Transfer of Problem-Solving Strategy Using Covlan

Jim Davies[a] Ashok K. Goel[b]

[a]Institute of Cognitive Science, Room 2201, Dunton Tower, Carleton University, 1125 Colonel By Drive, Ottawa, Ontario, Canada, K1S 5B6

[b]College of Computing, Georgia Institute of Technology, 801 Atlantic Drive, Atlanta, Georgia, USA, 30332

Psychological evidence suggests that humans use visual knowledge and reasoning in solving complex problems. We present Covlan, a visual knowledge representation language for representing visual knowledge and supporting visual reasoning. We describe Galatea, a computer program that uses Covlan for analogical transfer of problem-solving procedures from known analogs to new problems. We present the use of Galatea to model analogical visual problem solving by four human experimental participants, and describe one of the four cases in detail. The Galatea model of human problem solving suggests that problem-solving procedures can be effectively represented with Covlan.

1. Introduction

Some domains are intrinsically visual, that is, the objects and relations in these domains are fundamentally visual in nature, e.g., the domains of animal shapes and human faces. Other domains appear to be multi-modal in that much of the knowledge of the objects, relations and processes in the domain can be represented both visually and non-visually. For example, knowledge of an effective connection between a battery an a wire might be represented, among other ways, functionally (a specification that the battery needs to be physically touching the metal of the wire to conduct electricity) or visually (the image of the wire is spatially adjacent to the image of the battery.) Even though other kinds of knowledge and representations might be used to reason about these domains, human experimental participants report experiencing visual imagery when solving problems about them (9; 2; 15), indicating that visual knowledge and representation often plays an important role in human problem solving. There is also documentary evidence for visual reasoning in scientific problem solving, e.g. (16). Further, psychological evidence suggests that analogical problem solving is facilitated by animations (17), diagrams (1) as well as visually evocative phrases in stimuli (12). These results suggest that not only that visual knowledge and reasoning have an important function in human cognition, but also that complex problem solving might be usefully represented in a visual language.

Covlan is a visual knowledge representation language for symbolic representation of visual knowledge in complex problem solving, including problem-solving procedures. By procedures we mean solutions to problems that involve multiple sequential actions. A cooking recipe is a good example of a procedure because it consists of many steps that must be taken in a particular order. Our hypothesis is that Covlan enables representation of problem-solving procedures that is useful for analogical transfer of a procedure from a known analog to a new target problem.

Galatea is a computer program that uses known problem-solving procedures represented in Covlan to infer problem-solving solutions to new target problems (5). To support Galatea as a model of human visual problem-solving, we modeled four experimental participants who solved a visual analogy problem, one of whom is described in detail below.

2. Covlan

Covlan has been designed to describe human mental representations of visual and spatial properties of objects, relations and procedures with high-level, abstract symbols. It is a visual language because it represents visual and spatial information (only).

In Covlan all knowledge is represented as propositions relating two elements with a relation. There are several kinds of elements: primitive objects, primitive relations, general visual concepts, knowledge states, primitive transformations between knowledge states, and correspondence and transform relations among knowledge states. We call the knowledge states symbolic images, or `s-images`, which contain `visual elements, general visual concepts`, and `relations` between them. Procedures are represented by a series of `s-images`, connected with `transformations`.

`Transformations`, like computer program functions, take arguments to specify their behavior. These `transformations` control normal graphics transformations such as translation (`move-to-location`), and rotation (`rotate`). In addition there are `transformations` for adding and removing elements from the `s-image` (`add-element, remove-element`).

`Primitive Elements` are the visual objects. The element types are `rectangle, circle, arrow, line`, and `curve`. Each `element` is represented as a frame (in the artificial intelligence sense) with attribute slots, such as `location, size, orientation`, or `thickness`. A particular example of an `element` is referred to as an `element instance`.

`Primitive Visual Relations` are a class of symbols that describe how certain `visual elements` relate to each other and to the values taken by `general visual concepts`. The visual `relations` are `touching, above-below`, and `right-of-left-of`.

`General Visual Concepts` act as slot values for the primitive `elements` as well as arguments for the visual `transformations`. The concepts are `location, size, thickness, speed, direction, length, distance, angle`, and `direction`. Each concept has several values it can take. For example, the `size` can be `small, medium`, or `large`, and `thickness` can be `thin, thick` or `very-thick.` `Location` specifies an absolute qualitative location in ans`-image` (`bottom, top, center`, etc.)

`Correspondence and Transform Representations`. The knowledge of which objects in one `s-image` correspond to which objects in another is a `mapping`, which consists of a set of alignments between objects. Different sets of alignments compose different `mappings`. The *i*th `s-image` in the source and the *i*th `s-image` in the target have a `correspondence` between them; each `correspondence` can have any number of `mappings` associated with it (determining which mapping is the best is the "mapping problem.") The `correspondence`

Table 1
Covlan's primitive elements.

Primitive Element name	**attributes**
connection	subject, object, angle, distance
rectangle	location, size, height, width, orientation
circle	location, size, height
line	location, length, end-point1, end-point2, thickness
set	location, orientation, front, middle
curve	location, start-point, mid-point, end-point, thickness

and `mapping` between the initial `s-images` (*i*=1) in the source and target is given as part of the input to Galatea; the system generates the subsequent correspondences and mappings for the subsequent `s-images`.

Similarly, successive `s-images` in a series have `transform-connections` between them. These are needed so that Galatea can track how `visual elements` in a previous knowledge state change in the next.

2.1. Primitive Visual Elements

Covlan's ontology of `primitive visual elements` (Table 1) contains: `connection, rectangle, circle, line, set` and `curve`.

Symbols are connected to an `element` type with a relation called `looks-like-relation`. These symbols are `instances` of that `element`. The elements are frame-like structures with slots that can hold values. For example, a `rectangle` has a `location, size, height, width,` and `orientation`. All elements can have a `location`, which holds a value representing an absolute location on an `s-image` (e.g. `top`, `right`).

The `set` is a special `element`. A `set` can contain any number of `instances` of `elements`. These `instances` are connected with relationships to the set with the `in-set` relation. Sets also have an `orientation,` the value of which is one of the primitive `directions.` An `element instance` in the set is specified in the representation as the `front`, and another as the `middle`. The orientation is defined as an imaginary line from the `middle` to the `front` in the `direction` specified in the `orientation`.

Sometimes a *part* of an `element instance` must be referenced. For example, if a line touches the middle of another line, there must be some way to describe the *end* of the first line and the *middle* of the next. In Covlan different primitive elements have different kinds of `areas`.

`Lines` have `start` and `end points`, as well as `right` and `left-side mid-points`. The element instance's names are related to the symbols naming these areas (e.g. `line1-end-point` with `area-relations`: `has-end-point`, `has-start-point`, `has-rightsidemiddle`, and `has-leftsidemiddle`.

`Circles`, `squares`, and `rectangles` have `sides`, which are related to element instances with the following relations: `has-side1` (the top), `has-side2` (the right side), `has-side3` (the bottom), and `has-side4` (the left side).

Many spatial relationships between primitive elements are represented with `connections`. A `connection` is a primitive element with a `name`. `Connections` are frames with two four slots: `subject`, `object`, `angle` and `distance`, represented with `is-subject-for-connection`, `is-object-for-connection`, `has-angle` and `has-distance`. These relations connect the `connection` name to `distances` and `angles`, which are qualitative `miscellaneous slot values`. The object of the `connection` is `distance` away from the `subject` in the direction of `angle.`

The `distances` are `touching-distance`, `short-distance` and `long-distance`. The `angles` are `perpendicular-angle` (straight ahead), `right-angle-cw` (a right angle in the clockwise direction, or to the right), `45-angle-cw` (a forty-five degree angle to the right), `45-angle-ccw` (a forty-five degree angle in the counter-clockwise direction, or to the left), and `right-angle-ccw` (a right angle to the left). Figure 1 shows the different kinds of `connections` Covlan can represent. `Areas` of element instances, as well as element instances themselves, can be connected.

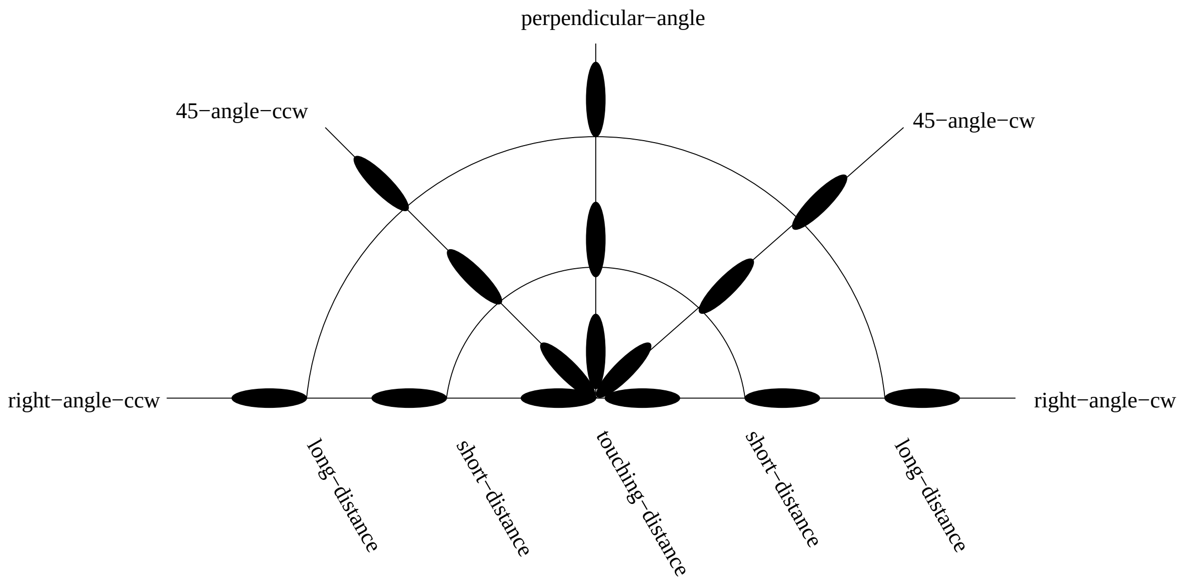


Figure 1. Each of the fifteen black dots in the Figure represents a qualitative `connection` area, with an `angle` and `direction`.

2.2. Miscellaneous Slot Values

`Miscellaneous slot values` are symbols that can give a value to `element` attributes or transformation arguments. See Table 2. They can be broken down into the following types: `angles`, `locations`[1], `sizes`, `thicknesses`, `numbers`, `directions`, and `lengths.`

[1]Relative locations, as opposed to absolute locations, are classified under `primitive visual relations.`

Table 2
Classifications of Miscellaneous Slot Values.

angles	perpendicular-angle, right-angle-cw, 45-angle-cw, 45-angle-ccw, right-angle-ccw
locations	bottom, top, right, center, off-bottom off-top, off-right, off-left
sizes	small, medium, large
thicknesses	thin, thick, very-thick
directions	left, right, up, down
lengths	short, medium, long

Table 3
Visual Relations.

Visual Relations	touching, above-below, right-of-left-of, in-front-of-behind

2.3. Primitive Visual Relations

The class of `primitive visual relations` (shown in Table 3) describe how certain visual `elements` relate to each other and `miscellaneous slot values.`

2.4. Analogy Representations

Covlan has representations for reasoning about analogies. `S-images` 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 between two `element` instances or `areas` in a given mapping is called a `map`.

Similarly `s-images` next to each other in sequences have `transform-connections.` These are necessary so the agent can track how `visual elements` in a previous `s-image` 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`. `Mappings` between sequential `s-images` are called `horizontal mappings`. Analogical `mappings`, between source and target `s-images` are `vertial mappings.`

`Transformations` are attached, in fact, to a `map` between two `elements` in sequential `s-images`. So if a `rectangle` changes into a `circle`, the agent knows which `rectangle` in the previous `s-image` turns into which `circle` in the next `s-image`.

2.5. Transformations

Table 4 shows Covlan's ontology of transformations. All are implemented to work with Galatea.

Table 4
Covlan's transformations.

Transformations	
Transformation name	**arguments**
add-element	object-type, location (optional)
add-connections	connection/connection-set
decompose	object, number-of-resultants, type
move-to-location	object, new-location
move-to-set	object, object2
put-between	object, object2, object3
replicate	object, number-of-resultants

`Add-element` adds a new primitive element in the next `s-image.` The ontology of primitive visual elements are described in the next subsection. The first argument, `object-type`, must be one of the members of the `primitive elements` (e.g. `square` or `circle`). It determines what kind of shape appears in the next `s-image`. The second argument is `location`, which must be one of Covlan's `locations`: `bottom`, `top`, `right`, `left`, and `center`. What this means is that the next `s-image` will have three relationships added: 1) The `s-image` connected with a `has-component` relation to the name identifying the new component, 2) the new component's name with a `looks-like` relation to the `object-type`, and 3) the component's name with a `has-location` relation to the `location`. See Figure 2. `Add-element` is used in the Maxwell example, and will be described in more detail in a later section.

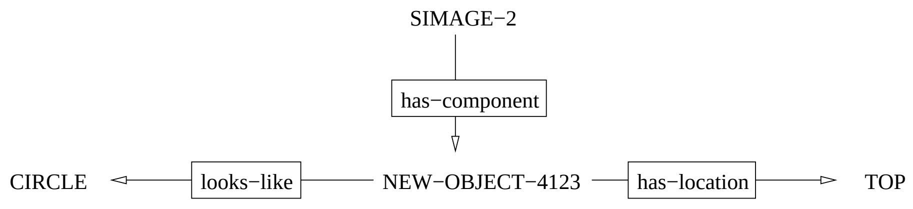


Figure 2. A graphical representation of the three relationships added by the `add-element` transformation. `Relations` are boxed. Objects at the beginning of arrows are in the `ThingX` slot; the objects at the end of the arrows are in the `ThingY` slot.

`Add-connections` is a transformation that inserts a set of connections into the next `s-image`. Input is the name of the set of connections in the source. To determine the nature of the connections in the target, Galatea uses substitution for all the symbols to find the analogous names, so that analogous connections are placed in the next target `s-image`.

`Decompose` takes a primitive element and replaces it in the next `s-image` with some `n` number of elements. It also reduces thickness for each of those elements.

`Move-to-location` changes the location of an element instance from one location to another. This means that in the next `s-image`, the old `has-location` relation is removed and a new `has-location` relation is added, relating the element to the input `location`, which can be an absolute location or another element.

`Move-to-set` takes in two sets as input (we will call them *set-a* and *set-b*). The members of *set-a* are moved to the locations of the members of *set-b*. If *set-a* and *set-b* have the same number of element instances, then each element of *set-a* is placed at the location of a distinct element in *set-b*. The element instance matching is arbitrary.

If *set-a* has more elements, then multiple members of *set-a* are placed at the locations of each member of *set-b*. The number of element instances in these groups is determined by the number of elements in *set-b* divided by the number of elements in *set-a*.

If *set-b* has more elements, then elements of *set-a* are distributed evenly across the locations of the members of *set-b*.

`Put-between` takes two objects that are assumed to be touching, and places some third object in between them. In the new `s-image` 1) the two objects are no longer touching and 2) the third is touching both of them.

`Replicate` takes in an element or set of elements and generates `n` new instances of that element or elements in the next `s-image`. Its behavior is similar to `decompose`, except that it does not change the size or thickness of elements, and can work on sets as well as single element instances.

3. Galatea

Analogy involves several steps: A reasoner starts with a target, and retrieves a similar source (or base) analog. Then the elements of the source analog are mapped to the elements of the target problem. This means finding alignments between the sub-parts of the two analogs (the source and the target). Next the source's procedure is transferred to the target, perhaps with some adaptation. Then the procedure is evaluated and finally stored in memory. Galatea models only the transfer stage of analogy.

Galatea takes as input 1) a source series of `s-images` connected with `transformations`, 2) a single `s-image` representing the target problem, and 3) a mapping between the target and the first `s-image` of the source. As output, Galatea produces a series of `s-images`, connected with `transformations`, starting from the input target. These are the steps to the solution of the problem. Each output target `s-image` is connected to its corresponding source `s-image` with an analogical `mapping`. The input `s-images` are created by hand, using Covlan to represent the systems described in an psychology experiment.

Dr. David Craig ran 34 participants in an analogical transfer experiment (3). Participants were shown a problem-solving solution with a laboratory, presented with text and a diagram. They were asked to solve an analogous problem with a weed-trimmer, presented with text only. Of these, 17 participants (in three conditions) correctly described the analogous solution. All participants were asked to draw a diagram to illustrate their suggested solutions. A laboratory clean room strategy is transferred by adding redundant doors to a weed-trimmer arm so that it can pass through street signs. The analogous solution is to design an arm with two latching doors, so that while one is open to let the sign pass, the other stays closed to support the arm and trimmer. Participants produced diagrams describing their solutions to the problems.

3.1. Algorithm

Following is the control structure for Galatea's transfer of procedures from a source case to the target. Figure 5 shows the `s-image` structure for L14, the participant on which we will use as a running example in this paper. We modeled L14 and three other of the experimental participants in Galatea. We will describe in detail our model of one of these participants, L14, in this paper, and briefly describe the results of modeling the other three. We use our ability to model these participant data as an evaluation of Covlan and Galatea as a cognitive theory.

The procedure (for the source, and then for the target) is that the doorway mechanism gets `replicated`, and then `moved` to the correct positions. Two walls are created to complete the vestibule, and finally they are placed in the correct position so that the vestibule is complete.

1. Identify the first `s-images` of the target and source cases. These are the current source and target `s-images`.

2. Identify the `transformations` and associated arguments in the current `s-image` of the source case. This step finds out how the source case gets from the current `s-image` to the next `s-image`. The model of L14 involves five `transformations` (see Figure 5). The first `transformation` is `replicate`. The second`transformation` is `add-connections` which places the door sets in the correct position in relation to the top and bottom walls. The third and fourth `transformations` are `add-component`, which adds the top and bottom containment walls. The fifth `transformation`, another `add-connections`, places these containment walls in the correct positions in relation to the door sets and the top and bottom walls.

3. Identify the objects of the `transformations`. The object of the `transformation` is what object the `transformation` acts upon. For L14's first `transformation`, this object is the parts of the door in the first `s-image` (we'll call it `door-set-l14s1`).

4. Identify the corresponding objects in the target. In the target, the trimmer arm's door mechanism is the corresponding object.

5. Apply the `transformation` with the arguments to the target component. A new `s-image` is generated for the target to record the effects of the `transformation`. `Replicate` takes two arguments: some object and some `number-of-resultants`. In this case the object is `door-set-b1s1` (`b1s1` means "base one, s-image two") and the `number-of-arguments` is `two`. The `replicate` is applied to the first L14 `s-image`, with the appropriate adaptation to the arguments: The `mapping` between the first source and target `s-images` indicates that the `door-set-b1s1` maps to the `door-set-l14s1`, so the former is used for the target's object argument. The number `two` is a literal, so it is transferred directly. Galatea generates `door-set1-l14s2` and `door-set2-l14s2` in the next `s-image`.

The second `transformation` is `add-connections`. The effect of this `transformation` is to place the `replicated` door-sets in the correct spatial relationships with the other `element instances`. It takes `connection-sets-set-b1s3` as the `connection/connection-set` argument. This is a set containing four `connections`. Galatea uses a function to recursively retrieve all connection and set proposition members of this set. These propositions are put through a function which creates new propositions for the target. Each proposition's relation and literals are kept the same. The element instance names are changed to newly-generated analogous names. For example, `door1-endpoint-b1s3` turns into `door1-endpoint-l14s3`.

Then, similarly to the replicate function, horizontal target `maps` are generated, and the other propositions from the previous `s-image` are instantiated in the new `s-image`.

The inputs to this `transformation` are `nothing` (a literal denoting that there is not any thing in the previous `s-image` that is being modified), the `connection set` `connection-sets-set-b1s3`, the source `s-image lab-base1-simage2`, the current and next target `s-images l14-simage2` and `l14-simage3`, the `mapping` `l14-simage2--l14-simage3-mapping1`, and the rest of the memory.

6. Map the original objects to the new objects in the target case. A `transform-connection` and `mapping` are created between the target `s-image` and the new `s-image` (not shown). `Maps` are created between the corresponding objects. In this example it would mean a `map` between door-sets, as well as their component objects. Galatea does not solve the mapping problem, but a `mapping` from the correspondences of the first `s-image` enables Galatea to automatically generate the `mappings` for the subsequent `s-images`.

7. Map the new objects of the target case to the corresponding objects in the source case. Here the parts of the door set in the target `s-image` are mapped to the parts in the second source `s-image`. This step is necessary for the later iterations (i.e. going on to another transformation and `s-image`). Otherwise the reasoner would have no way of knowing which parts of the target `s-image` the later `transformations` would operate on.

8. Determine if goal conditions are satisfied. If they are, exit, and the procedure is transferred. If not, and there are further `s-images` in the source case, set the current `s-image` equal to the next `s-image` and go to step 1.

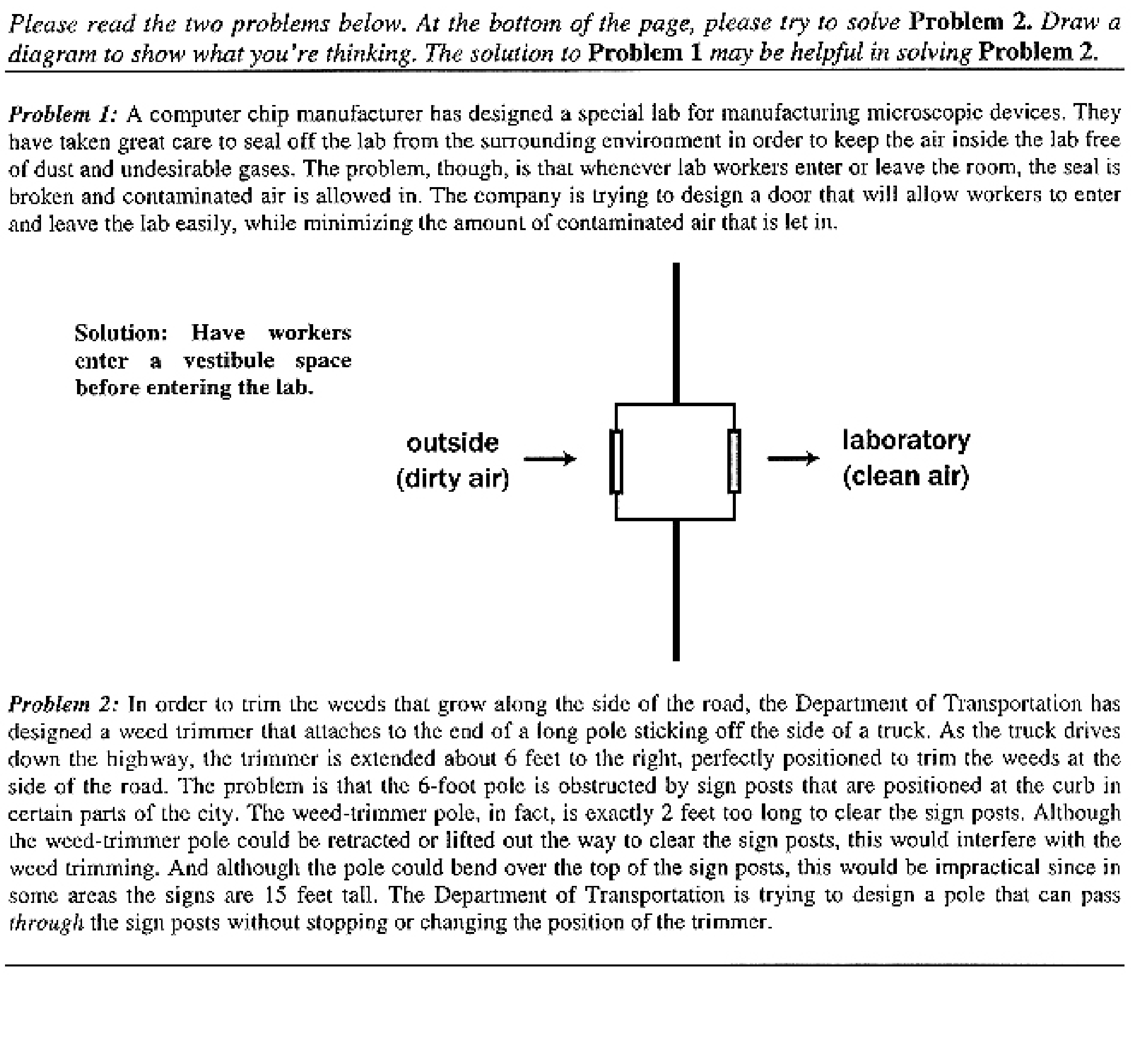
4. The Galatea Model of L14

The participants created solutions based on an analogy with a given stimulus. The drawings generated differed from the stimulus drawing in various ways. We chose to describe L14 in detail because it appeared to be the most difficult in that the drawing L14 created exhibited, among the drawings, the greatest number of differences. The stimulus L14 saw is reproduced in Figure 3.

Participants were divided into four experimental groups, each receiving the same text but a slightly different stimulus drawing. Figure 3 shows the condition that participant

L14 received. Figure 4 shows what L14 wrote on his or her data sheet during the experiment.

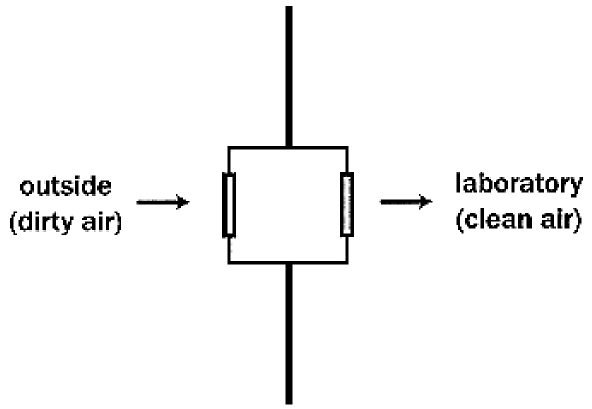
We represented the source analog as a series of `s-images` connected with `transformations`. See the top of Figure 5 for an abstract diagram of the source analog, and see Figure 6 for a diagram of some of the propositions in its first `s-image`.



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 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.



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 weed-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.

Figure 3. Condition 1: Plan view of lab, with the vestibule centered.

The model of L14 involves five `transformations` (See Figure 5). The first `transformation` is `replicate`. It takes in the `door-set-l14s1` as an argument, generating `door-set1-l14s2` and `door-set2-l14s2` in the next `s-image`.

The second `transformation is add-connections` which places the door sets in the correct position in relation to the top and bottom walls.

The third and fourth `transformations` are `add-component`, which add the top and bottom containment walls.

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 problem is 2 ft too long, so from 3½ ft to 4½ ft. Have a sliding bar that unlatches (twist) and retracts (Pull), for upcoming signs. And slides back up and latches when sign passes in. Then other arm does same and the sign passes through

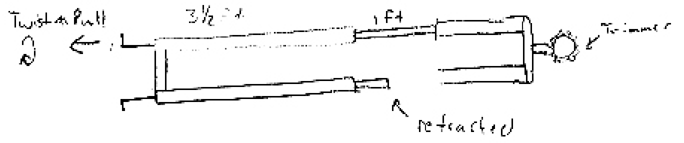


Figure 4. The source data for L14. The drawing above and handwritten text are what participant L14 produced on the experiment sheet.

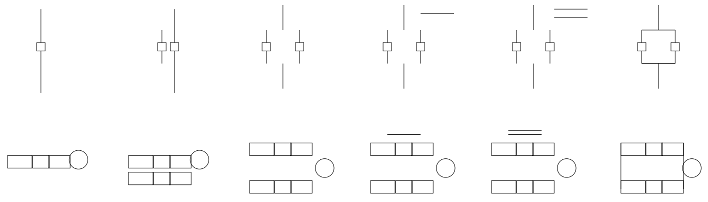


Figure 5. The implementation of L14. The top series of `s-images` represents the source analog (the lab problem) and the bottom series the target. There are six `s-images` for the five `transformations`.

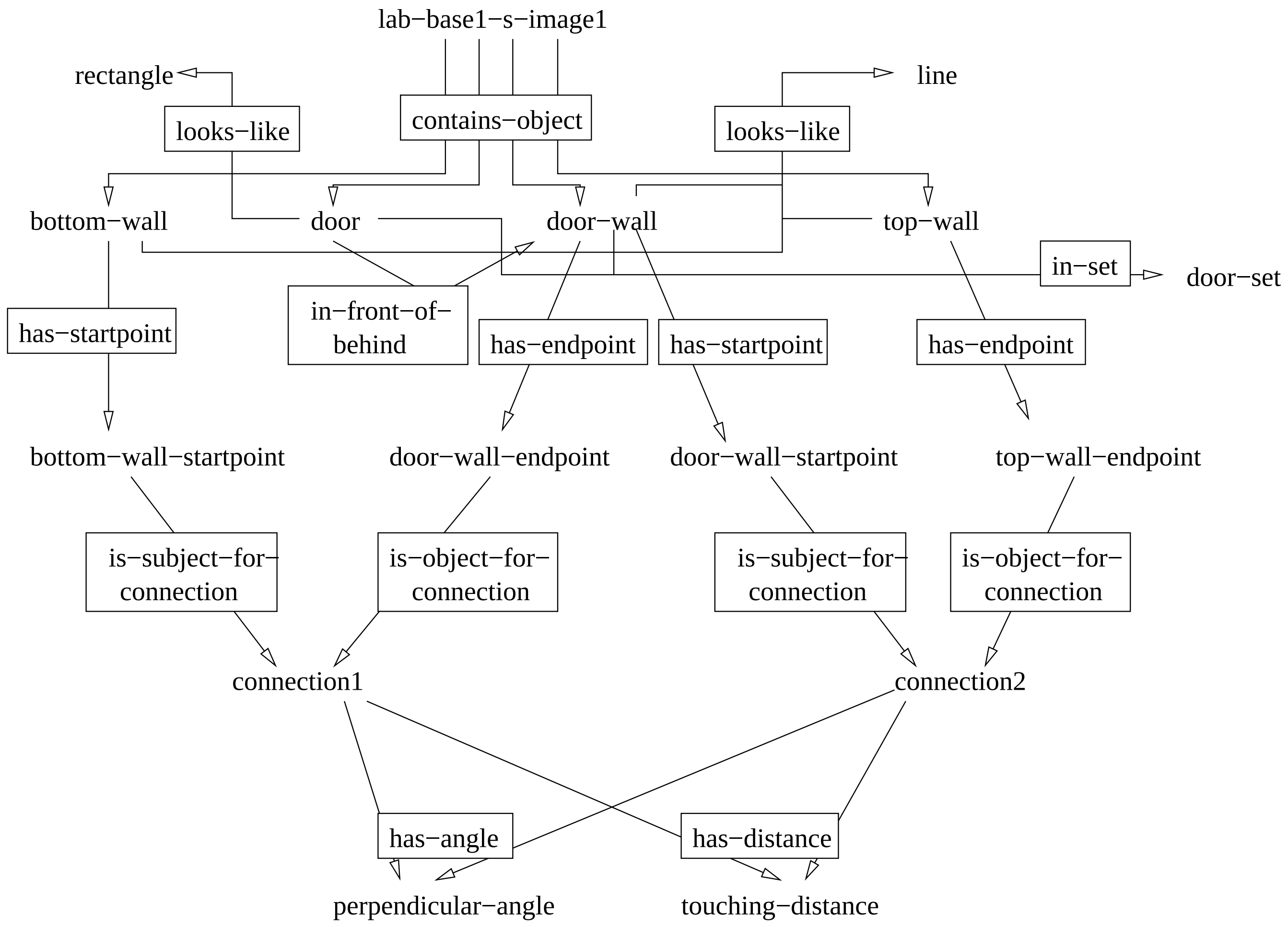


Figure 6. This Figure shows part of the first `s-image` in L14's source `s-image` series. Each `relationship` is represented as an arrow. At the beginning of the arrow is the first element of the relationship, and at the end of the arrow is the other. The boxed text in the middle of the arrow is the `Relation`. Each string of unboxed text is a `concept`.

The fifth `transformation`, another `add-connections`, places these containment walls in the correct positions in relation to the door sets and the top and bottom walls.

We will describe the first two `transformations` in detail. The first `transformation` in the lab-base1 source is a `replicate`, which takes two arguments: some `object` and some `number-of-resultants`. In this case the `object` is `door-set-b1s1` (represented as `door-set` in Figure 6. `b1s1` means "base one, `s-image` one.") and the `number-of-arguments` is `two`. The `replicate` is applied to the first L14 `s-image`, with the appropriate adaptation to the arguments: The `mapping` between the first source and target `s-images` indicates that the `door-set-b1s1` maps to the `door-set-l14s1`, so the former is used for the target's `object` argument. The number `two` is a literal, so it is transferred directly.

Using a function that takes in the name of an element instance or set (in this case `door-set-l14s1`) and recursively returns all set names and element instances, Galatea retrieves (from memory of the source `s-image` with the `replications` in it) all propositions containing any of those set names and element instances. These propositions are put through a function that creates the same number of new propositions with the same `relations` and `literals`, but with new names for the element instances. These new propositions are stored in memory. The effect of this is a replication of the intended structure. This occurs once for each replication.

Galatea chooses an arbitrary name for the superset of door-sets (in this case `door-sets-set-l14s2`) and connects `door-set1-l14s2` and `door-set2-l14s2` to it with `in-set` relations. It makes a map between L14's `s-image1` and `s-image2`, connecting `door-set-l14s1` to `door-sets-set-l14s2`. It also creates maps from `door-set-l14s1` to `door-set1-l14s2` and another to `door-set2-l14s2`.

The other propositions from L14's `s-image1` are put through a function that finds analagous propositions: `literals` and `relations` are kept the same, and element instance names are replaced with new names for the new `s-image`. For example, the `top-door-l14s1` becomes `top-door-l14s2`.

`Maps` between the element instances in the target `s-image1` and the target `s-image2` are stored in memory as well.

The `mapping` between `lab-base1-simage2` and `l14-simage2` is automatically generated. Element instances that are results of source `transformations` are mapped to newly-generated instances in the target. All other maps are carried over to the new `s-images` with their new names.

The inputs to `replicate` are the object to be replicated `DOOR-SET-L14S1`, the number of resultants `2`, the current and next target `s-images` `L14-SIMAGE1` and `L14-SIMAGE2`, the `mapping` `L14-SIMAGE1--L14-SIMAGE2--MAPPING1` and the memory.

The second `transformation` is `add-connections`. The effect of this `transformation` is to place the replicated door-sets in the correct spatial relationships with the other element instances. It takes `connection-sets-set-b1s3` as the `connection/connection-set` argument. This is a `set` containing four connections. Galatea uses a function to recursively retrieve all connection and set proposition members of this set. These propositions are put through a function which creates new propositions for the target. Each proposition's `relation` and `literals` are kept the same. The element instance names are changed to newly generated analogous names. For example, `door1-endpoint-b1s3` turns into `door1-endpoint-l14s3`.

Then, similarly to the `replicate` function, horizontal target maps are generated, and the other propositions from the previous `s-image` are instantiated in the new `s-image`.

The inputs to this transformation are `nothing` (denoting that there is not any thing in the previous `s-image` that is being modified), the connection set `connection-sets-set-b1s3`, the source `s-image lab-base1-simage2`, the current and next target `s-images l14-simage2 and l14-simage3`, the `mapping l14-simage2--l14-simage3--mapping1` and the memory.

It is important that 1) differences between the analogs can be represented in Covlan and 2) that the procedure can be transferred between the analogs. Too see the differences between the analogs, we now examine what made L14 (Figure 4) differ from the stimulus drawing: L14 features a longer vestibule in the drawing than the vestibule pictured in the stimulus. In fact, there is no trimmer arm (analogous to the wall in the lab problem) in the drawing at all that is distinct from the vestibule, save a very small section, apparently to keep the spinning trimmer blade from hitting the vestibule. The entire drawing is rotated ninety degrees from the source. The single lines in the source are changed to double lines in the target. The doors also slide in and out of the vestibule walls. What's interesting about this modification is that it does not appear that this kind of door opening is possible with the diagram given of the lab in the source: Since the door is a rectangle that is thicker than the lines representing the walls, the door could not fit into the walls. In contrast L14 explicitly makes the doors and walls thick (with two lines) and makes the doors somewhat thinner. L14 adds objects to the target not found in the source: a blade and a twisting mechanism to describe how the doors can work. L14 also included numerical parameters to describe the design of the trimmer, to describe length. Finally, L14 includes some mechanistic description of how the trimmer would work.

In summary, these differences are:

1. long vestibule
2. rotation
3. line to double line
4. sliding doors
5. added objects
6. numeric dimensions added
7. mechanisms added

Of these seven differences, Galatea successfully models four of them. The *rotation* of the source is modeled by a rotation in the target start `s-image`. In this `s-image`, all spatial relationships are defined only relative to other element instances in the `s-image`. Each instance is a part of a single set which has an orientation and direction. In the case of `s-image` 1 of the target, it is facing right. Since all locations are relative, there is no problem with transfer and each `s-image` in the model of L14 is rotated to the right.

The *line to double line* difference is accounted for by representing the vestibule walls with rectangles rather than with lines, as it is in the source. Because the `mapping` between the source and target correctly maps the `side1` of the rectangle to the `startpoint` of its analogous line, the rectangle/line difference does not adversely affect processing and transfer works smoothly.

The *long vestibule* difference is accounted for by specifying that the heights of the vestibule wall rectangles are `long`. In the source the vestibule wall lines are of length `medium`, but this does not interfere with transfer.

The trimmer head *added object* is accounted for by adding a circle to the first `s-image` in the target.

Unaccounted for are the two bent lines emerging from the vestibule on the left side, the numeric dimensions and words describing the mechanism. Also, L14 shows one of the doors retracting, and the model does not. The model also fails to capture the double line used to connect the door sections, because the single line is transferred without adaptation from the source. This could be fixed, perhaps, by representing the argument to the `add-component` as a function referring to whatever element is used to represent another wall, rather than as a `line`.

.

5. Related Work

Though much work on visual languages involves visual programming aids, there are some visual languages meant to model the visio-spatial representations in diagrams. Liu's PI system (19) represents objects (points) and operations (ruler and compass operations), deliberately limited to the Euclidean geometry domain. GeoRep (10) uses a set of "primitive shapes". Like Covlan, Georep has line segments, circles, and splines. In addition it has circular arcs, ellipses, and positioned text strings. Like Covlan GeoRep is intended to be a model of human visual reasoning. Rectangles, circles, curves, and lines are a part of Covlan because they were all found to be common visual elements of diagrams (18), suggesting that they are salient visual symbols in human beings. Covlan has connections and sets to represent simple grouping and spatial relationships between elements.

Erwig and Schneider (8) represent changes for geographic visual objects. Their system allows queries with respect to what things have happened (e.g. has a tornado ever passed through Iowa?). Like GeoRep this system does not represent visual changes to systems, which Covlan does with its visual transformations.

Letter Spirit (14) is a system that takes as input a stylized letter and generates the rest of the alphabet in the same style. It has visual representation at two levels– the lower level is a bunch of lines that can compose letters, much like an LED clock. At a higher level the system represents the "roles" for parts of letters, such as the cross-post in the letter 't'. Though Letter Spirit and Galatea have some similarity in terms of functionality, due to the domain-specific nature of Letter Spirit, there is no overlap in their visual languages.

Covlan's representations for reasoning about analogies resemble those of the analogy ontology of Forbus, Mostek and Ferguson (11), though theirs is focused more on the mapping problem and ours on transfer of multi-step procedures. Covlan explicitly represents sequences of visual operations and manages the complexity involved with, for example, adding objects and keeping track of what is done with them.

6. Discussion

As stated in the introduction, our hypothesis is that Covlan enables a visual representation of problem-solving procedures that is useful for analogical transfer of a procedure from a known analog to a new problem. We have described Galatea, a computer program that uses Covlan representations to transfer procedures from analogs to new problems.

There are seven models written in Covlan and Galatea that support this claim. We described the model of L14 in this paper. In addition we modeled three additional participants from the Craig et al. experiment, a historical example from the scientific thinking of Maxwell (6), the fortress/tumor problem (4) and the cake/pizza problem (5). Each of these models uses analogical reasoning to solve a problem using only visual knowledge.

We modeled three other participants in the lab/weed-trimmer experiment. In total, these four were, by our estimation, the four most difficult. The same transformations were used to model all four. Some of the three received different images of the lab problem (e.g. top view) but all received the same text, and the same target weed-trimmer problem.

The cake/pizza example splits a cake up into some n number of pieces and transfers the decompose transformation to split the pizza into some m number of pieces.

The Fortress/Tumor problem is used in a classic cognitive psychology experiment (12; 7). Participants read a story about a general with a large army who wants to overtake a fortress. The roads to the fortress, from all around, are mined such that if large numbers of people are on them they will explode. To solve the problem the general breaks his army into smaller armies which each take different roads. They simultaneously arrive at the fortress and overtake it. Participants are then asked to design solutions to a new problem in which a patient with an inoperable tumor need to have it removed through radiation. The problem is that the radiation will destroy healthy tissue on the way in, killing the patient. Some participants are able to produce the analogous solution, which is to have several weak rays meeting at the tumor such that no healthy tissue is radiated enough to hurt it, but the tumor gets the full force.

Our model transfers the fortress solution to the tumor problem through visuospatial similarity of the symbols representing the marching armies and rays of radiation. In both the fortress and tumor problems the marching armies and rays were represented as thick lines which get decomposed into thinner lines. Thus radically different entities can share a transformation, allowing a transfer of a solution through analogy.

James Clerk Maxwell used visual analogy to derive the electromagnetic field equations (16). He theorized that magnetic forces were caused by centrifugal forces of swirling vortices of aether. These vortices were packed together, and because they spun in the same direction, he theorized that they would grind to a stop. To solve this problem he introduced "idle wheel particles" spinning in the opposite direction between the vortices. Nersessian hypothesizes that Maxwell used a visual analogy drawn from another memory to obtain the notion of the idle wheels and then transfer the notion to the vortex idea. Maxwell noted that in machine mechanics such problems are solved with idle wheels (13). But gear systems and the fluid vortex model are very different. She hypothesizes that understanding the cross-sectional model of the vortices generically as "spinning wheels" enabled Maxwell to retrieve his knowledge of gear systems which in turn enabled him to generate the abstraction of "dynamically smooth connectors" and instantiate the "idle wheels" between the vortices. In the Galatea model of this process, the vortices are abstracted into circles, representing their cross-section. The dynamically smooth connectors and idle wheels are also represented as circles, allowing the addition of the idle wheels to be transferred to the vortex s-image series.

The fact that four of these models are based on human experimental participant sketches, and one is based on historical data, lends support to our hypothesis that Covlan captures some aspects of human visual representations. As shown above, our lab/weed-trimmer models accounted for most of the differences between source and target, as displayed in the participant data, showing that the system can successfully adapt changes to new cases as humans intend them to in many cases. The processing of Galatea's models is relatively simple because the Covlan provides a symbolic vocabulary for representing visual knowledge at a useful level of abstraction: solving a difficult problem is made easier by representing visual knowledge at a level of abstraction useful for that class of problems.

References

[1] M. Beveridge and E. Parkins. Visual representation in analogical problem solving. *Memory and Cognition*, 15(3):230–237, 1987.

[2] Hernan Casakin and Gabriela Goldschmidt. Expertise and the use of visual analogy: Implications for design education. *Design Studies*, 20:153–175, 1999.

[3] David L. Craig, Nancy J. Nersessian, and Richard Catrambone. Perceptual simulation in analogical problem solving. In Lorenzo Magnani and Nancy J. Nersessian, editors, *Model-Based Reasoning: Science, Technology, and Values*, pages 167–190. Kluwer Academic: Plenum Publishers, New York, 2002.

[4] Jim Davies and Ashok K. Goel. Visual analogy in problem solving. In Bernhard Nebel, editor, *Proceedings of the International Joint Conference for Artificial Intelligence 2001*, pages 377–382, Seattle, WA, August 2001. Morgan Kaufmann Publishers.

[5] Jim Davies and Ashok K. Goel. Visual case-based reasoning II: Transfer and adaptation. In *Proceedings of The 1st Indian International Conference on Artificial Intelligence*, Hyperbad, India, 2003. Springer-Verlag.

[6] Jim Davies, Nancy J. Nersessian, and Ashok K. Goel. Visual models in analogical problem solving. *Foundations of Science*, 10(1), 2005.

[7] K. Duncker. A qualitative (experimental and theoretical) study of productive thinking (solving of comprehensible problems). *Journal of Genetic Psychology*, 33:642–708, 1926.

[8] M. Erwig and M. Schneider. A visual language for the evolution of spatial relationships and its translation into a spatial-temporal calculus. *Journal of Visual Languages and Computing*, 14(2):181–211, 2003.

[9] Martha J. Farah. The neuropsychology of mental imagery: Converging evidence from brain-damaged and normal subjects. In Joan Stiles-Davis, Mark Kritchevsky,

and Ursula Bellugi, editors, *Spatial Cognition– Brain bases and development*, pages 33–59. Erlbaum, Hillsdale, New Jersey, 1988.

[10] R. W. Ferguson and K. D. Forbus. Georep: A flexible tool for spatial representation of line drawings. *Proceedings of the 18th National Conference on Artificial Intelligence*, 2000.

[11] Kenneth D. Forbus, Thomas Mostek, and Ron Ferguson. An analogy ontology for integrating analogical processing and first-principles reasoning. In *Proceedings of American Association for Artificial Intelligece 2002*, pages 878–885, 2002.

[12] Mary L. Gick and Keith J. Holyoak. Analogical problem solving. *Cognitive Psychology*, 12:306–355, 1980.

[13] James C. Maxwell. On faraday's lines of force. *Scientific Papers*, 1:155–229, 1855-6. is this a journal or a book?

[14] Gary McGraw and Douglas R. Hofstadter. Perception and creation of alphabetic style. Technical Report SS-93-01, AAAI, 1993. In Artificial Intelligence and Creativity: Papers from the 1993 Spring Symposium.

[15] J. M. Monaghan and J. Clement. Use of computer simulation to develop mental simulations for understanding relative motion concepts. *International Journal of Science Education*, 21(9):921–944, 1999.

[16] Nancy J. Nersessian. *Faraday to Einstein: Constructing Meaning in Scientific Theories.* Kluwer, Dordrecht, 1994.

[17] R. Pedone, John E. Hummel, and Keith J. Holyoak. The use of diagrams in analogical problem solving. *Memory & Cognition*, 29, 2001.

[18] Barbara Tversky, Jeff Zacks, Paul Lee, and Julie Heiser. Lines, blobs, crosses and arrows: Diagrammatic communication with schematic figures. In M. Anderson, P. Cheng, and V. Haarslev, editors, *Theory And Application Of Diagrams*, pages 221–230. Springer, Berlin, 2000.

[19] Liu Zhiqing. Semantics design of a visual language for constructing and animating geometric objects. In *1999 IEEE Symposium on Visual Languages*, Tokyo, Japan, September 1999.