Understanding Drawings by Compositional Analogy

Patrick W. Yaner and **Ashok K. Goel**
College of Computing, Georgia Institute of Technology
Atlanta, GA 30332-0280
{yaner,goel}@cc.gatech.edu

Abstract

We describe an analogical method for constructing a structural model from an unlabelled 2D line drawing. The source case is represented as a schema that contains its 2D line drawing, the lines and intersections in the drawing, the shapes in drawing, and the structural model of the device depicted in the drawing. Given a target drawing and a relevant source case, our method first constructs a graphical representation of the lines and the intersections in the target drawing, then uses the mappings at the level of line intersections to transfer the shape representations from the source case to the target, next uses the mappings at the level of shapes to transfer the structural model of the device from the source to the target. The Archytas system implements and evaluates this method of compositional analogy.

1 Motivation and Goals

We view the task of interpreting drawings as one of constructing a model of what the drawing depicts; the model enables higher-level (i.e. non-visual) inferences regarding the depicted content of the drawing. For example, in the context of CAD environments, the input to the task may be an unannotated 2-D vector graphics line drawing depicting a kinematics device, and the output may be a structural model of the device, i.e. a specification of the configuration of the components and connections in the device. Current methods [Ferguson and Forbus, 2000; Alvarado and Davis, 2001] for extracting a model from a drawing rely on domain-specific rules. In this paper, we propose to derive a model by analogy to a similar drawing whose model is known. This requires (1) an analogical mapping from the source (known) to the target (input) drawing on the basis of shapes and spatial relations, and (2) transfer and adaptation of the model of the source drawing to the target.

Structure-mapping theory [Falkenhainer *et al.*, 1990] views analogy-based comprehension as a process of mapping individual relations from the source to the target. Candidate inferences about these mappings are guided by higher-order relations such as causal relations. While this works well in certain domains, there are no clear higher-order relations in the geometric and spatial information explicitly encoded in a drawing; at best there are only first-order spatial relations between geometric elements. Further, in this context the mapping is no longer the principal product of the analogy; our goal is not to align two representations exactly and infer a single additional relation in the target, but to transfer an *entire structural model* of a device depicted in a drawing, which may involve many relations. What we need for this is (1) a method of analogically recognizing the shapes in the target drawing, and (2) a scheme for organizing the shape and structural models of a source drawing in such a way as to enable transfer of the structural model based on shape-level differences between the target and the source drawings.

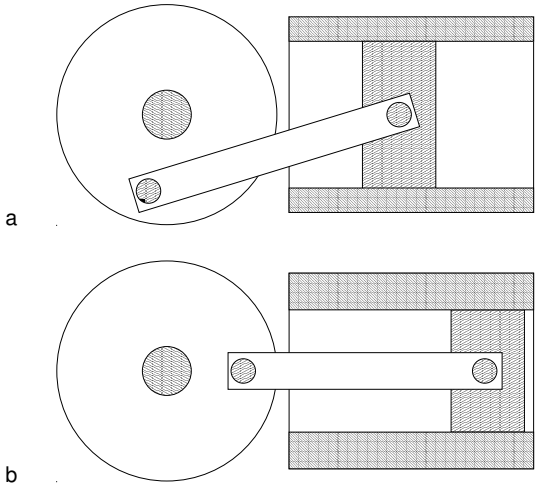

Figure 1: (a) A sample source drawing of a piston and crankshaft assembly. (b) A target drawing of the same device with the piston now at the top of its range of motion.

Let us consider the task of mapping the source drawing illustrated in figure 1(a) to the similar target drawing illustrated in figure 1(b) (figure 2 shows two more target drawings). If we have a low-level geometric reasoner to recognize the geometric elements and spatial relations among them, then we can treat this representation as a labelled graph: *A* contains *B*, *C* is adjacent to *D*, and so on. A graph-theoretic method for analogy-based recognition may then be used to find a consistent mapping between the graphs representing the source and target drawings. However, the method runs into difficulty

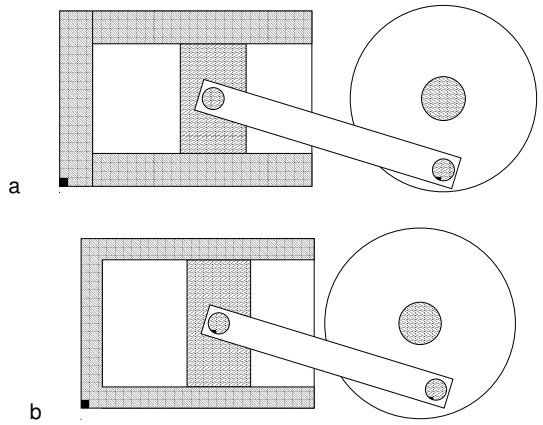

Figure 2: (a) and (b) illustrate two additional target drawings. The U-shaped cylinder in (a) is made up of three rectangles, but is a single polygon in (b), and also is thinner in (b) than in (a).

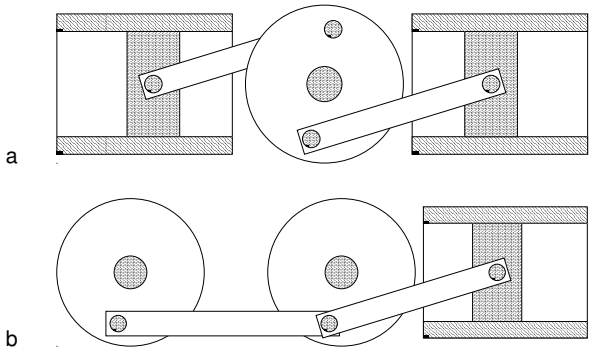

Figure 3: (a) a sample target drawing of a two piston and crankshaft assembly, and (b) a sample target drawing with two crankshafts and a single piston. In both cases, the source is from figure 1(a).

for the target drawings shown in figure 3 with figure 1(a) as the source drawing. In this problem, the *number* of components, and thus the number of shapes, is different, and either the graph-theoretic method would have to relax the constraint of one-to-one mapping, or else the analogy would have to be performed twice in order to transfer a model successfully from figure 1(a) to either figure 3(a) or (b).

To address the above difficulties, our method of compositional analogy performs analogy at multiple levels of abstraction. Figure 4 illustrates the organization of the knowledge in the source case. The structural model of the device in the source drawing specifies the components of the depicted device and their interconnections. This model is related to the drawing via the intermediate abstraction of a shape model. Our method first gathers individual lines and circles and intersection points into shapes, and then finds mappings between the source and the target at this level of intersections. Then, it groups these mappings and transfers shapes from the source to the target. Next, it finds a mapping at the shape level, and finally transfers the structural model. Mapping and transfer

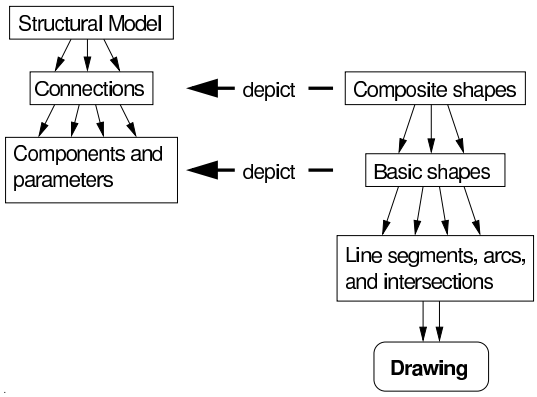

Figure 4: The multi-level hierarchy of drawing, basic shapes, and composite shapes, related to the structural model on the left with its components and connections between components. Basic shapes depict components, and composite shapes, made up of basic shapes, depict connections between components.

thus happen *iteratively* in a loop, working *up* to each subsequent level of abstraction.

The *Archytas* system implements our method of compositional analogy. Although analogical reasoning in general involves the tasks of retrieval, mapping, transfer, evaluation, and storage, Archytas presently implements only the tasks of mapping and transfer. So, given a target drawing, Archytas already knows of the source drawings relevant to the target.

2 Structural and Shape Models

The structure, shape, and drawing in a source case form an abstraction hierarchy. The structural model is a specification of components and structural relations, along with properties (height, width, etc.) and variable parameters (e.g. position, angle) of each. This structural model is similar to the representation of structure in Structure-Behavior-Function models [Goel and Chandrasekaran, 1989]. Figure 5 illustrates the graph representing the connections among the components in the structural model of the piston and crankshaft device in figure 1(a). In addition, for each component, the user specifies the coordinates of a *basic shape* corresponding to that component, and for each structural relation (or connection), the user specifies a *composite shape* as a composition of basic shapes depicting the components involved in the connection. This produces the drawing/shape/structure abstraction hierarchy shown in figure 4.

2.1 Lines, Circles, Arcs, and Intersections

The first step of the process is to match each shape in the source to the target drawing. We thus need a robust and canonical representation of shapes, one that is robust in face of the *visual binding problem*: line segments should match regardless of whether they are drawn in the vector graphics file as a single vector or as ten. For instance, the rectangle corresponding to the upper half of the cylinder in figure 1(a) might be drawn as four perpendicular lines, or it might be *six* lines (if the edge touching the piston is three segments instead of

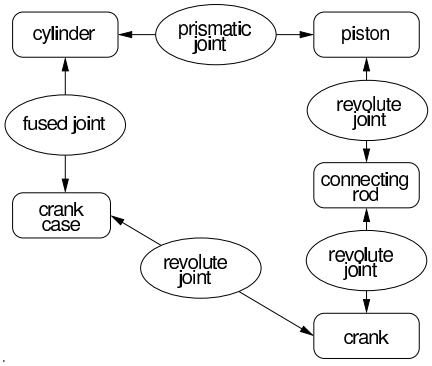

Figure 5: An illustration of the structural model, where rounded boxes represent components and ovals represent connections between components. Properties and variable parameters of components are specified in slot values for each component frame.

one); the shape must match either way. This is achieved using the graph-theoretic notion of an intersection graph.

Archytas takes each line segment and circle and circular arc as a set of geometric points in the plane, $\mathcal{S} = \{\ell_1, \ldots, \ell_k\}$, $\ell_i \subset \mathbb{R}^2$, and treats each one as the vertex of a *line intersection graph* $G = (V, E, \mathcal{L})$, $V(G) = \mathcal{S}$, and adds an edge $e \in E(G)$ between two ℓ_i, ℓ_j whenever $\ell_i \cap \ell_j \neq \varnothing$. That is, each set (line segment, etc.) is a vertex, and there is an edge between two vertices whenever the two corresponding sets intersect. Each edge $e \in E(G)$ is labelled by a point $\mathcal{L}(e) = p$, $p \in \ell_i \cap \ell_j$, so that the graph is a *multigraph* in general, with multiple edges distinguished by point labels (which is important since circles can intersect lines or other circles at two points). A *shape* is treated as a *subgraph* of this representation, so the mapping process looks for these shapes as subgraphs of the target.

Archytas thus represents the intersections between the basic elements of a 2D line drawing (line segments, circles and circular arcs) using this graph. It first reads in a drawing; fill patterns and layering are ignored, so after pre-processing, the input drawings to Archytas look like 6(a). Line segments are taken as maximally connected collinear segments, and likewise for cocircular connected arcs, while Archytas calculates all the intersection points between them. These elements form the vertices V of the line intersection graph, and the intersection points the labelled edges. Figure 6(b) shows an example of a line intersection graph.

The line intersection graph represents the most basic topological information in the drawing. To reduce the search space for analogical mapping, Archytas augments this graph with additional spatial information. Firstly, since each intersection is a pair of line (or arc) sets, Archytas adds a flag on each edge (intersection) indicating when those lines are perpendicular. This prevents Archytas from matching, say, a rectangle to a rhombus. Secondly, each topological face in the drawing bounded by a cycle of line and arc sets is represented as a special vertex linked to the points and lines on its boundary. This represents the planar dual of the input drawing (regarded as a plane graph).

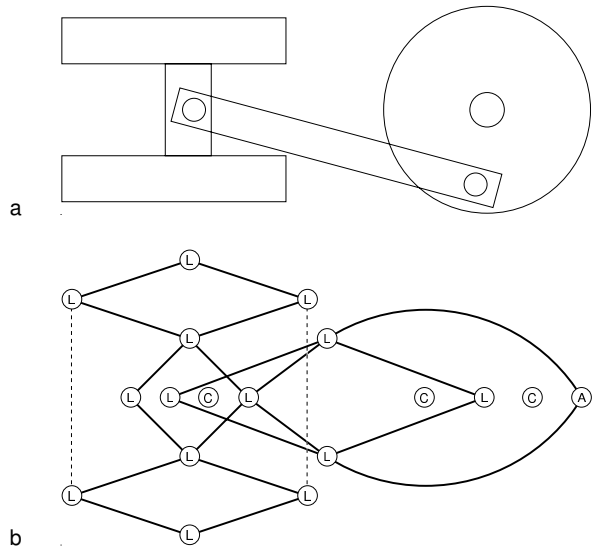

Figure 6: An example of (a) a line drawing of a piston and crankshaft with fill patterns and layering ignored (as they are in Archytas), and (b) its corresponding line intersection graph representation: L stand for for line, C for circle, A for arc (edge labels have been omitted for clarity). In addition, each bounded region (or *face*) is represented as a separate node (not shown) linked to the vertices and edges on its boundary. The dotted lines indicate collinear but disconnected lines.

2.2 Basic and Composite Shapes

Depending on the perspective of the drawing, a particular component may be depicted by an actual shape, regarded as a set of connected lines or arcs, or not at all. When a component is depicted by a shape, which is a subgraph S_i of the intersection graph. Each component in the model shown in figure 5 is linked to a subgraph of the intersection graph called a "basic shape". A connection between components in the model is a union of several entities, and so each connection is linked to a subgraph called a "composite shape." A composite shape is a union of two or more "basic shapes", forming a larger subgraph of the intersection graph. This forms the multi-level hierarchy shown in figure 4.

3 Analogical Mapping and Transfer

The goal of the mapping stage is to align shapes in the representations of the source and target drawings by finding correspondences between the basic and composite shapes in their augmented line intersection graphs. In particular, Archytas attempts to find mappings from each basic and composite shape to some subgraph of the target drawing's line intersection graph. Here the mapping must be exact: each element of the shape description must match, and so the algorithm is one of *subgraph isomorphism*.

The source drawing is divided into basic and composite shapes. The drawing in figure 1(a), for instance, has the following depicted components, each of which has one basic shape associated with it: (i) piston, (ii) connecting rod, (iii) crankshaft, and (iv) cylinder. In addition, the following struc-

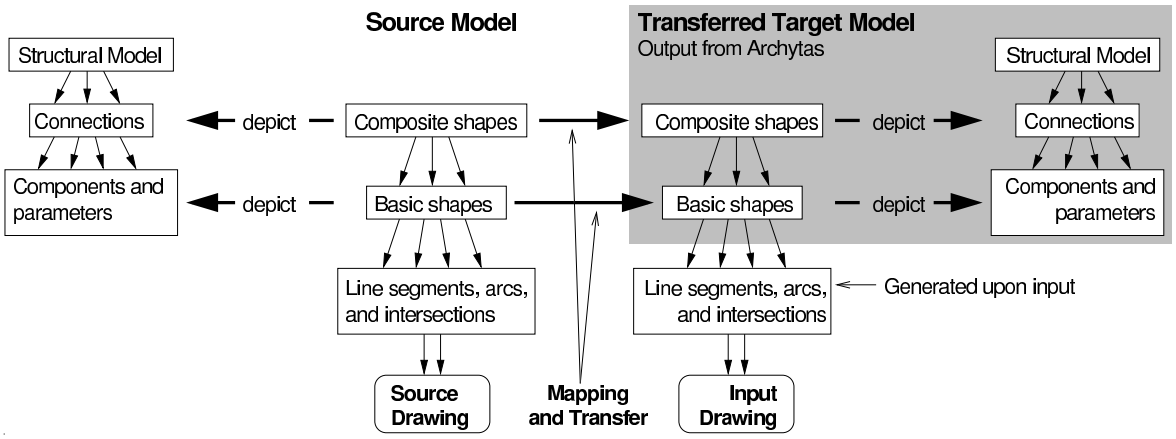

Figure 7: The multi-level hierarchy shown in figure 4 is transferred onto the target by first generating the line intersection graph of the input drawing, and then matching first composite shapes and then, when two composites overlap appropriately, basic shapes. Each basic and composite shape implies a component and structural relation, respectively, thereby suggesting a hypothetical structural model of the target.

tural relations are shown in the drawing, each of which has a composite shape associated with it: (i) cylindrical joint between piston and cylinder, (ii) revolute joint between piston and connecting rod, and (iii) revolute joint between connecting rod and crankshaft. Thus there are three composite shapes that overlap with four basic shapes. The crankcase and it's structural relations, recall, are undepicted.

In order to match these shapes to either figure 1(b) or figure 3(a), Archytas first begins by trying to compute a subgraph isomorphism from *each* composite shape to the target intersection graph. Upon finding these, Archytas breaks the composite shape mappings into corresponding basic shape mappings, so that when two composites with a basic shape in common overlap in the mapping as well, that basic shape can be inferred of the target.

Stated more formally, let's say a composite source shape S_X is made up of two basic shapes S_A and S_B, so that $S_X = S_A \cup S_B$, and another source composite shape S_Y is made up of two others, $S_Y = S_B \cup S_C$.[1] Thus, the overlap between the two composites is just the basic shape they have in common: $S_X \cap S_Y = S_B$. Then, when we have two mappings $m(S_X)$ and $m(S_Y)$, we can look for an overlap between them, $m(S_X) \cap m(S_Y)$, and when we do, we should expect it to be a valid mapping of the basic shape S_B, that is $m(S_X) \cap m(S_Y) = m(S_B)$. This may not be true, but when it is, we can infer by analogy that the basic shapes S_A, S_B, and S_C as well as the composites S_X and S_Y are present in the target.

3.1 Symmetric Mappings

In figure 1(a), the composite shape corresponding to the piston-cylinder connection has four potential mappings onto the target 1(b). The piston/connecting-rod connection has two rectangles and some circles as its constituents, and there are again four mappings to the target, but not each of them overlaps properly with each mapping for the piston-cylinder shape. Thus, if we chose the wrong pair, we might get *two* pistons instead of one, with one connected to the connecting rod, and the other connected to the cylinder. This is clearly an error of reasoning.

Note that the various mappings of a single shape in the target are *symmetric*. To see this, let $m_1, m_2 : S \to T$ be two mappings for a shape, where $S = (V,E)$ is the source shape, and $T = (V,E)$ is the target intersection graph (leaving aside the edge labels for a moment). In general, m_1 and m_2 may map the source shape onto different areas of the target, but if they do not, they still may be incompatible, for instance, with

$$m_1(a) = x \qquad m_2(a) = y$$
$$m_1(b) = y \qquad m_2(b) = x$$

If such a relationship generalizes across a pair of mappings then the mappings are *symmetric* in a certain sense. This is important from the perspective of transfer: the target shape will be identical with both mappings, so they are effectively equivalent. So, two mappings $m_i, m_j : S \to T$ are *symmetric* (and, thus, equivalent from the perspective of transfer) when $\mathrm{Rng}(m_i) = \mathrm{Rng}(m_j)$, where $\mathrm{Rng}(m)$ is the *range* of m: $\mathrm{Rng}(m) = \{y : \exists x,\ m(x) = y\}$. Another way of saying this is that the two mappings are *permutations* of each other.

We can then *group* symmetric mappings into sets. If G_X is a set of symmetric mappings for composite shape S_X, and composite shape S_X is composed of basic shapes S_A and S_B, then each mapping $m_i \in G_X$ can be divided into two mappings for the basic shapes, $m_{i,A}$ and $m_{i,B}$, and that for every such division, the $m_{i,A}$ and $m_{i,B}$ will be in the same symmetric mapping sets G_A and G_B. With this in mind, Archytas can compute *all* mappings of each composite shape in the target, and divide them into these sets of symmetric mappings. For each such set, Archytas can compute the *sets* of symmetric basic shape mappings that each composite mapping can be divided into, and then transfer one composite shape in the target for each composite shape mapping set, and one basic shape for each basic shape mapping set. The algorithms are presented in the next section.

[1] Strictly speaking, the composition of two basic shapes may contain additional edges not in either basic shape, and so $S_A \cup S_B \subseteq S_X$

3.2 Shape Mapping and Transfer

The goal of the shape transfer algorithm is to structure the target drawing: that is, to use patterns to divide the lines and intersections into basic and composite shapes, which inform Archytas of visual patterns depicting components and structural relations. It is important to compute *all* the mappings of a given shape so that all the relationships can be found. Each mapping group informs a new shape in the target. The result is a *shape-level mapping*: a mapping at the level of whole shapes rather than individual lines and intersection points. From here, the transfer of structural elements—components and structural relations—can take place.

The algorithm for shape matching is a basic backtracking constraint satisfaction algorithm with one modification: all assignments of values to variables are returned instead of just the first one found. The variables are the intersection points $e_p \in E(S)$ of the source composite shape, treated as *edge units* together with the pair of line segments or arcs that the intersection connects, $e_p = \ell_i\ell_j$, $\ell_i, \ell_j \in V(S)$. For $e_p, e_q \in E(S)$ and $e_r, e_s \in E(T)$, where $e_p = \ell_i\ell_j$, $e_q = \ell_j\ell_k$, $e_r = \ell_x\ell_y$, $e_s = \ell_w\ell_z$, and $\ell_i, \ell_j, \ell_k \in V(S)$ are distinct, two individual *maps* or *assignments*,

$$m(\ell_i\ell_j) \mapsto \ell_x\ell_y$$
$$m(\ell_j\ell_k) \mapsto \ell_w\ell_z$$

are consistent when $|\{\ell_x, \ell_y\} \cap \{\ell_w, \ell_z\}| = 1$, that is, when they have exactly one line or arc set in common (since the source intersections have exactly one such line/arc set in common, namely ℓ_j). For instance, if $\ell_y = \ell_w$, but the rest of the target vertices are distinct, the two maps are consistent. Here, the mapping is actually a two-level mapping: there is the edge mapping with $m(e_p) \mapsto e_r$ and $m(e_q) \mapsto e_s$, and there is the implied vertex mapping $m(\ell_i) = \ell_x$, $m(\ell_j) = \ell_y = \ell_w$ and $m(\ell_k) = \ell_z$.

In addition to this, for $m(a) = x$, $m(b) = y$, $a, b \in E(S)$, $x, y \in E(T)$, if a, b are both on the boundary of a common face or region in the drawing, so must x, y in the target. Finally, there are two singulary constraints: (1) perpendicular intersections map to perpendicular intersections, but non-perpendicular intersections map to either perpendicular or non-perpendicular intersections, and (2) lines only map to lines, circles to circles, and arcs to arcs.

With all this, we are all set to state the shape transfer algorithm. The outline of the algorithm is as follows:

1. Apply each composite shape, matching it to the target drawing as many times as possible using backtracking constraint satisfaction with the composite shape elements as variables, target intersection graph elements as values, and matching graph structure as constraints.
   - Group symmetric composite shape mappings
2. For each composite shape mapping, break it into its basic shape mappings
   - Group symmetric basic shape mappings
3. For each set of symmetric basic and composite shape mappings, instantiate a new shape in the target drawing
4. Return a mapping from the source shapes to the target shapes that each one instantiated

shape-analogy $(\mathcal{B}, \mathcal{C}, T)$

Input:	source basic shapes $\mathcal{B}$ source composite shapes $\mathcal{C}$ target line intersection graph T
Output:	shape-level mapping M from source shapes to new target shapes

Let $\mathcal{G}_\mathcal{B} = \varnothing$ {Basic shape mapping sets}
Let $\mathcal{G}_\mathcal{C} = \varnothing$ {Composite shape mapping sets}
Let $M = \varnothing$ {Shape-level mapping}
for each $C \in \mathcal{C}$ **do**
  Find all consistent mappings $m_C : C \to T$
for each composite shape mapping m_C, $C \in \mathcal{C}$ **do**
  Let $B_i, B_j \in \mathcal{B}$ be the basic shapes of which C is composed
  Let $m_{B_i} = \{(x,y) \in m_C : x \in B_i\}$
  Let $m_{B_j} = \{(x,y) \in m_C : x \in B_j\}$
  Find the basic shape mapping set for each of m_{B_i}, m_{B_j} in $\mathcal{G}_\mathcal{B}$, creating a new one if none is found, adding each mapping to its respective set
  Find $G_C \in \mathcal{G}_C$ by the same procedure
for each $G_B \in \mathcal{G}_\mathcal{B}$ **do**
  Let m_B be any mapping in G_B
  Let $E = \{e \in E(T) : \exists x \in E(S),\ m_B(x) = e\}$
  Let $V = \{v \in V(T) : \exists y \in V(S),\ m_B(y) = v\}$
  Let $T_B = (E, V)$ be a new target basic shape
  Create a new *shape-level* map $M(B) \mapsto T_B$
for each $G_C \in \mathcal{G}_C$ **do**
  Let m_C be any mapping in G_C
  Let $E = \{e \in E(T) : \exists x \in E(S),\ m_C(x) = e\}$
  Let $V = \{v \in V(T) : \exists y \in V(S),\ m_C(y) = v\}$
  Let $T_C = (E, V)$ be a new target composite shape
  Create a new *shape-level* map $M(C) \mapsto T_C$
Return M

Table 1: Algorithm: computes a *shape analogy*: a mapping and transfer first of individual shapes and, once transferred, a mapping of the source shapes to the newly instantiated target shapes.

Note that, in general, this shape-level mapping will *not be one-to-one*, and this is part of the idea for mapping from figure 1(a) to figure 3(a) or (b) could be found. This algorithm is described in detail in table 1.

3.3 Transfer of Structural Elements

Once Archytas has a mapping between the basic and composite shapes of the source drawing and newly instantiated shapes of the target drawing, it needs to transfer the structural model from the source to the target. From these shape-level mappings we can hypothesize that if two shapes match then they should therefore depict the same component, and likewise for composite shapes and connections. The steps are to begin with the mapped shapes, and transfer the components and connections depicted by those mapped shapes, reconstructing the model iteratively.

As input to the structure transfer process, Archytas has a set

of shape-level maps, basic shape maps and composite shape maps, each one associating a source shape with a newly instantiated target shape. From each one we can hypothesize a structural element, along with all its properties and variable parameters. The only difficulty is that some components are undepicted. For instance, in figure 1, neither figure depicts the crankcase. Archytas implements the following algorithm:

1. For each basic shape map, propose a new component in the target, and map the source component to this new target component
2. For each composite shape map, propose a new structural relation in the target, and map the source structural relation onto this new target structural relation
3. *Transfer undepicted components*: For each *undepicted* component in the source, propose *one* such component in the target, and again map the source to the target component
4. *Transfer undepicted structural relations*: For each structural relation of a *depicted* and an *undepicted* component, propose one such structural relation in the target for *every instance* of that *depicted* component in the target. For instance, if component A is undepicted and connected to component B, and A maps to A_1 and B maps to B_1 and B_2, connect A to *both* B_1 and B_2.
5. *Remove disconnected chains*: for each component in the target, check that it is connected to all of the same components (through the mapping) that it's corresponding mapped source component is, otherwise remove it and all of its structural relations. For instance, if A is connected to B and C, and A maps to A_1, B to B_1 and C to C_1, but A_1 is connected to B_1 but *not* C_1 or any other mapped C, remove A_1 and its structural relations.
6. Return a component- and structural-relation-level mapping from the old, source structure to the newly instantiated target structure.

And so the output of this process will be a new structural model of the target and a mapping from the source to the target that, once again, *may not be one-to-one*.

There are two important constraints at work. First, when transferring undepicted components, there is no guide as to how many times that component ought to be transferred, in the absence of a depicting shape to give an answer, and so the system has little choice but to transfer it just once. Here it remains an open question as to when one would transfer a given component more than once to a target, and the answer must involve model-based reasoning about the model itself.

The second issue has to do with *disconnected chains*. The shape mapping algorithm sometimes matches a shape more than once to overlapping patterns when it ought to only map once. The result of this process will be a component that is not properly connected to the rest of the model, hence a disconnected chain, which can be removed. For instance, in our piston and crankshaft example, in one drawing the connecting-rod/crankshaft composite shape matched a given drawing twice, with the two targets overlapping, when in fact the shape was only present once. The overlap was due to a line across the middle connecting rod that made it unclear whether the rod ended at one location or another. This results in a model with one piston, one crankshaft, and two rods, one that properly connects the piston to the crankshaft, one that does not. Here, however, note again that the inference is a model-based inference based on the structural model: a component was transferred in a way that is inconsistent with its role, and so it must be removed. Our heuristic is a simple one, and as the domain is explored further more complex constraints may need to be devised.

Difference	**Can handle?**
Device State	yes
Dimension	yes
Orientation	yes
Perspective	no
Component Shapes	no
Number of Components	yes

Table 2: A brief summary of the differences between the source and target drawings that Archytas can and cannot handle. The categories are differences in the state of the device, the dimensions of some of the components, the orientation of the drawing in 2 dimensions, the perspective of the drawing in 3 dimensions, differences in the shapes of components, and differences in the number of components.

3.4 Results

Our Common Lisp implementation of Archytas was run with the drawing in figure 1(a) along with its model as the source. This drawing was matched against a total of 15 different target drawings representing a range of kinds of differences between source and target. In particular, these drawings represent differences in (1) the state of the device (e.g. piston at top dead center versus the middle of its range of motion), (2) differences of the dimensions of particular shapes (e.g. thinner versus thicker cylinder walls, longer versus shorter connecting rod, etc.), (3) differences of orientation in 2-D of the whole drawing (e.g. a mirror image of the original), (4) differences in the perspective in 3-D of the drawing, (5) differences of shapes depicting components (e.g. the cylinder as one U-shaped polygon, as in figure 2(b), versus two parallel rectangles as in most of the others), and even (6) differences in the number of components (e.g. 3(a) or (b)). The results are summarized in table 2.

The most interesting cases were the targets in figure 3(a) and (b). In the case of 3(a), Archytas derived both the correct shape model and the correct structural model, with several shapes and components and connections in the source mapping to *two* targets each (the piston, the connecting rod, the cylinder, their connections, their shapes). In the case of figure 3(b), however, the shape model was derived correctly, and the structural model *nearly*, but our rule for removing disconnected chains had the unfortunate consequence of removing the connecting rod that connects the two crankshafts, and step 3 (transfer undepicted components) only transferred the crankcase once, connecting both crankshafts up to the same crankcase. Thus, more research is needed to determine what are reasonable constraints.

In 2(a), the addition of the "cylinder head" (the rectangle at

the top of the cylinder) did not interfere with the mapping and the reconstruction of the structural model, although no shape included those lines as a part of it. In figure 2(b), on the other hand, Archytas fails to map and transfer the cylinder, since its shape does not exactly match the target.

4 Related Work

We have already discussed Structure-Mapping in §1. GeoRep [Ferguson and Forbus, 2000] is a diagrammatic reasoning system which takes an arbitrary 2D line drawing as input and gives as output a description of what is depicted in the drawing. GeoRep is organized as a two-stage forward-chaining reasoner, in which a vector graphics line drawing passes through a domain-independent low-level relational describer (LLRD) that recognizes lines and polygons, and from there a high-level relational describer (HLRD) that applies a domain-specific rule set to produce a final relational description of the diagram. One application of GeoRep, JUXTA [Ferguson and Forbus, 1998], uses SME to compare two nearly identical drawings of a physical process, such as two examples of heat transfer that differ only by the thickness of a metal bar conducting the heat. JUXTA first uses GeoRep for deriving structure from shape and then SME to look for "alignable differences", and draw candidate inferences based on these differences. Here both shape and structure are derived by forward chaining, and only a single candidate inference comes by analogy. In contrast, our work derives both structure *and* shape by analogy.

The LetterSpirit program [McGraw and Hofstadter, 1993; Hofstadter and McGraw, 1995] takes a stylized seed letter as input and outputs an entire font that has the same style (or "spirit"). The system understands "roles", such as the crossbar of an f or a t. It makes new fonts by determining some attributes such as "role traits" (e.g. crossbar suppressed), and transferring these attributes to the other letters to build an entire alphabet in a new style. It overlays a grid on the seed (target) letter and all the source letters to determine the right attributes to transfer. In general, however, such a grid is not available to facilitate analogical mapping, as, for example, in figures 1(a) and (b) versus 3(a) or (b). Our work presents a method for determining shape-level differences between target and source drawings in the absence of a global coordinate system for assigning "roles". The method also uses shape-level differences to guide the transfer of a structural model from a source to the target drawing.

5 Conclusions

Visual communication between humans and machines requires AI programs that can understand drawings of different kinds. In this paper, we considered the task of understanding a 2D line drawing of a kinematics device by deriving constructing its structural model. Our method of compositional analogy derived first the shapes and spatial relations in the target drawing and then the structural model of the target drawing, both by analogy to a similar source drawing for which a structural and shape models are already known. Our method first constructs a graphical representation of the lines and the intersections in the target drawing, then uses the mappings at the level of line intersections to transfer the shape representations from the source case to the target, next uses the mappings at the level of shapes to transfer the structural model of the device from the source to the target. We have shown that by relaxing the constraints of one-to-one mapping and by interleaving the mapping and transfer processes, a model can be derived of the device depicted in a drawing by analogy without any predefined vocabulary of shapes beyond those that are actually present in a known drawing.

Acknowledgments

This research has been supported in part by an NSF (IIS) grant (Award number 0534266) on Multimodal Case-Based Reasoning in Modeling and Design.

References

[Alvarado and Davis, 2001] Christine Alvarado and Randall Davis. Resolving ambiguities to create a natural computer-based sketching environment. In *Proc. 17th Int'l Conf. on Artificial Intelligence (IJCAI-01)*, pages 1365–1371. Morgan Kaufmann Publishers, 2001.

[Falkenhainer *et al.*, 1990] Brian Falkenhainer, Kenneth D. Forbus, and Dedre Gentner. The structure-mapping engine: Algorithm and examples. *Artificial Intelligence*, 41:1–63, 1990.

[Ferguson and Forbus, 1998] Ronald W. Ferguson and Kenneth D. Forbus. Telling juxtapositions: Using repetition and alignable difference in diagram understanding. In K. Holyoak, D. Gentner, and B. Kokinov, editors, *Advances in Analogy Research*, pages 109–117. New Bulgarian University, Sofia, Bulgaria, 1998.

[Ferguson and Forbus, 2000] Ronald W. Ferguson and Kenneth D. Forbus. GeoRep: A flexible tool for spatial representation of line drawings. In *Proc. 17th National Conf. on Artificial Intelligence (AAAI-2000)*, Austin, Texas, 2000. AAAI Press.

[Goel and Chandrasekaran, 1989] Ashok K. Goel and B. Chandrasekaran. Functional representation in design and redesign problem solving. In *Proc. 11th Int'l Conf. on Artificial Intelligence (IJCAI-89)*, pages 1388–1394. Morgan Kaufmann, 1989.

[Hofstadter and McGraw, 1995] Douglas Hofstadter and Gary McGraw. Letter spirit: Esthetic perception and creative play in the rich microcosm of the roman alphabet. In D. Hofstadter and the Fluid Analogies Research Group, editors, *Fluid Concepts and Creative Analogies: Computer Models of the Funamental Mechanisms of Thought*, chapter 10, pages 407–466. Basic Books, 1995.

[McGraw and Hofstadter, 1993] Gary McGraw and Douglas Hofstadter. Perception and creation of diverse alphabetic styles. In S. Harrison, editor, *Artificial Intelligence and Creativity: Papers from the 1993 Spring Symposium*, Technical Report SS-93-01, pages 11–18. AAAI Press, Menlo Park, California, 1993.