Title
The Right Concept at the Right Time How Concepts Emerge as Relevant in Response to Context-Dependent Pressures

Permalink
https://escholarship.org/uc/item/8t04d6g3

Journal
Proceedings of the Annual Meeting of the Cognitive Science Society, 12(0)

Authors
Mitchell, Melanie
Hofstadter, Douglas R.

Publication Date
1990

Peer reviewed

The Right Concept at the Right Time: How Concepts Emerge as Relevant in Response to Context-Dependent Pressures

Melanie Mitchell and Douglas R. Hofstadter
Center for Research on Concepts and Cognition
Indiana University

Abstract

A central question about cognition is how, faced with a situation, one explores possible ways of understanding and responding to it. In particular, how do concepts initially considered irrelevant, or not even considered at all, *become* relevant in response to pressures evoked by the understanding process itself? We describe a model of concepts and high-level perception in which concepts consist of a central region surrounded by a dynamic nondeterministic "halo" of potential associations, in which relevance and degree of association change as processing proceeds. As the representation of a situation is built, associations arise and are considered in a probabilistic fashion according to a *parallel terraced scan*, in which many routes toward understanding the situation are tested in parallel, each at a rate and to a depth reflecting ongoing evaluations of its promise. We describe a computer program that implements this model in the context of analogy-making, and illustrate, using screen dumps from a run, how the program's ability to flexibly bring in appropriate concepts for a given situation emerges from the mechanisms we are proposing.

Suppose you invite your friend Greg to dinner, and he doesn't show up on time. What do you do? At first, simple, standard explanations and actions come to mind: he was briefly delayed; he ran into traffic; he had trouble parking. But as half an hour passes, then an hour, then two, the explanations and actions you think of become more and more out of the ordinary. The following might come to mind: call his office (no answer); call his apartment (no answer); check your calendar to make sure the dinner date *is* tonight (it is); rack your brains trying to remember if he warned you he might be late (no such memory); call friends of his to see if they know where he is (they don't); call his parents in Philadelphia (haven't heard from him in weeks); call the police (they suggest checking the hospital); call the hospital (not there); go to his apartment (not there); ask his neighbors if they've seen him lately (last saw him this morning); drive along routes he would likely have taken (he's nowhere to be seen); buy a megaphone and call out his name as you drive along; call several airlines to see if he's on a plane leaving town tonight; turn on the TV to see if you can spot him sitting in the audience of his favorite talkshow; and so on. Though the last few are outlandish, most of these thoughts *did* occur to the authors when they were in such a situation. The point is: as time goes by and pressure builds up, one's thoughts go farther and farther out on a limb. One considers things one never would have considered initially, letting seemingly unquestionable aspects of the situation "slip" under mounting pressure (e.g., Did I dream I invited him? Did we have a falling-out I forgot about? Did he leave town and not tell me?).

This example illustrates some critical issues in cognition: Faced with a situation, how does one explore possible ways of understanding it, explaining it, or acting in response to it? How do concepts initially considered irrelevant, or not even considered at all, *become* relevant in response to pressure? How does one let go of notions that *looked* relevant but turn out not to be of help after all?

We are studying these issues by developing a model of concepts and high-level perception in which a concept consists of a central region surrounded by a dynamic, probabilistic "halo" of potential associations (Hofstadter, 1988). In its halo, "driving" has such concepts as "parking", "getting stuck in traffic", "having an accident", etc., each with a degree of association that changes in response to context (a phenomenon often discussed by psychologists, e.g., Tversky, 1977; Barsalou, 1989). The halo has no fixed boundary; it cannot be said absolutely that a given concept *is* or *is not* associated with "driving". Instead, different degrees of association reflect probabilities that once a concept is seen as relevant, various associated concepts will also become relevant. The dynamic nature of relevance and conceptual distance imbues human concepts with flexibility and adaptability.

Not only are certain concepts *explicitly* present in one's mental representation of a situation (you consciously believe Greg is *driving*); there are also *implicit* associations with those concepts, most of which stay well below the level of awareness. Given Greg's lateness, the thought that he's driving might easily evoke an image of his having trouble parking (a strong association). However, it is less likely that, early on, you will imagine him in a car accident. This weaker association is *potentially* there, but will not be brought into the picture without pressure (he is quite late, it is dark outside, etc.) This illustrates a general point: far-out ideas (or even ideas slightly past one's defaults) cannot continually occur to people for no good reason; a person to whom this happens is classified as crazy or crackpot. Time and cognitive resources being limited, it is vital to resist nonstandard ways of looking at situations without strong pressure to do so. As an extreme example, had the Michelson–Morley experiment come out the other way (i.e., it had proved there *is* an "ether") and had Einstein *still* proposed special relativity, with all its deeply counterintuitive notions, it would have been seen as just a fascinating crackpot theory, not a great scientific advance. Not only is pressure needed for one to bring in previously uninvolved concepts in trying to make sense of a situation, but the concepts brought in are related to the *source* of the pressure; they are a function of the pressure. (These ideas overlap with Kahneman & Miller's 1986 treatment of counterfactuals.)

One aim in our model is to avoid two opposite strategies, both psychologically implausible, for searching through concepts to be used in understanding a given situation: (1) All concepts are *explicitly* and *equally* available from the start (e.g., you have a preconstructed list of concepts relevant to "late-dinner-guest" situations — you may not need to try them all out, since Number 4 on the list might fill the bill, but they are spelled out nonetheless. An equally implausible variant of this would be that the possibilities are not spelled out explicitly, but it is easy to generate the next one on the list if a given entry fails); and (2) Certain concepts are definitively excluded from the start, and can *never* be brought in as relevant. A premise of our model is that in humans, the presence or absence of a concept in a situation is not black-and-white; rather, all one's concepts should have the *potential* to become relevant in any situation, but due to the necessity for cognitive economy, they can't all be made available all the time or to the same degree. People resist even *generating* less standard views, not to mention *exploring* them; the less standard a view, the more it is resisted.

In our model, every concept possessed by the system has *some* probability of becoming relevant in every context, but different concepts have vastly different probabilities, and these vary with context. There are many possible explanations (you could have written down the wrong date or given Greg the wrong address; your street's name could even have been secretly changed), so it is important not to *absolutely* exclude any particular pathway ahead of time. All must be potentially open, but there is not enough time to explore all equally, or even to generate all. Allocation of cognitive resources to different pathways must be a *dynamic* function of context-dependent pressures, because those pressures might change as exploration proceeds (when you try to call Greg, you find your phone is out of order and no one can call in; this will tend to make the "car accident" pathway less plausible). Our model proposes that many potential pathways are being tested out all the time, but at different speeds and to different levels of depth: due to context-dependent pressures, not all pathways are tested equally. Some may not be considered at all, but that's the luck of the (biased) draw; the point is that they are *potentially* open for exploration. We term this non-egalitarian style of exploration a *parallel terraced scan*: many different pathways are explored in parallel, but not equally; each pathway is explored at a rate and to a depth proportional to moment-to-moment estimates of its promise.

Our model thus has two interrelated aspects: The first is the existence of a probabilistic halo of potential associations around the central region of each concept. Like an electron orbit in an atom, a concept is blurry and distributed in "semantic space", with various probabilities that it will "be" at a given spot. For concepts, as for electrons, the probability distribution changes in a context-dependent way. The seond aspect is the notion of a parallel terraced scan. These ideas are implemented in Copycat, a computer model of concepts and high-level perception in analogy-making.

To isolate many issues of general psychological import, we use an idealized microworld in which these issues emerge very clearly. Our methodology resembles that of physics, where problems are *idealized* in order to isolate what is interesting about them and to allow them to be studied more precisely. In this spirit, Copycat operates in a "frictionless" world consisting of analogy problems involving letter strings; despite their apparent simplicity, these problems capture many of the broad issues we are investigating. Four sample problems in Copycat's microworld are:

1. **abc ⇒ abd**; **ijk** ⇒ ?
2. **abc ⇒ abd**; **iijjkk** ⇒ ?
3. **abc ⇒ abd**; **kji** ⇒ ?
4. **abc ⇒ abd**; **xyz** ⇒ ?

Solving such problems requires many abilities necessary for high-level perception and analogy-making in general: mentally building a coherently structured whole from initially unconnected parts; describing objects, relations, and events at an "appropriate" level of abstraction; paying attention to relevant aspects and ignoring irrelevant and superficial aspects of situations; deciding which elements of a situation to chunk and which to view individually; deciding which descriptions to take literally and which to let slip when perceiving correspondences between aspects of two situations; and allowing competition among various ways of interpreting and mapping the situations. Discussions of how problems 1–4 require these abilities and how Copycat solves 1–4 are given in Hofstadter & Mitchell (1988) and Mitchell & Hofstadter (1990). Our goal is not to study the domain-specific mechanisms people use in solving letter-string analogies, but to develop a computer model of human flexibility and insight in general; we use this microworld because it cleanly isolates many of the abilities we are investigating.

The central issue of this paper — how dormant concepts "bubble up" in response to pressure and become relevant — arises somewhat in problems 1–4, but is manifested most clearly in this one:

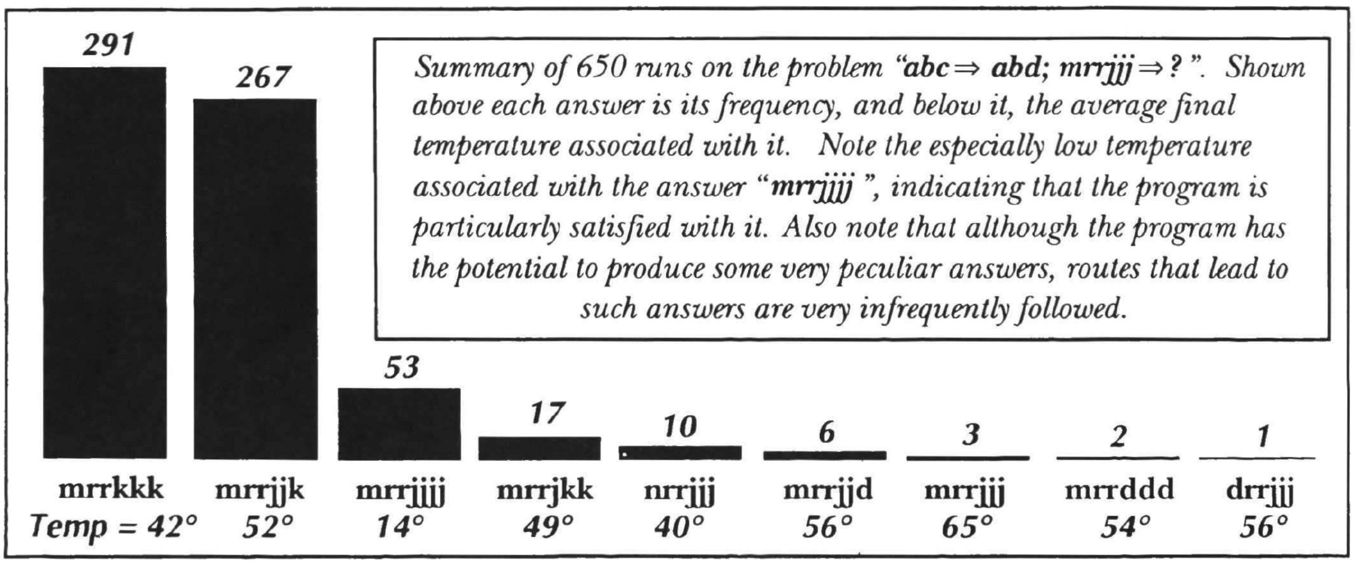
5. **abc ⇒ abd; mrrjjj ⇒ ?**

This problem has a seemingly reasonable, straightforward solution: **mrrkkk**. Most people give this answer, reasoning that since **abc**'s rightmost letter was replaced by its successor, and since **mrrjjj**'s rightmost "letter" is actually a *group* of 'j's, one should replace all the 'j's by 'k's. Another possibility is to take "rightmost letter" literally, thus to replace only the rightmost 'j' by 'k', giving **mrrjjk**. However, neither answer is very satisfying, since neither takes into account the salient fact that **abc** is an alphabetically increasing sequence. This internal "fabric" of **abc** is a very appealing and seemingly *explanatory* aspect of the string, so you want to use it in making the analogy, but how? No such fabric seems to weave **mrrjjj** together. So either (like most people) you settle for **mrrkkk** (or possibly **mrrjjk**), or you look more deeply. But where to look when there are so many possibilities?

The interest of this problem is that there happens to be an aspect of **mrrjjj** lurking beneath the surface that, once recognized, yields what many people feel is a more satisfying answer. If you ignore the *letters* in **mrrjjj** and look instead at *group lengths*, the desired successorship fabric is found: the lengths of groups increase as "1-2-3". Once this hidden connection between **abc** and **mrrjjj** is discovered, the rule describing **abc ⇒ abd** can be adapted to **mrrjjj** as "Replace the length of the rightmost group by its successor", yielding "1-2-4" at the abstract level, or, more concretely, **mrrjjjj**. Thus this problem demonstrates how a previously irrelevant, unnoticed aspect of a situation emerges as relevant in response to pressures (e.g., the unsatisfied desire for a common fabric, among others).

How can the notion of group length, which in most problems remains essentially dormant, come to be seen as relevant by Copycat? *Length* is certainly in the halo of the concept *group*, as are concepts such as *letter-category* (e.g., 'j' for the group 'jjj'), *string-position* (e.g., *rightmost*), and *group-fabric* (e.g., *sameness between letters*). Some are more closely associated with *group* than others; in the absence of pressure, the notion of *length* tends to be fairly far away in conceptual space. Thus in perceiving a group such as 'rr', one is virtually certain to notice the letter-category ('r'), but not very likely to notice, or at least attach importance to, the length. However, since *length* is in *group*'s halo, there is *some* possibility that lengths will be noticed and used in trying to make sense of the problem. One might consciously notice a group's length at some point, but if this doesn't turn out to be useful, *length*'s relevance diminishes after a while. (For example, this might happen in the variant problem **abc ⇒ abd, mrrrrjj ⇒ ?**.) This *dynamic* aspect of relevance is very important: even if a new concept is at some point brought in as relevant, it is counterproductive to continue spending much of one's time exploring avenues involving that concept if none seems promising.

Since Copycat is nondeterministic, it follows different paths on different runs; thus not only does it come up with a variety of answers, but it can reach each answer in myriad ways. Indeed, Copycat's flexibility depends on the fact that all pathways involving any of its concepts are potentially open; despite this, the program generally manages to avoid exploring unpromising pathways, except fleetingly. Below is a chart showing the results of running Copycat some 650 times on problem 5. Its answers, ordered by frequency, range from the superficially alluring **mrrkkk** to the downright bizarre **mrrjkk** (in which the two rightmost 'j's were perceived as a chunk), **nrrjjj** (in which **abc**'s *rightmost* letter was equated with **mrrjjj**'s *leftmost* letter), **mrrjjj** (using the rule "Replace all 'c's by 'd's"), and **drrjjj**.



*Summary of 650 runs on the problem "**abc ⇒ abd; mrrjjj ⇒ ?**". Shown above each answer is its frequency, and below it, the average final temperature associated with it. Note the especially low temperature associated with the answer "**mrrjjjj**", indicating that the program is particularly satisfied with it. Also note that although the program has the potential to produce some very peculiar answers, routes that lead to such answers are very infrequently followed.*

Although there are only nine distinct answers, each run was unique on a fine-grained level. Under each answer is the *final temperature* averaged over all runs yielding that answer. Temperature is explained later; for the time being, think of a run's final temperature as a measure of Copycat's "happiness" with the answer produced, with high temperature corresponding to low happiness and vice versa. Thus Copycat is by far the happiest with **mrrjjjj**. Note the lack of correlation of frequency with final temperature — meaning, roughly, that obviousness and elegance are independent.

The frequencies shown in the chart are not meant to be strictly compared with the frequencies of various answers given by people to this problem, since, as we said earlier, the program is not meant to model the domain-specific mechanisms people use in solving these letter-string problems. Rather, what is interesting here is that the program *does* have the potential to arrive at very strange answers (such as **mrrjkk** and **drrjjj**), yet manages to steer clear of them almost all the time; most always, it gets answers that people find reasonable and, sometimes, even insightful.

In a complex world (even one with the limited complexity of Copycat's microworld), one never knows in advance what concepts may turn out relevant in a given situation. It is thus imperative not only to avoid dogmatically *open-minded* search strategies, which entertain all possibilities equally seriously, but also to avoid dogmatically *closed-minded* search strategies, which in an ironclad way rule out certain possibilities *a priori*. Copycat opts for a middle way, which of course leaves open the potential for disaster — and indeed, disaster occurs once in a while. This is the price that must be paid for flexibility. People, too, occasionally explore and even favor peculiar routes. Our program, like us, has to have the potential to concoct far-out solutions in order to be able to discover subtle and elegant ones like **mrrjjjj**. (In fact, Copycat still lacks important mechanisms that would allow it to pursue yet stranger pathways!) To rigidly close off any routes *a priori* would necessarily remove critical aspects of Copycat's flexibility. On the other hand, the fact that Copycat so rarely produces weird answers demonstrates that its mechanisms manage to strike a pretty effective balance between open-mindedness and closed-mindedness, imbuing it with both flexibility and robustness.

We now sketch one way Copycat arrives at **mrrjjjj** (more details given below). The input consists of three "raw" strings (here, **abc**, **abd**, **mrrjjj**) with no preattached relations or preformed groups; it is thus left entirely to the program to build up perceptual structures constituting its understanding of the problem in terms of concepts it deems relevant. On most runs, the groups 'rr' and 'jjj' are constructed (the program is able to perceive copy-groups — groups consisting of repeated copies of a given letter — quite readily). Each group's letter-category ('r' and 'j' respectively) is explicitly noted, since *letter-category* is relevant by default. There is some probability for lengths to be noticed at the time the groups are made, but it is low, since *length* is not strongly associated with *group*. Once 'rr' and 'jjj' are made, *copy-group* becomes very relevant. This creates top-down pressure for the system to describe other objects — especially in the same string — as copy-groups if possible. The only way to do this here is to describe the 'm' as a copy-group with just one letter. This is strongly resisted by an opposing pressure: a single-letter group is an intrinsically weak and far-fetched construct. However, the existence of two other copy-groups in the string, coupled with the system's "unhappiness" at its failure to incorporate the lone 'm' into any large, coherent structure, pushes against this resistance.

These opposing pressures fight; the outcome is decided probabilistically. If the 'm' is perceived as a single-letter group, its length will very likely be noticed (single-letter groups are noteworthy precisely because of their abnormal length), making *length* more relevant in general, and thus increasing the probability of noticing the other two groups' lengths. Moreover, *length*, once brought into the picture, has a good chance of staying relevant, since descriptions based on it turn out to be useful. (Note that had the string been **mrrrrjj**, *length* might be brought in, but it would not turn out useful, so it would likely fade back into obscurity.) In **mrrjjj**, once lengths are noticed, the successor relations among them are quickly constructed by relation-detectors continually seeking new relations. There is also an independent top-down pressure to see successor relations in **mrrjjj**, coming from the already-seen successor relations in **abc**. As this satisfying new view of **mrrjjj** begins to emerge, the importance of the groups' letter-categories fades and *length* becomes their most salient aspect. Thus the crux of discovering this solution lies in the triggering of the concept *length*.

In summary, Copycat's solution of **abc** ⇒ **abd**, **mrrjjj** ⇒ **mrrjjjj** requires the interaction of:

- *concepts consisting of a central region surrounded by a halo of potential associations;*
- *a mechanism for probabilistically bringing in new associations related to the current situation;*
- *a mechanism by which concepts' activations decay over time, unless reinforced;*
- *agents that continually seek new relations, groups, and correspondences;*
- *mechanisms for applying top-down pressures from concepts already brought in;*
- *mechanisms allowing competition among pressures;*
- *the parallel terraced scan, allowing rival views to develop at different speeds.*

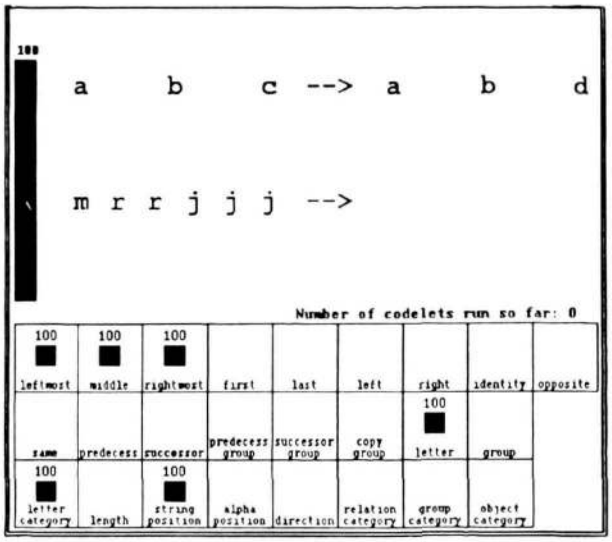
We now describe and illustrate how these mechanisms are implemented. (For more details, see Mitchell & Hofstadter, 1990.) Copycat's concepts reside in a network of nodes and links called the *Slipnet.* A concept's central region is a node, and its associative halo corresponds to other nodes linked to the central node. A node (such as *copy-group* or *successor*) becomes activated when instances of it are perceived (by *codelets,* as described below), and loses activation unless its instances remain salient. A node spreads activation to nearby nodes as a function of their proximity. Activation levels are not binary, but can vary continuously. The probability a node will be brought in or be considered further at any given time as a possible organizing concept is a function of the node's current activation level. Thus there is no black-and-white answer to the question of whether a given concept is "present" at a given time; continuous activation levels and probabilities allow different concepts to be present to different degrees. All concepts have the potential to be brought in and used; which ones become relevant and to what degree depends on the situation the program is facing, as will be seen below.

In addition to the Slipnet, where long-term concepts reside, Copycat has a working area in which perceptual structures (e.g., descriptions, relations, groups, and correspondences) are built hierarchically on top of the "raw" input (the three letter-strings). This building process is carried out by large numbers of simple agents called *codelets.* A codelet is a small piece of code that carries out some small, local task that is part of the process of building a structure (e.g., one codelet might notice that the two 'r's in **mrrjjj** are the same letter; another codelet might estimate how well that proposed relation fits in with already-existing relations; another codelet might build the relation). *Bottom-up* codelets work toward building structures based on whatever they happen to find, without being prompted to look for instances of specific concepts; *top-down* codelets look for instances of particular active nodes, such as *successor* or *copy-group.* The probability at a given time that a node in the Slipnet will add a top-down codelet to the current codelet population is a function of the node's activation level. Any structure is built by a series of codelets running in turn, each deciding probabilistically on the basis of its estimation of some aspect of the structure's strength whether to continue, by generating one or more follow-up codelets, or to abandon the effort at that point. If the decision is made to continue, the running codelet assigns an *urgency* value (based on its estimate of the structure's promise) to each follow-up codelet. This value helps to determine how long each follow-up codelet will have to wait before it can run and continue the evaluation of that particular structure.

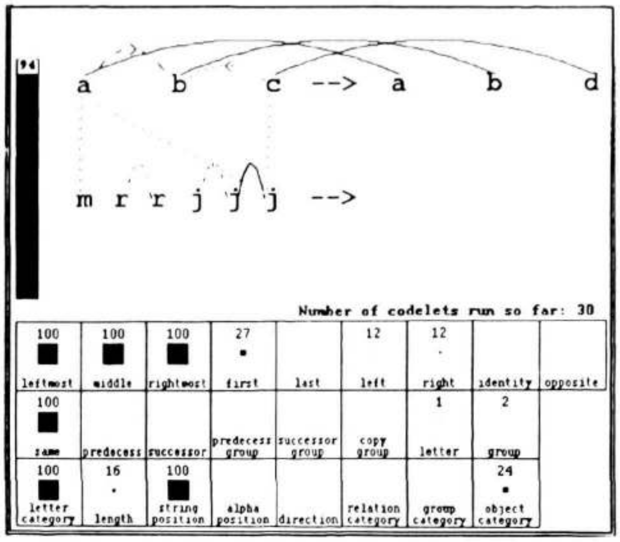
Any run starts with a standard initial population of bottom-up codelets with preset urgencies; at each time step, one codelet is chosen to run and is removed from the current codelet population. The choice is probabilistic, based on relative urgencies in the current population. As the run proceeds, new codelets are added to the population either as follow-ups to previously-run codelets, or as top-down scouts for active nodes. A new codelet's urgency is assigned by its creator as a function of the estimated promise of the task it is to work on. Thus the codelet population changes as the run proceeds, in response to the system's needs as judged by previously-run codelets and by activation patterns in the Slipnet, which themselves depend on what structures are built. There is no top-level executive directing the system's activity; all processing is carried out by codelets.

The fine-grained breakup of structure-building processes serves two purposes: (1) it allows many such processes to be carried out in parallel, by having their components interleaved; and (2) it allows the computational resources allocated to each such process to be dynamically regulated by moment-to-moment estimates of the promise of the pathway being followed. A process is not a predetermined macroscopic act that is then broken up into convenient chunks; rather, any sequence of codelets that amounts to a coherent macroscopic act can *a posteriori* be labeled a process — thus processes are *emergent.* The speed of any process emerges dynamically from the urgencies of its component codelets. The upshot is a parallel terraced scan — more promising views tend to be explored faster than less promising ones.

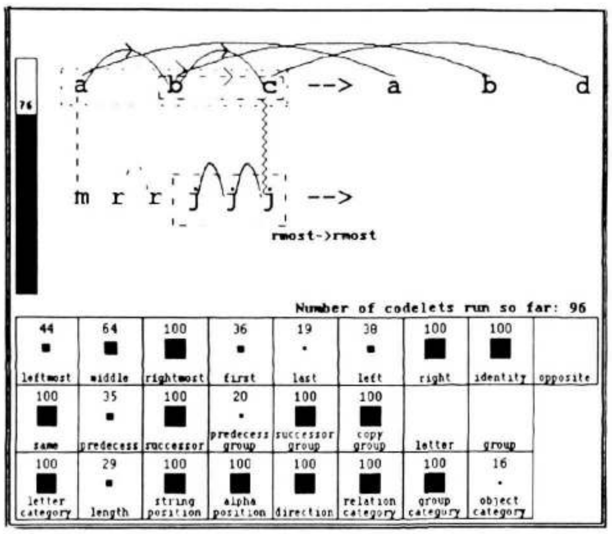
A final mechanism, *temperature,* both measures the degree of perceptual disorganization in the system (its value at any moment is a function of the amount and quality of structure built so far), and controls the degree of randomness used in making decisions (e.g., which codelet should run next, which structure should win a competition, etc.). 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. Temperature in Copycat is described in detail in Mitchell & Hofstadter (1990). All these mechanisms are illustrated in the set of screen dumps from a run of the program, given below.



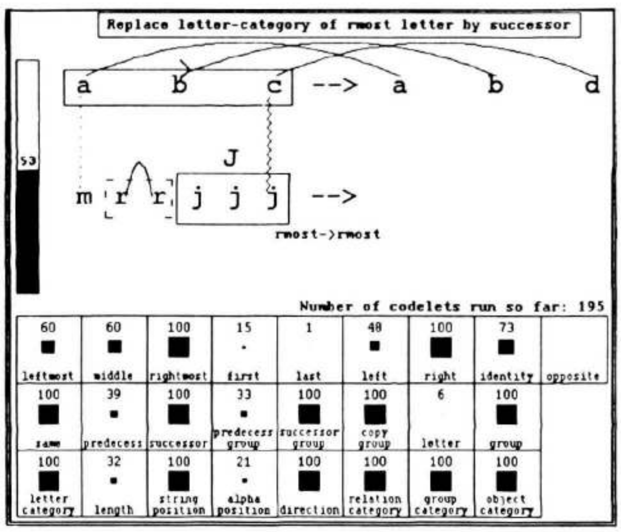
1. The problem is presented. Temperature, shown on a "thermometer" (left), is at its maximum of 100 (no structures yet built). At the bottom, some Slipnet nodes are displayed (links not shown). (Due to limited space, many nodes are not shown, e.g., those for 'a', 'b', etc.) A black square represents a node's current activation level (the actual value, from 0 to 100, is shown above). Nodes here displayed include *string-positions* of objects (*leftmost, middle,* and *rightmost*); *alphabetic-positions* of letters (*first,* filled by 'a', and *last,* by 'z'); *directions* for relations and groups (*left* and *right*); *identity* and *opposite* (some of the possible relations between concepts); *relation-categories* for relations between letters and groups (*same, predecessor,* and *successor*); *group-categories* (*predecessor-group, successor-group,* and *copy-group*); *object-categories* (*letter* and *group*); and in row 3, nodes representing these various categories of descriptions, including *length.* Every letter has some preattached descriptions: *letter-category* (e.g., 'm'), *object-category* (*letter,* as opposed to *group*) and *string-position* (*leftmost, middle, rightmost,* or none — e.g., the fourth letter in **mrrjjj** has no string-position description). These nodes start out highly activated.



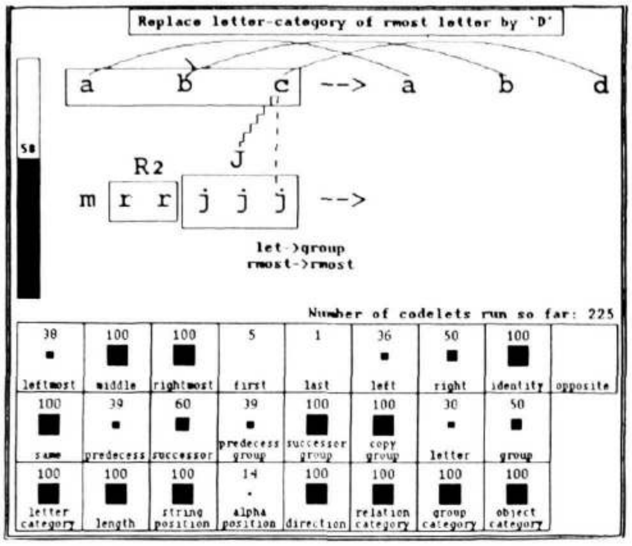
2. The 30 codelets so far run have begun exploring many possible structures. Dashed lines and arcs are structures in various stages of consideration; solid ones are structures actually *built,* which can thus influence temperature, and the building of other structures. Relations and correspondences between letters are being considered (the 'a'–'j' potential mapping is based on the weak *leftmost–rightmost* Slipnet link; being implausible, it won't be pursued much further). The **abc/abd** correspondences and the rightmost 'j'–'j' sameness relation have been built by bottom-up codelets; this activated *same,* resulting in top-down pressure (new codelets) to seek sameness elsewhere. Some nodes have become lightly activated via spreading activation (e.g., the node *first,* from 'a' [not shown]). *Length's* activation comes from its weak association with *letter-category* (letters and numbers are increasing sequences and thus similar; numbers are associated with length). Temperature has fallen in response to structures so far built. Many non-displayed fleeting explorations are occurring (e.g., "Any relation between the 'm' and its neighbor 'r'?").



3. The successorship fabric of **abc** has been seen, and two mutually competing groups based on it are being considered: 'bc' and 'abc'. The latter is much stronger than the former, thus has a much higher chance. Exploration of the diagonal 'a'–'j' correspondence was aborted. A 'c'–'j' correspondence has been built (jagged vertical line); its reason for existence (both letters are rightmost) is given beneath it. A 'jjj' group is being strongly considered. Since *successor* and *sameness* relations have been built, these nodes are highly active; they in turn have spread activation to *successor-group* and *copy-group,* which creates top-down pressure to look for such groups. Also, since *first* was active, *alphabetic-position* became highly active (a probabilistic event), making alphabetic-position descriptions likely to be considered.

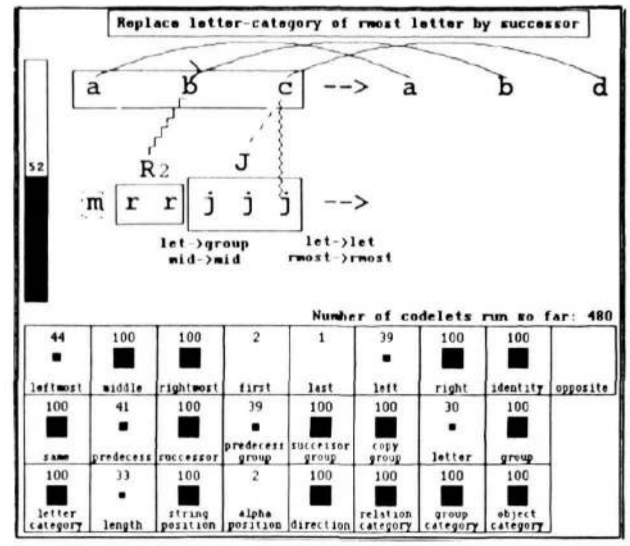


4. Groups 'abc' and 'jjj' have been built (relations between letters are no longer being displayed). An 'rr' group is being considered (the group 'jjj' strongly supports it, so its construction is accelerated). Meanwhile, a *rule* (at top) has been constructed to describe how **abc** changed. The current version of Copycat assumes the example change involves replacing exactly one letter, so rule-building codelets fill in the template "Replace ____ by ____", choosing probabilistically from descriptions the program has given to the changed letter and its replacement, with a default bias toward more abstract descriptions (e.g., usually preferring "rightmost letter" to 'c'). Nodes *first* and *alphabetic-position* didn't turn out useful and thus have faded. Also, *length* received additional activation from *group* but is still not very activated, so noticing lengths is still unlikely.

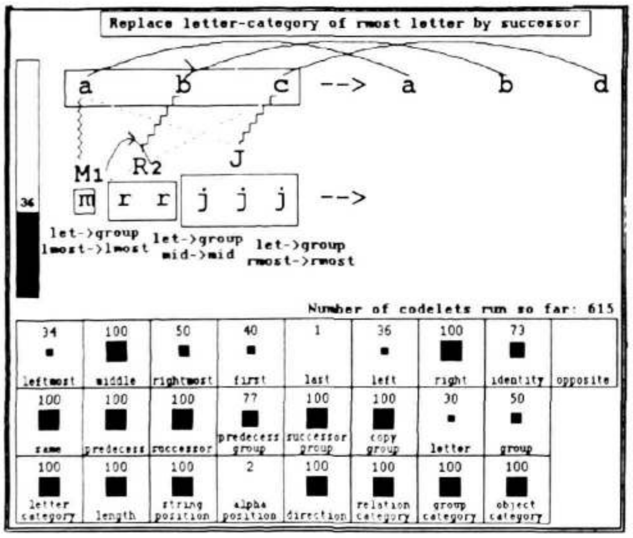


5. Now, 225 codelets into the run, the letter-to-*letter* 'c'–'j' correspondence was defeated by a stronger letter-to-*group* 'c'–'J' correspondence, though the former possibility still lurks in the background. Meanwhile, an 'rr' group was built whose length was noticed (a probabilistic event) and is displayed at the top of the group. *Length* is now fully active.

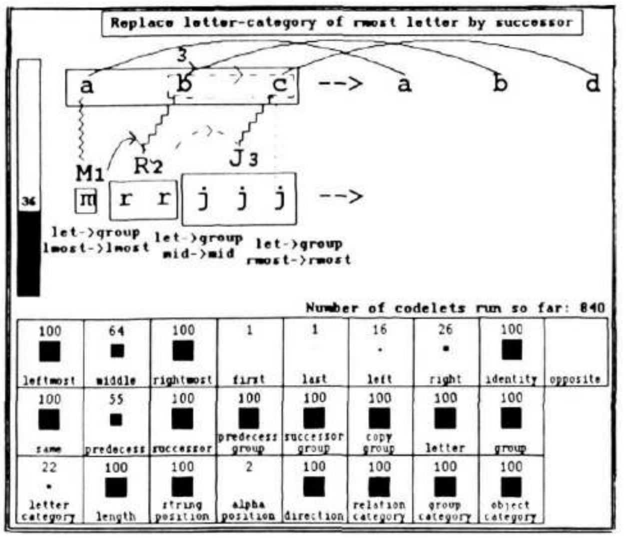
A new rule, "Replace the letter-category of the rightmost letter by 'D'", appears at the top of the screen; although it is weaker than the old rule, fights are decided probabilistically, and it won. However, its weakness caused temperature to go up. If the program were to stop now (unlikely, as temperature is still fairly high; the program decides probabilistically when to stop, based on temperature), the rule would be adapted for **mrrjjj** as "Replace the letter-category of the rightmost *group* by 'D'" (the 'c'–'J' correspondence establishes that the role of *letter* in **abc** is played by *group* in **mrrjjj**), yielding **mrrddd** (and Copycat does get this answer on occasion).



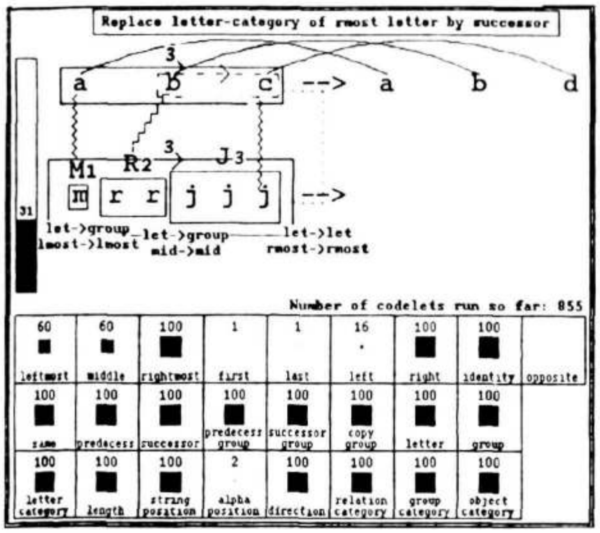
6. The previous, stronger rule has been restored (again the result of a fight having a probabilistic outcome), but the 'c'–'J' correspondence has been defeated by a 'c'–'j' correspondence. The activation of *length* has decayed a good deal, since the length description given to 'rr' hasn't been found to be useful. (This is graphically indicated by the fact that the '2' is no longer in boldface.) The temperature is still fairly high, since the program is having a hard time making a single, coherent structure out of **mrrjjj**, as it did with **abc**. That fact, combined with strong top-down pressure from the two copy-groups in **mrrjjj**, makes it somewhat plausible for the system to flirt with the idea of a single-letter group (dashed rectangle around the 'm').



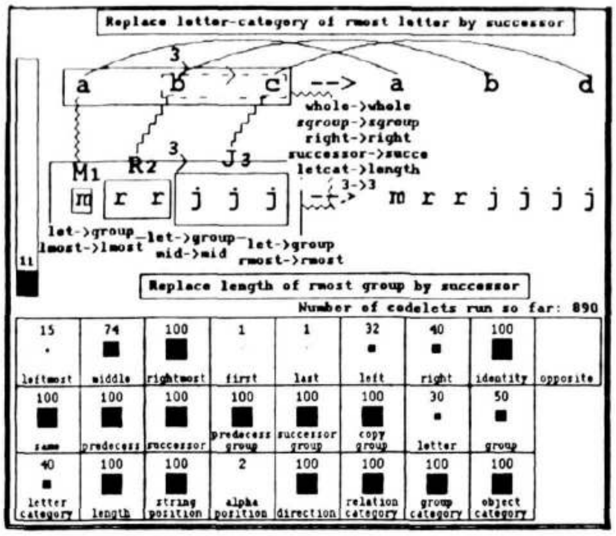
7. As a result of these pressures, the *a priori* extremely unlikely single-letter copy-group 'm' has been built, and its length of 1, being very noteworthy, has been attached as a description. The relation between the 1 and the 2 has been built; all of this is helping *length* to stay active. A complete set of *letter* ⇒ *group* correspondences has now been made, and as a result of these promising new structures, the temperature has fallen to 36, which in return helps to lock in this emerging view.



8. As a result of *length*'s continued activity, length descriptions have been attached to the other two groups in the problem ('jjj' and 'abc'), and a relation between the 2 and the 3 (for which there is much top-down pressure coming from both **abc** and the emerging view of **mrrjjj**) is being considered. *Letter-category* has decayed, indicating that it hasn't lately been of use in building structures.



9. The 2–3 relation was built and a successor-group was built out of the group-lengths in mrrjjj (large rectangle surrounding the three copy-groups). Also, a correspondence (dashed vertical line to the right of the two strings) is being considered between abc and mrrjjj in their entireties.



10. A correspondence has been built between the two strings as wholes, and its concept-mappings (e.g., *letter-category* ⇒ *length*) are listed to its right. The original rule has been translated, using these concept-mappings; the translated rule appears just above the Slipnet, and the answer mrrjjjj at the right. The low temperature reflects the program's satisfaction with this answer.

In summary, Copycat is a model of concepts and perceptual mechanisms flexible enough to deal with a range of problems in its microdomain, reflecting many central psychological issues. It differs from other computer approaches to analogy-making in that it models not only how situations are mapped onto each other, but also the mechanisms by which initially uninterpreted situations are mentally structured. Models such as SME (Falkenhainer et al., 1986), and ACME (Holyoak & Thagard, 1989), are concerned with just the mapping process; all relations and other perceptual structures are precoded into predicate-logic representations that serve as input. A detailed comparison of Copycat with these models is given in Mitchell (1988). Copycat starts on any problem from a standard initial state; however, it quickly senses unique aspects of the problem, bringing out certain associations while downplaying others, allowing it (usually) to home in on a suitable set of relevant concepts and avenues of approach. Copycat achieves, through mechanisms we believe are psychologically plausible, a delicate balance between being too open-minded (exploring every avenue indiscriminately, thus grossly wasting computational resources) and being too closed-minded (rigidly cutting out certain avenues *a priori*, thus preventing many creative pathways from ever being looked at). Walking this fine line imbues Copycat with its robustness and flexibility.

Acknowledgments

We thank Robert French for contributions to the Copycat Project, and Liane Gabora for writing the statistics-gathering program. Thanks also to David Chalmers and David Moser for helpful comments on this paper, and to Greg Huber for a "late" inspiration. This research has been supported by grants from Indiana University, the University of Michigan, and Apple Computer, Inc., as well as a grant from Mitchell Kapor, Ellen Poss, and the Lotus Development Corporation, and grant DCR 8410409 from the National Science Foundation.

References

[1] Barsalou, L. W. (1989). Intraconcept similarity and its implications for interconcept similarity. In Vosniadou, S. and A. Ortony, *Similarity and analogical reasoning,* 76–121. Cambridge, England: Cambridge University Press.

[2] Falkenhainer, B., K. D. Forbus, and D. Gentner (1986). The Structure-Mapping Engine. In *Proceedings of the American Association for Artificial Intelligence, AAAI-86.* Los Altos, CA: Morgan Kaufmann.

[3] Hofstadter, D. R. (1988). Common sense and conceptual halos. *Behavioral and Brain Sciences, 11* (1), 35–37.

[4] Hofstadter, D. R. and M. Mitchell (1988). Conceptual slippage and analogy-making: A report on the Copycat project. *Proceedings, Tenth Annual Cognitive Science Society Conference.* Hillsdale, NJ: Lawrence Erlbaum Associates.

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

[6] Kahneman, D. and D. T. Miller (1986). Norm theory: Comparing reality to its alternatives. *Psychological Review, 93* (2), 136–153.

[7] Mitchell, M. (1988). A computer model of analogical thought. Unpublished thesis proposal. University of Michigan, Ann Arbor, MI.

[8] Mitchell, M. and D. R. Hofstadter (1990). The emergence of understanding in a computer model of concepts and analogy-making. *Physica D,* in press.

[9] Tversky, A. (1977). Features of similarity. *Psychological Review, 84,* 327–352.