Multimodal Analogies in Modelling and Design

Patrick W. Yaner
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280
<yaner@cc.gatech.edu>

Ph.D. Dissertation Proposal

April 27, 2005

Abstract

Drawings, on the one hand, and teleological models, on the other, are two ways of understanding and communicating design information. Drawing on previous work, Structure-Behavior-Function (SBF) theory claims that teleological knowledge is comprised of three basic kinds of knowledge: structural knowledge, behavioral knowledge, and functional knowledge. However, the design task, in practice, revolves around drawings. For example, Computer-Aided Design (CAD) software manipulates and produces drawings, and to produce *a design* is to produce a set of documents that primarily consists of *drawings* of that design. And yet, drawings can, at best, represent only structural knowledge, and even that incompletely, as components' roles in the design are not and cannot be determined solely by the drawing. Two questions motivate this research: (1) what role, precisely, do drawings place in the design process? and (2) how can we enhance CAD tools to make more direct use of teleological knowledge?

This work explores two problems: model construction and design generation. The problem of constructing an SBF model of a mechanical device from a drawing can be solved in a robust and tractable way using analogical reasoning, by reasoning from visual and topological differences to structural, causal, and functional differences in models. Patterns of adaptation and transfer can be captured by Generic Visual Teleological Mechanisms (GVTMs), generic patterns of adaptation that capture particular abstractions. Likewise, the problem of design generation, proceeding from a functional specification to a drawing associated with a complete SBF model, can be solved in a robust and tractable manner using analogical reasoning from functional differences to structural and behavioral differences in models, and ultimately to visual differences in drawings. Patterns of adaptation, here, can be effectively solved by Generic Teleological Drawing Mechanisms (GTDMs), capturing similar patterns to GVTMs, but proceeding from function to drawing instead of the reverse. Both are essentially constructive, and together help to elucidate the nature and interplay between visual structure and topology, on the one hand, and causality and teleology, on the other hand, in modelling and design.

Contents

1 Motivation

Diagrams, drawings, and sketches convey certain kinds of information very well. Shape, connection, and topology are very effectively communicated by visual means, and not as effectively or as efficiently communicated by other means. In design, the communication of the essential aspects of a design generally involves a diagram or drawing of some sort. For instance, patents, in the United States at least, according to regulations are supposed to contain drawings of the invention being patented.

Essential knowledge of a design is *teleological*, in that to know what a design is *qua* a design (as opposed to an artifact or a work of art, say) is to know its purpose, and, better yet, to know something about how it achieves that purpose. Teleological information, thus concieved, consists to three basic kinds (see, e.g., John Gero [38]) of knowledge: (1) structural knowledge of the components and connections in the device in question, and their properties; (2) behavioral knowledge of the causal interactions between components, causal processes, and causal mechanisms at work; and (3) functional knowledge of the intrinsic purpose of the design as a kind of abstraction of behavior. It is important to point out that this view of design is an engineering-centric one, in that it gives little consideration to aesthetic concerns (for example). More specifically, this work will deal with mechanical design as the design domain.

If what has just been said is correct, then design understanding must involve knowledge of the design in the form of teleological knowledge, and design generation must involve the communication of teleological knowledge. Now, teleological knowledge, as has been said, consists of structural, behavioral, and functional knowledge, but drawings can only communicate structure, at best, and even that only in an incomplete manner: a drawing cannot directly determine a component's *role* in a design. What roles, then, are drawings playing in the design process? Computer-Aided *Design* (CAD) software is *drafting* software, and does not generally deal with teleology explicitly. Can we develop technology that would allow CAD tools to make more direct usage of teleological models? These two questions will serve as motivating themes in this work.

It seems clear that whatever the proper role of drawings are, the amount of information they are capable of communicating is substantial. Randall Davis [13] gives an example of a rough hand-drawn sketch of a circuit breaker, and points out that in just a couple of sentences and some gestures to the drawing, the operation of the device can be communicated to a subject quite clearly and effectively—that is to say, a subject, upon seeing and hearing this, will claim she "understands" the design of the device. Well, what exactly does she understand? Presumably it means that the subject has some model of the device in question, however abstract. What information is being carried in a diagram or drawing of a design such that one can recall or construct some model of the thing depicted *just* from the drawing? If the ability to create and understand new designs is a measure of intelligence, then progress in understanding intelligence can be made by exploring this process.

The design process is a creative one; something must be created or produced, a new artifact whose existence is as yet only hypothetical must be proposed. The knowledge and understanding of this new hypothetical design has two broad categories: what it does, and how it does it. One needs both to do design, as one needs to produce both to have a design. In addition, fundamental to the latter is what the components of the device are and how they come together to achieve the functioning of the device. Drawings can capture much knowledge in this latter category quite clearly, but their relationship to knowledge of functioning of the device is less clear. The primary goal, then, of this work is to understand drawings: what kinds of knowledge do they capture, what sorts of inferences do they support? On the subject of design, then, the questions are: what role do they play in design, *per se*, and how can one abstract and extract such information as teleology, causality, and function of a design of a mechanical device from drawings; how can models be constructed by analogy from a drawing, and how can drawings enable design analogy?

Let us conceive of design, for the moment, as a series of attempts to vary appearances, and then map different sorts of devices and functions onto some drawing or depiction, or else to attempt different drawings looking for one that best achieves the desired purpose. The first question is how one does that initial mapping, and the second is how one produces the varied drawings at all. The first question is the one I intend to take up in this research. The focus of this work, then, is more on drawings and visual understanding then on design *per se*, but nevertheless there is some additional hope that it will shed light on the design process.

Three terms were used above: “diagram”, “drawing”, and “sketch”. The first of these terms, “diagram”, is generally taken to mean a drawn representation that conforms to particular conventions of the community or context, such as a UML diagram or an electric circuit diagram. The term “sketch”, on the other hand, is generally taken to refer to a freehand drawing, with few conventions, that is not domain specific, conveys little in the way of specific geometric constraints and more in the way of generic topological constraints. These two kinds of drawn depictions are at two ends of a spectrum: on the one hand, specific things in a diagram represent specific things in the world or about some subject in a fairly clear and unambiguous way; and on the other hand, those aspects of sketches that are doing the work of representing specific things are quite ambiguous, as is even what those specific things are or whether there are any, or if the depiction is more abstract in character. A drawing, then, might be taken to mean something intermediate between the two. More specifically a drawing is *depictive* in character, in that shapes and spatial relationships *in the drawing* reflect the shapes and spatial relationships *of the objects they depict*, observed from some point of view. This is the sense in which I shall employ the term.

There is another way of regarding this problem: much work in sketch-based recognition, such as that of Randall Davis (cited above), focuses on how to resolve ambiguities in hand-drawn sketches in order to turn them into a line drawing that can be interpreted by some model. Usually the model is quite simple—intentionally, so that they can demonstrate their methods for resolving ambiguities, and their ideas and methods for making the interface more naturally expressive for users. The hard problems generally have to do with anticipating and inferring the user's intentions. My work, on the other hand, leaves aside the questions of turning sketches into line drawings, and focuses instead on the problem of constructing a model of a device on the basis of that line drawing. As such, I begin with a line drawing, and attempt to construct a more expressive device model for it. There is another aspect to the sketch-based work, however, which is the communication of which dimensions and aspects of the sketch (or resulting line drawing) are the ones doing the work, and which are incidental, and I inherit this problem. As for what's serving the role of interpretation, the models I'm making use of are structure-behavior-function models, and are described later.

I proceed first through a high-level task analysis and a discussion of the specific research problems, where I first describe my proposed methods and hypotheses but put off discussing them in detail until much later. I then proceed through a discussion of visual structure, which will inform the kinds of representations of drawings and shape and topology that I plan to develop. Next, a discussion of the tasks, methods, and knowledge I intend to employ, followed by a discussion of testing and evaluation of my proposed system, and finally a discussion of related work, finishing with my expected contributions from this work.

2 Multimodal Analogical Reasoning and Analogy-Based Design

This section proceeds through a problem decomposition of the technologies being proposed, which will help to illuminate the issues involved. First, though, the models I intend to employ should be discussed, so that the tasks will make sense. Above I've been alluding to "models of designs" or "device models" without saying precisely what I mean. Here I set it down precisely.

The domain in which I plan to work is that of mechanical design. Broadly speaking, then, a design consists of three kinds of knowledge:

1. *Structural* knowledge: what are the components of the design, how to they connect to and interact with one another, what are the relevant dimensions and parameters that describe the design?

2. *Behavioral* knowledge: how does the system behave? What are the causal processes and mechanisms? What motions produce what other motions?

3. *Functional* knowledge: what aspects of behavior are relevant to its usage? What, essentially (as opposed to incidentally), is it supposed to *do* (as opposed to what it happens to do)?

These three kinds of knowledge of a device can be captured in a model known as a Structure-Behavior-Function (SBF) model [6, 44, 45]. I will discuss the details of these models later on, but generally speaking, the structural model is a decomposition of the device into components and connections, giving the important dimensions and aspects of each component and the parameters that can vary in behavior. The behavioral model, then is a qualitative state-transition model, showing the major transitions between qualitative states, and their immediate causes. The functional model is a specification of those constraints on the behavior (and structure, to a lesser extent) that are required for the device to do what it is supposed to do, noting that "what it is supposed to do" is only represented in terms of some abstract state or transition between states (achieving some state or pattern of motion, for instance).

Reflecting on this decomposition for a moment, we might observe that *function*, ultimately, tells you what a device is good for. A design problem can be understood in terms of function: "I need a device that achieves X", where X is a functional specification. The solution is a structure, a component model, and the behavior—the causal mechanisms at work—is what maps that structure onto function:

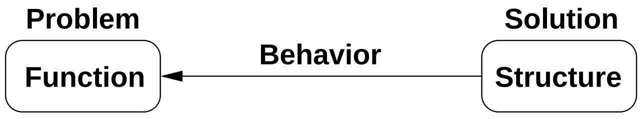

Figure 1: In the structure-behavior-function (SBF) theory of design, behavior maps structure onto function, where the function specification serves as the design problem, and the structural model as the solution.

Adding drawings to the mix adds an additional step: mapping the structure onto a drawing of that structure. Before doing this, however, we must ask how one can understand drawings at all, for this will inform what sort of drawings to produce. Thus, the first problem with which this work is concerned is to construct a model of a device from a draing, and the second is that of producing a model and a drawing from a functional specification. I proceed through each of these in turn.

2.1 Analogical Model Construction from Drawings

The system, and, more generally, the technology I am proposing to build will begin with a drawing, on the one hand, and a memory of device designs indexed by drawings, on the other. Each design consists of a teleological (SBF) model and some set of drawings that are associated with that model, the associations being detailed mappings from the representation of visual and spatial structure in the drawing to the important structural aspects of the SBF model of the device. What I would like the system and technology to be able to do is, for some input drawing, retrieve a set of matching drawings from memory and produce a set of mappings for the best ones. The best candidate design that was retrieved (best as judged from the mappings between the drawings) should then enable a transfer of an SBF model—or some adaptation of the SBF model—over to the new drawing, and the new mapping from drawing to model may then be evaluated for consistency and plausibility. That is to say, I produce a drawing of some device I'm interested in and the system builds for me a functional model of the drawn device.

More precisely, the *problem* is to construct a possible SBF model of a device or design based on just a drawing, and the proposed *method* is analogical reasoning, or analogy-based model construction. As such, the problem decomposition of this analogical method of solving the problem of model construction looks like this:

1. Given an input (target) drawing, *retrieve* all similar drawings from memory and *produce mappings* from these sources to the target drawing.
2. Using these mappings, *select* that functional (SBF) model whose drawings have the best mappings onto the target, and select that as a candidate design, the source model, for transfer and adaptation, making special note of the differences between the drawings.
3. *Transfer* the (relevant parts of the) source model over to the target, using the mappings between the source and target drawings as a starting point, and *adapt* those aspects of this model that does not entirely fit.
4. *Evaluate* the new model in terms of consistency and plausibility.
5. *Store* the new model in memory, along with the drawing.

The range of possibilities for this will not become clear until some more detailed examples and an outline of the processes and models is considered.

2.2 Analogical Design Generation From Functional Differences

For purposes of comparison, I will here outline the approach from a functional perspective. This approach was developed in the IDEAL system, developed by Bhatta et al. [6], which worked as follows, in broad outline: beginning with a problem stated as a function ("design me a device that does *this*"), return a case (SBF model) that matches that functional specification and attempt to adapt it to the new case. Here, we may only have to change a few details of the old model, or we may have to abstract away the basic mechanism and instantiate it in a new way in the new case. In this latter instance, schemas, known as *Generic Teleological Mechanisms* (GTMs), become useful.

The *problem* this time is to take a functional specification as input and produce a teleological (SBF) model that achieves or performs that function, as well as a drawing of the device. In this last aspect I am departing from the previous work. The *method* proposed is analogical reasoning, or analogical design generation. The problem decomposition of this method, then, is as follows:

1. Given an input (target) functional specification, *retrieve* all device models with similar functional specifications, producing mappings or correspondences between these sources and the target.
2. Choose, or *select* the "closest" one, and return it and a set of differences between it, the selected source, and the target functional spec (this requires an ontology of functional differences, which was a part of the IDeAL work).
3. Using these differences, *adapt* the old model to the new function. If necessary, abstract and *transfer* the knowledge of the generic mechanism that achieves the function.
4. *Evaluate* the new model for consistency and plausibility.
5. *Store* the new model in memory.

In the third step, above, it is necessary to employ schemas that abstract away from model specifics to the generic mechanisms at work. This is where the GTMs mentioned above come into play, and part of the IDeAL work involved a theory of how these GTMs are acquired and learned by example.

I begin, then, with a discussion image structure and visual reasoning, and move on to a discussion of the tasks I am attempting to perform (or, more precisely, that I will build a system to perform), explained using an example, and outlining the models and methods I am developing using this example. I then turn to the methods of evaluation and finally the discussion of related work and conclude with the expected claims and contributions of this work.

2.3 Research Problems

Design Modelling and the Design Task

One way of thinking of the above outlined idea is as a kind of "visual search task": the user is asking a question of a memory (of known designs, in this case) by drawing instead of by building up a description of the desired design in a query language designed to represent relevant constraints on models. This said, it is important that this is a *drawing*, and specifically a drawing *of the design*, and not a formalized diagrammatic expression of that same query language. The more interesting case—really, the most general case—is what to do when there is no one specific answer; one possibility is to return all "similar" devices, though they may not be exactly the same, and another possibility is to attempt to construct a plausible model (out of the models of those "similar" devices) as to what the drawing might, in fact, represent.

In fact, though, what is going on is, in a sense, *design modelling*; that is, the generation or production of a new model of a potentially new design. This constructive aspect is important: the recall and relation to memory (that is, analogy) is a proposed method by which this can be done. Thus, the first problem is the following:

Problem 1 *Given a drawing of a device, how might an intelligent agent produce a potential teleological model of the depicted device?*

The issue, then, is how to perform a given task, and my hypothesis is that this can be done by analogy to existing models on the basis of visual similarity and visual retrieval and mapping, more specifically—that is, using the drawing as an enabler for the rest of the process. More precisely:

Hypothesis 1 *Analogical reasoning offers a robust and tractable way for an intelligent agent to produce a teleological model of a device depicted in a drawing, the model being devised by analogy to existing models on the basis of the drawing, and specifically it's resemblance to drawings of other device models.*

The second basic problem, then, with which this work is concerned is stated in the following:

Problem 2 *Given a functional specification of a device, how might an intelligent agent produce a teleological model of the device as well as a drawing of that device?*

And once again my hypothesis is that this can be done using analogical reasoning:

Hypothesis 2 *Analogical reasoning offers a robust and tractable way for an intelligent agent to produce a teleological model of a device that achieves a given function, the model being devised by analogy to existing device models, and in addition can, by this same process, produce a drawing of that same device.*

These hypotheses I intend to demonstrate by building a system capable of both, the *Archytas* system I describe herein. The remaining problems discussed in this section are, properly speaking, *sub*problems of these two, which define the work.

These two problems and these hypotheses, then, immediately suggest two kinds of subproblems: those relating to the knowledge being reasoned about, and those related to the proposed reasoning processes. I discuss them in this order.

Multimodal Knowledge in Design

A drawing has both geometric and topological aspects, and both can play a role in the depiction of a device, and the device components, and the interesting properties and aspects of that device and its components:

Problem 3 *How might the geometric and topological aspects of a drawing for purposes of design and qualitative design modelling be represented?*

The answer to this, I hypothesize, is that shape, shape properties, and spatial relationship primitives can represent enough of the relevant aspects of a design (for the purposes of the tasks I am outlining):

Hypothesis 3 *Shape, shape property, and spatial relationship primitives, or visual structural primitives, can represent enough of the relevant aspects of the geometry and topology of a drawing for purposes of design.*

A design has both teleological aspects—that is, the intended purposes of the device—as well as causal aspects—the operation of the device, the actions performed and the events that take place, and the causal relationships between them. Function, then, might be thought of as mediating between the two, as representing those aspects of behavior that allow the purposes to be fulfilled. This is what I mean when I speak of "teleological models." Thus, we get the following issue:

Problem 4 *How might teleological models of devices be represented? That is, how might the relevant teleological, causal, and functional aspects of a device, for purposes of design and qualitative design modelling, be captured?*

My hypothesis is that device design can be divided into three kinds of knowledge: information about the structure of the device, the components, their connections, and their properties and means of interaction with each other; the behavior that this structure realizes, the causal primitives and their interrelationships; and the function of the device, those aspects of behavior which are the ones that allow the device to perform the task which it was designed to perform. Hence:

Hypothesis 4 *Structural, behavioral, and functional primitives can capture the relevant teleological and causal knowledge of a design at a qualitative level to effectively enable design and design modelling.*

Once again, the previous work cited above has been, in part, aimed at precisely this hypothesis, and so I will be drawing on exactly that work. But discussion of these two kinds of knowledge then leads us to the following:

Problem 5 *How might the teleological models on the one hand, and drawings, and specifically their geometric and topological aspects, on the other hand, be associated with one another in the proper way?*

A drawing, on the one hand, and a causal model, are both representations *of* some artifact. A drawing, as it stands, does not represent things in a way that is useful to a computer algorithm, and so the proper visual primitives must extract that information from the drawing. That being done, the visual representation, on the one hand, and the causal model, on the other, both represent aspects of some physical artifact, and insofar as they do, and insofar as they refer to the different roles of the *same aspects* of that physical artifact, they may be related to each other, one primitive to another:

Hypothesis 5 *Causal primitives, on the one hand, can act as pointers to just those visual structural primitives that play a role in that specific type of causality (represented by that causal primitive, that is), and the visual structural primitives, on the other hand, can likewise act as pointers to those causal primitives that represent the type of causality they depict.*

The kinds of knowledge being employed and the ways in which it gets represented will naturally enable certain kinds of inferences, and so I move on to the reasoning process.

Multimodal Reasoning in Design

Analogy proceeds with retrieval and selection, adaptation, and transfer, and evaluation and storage. The most critical of these steps, then suggest some issues with regard to process immediately. The first of them is retrieval:

Problem 6 *How can similar drawings of devices, or drawings of similar devices, be retrieved in such a way as to facilitate analogy-based model construction of a target model based on sources from memory?*

The hypothesis here is that representation of shapes, shape properties, and spatial relations, and in particular the primitives involved in such a representation, can facilitate the retrieval process by providing a language with which to guide the search process. Stated differently, since retrieval must ultimately be based on similarity of content, then the solution should be clear: with the proper representation of the relevant aspects of content for the task at hand, the retrieval should focus on similarity of representation:

Hypothesis 6 *Visual structural primitives—that is, representation of shapes, shape properties, and spatial relationships—can be assembled into semantic networks that can thereby be matched against those in memory in a tractable and robust way using constraint satisfaction.*

This draws to a large extent on previous work on the Geminus system [78, 79, 80, 81] where it has been shown that constraint satisfaction can be tractable for visual retrieval and mapping. I intend to build on that work here.

The core of the analogical reasoning process, of course, is adaptation and transfer, for the transferring of knowledge from an old to a new problem is no more and no less than *why* an agent uses analogy in the first place. Hence:

Problem 7 *How might adaptation and transfer of teleological models occur, on the basis of drawings, and visual, geometric, and topological similarity?*

This problem is somewhat more subtle than it might first appear: retrieval is based on one form of knowledge, but that which is to be transferred is of another sort, and so the two kinds of knowledge must be associated with each other in a specific and detailed way in order to facilitate the transfer:

Hypothesis 7 *The association between the visual structural primitives, on the one hand, and the causal primitives, on the other, enables adaptation and transfer of causal knowledge where similarities and differences are known in terms of visual, geometric, and topological structure, using generic patterns of adaptation called* Generic Visual Teleological Mechanisnms *(GVTMs).*

Likewise, the system I am proposing will be retrieving models based on functional specifications, but transferring causal models:

Problem 8 *How might adaptation and transfer of teleological and causal models, as well as drawings, occur, on the basis of similarities and differences in functional specification?*

And once again, since the problem surrounds the association of two kinds of knowledge, the answer is going to be to associate them in the right way, so as to enable to the desired inferences from the one to the other:

Hypothesis 8 *Structural, behavioral, and functional models provide a connection between functional and causal primitives to effectively facilitate the adaptation and transfer of the one (causal knowledge) based on similarity of the other (functional specifications), and similarly, the association between visual structural primitives and causal primitives effectively enables production of new drawings. This can be done using generic patterns of adaptation called* Generic Teleological Drawing Mechanisms *(GTDMs).*

Of course, this will draw very heavily on, and continue the previous work of Goel and colleagues on SBF models [45, 6, 44], especially in the use of GVTMs and GTDMs for transfer and adaptation, both of which build on the notion of a Generic Teleological Mechanism (GTM).

I intend to seek evidence in support of each of these hypotheses in my work by building and testing the system herein described, and by the evaluation methods outlined below.

3 Visual Structure

I describe, here, the representation of drawings, and specifically of their spatial structure. I am interested in specification rather than forward reasoning for the present purpose—that is, devising a vocabulary which adequately expresses the interesting and relevant aspects of the spatial structure of a digram, and showing how it looks in particular cases. Later, it will become necessary to consider its role in reasoning processes, but until then, I leave the issue aside.

The discussion that follows employs set-theoretic language for the specification of my ontology, rather than symbolic logic. In part, this is because I want to draw a distinction between specification languages and representation languages, and by that I mean that the specification of an ontology at the knowledge level [65] is not identical with its representation in some system. Here I am dealing with the former. Although I occasionally speak of the "representation" of something, below, this is simply for convenience of expression; the actual representation *language* will need to be developed as part of the work being proposed.

The vocabulary employed herein is a simple one of line segments, circles, and circular arcs. Shapes that cannot be assembled from these are not considered.

I explain: engineering students in courses on drafting[1] are often taught that a simple geometric vocabulary of lines, circles, and circular arcs can suffice for nearly any shape a designer could wish to employ. The reason is pragmatic, of course; these shapes are easier to machine than fancy curves assembled from, say, cubic splines (though I wonder if the increasing use of and sophistication of industrial robotics in manufacturing isn't offsetting this). In any case, useful approximations to those fancier shapes such as ellipses and s-curves can be assembled through clever arrangements of these simple primitives.

3.1 Basic Terminology

Let us begin with a *point*, a location in space. In general, Cartesian and polar coordinates are the most common means, and so I allow a point p to be either an (x, y)-coordinate or an (r, θ)-coordinate, though mostly I will stick with Cartesian coordinates. A *vector* $\vec{v}$ has magnitude and direction, and can be determined by a point (direction is going from the origin to the point itself, magnitude being the distance). A vector is little more than a different way of looking at a point; mathematically they are identical. For measuring distance between points, I, of course, use the standard Euclidean distance metric

$$\begin{aligned} d(a, b) &= \|a - b\| \\ &= \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2} \end{aligned}$$

Given a pair of vectors $\vec{a}$ and $\vec{b}$, a line is defined by the set $L = \{\vec{a} + t\vec{b} \mid t \in \mathbb{R}^n\}$. For our purposes, n will generally be 2, and so this set is equal to the set $L = \{(x, y) \mid ax + by = c\}$, where a, b, and c are real numbers. Observe that when $a = 0$ one gets a horizontal line (since it reduces to $by = c$, which yields $y = c/b$, so y must be constant), and when $b = 0$ one gets a vertical line (since, similarly, $x = c/a$). In all other cases one will get a line with a slope of $-a/b$ and y-intercept of c/b, i.e. $y = (c - ax)/b$. Also recall that any pair of points p_1 and p_2 ($p_1 \neq p_2$) will determine a line $L[p_1, p_2]$. I will sometimes write $L[p_1, \ldots, p_n]$ when a set of n points are all collinear and determine a single line (this will be useful later on).

A *line segment* $\ell = \overline{ab}$ is that portion of a line bounded by two points a and b. Note that these same two points also determine exactly the line. I will sometimes denote the line segment generated by the points a and b by $\ell[a, b]$. Likewise with lines, I will sometimes write $\ell[p_1, \ldots, p_n]$ when a set of n points $\{p_1, \ldots, p_n\}$ determine a single line (i.e. are all collinear).

A pair of vectors $\vec{a}$ and $\vec{b}$ can have an angle, $\theta(\vec{a}, \vec{b})$, which is determined entirely by their directions, not their magnitudes, and of course is a single number. Observe that order is important: if $\theta(\vec{a}, \vec{b})$ is $\pi/2$ then $\theta(\vec{b}, \vec{a})$ is $-\pi/2 = 3\pi/2$. I will generally measure angles in radians; angles always increase going in the counter-clockwise direction, following mathematical convention.

[1]Such as I myself was, once, as an undergraduate years ago.

The numbers specifying a point or vector (or angle) can, in general, be either equal to or less than each other. Thus, one can, given two points a and b, say, for instance, that $x_a < x_b$ and $y_a = y_b$, which would tell you that a is directly to the left of b. Angles can be compared similarly.

A *circle* is all points equidistant from a given center point c, i.e.

$$C = \{x \in \mathbb{R}^2 \mid d(p, x) = r\}$$

A circle can be generated from a point p and a radius r, and I will often denote it by $c[p, r]$. Given two points a and b in the set C, above, with $a \neq b$, the portion of the circle that is between the two points (going from one to the other—hence, order is important) is an *arc*. An arc A is

$$A = \{x \in c[p, r] \mid \angle apx < \angle apb\}$$

where $\angle apx$ is the angle between vectors $\overrightarrow{pa}$ and $\overrightarrow{px}$ (i.e. $\theta(\overrightarrow{pa}, \overrightarrow{px})$), and similarly $\angle apb$ is the angle between vectors $\overrightarrow{pa}$ and $\overrightarrow{pb}$ (i.e. $\theta(\overrightarrow{pa}, \overrightarrow{pb})$).[2] An arc can be determined several ways. Given a circle $C = c[p, r]$ (p being the center point), an arc can be determined by two angles θ_1 and θ_2, with $\theta_1 \neq \theta_2$, and denote the resulting arc $a[C, \theta_1, \theta_2]$. An arc can also be determined by three points on the circumference of a circle, so that $a[p_1, p_2, p_3]$ may also determine an arc, as well as a circle $c[p_1, p_2, p_3]$. There are some common algorithms for calculating the center and radius of such an arc, and hence the corresponding circle.

These, then, are the basic elements out of which everything else can be built, as I will illustrate below. Briefly, a list of the actual terms that have been introduced:

- Point $p = (x, y)$ or $p = (r, \theta)$
- Vector $\vec{v} = (x, y)$ or $\vec{v} = (r, \theta)$
- Given vectors $\vec{u}$ and $\vec{v}$, the angle $\theta(\vec{u}, \vec{v})$, measured from $\vec{u}$ to $\vec{v}$ (in radians)
- Line segment $\ell[a, b]$ given two points a and b
- Line segment $\ell[p_1, p_2, \ldots, p_n]$ when n points are all collinear and $\ell[p_1, p_n] = \ell[p_1, \ldots, p_n]$.
- Line $L[\vec{a}, \vec{b}]$ determined by two vectors $\vec{a}$ and $\vec{b}$; note $\ell[a, b] \subseteq L[a, b]$
- Line $L[p_1, p_2, \ldots, p_n]$ when n points are all collinear.
- Circle $c[p, r]$, given point p and radius r
- Arc $A = a[C, \theta_1, \theta_2]$ given a circle $C = c[p, r]$ and two angles θ_1 and θ_2; note $A \subseteq C$

[2]Recall, following the discussion of angles between vectors, above, that a common left-to-right order must be established in how angles are measured for the definition to avoid ambiguity. Thus, angles are always measured counter-clockwise.

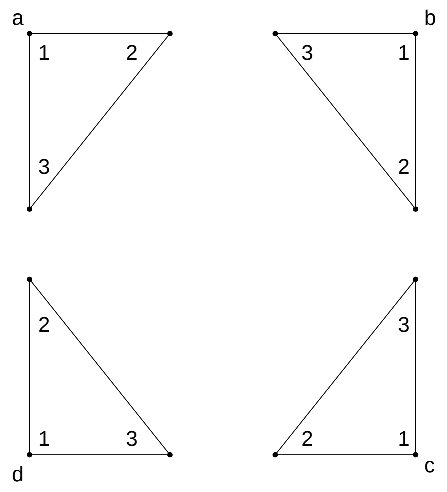

Figure 2: A sample figure with four right triangles arranged as the corners of a rectangle.

- Arc $a[p_1, p_2, p_3]$ given three points p_1, p_2, and p_3
- Given $p = (x_p, y_p)$, $q = (x_q, y_q)$, one can compare their relative locations using equalities and inequalities, e.g. $x_p < x_q$ or $y_p = y_q$.

Later on, when the actual representation is developed, the need for particular relations (e.g. "left-of", "above", what have you) will present itself, though this scheme suggests the possibility of those relations being, not atomic, but, rather, complex terms or frames of some sort. At any rate, such relations can be defined in the above terms, and so there is really no need to go through such a list just now. Part of the point of this exercise is to discover the needed relations, and to explore if this vocabulary will work for the kinds of examples I'm considering.

3.2 Outlining the Representation

In order to give a better sense of what the shape representation looks like, I'll go over some example drawings, which are all intentionally abstract. I begin with a simple example and move on to a more interesting example, later. The examples are purely geometrical and are not figures of anything in particular, just to emphasize the important aspects of the ontology.

Consider the image in figure 2. I begin by considering only the vertices. The complete set of vertices in this figure is a set of 12 points. On top of this, one can add the complete set of line segments in the drawing, of which there are also 12.

If one is interested in shape representation, then one can partition the set of line segments into four disjoint sets:

$$\begin{aligned} S_1 &= \{\ell[a_1,a_2], \ell[a_2,a_3], \ell[a_3,a_1]\} \\ S_2 &= \{\ell[b_1,b_2], \ell[b_2,b_3], \ell[b_3,b_1]\} \\ S_3 &= \{\ell[c_1,c_2], \ell[c_2,c_3], \ell[c_3,c_1]\} \\ S_4 &= \{\ell[d_1,d_2], \ell[d_2,d_3], \ell[d_3,d_1]\} \end{aligned}$$

In this case, the sets alone determine the fact that they are triangles, and so the only other interesting facts to represent about the shapes themselves are the fact that they are right triangles at particular orientations, i.e., that

$$\begin{aligned} \ell[a_1,a_2] &\perp \ell[a_3,a_1] \\ \ell[b_1,b_2] &\perp \ell[b_3,b_1] \\ \ell[c_1,c_2] &\perp \ell[c_3,c_1] \\ \ell[d_1,d_2] &\perp \ell[d_3,d_1] \end{aligned}$$

These facts together entail the shape representation of these objects. If one prefers to stick strictly to inequalities (as suggested in the last section) instead of using a special “perpendicular” relation (written as $\perp$ above), then one can state the above relations like so:

$$\begin{aligned} \theta[\overrightarrow{a_1a_3}, \overrightarrow{a_1a_2}] &= \pi/2 \\ \theta[\overrightarrow{b_1b_3}, \overrightarrow{b_1b_2}] &= \pi/2 \\ \theta[\overrightarrow{c_1c_3}, \overrightarrow{c_1c_2}] &= \pi/2 \\ \theta[\overrightarrow{d_1d_3}, \overrightarrow{d_1d_2}] &= \pi/2 \end{aligned}$$

Later I will consider adding actual relations between shapes into the mix, but for now I leave that issue aside.

This example of the four triangles is almost too trivial to be interesting. A more complex (and potentially ambiguous) example will better illustrate the important aspects of the representation. Consider, then, the drawing in figure 3, which has 22 points and 18 line segments, and some important ambiguities regarding shape identity.

Now—ignoring the circles for the time being—there is, really, no strict partitioning of the set of line segments that picks out the shapes, unless one represents two lines on top of each other. In fact, that turns out to be the most natural way to describe the figure, and so I give that partitioning:

$$\begin{aligned} S_1 &= \{\ell[p_1,p_2], \ell[p_2,p_3], \ell[p_3,p_4], \ell[p_4,p_1]\} \\ S_2 &= \{\ell[p_5,p_6], \ell[p_6,p_7], \ell[p_7,p_8], \ell[p_8,p_5]\} \\ S_3 &= \{\ell[p_9,p_{10}], \ell[p_{10},p_{11}], \ell[p_{11},p_{12}], \ell[p_{12},p_9]\} \\ S_4 &= \{\ell[p_{13},p_{14}], \ell[p_{14},p_{15}], \ell[p_{15},p_{13}]\} \\ S_5 &= \{\ell[p_{16},p_{17}], \ell[p_{17},p_{18}], \ell[p_{18},p_{15}]\} \end{aligned}$$

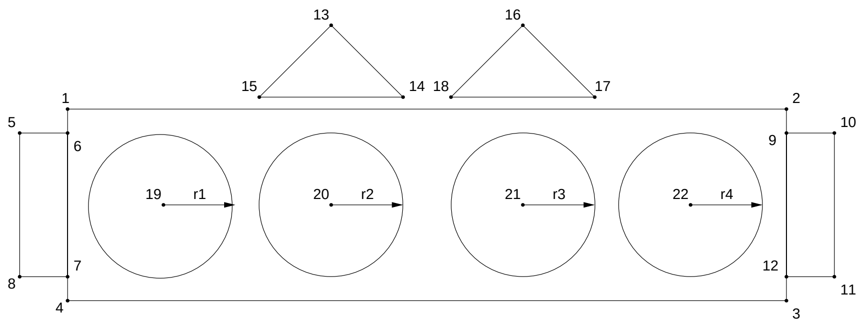

Figure 3: A sample figure. Vertices have been numbered.

Now this partitioning depends to some extent on how the drawing is drawn and represented internally,[3] but addressing this sensibly will be fruitless without considering the larger reasoning context, and so I put off the question until later.

Three of the line sets, above—S_1, S_2, and S_3—are rectangles. I'll get to their representation in a moment, but the first question is whether they should be represented as such, three rectangles, or as single shape with three rectangles as subcomponents, in a sense, or as one non-convex 12-sided polygon with a pair of chords in it (lines $\ell[p_6, p_7]$, and $\ell[p_9, p_12]$). The answer is undoubtedly going to depend on context, and specifically on what, from a non-visual standpoint, this drawing is supposed to represent. That is, if it is a digram of some device, what device is it, what is the structural model that the shapes must correspond to? Answers to these questions will suggest which of the three is most appropriate, and so one would want the representation to have the *power* to employ any of the three interpretations but to *actually* employ only the one that is appropriate.

So, let us show how the three would be represented. The three interpretations are, respectively, (1) three rectangles, (2) one complex shape consisting of three rectangles as sub-shapes, and (3) a single non-convex 12-sided polygon with two chords in its edge set. I take them one at a time.

The three rectangles are straightforward. Along with the partitioning above, to each of the three sets of lines S_1, S_2, and S_3 we would add the following:

$$\begin{aligned} d(p_1, p_2) &= d(p_4, p_3) \\ d(p_1, p_4) &= d(p_2, p_3) \\ \ell[p_1, p_2] &\perp \ell[p_1, p_4] \\ \\ d(p_5, p_6) &= d(p_8, p_7) \\ d(p_5, p_8) &= d(p_6, p_7) \\ \ell[p_5, p_6] &\perp \ell[p_5, p_8] \end{aligned}$$

[3]For instance, is $\ell[p_1, p_4]$ drawn as one line, or is it drawn as three? If the latter, then to use the partitioning above, one would have to first join such line segments together into a single segment.

$$d(p_9, p_{10}) = d(p_{12}, p_{11})$$
$$d(p_9, p_{12}) = d(p_{10}, p_{11})$$
$$\ell[p_9, p_{10}] \perp \ell[p_9, p_{12}]$$

There are several logically equivalent ways of describing a rectangle, and I have chosen only one: the first two conditions imply that it is a parallelogram (opposite sides are equal length), and the third implies that all the angles are 90°, and so it is a rectangle. One may want the actual representation for rectangle to contain all of the relevant information, or simply a lot of it, as well as a notion that only certain sets of conditions are required to justify being a rectangle, and the rest are implied by just one of those sets being true. Most likely there will be just a simple token corresponding to "rectangle", and the inference procedures will determine the rest. Exactly how to go about dealing with this will, of course, have to wait until later; for now, I simply want to illustrate the point with a set of relations that can characterize being a rectangle.

The second way to characterize the above figure, as I mentioned, is as a shape complex of three rectangles as sub-shapes. Going with the representation above, one gets three entirely separate rectangles. The fact that they meet at the sides is not represented at all. Additional information is needed. Given that this discussion treats the line segment definitions as sets of points, one could simply note that, for instance, $\ell[p_6, p_7] \subseteq \ell[p_1, p_4]$ but $\ell[p_6, p_7] \neq \ell[p_1, p_4]$, or one could simply say $\ell[p_6, p_7] \subsetneq \ell[p_1, p_4]$, indicating a proper subset. However, one also needs to represent the collinearity somehow. It will be necessary to represent the figure as one set of connected lines with three rectangles within its representation. The one line set would have to look like the following:

$$S = \left\{ \begin{array}{l} \ell[p_1, p_2], \ell[p_2, p_9], \ell[p_9, p_{10}], \ell[p_{10}, p_{11}], \ell[p_{11}, p_{12}], \ell[p_9, p_{12}], \\ \ell[p_3, p_4], \ell[p_4, p_7], \ell[p_7, p_8], \ell[p_8, p_5], \ell[p_5, p_6], \ell[p_7, p_6] \end{array} \right\}$$

With both of these pieces would yield the following:

$$d(p_1, p_2) = d(p_4, p_3)$$
$$d(p_1, p_4) = d(p_2, p_3)$$

$$\ell[p_1, p_6, p_7, p_4] = \ell[p_1, p_4]$$
$$\ell[p_1, p_2] \perp \ell[p_1, p_4]$$

$$d(p_5, p_6) = d(p_8, p_7)$$
$$d(p_5, p_8) = d(p_6, p_7)$$

$$\ell[p_5, p_6] \perp \ell[p_5, p_8]$$
$$\ell[p_6, p_7] \subsetneq \ell[p_1, p_4]$$

$$d(p_9, p_{10}) = d(p_{12}, p_{11})$$
$$d(p_9, p_{12}) = d(p_{10}, p_{11})$$

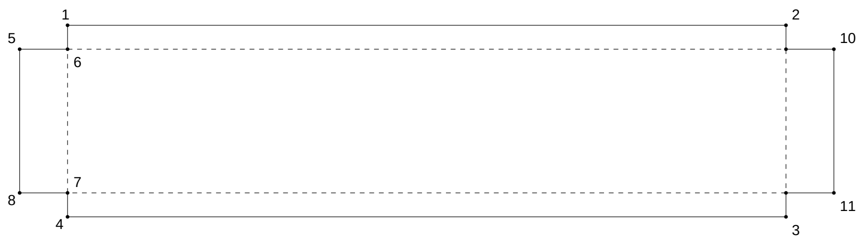

Figure 4: The outline of the rectangular shape from figure 3, with the chords (edges between points 6 and 7 and between points 9 and 12) not shown. The dashed lines indicate collinearity of on-adjacent polygon edges. Vertices, once again, have been numbered.

$$\begin{aligned}\ell[p_2, p_9, p_{12}, p_3] &= \ell[p_2, p_3]\\ \ell[p_9, p_{12}] &\subsetneq \ell[p_2, p_3]\\ \ell[p_9, p_{10}] &\perp \ell[p_9, p_{12}]\end{aligned}$$

In the above, I used $\ell[p_1, p_6, p_7, p_4]$ to indicate the left-hand side of the large rectangle. If the pattern that makes a rectangle is known, then one also would know that the above is composed of three rectangles. One way we could do this is by a kind of (micro-)analogical mapping: imagine a set of "variablized" line segments $\ell[x, y]$, $\ell[y, z]$, $\ell[z, w]$, and $\ell[w, x]$, where x, y, z, and w are the variables, and associated with that set is the following:

$$\begin{aligned}d(x, y) &= d(w, z)\\ d(x, w) &= d(y, z)\\ \ell[x, y] &\perp \ell[x, w]\end{aligned}$$

then one can represent (somehow) the fact that there are three mappings from this to the representation above, and so one has three rectangles.

For the third way of interpreting this figure, as one 12-sided non-convex polygon with two chords, observe the operative characteristics are the identity of the lines. Thus, one gets the one set of connected lines given above. The convexity property is not a local property, but it can be inferred from combinations of local properties. In fact, the property of non-convexity can be inferred from the above representation, though there are other ways of representing it. The problem is beyond that, however: I am demanding that there be an explicit representation of a non-*specific* property, in that that there is more than one way a figure can be non-convex. What I'm really interested in, here, is representing, somehow, explicitly, the interesting characteristics of the *outline* of the figure considered as a 12-sided polygon, as in figure 4. Shown this way, the shape is really the outline of two overlapping rectangles, with p_1, p_2, p_3, and p_4 being the corners of the one, as before, and p_5, p_{10}, p_{11}, and p_8 being the other. Not all the edges of the latter are actually drawn in the original drawing (figure 3).

To make this distinction concrete, let the set of edges stand as the set S defined above, and use the following representation:

$$\begin{aligned}
d(p_1, p_2) &= d(p_4, p_3) \\
d(p_1, p_4) &= d(p_2, p_3) \\
\ell[p_1, p_2] &\perp \ell[p_1, p_4] \\
\\
d(p_5, p_{10}) &= d(p_8, p_{11}) \\
d(p_5, p_8) &= d(p_{10}, p_{11}) \\
\ell[p_5, p_{10}] &\perp \ell[p_5, p_8] \\
\\
\ell[p_1, p_6, p_7, p_4] &= \ell[p_1, p_4] \\
\ell[p_2, p_9, p_{12}, p_3] &= \ell[p_2, p_3] \\
\ell[p_6, p_7] &\subsetneq \ell[p_1, p_4] \\
\ell[p_9, p_{12}] &\subsetneq \ell[p_2, p_3] \\
\\
\ell[p_5, p_6] &\subsetneq \ell[p_5, p_{10}] \\
\ell[p_8, p_7] &\subsetneq \ell[p_8, p_{11}]
\end{aligned}$$

It is important that the lines $\ell[p_5, p_{10}]$ and $\ell[p_8, p_{11}]$, which appear in the representation above, do not appear in the set S, which implies they do not appear in the drawing. Usage of hypothetical lines like this is important in establishing collinearity and the presence of patterns like this. Once again, as with the previous representation, one can observe the (potential) mapping between the variablized rectangle representation given above and the two rectangles within the above representation.

In keeping with the spirit of this representational scheme, inter-relationships between the shapes should not be represented as explicit predicates like "left-of" and "above", or what have you. Instead, one can represent them using simple inequalities and equalities between the coordinates of points, i.e., of the form $x_1 < x_2$, $y_3 = y_4$, and so on. All that would be needed is some kind of convention or limit on which points should be represented, or else the size of a description of a drawing could get to be enormous. An example of a potentially useful heuristic is to only represent such inequalities when the two points have a delaunay edge between them, treating the set of all vertices and points as a "point cloud", so to speak. Delaunay triangulations are somewhat non-trivial to explain, and a little off-topic, so let us just say, here, that it connects "nearby" points with an edge in a natural way. The complete set of delaunay edges (the union of all possible delaunay triangulations–in general, they need not be unique) for our drawing is given in figure 5.

Note that there are many more edges, here, then actual lines in the drawing, and many of the lines in the drawing do not correspond to delaunay edges. It

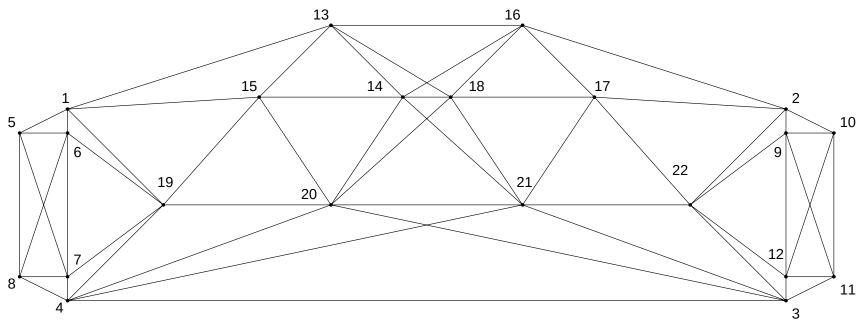

Figure 5: The complete set of delaunay edges between the vertices (and circle center points) of figure 3. In this case, the delaunay edge set is not a proper superset of the set of polygon edges, though there is considerable overlap.

should be clear that there will be *many* inequalities to give, depending on which of the three representations one adopts, so I am not going to list them all here. One could also add many of these inequalities to the shape representations.

4 Multimodal Design Tasks

4.1 An Example

The device shown in figure 6 is an instance of a standard kinematic chain called a *slider crank*, which in this case is just a basic piston and crankshaft assembly. In this device there are five components:

- The piston
- The wheel, or crank itself
- The connecting rod, connecting piston to crank
- The cylinder in which the piston moves
- The crank case in which the crank turns

The large circle is the crankshaft, the shape connecting the piston and crank is the connecting rod (the circles representing revolute joints). At the top of the connecting rod is the piston, which moves horizontally in the cylinder. The function, then, is to transform the linear motion of the piston into the rotational motion of the crank, or, equivalently, simply to produce the rotational motion of the crankshaft (the linear motion of the piston being the means by which this is achieved).

A brief behavioral description follows. The cylinder has no motion, and hence no behavior, as does the crank case. The two are, of course, bolted

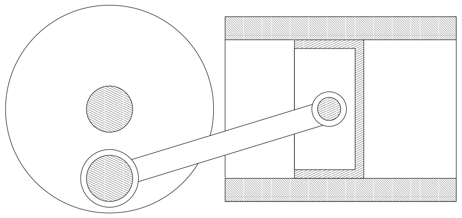

Figure 6: A piston and crankshaft.

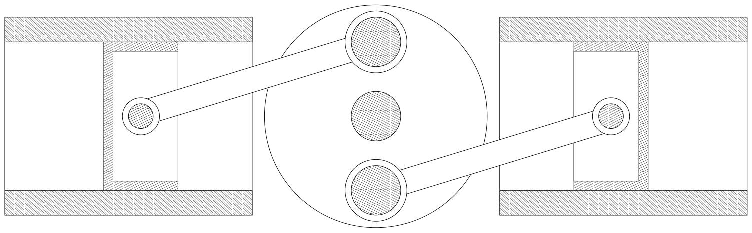

Figure 7: Two pistons driving a single crankshaft.

or fused together.[4] At any rate their only role is to serve as a ground, or frame of reference, against which other motions are defined, and to restrict the motion of the piston and crankshaft. The piston, then, moving inside the cylinder, has a simple periodic linear motion, characterized by moving vertically between two extrema, at the top and the bottom of its motion. We're dealing with an incomplete model, here, as the forcing part (that which forces the piston downward) is not present in the model. Something must force the piston downwards, and the rotational momentum of the crank forces the piston back up.

The crank, then, moving in the crank case, simply has rotational motion. The only complication is that when the piston is forcing it, it will accelerate, and when it is not, friction will decelerate it. The connecting rod is slightly complicated. Relative to the piston, it swings back and forth like a pendulum. Relative to the reference frame of the cylinder, its motion may be characterized by considering each end separately: the top end moves in linear motion exactly like the piston, and the bottom end moves in rotational motion exactly like the crank. The function of the device, then, is to transform the linear motion of the piston into the rotational motion of the crank.

As an example problem, imagine the task is to produce a model of a device with two pistons driving a single crankshaft, as in figure 7. Once again, the idea is that a drawing would be provided, and the system would attempt to construct a model from what it knows to the new device. This examples has the pattern of taking some subassembly of the original device and duplicating

[4]Really, they are one component, but calling them both "cylinder" is somewhat misleading. In a car or truck engine, say, the one component would simply be called the "engine block".

it elsewhere in a way that changes and adds to the old design in a substantial way.

Model Outline

The models I'm considering for this work are a variation on the theme of structure-behavior-function (SBF) models [6, 44, 45], as stated above. The idea behind these models is that a conceptual design of a device has three basic aspects: the structure, that being the components and the connections between them, and their dimensions and aspects as components of the design that help achieve some function; the behavior, the causal model of the device's operation; and the function of the device, that is, that aspect of behavior by which the device can be employed for some purpose.

Bylander and Chandrasekaran [9] devised a component-substance ontology which was used as the original basis for SBF models. In short, this ontology holds that devices behave by transporting substances around or changing the state of components (say, a valve goes from "open" to "closed"). This worked fine for pumps and heat exchangers, provided heat could be regarded as a substance (a simplification that much early qualitative physics work made), and for electrical devices, if electricity is regarded as a substance. In this formulation, a substance is anything that is conserved. However, once you get to mechanical devices, it begins to break down: can momentum be regarded as a substance? With changes in mechanical motion, what is being conserved is not momentum, exactly, but energy. In fact, that, as well as matter, is what is conserved in the other examples, as well: matter and energy.

However, modern physics has no precise ontological theory of energy, only a rather large set of calculations that can be made on different sorts of physical phenomena together with the observation that there is a certain quantity that always comes out the same in those calculations (cf Feynman's lectures on physics, specifically the lecture on energy [27, chapter 4]). More interesting than the unchanging quantity (that being energy, of course) is that it seems to "move" or "transform" in that it changes location or form along with changes in motion and heat and so on. Energy is that which either determines or is determined by any change in motion (that ambiguity between energy determining motion and being determined by motion is part of the key to why physics cannot claim to have an ontology of energy: a set of equations, however useful, do not constitute an ontological theory).

Reformulating SBF models in terms of energy transfer will undoubtedly prove to be a non-trivial amount of work, and so here I stick to the formulation in terms of parameters and properties that vary along some dimension, as per qualitative physics. Such a formulation is ontologically quite neutral, however: the hard problem is the choice of parameters, and that is precisely the problem I have chosen to avoid for now. I will inevitably have to come back to it in order to complete this work.

Having said all that, what follows is a outline of some aspects the SBF representation of this device (the piston and crankshaft from figure 6), described

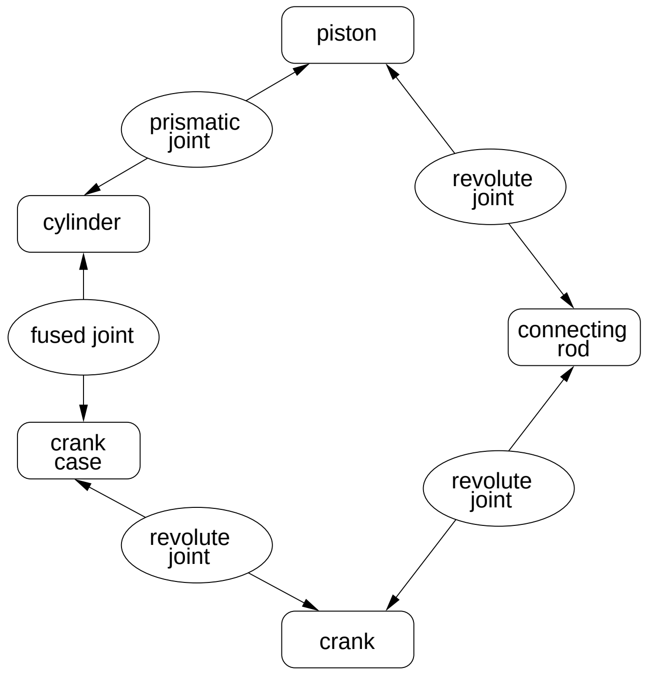

Figure 8: A diagram of the structural model of the crank and piston model, showing only the connections between the frames. A revolute joint is simply one in which one thing is free to rotate with respect to the other, and a prismatic joint is one in which the one thing slides with respect to the other. Each connection would also refer to the parameters that will be the basis of the behaviors.

at the knowledge level [65] rather than the level of data-structures.

Structural Model We begin with the structure. What follows draws to some extent on the situation calculus [61], as well as the work on the compositional modeling language (CML) [8, 20], in addition to the previous SBF work cited earlier. Figure 8 shows the components and connections of the model of the device described above. The connections between components are represented explicitly, and each one has one or more parameters and dimensions or axes along which those parameters can vary. Components, likewise, have dimensions and parameters, and participate in connections. The connections and components are where behavior takes place, and the parameters of the connections and components become the varying quantities in behaviors.

A simple way to begin to describe this, which is closer to what SBF originally had, might be to simply use the two-term relation

$$\text{connected}(A, B)$$

where A and B are components. However, the connection itself must have properties, and this notation doesn't allow for that. Even using the slightly more "frame-ish" notation

$$B \in \text{connections}(A)$$

it is still just a relation. A better way is to say

$$\begin{aligned} AB &\in \text{connections}(A) \\ AB &\in \text{connections}(B) \\ A &\in \text{components}(AB) \\ B &\in \text{components}(AB) \end{aligned}$$

which treats connections as separate entities.

This allows properties of them to be referred to. For instance, to assert that a component or connection A has some parameter p, one might say

$$p \in \text{parameters}(A)$$

This treats "connections" as a function or mapping from the set of components to the set of all sets of connections, and likewise "components" is a function from the set of connections to the set of all sets of components. Likewise with parameters. Something like joint type could be specified using simple set membership (i.e. a one-place predicate), or else it could be pulled apart into its defining properties. For instance, a revolute joint has just one degree of freedom, that being rotation about an axis, and likewise with a prismatic joint the one degree of freedom is translation along an axis. The simple fact of the connection having a parameter that is an angle of rotation or a position along an axis implies what kind of joint it is. And, since I've introduced parameters as entities that can be referred to by name, it is possible to give them properties, such as having a particular range, or being an angle as opposed to a position.

Two major roles for the structural model are (1) to specify those constraints which the behavior must conform to, and (2) to provide referents for the shape representation. The significance of each will become more clear below.

Behavioral Model The behavior, then, in terms of qualitative states and transitions, would look something like that in figure 9. The values each correspond to a joint angle or the position along an axis. The horizontal arrows are transitions between states, and the vertical arrows represent causal links between the transitions. That is, a transition from one state to another would be labelled as being *due to* another transition. Here, causal links are attached to the *transitions* and not the states; after all, it is the *change* in the value of a variable that causes some other variable to change. Also, observe that the first state repeats itself: that is, the behaviors loop.

Behavior is intimately related to the structural model. In particular, it is the components that actually move or change, but the connections which both allow and restrict behavior. Recalling the discussion of energy, above, in the

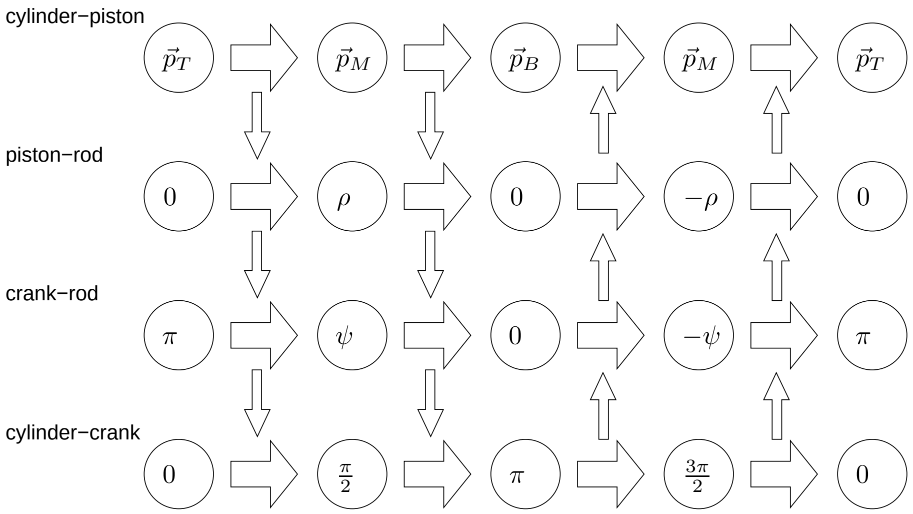

Figure 9: A diagram of the behavioral model of the crank and piston model, showing only the states, transitions, and causal links. Each state corresponds to a joint angle or a position along a particular axis. The sequence $(\vec{p}_T, \vec{p}_M, \vec{p}_B)$ corresponds to a point on the piston (say, top center of the piston) going from the top of its range of motion through the midpoint to the bottom. The angle $\rho < \pi/2$ is the widest angle the connecting rod makes on its downward motion with the piston, and the angle ψ would then be equal to $\pi/2 - \rho$. The actual states would also contain a reference to the component of which it is a state.

piston and crankshaft, an external stimulus on the piston introduces kinetic energy to the piston in the form of downward motion, and this is transmitted (or "pumped" in old SBF terminology) through the connecting rod to the crankshaft, where the kinetic energy becomes rotational motion. The variables, then, describing angles and positions in the behavior shown in figure 9 thus correspond to different forms of kinetic energy. Energy can also be dissipated through friction—i.e. converted into heat, at any moving joints. Because this is a drain on kinetic energy, this would have the effect of slowing the system down, and is why the external stimulus on the piston is needed continually, on every cycle, instead of just once to start it moving.

Drawing on the situation calculus, we could describe the behavior of some component or connection by saying that "the value of parameter p in state S is x", or, more formally,

$$\mathrm{val}(p, S) = x$$

A sequence of such assertions could constitute a description of the various of the states of the component or connection in question. A state, then, would be the values of all of the parameters of some component or connection at some time. Such statements only describe what the states are, but not their sequence or causes. Sequencing is straightforward:

$$s_2 = \mathrm{result}(a, s_1)$$

where a would be an action or event. This is where it becomes difficult: the action or event is what causes the transition, and thus knowing what kind of thing an action or event is becomes quite important. Halpern and Pearl [47, 48] take an event to be a variable taking on some value (the intention being to describe a *random* variable, since their discussion draws on previous work in causality from the perspective of Bayesian reasoning). Although this is ontologically quite neutral, the intention was for these to correspond to observations.

At any rate, the problem here becomes quite subtle: we need to be able to refer to a transition—a change in the value of a variable, say—as being the cause of another transition. Causes cannot simply be logical connections, as they so often are (e.g. in much of the literature on situation calculus), as that would require a second-order logic to be able to state what needs to be stated. A relatively straightforward way out, drawing on situation calculus, once again, might be to allow the transition itself to be an entity just like the states. If, for instance, one has the following:

$$\begin{aligned} \mathrm{val}(p, s_1) &= v_1 \\ \mathrm{val}(p, s_2) &= v_2 \\ s_2 &= \mathrm{result}(a, s_1) \end{aligned}$$

Then one might describe the transition by

$$t_{1,2} = \mathrm{transition}(s_1, s_2)$$

which would allow one to refer to the transition explicitly as a cause or an effect, or even has holding certain invariants.

Following this notation, the behavior from figure 9 could be represented by a straightforward sequence of statements of the above form.

As an aside, I have followed an important heuristic in devising all of this: although I use a logical and mathematical notation freely, I do not use logical connectives (implications in particular) and quantifiers. Such notation puts the representation squarely in the realm of first-order logic, and that makes devising algorithms and data structures an order of magnitude more difficult. As long as the statements above are just *terms* in logic (that is, a single predicate over one or more variables), then they can be represented in a straightforward way in a computer. For instance, the notation "$p(x) = y$", if x is a meaningful entity, can be represented by making x an instance of a frame, p as a slot in that frame, and y as the value of that slot. Terms like $r(a, b) = x$ are more difficult, but still well within the realm of possibility: they only stretch our notion of what a frame is very slightly. Having said that, the statements defining a *class* are necessarily statements of first-order logic, quantifiers and all. But that is only relevant if one intends to do a significant amount of reasoning *about* the classes themselves, as opposed to merely reasoning about the entities, which is what is going on here.

Functional Specification Finally, the function would be a subset of the behaviors, the output behavior in particular, and the conditions under which it would be expected to occur. In this case, say, states WR1 and WR3, corresponding to two particular angles in the turning of the crank, might be the function, something like that shown in figure 10.

A function, so the idea goes, is really just a transformation from one state of affairs to another, under some conditions, and as realized by some behavior. The function above is stated in terms of an angle, and specifically it states that it causes something (the crankshaft, in particular) to turn. It might alternatively be stated as transforming the kinetic energy realized in the linear motion of the piston into kinetic energy realized in the rotational motion of the crankshaft.

Functional specification is where the above mentioned heuristic breaks down. A function, in SBF, is an abstraction of some aspect of behavior. That is to say, it is a characterization that is *true of* the behavior of the device. This will require a statement in first-order logic, because it will necessarily employ quantified variables.

A function can be stated as producing some state of affairs from some other state of affairs. Each of these states of affairs are descriptions of some state, as described above. Thus, they are collections of terms that characterize a state, which is the state of a component or connection, specified in terms of the values of parameters. Thus, a functional description will necessarily come with a minimal structural description, otherwise the parameters mentioned in the states and transitions will not have any meaning.

For instance, I might say that the function is, given $\mathrm{val}(\theta_{CW}, s_i) = 0$, the

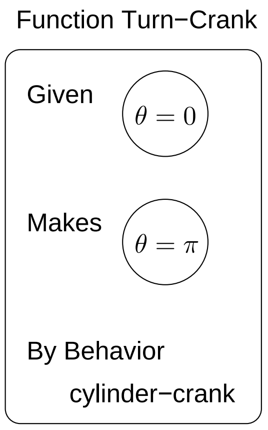

Figure 10: A diagram of the functional model of the crank and piston model, referring to two states from one of the behaviors from figure 9. The two states are the first and third states from the cylinder-crank behavior. It may be desirable to add a condition to this that this transition (from the "given" to the "makes" state) happens due to the motion of the piston, but it's not clear if that belongs in the function or is merely an artifact of the behavior and hence of the design.

devices produces the condition val$(\theta_{CW}, s_j) = \pi$, where θ_{CW} is the angle of a joint (the crank-crank-case connection mentioned above), and s_i and s_j are states of this joint, and that all this is implemented by (meaning true of) a certain behavior.

Visual Representation The remaining question is how a drawing is associated with an SBF model and vice versa. The outline above showed that components and connections can be associated with one or more shapes, by which I mean specific shape frames or instances from some specific drawing. In addition, specific terms in the visual representation (e.g. describing the width of some square) would be associated with particular parameters of particular components and connections.

With regard to our example outlined above, broadly speaking, there are six pieces to the drawing: (1 and 2) the cylinder, represented as two parallel pieces on either side of the piston; (3) the piston, (4) the connecting rod, (5) the crank; and (6) the hinge on which the crank turns (the link to the crank-case). The actual drawing has two shapes associated with the piston, three with the connecting rod, one with the crank, and one with its joint with the crank case. thus, there are nine shapes in the drawing: three rectangles, a triangle, a line, and four circles.

Although I've discussed some ontological issues in §3, above, I have not covered the actual representation; it's very likely, at this point, that whatever

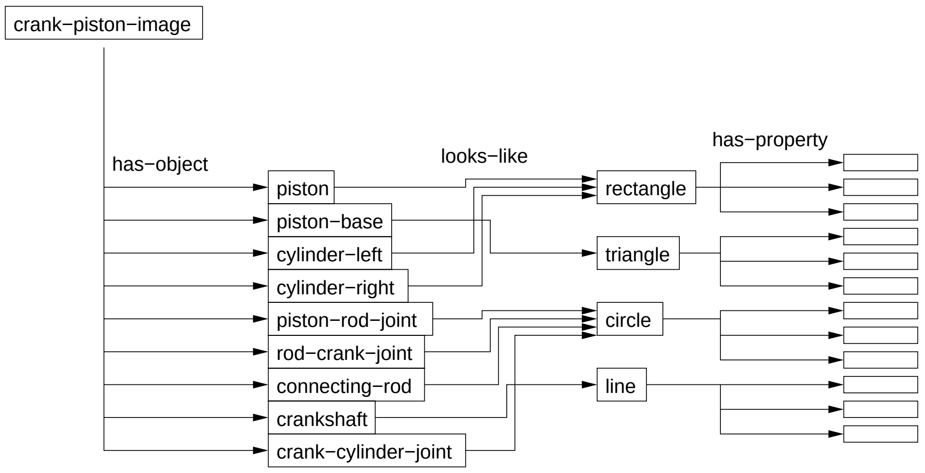

Figure 11: An outline of part of the potential representation of the drawing of the piston and crank from figure 6. This has been simplified somewhat, with the cylinder assumed to be two rectangles, and the crank-cylinder joint to be a circle. The other pieces of the representation would be the terms defining of being a rectangle or what have you (shown, abstractly, on the right side), as well as particular slot values (not shown) for things like size, height, width, diameter, connections between objects, sets (such as the pair of rectangles representing the cylinder), and also spatial relations (above-below, left-of-right-of, and so on)

I say here will end up changing, potentially quite radically, so I'm just going to suggest the kind of "look and feel" it will probably have rather than going over it in detail. With that in mind, I use Covlan [12], developed by Davies and Goel, as a starting point, even though the language is not quite detailed or complete enough for my purposes. If one follows its example, then, the representation would look, in outline, like figure 11. This diagram would also need to include detailed dimensions and aspects of each specific shape, and relationships between them. These are not shown, but it's not hard to imagine what they might look like; they would take the form of additional relations to values of slots and links between objects in the representation.

Since the language has not been worked out yet, I do not wish to show more detail than I have actually worked out.[5]

As I develop the system, the specifics of the languages and the representations are almost certainly going to change, since my understanding of the precise roles of each term and relation are naturally going to develop. Thus, where I have indicated the terms in which something may be represented, they are only potential schemas, and the exact representations remain to be developed. This is especially true for the visual and spatial representation.

[5]It was Niels Bohr who quipped "never express yourself more precisely than you think."

Method Outline

Expanding on the outline given in §2.1 using the example above, one would start with a drawing that looks something like the one in figure 7, let us say. Now, hopefully, it will match with the shapes from the drawings associated with the device illustrated in figure 6, so let's say it does. Most likely it will have two (coherent, useful, relatively complete) mappings: one mapping the single piston assembly from figure 6 onto the left piston assembly from figure 7, and the other mapping it onto the right piston assembly. One would like the system to be capable of using these two mappings to suggest a decomposition of the model and a duplication of some of the parts, and a rebuilding of the model, so to speak, with the added components and connections in it.

Decomposing into tasks and methods, in more detailed outline form, the process would look something like this:

1. **Task** : Given a drawing of a device, produce a possible SBF model of the device as an interpretation of that drawing

   - **Method** : Construct the SBF model by analogy (to drawing/SBF model associations in memory)

     (a) **Task** : Given an input (target) drawing represented as shapes, properties of shapes, and spatial relations between shapes, select similar drawings from memory based on similar shapes, shape properties, and similar relational structure amongst the shapes

       - **Method** : Match shapes, properties, and spatial relations using (partial?) constraint satisfaction

         i. **Task** : Match target shapes to source shapes from memory if the properties and spatial relations also match.

     (b) **Task** : Given the input drawing and a list of potential matches from memory (drawings), determine which drawing from memory best matches with the input drawing.

     (c) **Task** : Given an input drawing and a partially-matching source drawing, adapt the SBF model associated with the source drawing to the input drawing.

       - **Method** : Transfer and adapt matching portion of source model, provided the source matches completely with the target (that is, all the target shapes, properties, and relations are accounted for)
       - **Method** : Transfer and adapt matching portion of source model, and use source to hypothesize roles for remaining portions of target
       - **Method** : Use multiple matchings with the source or matchings with other sources from memory to construct a model out of pieces of other models

     (d) **Task** : Evaluate the new SBF model

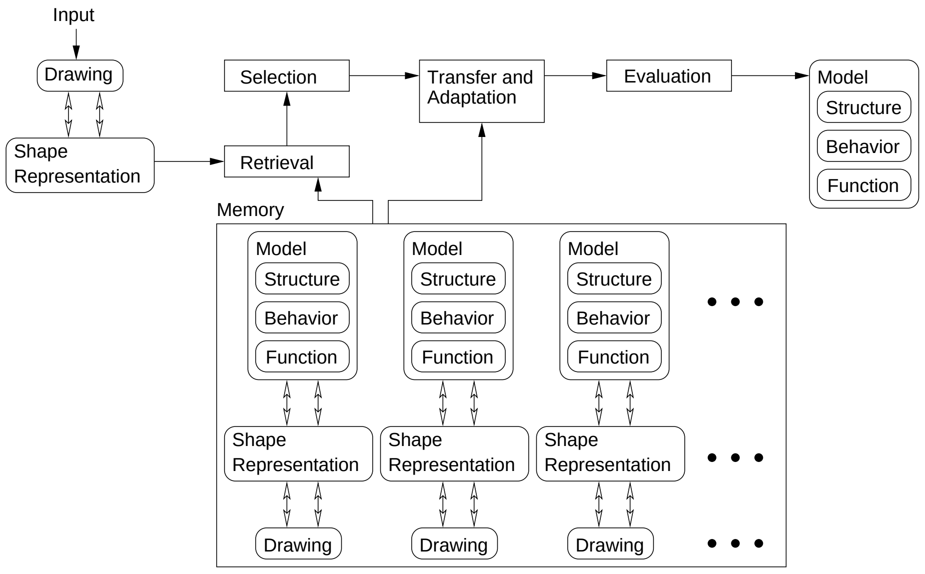

Figure 12: The proposed system architecture.

- **Method** : Determine if the behavior is derivable from the structure, and check the consistency of the structure with the drawing and the behavior with the function.

(e) **Task** : Store the input drawing and new SBF model in memory.

The last step is likely to resemble, at least in broad outline, an envisionment of the sort one sees in qualitative physics [15, 29]. However, instead of generating a complete envisionment of the state space according to the constraints imposed by the structural model (which is approximately what qualitative physical reasoners generally do), it would simply be a verification, tracing through the actual behavior and determining its consistency with regards to the structural model.

From the above outline, we get an architecture with the following: a memory of models and associated visual representations and drawings, and a method for transferring and adapting these models to a complete model of a new device. All this is illustrated in figure 12.

On the specific example from above, it might go as follows:

1. Given the input drawing, D, produce a relational description, R, of the drawing.

2. Retrieval: retrieve all potential models on the basis of their associated drawings by matching the structure of the relational description of the drawings, and returning all that are “sufficiently similar” to the target (this measure will inevitably need to be determined empirically).

3. Of all those returned, find the best one R' amongst the sources—that

is, the source drawing whose relational description best matches with the target, along with all mappings from source to target.

4. This description R' should be of some drawing D' which is associated with some model $M' = (S', B', F')$, where S' is the structural model, B' is the behavioral model, and F' is the functional specification. Using the mappings from R' to R, attempt to construct a hypothetical set of components and connections for the new design.
   (a) The piston in the old model maps to two separate parts of the drawing (under two mappings), and so hypothesize two pistons. Likewise for the connecting rod and the shapes representing the connection between them.
   (b) The connection between the connecting rod and the crankshaft maps to two places in the target drawing, one corresponding to one of the two new hypothesized connecting rods, and one corresponding to the other. So, hypothesize a connection between each one and the one crankshaft.
   (c) The crankshaft and the connection between the crankshaft and cylinder only map once to the target drawing, so we hypothesize one crankshaft and one connection between it and some (yet to be transferred) cylinder in the new model.
   (d) The piston and crankshaft are both connected to one cylinder, though it has two representations in the drawing, something like this: $F_1' \mapsto C'$, $F_2' \mapsto C$, and $F_1' \mapsto F_1$ and $F_2' \mapsto F_2$, thus, hypothesize a C such that $F_1 \mapsto C$ and $F_2 \mapsto C$, where the F_i are figures (shapes or assemblages thereof) in the target drawing, the F_i' are figures in the source drawing corresponding to the cylinder, and C' is the actual cylinder component in in the structural model S'.
   (e) By a similar process, hypothesize a single cylinder corresponding to the other piston assembly, but observe that the same crankshaft is in both, and this one crankshaft is connected to one cylinder, thus, they must be the same cylinder.
   (f) This gives us the new structural model S.

5. Using a similar process as above, hypothesize a new behavioral model, B, corresponding to the new structural model, and a new functional specification F (the former should make the latter straightforward).

6. As part of this, it may be that not all the associations between dimensions and aspects and parameters and quantities in the structural and behavioral model have been set up correctly in the new model, so evaluate and correct any of these as necessary.

7. Evaluate the overall model for internal consistency and general plausibility (do we believe this is a plausible design?). That is, determine if the

behavior is consistent with the structure of the device, and if the function is consistent with the behavior.

8. Store the new model in memory for future usage, and return it.

The other process outlined in §2.2, above, is analogical design generation. This is somewhat closer to what one might think of as "design" captured in a process: some outside context provides us with a function that needs to be achieved by some device, and so we design a device that achieves that function, possibly by adapting a previous design which seems relevant. This is more or less what the IDeAL system did [5], the difference being that now we're making a drawing to go with the model. With respect to the present example, the desired function might be an engine that produces a certain amount of torque, where the old engine, the single piston engine, produces much less torque.

Decomposing into tasks and methods, the outline of that process would look something like this:

1. **Task** : Given a functional spec, produce a SBF model of a device that performs that function, as well as a drawing.
   - **Method** : Construct drawing and SBF model by analogy to existing drawing/SBF model pairs in memory.
     - (a) **Task** : Given an input (target) function, select similar functions from memory, based on functional differences.
     - (b) **Task** : Determine which function from memory best matches with the target.
     - (c) **Task** : Given an input function and a partially-matching source function, adapt the SBF model associated with the source drawing to the input function.
       - **Method** : Transfer and adapt matching portion of source model, provided the source matches completely with the target
       - **Method** : Transfer and adapt matching portion of source model, and use source to hypothesize roles for remaining portions of target
       - **Method** : Use multiple matchings with the source or matchings with other sources from memory to construct a model out of pieces of other models
     - (d) **Task** : Evaluate the new SBF model
       - **Method** : Determine if the behavior is derivable from the structure, and check the consistency of the structure with the drawing and the behavior with the function.
     - (e) **Task** : Store the input drawing and new SBF model in memory.

More specifically, on the example from above:

1. Given the input functional spec, F, retrieve all models from memory with similar functional specs, and return the model $M' = (S', B', F')$ with the fewest functional differences, along with those differences, F_D.
2. If F_D is empty, then return M'. Otherwise, diagnose the functional differences by tracing through the behavioral model B' to determine what structural features in S' are ultimately responsible for F_D. Call this set of model parameters P'.
3. Given P', F_D, and F and F' and M', try each known schema to see if the preconditions for that schema are met. A simple schema may be to see if there is only one parameter in P', and if so, to simply alter that parameter. More complex schemas may involve cascading or feedback.
   (a) In this case, there are several properties which may influence the torque. For instance, the strength of the external stimulus which moves the piston, the diameter of the crankshaft, the diameter of the piston (and cylinder), and so on.
   (b) If one of our schemas is to duplicate that which provides the initial momentum (increasing the number of props on a propeller, increasing the number of rotors in a rotary engine, increasing the number of turbine sections in a gas turbine, etc.)—this being the piston in this case—then that schema, if it's preconditions were met, might attempt to add another piston and crankshaft, which would need to be connected to the original crankshaft in some way.
   (c) Some model parameters (e.g. the angle between the pistons) would not be able to be determined by this schema unless another process could produce a drawing D of the proposed design.
   (d) The output, then, would be a proposed structure S, a proposed behavior B, and of course the functional spec F. along with a drawing D of the device.
4. Evaluate the overall model for internal consistency and general plausibility, store a copy in memory for future usage, and return the new model.

The remaining question is where these convenient schemas come from, and the short answer is they would have to be learned using a process that begins with design failures.

The key difference between these two approaches outlined above is that the first one must simply be capable of *analyzing* and comparing drawings, and the second must be capable of *generating* a design. For instance, it's one thing to say that another piston must be added and connected to the crankshaft, but it's another to work out the geometry of where it's connected, how, and at what angle, because these geometric aspects will impact the behavioral model: they are parameters of connections and dimensions and aspects of components, and thus the parameters will become the changing quantities in the behavior and the dimensions and aspects will become parameters and influences in those behaviors.

More Examples

Some other potential design problems along similar lines:

Door Latch Design Starting with a door latch which you push down on the handle, rotating it to pull back the latch, how do you change it to turn both ways? Alternatives might involve turning a push-in type door latch design (you push in the handle) into one involving a knob that you turn. More complex designs might attempt to integrate an actual lock mechanism, and surround the design of the lock with respect to the latch, for instance.

Cam Follower Cam followers are quite subtle, since every aspect of the shape of the cam lobe relates to function. However, we might find some simple examples, for instance going from one cam pushing one rod of some sort (say, a valve) to a desmodromic valve design (two cams, one raising and one lowering the valve, exactly in sync with one another), or else going from a slider cam to a roller cam. If it must involve altering the shape of the cam lobe, I'm not sure how it would have to work. Other possibilities might involve chaining together, as for example going from one cam operating one valve to two cams operating two valves at some particular timing with respect to each other.

Automobile Suspension Systems A very simple and primitive suspension system for a car might have a single solid axle connecting the two wheels, with two spring/shock units (called "struts") mounting it to the chassis. This has the problem of allowing not just vertical movement, which is what it's supposed to allow, but horizontal movement, as well. If you add a control arm going across the axle, hinged on the chassis at one end and the axle at the other, then this movement is prevented. More complex problems could be made, getting essentially arbitrarily complex, as automobile suspension systems can be quite complicated.

Automatic Transmissions If we abstract away from the shapes of gear teeth and take gears to be simply wheels that are touching, then an interesting question with regard to automatic transmissions is how to get some small number of planetary gears to spin in various combinations to make the right output gear ratios, and all the speeds you need, and how to pack them all into a gear box. For instance, one question might be how to turn a 3-speed transmission (third gear is 1:1) into a 4-speed transmission (adding an overdrive gear as fourth gear).

4.2 Differences in Visual Structure

A key to the method of analogical model construction from drawings, the first of the two outlines above, is knowledge of visual differences—that is, enumerating the useful and interesting differences between two drawings (specifically, the target and the selected source). Having approximately the same shapes in approximately the same arrangement must be distinguished from having the same

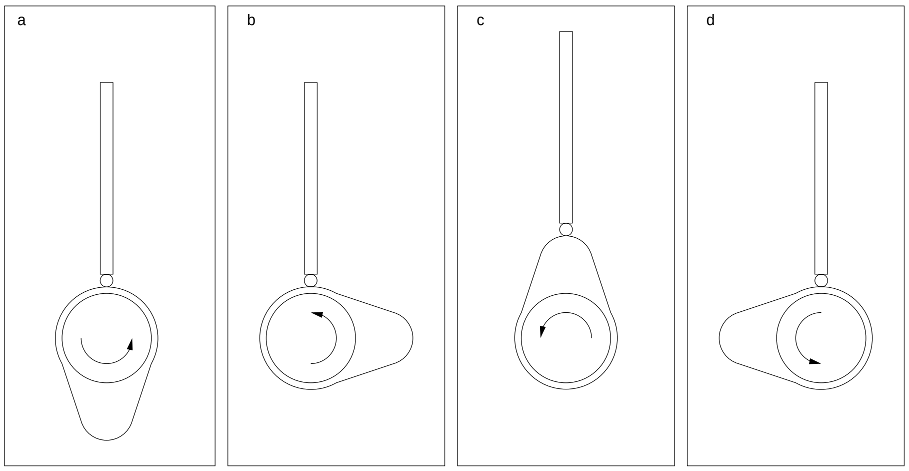

Figure 13: Four juxtaposed drawings of a cam follower, tracking it in four states.

shapes in a different arrangement, which itself is different from the addition or removal of one or more shapes. In the piston engine example discussed, several new shapes are added.

Going from figure 6 to figure 7, differences in visual structure entail differences in the structure of the model. Likewise, in figure 13, the four drawings shown pertain to differences in the various states of the device, that being a cam follower, and do not entail any structural difference at all.

In figure 13, the same shapes occur in the same arrangement, except for the orientation of one of them. However, it's not even true that differences between *what* shapes are present, and not just their orientations, necessarily entail structural differences; e.g. there is more than one way to draw a diagram of the same device. For instance, figure 14 shows exactly the same device as figure 6, but with considerably more visual detail. However, the same components are present (other than the addition of a cylinder head with two valves in it and a spark plug). The visual differences are quite substantial, to say the very least.

Within the framework provided by a representational theory such as the one I have begun here (in section §3), a theory of alignable differences can be developed by starting from the representation of drawings and proceeding to enumerate differences or transformations from one to the other. Now it is unlikely that any theory of visual differences is going to completely account for all these variations. In the extreme, for instance, it's always possible to vary both visual and causal structure, and sometimes not in the extreme: slight visual differences can mean substantial differences in causal structure. However, some theory must be developed for this work to succeed. It will likely rely heavily on abstraction, and it should be emphasized that the task of recognizing shapes is not a part of this work, and is likely going to be beyond what can be done in the space of a single dissertation.

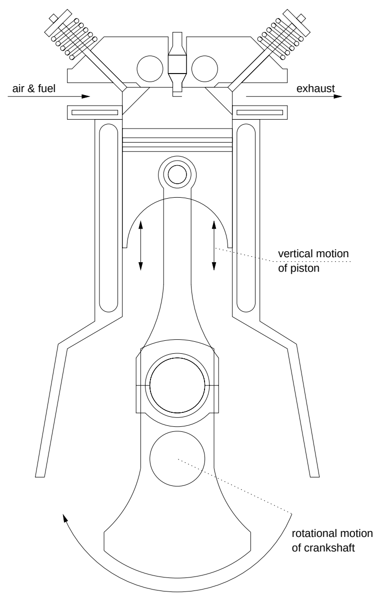

Figure 14: A drawing of a piston engine, shown in cross section, with somewhat more detail than figure 6. The textual annotations are to aid understanding of the drawing.

4.3 Patterns of Adaptation

In the first problem, that of generating a model of a drawn device by analogy, the core of the analogical reasoning process is *adaptation and transfer*. In general, there are different levels at which knowledge may be transferred from one problem to another. At the very bottom, level zero, if you will, the problems are identical, in which case it is a problem of recall and matching. Moving beyond that, however, you get something like the following:

1. Problems differ in parameter values, but not in any structural way, in which case the altering of parameter values and the propagation of constraints is all there is to adaptation and transfer. Nevertheless, if not properly bounded, this can still turn into a hard problem.

2. When problems are structurally different, but not substantially different in size, then the solution may require a rearrangement of the basic components involved, and that being done, a selectio of parameter values and a propagation of constraints, as before.

3. When problems are structurally different but the new one is larger in some sense then the old, then duplicating or adding new components may be necessary.

4. The new problem may involve similar structure only at a certain level of

abstraction, but have different components in an otherwise similar configuration.

5. The new problem may involve similar underlying principles, but involve different components in a different configuration.

At the first level, cases are quite similar to each other, and adaptation knowledge can be captured in terms of the limits on parameters and teh means of propagating constraints. At the second level, the search task for a reconfiguration may be hard, but is bounded in principle by the finite size of the problem (there are a finite number of components, and so there must be a finite number of ways of configuring them)—beyond this reconfiguration, the problem becomes one of altering parameters and propagating constraints, as before.

Each of the last three levels—adding new components, abstracting away structure, and abstracting away principles—require progressively more knowledge and mechanisms for focussing the reasoning. Beyond similarity of principles, it's not clear that any kind of transfer can take place at all.

The first problem, design modelling from drawings, involves a pattern of reasoning that goes from drawings through teleological models to function. The patterns of adaptation I am calling *Generic Visual Teleological Mechanisms* (GVTMs). For the second problem, design generation from functional specifications, involves a pattern of reasoning going from function through models to drawings, just the reverse of before. The patterns of adaptation I am calling *Generic Teleological Drawing Mechanisms* (GTDMs). Each of these generic mechanisms proposes to solve the adaptatio problem by applying a previously learned abstract pattern of adaptation. I discuss each of these in turn.

Generic Visual Teleological Mechanisms

A Generic Visual Teleological Mechanism takes two kinds of conditions, (1) conditions on the source and target drawings, and (2) conditions on the source model, and associates these conditions with a transformation that draws on the source model and matches it to the target drawing. This is illustrated in figure 15

The pattern, here, is that the piston in the source case serves as an energy souce that moves the crankshaft, and in the target the shapes associated with it are duplicated on the opposite side, so that the new model will involve the duplication of this energy source, the piston. The requisite adaptation knowledge will come into play by hypothesizing that while there must be *two* cylinders, there is still only *one* crank case, something which cannot otherwise be inferred.

Generic Teleological Drawing Mechanisms

A Generic Teleological Drawing Mechanism also takes two kinds of conditions—conditions on the source and target functional specifications this time, and conditions on the source model and drawing—and associates these conditions with a transformation that draws on the source model and drawing and matches it to the target functional specification. This is illustrated in figure 16

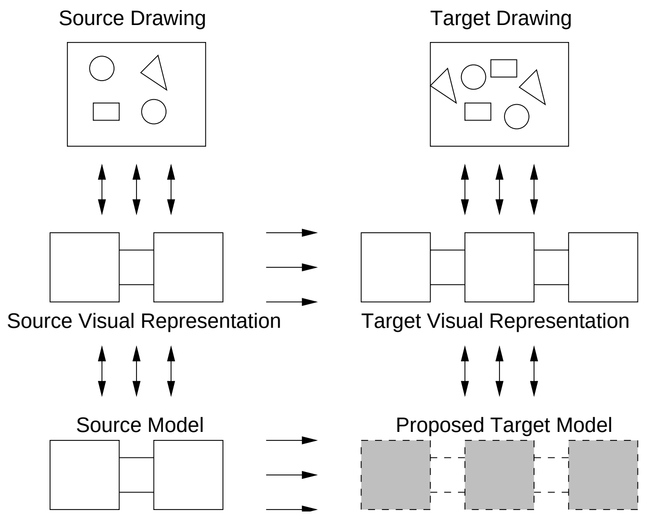

Figure 15: A Generic Visual Teleological Mechanism, associated with the problem of constructing a model to the two-piston engine in figure 7 by analogy to the drawing in figure 6 and the associated model discussed in §4.1. An analogy at the level of visual representation informs a potential new model, where the pattern is that one energy source (the piston) feeds into something that uses the energy (the crankshaft) in the source, and is duplicated so that two energy sources (pistons) are used instead.

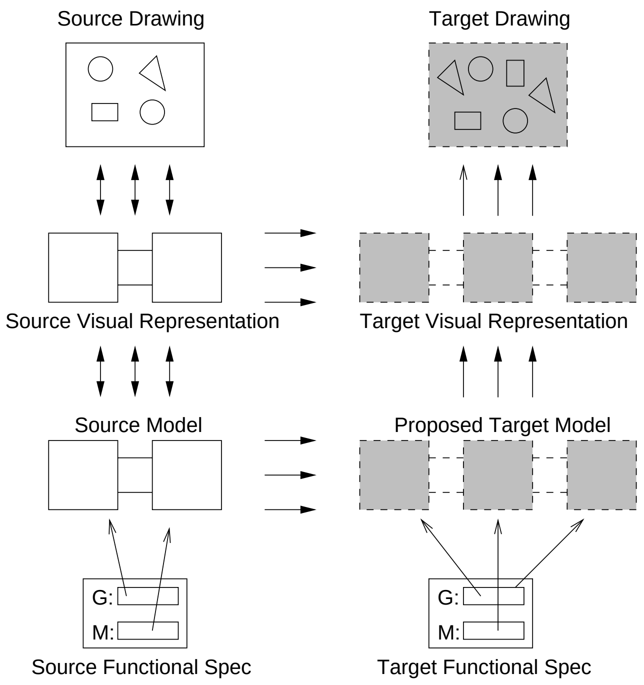

Figure 16: A Generic Teleological Drawing Mechanism, analogous to the GVTM shown in figure 15. This time, the targe functional specification would be that two energy sources (pistons) are feeding into one (crankshaft), and the pattern would attempt to construct a model and drawing consistent with that functional specification by analogy to a device with only one energy source (piston).

The GVTMs go from drawing through model to function. The GTDMs, on the other ha,d, go from function through model to drawing. Beyond that, they are not simply applying differences at one level to a transformation at another, they are constructing drawings, and are thus considerably more complex. Hence, they are different from GVTMs, and substantially different from the old GTMs (Generic Teleological Mechanisms) of IDeAL [5].

5 Testing and Evaluation

The deliverable I am proposing as a part of this work is as follows:

Archytas System To develop a (sub)system capable of (1) constructing a potential SBF model of a drawn device based on the shape representation and analogy to existing SBF models, and (2) producing a potential SBF model and drawing based on a the functional specification.

The building and testing of the above system and the implementation of the visual representation language are the method I am proposing for evaluating the hypotheses of this work. The hypotheses are presented in detail below, but they were described in outline in §2.3. All of the hypotheses involve methods and representations, and so demonstrating them will require evidence to support the claim that this system implements the methods I say it does, and evidence that these methods do indeed work, and evidence that the representations do indeed play the proper roles. In providing that evidence, variation and analysis of these programs, and thus the underlying theory, will occur along these three dimensions:

Context I will test the system on a variety of different design cases in a variety of different domains. Some of the proposed domains and cases were outlined in section 4.1 (page 36).

Models For a given design, whenever possible I will attempt to test a variety of different possible SBF models of that design, and different drawings of the devices, when practical.

Reasoning For each context and model, I will further test a variety of reasoning strategies, by alternatively adding and removing steps in the methods I've described above.

Variation of examples and means of expressing those examples is the general strategy, the point being to ensure the robustness of the methods and representations, and to determine the extent of their expressibility and usefulness.

5.1 Variation of Context

The outline above, in section 4.1 under "More Examples", indicates some of the other problems on which I can run these systems. By encoding and running some set of examples from those four contexts (say, 10 or 12 specific examples), I expect to show that the representation and the algorithms can generalize enough to be interesting. In particular, the contexts described—door latch design, cam followers, automobile suspension systems, and automatic transmissions—should be different enough from each other and from the described example so as to provide a good test of the generality of the models and the methods.

5.2 Variation of Models

The visual representation language I'm proposing to develop and the SBF language that I intend to use will almost certainly not be such that a given device can only be represented in one particular way. As the programmer, I may feel that there is one intended way that it ought to be used, or one particular means of expression that determines the success of that test case, rather than what I am claiming are the relevant factors. To avoid that problem, I propose to vary the representation of the models themselves insofar as a particular device can reasonably be represented in more than one way, so as to explore the impact of different choices of representation have on the process.

The most obvious way this can be afforded is variation of the drawings themselves. Figure 6 is not the only way that a piston and crankshaft can be drawn. It may be that quite subtle differences in the particular selection of shapes, such as the exact way in which the various joints are drawn (as circles, a circle on top of a triangle, or what have you) would make profound differences in the success of a problem solving episode or a test case. This possibility needs to be explored.

Other possibilities have to do with the qualitative behavioral state space. For the piston and crankshaft example, I've discretized it, above, as four states in a loop (the fifth state for each behavior in figure 9 is a repeat of the first state to indicate that loop). Does it have to be four states for each behavior? Where are the essential interactions and which transitions have to be there? Perhaps the choice of values at each state can be varied. Differences here might suggest different choices for the dimensions and aspects and parameters at the structure level, or perhaps even what we claim is the function of the device. Not to mention, I've used landmark values as my states, but perhaps it would be wiser to adopt the qualitative physics habit [15, 29] of using key inequalities to determine states. If so, perhaps there is more than one way to choose the parameters and thus the relevant inequalities. Again, this possibility should be explored.

5.3 Variation of Reasoning

By careful consideration of different reasoning strategies for these crucial steps, I intend to see what effect a variety of different strategies has on the success or failure of the reasoning tasks.

More generally, it is possible to perform a kind of ablation study, whereby different pieces of each reasoner are variously removed and "stubbed out" or simply left out in order to better understand their respective roles and whether they are really necessary. This should guard against the development of redundant steps in the algorithm, or an overly complex theory, and should also clarify the roles and generally inform the generalizations of this work to broader contexts.

6 Discussion

6.1 Related Work

The two biggest and most direct influences on this work are analogical and case-based reasoning [57, 37], one the one hand, and visual and diagrammatic reasoning [43, 50, 7] on the other. I'll begin with the latter.

Visual and Diagrammatic Reasoning

Diagrammatic reasoning is a burgeoning field at the intersection of several other fields, most notably the traditional cognitive science influences of psychology and computer science, but also receives some influences from architecture and other design-related disciplines. The literature in this field is vast, and beyond that it's difficult field to circumscribe, because it is rather new and very multidisciplinary, with papers appearing in scattered conferences and journals all over the various cognitive sciences, and often in the design literature [39, 41, 40], though there have been some attempts to unify the field [50, 7]. Here I will only spotlight a few notable examples of work in the area.

Much of the work in this field has been psychological, rather than from an AI perspective. Some of this is interesting in inspiring cognitive models, but my work is not attempting to do cognitive modelling. Nevertheless, there is plenty of relevant work on computational models and approaches. Broadly speaking, the idea behind the focus on diagrammatic reasoning is to use the diagram itself as a kind of external "analogical" representation, in contrast to so-called "Fregean" or logical representations [69]. The attempt, then, is to find ways of solving interesting problems by representing them externally in a diagram using location, shape, relative position, and so on, rather than in some more conventional symbolic language. In practice, of course, a computer program can only work with symbolic representations, and so in fact it is representing a diagram symbolically, and abstracting the relevant information from the diagram and translating into a more conventional symbolic representation. This is not to say that all that philosophizing about non-logical representations is just a bunch of baloney; rather, this amounts to the observation that in practice, so-called diagrammatic reasoning is really employing another *mode* of representation in problem solving, so that it might just as well be called *multi-modal* representation and problem solving.

On that note, there are some interesting bodies of work in exactly this direction. One of the first and most famous is that of Larkin and Simon [59], who created a system could reason about pulley systems, among other things. In this system, to answer a question about a pulley system, the reasoner uses a bit of non-visual physics knowledge along with knowledge of location, and all information associated with locations, rather than a specific object-based language like you might see in a qualitative physical reasoner.

An early system was Funt's WHISPER [34], which had a simple retina model with higher resolution towards the center, and solved blocks world problems in a visual manner.

Forbus [30] developed the FROB system [28]. FROB operated in "bouncing ball world", predicting where a ball (point mass) would bounce off of straight line segments using a combination of qualitative reasoning and information provided in what was then called a "metric diagram". Later, in the CLOCK project [32, 33], which reasoned about, for instance, a ratchet mechanism in a clock (hence the name of the system), this became the *metric diagram/place vocabulary* (MD/PV) model, where a qualitative spatial (or kinematic) *place vocabulary* is augmented with a metric diagram, providing specific information backing up the fairly abstract representation expressed in the place vocabulary. The point is that, whereas the place vocabulary, expressed in terms of shape and connection, may assert that a robot can fit through a certain doorway, if something happens to be on the robots head then the underlying assumptions have been violated, and so only the metric information in a diagram can answer the question correctly, ultimately. A better example is that a roller with a notch and a roller with a bump either may or may not roll together smoothly, and a qualitative representation will, in general, struggle with this, but a metric diagram can answer the question quickly and precisely. This is in the same spirit as Larkin and Simon's work, in many ways, and is an early version of the claim that has come to be the principle one of diagrammatic reasoning in general, as discussed above.

Within this project, Faltings's work [22] focussed on the generating of place vocabularies from (1) metric diagram information and (2) configuration spaces, which represent constraints on possible motions of parts. Nielsen's work [66] focussed on the qualitative analysis of the constrained motions of the parts, represented in this place vocabulary, and the generation of something like an envisionment of the sort qualitative physical reasoners perform. This pattern of reasoning is, in some ways, the reverse of what my work studies: the CLOCK system reasons bottom-up from a metric diagram and constraints on possible motions to analyze the overall behavior of the system (although, this difference may simply reflect the contrast between deductive and analogical reasoning at work). From appearances, the CLOCK system was limited in its scope to relatively simple kinematic pairs of the sort demonstrated by the ratchet mechanism.

Faltings, later, continued in this thread, taking it into the design domain and employing case-based reasoning with a system called FAMING [23]. FAMING is a case-based reasoning system that uses cases describing physical mechanism parts (such as the one from the CLOCK work, above). FAMING uses what it calls structure-behavior-function models to describe the cases, though from indications they are not entirely identical to what I'm employing, here. It takes the metric diagram and place vocabulary to be the structure, as well as the configuration space (constraints on possible motions, expressed in a compact way). Human designers choose cases and functions that should be used, dimensions to modify, and shape features to be unified, and the system uses qualitative kinematics to propose design solutions for the desired function. It's not described as a visual system, but in fact it is (the metric diagram and place vocabulary, of course). It's scope, from appearances, is limited in much the way the CLOCK system was, and it relies on user interaction to get over some of the hard problems this work is attempting to confront directly. At its core, it modifies cases according to shape substitution, which is an interesting but limited methodology for doing design problem solving.

Narayanan, Suwa, and Motoda [63, 64] have developed a model of a cognitive process for predicting the behavior of physical systems (pressure gauges, scales, etc.) from cross-sectional diagrams of these devices. The model was developed from protocol studies of human subjects performing the same task. The system brings together two representations: diagram frames, representing line segments, enclosed spaces, boundaries, and so on, and conceptual frames, representing components of some device. Diagram frames are organized in an array representation, and thus can be retrieved by location. This also allows conceptual frames to be linked to these same locations, and thus allowing reasoning to move back and forth naturally between the conceptual and the diagrammatic. The reasoning itself is a fairly basic forward chaining strategy, somewhat lacking in control knowledge (as the authors confess).

Sketch-based recognition work has become rather popular with the rise of pen-based interfaces [14]. Some interesting work has come out of this effort, most of it focused on low-level bottom-up processing such as recognition of freehand strokes and basic visual symbols. Some of the work is model-based, such as with Randall Davis [13, 1], but the underlying models used to interpret the sketches are generally quite simple, and this is intentional, since the bulk of the work has surrounded issues of ambiguity with regard to users' intention, such whether a sketch is supposed to convey certain geometric constraints or not as essential to the intended interpretation. This has lead to some interesting work on multi-modal interfaces, for instance, involving both sketch-based and vocal input.

The Electronic Cocktail Napkin [46] is an example of another interesting system along similar lines. This time the domain is architectural design, with the system supporting sketching (on overlays of actual architectural diagrams, possibly), recognizing visual symbols, shape recognition, and so on. More recent work has focused on the ability to project sketches of buildings onto a simple grid-based 3-D model [16] to better communicate spatial aspects of designs with this kind of interface.

SKETCHIT [72] can generate multiple new designs from a sketch, augmented with a simple state-transition diagram to describe the device's behavior. The system produces a "behavior-ensuring parametric model" (BEP-Model) of the device, and from this determines geometric constraints on the motion of the parts, generating all *qc-spaces* (or qualitative configuration spaces) consistent with the behavioral constraints. From this, motion types for each component are selected, and geometric interpretations provided by a library of interaction types, and from this a new BEP-model is generated of the new design. The system only handles rigid bodies and springs, and the models only handle geometric constraints on motion, and are not physical models of motion in any sense. However, more recent work [58] uses visual symbol recognition (which must be trained) and behavioral reasoning to identify intended meaning, not just behavioral constraints, and can deal with motors, heaters, pulleys, wheels, and so on. The new system produces a textual gloss describing the behavior or functioning of the sketched device at a high level ("lever rotates", for instance) based on a simple qualitative physical simulation, which is achieved with a forward-chaining rule system. This physical model is very simple (e.g. it does not distinguish direction of rotation, or direction of force) and is only used to disambiguate the roles of parts in functioning of the device.

On a practical side, computer-aided design (CAD) tools—sometimes referred to as *computer-aided drafting and design* (CADD) tools—are traditionally tools geared towards drafting. To that end, the more abstract and domain-specific advances in CAD software has involved object modelling, whereby objects can be given a geometric model or identity, so that they can be moved around, altered, or reused without the undue effort of moving and redrawing dozens or even hundreds of little lines and arcs in an engineering drawing. However, to date there has been no significant attempt to bring causal or functional models to bear on CAD tools and drawings—the knowledge captured by such models, though clearly important, is left up to the designer to manage. Within architecture and civil engineering, *Building Information Modeling* (BIM) [60, 2, 4] is an example of an attempt to begin to integrate a domain expertise into the CAD (and CAAD: computer-aided architectural design) systems.

Visual Analogy

Visual analogy is a subject, not a field, and even as such it is far from well explored. Nevertheless, scouring the literature one can turn up an interesting and diverse selection of work.

One of the most interesting systems in this line of work is Gary McGraw and Douglas Hofstadter's LetterSpirit [62, 52]. It takes a stylized seed letter as input and outputs an entire font that has the same style (or "spirit": imagine a table, each row containing the alphabet in some style; then each column represents a letter, and each row a spirit, hence the name). 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), "motifs" (geometric shapes, such as a parallelogram, used over and over again), and "abstract rules" (e.g. no diagonal lines, only horizontal and vertical), and using these attributes builds an entire alphabet in some new style. Obviously, the domain is pretty restricted, but the indirect analogies and transfer being done (by mapping the visual elements onto abstract concepts such as "roles") is an interesting idea, and it allows them to show how the "concept" of an A, for instance, can be a "fluid concept" rather than precisely delineated in some frame structure or symbolic logic.

Copycat [51, 53] is another system from the same group, this time Melanie Mitchell and Douglas Hofstadter. Though not exactly a *visual* analogy system, it nevertheless is work by the same group in the same vein as LetterSpirit, and so I discuss it here. Copycat is a system that performs analogies of the form "*A* is to *B* as *C* is to what?" In this case, the terms are letter strings. For instance `abc` is to `abd` as `ijk` is to what? Copycat searches stochastically through rules—that is,"codelets" in the "coderack"—to find one that applies to the first pair, and then applies it to the third string, yielding `ikl` in this case. It uses a "slipnet" to find similarities between non-identical relationships (e.g. `abc` is to `abd` as `xyz` is to what? Some prosaic answers are `xyd` and `xyy`, but a more creative one is `wyz`, which is had by reversing the mapping, so `a`$\mapsto$`z` and `c`$\mapsto$`x`). Copycat, which seems at first glance to be utterly trivial in its scope, actually reveals some interesting ideas about analogies and fluid concepts, as the group likes to speak of. It occasionally finds rather subtle analogies like the following: `abc` is to `abd` as `mrrjjj` is to what? A dull answer is `mrrjjk`, a more appropriate one is `mrrkkk`, but every now and then, by mapping the 1-2-3 sequence of `abc` onto the *number* of letters, it gets `mrrjjjj` (three `j`'s become four).

Both of these systems are examples of restricting the domain enough so that one can begin to use computational resources to break down the boundaries of concepts, if you will. They could not be achieved by traditional approaches to representation and inference in intelligent systems—at least, not without "canning" the sorts of answers you want in advance. This makes one wonder how these ideas could be used in more realistic and interesting domains.

A very early analogy system was Evans' ANALOGY program [18], which solved geometric analogies of the A : B :: C : ?? sort (once again) of the kind one might see on intelligence tests (see figure 17). Given a multiple choice question of this sort, ANALOGY attempted to find a procedure for turning A into B, and then turned C into each of the (say) 5 answers, and whichever of those transformation procedures, represented as a conceptual graph, was closest to the original was chosen. It had a simple language of primitives (dots, lines, closed polygons, etc.), relationships (above, left-of, and so on), and transformations (rotate, reflect, expand, contract, add, delete). The system is also described in detail by Winston in his book [77, chapter 2].

PAN [67] is an example of a more recent system that performs the same task using case-based reasoning, and also uses graph-based representations. Its representation and inference mechanisms appear to have more generality and robustness than Evans' early system. Both of these systems, ANALOGY and PAN, operate on meaningless images in a very narrow task. It is more interesting to see what is required of a visual analogy system when the context of the reasoning task becomes broader and more meaningful, and the possible interpretations of the images in question become central.

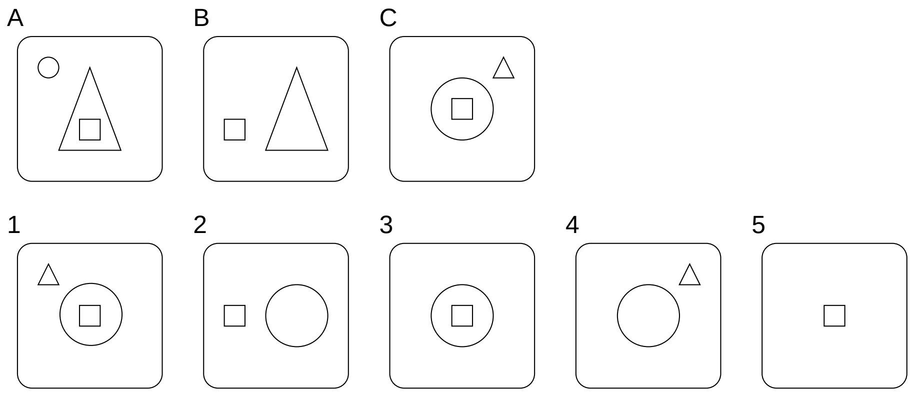

Figure 17: An example of a geometric analogy of the sort seen on intelligence tests that is solved by Evans' ANALOGY program [18], of the form A : B :: C : ??, where the task is to choose one of 1, 2, 3, 4, or 5 for the remaining image.

GeoRep [26], though not an analogy system, was developed to generate relational descriptions of drawings for visual analogy systems, and other diagrammatic reasoning systems. The system passes an input drawing through a two-stage reasoning process: (1) a domain independent low-level relational describer (LLRD) which recognizes lines, arcs, and so on, and looks for proximity, connectivity, parallel lines, and so on; and (2) a domain-specific high-level relational describer (HLRD) which uses the low-level representation as input to a forward chaining rule system and truth-maintenance system to produce a description in some place vocabulary.

MAGI [24] takes visual representations produced by GeoRep and looks for qualitative symmetry. The idea is that some figures, while not strictly symmetric, nevertheless display a central "axis" which can be found by a kind of symmetry in the visual representation. This it finds by running the structure-mapping engine (SME) [21] on the representation, attempting to map it onto itself (with rules prohibiting the identity mapping). the JUXTA system [25] uses MAGI in its processing of so-called "juxtaposition diagrams", where some situation is represented in two diagrams side-by-side, and the alignable differences between them can be used to understand some situation.

The VAMP systems [73] of Thagard et al. are visual analogical mapping systems. VAMP.1 uses an array-based representation scheme, and maps source onto target by comparing symbols in the same location, scaling the arrays to be of the same size if they are different, converting both to the least common multiple of their respective sizes. The VAMP.2 system represents images as agents with local knowledge, which can be relational (e.g a is left of b), with each agent seeking to map itself to one in the target, and using ACME [54] to actually perform the mapping. The authors do not mention retrieval, but it could using ARCS [74]. VAMP.2 can find the correct mapping for the fortress/tumor problem of Gick and Holyoak [42] (that is, Duncker's radiation problem [17]) when it is represented diagrammatically. However, neither VAMP.1 nor VAMP.2 do anything like transfer and adaptation, and so cannot be said to be doing problem solving. At their core, they associate images with one another by parts.

the DIVA system [11] of Croft and Thagard is another analogical mapping system employing visual information. The system can represent scenes in 3-D using simple geometric shapes, and associate them with semantic networks intended to represent the content of these 3-D scenes. The system computes mappings using ACME, and can find the mapping in one version of the fortress/tumor problem. Once again, the system does not do transfer, and so is not actually performing any problem solving.

FABEL [35] was an early project employing case-based reasoning to explore the automated reuse of diagrammatic cases in the domain of architectural design, employing domain-specific heuristics to guide pattern extraction and transfer. FABEL is a large-scale project with many subsystems, designed as an expert system intended to demonstrate the practical usability of its various approaches to all the stages of the process, and it relies heavily on user interaction.

Analogical and Case-Based Reasoning In General

The PRODIGY system [10, 75, 76] implements derivational analogy. In derivational analogy, transfer of problem-solving procedures from a source to a target uses memories of the justifications for each step, allowing for adaptation of the transferred procedure. A *derivation* is a trace, that is, a script of a problem solving procedure with justifications for taking of each particular step over others. Intermediate states generated are not stored, but are created anew each time a new analogy is made.

CHEF [49] is a case-based planning system that adapts cooking recipes to solve new problems. Like PRODIGY, CHEF stores the sequence of operators used to derive the solution, but not intermediate states. Two other case-based reasoning systems are ARCHIE [68] and AskJef [3], which contain multi-modal cases, i.e. cases that contain both visual (e.g. photographs, drawings, diagrams, animations, and videos) and non-visual knowledge (e.g. goals, constraints, plans, and lessons). Nevertheless, the multi-modal cases in these systems typically are indexed only by non-visual constructs such as goals and constraints. The target problem too typically is specified by its goal and constraints, and cases are retrieved based on a match between the goals and constraints with the source cases. Like FABEL they are intended for use by people, and they rely on user interaction for the transfer step.

In other case-based reasoning work, Smyth and Keane [70] have developed a theory of adaptation-guided retrieval (AGR). In this work, the authors call into question the *similarity assumption*, which, according to the authors, holds that similar experiences can guide future reasoning, problem solving, and learning. This, they say, leads to a model where retrieval and adaptation are entirely separate, with neither informing the other overmuch. Conceptual features in cases, they say, cannot be determined to be useful or not useful for adaptation *a priori*; instead they show how adaptation knowledge (such as transformation rules) can be turned into adaptation *capability* knowledge, that is, knowledge of the utility of particular features for particular kinds of adaptations. In related work on the same system, called Déjà Vu, Smyth, Keane, and Cunningham [71] developed a method of employing abstraction hierarchies in the adaptation of multiple cases for some target solution to coordinate the entire multiple-case reuse across different levels of abstraction. Abstract cases help decompose the solution into subproblems, and concrete cases help solve these subproblems.

Holyoak and Thagard [55] developed the Process of Induction (PI) model, which uses a production system to solve problems, using spreading activation by activating concepts associated with rules as a side effect of rule firing, which can happen in parallel. This search for a problem-solving procedure is bi-directional, employing both forward and backward chaining. The activation trace left from the activation of the source procedure guides it in a search for a new solution, which it then uses to generate an abstract schema that works as a single rule to apply to both problems in the future. This is not transfer in the usual sense of the word; instead, this model regards analogy as happening in three steps: retrieval, adaptation, and schema formation.

The Structure-Mapping Engine (SME) [21] was written as an implementation of Gentner's structure-mapping theory [36], although it's been pointed out by Philip Johnson-Laird [56] that, in fact, the operation of the one is the reverse of the other: in Gentner's theory, the object mappings inform the best global mapping, but in SME, it's just the reverse. At any rate, SME searches for mappings between source and target descriptions by regarding their representations as graph representations, and mapping the structures one-to-one onto each other, transferring bits of the source representation onto the target (*candidate inferences*). In SME, then, mapping is the core of analogy and transfer is a relatively minor step. Retrieval is added into the picture with MAC/FAC [31], which performs a relatively standard two-stage retrieval: the first uses feature vectors based on the target and source descriptions, and the second uses a "row of SMEs" to match the results of the first stage to the target. It is important to know that SME and MAC/FAC are not doing anything like *problem solving*; they are merely transferring bits of structure from one description to another, regardless of what that description actually represents. In order to be a problem solving system, SME and MAC/FAC would have to be hooked up to some other problem solving system.

PHINEAS [19] uses SME to make analogies between physical phenomena represented using Qualitative Process Theory (QPT) [29]. It explains new behaviors by analogy to old behaviors, transferring knowledge from source to target. Evaluation takes place by simulation, using past reasoning traces summarized by storing with each state an observation of the collection of theories used to explain it (e.g. with the example of alcohol evaporating from a flask, it may store theories about evaporation and containment). It can hypothesize *skolem objects* to generate alignment with unmapped entities in the source analog, but it still does not do anything like problem solving. This is understanding, in the qualitative physics sense of the word, which means envisionment, limit analysis, and other processes involving the analysis of the state space of some device or physical system in normal operation. It is not generating knowledge so much as completing partial descriptions of physical systems.

The Analogical Constraint Mapping Engine (ACME) of Holyoak and Thagard [54], along with its complementary retrieval model Analogical Retrieval by Constraint Satisfaction (ARCS) [74], is a general analogy model. Holyoak and Thagard proposed that the retrieval and mapping tasks can be productively viewed as constraint satisfaction problems. Their proposal incorporated structural, semantic and pragmatic constraints and used graph isomorphism as the primary similarity measure. ACME and ARCS provided a "localist" connectionist implementation of their proposal. Nodes are constructed for each map hypothesis (between a source element and a target element), with inhibitory and excitatory links between the nodes, and the network is run until it reaches quiescence. A drawback is that all of the map hypotheses must be "wired" into their network, which in even moderate-sized examples can grow to be quite large.

6.2 Expected Contributions

To recall from above, I am proposing the following specific deliverable with regard to this work:

Archytas System To develop a system capable of constructing a proposed SBF model of a drawn device based on the shape representation and analogy to existing SBF models.

The system, however, is not the interesting aspect of this work. The interesting aspects of this work are the methods and representations that the system demonstrates, as those methods and representations are ultimately my answer to the hypotheses outlined in §2.3.

On a general level, then, I am seeking two methods:

- A method for producing a structure-behavior-function model of a drawing of a device, that serves as an interpretation of that drawing, by analogy to known existing structure-behavior-function models that have been drawn.
- A method for producing a structure-behavior-function model and drawing of a device that achieves or performs a specified function by analogy to existing drawn structure-behavior-function models.

This, in turn, will entail methods for retrieval, transfer and adaptation, evaluation, and storage.

In addition, there are the representations. I expect two representational schemes to come out of this work:

- A shape representation useful for representing those shapes, their properties, and spatial relationships that can usefully depict mechanical devices.

- The beginnings of a reworked structure-behavior-function representation based on energy transfer.

It should be emphasized that both of these goals are secondary, and that the methods of analogy are the most important aspect of the work. As such, these both will draw on previous work to the greatest extent possible, rather than reinventing the wheel. On both accounts, there is substantial previous literature.

Finally, the overall goal of this work is to understand the role of drawings and the kinds of information they communicate and the kinds of inferences they support, especially in design where their role appears to be most central and important. Thus, the two basic tasks of design modelling and design generation attack both sides of this problem, and working on them should shed some light on important issues surrounding the interpretation of visual information, and the roles of visual reasoning, and multimodal reasoning more generally, in design.

References

[1] C. Alvarado and R. Davis. Resolving ambiguities to create a natural computer-based sketching environment. In B. Nebel, editor, *Proc. 17th International Conference on Artificial Intelligence (IJCAI-01)*, pages 1365–1371. Morgan Kaufmann Publishers, 2001.

[2] Autodesk. Building informatnion modeling. White Paper, Autodesk, Inc., 2002.

[3] J. Barber, M. Jacobson, L. Penberthy, R. Simpson, S. Bhatta, A. Goel, M. Pearce, M. Shankar, and E. Stroulia. Integrating artificial intelligence and multimedia technologies for interface design advising. *NCR Journal of Research and Development*, 6(1):75–85, 1992.

[4] K. Bentley and B. Workman. Does the building industry really need to start over? White Paper, Bentley Systems, Inc., January 2003.

[5] S. Bhatta. *Model-Based Analogy in Innovative Device Design.* PhD dissertation, College of Computing, Georgia Institute of Technology, Atlanta, Georgia, December 1995.

[6] S. Bhatta, A. Goel, and S. Prabhakar. Innovation in analogical design: A model-based approach. In *Proc. 3rd Int'l Conf. on Artificial Intelligence in Design (AID-94)*, August 1994.

[7] A. Blackwell, K. Marriott, and A. Shimojima, editors. *Diagrammatic Representation and Inference: 3rd. International Conf., Diagrams 2004*, volume 2980 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2004.

[8] D. Bobrow, B. Falkenhainer, A. Farquhar, R. Fikes, K. Forbus, T. Gruber, Y. Iwasaki, and B. Kuipers. A compositional modeling language. In Y. Iwasaki and A. Farquhar, editors, *Qualitative Reasoning: Papers from*

the Tenth Annual Workshop, Technical Report WS-96-01, pages 12–21. AAAI Press, Menlo Park, California, 1996.

[9] T. Bylander and B. Chandrasekaran. Understanding behavior using consolidation. In *Proc. 9th International Joint Conference on Artificial Intelligence*, pages 450–454. Morgan Kaufmann Publishers, 1985.

[10] J. Carbonell and M. Veloso. Integrating derivational analogy into a general problem solving architecture. In *Proc. 1st Workshop on Case-Based Reasoning*, pages 104–124. Morgan Kaufmann, 1988.

[11] D. Croft and P. Thagard. Dynamic imagery: A computational model of motion and visual analogy. In L. Magnani and N. J. Nercessian, editors, *Model-Based Reasoning: Science, Technology, and Values*, pages 259–274. Kluwer Academic Publishers/Plenum Publishers, New York, 2002.

[12] J. Davies and A. K. Goel. Representation issues in visual analogy. In *Proc. 25th Annual Conf. Cognitive Science Society*, Boston, MA, 2003. Lawrence Erlbaum Associates.

[13] R. Davis. Sketch understanding in design: Overview of work at the MIT AI Lab. In Davis et al. [14], pages 24–31.

[14] R. Davis, J. Landay, and T. F. Stahovich, editors. *Sketch Understanding: Papers from the 2002 Spring Symposium*, Technical Report SS-02-08, Menlo Park, CA, 2002. AAAI Press.

[15] J. De Kleer and J. S. Brown. A qualitative physics based on confluences. *Artificial Intelligence*, 24:7–83, 1984.

[16] E. Y.-L. Do. Graphics interpreter of design actions: The GIDA system of diagram sorting and analysis. In B. de Vries, J. P. van Leeuwen, and H. H. Achten, editors, *Computer Aided Architectural Design Futures 2001: Proc. 9th International Conf.*, pages 271–284. Kluwer Academic Publishers, 2001.

[17] K. Duncker. A qualitative (experimental and theoretical) study of productive thinking (solving of comprehensive problems). *Journal of Genetic Psychology*, 1926.

[18] T. G. Evans. A heuristic program to solve geometric analogy problems. In M. Minsky, editor, *Semantic Information Processing*. MIT Press, Cambridge, MA, 1968.

[19] B. Falkenhainer. A unified approach to explanation and theory formation. In J. Shrager and P. Langley, editors, *Computational Models of Scientific Discovery and Theory Formation*, chapter 6, pages 157–196. Morgan Kaufmann, 1990.

[20] B. Falkenhainer, A. Farquhar, D. Bobrow, R. Fikes, K. Forbus, T. Gruber, Y. Iwasaki, and B. Kuipers. CML: A compositional modeling language. Technical Report KSL-94-16, Knowledge Systems Laboratory, Department of Computer Science, Stanford University, Stanford, California, January 1994.

[21] B. Falkenhainer, K. D. Forbus, and D. Gentner. The structure-mapping engine: Algorithm and examples. *Artificial Intelligence*, 41:1–63, 1990.

[22] B. Faltings. Qualitative kinematics in mechanisms. In J. McDermott, editor, *Proc. 10th International Joint Conference on Artificial Intelligence (IJCAI-87)*, pages 436–442. Morgan Kaufmann Publishers, 1987.

[23] B. Faltings. FAMING: Supporting innovative design using adaptation - a description of the approach, implementation, and illustrative example and evaluation. In A. Chakrabarti, editor, *Engineering Design Synthesis: Understanding, Approaches, and Tools*, chapter 17, pages 285–302. Springer, 2002.

[24] R. W. Ferguson. Modeling orientation effects in symmetry detection: The role of visual structure. In L. R. Gleitman and A. K. Josh, editors, *Proc. 22nd Annual Conf. Cognitive Science Society*, Philadelphia, PA, 2000. Lawrence Erlbaum Associates.

[25] R. W. Ferguson and K. 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.

[26] R. W. Ferguson and K. 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.

[27] R. P. Feynman, R. B. Leighton, and M. Sands. *The Feynman Lectures on Physics: Mainly Mechanics, Radiation, and Heat*, volume I of *The Feynman Lectures on Physics*. Addison-Wesley, Reading, MA, 1963.

[28] K. D. Forbus. Spatial and qualitative aspects of reasoning about motion. In *Proc. 1st National Conf. on Artificial Intelligence (AAAI-80)*, pages 170–173. AAAI Press, 1980.

[29] K. D. Forbus. Qualitative process theory. *Artificial Intelligence*, 24:85–168, 1984.

[30] K. D. Forbus. Qualitative spatial reasoning: Framework and frontiers. In Glasgow et al. [43], pages 183–202.

[31] K. D. Forbus, D. Gentner, and K. Law. MAC/FAC: A model of similarity-based retrieval. *Cognitive Science*, 19(2):141–205, 1995.

[32] K. D. Forbus, P. Nielsen, and B. Faltings. Qualitative kinematics: A framework. In J. McDermott, editor, *Proc. 10th International Joint Conference on Artificial Intelligence (IJCAI-87)*, pages 430–435. Morgan Kaufmann Publishers, 1987.

[33] K. D. Forbus, P. Nielsen, and B. Faltings. Qualitative spatial reasoning: The CLOCK project. *Artificial Intelligence*, 51(1–3):417–471, October 1991.

[34] B. V. Funt. Problem-solving with diagrammatic representations. *Artificial Intelligence*, 13(4):201–230, 1980.

[35] F. Gebhardt, A. Voß, W. Gräther, and B. Schmidt-Belz. *Reasoning with Complex Cases*, volume 393 of *Kluwer Series in Engineering and Computer Science*. Kluwer Academic Publishers, Boston, 1997.

[36] D. Gentner. Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, 7(2):155–170, 1983.

[37] D. Gentner, K. J. Holyoak, and B. N. Kokinov, editors. *The Analogical Mind*. The MIT Press, Cambridge, MA, 2001.

[38] J. S. Gero. Design prototypes: A knowledge representation schema for design. *AI Magazine*, 11(4):26–36, Winter 1990.

[39] J. S. Gero and B. Tversky, editors. *Visual and Spatial Reasoning in Design*. Key Center of Design Computing and Cognition, University of Sydney, 1999.

[40] J. S. Gero, B. Tversky, and T. Knight, editors. *Visual and Spatial Reasoning in Design III*. Key Center of Design Computing and Cognition, University of Sydney, 2004.

[41] J. S. Gero, B. Tversky, and T. Purcell, editors. *Visual and Spatial Reasoning in Design II*. Key Center of Design Computing and Cognition, University of Sydney, 2001.

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

[43] J. Glasgow, N. H. Narayanan, and B. Chandrasekaran, editors. *Diagrammatic Reasoning: Cognitive and Computational Perspectives*. AAAI Press/The MIT Press, Menlo Park, CA, 1995.

[44] A. Goel, S. R. Bhatta, and E. Stroulia. Kritik: An early case-based design system. In M. Mahler and P. Pu, editors, *Issues and Applications of Case-Based Reasoning in Design*, pages 87–132. Lawrence Erlbaum Associates, Mahwah, New Jersey, 1997.

[45] A. K. Goel. A model-based approach to case adaptation. In K. J. Hammond and D. Gentner, editors, *Proc. 13th Annual Conf. of the Cognitive Science Society*, pages 143–148. Lawrence Erlbaum Associates, August 1991.

[46] M. D. Gross. The electronic cocktail napkin. *Design Studies*, 17(1):53–69, 1996.

[47] J. Y. Halpern and J. Pearl. Causes and explanations: A structural-model approach—part I: Causes. In *Proc. 17th Conf. on Uncertainty in Artificial Intelligence (UAI-01)*, pages 194–202. Morgan Kaufmann, 2001.

[48] J. Y. Halpern and J. Pearl. Causes and explanations: A structural-model approach—part II: Explanations. In *Proc. 17th International Joint Conf. on Artificial Intelligence (IJCAI-01)*, pages 27–34. Morgan Kaufmann, 2001.

[49] K. J. Hammond. CHEF: A model of case-based planning. In *Proc. 5th National Conf. on Artificial Intelligence (AAAI-86)*, Philadelphia, PA, 1986. AAAI Press.

[50] M. Hegarty, B. Meyer, and N. H. Narayanan, editors. *Diagrammatic Representation and Inference: 2nd International Conf., Diagrams 2002*, volume 2317 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2002.

[51] D. Hofstadter. How could copycat ever be creative? In S. Harrison, editor, *Artificial Intelligence and Creativity: Papers from the 1993 Spring Symposium*, Technical Report SS-93-01, pages 1–10. AAAI Press, Menlo Park, California, 1993.

[52] D. Hofstadter and G. 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.

[53] D. Hofstadter and M. Mitchell. The copycat project: A model of mental fluidity and analogy-making. In D. Hofstadter and the Fluid Analogies Research Group, editors, *Fluid Concepts and Creative Analogies: Computer Models of the Funamental Mechanisms of Thought*, chapter 5, pages 205–267. Basic Books, 1995.

[54] K. J. Holyoak and P. Thagard. Analogical mapping by constraint satisfaction. *Cognitive Science*, 13(3):295–355, 1989.

[55] K. J. Holyoak and P. Thagard. A computational model of analogical problem solving. In S. Vosniadou and A. Ortony, editors, *Similarity and Analogical Reasoning*, chapter 8, pages 242–266. Cambridge University Press, 1989.

[56] P. N. Johnson-Laird. Analogy and the exercise of creativity. In S. Vosniadou and A. Ortony, editors, *Similarity and Analogical Reasoning*, chapter 11, pages 313–331. Cambridge University Press, 1989.

[57] J. Kolodner. *Case-Based Reasoning*. Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[58] T. Kurtoglu and T. F. Stahovich. Interpreting schematic sketches using physical reasoning. In Davis et al. [14], pages 78–85.

[59] J. H. Larkin and H. A. Simon. Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science*, 11:65–99, 1987.

[60] G. Lee, R. Sacks, and C. M. Eastman. Embedding expert knowledge in a building information modeling system. Submitted for publication, December 2004.

[61] J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 4, pages 463–502. Edinburgh University Press, 1969.

[62] G. McGraw and D. 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.

[63] N. H. Narayanan, M. Suwa, and H. Motoda. How things appear to work: Predicting behaviors from device diagrams. In *Proc. 12th National Conf. on Artificial Intelligence (AAAI-94)*, pages 1161–1167. AAAI Press, 1994.

[64] N. H. Narayanan, M. Suwa, and H. Motoda. Hypothesizing behaviors from device diagrams. In Glasgow et al. [43], pages 501–534.

[65] A. Newell. The knowledge level. *AI Magazine*, 2(2):1–20, 33, Summer 1981.

[66] P. Nielsen. A qualitative approach to mechanical constraint. In *Proc. 7th National Conf. on Artificial Intelligence (AAAI-88)*, pages 270–274. AAAI Press, 1988.

[67] S. O'Hara and B. Indurkhya. Incorporating (re)-interpretation in case-based reasoning. In S. Wess, K.-D. Althoff, and M. M. Richter, editors, *Topics in Case-Based Reasoning: First European Workshop (EWCBR-93)*, volume 837 of *Lecture Notes in Computer Science*, pages 246–260. Springer, 1994.

[68] M. Pearce, A. K. Goel, J. L. Kolodner, C. Zimring, L. Sentosa, and R. Billington. Case-based design support: A case study in architectural design. *IEEE Expert: Intelligent Systems & their Applications*, 7(5):14–20, 1992.

[69] A. Sloman. Musings on the roles of logical and nonlogical representations in intelligence. In Glasgow et al. [43], pages 7–32.

[70] B. Smyth and M. T. Keane. Adaptation-guided retrieval: Questioning the similarity assumption in reasoning. *Artificial Intelligence*, 102(2):249–293, 1998.

[71] B. Smyth, M. T. Keane, and P. Cunningham. Hierarchical case-based reasoning integrating case-based and decompositional problem-solving techniques for plant-control software design. *IEEE Transactions on Knolwedge and Data Engineering*, 13(5):793–812, 2001.

[72] T. F. Stahovich, R. Davis, and H. Shrobe. Generating multiple new designs from a sketch. *Artificial Intelligence*, 104(1–2):211–264, 2001.

[73] P. Thagard, D. Gochfeld, and S. Hardy. Visual analogical mapping. In *Proc. 14th Annual Conf. of the Cognitive Science Society*, pages 522–527. Lawrence Erlbaum Associates, 1992.

[74] P. Thagard, K. J. Holyoak, G. Nelson, and D. Gochfeld. Analog retrieval by constrain satisfaction. *Artificial Intelligence*, 46:259–310, 1990.

[75] M. Veloso and J. Carbonell. Learning analogies by analogy - the closed loop of memory organization and problem solving. In *Proc. 2nd Workshop on Case-Based Reasoning*, pages 153–158. Morgan Kaufmann, 1989.

[76] M. Veloso, J. Carbonell, P. Alicia, D. Borrajo, E. Fink, and J. Blythe. Integrating planning and learning: The PRODIGY architecture. *Journal of Experimental and Theoretical Artificial Intelligence*, 7(1), 1995.

[77] P. H. Winston. *Artificial Intelligence*. Addison-Wesley, Reading, MA, third edition, 1992.

[78] P. W. Yaner and A. K. Goel. Retrieving 2-D line drawings by example. In Hegarty et al. [50], pages 97–99.

[79] P. W. Yaner and A. K. Goel. Visual case-based reasoning I: Memory and retrieval. In *Proc. IICAI-03*, Hyperbad, India, 2003. Springer-Verlag.

[80] P. W. Yaner and A. K. Goel. Visual analogy: Reexamining analogy as a constraint satisfaction problem. In *Proc. 26th Annual Meeting of the Cognitive Science Society*, pages 1482–1487. Lawrence Erlbaum Associates, August 2004.

[81] P. W. Yaner and A. K. Goel. Visual analogy: Viewing retrieval and mapping as constraint satisfaction problems. In J. S. Gero, B. Tversky, and T. Knight, editors, *Visual and Spatial Reasoning in Design III*, pages 233–253. Key Centre of Design Computing and Cognition, University of Sydney, NSW, Australia, July 2004.