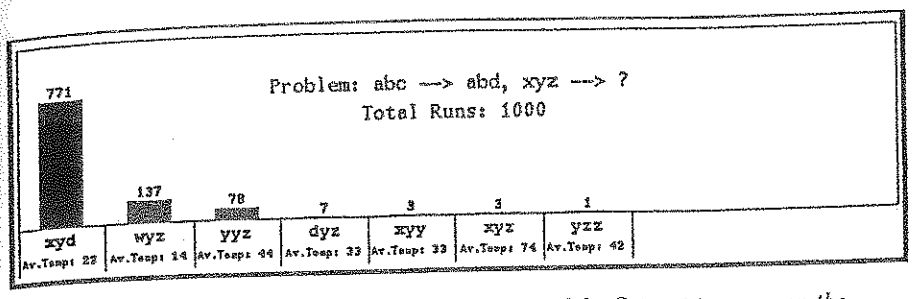


*Figure V–6. Bar graph summarizing 1,000 runs of the Copycat program on the analogy problem "**abc** ⇒ **abd**; **xyz** ⇒ ?".*

idea carried out only halfway, people very often come up with it. Indeed, such blends and half-completed trains of thought are very characteristic of human cognition (Hofstadter & Moser, 1989 gives examples of many types of cognitive blends and discusses their origin).

The other four answers are much farther out on the low-frequency fringes. The answer ***dyz*** (much like ***dji*** in Problem 3) is a highly implausible blend of insight and simple-mindedness, the insight being the subtle perception of the abstract symmetry linking ***abc*** and ***xyz***, and the simple-mindedness being the extremely concrete and unimaginative way of conceiving the ***abc*** ⇒ ***abd*** change. Amusingly, this answer is self-descriptive, in that ***dyz*** can be pronounced "dizzy". Indeed, some people find this answer so dizzy in its style of thought that it evokes laughter. In Hofstadter *et al.* (1989), this and several other Copycat analogies are mapped onto real-world jokes, and are thereby used to suggest a theory of "slippage humor", one of whose tenets is that there is a continuum running from sensible through "sloppy" answers and winding up in "dizzy" answers, where "sloppy" and "dizzy" can be given semi-precise definitions in terms of the degree of consistency with which conceptual slippages are carried out.

The answer ***xyy*** allows that the two strings are to be perceived in opposite *alphabetic* directions (thus a *successor* ⇒ *predecessor* slippage), yet refuses to give up the idea that the strings have the same *spatial* direction; it thus insists on changing the *rightmost* letter, as was done to ***abc***. It is amusing to note that ***ijj*** — Problem 1's analogue to this answer[6] — was produced one time in 1,000 runs, even without the pressure of a snag.

6. It is ironic that a claim of analogousness of answers to different Copycat problems (such as the offhand remark made in the text that ***ijj*** in Problem 1 is "the analogue" to ***xyy*** in Problem 6) comes across as objective and unproblematic to most people, despite the fact that many people express doubt about the notion of "rightness" or "wrongness" of letter-string analogies. The fact is, most people *do* have a strong intuitive sense of right and wrong analogies — it's just that when the psychological context is "Solve this analogy puzzle", they put their guard up and become wary of any claims, whereas when the context is "commentary on our program's behavior", they lower their guard and go with their intuitions, without even realizing the change in their attitude.

The answer *xyz*, whose very high temperature of 74 indicates that the program did not "like" it at all, comes from interpreting the *abc* ⇒ *abd* change as "Replace *c* by *d*". (This is not nearly as clever as positing that the *z* might serve as its *own* successor, which is an entirely different way of justifying this same answer, and one that people fairly often suggest. In fact, when *xya* is barred, *xyz* is what people come up with most often.) Note that *ijk*, the analogous answer[6] to Problem 1, was *never* produced. It takes the "desperation" caused by the *z*-snag to allow such strange ideas any chance at all.

Finally, answer *yzz* is a peculiar, almost pathological, variant of the above-discussed answer *yyz*, in which the *x* and *y* in *xyz* are grouped together as one object, which is then replaced *as a whole* by its "successor" (the successor of each letter in the group). Luckily, it was produced only once in 1,000 runs, and was considered a poor answer.

In Problem 6, pressure for a crosswise mapping (leading to the answer *wyz*) comes both from the existence of an impasse and from the possibility of an appealing way out of that impasse — namely, a high-quality bridge linking instances of the two "distinguished" Platonic letters, *a* and *z*. Suppose that the impasse was retained while the appeal of the "escape route" was greatly reduced — what would be the effect on Copycat's behavior? The following variant explores that question.

7. Suppose the letter-string *rst* were changed to *rsu*; how would you change the letter-string *xyz* in "the same way"?

As Figure V–7 shows, *wyz* was produced on only 1 percent of the runs, whereas in Problem 6 it was given on almost 14 percent of the runs. Here, there is very little to suggest building a crosswise bridge, because the *r* and *z* have almost nothing in common, aside from the rather irrelevant fact that the *r* is leftmost in its string and the *z* rightmost in *its* string — hardly a powerful reason to make an *r*–*z* bridge.

For some perspective, compare this to Problem 1. How much appeal is there to the idea of mapping the leftmost letter of *abc* onto the rightmost letter of *ijk*? Such a crosswise *a*–*k* bridge would result either in the answer *hjk* (characterized at the beginning of this article as "unmotivated fluidity"), or possibly in *jjk* or *djk*. However, in 1,000 runs on Problem 1, Copycat never produced any of those answers, nor have we ever run into a human who has suggested any of them as an answer to Problem 1. (Actually, one person once *did* propose *hjk* in response to Problem 1, but this was under the influence of having just seen the *wyz* answer to Problem 6.) In colloquial terms, answers to Problem 1 based on a crosswise mapping seem completely "off the wall".

In Problem 7, of course, things are different, because there is, after all, a snag, and hence a kind of "desperation". The various emergency measures —

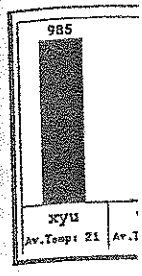


Figure
analog

especially th
r–*z* bridge a
point on, th
wyz is the ou
In som
wyz in Prob
motivated f
different co
intuition fe

Families of
It can
Copycat thi
running C
its perform
experience
a kind of "
Copycat wa
bring out s
analogy to
Epilogue
families of
with flying
and 5 of
The
show how
degree to
to differe
the essen
analogies

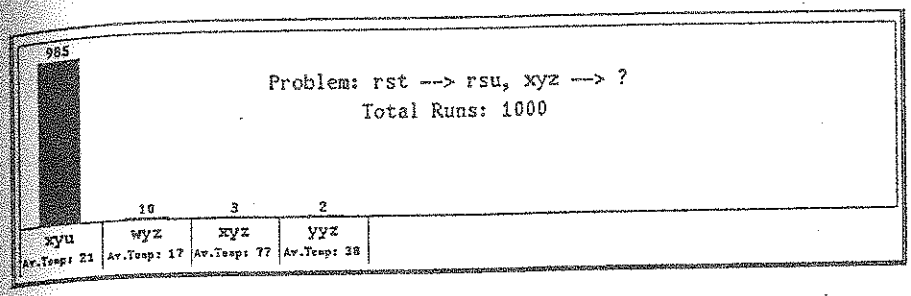


Figure V–7. Bar graph summarizing 1,000 runs of the Copycat program on the analogy problem "rst ⇒ rsu; xyz ⇒ ?".

especially the persistent high temperature — make the normally unappealing *r–z* bridge a bit more tempting, and so, once in a while, it gets built. From that point on, the whole paradigm shift goes through exactly as in Problem 6, and *wyz* is the outcome.

In some sense, Problem 7 lies halfway between Problems 1 and 6, so answer *wyz* in Problem 7 represents an intermediate stage between unmotivated and motivated fluidity. It is most gratifying to us that Copycat responds to the different constellations of pressures in these problems in much the way that our intuition feels it ought to.

Families of problems as a "miniature Turing Test"

It cannot be overestimated how critical we feel this method of probing Copycat through various families of subtly related problems is. When we began running Copycat on a large number of problems, we had no clear idea of what its performance would be, and we were, frankly, somewhat nervous. To us, the experience of watching Copycat reacting to each new problem had the feel of a kind of "miniature Turing Test", in the sense that each new problem posed to Copycat was like a question and answer in the Turing Test that would inevitably bring out some new and unanticipated aspect of the program's personality (this analogy to the Turing Test is further discussed in French, 1995; see also the Epilogue of this book). Copycat's mechanisms were truly put to the test by the families of problems we challenged it with, and by and large it came through with flying colors. A thorough discussion of those tests is found in Chapters 4 and 5 of Mitchell (1993).

The bar graphs just presented suggest the range of Copycat's abilities, and show how diverse constellations of pressures affect its behavior. They show the degree to which the program exhibits rudimentary fluid concepts able to adapt to different situations in a microworld that, though idealized, captures much of the essence of real-world analogy-making. They also reveal, by displaying *bad* analogies Copycat makes, some of the program's flaws and weaknesses. But they

also show that though Copycat has the *potential* to get far-fetched answers — a potential essential for flexibility — it still manages to *avoid* them almost all the time, which shows its robustness.

It is important to emphasize once again that our goal is not to model specifically how people solve these letter-string analogy problems (it is clear that the microworld involves only a very small fraction of what people know about letters and might use in solving these problems), but rather to propose and model mechanisms for fluid concepts and analogy-making *in general*. These mechanisms, described earlier on, will be illustrated in detail in the next section, which follows the temporal progression of Copycat as it solves two problems. First we take a particular run of the program on Problem 4 and present it through a series of screen dumps; then we move to Problem 6 and discuss the abstract pathway followed by almost all runs that lead to answer ***wyz***.

Copycat's Performance: A Tree-level Close-up

A problem where perception plays a crucial role is chosen as a focus

We now illustrate the mechanisms described in this article by presenting a detailed set of screen dumps from a single run of Copycat on Problem 4. As was discussed above, this problem has a seemingly reasonable, straightforward solution, ***mrrkkk***, but neither this answer nor the more literal ***mrrjjk*** is very satisfying, since neither reflects an underlying successorship structure in ***mrrjjj*** analogous to that in ***abc***. Such a successorship fabric can be found only if relationships between group lengths are perceived in ***mrrjjj***. But 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 other concepts, such as *letter category* (*e.g.*, ***j*** for the group ***jjj***), *string position* (*e.g.*, *rightmost*), and *group fabric* (*e.g.*, *sameness*). Some of these concepts are more closely associated with *group* than others; in the absence of pressure, the notion of *length* tends to be fairly far away from *group* in conceptual space. Thus in perceiving a group such as ***rr***, one is virtually certain to notice its letter category (namely, *r*), but not very likely to notice, or at least attach importance to, its length (namely, 2). However, since the concept *length* is in *group*'s halo, there is some chance that lengths will be noticed and used in trying to make sense of the problem. One might, for instance, consciously notice a group's length at some point, but if this doesn't turn out useful, *length*'s relevance will diminish after a short while. (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 spend-

ing much of one's time exploring avenues involving that concept if none seems promising.

The story in quick strokes

One way Copycat arrives at ***mrrjjjj*** is now sketched (of course, since the program is nondeterministic, there are many possible routes to any given answer). The input consists of three "raw" strings (here, ***abc***, ***abd***, and ***mrrjjj***) with no preattached bonds or preformed groups; it is thus left 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*** get built (the program tends to see sameness groups quite fast). Each group's letter category (***r*** and ***j***, respectively) is noted, as the concept *letter category* is relevant by default. Although there is some chance for length to be noticed when a group is made, it is low, as *length* is only weakly associated with *group*. Once ***rr*** and ***jjj*** are made, *sameness group* becomes very relevant, which creates top-down pressure to describe other objects, especially in the same string, as sameness groups, if possible. The only way to do this here is to describe the ***m*** as a sameness group having just one member. But this is resisted by a strong opposing pressure: a single-member group is an intrinsically weak and far-fetched construct. It would be disastrous if Copycat were willing to bring in unlikely notions such as single-member groups without strong pressure: it would then waste huge amounts of time exploring ridiculous avenues in every problem. However, the prior existence of two other strong sameness groups in the same string, coupled with the system's unhappiness at its failure to incorporate the lone ***m*** into any large, coherent structure (revealed by a persisting high temperature), pushes against this intrinsic resistance.

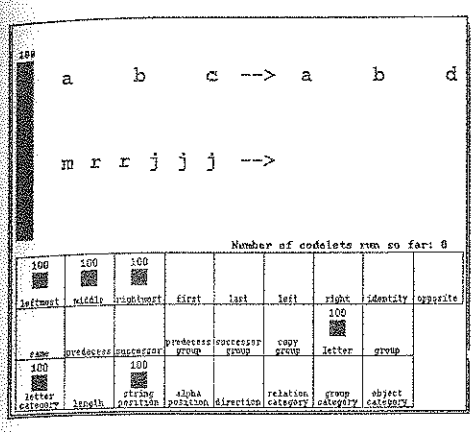
These opposing pressures fight; the outcome is decided only as a statistical result of probabilistic decisions made by a large number of codelets. If the ***m*** chances to be perceived as a single-letter sameness group, that group's length will very likely be noticed (single-letter groups are noteworthy 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. (Without reinforcement, a node's activation decays over time. Thus, for instance, had the target string been ***mrrrrjj***, *length* might get brought in at some point, but it would not turn out useful, so it would likely fade back into obscurity.)

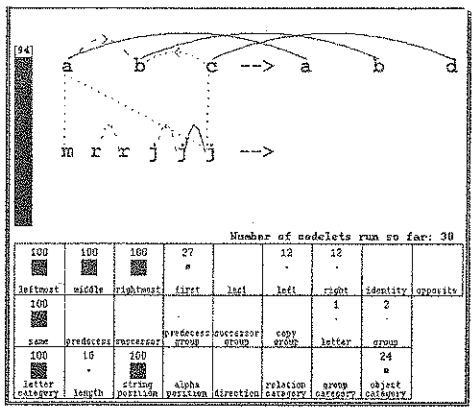
In ***mrrjjj***, once lengths are seen, the (numerical) successor relations among them might be spotted by bottom-up codelets, ever-present in the Coderack, continually seeking new relations in the Workspace. (Note that such

spontaneous bottom-up noticing could happen only in a parallel architecture where many types of properties can be continually being looked for at once without a need for explicit prompting.) Another way, perhaps more likely, that the noticing of successor relations in ***mrrjjj*** could occur is through top-down pressure caused by the already-seen successor relations in ***abc***. In any case, as soon as the numerical successorship relations are seen and a much more satisfying view of ***mrrjjj*** begins to emerge, interest in the groups' letter categories fades and *length* becomes their most salient aspect. Thus the crux of finding this solution lies in the triggering of the concept *length*.

Screen dumps tell the story in detail

Figure V–8 is a series of screen dumps from a run of Copycat, showing one way it arrives at the answer ***mrrjjjj*** (note that this answer is not very typical: according to Figure V–4, this answer is given only about 4 percent of the time).





1. The problem is presented. Temperature, shown on a "thermometer" (at the left), is at its maximum of 100, since no structures have yet been built. At the bottom, some Slipnet nodes are displayed. (Note: links are not shown. Also, 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 numerical value, between 0 and 100, is shown above the square).

Nodes here displayed include *leftmost, middle,* and *rightmost* (the possible *string positions* of objects in the Workspace); *first* and *last* (the distinguished *alphabetic positions* of Platonic letters ***a*** and ***z***); *left* and *right* (the possible *directions* for bonds and groups); *identity* and *opposite* (two of the possible relations between concepts); *same, predecessor,* and *successor* (the possible *bond categories* for bonds between Workspace objects); *predecessor group, successor group,* and *copy-group* [1] (the various *group categories*); *letter* and *group* (the possible *object categories* for Workspace objects); and in row 3, nodes representing these various categories of descriptions, including *length*.

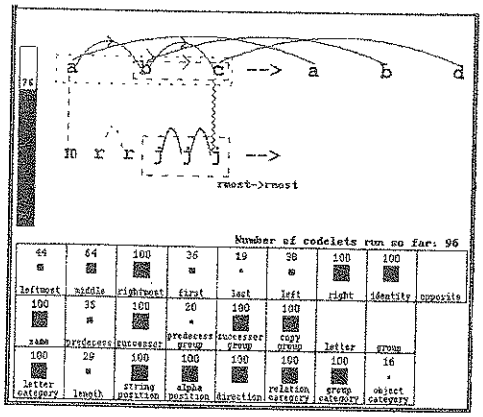
Every letter comes with some preattached descriptions: its *letter category* (*e.g.*, ***m***), its *string position* (*leftmost, middle, rightmost,* or none — *e.g.*, the fourth letter in ***mrrjjj*** has no string-position description), and its *object category* (*letter,* as opposed to *group*). These nodes start out highly activated.

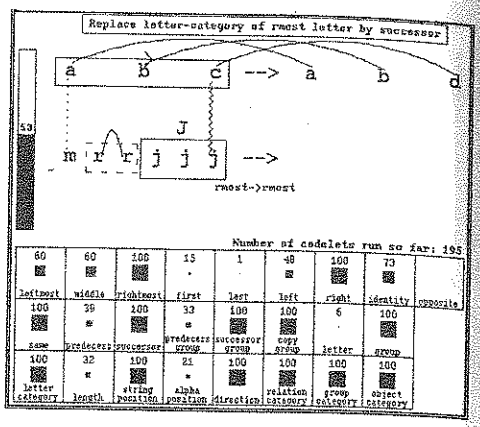
2. The 30 codelets so far run have begun exploring many possible structures. *Dotted* lines and arcs represent structures in early stages of consideration; *dashed* lines and arcs represent structures in more serious stages of consideration; finally, *solid* lines and arcs represent structures actually built, which can thus influence temperature as well as the building of other structures. Various bonds and bridges between letters are being considered (*e.g.*, the dotted ***a–j*** bridge, which is based on the relatively long *leftmost–rightmost* Slipnet link; being implausible, it won't be pursued much further).

Bridges connecting letters in ***abc*** with their counterparts in ***abd*** have been built by bottom-up codelets, as has a ***j–j*** sameness bond at the right end of ***mrrjjj***; this latter discovery activated the node *same,* resulting in top-down pressure (*i.e.*, new codelets) to seek instances of sameness elsewhere.

Some nodes have become lightly activated via spreading activation (*e.g.*, the node *first,* via activation from the node ***a*** [not shown]). The slight activation of *length* comes from its weak association with *letter category* (letters and numbers form linear sequences and are thus similar; numbers are associated with *length*). The temperature has fallen in response to the structures so far built. It should be pointed out that many fleeting explorations are constantly occurring (*e.g.*, "Are there any relations of interest between the ***m*** and its neighbor ***r***?"), but they are not visible here.

1. Note that the term *copy-group*, in this set of screen dumps and captions, means *sameness group*, in the terminology of the text.





3. The successorship fabric of ***abc*** has been observed, and two rival groups based on it are being considered: ***bc*** and ***abc***. Although the former got off to an early lead (it is dashed while the latter is only dotted), the latter, being intrinsically a stronger structure, has a higher chance of actually getting built.

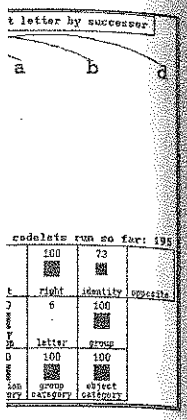
Exploration of the crosswise ***a–j*** bridge was aborted, since it was (probabilistically) judged to be too weak to merit further consideration. A more plausible ***c–j*** bridge has been built (jagged vertical line); its reason for existence (namely, both letters are *rightmost* in their respective strings) is given beneath it, in the form of an identity mapping.

Since *successor* and *sameness* bonds have been built, these nodes are highly active; they in turn have spread activation to *successor group* and *copy-group* (*i.e., sameness group*), which creates top-down pressure to look for such groups. Indeed, a ***jjj*** copy-group is being strongly considered (dashed box). Also, since *first* was active, *alphabetic position* became highly active (a probabilistic event), making alphabetic-position descriptions likely to be considered.

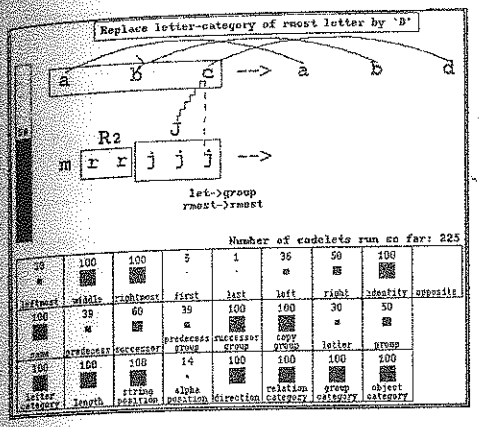
4. Groups ***abc*** and ***jjj*** have been built (the bonds between their letters still exist, but for the purposes of graphical simplicity are no longer being displayed). An ***rr*** copy-group is being considered. The already-built copy-group ***jjj*** strongly supports this potential move, which accelerates it, in the sense that codelets investigating the potential structure will be assigned higher urgency values.

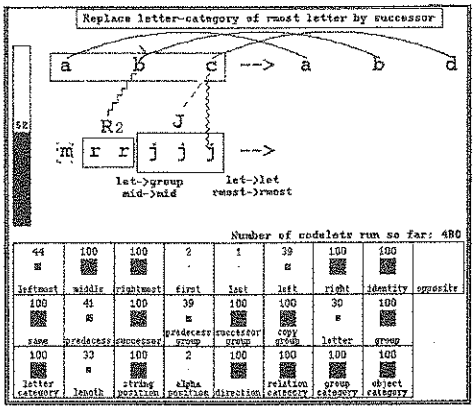
Meanwhile, a *rule* (shown at the top of the screen) has been constructed to describe how ***abc*** changed. The current version of Copycat assumes that the example change involved the replacement of exactly one letter, so rule-building codelets fill in the template "Replace ____ by ____", choosing probabilistically from descriptions that the program has attached to the changed letter and its replacement, with a probabilistic bias toward choosing more abstract descriptions (*e.g.*, usually preferring *rightmost letter* to *c*).

Since the nodes *first* and *alphabetic position* didn't turn out useful, they have faded. Also, although *length* received additional activation from *group*, it is still not very activated, and so lengths are still unlikely to be noticed.



been built (the
still exist, but for
simplicity are no
a *rr* copy-group is
ready-built copy-
ts this potential
in the sense that
otential structure
ency values.
a at the top of the
cted to describe
rrent version of
example change
of exactly one let-
ts fill in the tem-
____", choosing
riptions that the
ie changed letter
probabilistic bias
ract descriptions
utmost letter to ***c***).
alphabetic position
have faded. Also,
additional activa-
ot very activated,
ely to be noticed.





5. Now, some 225 codelets into the run, the letter-to-*letter* ***c–j*** bridge has been defeated by the stronger letter-to-*group* ***c–J*** bridge, although the former possibility still lurks in the background. Meanwhile, an ***rr*** copy-group has been built whose length (namely, 2) happened to be noticed (a probabilistic event); therefore, a "2", along with the group's letter category (namely, ***r***), is displayed just above the group. *Length* is now fully active, and for this reason the "2" is a salient Workspace object (indicated by boldface).

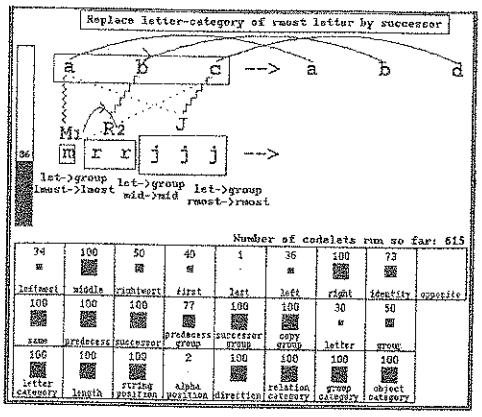
A new rule, "Replace the letter category of the rightmost letter by ***d***", has replaced the old one at the top of the screen. Although this rule is weaker than the previous one, fights between rival structures (including rules) are decided probabilistically, and this one simply happened to win. However, its weakness has caused the temperature to go up.

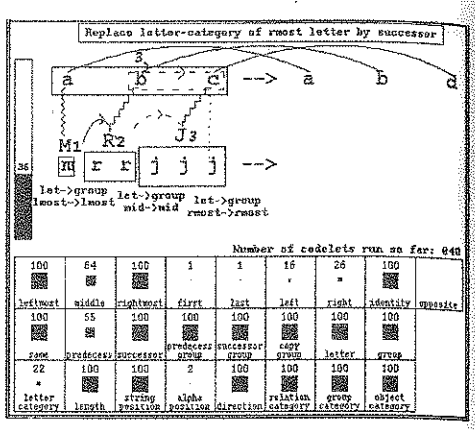
If the program were to stop now (which is quite unlikely, since a key factor in the program's probabilistic decision when to stop is the temperature, which is now quite high), the rule would be adapted for application to the string ***mrrjjj*** as "Replace the letter category of the rightmost *group* by ***d***" (the ***c–J*** bridge establishes that the role of *letter* in ***abc*** is played by *group* in ***mrrjjj***), yielding the answer ***mrrddd*** (an answer that Copycat does indeed produce, on rare occasions).

6. The previous, stronger rule has been restored (again the result of a fight having a probabilistic outcome), but at the same time, the strong ***c–J*** bridge also happened to get defeated by its weaker rival, the ***c–j*** bridge. As a consequence, if the program were to stop at this point, its answer would be ***mrrjjk***. This, incidentally, would also have been the answer in screen dump #4.

In the Slipnet, the activation of *length* has decayed a good deal, since the length description given to ***rr*** wasn't found to be useful. In the Workspace, the diminished salience of the ***rr***'s length description "2" is represented 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***, something that it did easily with ***abc***. That continuing difficulty, combined with strong top-down pressure from the two copy-groups that have been built inside ***mrrjjj***, makes it now somewhat tempting for the system to flirt with the *a priori* extremely unlikely idea of making a single-letter copy-group (this flirtation is represented by the dashed rectangle around the letter *m*).

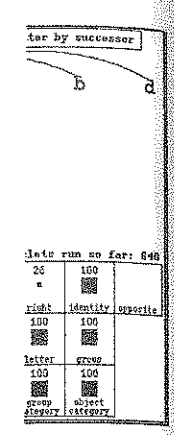




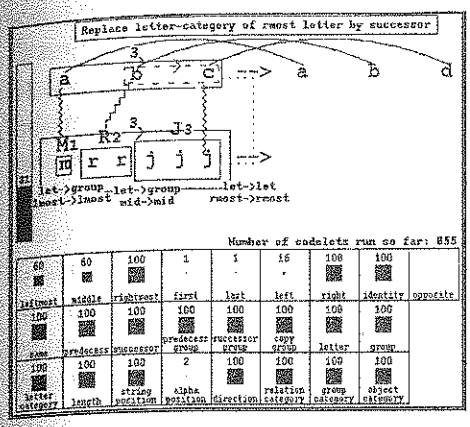
7. As a result of these combined pressures, the *a priori* extremely unlikely single-letter copy-group ***m*** happened to get built, and its length of 1, being very noteworthy, has been attached to the group as a description. A successorship bond between that "1" and its right-neighbor "2" has already been built; all of this is helping *length* to stay active. A consistent trio of *letter* ⇒ *group* bridges has now been made, and as a result of these promising new structures, the temperature has fallen to the relatively low value of 36, which in return helps to lock in this emerging view.

If the program were to halt in this screen dump or in the following one, it would produce the answer ***mrrkkk***, which is its most frequent answer.

8. As a result of *length*'s continued activity, length descriptions have been attached to the remaining two groups in the problem (***jjj*** and ***abc***), and a successorship bond 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 (dashed arc). *Letter category* has decayed, indicating that it hasn't lately been of use in building structures.

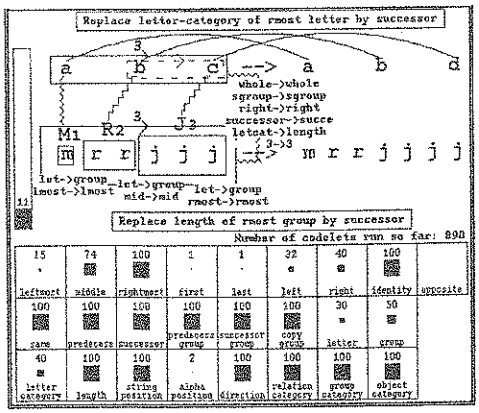


inued activity,
n attached to
the problem
sorship bond
or which there
coming from
w of *mrrjjj*) is
. *Letter category*
t hasn't lately
ures.



9. The "2"–"3" bond was built, whereupon a very abstract, high-level numerical successor group involving the group lengths in *mrrjjj* was perceived (large solid rectangle surrounding the three copy-groups). Also, a bridge (dotted vertical line to the right of the two strings) is being considered between strings *abc* and *mrrjjj* in their entireties.

Ironically, just as these sophisticated ideas seem to be converging toward a highly insightful answer, a small renegade codelet, totally unaware of the global momentum, has had some good luck: its bid to knock down the *c–J* bridge and replace it with a *c–j* bridge was accepted. Of course, this is a setback on the global level. If the program were forced to stop at this point, it would answer *mrrjjk* — the same rather dull answer as it would have given in screen dumps #6 and #4. However, at either of those stages, that answer would have been far more excusable than it is now, since the program hadn't yet made the subtle discoveries it has now made about the structure of *mrrjjj*. It would seem a shame for the program to have gotten this far and then to "drop the ball" and answer in a relatively primitive way. However, 31 is a high enough temperature that there is a good chance that the program will get back on track and be allowed to explore the more abstract avenue to its logical conclusion.



10. Indeed, the aberrant bridge from the *c* was quickly destroyed and replaced by a rebuilt *c–J* bridge, in keeping with the emerging sophisticated view. Also, the high-level bridge between *abc* and *mrrjjj* as wholes, which in the previous screen dump was merely a dotted candidate, has now been promoted through the dashed state and actually built. Its six component concept-mappings, including identity mappings (such as *right* ⇒ *right*, meaning that both strings are seen as flowing rightwards) as well as conceptual slippages (such as *letter category* ⇒ *length*, meaning that letters are mapped onto numbers — specifically, onto group lengths), are listed in the middle of the screen, somewhat obscuring the bridge itself.

The original rule has been translated, according to the slippages, for application to the target string *mrrjjj*. The translated rule, "Replace the length of the rightmost group by its successor", appears just above the Slipnet, and the answer *mrrjjjj* at the right. The very low final temperature of 11 reflects the program's unusually high degree of satisfaction with this answer.

Although the preceding run may look quite smooth, there were many struggles involved in coming up with this answer: it was hard not only to make a single-letter group, but also to bring the notion of *length* into the picture, to allow it to persist long enough to trigger the noticing of all three group lengths, and to build bonds between the group lengths. The program, like people, usually gives up before all these hurdles can be overcome, and gives one of the more obvious answers. Arriving at the deeper answer ***mrrjjjj*** requires not only the insights brought about by the strong pressures in the problem, but also a large degree of patience and persistence in the face of uncertainty.

The moral of all this is that in a complex world (even one with the limited complexity of Copycat's microworld), one never knows in advance what concepts may turn out to be relevant in a given situation. This dilemma underscores the point made earlier: it is 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, in which it quite literally takes calculated risks all the time — but the *degree* of risk-taking is carefully controlled. Of course, taking risks by definition opens up the potential for disaster — and indeed, disaster occurs once in a while (as was evidenced by some of the far-fetched answers displayed earlier). But this is the price that must be paid for flexibility and the potential for creativity.

People, too, occasionally explore and even favor peculiar routes. Copycat, like people, has to have the potential to concoct strange and unlikely solutions in order to be able to discover subtle and elegant ones like ***mrrjjjj***. 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 strange answers demonstrates that its mechanisms manage to strike an effective balance between open-mindedness and closed-mindedness, imbuing it with both flexibility and robustness.

Hopefully, these screen dumps have made clearer the fundamental roles of nondeterminism, parallelism, non-centralized and simple perceptual agents (*i.e.*, codelets), the interaction of bottom-up and top-down pressures, and the reliance on statistically emergent (rather than explicitly programmed) high-level behavior. Large, global, deterministic decisions are never made (except perhaps towards the end of a run). The system relies instead on the accumulation of small, local, nondeterministic decisions, none of which alone is particularly important for the final outcome of the run. As could be seen in the screen dumps, large-scale effects occur only through the statistics of the lower levels: the ubiquitous notion of a "pressure" in the system is really a shorthand for the statistical effects over time of a large number of actions carried out by codelets, on the one hand, and of activation patterns of nodes in the Slipnet, on the other.

As was seen in the screen dumps, as structures are formed and a global interpretation coalesces, the system gradually makes a transition (via gradually falling temperature) from being quite parallel, random, and dominated by bottom-up forces to being more serial, deterministic, and dominated by top-down forces. We believe that such a transition is characteristic of high-level perception in general.

The importance of such a transition in degree of risk-taking was confirmed by two experiments we performed on the model (described in Mitchell, 1993). Temperature was clamped throughout a run, in one experiment at a very high value and in the other at a very low value. In each experiment, 1,000 runs were made. In neither test did the hobbled Copycat ever come up with the answer *mrrjjjj* — not even once.

Micro-anatomy of a paradigm shift

We now turn our attention from Problem 4 to Problem 6, and give a sketch of how Copycat can, on occasion, come up with the answer *wyz*. It turns out to be a surprisingly intricate little tale. The reason for this is that it is an attempt to show in slow motion how a human mind, under severe pressure, can totally transform its perception of a situation in a blinding flash (colloquially termed the "Aha!" phenomenon). Since such paradigm shifts are often found at the core of deeply creative acts, one should expect their microstructure to be very complex (otherwise, the mystery of creativity would long ago have been revealed and put on a mass-produced microchip). Indeed, the challenge of getting Copycat to produce *wyz properly* — faithfully to what we believe really goes on in a human mind at the most informative subcognitive level of description — has, from the very outset, been the central inspiration in guiding the development of the Copycat architecture.

The very cursory justification for *wyz* given in the previous section lies at far too high and coarse-grained a level to be informative about the mental mechanisms responsible for paradigm shifts. The following detailed story, by contrast, evolved hand in hand with the architecture itself, and is intended not only as a description of Copycat, but hopefully as an accurate description of the underpinnings of a typical paradigm shift in a human mind. (An annotated series of screen dumps of a particular run on Problem 6 is given in Mitchell & Hofstadter, 1990a.)

Emergency measures convert a serious snag into a set of exploratory pressures

Things start out essentially analogously to a typical run on Problem 1, in terms of bonding, grouping, bridge-building, and such — that is, both source and target strings come quite quickly to be perceived as successor groups, and the raw rule "Replace rightmost letter by its successor" is effortlessly produced.

Everything thus proceeds pretty smoothly up to the point of trying to take the successor of *z*, which is impossible. This serious snag causes several coordinated "emergency measures" to be taken:

- the *physical* trouble spot — here, the instance of *z* in the Workspace — is highlighted, in the sense that its salience is suddenly pumped up so high that, to codelets, it becomes the most attractive object in the entire Workspace;
- the *conceptual* trouble spot — here, the node *z* in the Slipnet — is highlighted, in the sense that a huge jolt of activation is pumped into it, and as a consequence, its halo broadens and intensifies, meaning that related concepts are more likely to be considered, at least fleetingly;
- the temperature is pumped up to its maximum value of 100 and temporarily clamped there, thus encouraging a broader and more open-minded search;
- the high temperature enables previously dormant "breaker" codelets to run, whose purpose is to arbitrarily break structures that they find in the Workspace, thus reducing the system's attachment to a viewpoint already established as being problematic.

Note the generality of these "impasse-handling" mechanisms: they have nothing to do with this snag itself, with the particular problem, with the alphabetic domain, or even with analogy-making! The reason for this is of course that running into an impasse is a critical and common event that any cognitive system must be capable of dealing with. To be sure, no set of mechanisms can be guaranteed to resolve *all* snags (otherwise we would be dealing with omniscience, not intelligence). The best that can be hoped for is that the impasse itself can be "read" as a source of *cues* — possibly very subtle ones — that may launch tentative forays down promising new avenues. A "cue", in the Copycat architecture, is essentially the creation of a *pressure* that pushes for exploration along a certain direction. Thus the idea of *interpreting the snag as a source of pressures* is the philosophy behind the four mechanisms above, especially the first two.

Although these emergency measures are not powerful enough to guide Copycat to coming up with *wyz* all that often, when it does get there, it does so essentially according to the following scenario.

The spotlight focused on Platonic *z* has the effect of making all concepts in *z*'s halo — including the closely-related concept *alphabetic-last* — somewhat more likely to be looked at by description-building codelets. The probability is thus significantly increased that the instance of *z* in the Workspace will get explicitly described as *alphabetic-last*.

Note that in most problems — even ones that involve one or more instances of the letter *z* — there is little or no reason to pay attention to the notion *alphabetic-last*, and therefore, this conceptually deep neighbor of Platonic *z* usually remains — and *should* remain — dormant (as it did in Problems 1–5). After all, as was brought out in the discussion of Problem 4, it is extremely crucial to avoid cluttering up the processing with all sorts of extraneous interfering notions, no matter how conceptually deep they may be. But now, under the emergency measures, unusual avenues are more likely to at least be "sniffed out" a short ways.

If the description *alphabetic-last* does indeed get attached to the *z*, which is a dicey matter, then a further boost is given to the node *alphabetic-last*, as the system has deemed it potentially relevant. So now that *alphabetic-last* (part of the halo of Platonic *z*) has been lifted considerably out of dormancy, concepts in *its* halo will in turn receive more activation, which means codelets will tend to pay more attention to them (probabilistically speaking). One such neighbor-concept is *alphabetic-first*, which is now briefly given the chance to show its relevance. Obviously, if there were no instance of ***a*** in the problem, *alphabetic-first* would be found to be completely irrelevant and would soon decay back to dormancy, but since there *is* an ***a*** inside ***abc***, it has a fair chance of getting explicitly described as *alphabetic-first*, in much the same way as the *z* in *xyz* got described as *alphabetic-last*.

If both these descriptions get attached — and that is a big "if" — then both letters become even more salient than before; in fact, they almost cry out to be mapped onto each other — not because the system can anticipate the great insight that such a mapping will bring, but simply because both letters are so salient! Once the system tries it out, however, the great appeal of the tentative mapping instantly becomes apparent. Specifically, a pair of conceptual slippages are entailed in the act of "equating" the ***a*** with the *z* (*i.e.*, building an ***a***–*z* bridge): *alphabetic-first* ⇒ *alphabetic-last*, and *leftmost* ⇒ *rightmost*.

How resistance to a deep slippage is overcome — a tricky matter

Although normally the deep slippage of *alphabetic-first* into *alphabetic-last* would be quite valiantly resisted (recall the motto given earlier, "Deep stuff doesn't slip in good analogies"), here a special circumstance renders it a bit more probable: the companion would-be slippage *rightmost* ⇒ *leftmost* is of the same type — in particular, each of these slippages involves slipping a concept representing an extremity into its opposite concept. These two would-be slippages are thus conceptually parallel, so that each one on its own reinforces the other's plausibility. This fact helps to overcome the usual resistance to a deep slippage. (Incidentally, this is the kind of subtlety that was not apparent to us before the computer implementation was largely in place; only at that point

were Copycat's flailings and failures able to give us pointers as to what kinds of additional mechanisms were needed.)

Another fact that helps overcome the usual resistance to the deep slippage in this bridge is that *any* two slippages, whether parallel or not, provide more justification for building a bridge than either one alone would. Altogether, then, there is a fairly good chance that this bridge, once tentatively suggested, will actually get built. Once this critical step has taken place, essentially it's all downhill from there. This is why we have paid particularly close attention to the pathway via which such a bridge can emerge.

Locking-in of a new view

The first thing that is likely to happen as a result of an *a–z* bridge getting built is that the temperature will get unclamped from its value of 100. In general, what unclamps the temperature is the construction of any strong structure different from those that led up to the snag — in other words, a sign that an alternative way of looking at things may be emerging — and this bridge is a perfect example of such a structure. Like most actions in Copycat, the unclamping of temperature is probabilistic. In this case, the stronger the novel structure is, the more likely it is to trigger the unclamping. Since the *a–z* bridge is both novel and very strong, unclamping is virtually assured, which means that the temperature falls drastically right after the bridge is built. And when the temperature falls, decisions tend to get more deterministic, which means that the emerging new view will tend to get supported. In short, there is a powerful kind of *locking-in* effect that is triggered by the discovery of an *a–z* bridge. This is a critical effect.

Another aspect of locking-in is the following idea. The building of this first bridge involving the simultaneous slippage of two concepts into their opposites sends a burst of activation into the very deep concept *opposite*; as a result, all pairs of concepts connected via links labeled *opposite* are drawn much closer together, facilitating the slippage of one into the other. Such slippages will still not happen without reason, of course, but now they will be much easier to make than in ordinary circumstances. Thus in a sense, making *one* bridge based on conceptual opposites sets a tone making it easier to make *more* of them. The emerging theme of the concept *opposite* can fairly be characterized as a kind of "bandwagon".

Given all this, one of the most likely immediate consequences of the crosswise *a–z* bridge is the building of the "mirror" crosswise bridge connecting the *c* with the *x*. It, too, depends on the slippage between *leftmost* and *rightmost*, and is thus facilitated; in addition, once built, it strongly reinforces the emerging relevance of the concept *opposite*. Moreover, the temperature will fall significantly because this bridge, too, will be very strong. Thanks to all of this, the

locking-in effect may by now be so strong that it will be hard to stop the momentum towards building a completely new view of the situation.

The reversals taking place become a near-stampede at this point, with significant pressure emerging to flip the direction of the fabric of the group *xyz* from *rightwards* to *leftwards,* which means also that the perceived fabric itself would switch from *successor* to *predecessor.* Thus at this point, Copycat has carried out both a spatial and an alphabetical reversal of its vision of *xyz.* The paradigm shift has been completed. At this point, Copycat is ready to translate the raw rule, and, as was said above, the result is the new rule *replace the leftmost letter by its alphabetic predecessor,* which yields the answer *wyz.*

It must be stressed that all the multifarious activity just described — shifting degrees of activation of various key concepts; deep slippages; interrelated spatial and conceptual reversals — all this takes place in a flash in a human mind. There is no hope of making out all the details of this paradigm shift (or any other) in one's own mind through mere introspection. In fact, it has taken the authors several years to settle on the above account, which represents our current best stab at the true story's intimate details.

How hard is it to make this paradigm shift?

As was pointed out a moment ago, the motto "Deep stuff doesn't slip in good analogies" is violated by the answer *wyz,* in that *alphabetic-first* is a deep concept and yet is allowed to slip into *alphabetic-last* here. This is one reason that makes it so hard for many people to discover it on their own. Yet many people, when they are shown this answer, appreciate its elegance and find it very satisfying. Problem 6 is thus a circumstance where a constellation of pressures can occasionally overcome the powerful natural resistance expressed by the motto; in fact, making such a daring move results in what many people consider to be a deep and insightful analogy.

There is an important irony here. In particular, even though slippages tend to be (and should be) *resisted* in proportion to their depth, once a very deep slippage has been made, then it tends to be (and should be) *respected* in proportion to its depth. We consider this to be characteristic of creative breakthroughs in general. More specifically, we consider the process of arriving at answer *wyz* to be very similar, on an abstract level, to the process whereby a full-scale conceptual revolution takes place in science (Kuhn, 1970).

Now we come back to the point raised in our earlier discussion of Problem 6 about "levels of subtlety" of answers. Specifically, we claimed above that, because finding the answer *wyz* to Problem 6 is far subtler *for people* than finding the similar answer *hjkk* to Problem 2, any model of mental fluidity should respect this difference in levels of subtlety. Yet when one compares the bar graphs for these problems, one discovers that *wyz* was found far more often than *hjkk* was

found (a ratio of 137 to 47, when both problems were run 1,000 times). This seems to completely contradict the claim that the former answer is subtler than the latter. How can one account for this unexpected ratio?

There are two basic factors that explain it. The first has to do with the fact that there was a snag in one problem and no snag in the other. In attacking Problem 6, Copycat was *forced* to look for solutions other than taking the successor of the rightmost letter, because that route turned out to be impossible. By contrast, all sorts of superficially attractive routes led directly to solutions in Problem 2. There was no snag that prevented any attractive route from being taken all the way to its natural conclusion. Had all or most of the easy routes been barred, then of course ***hjkk*** would have constituted a much larger percentage of the answers found.

The second factor is that the average *length of time* taken to find various solutions (measured in terms of number of codelets run) is a key notion. This fact is not apparent, because average run-lengths are unfortunately not represented in the bar graphs. When Copycat came up with ***hjkk*** in Problem 2, it was essentially always a relatively direct process involving no backtracking or getting stuck for a while in a loop. To be specific, the average number of codelets taken to get ***hjkk*** was 899. By contrast, the average number of codelets taken to get ***wyz*** was 3,982 — over four times as long. The reason for this is that in most runs, the program came back time and time again to the standard way of looking at ***xyz***, and thus hit the snag over and over again: it was stuck in a kind of rut. This means that on runs where Copycat was lucky enough to come across the double reversal, by the time it did so it had usually tried out all sorts of other pathways in vain beforehand. In this sense of *time needed to make the discovery*, ***wyz*** was an extremely elusive answer for the program, whereas ***hjkk*** was not at all elusive. In sum, ***wyz*** was indeed far subtler for Copycat than ***hjkk*** was, as ought to have been the case.

Conclusion: The Generality of Copycat's Mechanisms

The crucial question of scaling-up

As was stated at the outset, the Copycat project was never conceived of as being dependent in any essential way on specific aspects of its small domain, nor even on specific aspects of analogy-making *per se*. Rather, the central aim was to model the emergence of insightful cognition from fluid concepts, focusing on how slippages can be engendered by pressures.

One of the key questions about the architecture, therefore, is whether it truly is independent of the small domain and the small problems on which it now works. It would certainly be invalidated if it could be shown to depend on the relative smallness of its repertoire of Platonic concepts and the relatively

few instances of those concepts that appear in a typical problem. However, from the very conception of the project, every attempt has been made to ensure that Copycat would not succumb to a combinatorial explosion if the domain were enlarged or the problems became bigger. In some sense, Copycat is a caricature of genuine analogy-making. The question is, what makes a caricature faithful? What is the proper way to construct a cognitive model that will scale up?

Shades of gray and the mind's eye

Real cognition of course occurs in the essentially boundless real world, not in a tiny artificial world. This fact seems to offer the following choice to would-be "cognition architects": either have humans scale down all situations by hand in advance into a small set of sharp-edged formal data-structures, so that a brute-force architecture can work, or else let the computer effectively do it instead — that is, use a heuristic-based architecture that at the outset of every run makes a sharp and irreversible cut between concepts, pathways, and methods of attack that might eventually be brought to bear during that run, and ones that might not. There seems to be no middle ground between these two types of strategy, because either you must be willing to give *every* approach a chance (the brute-force approach), or you must choose some approaches while *a priori* filtering others out (the "heuristic-chop" approach).

The only way out would seem to involve a notion of "shadedness", in which concepts, facts, methods of attack, objects, and so on, rather than being ruled "out" or "in" in a black-and-white way, would be present in shades of gray — in fact, shades of gray that change over time. At first glance, this seems impossible. How can a concept be invoked only *partially*? How can a fact be neither fully ignored nor fully paid attention to? How can a method of attack be merely "sort of" used? How can an object fall somewhere in between being considered "in the situation" and being considered "not in the situation"?

Since we believe that these "shades of gray" questions lie at the crux of the modeling of mind, they merit further discussion. A special fluid quality of human cognition is that often, solutions to a problem — especially the most ingenious ones, but even many ordinary ones — seem to come from far outside the problem as conceived of originally. This is because problems — or more generally, *situations* — in the real world do not have sharp definitions; when one is in, or hears about, a complex situation, one typically pays no conscious attention to the question of what counts as "in" the situation and what counts as "out" of it. Such matters are almost always vague, implicit, and intuitive.

Using the metaphor of the "mind's eye", we can liken the process of considering an abstract situation to the process of visually perceiving a physical scene. Like a real eye, the mind's eye has a limited field of vision, and cannot focus on several things simultaneously, let alone large numbers of them. Thus one has to

choose where to have the mind's eye "look". When one directs one's gaze at what one feels is the situation's core, only a few centrally located things will come into clear focus, with more tangential things being less and less clear, and then at the peripheries there will be lots of things of which one is only dimly aware. Finally, whatever lies beyond the field of vision seems by definition to be outside of the situation altogether. Thus "things" in the mind's eye are definitely shaded, both in terms of how clear they are, and in terms of how aware one is of them.

The very vague term "thing" was used deliberately above, with the intent of including both *abstract Platonic concepts* and *concrete specific individuals* — in fact, to blur the two notions, since there is no hard-and-fast distinction between them. To make this clearer, think for a moment of the very complex situation that the Watergate affair was. As you do this, you will notice (if you followed Watergate at all) that all sorts of different events, people, and themes float into your mind with different degrees of clarity and intensity. To make this even more concrete, turn your mind's eye's gaze to the Senate Select Committee, and try to imagine each different senator on that committee. Certainly, if you watched the hearings on television, some will emerge vividly while others will remain murky. Not just *Platonic abstractions* like "senator" are involved, but many *individual* senators have different degrees of mental presence as you attempt to "replay" those hearings in your mind. Needless to say, the memory of anyone who watched the Watergate hearings on television is filled to the brim both with Platonic concepts of various degrees of abstractness (ranging from "impeachment" to "coverup" to "counsel" to "testimony" to "paper shredder") and with specific events, people, and objects at many levels of complexity (ranging from the "Saturday night massacre" to the Supreme Court, from Maureen Dean to the infamous 18½-minute gap, and all the way down to the phrase "expletive deleted" and even Sam Ervin's gavel, with which every session of the committee was rapped to order). When one conjures up one's memories of Watergate, all of these "things" have differential degrees of mental presence, which change as one's mind's eye scans the "scene".

Note that in the preceding paragraph, all the "things" mentioned were carefully chosen so that readers — at least readers who remember Watergate reasonably well — would give them unthinking acceptance as genuine "parts" of Watergate. However, now consider the following "things": England, France, communism, socialism, the Viet Nam War, the Six-Day War, the Washington Monument, the *New York Times*, Spiro Agnew, Edward Kennedy, Howard Cosell, Jimmy Hoffa, Frank Sinatra, Ronald Reagan, the AFL–CIO, General Electric, the electoral college, college degrees, Harvard University, helicopters, keys, guns, flypaper, Scotch tape, television, tape recorders, pianos, secrecy, accountancy, loyalty, and so on. Which of these things are properly thought of as being "in" Watergate, and which ones as "out" of it? It would obviously be ludicrous

to try to draw a sharp line. One is forced to accept the fact that for a model of a mind to be at all realistic, it must be capable of imbuing all concrete objects and individuals, as well as all abstract Platonic concepts, with shaded degrees of mental presence — and of course, those degrees of presence must be capable of changing over time.

Like a real eye, the mind's eye can be attracted by something glinting in the peripheries, and shift its gaze. When it does so, things that were formerly out of sight altogether now enter the visual field. By a series of such shifts, "things" that were totally outside of the situation's initial representation can eventually wind up at the very center of attention. This brings us back, finally, to that special fluid quality of human thought whereby initially unsuspected notions occasionally wind up being central to one's resolution of a problem, and reveals how intimately such fluidity is linked with the various "shades of gray" questions given above.

Copycat's shaded exploration strategy

Let us thus return to the list of "shades of gray" questions: How can a concept be invoked only *partially*? How can a fact be neither fully ignored nor fully paid attention to? How can a method of attack be merely "sort of" used? How can an object fall somewhere in between being considered "in the situation" and being considered "not in the situation"? These questions were not asked merely rhetorically; in fact, it was precisely to respond to the challenges that they raise that the probabilistic architecture of Copycat was designed.

Copycat's architecture has in common with brute-force architectures the fact that every possible concept, fact, method, object, and so on is *in principle* available at all times;[7] on the other hand, it has in common with heuristic-chop architectures the fact that out of all available concepts, facts, methods, objects, and so on, only a few will get very intensely drawn in at any given moment, with most being essentially dormant and an intermediate number having a status somewhere in between. In other words, virtually all aspects of the Copycat architecture are riddled by shades of gray instead of by hard-edged, black-and-white cutoffs. In particular, *activation* (with continuous values rather than a binary on/off distinction) is a mechanism that gives rise to shadedness in the Slipnet, while *salience* and *urgency* serve similar purposes in the Workspace and Coderack, respectively. These are just three of a whole family of related "shades-of-gray mechanisms" whose entire *raison d'être* is to defeat the scaling-up problem.

7. Note that the claim is not that every single concept imaginable to humans is available, but simply that all concepts *within the system's dormant repertoire* are in principle accessible at any point during a run. The fact that Copycat cannot reach beyond its own conceptual repertoire, thus effectively "transcending itself", is not a defect, but simply a fact of existence that it shares with every finite cognitive system, such as human minds. Put another way, if this property is a defect of Copycat, then it is a defect that Copycat shares with human minds.

An architecture thus pervaded by shades of gray has the very attractive property that although no concept or object or pathway of exploration is ever *strictly* or *fully* ruled out, only a handful of them are at any time *seriously* involved. At any given moment, therefore, the system is focusing its attention on just a small set of concepts, objects, and pathways of exploration. However, this "searchlight of attention" can easily shift under the influence of new information and pressures, allowing *a priori* very unlikely concepts, objects, or pathways of exploration to enter the picture as serious contenders.

The chart below summarizes the various mechanisms in the Copycat architecture that incorporate shades of gray in different ways. In it, the term "shaded" should be understood as representing the opposite of a binary, black/white distinction; it often means that one or more *real numbers* are attached to each entity of the sort mentioned, as opposed to there being an on/off distinction. The term "dynamic" means that the degree of presence — the "shade", so to speak — can change with time.

Shades of gray in the Slipnet

- shaded, dynamic presence of Platonic concepts (via dynamic activation levels)
- shaded, dynamic conceptual proximities (via dynamic link-lengths)
- shaded, dynamic spreading of activation to neighbor concepts (giving rise to "conceptual halos")
- shaded conceptual depths of nodes
- shaded decay rates of concepts (determined by conceptual depths)
- shaded, dynamic emergence of abstract themes (stable activation patterns of interrelated conceptually deep nodes)

Shades of gray in the Workspace

- shaded, dynamic number of descriptions for any object
- shaded, dynamic importance of each object (via activation levels of descriptors in Slipnet)
- shaded, dynamic unhappiness of each object (determined by degree of integration into larger structures)
- shaded, dynamic presence of objects (via dynamic salience levels)
- shaded, dynamic tentativity of structures (via dynamic strengths)

Shades of gray associated with the Coderack

- shaded, dynamic degrees of "promise" of pathways

- shaded, dynamic emergence of pressures (via urgencies of codelets and shifting population of Coderack)
- shaded, dynamic degree of willingness to take risks (via temperature)
- shaded, dynamic mixture of deterministic and nondeterministic modes of exploration
- shaded, dynamic mixture of parallel and serial modes of exploration
- shaded, dynamic mixture of bottom-up and top-down processing

There is one further aspect of shadedness in Copycat that is not localized in a single component of the architecture, and is somewhat subtler. This has to do with the fact that, over time, higher-level structures emerge, each of which brings in new and unanticipated concepts, and also opens up new and unanticipated avenues of approach. In other words, as a run proceeds, the field of vision broadens out to incorporate new possibilities, and this phenomenon feeds on itself: each new object or structure is subject to the same perceptual processes and chunking mechanisms that gave rise to it. Thus there is a spiral of rising complexity, which brings new items of ever-greater abstraction into the picture "from nowhere", in a sense. This process imbues the Copycat architecture with a type of fundamental unpredictability or "openness" (Hewitt, 1985) that is not possible in an architecture with frozen representations. The ingredients of this dynamic unpredictability form an important addendum to the list of shades of gray given above.

Dynamic emergence of unpredictable objects and pathways

- creation of unanticipated higher-level perceptual objects and structures
- emergence of *a priori* unpredictable potential pathways of exploration (via creation of novel structures at increasing levels of abstraction)
- creation of large-scale viewpoints
- competition between rival high-level structures

By design, none of the mechanisms in the lists presented above has anything in the least to do with the size of the situations that Copycat is currently able to deal with, or with the current size of Copycat's Platonic conceptual repertoire. Note, moreover, that none of them has anything whatsoever to do with the subject matter of the Copycat domain, or even with the task of analogy-making *per se*. Yet these mechanisms and their emergent consequences — especially commingling pressures and the parallel terraced scan — are what Copycat is *truly* about. This is the underpinning of our belief in the cognitive generality of the Copycat architecture.