In the *CRC Handbook of Dynamic Systems Modeling*. Paul Fishwick (ed.), Boca Raton, FL: CRC Press LLC. (2007).

The dynamics of the computational modeling of analogy-making

Robert M. French
LEAD-CNRS UMR 5022, Dijon, France
robert.french@u-bourgogne.fr

Introduction

In this paper we will begin by introducing a notion of analogy-making that is considerably broader than the normal construal of this term. We will argue that analogy-making, thus defined, is one of the most fundamental and powerful capacities in our cognitive arsenal. We will claim that the standard separation of the representation-building and mapping phases cannot ultimately succeed as a strategy for modeling analogy-making. In short, the context-specific representations that we use in short-term memory — and that computers will someday use in their short-term memories — must arise from a continual, dynamic interaction between high-level knowledge-based processes and low-level, largely unconscious associative memory-processes. We further suggest that this interactive process must be mediated by *context-dependent computational temperature*, a means by which the system dynamically monitors its own activity, ultimately allowing it to settle on the appropriate representations for a given context.

It is important to be clear about the goals of the present paper. This paper is not intended to be a review of computational models of analogy-making. For such a review, see, for example, Hall (1989), Gentner et al. (2001), French (2002), or Kokinov & French (2003). Rather, I will present a particular class of models developed, in the main, by Hofstadter and colleagues from the mid-1980's, in which dynamic, stochastic control mechanisms play a defining role. This, of course, is not to say that no other computer models of analogy-making incorporate dynamic control mechanisms. Certainly, for example, the settling mechanisms of Holyoak & Thagard's (1989) ACME, a constraint-satisfaction connectionist model, or the mechanisms of dynamic binding over distributed representations of Hummel & Holyoak's (1997) LISA model, are dynamic. The models by Gentner and colleagues (e.g., Gentner, 1983; Falkenhainer et al., 1989; Forbus et al., 1995; etc.) clearly have dynamic mechanisms built into them. Why, then, do I choose to discuss the Hofstadter family of models?

Several points set these models apart from all others (with the exception of a model, independently developed by Kokinov (1994) that adopted a similar design philosophy). One key principle is the eschewal of hand-coded representations. Instead, these programs rely on a dynamic feedback loop between the program's workspace and its long-term semantic memory that allows it to gradually converge on context-appropriate representations. This architecture was explicitly designed to allow scaling up without combinatorial explosion. The second key feature was the use of a context-dependent computational temperature function that mediated the degree to which the activity of the program was deterministic: the higher the temperature, the more random the program's choices became. Temperature is a measure of the overall quality of the structures perceived and as that structure becomes more and more coherent, temperature gradually falls and the program settles into a set of coherent, stable representations. When temperature is low enough, the program will stop.

Analogy-making as *sameness*

Before entering into a discussion of the dynamics of computational modeling of analogy-making, we must first make clear what we mean *analogy-making*. Frequently, what is understood by analogy-making is the classic, but more restricted, Aristotelian notion of *proportional* analogies. These take the form "A is to B as C is to D." For example, "*Left* is to *right* as *up* is to *down*" is an example of this kind of analogy. While this is certainly part of the story, I will take a broader view of analogy-making, one originally adopted, among others, by Hofstadter (1984), Mitchell & Hofstadter (1990), Chalmers, French, & Hofstadter (1992), Mitchell (1993), French (1995), Hofstadter et al (1995) and Kokinov (1994). In this view, analogy-making involves our ability to view a novel object, experience, or situation that belongs to one category as being *the same as* some other object, experience or situation, generally belonging to another category. This view is summed up by French (1995, p. *xv*): as follows:

> If only by definition, it is impossible for two things, *any* two things, to be exactly the same. And yet, there is nothing puzzling or inappropriate about our everyday use of the word "same." We see nothing odd or wrong about ordinary utterances such as: "That's the same man I saw yesterday at lunch," or "We both wore the same outfit," or, "I felt the same way when Julie and I broke up," or, finally, "That's the same problem the Americans had in Vietnam." What makes all these uses of "the same" (and this one, too) the same?
>
> The answer is: analogy-making. .... Since no two things are ever identical, what we really mean when we say "X is the same as Y," is that, within the particular context under discussion, X is the *counterpart* of Y. In other words, X is analogous to Y.

This way of looking at analogy-making, unlike the classic view of proportional analogy, allows us to speak of a *continuum* of analogy-making. This continuum runs from simple recognition — an apple we have never seen before is recognized as being a member of the category Apple because it is "the same as" other apples we have seen before — to deep "structural" analogies where elements in one situation are mapped to completely dissimilar elements in another situation — a fellow baseball player once said of homerun king, Hank Aaron, "Trying to sneak a fastball by Hank Aaron is like trying to sneak the sun past a rooster." In this analogy, Hank Aaron is mapped, completely naturally, to a rooster, and fastballs are mapped to the sun, even though, under normal circumstances, roosters have precious little to do with Hank Aaron and fastballs have even less to do with the sun.

Analogy-making as a means of "bootstrapping" cognition

The ability to see new things as being already familiar things with a twist is, unquestionably, one of the most powerful tools in the human cognitive arsenal from early infancy to adulthood. This ability to use analogies to understand novel situations allows infants (and adults in unfamiliar settings) to "bootstrap" new knowledge based on previously learned knowledge. In short, analogy-making allows us to comprehend new situations by seeing them as being "the same" as familiar situations that we already know how to handle, even if they require a bit of behavioral fine tuning.

In its simplest form, analogy-making involves finding a set of correspondences between a “base” object (or situation, experience, etc.) and a corresponding “target”. For example, you understand the statement “A credit card is like a check book,” because, even though credit cards do not physically resemble check books in the least, you effortlessly extract the appropriate parts of the representations of both — in this case, attributes related to their monetary function — and bring them into correspondence.

Young children, as we have said, constantly engage in analogy-making of the most complex kind. A perfectly run-of-the-mill example was provided one day by my not-yet-three-year-old son. He lightly touched the front bumpers of two of his little toy cars together and told me, “The cars are giving each other a little kiss, Dada.” What most people fail to realize, because it happens so often and seems so completely natural — “Of course little kids say things like that! It’s so cute!” remarked one of my friends, thoroughly unimpressed — is that his remark constitutes an amazing cognitive leap for a two year old. Think of the machinery put into play for him to have made that remark: people must be mapped to cars, cars front bumpers to lips, touching them together lightly (as opposed to slamming them together) constitutes “a little kiss”, etc.

The necessity of malleable representations

Well over a century ago, William James (1890) recognized just how malleable our representations of the world had to be:

> "There is no property ABSOLUTELY essential to one thing. The same property which figures as the essence of a thing on one occasion becomes a very inessential feature upon another. Now that I am writing, it is essential that I conceive my paper as a surface for inscription. . . . But if I wished to light a fire, and no other materials were by, the essential way of conceiving the paper would be as a combustible material. . . . The essence of a thing is that one of its properties which is so important for my interests that in comparison with it I may neglect the rest. . . . The properties which are important vary from man to man and from hour to hour. . . . many objects of daily use -- as paper, ink, butter, overcoat -- have properties of such constant unwavering importance, and have such stereotyped names, that we end by believing that to conceive them in those ways is to conceive them in the only true way. Those are no truer ways of conceiving them than any others; there are only more frequently serviceable ways to us." (James 1890, pp. 222-224)

This point, largely overlooked throughout the heyday of traditional AI, was reiterated by Chalmers, French & Hofstadter (1992). They apply James’ key insight to the domain of analogy-making and provide a number of examples to illustrate their point.

Consider, they say, deoxyribonucleic acid, DNA. There is an obvious — physical —analogy between DNA and a zipper. Here, we focus on two strands of paired nucleotides “unzipping” for the purposes of replication. A second analogy — focusing on information this time — involves comparing DNA to the source code of a computer program. What comes to mind now is the fact that information in the DNA gets "compiled" into enzymes, which correspond to machine code (i.e., executable code). In the latter analogy, the focus of the representation of DNA is radically different: the DNA is seen as an information-bearing entity, whose physical aspects, so important to the first analogy, are of virtually no consequence. While we obviously have a single rich representation of DNA in long-term memory, very different facets of this large, passive representational structure are selected out as being relevant, depending the pressures of a particular analogy context. The key point is that, independent of the vast, passive content of our long-term representation of DNA, the active content that is processed at a given time is determined by a very flexible representational process.

But one might argue that counterexamples of this kind do not really prove much of anything. One might claim that they are vanishingly small in number and, therefore, need not be taken into consideration when attempting to model the broad sweep of cognitive mechanisms, any more than one must worry about monotremes (i.e., egg-laying mammals, like the duck-billed platypus) when discussing mammals, in general.

For this reason, we consider the representation of an utterly ordinary object —in this case, a *credit card* — and demonstrate that the representational problems encountered for DNA, also exist for this ordinary object, and, by extension, *any* real-world object that we try to represent. We will consider a number of simple statements involving a "credit card" that will serve to illustrate how fluidly our representational focus changes depending on the analogy we wish to make. All of the following utterances are perfectly ordinary and easy to understand (for us humans):

"A credit card is like:

- money (here, the representational emphasis is on its *function*)
- a check book (again, an emphasis on its *function*)
- a playing card (emphasis on its size, shape, and relative rigidity; function becomes unimportant)
- a key (a thief can slide it between the door stop-strips and the door frame to open the door)
- a CD (bits of information are stored on its magnetic strip; bits of information are stored on the CD)
- a ruler (emphasis on its straight edges)
- Tupperware (both are made of plastic)
- a leaf (drop one from a high building and watch how it falls)
- a Braille book (emphasis on the raised letters and numbers on its surface)
- the Lacoste crocodile (emphasis on its potential snob-appeal)
- a suitcase full of clothes (Advertising jingle: "You can travel light when you have a Visa card...")
- a ball-and-chain (rack up too much debt on one and it's like wearing one of these...)
- a letter-opener (emphasis on its thin, rigid not-too-sharp edge)
- a hospital, a painting, a pair of skis, a Scotsman's kilt, etc. (left to the reader)

The point should by now be clear: With a little effort, by focusing on particular aspects of our passive long-term memory representations of "credit card" and the object to be put into correspondence with it, we can invariably come up with a context for which the two objects are "the same." In short, given the right contextual pressure, we can see virtually *any* object as being like any other.

W will now describe a process, which we call *dynamic context-dependent representation-building*, by means of which the analogical alignment between two objects (or experiences or situations, etc.) takes place in a progressive fashion, involving a continual interaction between long-term (passive) memory and working (active) memory. Most importantly, we claim that this dynamic process is machine-implementable, thereby allowing a *machine* to develop representations without the guiding hand of a programmer who knows ahead of time the analogy that the machine is supposed to produce.

The dynamics of representation-building in analogy-making

The representation of anything is, as I hope to have shown, highly context-dependent. Unfortunately, however, the early work of computational modeling of cognition was dominated by a far more rigid view of representations, one which Lakoff (1987) has called *objectivism*, described as follows:

> "On the objectivist view, reality comes complete with a unique correct, complete structure in terms of entities, properties and relations. This structure exists, independent of any human understanding."

Were this actually the case, a universal representation language and independent representation modules would make sense. "Divide and conquer" strategies for making progress in artificial intelligence would be reasonable, with some groups working on the "representation problem" and other groups independently working on how to process representations.

We argue, however, that such a division of labor is simply not possible (Chalmers, French, and Hofstadter, 1992; Mitchell, 1993; Hofstadter et al, 1995, French, 1995, 1997; etc.), at least in the broad area of computational modeling of analogy-making. While analogy-making does, indeed, consists of *representing* two situations and *mapping* the corresponding parts of each of these representations onto one another, these two operations are neither independent nor sequential (i.e., first, representation, then mapping). Representation and mapping in real analogy-making are inextricably interwoven: the only way to achieve context-appropriate representations is to continually and dynamically take into consideration the mapping process. And conversely, in order to produce a coherent mapping, the system must continually and dynamically take into consideration the representations being processed. In other words, representations depend on mapping and vice-versa and only a continual interaction of the two will allow a gradual convergence to an appropriate analogy.

The inevitable problem with any dissociation of representation and mapping can be summed up as follows. The complete representation of any object must include every possible aspect of that object, in all possible contexts, in order to produce the vast range of potential analogies with other objects. Now, presumably, we have stored in our long-term memory just this kind of mega-representation of all the objects, situations, experiences, etc., with which we are familiar. The problem is, though, that, even if it were possible (or desirable) to activate in working memory the full mega-representations for both the base and target objects, the very size of these representations would produce a combinatorial explosion of possible mappings between them. To determine precisely which parts of each mega-representation mapped onto one another would require a highly complex process of selection, filtering, and organizing of the information available in each representation. But in that case, we are back to square one, because the very reason for separating representation from mapping was to avoid this process of mapping-dependent filtering and organizing! On the other hand, if, in order to avoid the problem of this combinatorial explosion of mappings, you use smaller, reduced representations of each object, then the obvious question becomes, "On what basis did you reduce the original representation?" And this basis has been, almost invariably: the mapping that is to be obtained by the program!

The question then is: How do we arrive at the "right" representations that allow us to map people to cars, famous baseball players to roosters, and credit cars to kilts? We suggest that representations must be built up gradually by means of a continual interaction between (bottom-up) perceptual processes and (top-down) mapping processes. If a particular representation seems approximately appropriate for a given mapping, then that representation will continue to be developed and the associated mappings will continue to be made. If, on the other hand, the representation seems less promising, then alternative directions will be explored by the bottom-up (perceptual) processes. These two processes of representation-building and mapping are interleaved at all stages. In this way, the system gradually converges to an appropriate analogy, based on overall representations that dovetail with the final mappings.

Context-dependent computational temperature

The entire representation-building/mapping process is dynamically mediated by a feedback loop (*context-dependent computational temperature*, Hofstadter, 1984; Mitchell & Hofstadter, 1990; Mitchell, 1993; French, 1995) that indicates the quality of the overall set of representations and mappings that have been discovered. It is important to distinguish context-dependent computational temperature from the temperature function that drives simulated annealing, as described in Kirkpatrick et al. (1983). The former depends on the overall quality of the representational structures and mappings that have been built between them, whereas the latter is based on a pre-established annealing schedule.

The role of temperature is to allow the system to gradually, stochastically settle into a “high-quality” set of representations and mappings. It measures, roughly speaking, how willing the system is to take risks. At high temperature, it is very willing to do so, to abandon already created structures, to make mappings that would, under normal circumstances, be considered implausible, etc. Conversely, at low temperature, the program acts far more conservatively and takes few risks. This all-important mechanism will be discussed in greater detail later in this paper. Suffice it to say, for the moment, that temperature is inversely proportional to the program’s perception of the overall quality and coherence of the representations and mappings between them. As these representations and mappings gradually improve, temperature falls. With each temperature, there is a probability that the program will stop and select as “the analogy” the set of mappings that it has developed up to that point. The lower the temperature, the more probable that the program will conclude that it has found a set of good representations and mappings and stop.

Note that the use of a dynamic, context-sensitive computational temperature function is not limited to the computational modeling of analogy-making. For example, in the area of human mate-choice, French & Kus (2006) have built a model centered on context-sensitive computational temperature applied to individuals in a population. They likened this variable (actually, the inverse of it) to an individual’s *choosiness* in picking a mate: the higher the temperature, the less choosy an individual is about his/her mate; the lower the temperature, the more choosy.

Interaction between top-down and bottom-up processes: an example

Let us begin by illustrating what we mean by this interaction between top-down and bottom-up processing that, we claim, is the way for a system to converge to context-appropriate representations and mappings for an analogy. The example below actually occurred. I was asked to review a paper and accepted. But, unfortunately, I was unable to complete the review on time. Finally, the Action Editor contacted me and said, “I need your review quickly.” I thought, “OK, I’ll put the paper to review in plain sight next to my computer keyboard. Its continual presence there will goad me into getting it done.” I wanted to describe my strategy to the editor (a friend) in a more creative way than simply saying, “The paper is right next to my computer which will constantly remind me to get it done.” I chose to resort to an analogy. Below is my thought process as I recorded it immediately afterward (Figure 1). Notice that there is nothing out of the ordinary about this process — in fact, it is a completely everyday cognitive occurrence — but it clearly illustrates the back-and-forth switching between bottom-up, stochastic processes and top-down, knowledge-based processes.

The problem, then, is to find an analogy with the base situation: "Putting the paper in plain sight, so as to continually remind me to get the review done."

Bottom-up: "Feeding a crying child who is hungry" popped up as something that doesn't stop until you take care of it.

Top-down: Comparing a paper to review to feeding a crying, hungry child feels inappropriate because one is intimate and personal, while the other is purely professional. This then modifies the representation of the original situation: we realize that it is not merely something that won't go away until you complete it, but it is a *non-intimate* that won't go away until you do it and, thus, doesn't map well, to *intimate* situations.

Bottom-up: "Scratching an itch."

Top-down: But normal itches are *too easy* to get rid of, you simply scratch them and they disappear. Again, this redefines the representation of the original situation. It is a non-intimate something that won't go away until you've done it, but it's something that, nonetheless, takes some energy and effort to achieve.

Bottom-up: "Removing dandelions from your yard."

Top-down: No, getting rid of dandelions is *too hard*! Reviewing a paper isn't *that* hard. So, the target object has to be something that demands our attention and won't go away until we attend to it, is not too intimate, is not too easy to take care of, but not too hard, either.

Bottom-up: "Swatting a mosquito that's preventing you from sleeping." Who has never had their sleep troubled by the supremely irritating high-pitched whine of a mosquito? And, as we all know, it NEVER goes away until you get up out of bed, turn on the light, and chase it all over the room with a rolled up newspaper. And only with the Swat! that transforms the offending mosquito into a speck on the wall does the problem end. So, it fits the above criteria: it's a problem that won't go away till you take care of it, it's not to easy to do (unfortunately...), and it's not too hard either. That is the analogy I used.

It is crucial to note the aspects of my extensive (and passive) long-term memory representation of "mosquito" that became active in working memory were entirely dependent on the mapping that I wanted to achieve. In other words, the working memory representation does not include myriad other characteristics of mosquitoes, such as their disease-bearing nature, that they breed in water, that they feed on blood, that they have six legs, that they don't weigh very much, that they live from 3 to 100 days, that most of their head consists of their compound eye, etc. Further, in finding this analogy, we also modified our working-memory representation of "review a paper", in particular, focusing on reviews' strictly professional, non-intimate nature, on the fact that reviews are neither too easy or very hard to do, of the fact that no amount of waiting will allow you to make the slightest progress on the review, etc., all aspects that were not part of my working-memory representation of "review a paper" when I began looking for an appropriate analogy.

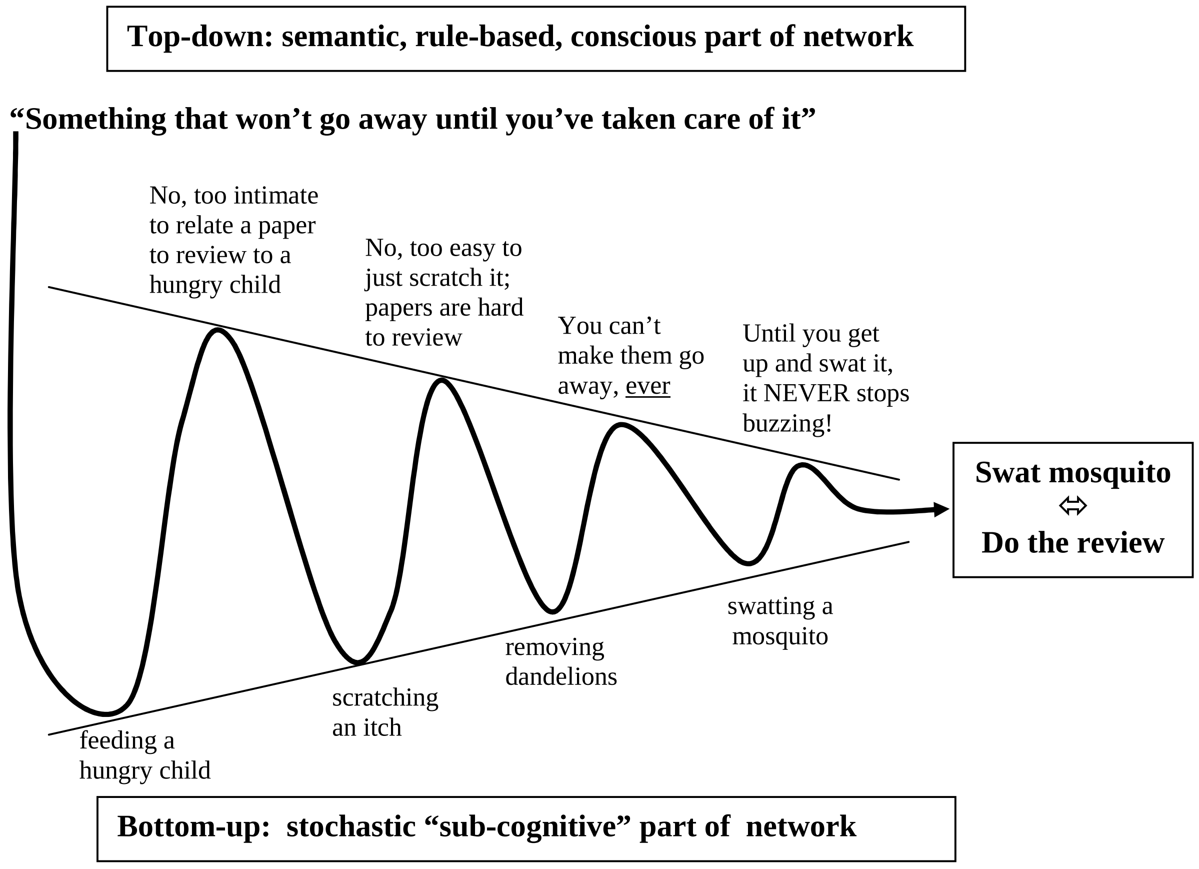

Top-down: semantic, rule-based, conscious part of network
"Something that won't go away until you've taken care of it"
No, too intimate to relate a paper to review to a hungry child
No, too easy to just scratch it; papers are hard to review
You can't make them go away, ever
Until you get up and swat it, it NEVER stops buzzing!
Swat mosquito ⇔ Do the review
feeding a hungry child
scratching an itch
removing dandelions
swatting a mosquito
Bottom-up: stochastic "sub-cognitive" part of network

Figure 1. The gradual interaction between top-down (knowledge-based) and bottom-up (stochastic) processes that lead to context-appropriate representations

Computational models implementing this bottom-up/top-down interaction

"Hybrid" models of analogy-making (Kokinov & French, 2003) explicitly attempt to integrate mechanisms from symbolic and connectionist approaches to artificial intelligence. Crucially, they rely on a distinction between long-term memory and working memory, although the implementational details of this separation may vary.

The overarching principles of the class of models that I will briefly describe were originally formulated in Douglas Hofstadter's research group from the 1980's to the present. Many people have contributed to the development and implementation of these ideas, including Hofstadter, 1984; Defays, 1986; Mitchell & Hofstadter, 1990; Hofstadter & Mitchell, 1992; Mitchell, 1993; French, 1995; Hofstadter et al., 1995; McGraw, 1995; Bolland & Wiles, 2000; Marshall, 2002a,b; Bolland, 2005; etc.).

These models — Copycat, Numbo, Tabletop, Letter Spirit, Metacat, Fluid Analogies Engine, etc. — are all agent-driven models that operate in micro-worlds that are, by design, felt to be rich enough to demonstrate the architectural principles of the models, but not so complicated that extraneous details interfere with understanding. It is beyond the scope of the present paper to present a detailed defense of micro-worlds. Suffice it to say that micro-domains can be appropriately likened to the "ideal objects" studied in elementary physics to understand the principles of movement, gravity, force, reaction, etc. Only once these principles are mastered in the "micro-domain" of these ideal objects does it make sense to be concerned with friction, air resistance, dilation due to heat, etc. The micro-domains in which these analogy-making programs work are to designed to help isolate the principles of analogy-making. Micro-domains are to analogy-making what "ideal objects" are to physics.

Architectural principles

The basic features of the Hofstadter family of architectures are (cf. French, 1995, ch. 3; see also, Bolland, 1997, for a particularly clear presentation of Copycat, the mainstay program of this class of programs).

A semantic network

This is a semantic network with fixed concepts, but in which the distances between concepts vary according to the situation being processed. This context-dependent malleability is a crucial feature of the architecture. A rather vivid example (Bonnetain, 1986) will serve to illustrate how significantly context can modify distances between concepts in a semantic network. In France, unlike in the United States, a driver's license is needed *only* when one is operating a motor vehicle and does not serve as an identity card (the French are required to carry a National Identity Card at all times). Further, while laws aimed at preventing under-age drinking in France do exist, they are rarely enforced. In other words, no one ever checks young people's National ID Card at the entrance to a bar in France, much less asks for a driver's license. As a result, in the minds of a nineteen-year-old French college student, the concepts "beer" and "driver's license" are every bit as distant as, say, "refrigerator" and "tennis ball". But were the same French college student to go to the United States for a year — where drinking age *is* enforced and a driver's license is invariably used as the means of checking ages — his/her conceptual distance between "driver's license" and "beer" would soon be radically altered. This would result in a change in the Slipnet structure — namely, the shortening of the conceptual distance between the concepts "beer" and "driver's license."

A workspace

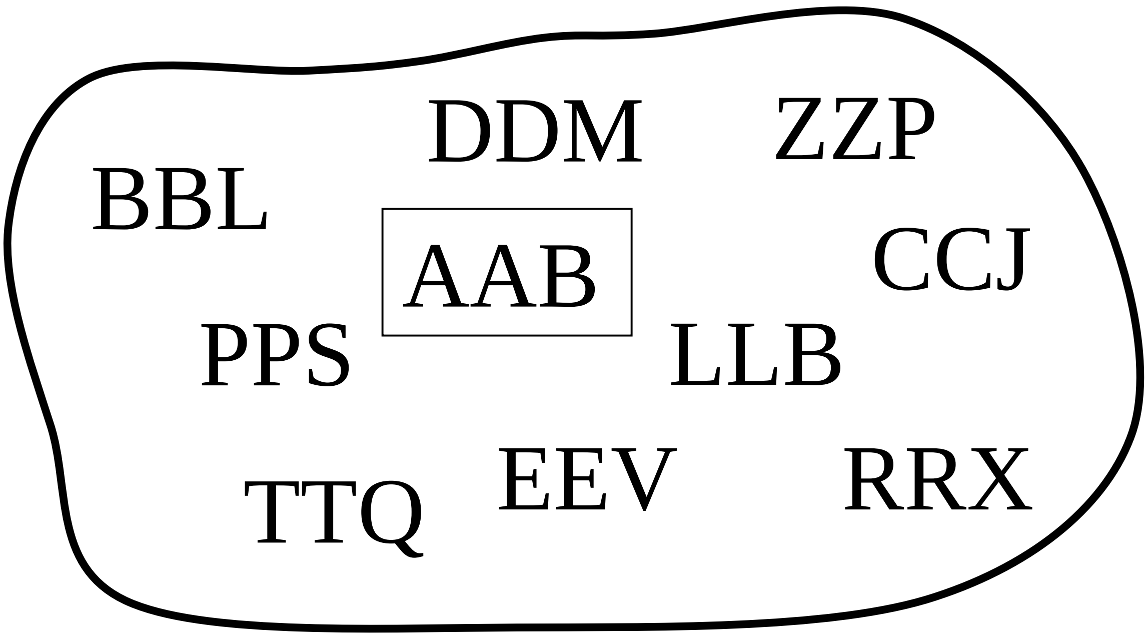
This is where all the representational structures needed to perceive the situation being examined are gradually built up. Consider the following group of letters to be parsed:

AAB

There are two obvious possible parsings of this three-letter sequence — namely:

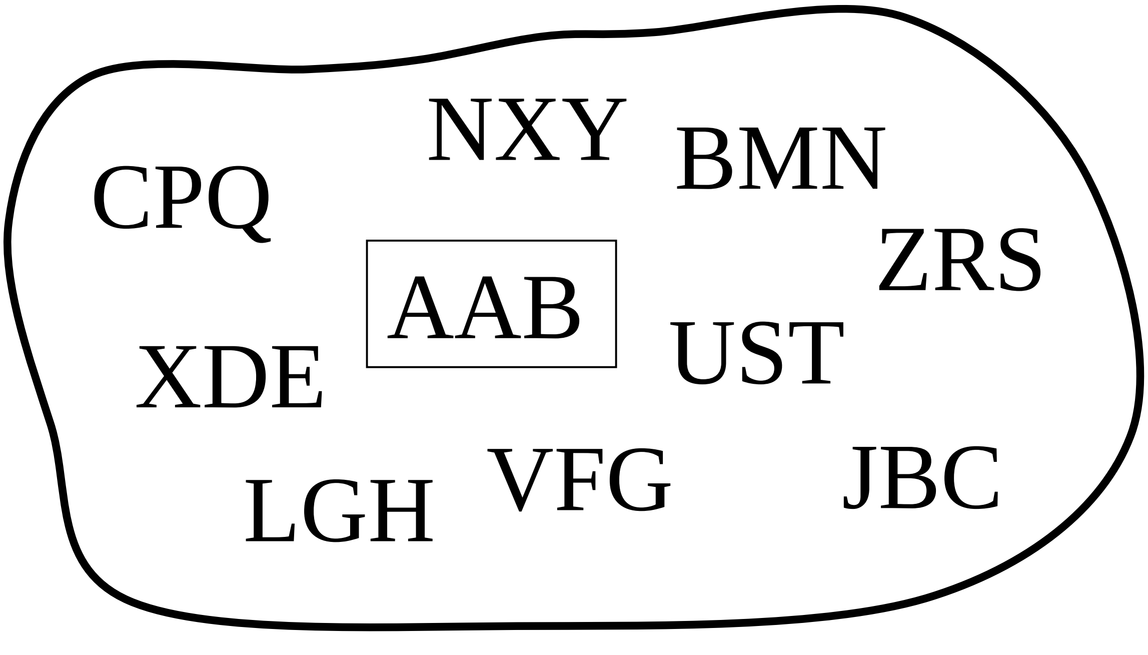
- An AA group, followed by a B,
- An AB group, preceded by an A.

However, neither representation is a priori more correct than the other. So, if this A-A-B group is embedded in a context, such as:



the chances are that the letter string will be parsed as a “An AA group, followed by a B.”

On the other hand, if the context in which this image is embedded is:



this will significantly increase the probability that it will be parsed as a “an AB group, preceded by an A”

So, in a setting in which the program, in order to make an analogy, must parse various configurations of AAB, both of the following structures:

AA and AB

are likely to be included in the workspace. These two potential representations — which cannot co-exist simultaneously in conscious perception, i.e., in the Worldview (see below) — will then compete with each other based on what else is found on the table. Ultimately, one will win out and will be used in the final representation.

The Worldview

These are the structures that are currently, consciously being perceived. The assumption here is that, just as in a Necker cube (or the Old Woman/Young Woman illusion, or any other ambiguous figure), one cannot perceive both of the possible interpretations simultaneously. Thus, there is one representation, based on the context as it is perceived up to that moment (say, for example "a pair of forks with a knife to the right"). Structures move in and out of the Worldview during the process of building up the final representation of the objects on the table. The longer a given representation remains in the Worldview, the lower the computational temperature of the simulation, meaning that the program believes that it has found a good, coherent way of representing the situation at hand. When the program stops, what is in the Worldview is considered to be the analogy that the program has discovered.

Codelets

These are the task-specific "worker ants" of the program. They carry out all of the processes necessary to produce an emerging understanding of the situation being represented and the mappings being made. None of these task-specific agents has an overview of what is going on, any more than a single ant has an overview of the anthill in which it toils. Agents are released based on numerous triggers, such as activation levels of concepts in the semantic network, computational temperature, structures that have already been discovered, etc. They each carry out very simple tasks. They build and break correspondences between structures, the discover relations between individual objects, etc. They perform a single task and then "die."

The Coderack

This is the "staging ground" for the task-specific agents. When an agent is called, it does not run immediately, but rather it is put onto the coderack. Each agent has an "urgency" value that expresses its importance and agents are probabilistically selected to run based on their urgency.

Dynamic codelet selection via computational temperature

This is where computational temperature plays a key role. Assume that there are N agents on the coderack waiting to run, each of which has an urgency u. The general idea of temperature-based selection is that, if the overall temperature (T) is very high ($T = 1$ is the "neutral" temperature), the probability of selection should be essentially independent of their urgency, and each agent will be as likely to be selected as any other (i.e., a uniform selection distribution over all agents, regardless of their urgency). On the other hand, the lower the temperature, the more codelet selection will take based strictly on the value of a codelet's urgency (i.e., in this case, selection is essentially deterministically based on urgency).

If there are N agents, $A_1, A_2, \ldots A_N$, each with a raw urgency, u_i, then for a given computational temperature, *T*, the probability that a given codelet, A_i, will be selected is given by:

$$\Pr(A_i) = \frac{u_i^{\alpha}}{\sum_{k=1}^{N} u_i^{\alpha}}$$

where $\alpha = \frac{1}{T}$

It is clear that as $T \to \infty$ (or gets large), that this formula reduces to:

$$\Pr(A_i) = \frac{u_i^0}{\sum_{k=1}^{N} u_i^0} = \frac{1}{\sum_{k=1}^{N} 1} = \frac{1}{N}$$

which is what we wanted, i.e., a uniform selection distribution. On the other hand, when temperature is low, α becomes large and the agent with the highest urgency will, if the temperature is low enough, be picked almost deterministically over its rivals with lower raw urgency.

An example will be useful to clarify this. Assume that there are three agents, A_1, A_2, and A_3, with respective urgencies, u_i, of 2, 3 and 5. When the temperature, *T*, is high, $\alpha \to 0$ and we have:

$$\Pr(A_1) = \frac{2^0}{2^0 + 3^0 + 5^0} = \frac{1}{3}$$
$$\Pr(A_2) = \frac{3^0}{2^0 + 3^0 + 5^0} = \frac{1}{3}$$
$$\Pr(A_3) = \frac{3^0}{2^0 + 3^0 + 5^0} = \frac{1}{3}$$

When T = 1, then $\alpha = 1$. This is the "neutral" selection condition, i.e., each agent has a selection probability determined by its urgency divided by the total urgency of all agents on the Coderack.

$$\Pr(A_1) = \frac{2^1}{2^1 + 3^1 + 5^1} = \frac{2}{10} = 0.2$$
$$\Pr(A_2) = \frac{3^1}{2^1 + 3^1 + 5^1} = \frac{3}{10} = 0.3$$
$$\Pr(A_3) = \frac{5^1}{2^1 + 3^1 + 5^1} = \frac{5}{10} = 0.5$$

Finally, assume the temperature falls to, say, 1/5. This means that $\alpha = 5$ and we have:

$$\Pr(A_1) = \frac{2^5}{2^5 + 3^5 + 5^5} = \frac{32}{3400} \approx 0.01$$
$$\Pr(A_2) = \frac{3^5}{2^5 + 3^5 + 5^5} = \frac{243}{3400} \approx 0.07$$
$$\Pr(A_3) = \frac{5^5}{2^5 + 3^5 + 5^5} = \frac{5}{3400} \approx 0.92$$

In this case, the selection probability of agent A_3 leaps to 0.92. In other words, the fact that A_3 has a raw urgency of 5, as opposed to its rivals with urgencies of 3 and 2, means that it will be chosen to run 92 times out of 100 when $T = 1/5$.

One might wonder why this mechanism is of any importance. In the language of classic tree-searching artificial intelligence, the answer is that it allows the program to occasionally (esp. when temperature is high) explore branches of the search tree that it would never have explored, thereby potentially arriving at a truly unexpected, excellent solution to a problem at the end of an exploration path that would not have been taken without this mechanism.

But, at the same time, one does not want the program to be searching these unusual paths very often. But one must not restrict the program to *never* search these paths. And, especially, in the event of hitting what appears to be a “dead-end” when attempting to find the answer to a problem, computational temperature, as implemented above, allows the program to break structures it has already made, to reform structures and make mappings in a new way, and then to gradually settle using these new structures. If another dead-end is encountered, the process will repeat.

One might wonder what would prevent this process from going on forever in the event that there are no good structure to be found. The answer is that at each temperature, there is a “StopWork” agent that is on the Coderack and, when it runs, all further processing comes to a halt. The structures that are currently in the Worldview, good or bad, are taken to be the best answer for that run. The “StopWork” agent will eventually run and bring to an end this cycle of building-breaking structures brought on by temperature oscillations. What frequently does, in fact, happen is that the program will find what it believes to be good structures, the temperature will therefore fall, but overall coherence never comes and so the temperature rises. But then the program once again discovers the same structures that led it down the fruitless path, and so on. In short, the program must remember where it has been, a capability which the earlier versions of this architecture did not have. This problem has been dealt with, albeit to a limited extent, in later versions of this architecture (eg., Marshall, 2002a,b).

In conclusion, Mitchell (1993, p. 44) sums up the importance of context-dependent computational temperature as follows:

> "...[Temperature] has two roles: it measures the degree of perceptual organization in the system (its value at any moment is a function of the amount and quality of structure built so far) and it controls the degree of randomness used in making decisions....Higher temperatures reflect the fact that there is little information on which to base decisions; lower temperatures reflect the fact that there is greater certainty about the basis for decisions."

Local, stochastic processing.

It is worth reiterating here that there is nothing that resembles a global executive in these programs, any more than there is a global executive in an anthill. All results emerge from the limited actions of a large number of agents responsible for finding/breaking structures, making maps, communicating with the semantic network, etc. All decisions — whether or not to group certain elements, whether or not to put certain structures into correspondence, etc. — are made stochastically.

Integration of representation-building and correspondence-finding.

Finally, and arguably, most importantly, there is no separation of the processes of building representations and finding correspondences between elements of different representations. A fundamental tenet of Copycat and Tabletop is that it is impossible to separate the process of representation-building from the process of correspondence-building (Chalmers, French, & Hofstadter, 1992).

How this type of program works: details

Certainly the best known implementation of this type of architecture is the Copycat program (Mitchell & Hofstadter, 1990; Mitchell, 1993). The best place to find a reasonably complete, yet succinct description of this program is in Mitchell (2001), an article was written by the program's author.

In what follows we will illustrate the principles discussed above by taking a look at another of these programs, Tabletop (French, 1995), designed to work in a simulated tabletop micro-world. The basic idea is that there are some ordinary objects (forks, spoons, cups, saucers, glasses, plates, salt and pepper shakers, etc.) on a small table. There are two people (Henry and Eliza) seated across from one another at the table. Henry touches one object and says to Eliza, "Do this!" (meaning, of course, "Touch the object you think is analogous to the one I just touched").

So, for example, if there is a cup in front of Henry, and only silverware in front of Eliza, and Henry touches his cup, Eliza will probably reach across the little table and touch Henry's cup (reason: silverware is semantically "too far" from the object that Henry touched, so she might as well touch exactly the same object he touched). Or suppose there is a cup in front of Henry and a glass and a two forks in front of Eliza. Henry touches the cup in front of him. In this case, Eliza will (most likely) touch the glass in front of her, even though the glass is not, strictly speaking, "the same" as a cup.

The contextual factors that can be brought to bear to shift the initial bottom-up pressures to touch one or the other object are almost limitless. Suppose Henry's cup is surrounded by two forks and a knife on one side of the cup and two spoons on the other and Eliza has, on her side of the table, only an isolated cup and an isolated glass. Henry touches his cup. Eliza will, almost certainly, touch the cup on her side of the table. If, on the other hand, her glass is surrounded by the same objects as Henry's cup, i.e., by two forks and a knife on one side and two spoons on the other (see Figure 2), she will be considerably more likely to allow this context to shift her choice towards touching the glass rather than the isolated cup, in spite of the literal identicality of the two cups. In other words, even if her first ("bottom-up") response might have been to touch the cup on her side of the table, noticing the groups of objects around Henry's cup and her glass, and noticing that this context can be mapped perfectly, affects the initial bottom-up impulse to touch her cup. The probability of her touching her glass is considerably increased by the "top-down" pressures induced by the discovery of the mappings between the groups of silverware surrounding the glass and the cup..

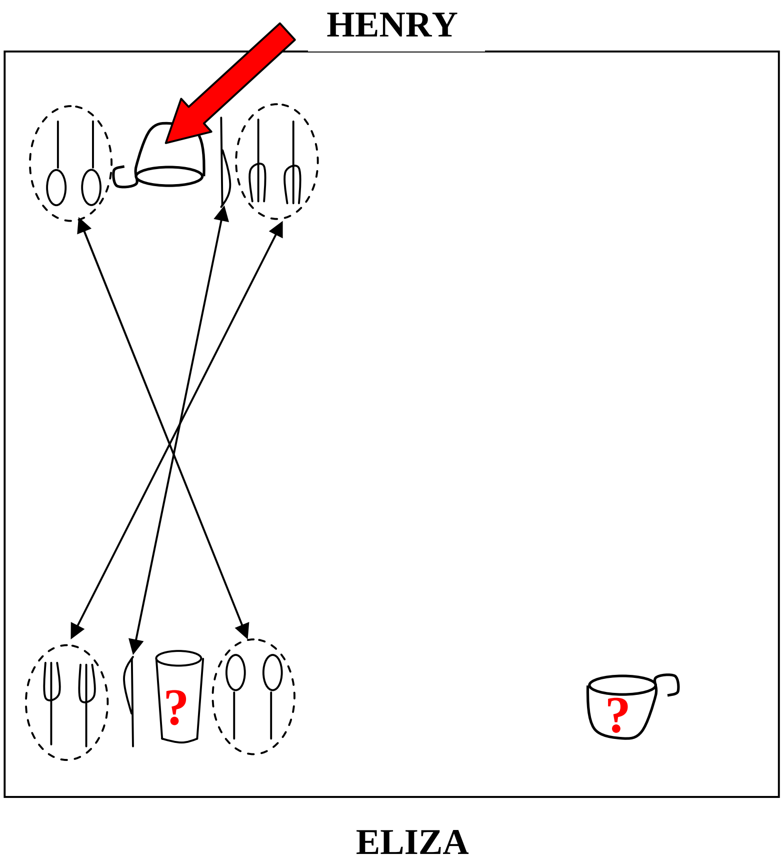


Figure 2. A table configuration where the context provided by the silverware surrounding Henry's cup and Eliza's glass influences Eliza's initial bottom-up "impulse" to respond by simply touching the cup on her side of the table.

How the program finds a reasonable solution

How the program, Tabletop, finds a solution to this particular problem is explained in detail for the interested reader in French (1995), pp 95-111. We will give an overview of the algorithm below.

First, consider the semantic memory (called the “Slipnet”). This network is designed to represent the interconnections of the concepts acquired through a lifetime of interacting with the world. It represents, in some sense, our “top-down” knowledge of the world. The links represent connections between concepts in the world and the semantic distances between them (Shepard, 1962). Most importantly, it is a *malleable*. If the concept “smaller than” becomes active because, for example, we notice that dessert forks are smaller than dinner forks, then we become more sensitive to this relationship for all pairs of concepts for which it applies (e.g., saucers vs. dinner plates, wine glasses vs. water glasses, etc.). See Figure 3.

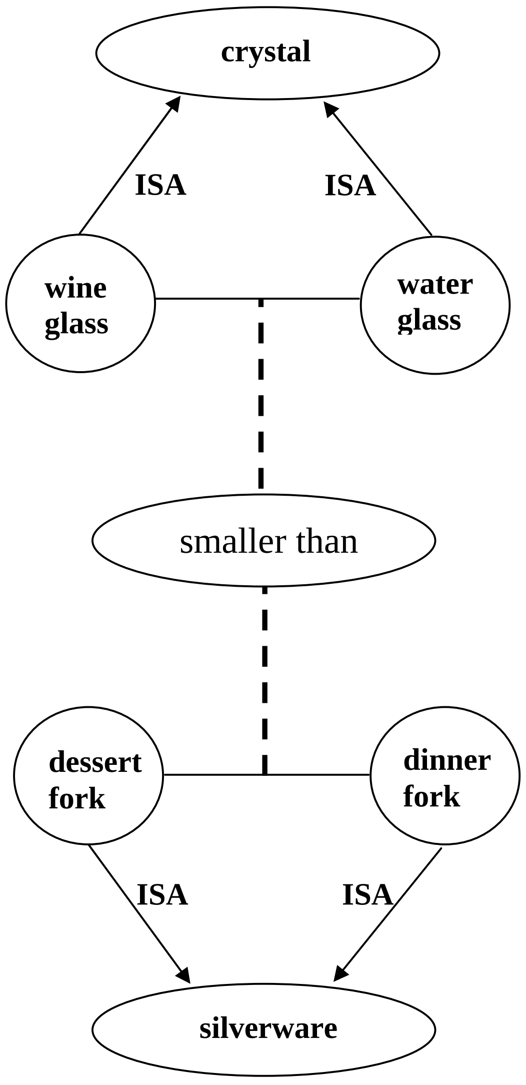


Figure 3. Part of the semantic memory (“Slipnet”) used in solving the Tabletop configuration shown in Figure 2.

Among the concepts in the semantic memory is "group". Concepts in semantic memory correspond, at least approximately, to the agents that will be building structures in Working Memory.

Let's start the process by launching some agent, for example, a "Find Group" agent. It looks for a "group" on the table. It does not see forks, knifes, spoons, cups, etc. It merely looks for clusters of objects, any objects. It finds one (say, the group closest to Henry). This causes several things to happen:

- Activation is sent by the agent to the "group" node in the semantic network. The "group" node in the semantic network becomes more active.
- It spawns (i.e., places on the Coderack) a number of finer-grained agents that will, if they run, look for things that might be inside the just-discovered group. For example, a Find-Group agent that has found a group on the table, would put several Find-End-Object agents on the Coderack, each having as a parameter the Group that has just been found. When one of these Find-End-Object agents runs it will explore an extremity of the Group that was found by its parent Find-Group agent to determine what the objects at the extremity of the group are..

The added activity of the "group" node in the semantic network causes new "Find Group" agents to be put on the Coderack, waiting there to be selected to run. (Here we see the permanent interaction between the bottom-up processes of the agents and the semantic network.) After a while the Find-Group agents will be unable to find any more groups. Agents that happen to be on the Coderack will continue to be selected for a while, but since they won't find any new groups, activation of "group" in the Slipnet will fall. This will mean that no more new Find-Group agents will be put on the Coderack. The activity of exploring the table will now be in the hands of other codelets, some of which have been spawned by the Find-Group agents.

As structures are gradually discovered – for example, the fork-subgroups, the spoon-subgroups, the correspondence between the two groups of two spoons (a "correspondence" being a structure, just like a "group" is a structure), etc. – they are put into the program's Workspace, where they can be accessed by the agents.

The best, most coherent structures discovered to date are put into a special area of the Workspace called the Worldview. But structures are continually competing to get into the Worldview. So, for example, early in the exploration of the table configuration in Figure 2, the structure mapping the two cups onto one another was part of the Worldview. In other words, early on it was considered to be the best mapping of the object touched by Henry to the object that Eliza should touch. But as the program gradually discovered the other structures surrounding Henry's cup and Eliza's glass, the overall coherence of the mapping between Henry's cup and Eliza's glass grew stronger, until (usually, at least) it displaces the cup-cup mapping in the Worldview.

The overall coherence of structures in the Worldview determine the temperature of the program. If there is little coherence, then temperature rises, causing structures in the Workspace to be broken. This allows new agents to try again and gives the program the possibility of discovering new structures from the "rubble" of the broken structures. On the other hand, if the structures in the Worldview are good and coherent, temperature falls. When temperature falls low enough, the program stops and picks as its answer the correspondence in the Worldview containing the touched object.

The issue of scaling up

One of the important claims about this architecture is that, at least theoretically, it should be capable of scaling up. This is because the program, by design, *never examines all possible structures* on its way to an answer. It starts its exploration very broadly and gradually narrows its focus as time goes on, based on what it has already found. This means that many structures, many potential mappings, many potential groupings, etc., never get examined to any depth (or, in some cases, do not get seen at all) as the program moves towards an answer.

So, in the example in Figure 2, if there were, say, thirty groups of objects on the table, instead of two as there are now, the program would, almost certainly, not explore (or potentially, even find) all of them. Why? Because very soon after the discovery of the initial, most salient groups, finer-grained agents would already be exploring these groups. Most probably, if there were a reasonable answer to be found in these groups, it would be found. Only if this exploration didn't lead to an answer in a reasonable time, would temperature rise and the old structures be broken, thus pushing the program to explore in new directions. In other words, the amount of exploration that is done is not proportional to the amount of potential exploration.

The potential long-term impact of the mechanisms presented

We have attempted to explain, by means of a number of simple everyday examples, the flexibility of human analogy-making capabilities. This argues for the importance of programs that develop representations by means of a continual interaction between a knowledge store and working memory. This crucial ability potentially will allow scaled-up versions of these programs to navigate safely between the Scylla of hand-tailored representations designed with the desired mappings in mind and the Charybdis of unfiltered mega-representations that entail a combinatorial explosion of mappings. We believe that the mechanism of context-dependent computational temperature will be a necessary ingredient in programs that will one day be called truly creative. It allows programs to explore — infrequently, but at least occasionally — highly improbable areas of space where wonderfully creative answers to a problem could lie.

Conclusions

I would go so far as to say that analogy-making is, to steal a phrase from the literature on consciousness, The Hard Problem of AI. Not the Impossible Problem, but Very, Very Hard, all the same, and perhaps also The Central Problem. And one that will not be fully solved for a very long time. Unlike chess, unlike optical character recognition, unlike wending your way across a desert with a large database of what to expect, unlike finding someone's fingerprints or face in a huge database of fingerprints or faces, etc. -- all of which, however difficult, can be precisely specified – the mechanisms of analogy-making are much harder to pin down. They involve representing situations so that they can be seen one way in one context, another way in another context. They involve enormous problems of extracting and aligning just the right parts of different representations. And what makes this so hard is that almost anything can, under the right circumstances, be "like" something else. A claw hammer is like a back scratcher. Stiletto heels are like a Bengal tiger. And thinking a high IQ is enough to succeed at research is like thinking that being tall is enough to succeed at professional basketball. And so on, *ad infinitum*. Analogies cover all domains of human thought. We abstract something from an event in one domain and explain in by an analogy to something in a completely different domain. And we do this constantly. It is one of our most powerful means of explaining something new or unusual.

Progress in this area will, no doubt, be slow but will, without question, be one of the key means by which we will move forward in artificial intelligence.

References

Bolland, S. (2005). *FAE: The Fluid Analogies Engine. A Dynamic, Hybrid model of. Perception and Mental Deliberation*, Unpublished Ph.D. dissertation, Computer Science and Electrical Engineering, University of Queensland, Brisbane, Australia.

Bolland, S. & Wiles, J. (2000). Adapting Copycat to Visual Object Recognition. *Proceedings of the Fifth Biennial Australasian Cognitive Science Conference*, Adelaide, Australia.

Bolland, S. (1997). http://www2.psy.uq.edu.au/CogPsych/Copycat/ and http://www2.psy.uq.edu.au/CogPsych/Copycat/Tutorial/

Bonnetain, J.-L. (1988). Personal communication.

Chalmers, D. J., French, R. M., & Hofstadter, D. R. (1992). High-level perception, representation, and analogy: A critique of artificial intelligence methodology. *Journal of Experimental & Theoretical Artificial Intelligence* 4:185--211.

Defays, D. (1990). Numbo: A Study in Cognition and Recognition. The Journal for the Integrated Study of Artificial Intelligence, Cognitive Science and Applied Epistemology. Issues in Connectionism: Part I. Vol. 7, No. 2, pp. 217-243.

Falkenhainer, B., Forbus, K. D. & Gentner, D. (1989). The structure-mapping engine. *Artificial Intelligence, 41*(1), 1-63.

Forbus, K., Gentner, D., Law, K. (1995). MAC/FAC: A Model of Similarity-Based Retrieval. Cognitive Science 19(2): 141-205.

French, R. M. & Kus, E. (2006). KAMA: A Temperature-Driven Model of Mate-Choice using Dynamically Evolving Representations. (under review at *Adaptive Behavior*).

French, R. M. (1995). *The Subtlety of Sameness*, Cambridge, MA: The MIT Press.

French, R. M. (1997). When coffee cups are like old elephants or Why representation modules don’t make sense, *Proceedings of the International Conference New Trends in Cognitive Science,* A. Riegler & M. Peschl (eds.), Austrian Society for Cognitive Science, p. 158-163

French, R. M. (2002). The Computational Modeling of Analogy-making. *Trends in Cognitive Sciences, 6*(5), 200-205.

Gentner, D. (1983). Structure-Mapping: A Theoretical Framework for Analogy, *Cognitive Science*, 7(2), 155-170

Gentner, D., Holyoak, K., & Kokinov, B., eds. (2001) *The Analogical Mind: Perspectives from Cognitive Science.* Cambridge, MA: MIT Press.

Hall, R. (1989). Computational approaches to analogical reasoning: a comparative analysis. *Artificial Intelligence, 39*, 39-120.

Hofstadter, D. R. & Mitchell, M. (1992). An overview of the Copycat project. In Holyoak, K.J. & Barnden, J. (Eds.) *Connectionist Approaches to Analogy, Metaphor, and Case-Based Reasoning* (Norwood, NJ: Ablex).

Hofstadter, D. R. & the Fluid Analogies Research Group (1995). *Fluid Concepts and Creative Analogies.* New York: Basic Books.

Hofstadter, D. R. (1984). *The Copycat project: An experiment in nondeterminism and creative analogies.* AI Memo No. 755, Massachusetts Institute of Technology, Cambridge, MA.

Holyoak K. & Thagard P. (1989). Analogical Mapping by Constraint Satisfaction. *Cognitive Science*, 13, 295-355

Hummel, J. & Holyoak, K. (1997). Distributed Representations of Structure: A Theory of Analogical Access and Mapping. Psychological Review, 104, 427-466.

James, W. (1890). The principles of psychology (Henry Holt & Co).

Kirkpatrick, S., Gelatt Jr., C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science, 220,* 671-680.

Kokinov, B. (1994). A Hybrid Model of Analogical Reasoning. In: Holyoak, K. & Barnden J. (eds.) *Advances in connectionist and neural computation theory, vol. 2, Analogical Connections,* Norwood, NJ.: Ablex

Kokinov, B. and French, R. M. (2003) . Computational Models of Analogy-making. In Nadel, L. (Ed.) *Macmillan Encyclopedia of Cognitive Science. Vol. 1,* pp.113 - 118. London: Nature Publishing Group.

Lakoff, G. (1987). *Women, Fire, and Dangerous Things: What categories reveal about the mind.* Chicago: University of Chicago Press.

Marshall, J. (2002a). METACAT: http://www.cs.pomona.edu/~marshall/metacat/

Marshall, J. (2002b). Metacat: a self-watching cognitive architecture for analogy-making. In W. D. Gray & C. D. Schunn (eds.), *Proceedings of the 24th Annual Conference of the Cognitive Science Society*, pp. 631-636. Mahwah, NJ: Lawrence Erlbaum Associates.

McGraw, G. (1995) Letter Spirit (part one): Emergent High-Level Perception of Letters Using Fluid Concepts. Unpublished Ph.D. dissertation, Indiana University, Bloomington, IN. Available at: <http://www.cogsci.indiana.edu/farg/mcgrawg/thesis.html>.

Mitchell, M. & Hofstadter, D. R. (1990). The emergence of understanding in a computer model of concepts and analogy-making. *Physica D* 42:322--334.

Mitchell, M. (1993). *Analogy-Making as Perception: A Computer Model.* Cambridge: The MIT Press.

Mitchell, M. (2001). Analogy-Making as a Complex Adaptive System. In L. Segel and I. Cohen (editors), Design Principles for the Immune System and Other Distributed Autonomous Systems. New York: Oxford University Press, 335–359.

Shepard, R. N. (1962). The analysis of proximities: Multidimensional scaling with an unknown distance function. I. *Psychometrika 27,* 125-140.