

6 Comparisons with other Models of Analogy-Making

*There is no word which is used more loosely or in a greater variety of senses,
than 'Analogy'.*

-J.S. Mill

As J.S. Mill has noted, the word 'analogy' has been used in many ways. Even the 'experts' use the term in various senses; I suspect that no word has been used more often to describe cognitive models than 'Analogical'. Much has been written describing computation models in the areas of "Analogical Reasoning", "Analogical Mapping", "Analogical Learning", and "Analogical Problem Solving". Each of these areas defines

and examines analogy-making from a slightly different perspective. Although not unrelated, analogy-making models from one camp often have little in common with models from another.

Many of the differences have arisen between theories in the various camps because of very different initial assumptions made about analogy-making, as noted in Chapter 1. Consider this statement made by Keane, Ledgway, and Duff (1994): “In order to solve a problem by analogy one must first represent the problem in some form.” All researchers would probably agree that this is the first step in creating a computational model of analogy-making. However, as we have seen, the exact details of the initial representation have a large impact on exactly what problem is to be solved by the analogical model. The assumptions about the initial representations divide computational models into two major camps: the traditionalists, and, for lack of a better term, the non-traditionalists. This chapter will examine these two groups and their specific computational models.

6.1 Traditional Analogical Computer Models

In Chapter 1, I introduced the assumptions made by the traditionalists. Recall the assumptions:

1. *Analogy-making begins with two structures.*
2. *Analogy-making is a search through the structures in an attempt to find analogous parts.*
3. *Syntax alone determines the similarity between any two objects, attributes, or relations.*

4. *For any two relations to be seen as analogous, they must exactly match in terms of their number of arguments, and types of arguments.*
5. *Relations, attributes, and objects are forever distinctly different things.*
6. *Context plays no part in making the analogy.*
7. *The result of making an analogy is the creation of a mapping between corresponding pieces of the two structures.*

In a nutshell, the traditional view defines “analogy” as a special-purpose module that takes two structures as arguments and returns a set of links connecting analogous items. Analogy-making is construed as a search for the set of coherent correspondences between two structured representations. The set of coherent correspondences connects items from two domains. Often, one domain is considered better defined and is called the source domain or the base domain. The other is called the target domain, and the mapping of the analogy to the target is called *analogical transfer*.

In modeling analogy-making, the traditionalists have not been limited to GOF AI (“Good Old Fashioned Artificial Intelligence”, Haugeland, 1985) symbolic techniques. Many researchers have used more modern mechanisms, such as connectionist networks. The following section discusses those models, connectionist and symbolic, that make some, or all, of the traditional assumptions.

6.1.1 Structure Matching Engine

Gentner’s theories of structure-mapping (Gentner, 1980; Gentner, 1982; Gentner, 1983; Gentner and Gentner, 1983; Gentner, 1986; Gentner, 1988) are well-known in cognitive science. Her theories have been implemented in a model called the Structure

Mapping Engine, or SME (Falkenhainer, Forbus, and Gentner, 1986). SME may be the best known of all the analogy-making models and is the prototypical traditional model.

The intuition behind Gentner's structure-mapping theory "is that an analogy is a mapping of knowledge from one domain (the base) into another (the target) which conveys that a system of relations that hold among the base objects also holds among the target objects." (Gentner, Markman, Ratterman, and Kotovsky, 1990). Of central importance to Gentner's structure-mapping theory is the notion of *systematicity*: "... People prefer to map connected *systems of relations* governed by higher-order relations with inferential import, rather than isolated predicates." (Gentner, 1989). A *higher-order relation* is a relationship that has as an argument another relation. Therefore, those relations that connect other relations are considered more important in an analogical mapping.

Gentner's main focus is the explanation of an analogy after a target has been retrieved from memory. As an example, consider the situation diagramed in Figure 6-1. On the left sits a large, full beaker of water connected to a nearly empty, small vial via an open tube. On the right is a warm cup of coffee containing a silver bar with an ice cube on the end. The Water-flow situation is to be considered well understood. That is, it is known that water will begin to flow into the small vial because of the unequal pressure. The question is then: What will happen in the Heat-flow situation?

SME processes the analogy as follows. First, representations of the *relevant information* are constructed from the source and target domains. In this case, the Water-flow information forms the source domain, and the Heat-flow information forms the target domain. Breaking the two domains into the object, attribute, and relation types, one might end up with a representation similar to that shown in Figure 6-2. Notice that the relation CAUSE in the source domain is a higher-level relation having the relation GREATER as one of its arguments.

It is worth noting at this point that this initial step of SME, which Hofstadter calls “gist extraction” (1995), is performed by the researcher. Humans typically find it very easy to consider Figure 6-2 and Figure 6-3 as being, in some sense, equivalent. However, it is important to realize how much mental processing occurs in order to make the leap

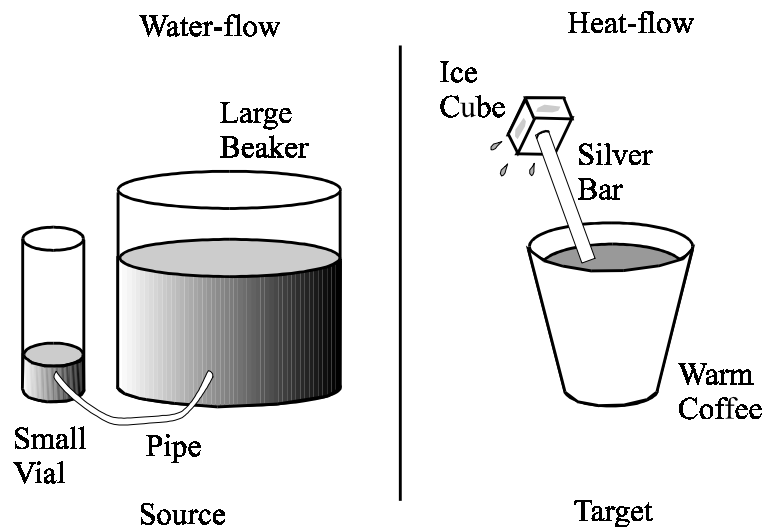


Figure 6-1. A problem for SME.

from the pictures to the gists. There are myriad details that can be seen or inferred from the pictures that did not make it into the small, structured representations (i.e. coffee is

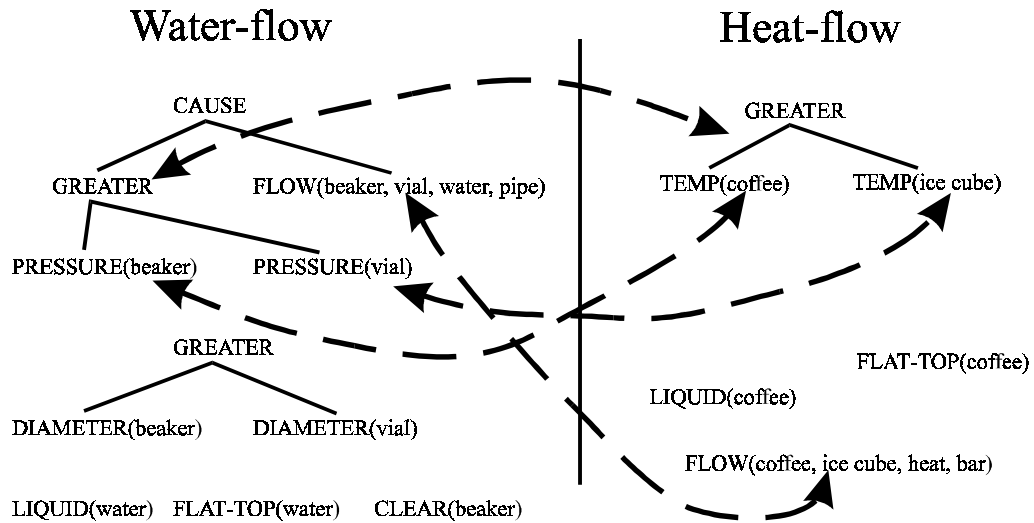


Figure 6-2. The problem from Figure 6-1 boiled down to its gists. The dashed lines indicate a valid mapping from source to target.

black, ice is made out of water, water is clear, the cup is sitting on a table, people sometimes drink coffee in the morning, etc.) Also, it is only in this specific context that we might consider the gists in Figure 6-2 to be the “correct” ones.

In the next step, SME is handed the representations of the two domains. SME then begins searching for possible local matching relations. This is dictated by a set of simple rules:

1. If two relations have the same name, create a match hypothesis
2. For every match hypothesis between relations, check their arguments. If both are objects, or both are relations then create a match hypothesis between them.

(after Gentner, 1989)

Matches are made between all possible entities. A matching is only possible between relations with identical numbers of arguments. For instance, the relation FLOW would not be even considered for comparison to the relation TEMP. Also, matches are

not attempted between two entities unless they are of the same type (i.e., object or relation). Each match hypothesis is given a score based on how good the match is. Scores are based on whether or not two entities have the same name, etc. Next, processing turns toward finding global matches. A global match is a system of matches that is consistent (systematic). Match hypotheses turn into full-fledged matches if their scoring is high enough. Finally, from the set of matches, a single coherent set of a mappings is chosen based on its total matching score, and the analogy is “understood” (the gray arrows in Figure 6-1). In the example shown in Figure 6-1, **beaker** would be found to map to **coffee**, **vial** to **ice cube**, **water** to **heat**, and **pipe** to **bar**. Transferring the well-known information from the Water-flow to the Heat-flow situation produces the previous unknown relation:

CAUSE(GREATER(TEMP(**coffee**), TEMP(**ice cube**)), FLOW(**coffee**, **ice cube**, **heat**, **bar**))

That is, “the coffee’s warmer temperature causes heat to flow up the bar to the ice cube.”

SME’s basic algorithm has provided the foundation for many extensions (see for example, Gentner and Forbus, 1991; Forbus and Oblinger, 1990), and as the basis for many models in psychology and artificial intelligence (see for example, Fallkenhainer, 1987; Bhansali and Harandi, 1997). In addition, Gentner’s structure mapping theories provide the foundation for many other models, including some of the most recent connectionist models (see Gentner and Markman, 1993, for a discussion of analogy-making directed at connectionists; see also, Plate, 1991; Kanerva, 1996; Handler and Cooper, 1993).

In summary, SME is a serial *generate-test-and-select* algorithm based on the syntax of pre-formed representations. It narrowly defines analogy-making as a specialized module that has no connection to other processes of similarity, such as categorization. Context plays no role in the model. It is deterministic, and does not incorporate learning.

As its focus is on the psychological explanation of mapping, it probably will not scale-up well as it requires all possible matches to be evaluated.

6.1.2 Analogical Constraint Matching Engine

The Analogical Constraint Matching Engine (ACME) created by Holyoak and Thagard (1989) was the first connectionist analogy-making model. However, it has far more in common with SME than Analogator. ACME, like SME, also creates mappings between symbolic structures, and, in fact, has been tested on many of the same problems that SME has tackled. ACME solves analogy problems by allowing activation to settle in a localist network with weighted links acting as soft constraints.

The ACME algorithm works as follows. First, like SME, the structured representations of the source and target domains are designed by the researchers and fed into the analogical engine. ACME then creates a node for each possible corresponding pair, although some mappings are not allowed based on syntactic constraints (see Figure 6-3). For example, an object cannot map onto a relation, and so that pairing would not receive a node. The nodes explicitly represent all possible analogous pairings. Weighted links are then created between pairs of nodes. Links allow activation to flow through the network. The links can have positive weightings if the pairings support each other, or may be negatively weighted if the pairings are mutually exclusive. The network also has positive links connecting all pairs of relations based on their *a priori*-judged similarity. The researchers assign these links, called *semantic links*, in advance. For example, one

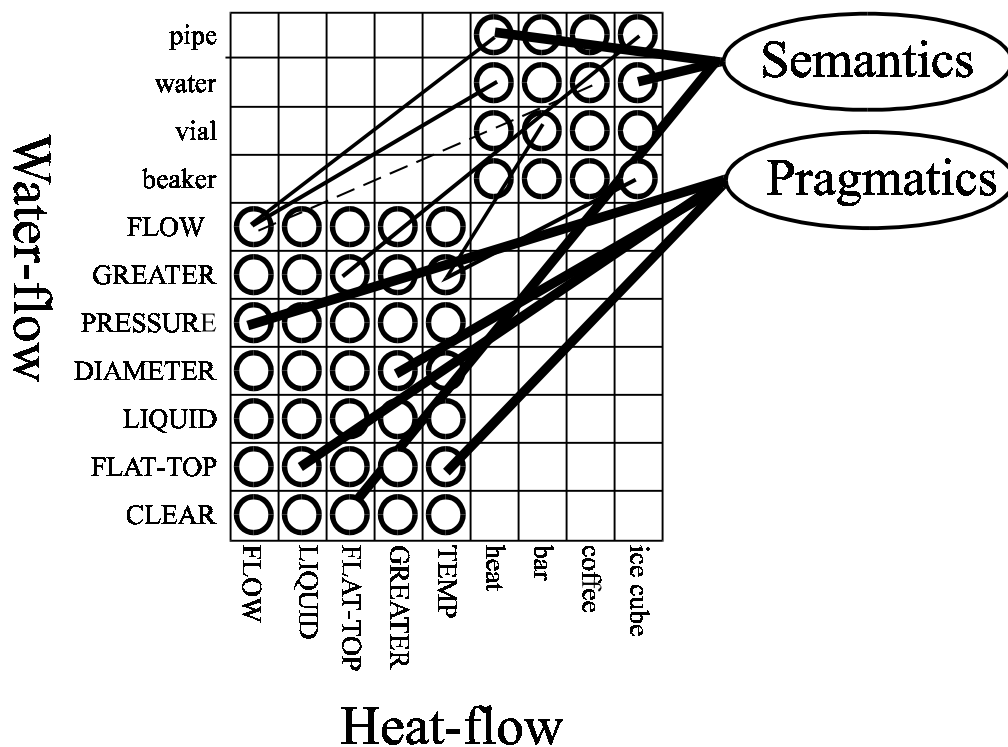


Figure 6-3. An example ACME network. This network depicts ACME's representation of Figure 6-1. ACME actually takes as input representations similar to those shown in Figure 6-2. It then constructs a network similar to that shown here. Not all links are shown.

may decide that the relations FLOW and FLUSH are very similar, and so should tend to be mutually excitatory. Finally, there is another set of links representing other important context and goal information. These links are called *pragmatic links*, and are also assigned by the designer prior to processing.

Once the network has been established activation is allowed to ebb and flow over the links as defined by a relaxation algorithm. If the network settles into a fixed state, a solution has been found for the analogy problem. Researchers have claimed that ACME “has achieved considerable success in simulating human intuitions about the natural mappings for a broad range of cases, including examples of scientific explanatory analogies, story analogies, problem analogies, metaphors, analogical arguments, and analogies between political situations” (Hummel, Burns, and Holyoak, 1994). Again, it is important to realize the distinction between a situation and its gist. It could be the case that the representations used for making an analogy between Vietnam and the United States in the 1960’s are exactly the same symbolic structures as used in an analogy between the solar system and the Rutherford atom. The names of the symbols might be different, but, of course, symbols’ names do not effect their use in the model. We will further discuss this issue later in this chapter.

One of the major criticisms against ACME has been its psychologically implausible requirement to allocate a node for every possible analogous correspondence (Hofstadter and Mitchell, 1994).¹⁷ Hummel, Burns, and Holyoak (1994) have addressed this issue by applying dynamic binding via synchrony of neural firing (von der Malsburg, 1981) to analogy-making. Hummel *et al.* (1994) have shown that an ACME-like connectionist network can make analogies via dynamic binding without explicitly

¹⁷ It should be noted that many of the traditional models consider all possible pairings. SME, for example, explicitly calculates scores for all legal, possible matches.

representing all possible matches. This is an important improvement over ACME. ACME has been extended in many other ways (Holyoak, 1994; Nelson, 1994). However, all of the extensions rely on the pre-structured initial representations.

In summary, ACME is a parallel constraint satisfaction network. It solves analogy problems via a relaxation algorithm that satisfies many soft constraints in parallel. Context may play a part in the analogy; however, goals and other contextual pragmatic information must be set by the researcher. ACME does not incorporate learning. In its most basic design it does not scale well in terms of capacity as it requires a node for all possible pairings.

6.1.3 Incremental Analogy Machine

Keane's Incremental Analogy Machine (IAM; Keane, 1994) adds a serial processing dimension to analogy-making not found in SME or ACME.¹⁸ IAM has the ability to eliminate many possible matches by incrementally establishing further constraints. By starting with a few "seed" matches, IAM limits the number of pairings that it might otherwise consider. If those seeds turn out to not pan out, then the system is capable of backtracking to find other seeds, and incrementally building coherent structures from those.

Although Keane (1994) has identified evidence of such serial processing in humans, his model still requires researchers to enter the initial representations.

6.1.4 Similarity as Interactive Activation and Mapping

Goldstone (1991) has applied some of the core ideas from SME and ACME to the problem of similarity judgment. Similarity judgment has traditionally been viewed as a simple calculation based on a comparison between two static lists of features, or between two static lists of dimension values. Goldstone's model, Similarity as Interactive Activation and Mapping (SIAM), accurately predicts subtle similarity judgment choices made by humans (Goldstone and Medin, 1994). In addition, SIAM makes accurate predictions about serial processing effects, like IAM does.

Although the inputs to SIAM are as static, rigid, and explicit as those used by ACME, SME and IAM, SIAM does allow for attribute values to dynamically become aligned during processing. Although this ability has limited effect, it allows SIAM more flexibility than the other traditional models discussed so far. Recall that SIAM was designed to only model the judgement of similarity, while the other models were specifically designed to process analogies. However, Goldstone (1991) introduced the ability for SIAM to not only make attribute-to-attribute and object-to-object connections, but also role-to-role connections as well. Although this could lead to a model capable of making more abstract similarity measures, or even analogies, it is still based on static, explicit representations.

¹⁸ Forbus, Ferguson, and Gentner (1994) have created an incremental version of SME (called Incremental SME, or I-SME) in response to Keane's suggestions (1994). ACME could also have such an incremental mechanism added. The criticisms made regarding ACME and SME's representations remain, however. Keane (1994) provides a detailed comparison between IAM, SME and ACME.

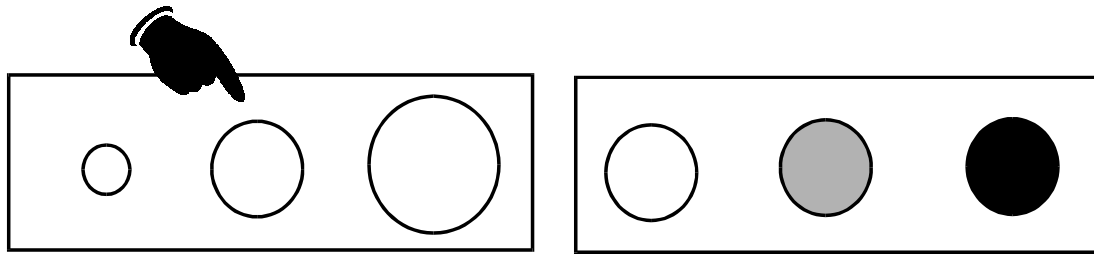


Figure 6-4. A sample problem from Handler and Cooper's SMERF domain (1993).

6.1.5 SMERF

Handler and Cooper (1993) have implemented a connectionist model that is capable of solving problems similar to those that lie in Analogator's geometric-spatial domain (see Figure 6-1). Their system, called SMERF, is meant to be an instantiation of Gentner's structure-mapping theory, and, not surprisingly, is very similar to SME in many respects. Like SME, SMERF also attempts to make a match between every pair of predicates. Like its connectionist cousin ACME, it does not incorporate learning. SMERF uses an algorithm that is very similar in style to the constraint satisfaction algorithm used by Holyoak and Thagard (1989). However, Handler and Cooper have adapted Markov Random Fields (Kindermann and Snell, 1980) to satisfy their constraints.

One point of similarity with Analogator is the form of the representations of relations. Rather than having explicitly structured representations like SME and ACME, SMERF's input is simply a set of features, one set for each object. They claim that their representation is better than that of ACME or SME because it does not explicitly represent relations between objects (Handler and Cooper, 1993). However, their solution has been to simply convert a relation between two objects, such as LARGER-THAN, into two values along a dimension. Instead of having explicitly structured representations, such as LARGER-THAN(A, B), they encode the size difference as values along a

dimension, such as SIZE(A, .8) and SIZE(B, .4). Although, this does make the representation implicitly structured and is similar in spirit to the representation used by Analogator, their model bypasses a major operation: perception. Their (unstated) assumption is that they do not need to have perception in the model as long as they can represent relevant attributes as scalar values along a dimension. This works fine for linear attributes, such as size and shading (two dimensions they happen to model). But this methodology does not necessarily work for dimensions such as color or position in two dimensions. Their representation is better than ACME's only in that they refuse to use explicitly structured representations, even if it means avoiding the problems that would seem to require them. Because they use values along a dimension, their model is similar to Goldstone's SIAM. However, it is doubtful that SMERF could account for the subtle judgment of similarity effects that SIAM does.

Like ACME, SMERF explicitly represents object-to-object mappings. The mappings are created in a constraint-satisfaction-like manner based on prior probabilities and weightings provided by the researcher. This information dictates to the network how to weigh the attribute dimensions in order to make the mapping.

Although they claim that their model could "naturally incorporate real perceptual input," I do not see how. First, their network is created based on the number of objects in a scene. Adding another object to a scene would require a completely different network. In addition, placing a fourth object into a scene already containing three would require a network nearly twice as large. Secondly, their network is incapable of creating, or perceiving, groupings within a scene. For example, a pair of circles could not be perceived as such. As mentioned, their method of dealing with structure is to simply avoid it. Finally, SMERF is incapable of representing context, let alone being influenced by it. For these reasons, I find it doubtful that SMERF could accommodate real perceptual input.

6.1.6 Other Connectionist Traditional Implementations

Recently, many connectionist models have been designed to study analogy-making. These traditional connectionist models fall roughly into two categories: those that are representation-centered, and those that are process-centered. The representation-centered projects have focused on distributed representations, and how they can naturally be used in analogy-making and similarity judgment. The process-centered projects have focused on incorporating connectionist mechanisms into large, analogy-making systems.

Kanerva's spattercode (1996), and Plate's Holographic Reduced Representations (HRR, 1991) are two connectionist representation-centered projects. In fact, the spattercode is equivalent to a HRR with certain limitations (Kanerva, 1996). Both are techniques for representing a collection of variables and their values, much like Smolenky's tensor product. However, both the spattercode and the HRR are able to bind sets of variables to their values without increasing the dimensionality of the representation. That is, all resulting representations remain the same size as the originating patterns.

In addition to this ability, Plate has applied the HRR to the task of estimating analogical similarity (Plate, 1993).¹⁹ Plate's goal was to devise a method that could quickly and efficiently estimate the similarity between two structured representations. This is a required step in Gentner and Forbus' MAC/FAC extension (1991) to SME.

First, we will describe the problem as presented by Gentner and Markman (1992). Imagine that we have many situations like the Water-flow and the Heat-flow

¹⁹ It is assumed that the spattercode is equally capable of performing the tasks to which Plate applied the HRR.

representations from Figure 6-2. Imagine that we now have a new situation. How can we quickly find the old situations that are most similar to the new situation? Plate found that if he encoded two situations via an enhanced HRR, the dot-product between the two can be used as a direct metric for their structural similarity. This appears as a much better solution than that originally proposed by Gentner and Forbus (1991). However, there are two problems with this methodology. The first is that this technique requires calculating the dot-product between the target and all sources in memory. This may be faster than other methods, however, it is still too computational intensive as a viable model of memory retrieval. The second problem is that the representations are still the same old prefabricated symbols and structures. This methodology, like most of the traditional models, leaves little room for perceptual and contextual effects.

Kanerva (personal communication) has suggested that the spattercode and HRRs may be able to be built from low-level sensory stimuli rather than from traditional symbolic representations. If this could occur, and can reflect structure and contextual pressures, this could be a very useful mechanism. However, without those abilities I don't believe HRRs will advance the state-of-the-art of analogy-making.

Halford, Wilson, Guo, Gaylor, Wiles, and Stewart's STAR model (1994) is another connectionist representation-centered project, and is similar to the spattercode and HRRs. STAR uses Smolensky's tensor product to create representations of the source and target scenarios. The STAR model uses a tensor of rank-3 to represent a predicate of two arguments, one for the predicate, and one each for the two arguments. In that manner, the relation *ABOVE(square, circle)* would be a three-way binding of *ABOVE*, *square*, and *circle*. STAR, like the HRRs and spattercode, uses explicitly structured representations, and therefore, places it squarely in the traditional camp.

Although STAR, HRR, and the spattercoding use explicit structures, all of them allow for holistic connectionist processes to operate on their distributed patterns. Such a

holistic process might be able to learn to manipulate the patterns without explicitly decomposing the patterns into their constituent components (see Chalmers, 1990, and Blank, Meeden, Marshall, 1992). For example, a network could learn to extract the representation of the figure given some context, much like Analogator does.

One might wonder whether representations of this type could be used as the input representation in an Analogator network. I think that this prospect is unlikely to work with HRRs and the spattercode. HRRs are complex encoding schemes with varying patterns for each element being represented. That is, the representation of **square** in the relation **OVER(square, triangle)** would likely be very different from the representation of **square** in the relation **OVER(square, circle)**. In Analogator the representation of any object or person was always the same (on the input layer) no matter what the relation was.

Because STAR contains explicit patterns for relations, I think that using its representation as inputs to Analogator would prevent generalization. As an example, consider the family tree cross-domain analogies from Chapter 5. If the relationships had had explicit patterns, the network would have “paid attention” to them, as they relay useful information. However, in order to make cross-domain analogies, the network needs to ignore the specific relations, and look instead to the implicit structure of the relations. I believe that using implicit relations forces Analogator to learn the structure; ironically, providing more information (i.e., the relation names) would make Analogator unable to generalize across domains.

The second group of traditional connectionist models are those focused on process. Barnden’s ABR-Conposit (1994), Eskridge’s ASTRA (1994), and Kokinov’s AMBR (1994) are all complex, hybrid analogy-making systems revolving around connectionist mechanisms. All of them incorporate spreading activation and marker passing as methods of coordinating processing between symbolic and connectionist

subsystems. All three models are impressively complex systems that have shown how symbolic and subsymbolic processes may operate in concert. Unfortunately, all three systems also subscribe to the traditional assumptions regarding pre-structured representations. Although all of these three models incorporate learning, none of them address the problem of how the analogy-making mechanism could develop.

6.2 Non-traditional Analogical Computer Models

Recall that the non-traditional models do not subscribe to the traditional assumptions. Like the traditional models, the non-traditional models come in GOFAI and connectionist versions.

6.2.1 Evans' ANALOGY program

The first AI program to address head-on the issues of analogy-making was Evans' ANALOGY program (1968). ANALOGY was designed to answer proportional geometric analogy problems (see Figure 6-5) that were, at one time, common on American I.Q. tests. The idea of the ANALOGY task is to answer the question "if the picture A changes into picture B, then what does picture C change into?" The question is answered by simply selecting one of five pictures (the bottom row of pictures in Figure 6-5). Based on ANALOGY's abilities, there are at least a couple of answers for Figure 6-5. One possible choice is #4, as it completes the analogous abstraction "removal of the inner object." However, ANALOGY could also have answered choice #2, as it was capable of complex transformations, such as "take the big object and delete it. Then, make the little object bigger."

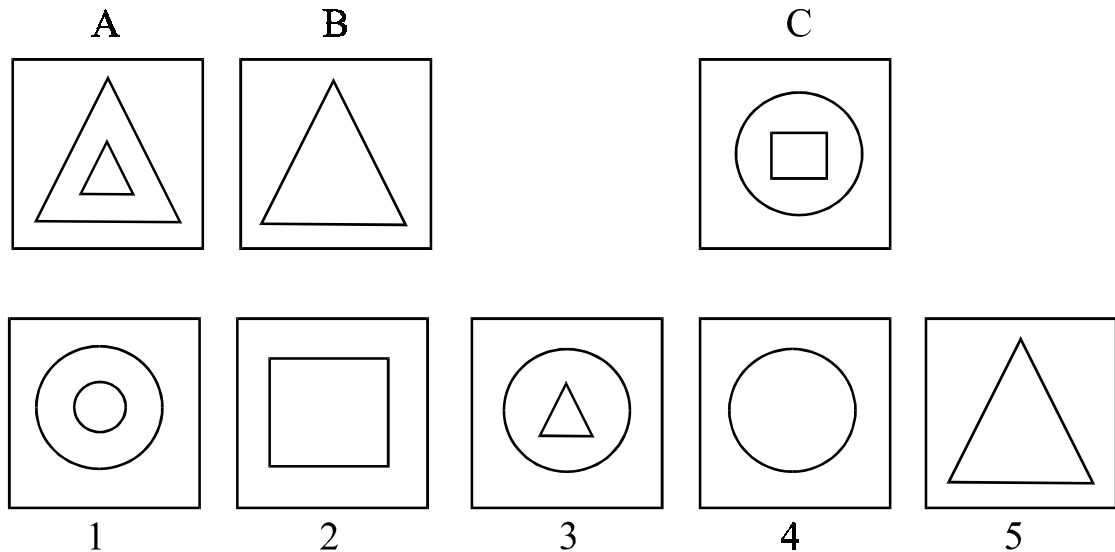


Figure 6-5. A sample problem from Evans' ANALOGY domain.

Unlike any program so far discussed in this chapter, objects and relations in ANALOGY were provided implicitly. For instance, the two triangles in frame A of Figure 6-5 might have been initially represented as:

```
(OBJECT-1
  (LINE .5 .2 .8 .8)
  (LINE .8 .8 .2 .8)
  (LINE .2 .8 .5 .2))
(OBJECT-2
  (LINE .5 .4 .5 .5)
  (LINE .5 .5 .4 .5)
  (LINE .4 .5 .5 .4)).
```

Relations such as *inside*, had to be recognized by the program. The categorization of objects and relations could be slightly effected by context, but this was not a major focus of the model. The size of an object was its only attribute, and was called *scale*. ANALOGY sequentially turned each frame's implicit representation into an explicit representation.

After the creation of explicitly structured representations for all of the frames, a list of possible transformations that could turn picture A into picture B was then created. Any series of transformations that could successfully transform A into B was considered as a possible ‘rule’ with which described the analogy. These rules were then compared with the set of rules that transformed picture C into each of the pictures 1 through 5. Rules that matched, either partially or exactly, were then ranked according to a similarity measure. Partially matching rules were penalized. If a rule described a transformation that left the scene unchanged, then the rule described two scenes that were very similar, and given a high ranking. The various kinds of transformations (e.g., scaling, rotation, and reflection) and their combinations were given lower rankings, as their related scenes were less similar. For instance, a rule that described a simple rotation was ranked higher on the similarity measure than a rule describing a rotation, a scaling, and a reflection.

Once the rankings were made for all of the remaining rules, the one with the highest score won. At the time, ANALOGY was highly regarded and has been touted as one of the classic AI programs. Recently, however, it has had its critics:

Yet ANALOGY has its serious limitations. Using the domain of geometric figures is the most serious limitation of the program. Because of the many hidden assumptions carried along with such a domain, AI work in analogical reasoning has been misdirected in a number of critical ways.

(Kedar-Cabelli, 1988)

Of course, as ANALOGY doesn’t learn, I, too, think that ANALOGY has its limitations. However, I disagree with Kedar-Cabelli as to what those problems are. She states that the domain of geometric figures has “misdirected” the entire study of analogy-making because of “many hidden assumptions.” By this, I suspect that she means that ANALOGY was specifically designed for geometrical analogies, and, as such, has relied on techniques that do not transfer to other domains, such as analogies involving solar systems, Vietnam, heat flow, etc. Tabletop (French, 1992), as we shall see, has also been

criticized in this manner. The idea that analogy-making should be free of any domain specifics and based on syntax alone has become quite popular in the last decade.

Although ANALOGY was very primitive in many ways, it did touch on issues (such as the perception of objects) that were completely ignored in the field for a couple of decades.

6.2.2 Tabletop

As mentioned, Hofstadter and his colleague's ideas have provided much inspiration for the Analogator project. Their influences on this project have been in three major areas: 1) the idea of analogy as high-level perception, 2) the idea of modeling high-level cognitive behavior via low-level, emergent processes, and 3) the Tabletop domain itself. These three points will be examined in this section.

Tabletop was designed to model *high-level perception* (French, 1992; Hofstadter *et al.*, 1995). High-level perception is the level of perceiving similarity at which concepts begin to play a role. Indeed, the concept of "concept" lies at the center of all of the projects that have come from Hofstadter's group, the Fluid Analogies Research Group (FARG). As stated previously, I believe the notion that analogy-making can be viewed as the categorization and recognition of conceptual patterns is one of the most important to cognitive science. This simple notion places the emphasis of analogy-making on the perception of situations rather than on a search through rigid representations. Of course, this is exactly the distinction used in this chapter in which to discuss all models of analogy-making.

The Tabletop domain was the inspiration for the geometric-spatial analogy domain used by Analogator. Tabletop operates in a microworld composed of a café tabletop, complete with saucers, cups, spoons, forks, etc. The Tabletop domain is more flexible than that used by Analogator in that the program has the ability to pick any object on either side of the table as the analogous object (see Figure 6-6). For instance, the program can point to the very same cup as the one being pointed to. It is also a richer environment, as each object may have many relationships between other objects.

In Tabletop, conceptual distances between concepts can dynamically change to reflect the how the scene is currently perceived. Tabletop analogy-making is a complex process which can be described at multiple levels. At the lowest level, a series of little programs, called *codelets*, take stochastic turns doing little bits of processing. For instance, one codelet might pick two objects and, if they are currently labeled the same, might suggest that a group be built from them. At a higher-level one might describe the behavior of the whole system as “heading toward” an answer as things look like they are

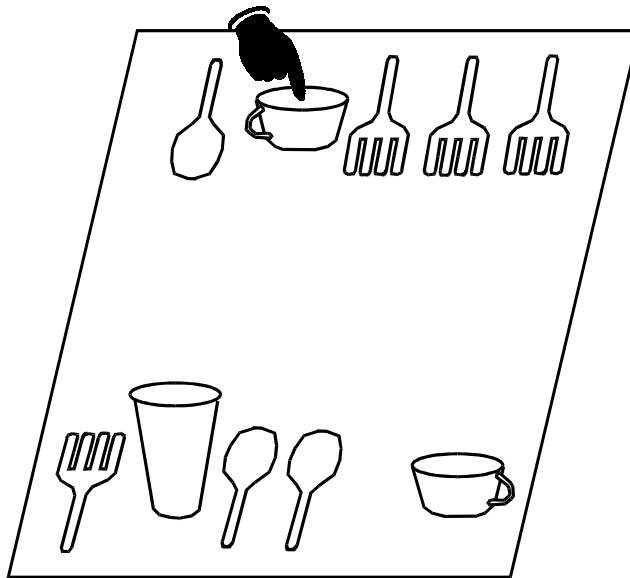


Figure 6-6. A sample problem from the Tabletop domain (French, 1992). Which object is analogous to the cup being pointed to? The cup would be a literal choice, but the glass would be a more abstract one.

falling into place. In this way, global control emerges from the tiny, random operations of the codelets.

Probably the biggest difference between Analogator and Tabletop is the role of concepts in the two systems. Tabletop was designed to operate at the level of concepts, and serves as a model to explore them. On the other hand, Analogator does not explicitly manipulate concepts, but was designed to focus on the learning of analogical behavior. In this context, Tabletop raises many questions: Where do codelets come from? Can they be learned? Where does the semantic knowledge come from? Can it be learned? What type of hardware could perform both low- and high-level perception? Tabletop does not incorporate learning, although it does in the course of running produce and destroy many structures built in working memory.²⁰

One common complaint heard about Tabletop, and its sibling Copycat (Mitchell, 1993), is that it is restricted to a single domain whereas other models, such as SME and ACME, are not (Holyoak and Thagard, 1995 pp. 259; Burns, 1995; Morrison and Dietrich, 1995). This is a prime example of the confusion that can be caused by conflating a situation with its gist. As we have seen, a gist is often represented by a set of symbols and relationships; no information is gained by labeling the symbol that represents our solar system as *SOLAR-SYSTEM*. That is, models that use these representations would operate exactly the same if that same symbol were re-label as *DELTA-GAMMA-64* (Hofstadter *et al.*, 1995). Of course, without descriptive labels, such as *SOLAR-SYSTEM*, one would be hard pressed to understand what a model was actually doing. However, by giving the representations meaningful names, one is willing to give much more credit to the system than it actually deserves. What is it, then, that differentiates one domain from

²⁰ Metacat is an extension to Copycat (Mitchell, 1993) and incorporates the learning of episodic memories (Marshall, forthcoming).

another? Unfortunately, the labels assigned to the symbols are often the only criteria used. Mitchell and Hofstadter (1995) appeal to another technique in an attempt to discover the “meaning” behind a symbol labeled *SIGMA*:

An astute human watching the performance of Copycat and seeing the term “SIGMA” evoked over and over again by the presence of successor relationships and successorship groups in many diverse problems would be likely to make the connection after a while, and then might say, “Oh, I get it – it appears that the Lisp atom ‘SIGMA’ stands for the idea of successorship.”

Of course, this is exactly the method required with connectionist networks, as there aren’t any nicely labeled symbols. Hofstadter’s models have also been criticized because they operate in a “toy domain” compared to richer domains like solar systems and water-flow-physics (Mitchell and Hofstadter, 1995). Of course, this is the same confusion detailed above manifested in a slightly different light. These are not mere technical arguments, but highlight many of the places that give rise to confusion when very different assumptions are made regarding analogy and representations.

Are the FARG programs and architecture fundamentally different from the traditional models? Burns and Holyoak (1994) have constructed initial representations for ACME to allow it to work on problems in Copycat’s domain. After setting up the problem, certain pragmatic and semantic values are clamped and certain structures are added or deleted. The model is capable of getting many of the same answers that the Copycat architecture does. To get a specific answer requires the researcher to assign pragmatic values and tweak the representations, a process that is built into the Copycat and Tabletop architecture. Burns and Holyoak (1994) claim that assigning the pragmatics and tweaking the representations “does not in itself provide ACME with a solution to the problem.” They argue that this merely biases the constraint satisfaction process so that the appropriate answer emerges. However, this step is required by the researchers in order for ACME to perceive the analogy problem in a particular way. Of course, as perception is a

major focus of the Copycat and Tabletop projects, these steps are critical and included in the models.

In summary, Tabletop is the most psychologically plausible model to date, and the most sophisticated. However, it does not incorporate learning.

6.2.3 Connectionist Non-Traditional Implementations

Although Analogator basically defines this category, there have been some non-traditional connectionist models that have addressed analogy-making, if only peripherally. One such example that we have already examined was Hinton's family tree model (1988) discussed in Chapter 5. Another such example, Harris' polysemy model (1994), demonstrates basic analogy-making in a simple feed-forward network. Harris trained sentences from a simple *noun-verb-noun* grammar to be associated with a particular meaning. The goal was to learn the different meanings of the word "over". Over can mean "beyond", "above", or "across", among other things.

Specifically, Harris took a sentence, such as "cow belong (over) hill", and trained it to produce as output the representation for "beyond", the meaning of over in this

Table 6-1. Simple grammar training examples (after Harris, 1994).

#	Sentence	Status	Meaning
1)	Cow belongs (over) hill.	Trained	Beyond
2)	Car drives (over) bridge.	Trained	Above
3)	Person lives (over) spot.	Trained	Above
4)	Person lives (over) bridge.	Not Trained	Beyond

situation.²¹ In the network, each word was represented by a random, orthogonal pattern. The input layer had three banks on the input layer, one for the verb, and two for the nouns. The network was trained on many sentences like those shown in Table 6-1.

To examine the network's generalization ability, consider sentence #4 in Table 6-1. Notice that it has much in common with sentences #2, and #3. Yet, the meaning of "over" as used in the sentence is actually more similar to that of sentence #1. Harris trained the network on sentences like 1-3, but did not train on sentence #4. Although sentence #4 is more similar to sentences that produced the representation for "above", she found that the network could ignore that superficial fact, and produce the proper output, namely "beyond".

Although this is far removed from the seemingly impressive analogies from other models discussed in this chapter, nothing about this task was built into the network or the representations; the network learned everything. This was a very simple task, however, models such as Harris' provided some of the initial inspiration for the Analogator project.

One non-traditional, connectionist model that did squarely confront more complex problems in analogy-making was Greber *et al.*'s GridFont network (1991). Although GridFont's designers used an incredibly small corpus, had a very impoverished representation, required the network to perform an impossibly difficult task, and used a simple feed-forward network, the model did not perform as badly as one might expect (see Hofstadter *et al.*, 1995, for a less generous critique). Although I believe that almost every aspect of GridFont was poorly designed, I also believe that it had the correct motivation and spirit: it attempted to learn how to make analogies in a given context.

²¹ The sentence did not actually contain the word "over" but it is included here to make it easier for the reader to understand.

This chapter has examined a few of the traditional and non-traditional models developed over the last 30 years. There are many more analogy-making models than are mentioned here. However, I believe that the selection provided is representative of the rest of those in the field. Having seen how Analogator compares and contrasts to other computational models of analogy-making, we now examine Analogator's limitations, and summarize.