Organizing and Representing Space for Visual Problem-Solving

Andrew Lovett
Kenneth Forbus
Qualitative Reasoning Group
Northwestern University
Evanston, IL, USA
andrew-lovett@northwestern.edu

Abstract

We present a qualitative, hierarchical approach for representing 2D space. Inspired by research on human vision, our approach supports computational models of visual problem-solving. A key idea is that hierarchical representations of space are constructed bottom-up; i.e., low-level representations support the construction of high-level representations. However, during problem-solving, representations are attended to top-down; the highest-level, most abstract representation contains the least detail, and thus is an ideal starting point for understanding the problem. We show how our representation scheme has been implemented in several successful models of visual problem-solving.

1 Introduction

Spatial representation is one of the key challenges in visual problem-solving. Representations must be sufficiently rich to capture the important features in a visual scene, but not so complex that they wash out those features with irrelevant details. Representations much also be flexible because the important details may vary greatly from one problem to another. For example, consider the *oddity task* (Figure 1), in which individuals see an array of images and choose the one that is different from the others. In Figure 1A, an individual must attend to the spatial relationships between the shapes to solve the problem. The features of the shapes themselves, particularly the edges in the larger shapes, are an irrelevant distractor. In contrast, in Figure 1B the edges *are* the key feature. Only by considering the number of edges or corners in each shape can one determine a solution to the problem.

Our research focuses on building computational models of visual problem-solving in humans. Such models allow our theories to be informed by both psychological research and the actual constraints imposed by the tasks. Previously [Lovett *et al.*, 2008; Lovett *et al.*, 2010], we have made two claims about people's spatial representations:

1) When possible, people use qualitative, structural representations of space [Biederman, 1987; Forbus *et al.*, 1991]. Such representations allow us to abstract out irrelevant quantitative details, such as the exact dimensions or orientations of objects in a visual scene and focus on important features, such as relative location, relative orientation, and topology (e.g., containment).

2) These representations are hierarchical [Palmer, 1977], meaning that any visual scene can be represented at multiple levels of abstraction. In our work, we distinguish between the *edge* level, the *shape* level, and the *group* level. The edge level describes the individual edges within a shape, and the relations between them. The shape level describes the shapes in a visual scene, and the relations between them. The group level describes relations between entire groups of objects. Representing an image at multiple levels simultaneously would be confusing, so we assume people represent an image at only one level at a time. Thus, one challenge in visual problem-solving is determining the appropriate level of abstraction to use when solving a problem. Here, we present two important refinements to the above claims:

1) Hierarchical representations of space are computed bottom-up but attended to top-down [Hochstein and Ahissar, 2002]. That is, in order to compute a spatial representation at any level (e.g., the shape level), one must first compute a representation at the previous lower level (the edge

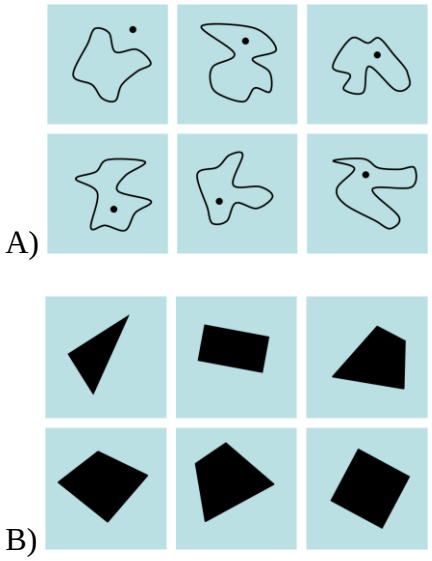

Figure 1. Oddity task problems from [Deheane *et al.*, 2006].

Lovett, A., & Forbus, K. (2011). Organizing and representing space for visual problem-solving. Proceedings of the 25th International Workshop on Qualitative Reasoning. Barcelona, Spain.

level); however, the highest-level representation is generally the best starting place for solving a problem. It is the most abstract, and thus the sparsest representation, making it the easiest to reason over. This claim is further supported by psychological research [Hochstein and Ahissar, 2002; Love *et al.*, 1999]; the human visual system appears to generate representations at a high level of abstraction, requiring explicit attention to focus in on the details of a visual scene.

2) Comparison plays a key role in both the bottom-up computation and the top-down examination of spatial representations. We model comparison as a structure-mapping process [Gentner, 1983] in which the relational structure in two representations is aligned in order to identify corresponding elements, compute differences, and determine the similarity of the things being compared.

In this paper, we describe how the above claims are implemented in our computational model. We begin by introducing two pre-existing components that play a role in our model: CogSketch [Forbus *et al.*, 2011] for spatial encoding, and the Structure-Mapping Engine [Falkenhainer *et al.*, 1989] for comparison. We then explain how our model computes hierarchical representations of space, using information at lower levels to generate representations at higher levels. We summarize the set of spatial terms in our model's qualitative vocabulary for each level. Next we consider the role comparison plays in these representations. Finally, we show how the model comes together to support visual problem-solving.

2 Model Components

Our model depends on two existing components: CogSketch for spatial encoding, and the Structure-Mapping Engine for comparison.

2.1 CogSketch

CogSketch [Forbus *et al.*, 2011] is an open-domain sketch understanding system. Given a sketch containing a set of objects, called *glyphs*, CogSketch automatically computes qualitative spatial relations between the glyphs, describing features such as relative location and topology. CogSketch does not automatically segment the sketch into glyphs; the user provides this information. Users create sketches either by drawing the glyphs—indicating where one glyph ends and the next begins—or by importing shapes from PowerPoint. The second approach allows users to create sketches based on stimuli from psychology experiments, since such stimuli often are created or can be recreated in PowerPoint.

2.2 Structure-Mapping Engine

The Structure-Mapping Engine (SME) [Falkenhainer *et al.*, 1989] is a computational model of comparison based on Gentner's [1983] structure-mapping theory of analogy. Although structure-mapping was initially proposed to explain how people compute abstract analogies, there has been increasing evidence [Markman and Gentner, 1996; Lovett *et al.*, 2009a] that structure-mapping can also explain people's concrete visual comparisons. Our approach uses SME in visual problem-solving to compare both concrete and abstract representations.

SME operates on structured, symbolic representations describing entities, attributes of entities, and relations between entities, as well as higher-order relations between other relations. Given two such cases, a *base* and a *target*, it computes one or more mappings between them by aligning their common relational structure. SME prefers to align deeper and broader structure. Because higher-order relations have more depth, they receive more weight, and thus they play a key role in the SME mappings. Each mapping consists of: 1) correspondences between elements in the base and target; 2) a similarity score based on the breadth and depth of aligned structure; and 3) a set of *candidate inferences*, guesses about the target based on expressions in the base that failed to align.

SME is useful in visual problem-solving because it can provide several pieces of information: the corresponding objects in two images (based on the correspondences), the facts common to two representations, the differences between two representations (based on the candidate inferences), and the overall similarity of two representations.

3 Hierarchical Representation

There are two necessary steps for computing a representation at any level in the spatial hierarchy: 1) perceptual organization [Palmer and Rock, 1994], in which a visual scene is divided into a set of entities; and 2) perceptual encoding, in which those entities are examined and assigned attributes and spatial relations. Our model depends on CogSketch to provide the initial entities for perceptual organization. CogSketch's glyphs become the entities at the shape level in the hierarchy. Starting with these glyphs, the model can identify the edge-level entities by decomposing a shape into edges, or find the group-level entities by joining several shapes to form a group.

Though our model begins perceptual *organization* at the shape level, perceptual *encoding*, and thus the production of finished representations, is performed bottom-up, beginning at the edge level. Each representation depends on information in the representation below it. We now describe each level in turn.

3.1 Edge Level

Organization

Our model computes edge-level representations using a single shape as input. Each shape is represented in CogSketch as a set of *polylines*, lists of points describing the lines drawn the user. The model segments polylines into perceptually salient edges by identifying junctions where two or more edges meet [Biederman, 1987]. Junctions between two edges are identified by discontinuities in the curvature. For example, a square shape would have four strong discontinuities where the four edges meet. See [Lovett *et al.*, 2009b] for details on the edge segmentation algorithm.

Attributes	Simple Edge Relations	Edge Cycle Relations
• PerceptualEdge	• edgesPerpendicular	• convex/concaveAngleBetweenEdges
• StraightEdge/CurvedEdge/EllipseEdge	• edgesParallel	• cycleAdjacentAngles
• length(Tiny/Short/Medium/Long)	• edgesCollinear	• adjacentAcuteToObtuseAngles/ adjacentObtuseToAcuteAngles
• axisAligned	• elementsConnected	• perpendicularCorner
	• elementsIntersect	• parallelEdgeRelation
	• elementIntersects	• collinearEdgeRelation

Table 1. Edge-level qualitative vocabulary.

Attributes	Simple Edge Relations	Edge Cycle Relations
• VerticalEdge	• rightOf/above	• leftToRightCorner/rightToLeftCorner/ topToBottomCorner/bottomToTopCorner
• HorizontalEdge	• edgesCurveCompati ble	
• ObliqueEdge-Upward/Downward	• edgeCurveCompatibleWith	• verticallyOrientedCorner/ horizontallyOrientedCorner
• CurvedEdge-Right/Left/Up/Down Bumped		

Table 2. Additional terms for orientation-specific representations.

After edges have been identified, they are grouped into *cycles*. A cycle is a series of consecutive connected edges that closes on itself. Thus, any closed shape will produce an edge cycle corresponding to its exterior. Cycles play a key role in perceptual encoding.

Encoding

Given the edges, junctions between edges, and edge cycles, the model computes a qualitative, structural representation describing the spatial relations between the edges. This representation consists of attributes that describe a single edge; simple edge relations that describe how two edges relate to each other; and edge cycle relations, which describe the edges found in a cycle (see Table 1).

The most basic attribute is **PerceptualEdge**, which describes any edge in a shape. Edges are further classified as **StraightEdge**, **CurvedEdge**, or **EllipticalEdge**, where the last is a single edge that closes on itself to make a cycle, such as a circle or oval. Straight edges can also be classified as **axisAligned** if they align with the vertical or horizontal axis. Finally, edges are assigned a length attribute, based on their length relative to the longest edge in the shape.

Simple edge relations are first-order relations between pairs of edges. Some describe relative orientation; two edges may be **parallel** or **perpendicular**; **parallel** edges might also be **collinear**. Others describe different ways the edges may intersect. If two edges meet at a corner, they are **connected**, but if they cross each other at an x-junction, they **intersect**. Edge cycle relations are more complex. First, every corner between consecutive edges in the cycle is classified as **convex** or **concave**. These corners then serve as the arguments for higher-order attributes and relations. **perpendicularCorner** is a higher-order attribute describing a corner between perpendicular edges. **cycleAdjacentAngles** describes two adjacent corners in a cycle. Such pairs of corners can be further classified as **acuteToObtuse** or **obtuseToAcute**, thereby capturing rich details about the shape of the cycle. Finally, **parallelEdgeRelation** and **collinearEdgeRelation**, are higher-order attributes for pairs of edges in a cycle. They are raised to the level of higher-order attributes to give them the same weight in the representation as the other edge cycle relations.

Orientation

The representation described above is primarily an *orientation-invariant* representation[1]. Orientation-invariant representations [e.g., Biederman, 1987] include no features that depend on an image's orientation, and thus they remain constant as the image rotates in space. They are useful in tasks such as object recognition, in which an object should look about the same to the viewer regardless of viewing angle. In contrast, *orientation-specific* representations [Tarr *et al.*, 1997] do include features about orientation. Thus, they contain richer details and are useful when more information is required. In our modeling, we have found that orientation-*invariant* representations are often useful at the edge level, for example when comparing the edges in two shapes that are drawn at different orientations. However, in most cases the orientation-*specific* representation is preferred when representing at the shape level or above.

[1] One term, **axisAligned**, does depend on orientation. However, we've found it to be a useful term when comparing shapes that have been rotated.

Basic Attributes	Edge-Based Attributes	Symmetry Attributes	Location Attributes
• 2D-Shape-Generic • 2D-Shape-Open/Closed • 2D-Shape-Forked • 2D-Shape-Oblique ^{O} • VerticalEdge/ HorizontalEdge^{O} • Dot-Shape • Implicit-Shape	• 2D-Shape-Convex • 2D-Shape-Curved/Straight/Ellipse • 2D-Shape-Axis-Aligned^{O} • 2D-Shape-Perpendicular	• Symmetric-Shape • Perpendicular-Symmetric-Shape • Multiply-Symmetric-Shape • Fully-Symmetric-Shape • Non-Elongated-Shape	• Centered-Element • OnTop-Element/ OnBottom-Element ^{O} • OnRight-Element/ OnLeft-Element^{O}

Size Attributes	Color/Texture Attributes
• narrowEdges/wideEdged • (Tiny/Small/Medium/Large)SizeShape	• (ObjectsColoredFn *color*) • (ObjectsBorderColoredFn *color*) • TexturedObject

Spatial Relations	Alignment Relations	Transformation Relations
• rightOf/above ^{O} • onRightHalfOf/onLeftHalfOf ^{O} • onTopHalfOf/OnBottonHalfOf ^{O} • centeredOn • elementsIntersect • elementsOverlap • elementContains	• parallelElements • perpendicularElements • collinearElements • centeredOn	• reflectedShapes-XAxis • reflectedShapes-YAxis • reflectedShapes • rotatedShapes-90 • rotatedShapes-180 • rotatedShapes

Table 3. Shape-level qualitative vocabulary. Terms marked with an ^{O} are orientation-specific.

When the task demands it, our model can supplement the qualitative terms in Table 1 with an additional set of terms (see Table 2), producing an orientation-specific representation at the edge level. Here, the attributes are more specific, supplying the orientation of each straight or curved edge, e.g., an **ObliqueEdge-Upward** is a straight edge that slants upward going from left to right, while a **CurvedEdge-RightBumped** is an edge that curves to the right. The simple edge relations describe the relative position of pairs of edges (**rightOf**/**above**), as well as curve compatibility: when two curved edges meet at a corner, is the curvature of the edges in the same direction as the curvature of the corner between them. This second relation is not strictly orientation-specific; however, it is an added detail not included in the sparser orientation-invariant representation. Lastly, the edge cycle relations describe the relative location of two edges connected at a corner (e.g., **leftToRightCorner**) and the overall orientation of a corner (a **verticallyOriented-Corner** is one in which one edge is above the other).

3.2 Shape Level

In contrast with the edge level, the shape level contains many more attributes, reflecting the greater complexity of each entity at this level. Importantly, the shape level builds upon the edge level in the sense that many shape attributes are based on edge-level features. Any time an edge attribute or relation holds for all the edges in a shape, that feature "bubbles up" to the shape level[2]. Thus, for example, a shape may be classified as **convex** if all its corners are convex, or **straight** if all its edges are straight.

Organization

Organization at the shape level is performed by the CogSketch user. Each CogSketch glyph is treated as a shape, with two exceptions: 1) If there are multiple glyphs that intersect and are not closed shapes, they will be grouped together to form a shape. For example, if two intersecting line segments have been imported from PowerPoint as sepa-

[2] Presently, we only allow features from the sparser orientation-invariant edge-level representation to bubble up.

rate glyphs, they will be grouped to form a single "X" shape. 2) A set of straight, parallel, adjacent line segments can be grouped to form a texture. In this way, the system can automatically detect simple textures and assign them to shapes.

Encoding

See Table 3 for the full set of attributes and relations. Here, we include orientation-specific and orientation-invariant terms together, although orientation-specific terms are marked with an "^{O}." Basic attributes describe general features of a shape, such as whether it is **open** or **closed** and whether it contains forks (junctions where more than two edges meet). There are also several special classes of shapes: shapes consisting of a single straight edge may be **vertical**, **horizontal**, or **oblique**. A **Dot-Shape** is a shape too small to have noticeable features. An **Implicit-Shape** is a shape that was not drawn as a glyph, but instead is implied by negative space between other glyphs.

The next two sets of attributes depend on the edge-level representation. Edge-based attributes are features that have "bubbled up" from the edge level. Symmetry attributes describe axes of symmetry detected over the edge-level representation; see section 4.1 for more information about this.

The following two sets of attributes describe relative location and relative size. Relative location is a shape's location relative to the other shapes in its image. In contrast, size is typically computed relative to the distribution of sizes across all the images being considered, so that shapes in different images with the same size will have the same size attribute. Relative size is given both for the width of a shape's edges and (for closed shapes) for the overall area of the shape.

The last set of attributes describe appearance, rather than shape. A shape may have separate colors for its border and its fill. Additionally, it may be classified as a **TexturedObject** if it is filled with a texture—a repeating series of parallel lines.

Spatial relations (Table 3, bottom left) are the basic relations computed by CogSketch between glyphs. **rightOf** and **above** give the location of one shape relative to another. The next three relations, are topological relations describing containment and intersection [Lovett and Forbus, 2009]. The final four relations describe relative location for contained shapes. For example, **onRightHalfOf** describes a shape that is contained within another shape but on its right half, whereas **centeredOn** describes a shape that is within and centered on another shape.

Alignment relations mostly apply to shapes that consist of only a single edge. Like edges, these shapes can be **parallel** or **perpendicular**. Additionally, if one shape is an edge, another shape can be **centered** on it, or a shape can be **collinear** with it if it lies along an extension of that edge. Note that **centeredOn** is thus both a spatial relation and an alignment relation.

The last set of relations describes two objects' relative shapes. One object might be a reflection or a rotation of another. These are described in greater detail in section 4.1.

3.3 Group Level

The group level is not entirely separate from the shape level. At the group level, shapes that can be grouped together are. Those that cannot are still encoded as individual shapes. Thus, there can be relations between groups, between shapes, and between groups and shapes.

The group level depends on the shape level in that only objects with the same shape can be grouped together. Thus, for example, a group might consist of a set of circles or a row of identical triangles. See section 4.1 for a description of how the model compares two objects to determine that they are the same shape.

Organization

There are several requirements for grouping shapes together; these are based upon the Gestalt grouping rules [Palmer and Rock, 1994], which describe how people tend to group objects in their visual field:

1) Similarity: Shapes should be the same shape and size.
2) Proximity: Shapes in a group should be equally distant from each other.
3) Good continuation: Shapes should be axis-aligned [Palmer, 1980]. That is, a line connecting the shapes should run along either their axes of symmetry or their axes of elongation. Because circles have an infinite number of axes of symmetry, they are the easiest shapes to group.

Encoding

The only spatial term unique to groups is **ProximalShapeGroup**, a general attribute applied to all groups. Otherwise, the qualitative vocabulary for groups is the same as that for shapes.

Some groups consist of a row of objects. Such groups may be treated the same as individual edges for alignment relations. Thus, two rows of shapes may be parallel, or a shape may be collinear with a row if it lies along that row's extension.

4 The Role of Comparison

Comparison has two important purposes in the spatial hierarchy. Firstly, during bottom-up representation building, comparison at the edge level supports perceptual encoding at the shape level. The model can determine that one shape is a rotation or reflection of another shape by comparing their edge-level representations. Similarly, the model can identify axes of symmetry in a shape by comparing its edge-level representation to itself [Ferguson, 1994].

Secondly, during problem-solving, comparisons between images can support perceptual reorganizing, in which an image is segmented into a different set of entities to make the images more similar [Medin *et al.*, 1993]. We term this *comparison-based segmentation*. We now describe each of these.

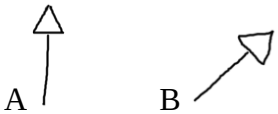

Figure 2. Two arrow shapes.

4.1 Comparing Edges in Shapes

The last set of shape-level relations are transformation relations, which describe rotations or reflections between shapes. Our approach for finding a transformation between two shapes is based on the research on mental rotation [Shepard and Metzler, 1971, in which an individual is shown two shapes and asked whether one is a rotation of the other. A common finding in studies of mental rotation is that the time required to identify a rotation between shapes is proportional to the degrees of rotation between the shapes [Shepard and Cooper, 1982]. This has led to the theory that people perform an analog rotation in their mind, mentally transforming one shape to align it with the other. However, another finding is that the time is usually proportional to the degrees of rotation along the *shortest* possible route between the shapes, suggesting that people know which direction to rotate one shape before they've even begun rotating.

We have proposed [Lovett *et al.*, 2009b] the following model to explain how people might perform mental rotations (see Figure 2 for an example):

1) Given two shapes, compare their edge-level, orientation-invariant representations. Comparison via structure-mapping will produce a set of correspondences between the edges in the two shapes.

2) Pick one pair of corresponding edges. For example, given the arrows in Figure 2, the corresponding long stems might be particularly salient. Find the shortest possible rotation between this pair. This should be easy for a single pair of edges. In this case, there is clearly a 45-degree rotation between one stem and the other.

3) Apply this rotation to the other edges in the first shape. Check whether the corresponding edges are now aligned. Applying the transformation to the full set of edges should be more difficult, and we suspect this is the part of the process that requires time proportional to the degrees of rotation.

Our model uses this approach to compare shapes and determine when there is a rotation or a reflection between the shapes. This information can then be encoded at the shape level.

A similar approach is used to identify axes of symmetry. A shape's edge-level representation is compared to itself [Ferguson, 1994] to compute one or more mappings between the edges. Each mapping is evaluated, meaning that it may be possible to find more than one axis of symmetry. There are several different symmetry attributes (see Table 2). An appropriate attribute is assigned during encoding based on the number of axes of symmetry and whether there are any perpendicular axes (such as would be found in a square, rectangle or rhombus).

4.2 Comparison-Based Segmentation

Because comparison-based segmentation takes place during problem-solving, rather than representation building, we now consider an actual problem. Figure 3 shows a progressive matrix problem, in which an individual must solve for the missing image in the 3x3 matrix. Our model solves problems like these [Lovett *et al.*, 2010] by comparing the images in each row to determine how the images change going across the row. This requires finding the corresponding entities in each row.

Let us consider the first row. Recall that representations are constructed bottom-up but attended to top-down. Thus, when solving this problem, the model would begin with a group-level representation. In each of these three images, there will be two entities: an arrow shape and a group of circles. However, these are not the optimal representations for solving this problem. Ideally, the group of circles in the middle image would be segmented into two columns of circles. This would allow the model to capture the fact that, going across the row, there is a group of circles to the left of the arrow in the first two images and to the right of the arrow in the last two images.

The model achieves this through the following steps:

1) Compare adjacent images via SME to find corresponding entities.

2) Evaluate each pair of corresponding entities by comparing their parts. For groups, this means comparing their shape-level representations. For shapes, this means comparing their edge-level representations. In this case, the arrow shapes would be compared, and the model would determine that the arrow is rotating 90 degrees. The groups of circles would be compared, and the model would determine that the two circles in the left image align with the leftmost two circles in the middle image, while the two circles in the right image align with the rightmost two circles in the middle image.

3) When possible, segment groups or shapes so that there will be identical corresponding entities in the different images. In this case, segment the four circles in the middle image into the column that will align with the left image and the column that will align with the right image.

In this way, comparison can guide perceptual

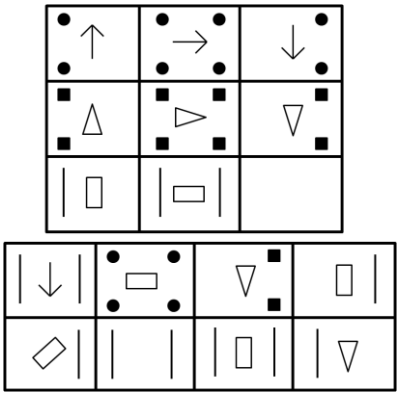
Figure 3. A progressive matrix problem. Choose which of the eight possible answers best completes the 3x3 matrix.

reorganization, in which an image is segmented into a different set of entities that better facilitates problem-solving.

5 Visual Problem-Solving

We now briefly consider three problem-solving tasks. Our models for all three tasks use the representation scheme described above.

5.1 Oddity Task

In the oddity task (Figure 1), an individual is shown an array of images and asked to pick the one that doesn't belong [Dehaene et al., 2006]. Our model of this task [see Lovett *et al.*, 2008 for a preliminary version] compares a subset of the images using SME to compute an *analogical generalization* [Kuehne *et al.*, 2000] which describes what is common to all their representations. It then compares the remaining images to the generalization to see if one of them is noticeably less similar.

The model always begins at the group level and then focuses down to the edge level if it fails to find an answer. For example, consider the problems in Figure 1. Neither of them contain any groups, so the initially representations will be equivalent to shape-level representations. In the first problem, a generalization over half the images (e.g., the bottom row) indicates that every image has a containment relation. The upper left image lacks this relation, so when it is compared to the generalization, it will be less similar.

On the second problem, the model will fail to produce an answer with the group-level representation. Because "quadrilateral" is not an attribute in the qualitative vocabulary, there is no shape-level feature that distinguishes one shape from the others. Thus, the model will move down to the edge level and try again. At this level, the model will form a generalization containing four edges with convex corners between them. The upper left image only has three edges, so its representation will be less similar.

5.2 Geometric Analogy

Geometric analogy problems (e.g., Figure 4) come in the form "A is to B as C is to…?" To solve them, one must determine the differences between images A and B, and then solve for an image D such that the same differences apply between C and D.

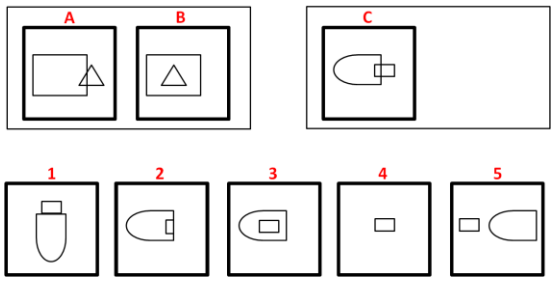

Figure 4. Geometric analogy problem from [Evans, 1968; Lovett *et al.*, 2009b]

Our model supports two alternative strategies for solving these problems [see Lovett *et al.*, 2009b for a model that only supports the first strategy]:

1) Compare images A and B to find Δ(A,B), the differences between A and B (see section 4.2 for a partial discussion of this). For each possible answer *x*, compare C to *x* to compute Δ(C,*x*). Compare Δ(A,B) to each answer's Δ(C,*x*). Pick the answer whose differences with C are the most similar to B's differences with A.

2) Compare images A and B to find Δ(A,B). Compare images A and C to find the corresponding entities between them. Apply the differences, Δ(A,B), to the corresponding entities in C to compute D', the expected answer. Compare D' to each actual answer, choosing the most similar.

Again, the model always begins with a group-level representation. It never fully reverts to the edge level, but it will perform comparison-based segmentation as needed.

5.3 Raven's Progressive Matrices

Raven's Matrices [Raven *et al*., 1998] is an intelligence test in which individuals are shown a 3x3 array of images with the bottom right image missing, and they must solve for the image that best completes the matrix. Figure 3 shows an example problem, although not one from the actual test.

Our model for solving these problems [Lovett *et al.*, 2010] is a more complex version of geometric analogy strategy 1) above. First, the model compares the images in the top row to compute Δ(top), a representation of the differences going across the top row. Then, it compares the images in the middle row to compute Δ(middle). It compares the two of these to compute an analogical generalization, Δ(row), which describes the differences going across each row of the matrix. Then, it iterates over each possible answer, inserting that answer into the bottom row and comparing the resultant Δ(bottom) to Δ(row). The answer which produces a Δ(bottom) most similar to the other rows is selected. As above, the model begins with the group-level representation and performs comparison-based segmentation as needed.

5.4 Results

Results for preliminary versions of each of these models have been reported elsewhere [Lovett *et al.*, 2008; Lovett *et al.*, 2009b; Lovett *et al.*, 2010]; see those publications for details on how the models are evaluated. We have now replicated those results using the representation scheme described above. In each case, the model performs at least as well as the typical human on the task, and problems that are difficult for the model are also difficult for people.

6 Discussion

A potential weakness of any computational model's representation is that it may be overly tailored to fit a target task. By using the same representation scheme across multiple tasks, we have provided evidence for the generality of our model. While the tasks require different processes for solving, they all begin with the same input representation. In

addition, they all apply the same principle to that representation: begin with the highest-level representation, and move down in the hierarchy as necessary to solve the problem.

Our three-level hierarchy is fairly basic—human representations of space likely possess more levels and greater flexibility. However, it is sufficient for demonstrating the utility of a hierarchical approach. Moving up the hierarchy, complex information at one level can be summarized as a single attribute or relation at the level above. Moving down the hierarchy, mappings between high-level entities can guide more in-depth explorations of corresponding parts within those entities. Future work will involve exploring more interactions between hierarchical representation and spatial cognition.

Acknowledgments

This work was supported by NSF SLC Grant SBE-0541957, the Spatial Intelligence and Learning Center (SILC).

References

[Bierderman, 1987] Irving Biederman. Recognition-by-components: A theory of human image understanding. *Psychological Review*, 94: 115-147.

[Dehaene *et al.*, 2006] Stanislas Dehaene, Véronique Izard, Pierre Pica, and Elizabeth Spelke. Core knowledge of geometry in an Amazonian indigene group. *Science*, 311: 381-384.

[Evans, 1968] Thomas Evans. A program for the solution of geometric-analogy intelligence test questions. In M. Minsky (Ed.), *Semantic Information Processing*. MIT Press: Cambridge, MA, 1968.

[Falkenhainer *et al.*, 1989] Brian Falkenhainer, Kenneth Forbus, and Dedre Gentner. The structure mapping engine: Algorithm and examples. *Artificial Intelligence*, 41: 1-63.

[Ferguson, 1994] Ron W. Ferguson. MAGI: Analogy-based encoding using regularity and symmetry. In *Proceedings of the 16th Annual Meeting of the Cognitive Science Society*, Atlanta, GA.

[Forbus *et al.*, 2011] Kenneth Forbus, Jeffrey Usher, Andrew Lovett, and Jon Wetzel. CogSketch: Sketch understanding for cognitive science research and for education. *Topics in Cognitive Science*, pp 1-19.

[Forbus *et al.*, 1991] Kenneth Forbus, Paul Nielsen, and Boi Faltings. Qualitative spatial reasoning: The CLOCK project. *Artificial Intelligence*, 51(1-3): 417-471.

[Gentner, 1983] Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, 7, 155-170.

[Hochstein and Ahissar, 2002] Shaul Hochstein and Merav Ahissar. View from the top: Hierarchices and reverse hierarchies in the visual system. *Neuron*, 36: 791-804, December 2002.

[Kuehne *et al.*, 2000] Sven Kuehne, Kenneth Forbus, Dedre Gentner, and Bryan Quinn. SEQL: Category learning as progressive abstraction using structure mapping. In *Proceedings of the 22nd Annual Meeting of the Cognitive Science Society*.

[Love *et al.*, 1999] Bradley C. Love, Jeffrey N. Rouder, and Edward J. Wisniewski. A structural account of global and local processing. *Cognitive Psychology*, 38: 291-316.

[Lovett and Forbus, 2009] Andrew Lovett and Kenneth Forbus. Computing human-like qualitative topological relations via visual routines. In *Proceedings of the 23rd International Qualitative Reasoning Workshop*, Ljubljana, Slovenia.

[Lovett *et al.*, 2010] Andrew Lovett, Kenneth Forbus, and Jeffrey Usher. A structure-mapping model of Raven's Progressive Matrices. In *Proceedings of the 32nd Annual Meeting of the Cognitive Science Society*, Portland, OR, August 2010.

[Lovett *et al.*, 2009a] Andrew Lovett, Dedre Gentner, Kenneth Forbus, and Eyal Sagi. Using analogical mapping to simulate time-course phenomena in perceptual similarity. *Cognitive Systems*, 10(3): Special Issue on Analogies – Integrating Cognitive Abilities, 216-228.

[Lovett *et al.*, 2009b] Andrew Lovett, Emmett Tomai, Kenneth Forbus, and Jeffrey Usher. Solving geometric analogy problems through two-stage analogical mapping. *Cognitive Science*, 33(7): 1192-1231.

[Lovett *et al.*, 2008] Andrew Lovett, Kate Lockwood, and Kenneth Forbus. Modeling cross-cultural performance on the visual oddity task. In *Proceedings of Spatial Cognition 2008*, Freiburg, Germany, September 2008.

[Markman and Gentner, 1996] Arthur B. Markman and Dedre Gentner. Commonalities and differences in similarity comparisons. *Memory & Cognition*, 24(2): 235-249.

[Medin *et al.*, 1993] Doug L. Medin, Rob L. Goldstone, and Dedre Gentner. Respects for similarity. *Psychology Review*, 100(2): 254-278.

[Palmer, 1980] Stephen E. Palmer. What makes triangles point: Local and global effects in configurations of ambiguous triangles. *Cognitive Psychology*, 12, 285-305.

[Palmer, 1977] Stephen E. Palmer. Hierarchical structure in perceptual representation. *Cognitive Psychology*, 9: 441-474.

[Palmer and Rock, 1994] Stephen Palmer and Irvin Rock. Rethinking perceptual organization: The role of uniform connectedness. *Psychonomic Bulletin & Review*, 1(1): 29-55.

[Raven *et al.*, 1998] John Raven, John C. Raven, and J. H. Court. *Manual for Raven's Progressive Matrices and Vocbulary Scales: Section 1. General Overview*. Oxford Psychological Press, Oxford, 1998.

[Shepard and Cooper, 1982] Roger N. Shepard and Lynn A. Cooper. *Mental Images and their Transformations*. MIT Press, Cambridge, Massachusetts, 1982.

[Shepard and Metzler, 1971] Roger N. Shepard and Jacqueline Metzler. Mental rotation of three-dimensional objects. *Science*, 171, 701-703.

[Tarr *et al.*, 1997] Michael J Tarr, Heinrich H. Bülthoff, Marion Zabinski, and Volker Blanz. To what extent do unique parts influence recognition across viewpoint? *Psychological Science*, 8(4): 282-289.