow.

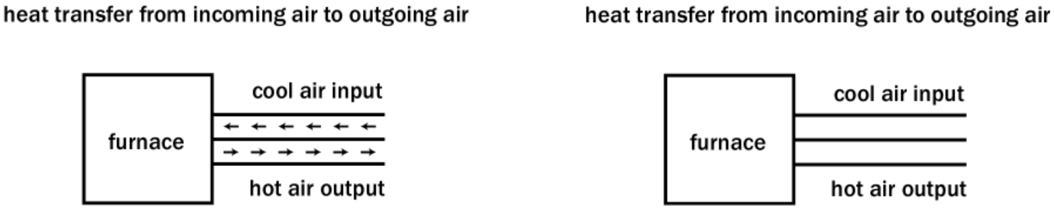

Fig. 13. This Figure shows the furnace problem.

The target problem follows: “Problem 2: Citizens of a small island nation in the South Pacific are forced to pay taxes directly to the King. Most citizens work in a shoe factory at the center of the island. To insure that taxes are properly paid, the King has set up a provisional tax office in the basement of the factory. On payday, workers leaving the factory at quitting time are ushered down the stairs into the office. When it's their turn, they empty their pockets of the coins they were just paid (all workers are paid the same– 10 small silver coins a month) and the King takes eighty percent. On a typical payday, the scene is pretty grim: Happy workers walking down the stairs to the tax office pass those walking up 80% poorer. Most of the citizens have grown tired of being taxed so heavily. Unfortunately, though, there doesn't seem to be a way out of paying the tax. There is only one exit to the factory, and workers are not allowed to leave before going down to the tax office. Ideally, the citizens need a way to avoid being so heavily taxed but they all still need to go through the tax office on payday so the King doesn't get suspicious. *In the space below, try to come up with a scheme that will help the factory workers avoid being taxed so heavily. Draw a diagram to illustrate what you're thinking.*”

The analogous solution is to have the workers pass money to each other as they pass on the stairwell, as the air in the furnace ducts pass heat.