The Metacat Project:
A Self-Watching Model of Analogy-Making

James B. Marshall
Douglas R. Hofstadter

Center for Research on Concepts and Cognition
Department of Computer Science
Indiana University
Bloomington, Indiana 47408 USA
{marshall,dughof}@cogsci.indiana.edu

Keywords: analogy, perception, fluid-concepts, self-watching, explanation

Abstract

This paper presents a broad overview of the Metacat project, an extension of the Copycat computer model of fluid concepts, high-level perception, and analogy-making. Copycat models the complex, subconscious interplay between concepts and perception that gives rise to the flexible human ability to perceive apparently-dissimilar situations as being "the same". A key feature of the architecture is the emergence of statistically-robust, high-level behavior from the interactions of many small, low-level, nondeterministic processing agents. All processing occurs through the collective actions of many agents working in parallel on different aspects of an analogy problem, without any higher-level executive process controlling the course of events. Current work on Metacat is focused on extending the Copycat model in a way that permits it to create much richer representations of the analogies it makes. This involves incorporating a long-term memory into the architecture, along with a "self-watching" ability, so that the program can recognize, remember, and recall important patterns that occur in its own processing as it solves analogy problems. Using this higher-order "meta-level" information, analogies can be compared and contrasted in an insightful way, allowing Metacat to understand and explain its answers in a way that Copycat cannot. Metacat's relationship to other work in AI and cognitive science is also examined, in particular work on case-based reasoning and derivational analogy.

1 Introduction

This paper describes the Metacat project, an extension of the Copycat computer model of fluid concepts, high-level perception, and analogy-making that simulates the complex, subconscious interplay between perception and concepts underlying human creativity [Hofstadter, 1984, Mitchell, 1993, Hofstadter and FARG, 1995]. Copycat operates in an abstract, idealized microworld of analogy problems that exhibits a surprising degree of subtlety and richness. Almost every imaginable analogy problem in this microworld admits a wide range of possible answers, although these answers tend to vary greatly in terms of their plausibility or quality, as judged by humans. For some problems, very high-quality, creative answers can sometimes be discovered, although initially they may not be obvious at all.

The central theme underlying the Copycat architecture is the emergence of statistically robust high-level behavior from the interactions of many small, low-level, nondeterministic processing agents called *codelets*. All processing in Copycat occurs through the collective actions of many codelets working in parallel, at different speeds, on different aspects of an analogy problem, without any higher-level executive process directing the course of events. The stochastic behavior of codelets is dynamically biased by the time-varying pattern of activation in the program's network of concepts it uses to build up an understanding of an analogy problem. In turn, the context-dependent pattern of activity in the concept network is itself an emergent consequence of codelet processing. Thus, Copycat, like connectionist models, lies firmly within the paradigm of emergent computation. At the same time, however, it incorporates many ideas from the more traditional paradigm of symbolic AI, staking out a kind of middle ground between these two opposites. Some further perspectives on the emergent–symbolic spectrum and Copycat's relationship to it can be found in [Blank et al., 1992].

For many problems, Copycat's behavior agrees well with human behavior, in terms of the range and frequencies of the answers it finds, as well as its ratings of their quality. But a serious limitation of the model is that it provides only a very crude numerical measure of answer quality. The program, unlike a human, has essentially no understanding of *why* an answer is good or bad, or what is interesting about it, or how it is similar to or different from another answer. Current work on Metacat is aimed at extending the original Copycat model in a way that allows it to create much richer representations of the analogies it makes. This involves incorporating a memory into the model, along with a "self-watching" ability, so that the program can recognize, remember, and recall important patterns that occur in its own processing as it solves an analogy problem. Using this higher-order "meta-level" information about answers, gleaned from self-watching, analogies can be compared and contrasted in an insightful way. The long-term goal of the current research is to computationally model how meaningful, high-level *understanding* can emerge dynamically from a stochastic, fluid, low-level "subcognitive" substrate.

The paper first outlines Copycat's central motivating ideas and its main architectural components, which constitute the foundation upon which Metacat builds. The crucial ideas behind Metacat are then introduced by way of an extended example to illustrate them more concretely. This leads to a discussion of Metacat's relationship to other work in artificial intelligence and cognitive science, in particular to work on case-based reasoning and derivational analogy.

2 Copycat's Underlying Philosophy

The motivation driving the Copycat project, and thus the Metacat project, is the belief that the ultimate source of the human mind's remarkable flexibility and power lies in the *dynamic, fluid nature of its concepts.* A typical human mind can recognize and understand a huge number of distinct concepts, which get activated in certain situations but not in others. The activation of a particular set of abstract concepts, in response to various low-level sensory stimuli impinging from the environment, constitutes an act of *perception* or understanding of the given situation in terms of these concepts. This type of perception can occur at extremely high levels of abstraction, such as when a person hears a piece of music for the first time and recognizes it as belonging to a particular composer or musical style. Indeed, examples of such high-level perception are ubiquitous in cognition.

A great many concepts are implicitly present simultaneously, to different degrees, in the "halo" surrounding the core of every concept in the mind. Under the right contextual pressures, these blurry conceptual boundaries can stretch and conform to the situation at hand. Consider, for example, a typical American adult's concept of *Vietnam.* On the surface, this concept refers to a particular nation located in Southeast Asia. But in the average American mind at least, it is also tightly associated with the huge political and military debacle in which the United States got mired in the late 1960's and early 1970's, along with all of the internal social and cultural strife that occurred as a result. This entire complex, political-historical situation is conventionally referred to in America as simply "Vietnam". Hovering about the central core of this concept are enormous numbers of other related concepts, some closer in, some farther out—concepts such as *communism*, *President Nixon*, *war*, *social unrest*, *the Pentagon*, *rice*, *1968*, *dominos* (in the sense of a chain of dominos falling one by one to communism), *failure*, and so on. Other concepts clearly lie very far away from the core: *penguins*, say, or *rollerskates*, or *computer software.* Or do they? Consider the following humorous quip seen recently on the World Wide Web [Leake, 1996]: "Windows-95: Microsoft's Vietnam?" Regardless of whether or not one agrees with its sentiment, its meaning is readily understood. Windows-95, a computer operating system, can be perceived as an instance of the concept *Vietnam.* In doing so, an implicit analogy has been made between two complex and apparently distinct entities, mapping, among other things, the United States onto Microsoft and, perhaps, ex-U.S. President Richard Nixon onto Microsoft Chairman Bill Gates. It is the complex, fluid nature of concepts in the mind that allows such analogies to be effortlessly perceived and understood.

Copycat's central aim is to take concepts seriously. It is our belief that the task of modeling the nature of concepts themselves has unfortunately been neglected all too often by researchers in cognitive modeling. We feel that most computational models currently claiming to operate in so-called "real-world" domains have essentially no deep understanding of the concepts they purport to deal with. In contrast, our approach is to start with a simple, yet rich, microdomain in which interesting cognitive phenomena can be carefully studied, and to make a real effort to model the dynamic, fluid properties of the concepts that give rise to the capacity for very flexible, abstract perception and analogy-making exhibited by humans on problems taken from this domain. The apparent simplicity of Copycat's domain, however, is deceptive, for it turns out to be an extremely difficult task to build a computer program capable of exhibiting a level of flexibility and creativity comparable to human behavior even in this tiny, restricted domain. We believe that realizing such a program would go a long way toward a genuine understanding of the powerful yet subtle conceptual machinery underlying human cognition.

3 A Sketch of Copycat's Architecture

A detailed exposition of the Copycat model can be found in [Mitchell, 1993] and [Hofstadter and FARG, 1995]. Here we shall give just a brief overview. Copycat perceives analogies between short strings of letters, which can be thought of as representing abstract, idealized situations. An example of such an analogy might be "If **abc** changes to **abd**, how does **iijjkk** change in an analogous way?" Despite its apparent simplicity, the letter-string domain actually exhibits a remarkable degree of subtlety, and constitutes an ideal laboratory in which to study analogy-making and high-level perception. An interesting feature of such problems is that there is no single "right" answer; rather, a range of answers is always possible for each problem. For the previous example, some possible answers might be **iijjll**, **iijjdd**, **iijjkl**, **iijjkd**, **iijjkk**, **iikjkk**, **iidjkk**, or even **abd** or **aabbdd**. For most analogy problems, some answers are consistently judged by people to be better than others. But a surprisingly large number of defensible answers can usually be found for each problem. Furthermore, for some problems, the answers judged to be the "best" are not at all the most obvious ones. Figure 1 shows Copycat's typical behavior on the above analogy problem. An answer's *frequency* can be interpreted as a measure of its "obviousness" to the program, while an answer's *quality* is measured by its associated "temperature" value, about which more will be said below.

3.1 Perceptual activity in the Workspace

In order to discover an answer to an analogy problem such as "**abc** ⇒ **abd**; **iijjkk** ⇒ **?**", codelets work together to build up a strong, coherent mapping between the *initial string* **abc** and the *target string* **iijjkk**, and also between the initial string and the *modified string* **abd**. These strings reside in the *Workspace* component of Copycat's architecture, which is where all perceptual activity takes place. Codelets also build hierarchical groups within strings, which serve to organize the raw perceptual data (the letters) into a coherent, chunked whole. For example, in the string **iijjkk**, codelets might build three "sameness-groups" **ii**, **jj**, and **kk**, and then a higher-level "successorship-group" comprised of these three groups encompassing the entire string. The distributed nature of codelet processing interleaves the chunking process with the mapping process, and as a result, each process influences and drives the other.

A mapping consists of a set of *bridges* between corresponding letters or groups that play respectively similar roles in different strings. Each bridge is supported by a set of *concept-mappings* that together provide justification for perceiving the objects connected by the bridge as corresponding to one another. For example, a bridge might be built between **a** in **abc** and the group **ii** in **iijjkk**, supported by the concept-mappings *leftmost* ⇒ *leftmost* and *letter* ⇒ *group*, which represent the idea that both objects are leftmost in their strings, and one is a letter and the other a group. Non-identity concept-mappings such as *letter* ⇒ *group* are called *slippages*, and form the basis of Copycat's ability to flexibly perceive superficially-dissimilar situations as being in fact "the same" at some appropriate level of description.

In addition to bridges and groups, another type of structure is necessary to produce an answer to an analogy problem. Once a mapping has been built between the initial string and the modified string (*i.e.*, between **abc** and **abd**), a *rule* based on this mapping must be created that succinctly captures the way in which the initial string changes. Of course, there are many possible ways of describing this change, some more abstract than others. For example, two possible rules for **abc** ⇒ **abd** are *Change letter-category of rightmost letter to successor* and *Change letter-category of rightmost letter to d.* Different ways of looking at the initial/modified change, combined with different ways of building the initial/target mapping, give rise to different answers. A bridge from the letter **c** in **abc** to the group **kk** in **iijjkk**, based on a *letter* ⇒ *group* slippage, may yield the answer **iijjll** or **iijjdd**, depending on the rule used to describe **abc** ⇒ **abd**. On the other hand, a bridge from **c** to the rightmost letter **k** in **iijjkk** may instead yield **iijjkl** or **iijjkd** as an answer, again depending on the rule. To produce an answer, the slippages underlying the mapping between **abc** and **iijjkk** are used by codelets to "translate" the rule describing **abc** ⇒ **abd** into a new rule that applies to **iijjkk**, such as *Change letter-category of rightmost group to successor.* Figure 2 shows a screen dump of the Workspace after the program has found the answer **iijjll**, illustrating the various Workspace structures involved in producing the answer.

3.2 Conceptual activity in the Slipnet

Actively influencing the perceptual activity occurring in the Workspace are the concepts that the program understands, which reside in the "long-term memory" component of Copycat, called the *Slipnet.* Most perceptual structures in the Workspace are, in fact, *instances* of particular Slipnet concepts. The Slipnet serves as the program's permanent repository of knowledge about its domain. It contains representations for various concepts relevant to solving letter-string analogies, such as *successor*, *predecessor*, the idea of *opposite*, the letters **a**, **b**, **c**, and so on, as well as an assessment of each concept's intrinsic degree of abstractness, called its *conceptual depth*. It also contains information about how these concepts relate to each other, including the inherent "associative distances" between them, which determine the propensities for various conceptual slippages to occur. A conceptual slippage between a pair of concepts occurs whenever two Workspace instances of the concepts are seen as playing identical roles in different contexts. For example, in order for codelets to build a bridge between the rightmost letter **c** in **abc** and the rightmost **kk** group in **iijjkk**, a *letter* ⇒ *group* slippage must occur. The probability of this happening is governed by the distance between the *letter* and *group* concepts in the Slipnet.

Although the Slipnet contains permanent information, it is not a static structure. Over the course of a run, influenced by the perceptual activity occurring in the Workspace, concepts in the Slipnet assume different levels of activation; as this happens, distances between concepts grow and shrink, changing the propensities for various slippages to occur. Conceptual activity in the Slipnet thus influences, and is influenced by, perceptual activity in the Workspace; together this results in a tightly-coupled feedback loop between these two architectural components.

In some ways, the Slipnet is similar to a traditional semantic network, in that it consists of a set of nodes connected by links. Each of these links has an intrinsic *length* that reflects the degree of association between the linked nodes, with shorter links connecting strongly associated nodes and longer links connecting weakly associated nodes. Each node corresponds to an individual concept, or rather, to the *core* of an individual concept. A concept is more properly thought of as being represented by a diffuse region in the Slipnet centered on a single node. Nodes connected to the core node by links are included in the central node's "conceptual halo" as a probabilistic function of the link lengths. This allows single nodes to be shared among several different concepts at once, depending on the links involved. Thus, concepts in the Slipnet are not sharply defined; rather, they are inherently blurry, and can overlap to varying degrees.

The model incorporates a simple numerical measure of overall Workspace "coherence", called *temperature*, which reflects, at any given moment, the amount and quality of structures built so far. Temperature values range from 0–100, with lower values indicating a higher degree of Workspace organization. Temperature is computed as a numerical function of the *strength* values of all of the structures existing in the Workspace, weighted by each structure's relative importance. A structure's strength value represents an estimate of how well the structure "fits into" the current set of mappings in the Workspace, and is itself a function of the activation levels of concepts in the Slipnet. For example, if the *group* concept in the Slipnet is highly activated, a bridge between **a** in **abc** and the leftmost letter **i** in **iijjkk** will tend to have a lower strength than a bridge between **c** and **kk**, because the active *group* concept encourages the incorporation of groups rather than letters into the mapping between **abc** and **iijjkk**. As more bridges are built involving groups (rather than letters) in **iijjkk**, the average strength of structures in the Workspace goes up, causing the overall Workspace temperature to decrease.

The final temperature of the Workspace when an answer is found can be interpreted as an indication of the answer's overall quality. Thus, high-quality answers are associated with low temperature, and low-quality answers are associated with high temperature. This measure of answer quality agrees well with the relative judgments of answer quality given by people for a wide range of Copycat problems. For example, Copycat's rating of the answer **iijjll** in the above problem will tend to be much higher than for "weirder" answers such as **iijjkd**.

3.3 Copycat's weaknesses

Unfortunately, such a simple numerical measure is extremely crude, and reflects a fundamental weakness of Copycat: its almost complete lack of any in-depth understanding of the answers it finds. Copycat is unable to explain why it considers particular answers to be good or bad. The reason is that Copycat's processing mechanisms focus almost exclusively on perceiving patterns and relationships in the perceptual data (the letter strings), while ignoring patterns that occur in its own processing when solving an analogy problem. Thus, although it may discover an insightful answer for some problem, it lacks any internal representation or knowledge of the underlying process that led it to discover that answer—knowledge that could provide a basis for explaining the answer's relative strengths or weaknesses, thereby permitting a much richer assessment of its quality. Copycat's lack of any such "self-watching" ability stands in marked contrast to people, who are typically able to give an account of why they consider one answer to be better or worse than another for a particular analogy problem.

For example, an interesting psychological phenomenon related to self-watching, dubbed the *self-explanation effect*, has been described and studied in the context of students learning to solve physics problems from examples [Chi et al., 1989, VanLehn et al., 1992]. In this series of studies, students monitored their own understanding or misunderstanding as they studied worked-out textbook examples of mechanics problems, generating verbal explanations of the example solutions in the process. Those students who learned most effectively from the examples were consistently able to generate more detailed and in-depth explanations of their understanding, demonstrating a greater capacity for accurate monitoring of their own cognitive processes, which in turn reduced their reliance on worked-out examples in solving subsequent problems. These studies clearly illustrate the ability of people to pay attention to patterns in their own thinking.

Another weakness of Copycat is the fact that answers are not retained after they are found. When the program discovers an answer to a problem, it simply reports its answer, along with the answer's final temperature, and then stops. On subsequent runs of the same problem, no recollection of previous answers is possible, so there is no way for the program to bring its past experience to bear on the current situation. This makes comparison of different answers impossible, either within a single analogy problem or among different problems. Furthermore, any type of learning that might occur over multiple runs is impossible. However, learning *per se* was never intended to be a central focus of the project, since the notion of learning to make "better" Copycat analogies is not entirely clear.

4 From Copycat to Metacat

The central objective of the Metacat project is to imbue the program with a rich, high-level understanding of the answers it finds—rich enough to allow it to give at least a limited explanation of an answer's strengths and weaknesses relative to other answers that have previously been found. This requires not only remembering answers, but also the ability to compare and contrast different answers to a single problem, or even answers to different problems, based on an in-depth understanding of the important ideas behind each answer. In short, it involves the ability to perceive and remember the *essence* of each answer found.

In contrast to Copycat, Metacat is able to find and remember many different answers during a single run on a given problem. Whenever it finds a new answer, rather than simply stopping, it pauses to display the answer, along with the answer's supporting groups, bridges, concept-mappings, and other Workspace structures. Furthermore, all of this answer-specific information, including the strings themselves, is then packaged together and stored in memory, after which the program continues searching for alternative answers to the problem. Gradually, over time, a repertoire of answers builds up in memory, each one containing much more information than just the answer string itself. Each stored answer represents a different way of interpreting or making sense of a particular analogy problem. The extra information stored with each answer in memory serves as the basis for comparing and contrasting the answers with each other.

4.1 Comparing and contrasting answers

As a simple example of comparing and contrasting analogies, consider the problem "**abc** ⇒ **abd**; **xyz** ⇒ **?**". This problem has been discussed at length elsewhere [Mitchell, 1993, Hofstadter and FARG, 1995], so we summarize briefly here. In Copycat's letter-string domain, **a** has no predecessor and **z** has no successor. The alphabet is explicitly designed not to "wrap around" from **z** back to **a**, so a straightforward answer based on taking the successor of **z** in **xyz** is impossible. One is forced to adopt a different strategy as a result of this constraint. One way out is simply the literal-minded answer **xyd**. On the other hand, if the symmetry between the "opposite" letters **a** and **z** is noticed, then the answer **wyz** suggests itself, based on seeing **abc** and **xyz** as starting at opposite ends of the alphabet, with **abc** going to the right based on successorship and **xyz** going to the left based on predecessorship. This answer is very elegant, and many people see it as being strongly analogous to **abd**, even though it is not at all obvious at first.

Copycat's average temperature for **xyd** is around 22, and around 14 for **wyz**. People typically judge **wyz** to be of higher quality than **xyd**, so Copycat's judgement of the relative strengths of these two answers is consistent with human psychological behavior. But here the psychological plausibility ends. Unlike people, Copycat cannot explain *why* it thinks **wyz** is better than **xyd**. It has no insight into how it arrives at either answer, and no appreciation of how they differ, beyond just a single number. People, however, can quite easily see that the **wyz** answer depends critically on the idea of oppositeness, or symmetry: the alphabetic-position symmetry of the letters **a** and **z**, and the directional and relational symmetry of the strings **abc** and **xyz**, which go in opposite directions based on the complementary ideas of successorship and predecessorship. The **xyd** answer, on the other hand, is based on the idea of directional and relational sameness: both **abc** and **xyz** are strings based on successorship (or possibly predecessorship) going in the same direction. The presence or absence of the idea of symmetry is clearly of critical importance in differentiating the two answers **xyd** and **wyz**. People can easily point out this distinction, but Copycat, unable to notice it, remains in the dark.

People are also quite good at recognizing when different answers to different Copycat problems are somehow similar. Often this similarity is noticed spontaneously, seemingly without any conscious effort involved. Furthermore, once the similarity has been noticed, people are then able to say why the answers seem similar, as well as how they differ. For example, consider the problem "**rst** ⇒ **rsu**; **xyz** ⇒ **?**". One possible answer results from viewing **rst** ⇒ **rsu** as changing the rightmost letter **t** to its successor, attempting to do the same thing to **xyz**, failing, and then falling back on seeing **rst** ⇒ **rsu** in a more literal-minded way as a change to the letter **u**, resulting in the answer **xyu**. This answer is rather similar to the **xyd** answer in the previous problem.

Or consider the answer **wyz**, gotten by seeing **rst** and **xyz** as strings going in the opposite direction, one based on successorship and the other on predecessorship. This way of interpreting things is strongly reminiscent of the **wyz** answer in the previous problem. Indeed, the two answers themselves are exactly the same. However, there is an important difference. In "**rst** ⇒ **rsu**; **xyz** ⇒ **?**", there is no alphabetic-position symmetry between **r** and **z**, so there is not as much justification in this problem for seeing **rst** and **xyz** as mirror images of each other as there was in the case of **abc** and **xyz**. The presence or absence of alphabetic-position symmetry is the crucial difference between the two **wyz** answers. Everything else about the answers is the same: both depend on failing to take the successor of **z**, as well as on noticing successorship–predecessorship symmetry, directional symmetry, and seeing the rightmost-letter change abstractly in terms of successorship, rather than literally as a change to a specific letter of the alphabet. The relative lack of justification for **wyz** in the second problem tends to diminish its overall quality. While arguably better than **xyu**, **wyz** is not nearly as superior to **xyu** as was **wyz** to **xyd** in the first problem. In short, **xyd** and **xyu** play essentially *identical* roles in their respective problems, and are thus of comparable quality, while the two **wyz** answers are quite *different*, even though on the surface they appear to be identical.

4.2 Modeling the cognitive level versus the subcognitive level

The reason that Copycat is unable to recognize what makes the **wyz** answers different from each other, or what makes **xyd** and **xyu** essentially the same, is that, although it uses many ideas *implicitly*, it does not build any *explicit* representations of the central ideas underlying the answers it finds, and thus it has nothing on which to base a detailed comparison of its answers. The answers' associated final temperature values simply do not provide enough information.

When Copycat discovers an answer, of course, many Workspace structures exist that, to a certain extent, characterize the answer just found. In the case of the first **wyz** answer, for instance, an abstract rule *Change letter-category of rightmost letter to successor* exists; bridges exist mapping the left and right letters of **abc** onto the right and left letters, respectively, of **xyz**; a bridge exists mapping the right-directed group **abc** onto the left-directed group **xyz**; and slippages involving the concept of *opposite*, such as *right* ⇒ *left* or *successor* ⇒ *predecessor*, support these bridges. But structures may also exist that have very little bearing on the crux of the matter, such as a bridge between **b** in **abc** and **y** in **xyz**, or the concept-mappings *letter* ⇒ *letter* or *group* ⇒ *group*.

The configuration of structures in the Workspace collectively represents the way in which a given analogy problem is *interpreted*; that is, the way in which the strings are perceived in relation to one another. A particular interpretation implies a particular answer for the problem. But which aspects of that interpretation are essential and which are irrelevant remain buried implicitly in the overall interpretation. Copycat lacks a way to identify and explicitly represent those aspects of the interpretation that are most important in characterizing the final answer. This amounts to abstracting out an answer's core essence from the many Workspace structures giving rise to it, and then building an explicit, concrete representation of this abstract essence.[1]

Copycat's lack of such an ability, however, is understandable, since it was intended first and foremost to be a model of the *subcognitive* mechanisms underlying cognition. All of the nondeterministic codelet activity occurring in the Workspace—the building of bridges and groups, the making of slippages, and so on—is intended to represent perceptual activity carried out at the subcognitive level, below the level of "conscious awareness". In principle, activity occurring at the subcognitive level is largely inaccessible to the cognitive level, at least in all of its exhausting,

[1] Actually, Copycat's final temperature measure can be regarded as a very crude attempt at abstracting out a high-level characterization of an answer from the answer's associated Workspace structures. This "high-level" characterization, however, is extremely weak, because no *ideas* pertaining to the answer are represented.

fine-grained detail. In Copycat, no attempt was made to model the cognitive level at all, so it is hardly surprising that the program has no insight into the answers generated by its subcognitive mechanisms.

The focus of Metacat is on developing mechanisms that support a higher cognitive level on top of the existing subcognitive level. To do this, Metacat needs to be able to watch what happens while its subcognitive mechanisms are building, destroying, and reconfiguring Workspace structures in pursuit of an answer to the problem at hand, and to build explicit representations of this lower-level, subcognitive perceptual activity. When it finds an answer, these representations will constitute a temporal trace of the subcognitive activity that led to the answer. A characterization of the answer can be formed by abstracting a high-level description of this temporal trace. This high-level description can then be stored in memory along with the answer itself.

4.3 Sketch of a cognitive-level characterization

In Metacat, the most important type of explanatory information associated with answers in memory consists of structures called *themes.* These structures, which get created as the program works on an analogy problem, explicitly represent important concepts that arise in building the mappings between the Workspace strings. Themes reside in Metacat's *Themespace*, which can be thought of as a special region of the Workspace. They are comprised of Slipnet concepts, and can themselves take on various levels of activation, according to the extent to which the ideas they represent are present or absent in the Workspace's configuration of structures. *Bridge themes* get created whenever a new bridge is built between two Workspace structures, based on the concept-mappings underlying the bridge. For example, in the problem "**abc** ⇒ **abd**; **xyz** ⇒ **?**", if an **a**–**z** bridge is built between the strings **abc** and **xyz**, supported by the concept-mappings *first* ⇒ *last*, *leftmost* ⇒ *rightmost*, and *letter* ⇒ *letter*, three bridge themes get created and added to the Themespace: *AlphaPos:opposite*, representing the idea of alphabetic-position symmetry; *Direction:opposite*, representing the idea of left/right symmetry; and *ObjectType:same*, representing the idea of mapping objects of the same type (*i.e.*, letters) onto each other. Other types of themes may get created as a result of other Workspace structures being built, such as *rule themes* which characterize the degree of abstractness of rules.

A newly-created theme starts with zero activation, but receives periodic boosts of activation from Workspace structures compatible with the idea represented by the theme. The more a particular idea serves as an organizing principle in the Workspace, the more active the themes associated with that idea will become. Active themes are intended to represent the awareness of particular aspects of an analogy-problem at the *cognitive* level, as opposed to the subcognitive level. Thus, in the example above, an active *AlphaPos:opposite* theme represents an explicit awareness of alphabetic-position symmetry as a key idea underlying the "mirror-image" interpretation of **abc** and **xyz**. The idea of alphabetic-position symmetry is present *implicitly* in the Workspace structures making up the **abc**–**xyz** mapping, but the active *AlphaPos:opposite* theme makes this idea explicit. Themes are thus higher-order perceptual structures. Whereas Copycat's structures (bridges, groups, etc.) represent an interpretation of its letter strings, Metacat's themes represent an interpretation of this interpretation, in the sense that different patterns of active themes emphasize or deemphasize different aspects of a configuration of structures in the Workspace.

In addition to storing the answers it finds in its memory, Metacat maintains, seperately, a temporal record (called the *Trace*) of the important *events* that occur during processing while it works on some analogy problem. Such events include the explicit recognition of themes important to the problem, which may occur when a theme attains a sufficiently high level of activation. Another type of important event, of course, is the actual discovery of a new answer. In the latter case, the themes most active at the time an answer is found represent the answer's key ideas, and the events recorded in the Trace prior to the discovery of the answer serve as a record of how the answer was found. This information is stored, along with the answer's other supporting Workspace structures, in Metacat's memory, forming a high-level characterization of the answer. In some ways, this idea is similar in flavor to work on derivational analogy [Carbonell, 1986, Veloso, 1994], in which a system stores temporal traces of a problem-solving session in memory for use in analogous situations that may arise later, so as to improve the system's level of performance. As mentioned earlier, however, the focus in Metacat is not on learning to make "better" analogies, or to make them more "efficiently", but rather on being able to explain why one analogy is judged to be more compelling than another.

Returning to the example discussed earlier, as Metacat works on the problems "**abc** ⇒ **abd**; **xyz** ⇒ **?**" or "**rst** ⇒ **rsu**; **xyz** ⇒ **?**", it notices interesting events that happen along the way—the activation of the concept of *opposite*, for instance, or the recognition of the symmetric relationship between **a** and **z**. As outlined above, whenever it finds a new answer, it stores a representation of the answer in memory, incorporating into this representation the most important themes involved in the discovery of the answer. Figure 3 shows a schematic representation of the contents of Metacat's memory after having found the four answers described earlier.

Each of the four memory structures in the figure represents a seperate analogy between letter strings. The stored information includes the analogy problem itself, the answer found, and the themes that led the program to discover the answer. The first two structures represent the "literal-minded" answers **xyd** and **xyu**, each one based on the theme of mapping the initial and target strings onto each other in the same direction (*i.e.*, left-to-right) and seeing the change from the initial string to the modified string in a literal manner (*i.e.*, *Change letter-category of rightmost letter to d* or *Change letter-category of rightmost letter to u*). The third structure represents the answer **wyz** for the problem "**abc** ⇒ **abd**; **xyz** ⇒ **?**". The themes underlying this answer include the theme of alphabetic-position oppositeness (*i.e.*, the symmetry between the alphabetic-first letter **a** and the alphabetic-last letter **z**), the theme of directional oppositeness (*i.e.*, the symmetry between the right-directed string **abc** and the left-directed string **xyz**), and seeing the **abc** ⇒ **abd** change in an abstract way (*i.e.*, *Change letter-category of rightmost letter to successor*). The fourth structure, representing the answer **wyz** for the problem "**rst** ⇒ **rsu**; **xyz** ⇒ **?**", is similar, except that in this case, the theme representing alphabetic-position oppositeness is absent, since in this problem no *first* ⇒ *last* slippage between **r** and **z** is possible.

Given these four answer-representations in memory, the stage is set for comparing and contrasting them based on the thematic information they contain. Clearly, the important difference between the two **wyz** answers is the absence of the alphabetic-position oppositeness theme in the **rst** case. Herein lies the difference in quality of **wyz** as an answer to these two problems. Furthermore, the thematic characterizations of the literal-minded answers **xyd** and **xyu** are identical, reflecting the absence of qualitative differences between them. In fact, it is even possible to see a kind of "meta-level" analogy between the weak answers **xyd** and **xyu** that is qualitatively much stronger than the corresponding analogy between the two **wyz** answers, even though in the latter case the answers are identical, which would at first glance seem to suggest that they could not be more analogous.

4.4 Recognizing repetitive behavior

When it first tries to solve the problem "**abc** ⇒ **abd**; **xyz** ⇒ **?**", Copycat almost invariably perceives **abc** and **xyz** as going in the same direction. This is certainly a reasonable predisposition. However, such an interpretation of the situation leads inevitably to an attempt to take the successor of **z**, since under this interpretation **c** and **z** are seen as corresponding. This attempt fails, and Copycat "hits a snag". It is forced to reinterpret the situation. Often it circumvents this difficulty by changing the rule from *Change letter-category of rightmost letter to successor* to *Change letter-category of rightmost letter to d*, leaving intact the same-direction mapping between **abc** and **xyz**, which then yields the answer **xyd**. More rarely, the mapping itself is broken and eventually replaced by a mapping based on the opposite **a**–**z** symmetry, yielding **wyz**. Unfortunately, after breaking the relevant structures, it tends to rebuild exactly those structures that led it to the snag in the first place, often going round and round in circles hitting the snag over and over, until it finally manages to stumble on some interpretation that actually yields an answer. In fact, Copycat hits the snag an average of nine times per run on this problem—sometimes even as often as twenty or thirty times on certain runs. This is quite unlike typical human behavior. People tend to "get the message" after attempting some unsuccessful strategy a few times. They are able to recognize that their strategy isn't working and that they should try something different.

Themes offer a way to address this problem. In addition to storing the answers it successfully finds, Metacat can take advantage of its ability to watch and remember events in its own processing by storing its failures as well. Snags represent one type of possible failure; another is the case when the program has simply ceased to make any progress in its attempt to understand a situation (*i.e.*, to build up a sufficiently coherent mapping). When it recognizes a failure situation, it can store information in its Trace about the themes that are currently active. Later, it can use this information to recognize when it is approaching a situation similar to the one in which it previously failed, by comparing the themes that are currently active with those that were active at the time of failure. By strongly biasing subsequent Workspace activity away from building the types of structures associated with previous failures, themes can both characterize and actively guide Metacat's processing, allowing the program first to detect and then to avoid the kind of senseless looping behavior so problematic in Copycat.

5 Relation to Other Work

As the previous discussion may suggest, Metacat touches on many of the issues underlying research in case-based reasoning (CBR) [Leake, 1996]. Metacat's memory may be thought of as storing "cases", representing either successes (*i.e.*, the discovery of a new answer) or failures (*i.e.*, hitting a snag). These cases form a corpus of experience on which the program can draw when faced with new situations. When it finds a new answer, its previous experiences may remind it of similar problems it has seen in the past, allowing it to compare and contrast the answers based on the thematic information stored with them. It can avoid running into the same troubles over and over again by using its stored failure experiences to help it recognize and avoid unproductive pathways it has explored before.

However, there are important differences between CBR and Metacat. First of all, even though Metacat is concerned with solving analogy problems, it is not intended to model problem-solving *per se*. Rather, its focus is on modeling the way fluid concepts allow analogies between different situations to be perceived in a natural and psychologically plausible manner. It is concerned with analogical *perception*, not analogical reasoning employed as a method of solving problems (as in CBR). Furthermore, much CBR work focuses on systems that learn to solve problems in a progressively faster and more efficient manner, whereas in Metacat the notion of learning to perceive analogies with ever increasing efficiency and speed is of no concern.

Metacat is actually closer to work on derivational analogy than to ordinary case-based approaches that store only a final problem solution. In derivational analogy, an entire trace of a problem-solving session is stored for future reference, not just the solution produced in the end, along with a series of annotations describing the conditions under which each step in the solution was taken [Carbonell, 1986, Veloso, 1994]. In Metacat, the thematic information stored with an answer summarizes the important concepts and events that together contributed to the discovery of the answer, much like the temporal problem-solving trace of derivational analogy.

In contrast to derivational analogy, however, the objective of Copycat is to explore how an understanding of a situation can be achieved via fluid concepts in its circumscribed domain, and how this understanding gives rise, as an automatic by-product, to a flexible and robust ability to see analogies between apparently disparate situations. Copycat's concepts, to be sure, are but a crude approximation to the true power and flexibility of human concepts. Still, there is a sense in which its concepts really are active, semantic entities within its tiny, idealized world—not just empty static symbols shunted around by the program. A concept node in the Slipnet—*successor-group*, for example—responds to the situation at hand in a continuous, context-dependent way. That is, the activation level of the node reflects the degree of perceived relevance or presence of the concept of successor-group in the Workspace at any given moment. A wide range of superficially dissimilar strings can in principle activate it—strings such as **abc**, **ijk**, **pqrstu**, **iijjkk**, **mrrjjj**, **mmrrrjjjj**, and **aababcabcd**. Given the program's ability to flexibly recognize a wide range of instances of the same concept, some of them quite abstract, we believe it is fair to say that the program's concepts have at least *some* small degree of meaningfulness, of genuine semantics, within the confines of its domain.

Metacat's objective is to deepen Copycat's understanding of its answers by incorporating the ideas of memory, reminding, and self-watching. Many deep ideas from case-based reasoning appear to be relevant to this aim, such as the storing of experience as a repertoire of cases in memory, the activation of stored cases by similar current situations, and the issue of analogical similarity of different situations. Unfortunately, case-based reasoning research concentrates on these issues at the expense of understanding the nature of concepts. Metacat can be seen as an attempt to broaden and enrich these important ideas by focusing on them in the context of dynamic, fluid, context-sensitive concepts. We worry that CBR's ultimate success, at least as a cognitive model, will necessarily be limited by its avoidance of this very difficult but critically important issue.

6 Conclusion

Enriching Metacat's understanding of its answers by incorporating higher-order thematic information gleaned from self-watching enables it to perceive abstract similarities and differences among the analogies it makes, effectively permitting analogies to be made between analogies. By applying the same processing mechanisms that it uses to perceive relationships in its perceptual input to the more abstract task of perceiving relationships among the answers that it finds, it is able to compare and contrast answers in a much more interesting way than is possible in Copycat. Endowing Copycat with a sophisticated self-watching capability is the central objective of present efforts to extend and refine the model, and is a logical next step along the road to understanding and capturing the full richness of high-level perception and analogy-making in a computational framework.

7 Acknowledgements

This research was supported in part by Sun Microsystems Co. Academic Equipment Grant #EDUD-NAFO-960418 and by grants to the Center for Research on Concepts and Cognition from the College of Arts and Sciences of Indiana University.

References

[Blank et al., 1992] Blank, D., Meeden, L., and Marshall, J. (1992). Exploring the symbolic/subsymbolic continuum: A case study of RAAM. In Dinsmore, J., editor, *The Symbolic and Connectionist Paradigms: Closing the Gap*, pages 113–148. Lawrence Erlbaum Associates, Hillsdale, NJ.

[Carbonell, 1986] Carbonell, J. (1986). Derivational analogy: A theory of reconstructive problem solving and expertise acquisition. In Michalski, R., Carbonell, J., and Mitchell, T., editors, *Machine Learning: An Artificial Intelligence Approach, Volume II*, pages 371–392. Morgan Kaufmann, Palo Alto, CA.

[Chi et al., 1989] Chi, M., Bassok, M., Lewis, M., Reimann, P., and Glaser, R. (1989). Self-explanations: How students study and use examples in learning to solve problems. *Cognitive Science*, 13:145–182.

[Hofstadter, 1984] Hofstadter, D. R. (1984). The Copycat project: An experiment in nondeterminism and creative analogies. AI Memo 755, MIT Artificial Intelligence Laboratory.

[Hofstadter and FARG, 1995] Hofstadter, D. R. and FARG (1995). *Fluid Concepts and Creative Analogies: Computer Models of the Fundamental Mechanisms of Thought.* Basic Books, New York.

[Leake, 1996] Leake, D. B., editor (1996). *Case-Based Reasoning: Experiences, Lessons, & Future Directions.* MIT Press/AAAI Press, Cambridge, MA.

[Mitchell, 1993] Mitchell, M. (1993). *Analogy-making as Perception.* MIT Press/Bradford Books, Cambridge, MA.

[VanLehn et al., 1992] VanLehn, K., Jones, R., and Chi, M. (1992). A model of the self-explanation effect. *The Journal of the Learning Sciences*, 2(1):1–59.

[Veloso, 1994] Veloso, M. (1994). *Planning and Learning by Analogical Reasoning.* Springer-Verlag, Berlin.

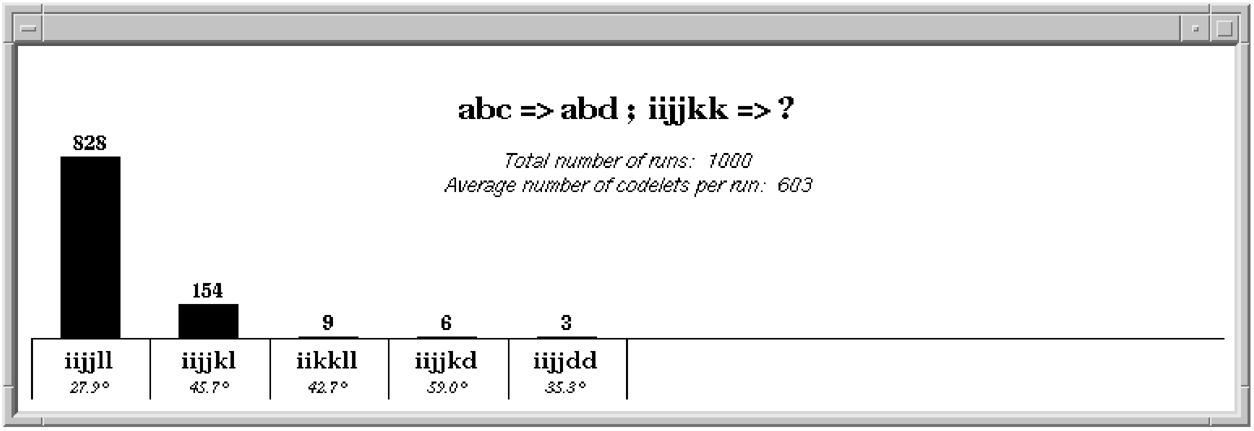
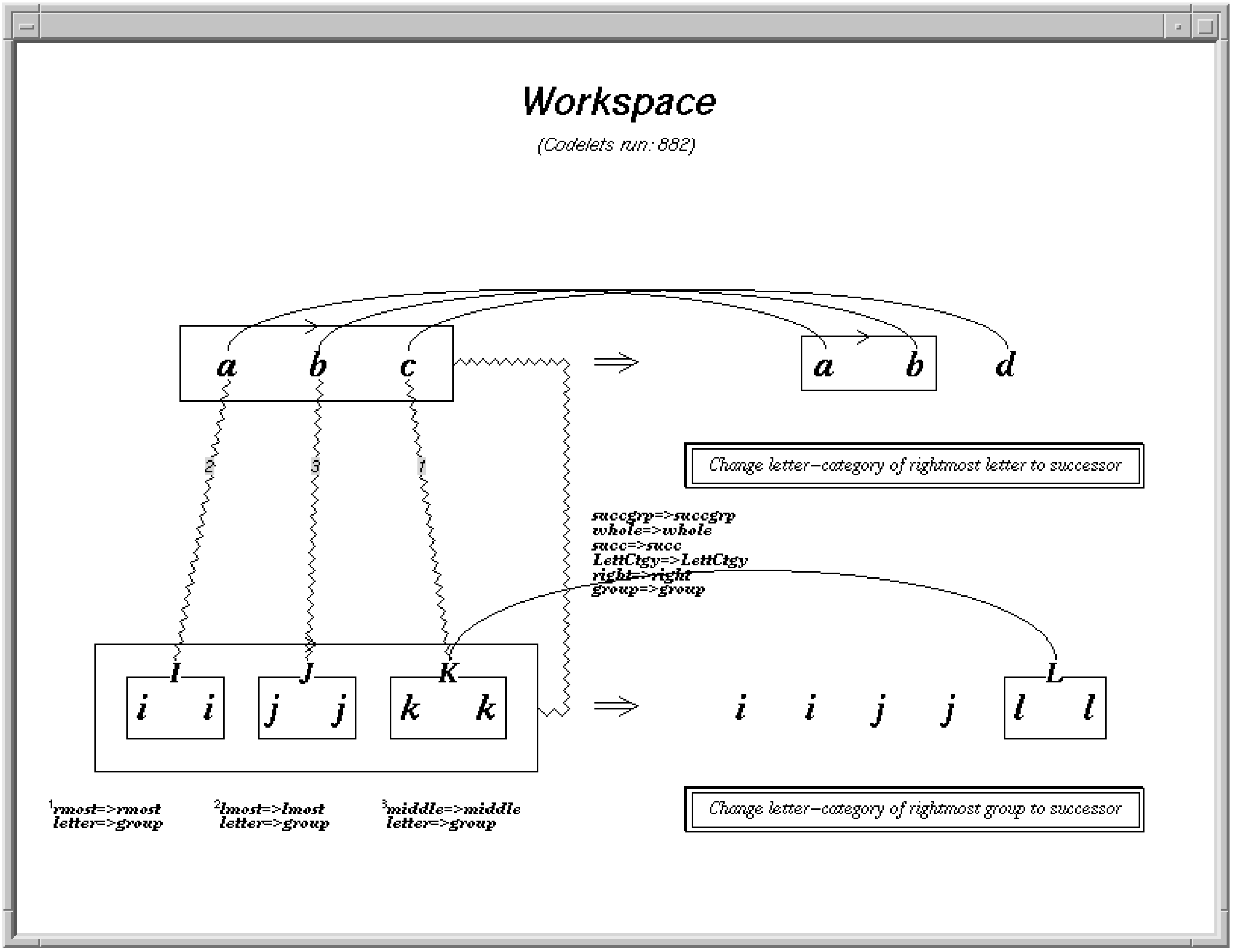


Figure 1: *Histogram of Copycat's answers for the problem "***abc** ⇒ **abd***;* **iijjkk** ⇒ **?***". Numbers above each bar indicate how often the answer was found in 1000 runs of the program. Temperature values below each answer represent the program's assessment of the answer's quality (with lower temperature indicating higher quality). The strange answer* **iikkll** *reflects weaknesses in Copycat's ability to form coherent groups. For this answer, the program saw* **jjkk** *as the "rightmost group" of* **iijjkk***, which it then changed to* **kkll**.



Change letter–category of rightmost letter to successor

Change letter–category of rightmost group to successor

Figure 2: *Screen dump of the Workspace after the program has found the answer* **iijjll** *for the problem "***abc** ⇒ **abd***;* **iijjkk** ⇒ **?***". The concept-mappings supporting the three vertical bridges are shown along the bottom of the screen. The final Workspace temperature for this answer was 27.*

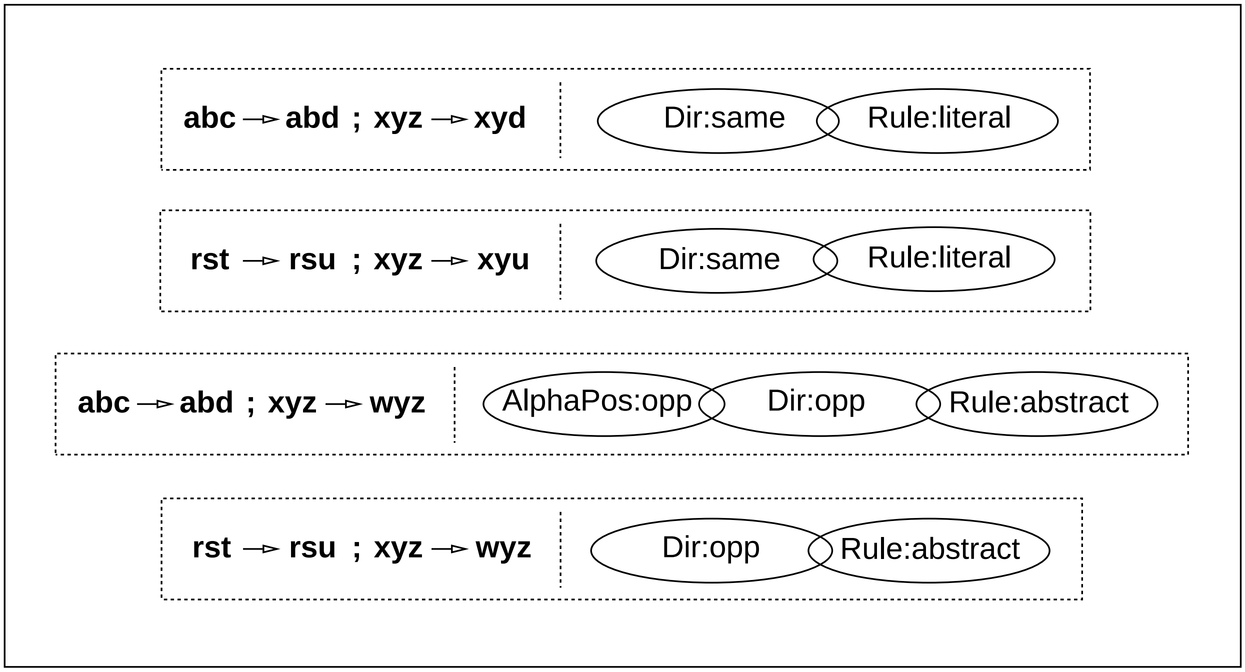


Figure 3: *Schematic representation of four analogies stored in Metacat's memory along with the most important themes that characterize each one.*