The Role of Computational Temperature in a Computer Model of Concepts and Analogy-Making

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

ABSTRACT

We discuss the role of computational temperature in Copycat, a computer model of the mental mechanisms underlying human concepts and analogy-making. In Copycat, computational temperature is used both to *measure* the amount and quality of perceptual organization created by the program as processing proceeds, and, reciprocally, to continuously *control* the degree of randomness in the system. We discuss these roles in two aspects of perception central to Copycat's behavior: (1) the emergence of a *parallel terraced scan*, in which many possible courses of action are explored simultaneously, each at a speed and to a depth proportional to moment-to-moment estimates of its promise, and (2) the ability to restructure initial perceptions — sometimes radically — in order to arrive at a deeper understanding of a situation. We compare our notion of temperature to similar notions in other computational frameworks. Finally, we give an example of how temperature is used in Copycat's creation of a subtle and insightful analogy.

1. DESCRIPTION OF THE PROJECT

In our research, we are attempting to model the mental mechanisms underlying the fluid nature of human concepts. Humans are able to perceive and categorize situations very flexibly, to see beyond superficial details and understand the essence of a situation, and to make analogies between situations, fluidly translating concepts from one situation into the other. These abilities are central to every facet of human intelligence, from perception and learning, to recognition of concrete and abstract objects and situations (faces, letters of the alphabet, artistic and musical styles), and even to acts of great insight and creativity.

In order to isolate and study the mechanisms underlying these abilities, we have developed a microworld in which analogies are to be made between idealized situations consisting of strings of letters. We believe that analogy-making in this microworld requires the essence of abilities central to perception and analogy-making in real-world situations. A simple analogy problem is this: If the string **abc** changes to **abd**, what is the analogous change for **ijk**? A reasonable description of the initial change is "Replace the rightmost letter by its successor", and straightforward application of this rule to the target string **ijk** yields the commonsense answer **ijl** (other, less satisfying answers, such as **ijd**, are of course possible). However, given the alternate target string **iijjkk**, a straightforward, rigid application of the original rule would yield **iijjkl**, which ignores the strong similarity between **abc** and **iijjkk** when the latter is seen as consisting of three *letter-groups* rather than as six *letters*. If one perceives the role of *letter* in **abc** as played by *letter-group* in **iijjkk**, then in making a mapping between **abc** and **iijjkk** one is forced to let the concept *letter* "slip" into the similar concept *letter-group*. The ability to make appropriate *conceptual slippages* — in which concepts in one situation are identified with similar concepts in a different but analogous situation — is central to analogy-making and to cognition in general (Hofstadter, 1985), and our research centers on investigating how concepts must be structured and how perception must interact with concepts to allow the fluidity necessary for insightful slippages.

The letter-string microworld was designed to capture the essence of the issues of concepts and perception that we are investigating. Although the analogies in this microworld involve only a small number of concepts, they often require considerable insight. An example of such an analogy is the following: if **abc** changes to **abd**, what does **xyz** change to? At first glance, this problem is essentially the same as the one with target string **ijk** discussed above, but there is a snag: **Z** has no successor. (Notational note: in this discussion, lowercase boldface letters designate *instances* of letter categories, and uppercase boldface letters designate the categories themselves. For example,

z is an instance of the category **Z**.) Many people answer **xya**, but in our microworld the alphabet is not circular; this answer is intentionally excluded since the snag forces the analogy-maker to restructure their original view, to make conceptual slippages that were not initially considered, and hopefully to discover a more useful and insightful way of understanding the situation. One such way is to notice that **abc** is "wedged" against the beginning of the alphabet, and **xyz** is similarly wedged against the far end of the alphabet. Thus the **A** in **abc** and the **Z** in **xyz** can be seen to correspond, and then one naturally feels that the **C** and the **X** correspond as well. Underlying these object correspondences is a set of conceptual slippages that are mutually parallel: *alphabetic-first* ⇒ *alphabetic-last*, *right* ⇒ *left*, and *successor* ⇒ *predecessor*, which together yield an insightful answer: **wyz**. (For a detailed discussion of the microworld and a large number of sample analogy problems, see Mitchell, 1988.)

This example illustrates how problems in the microworld can contain the essence of many issues central to perception in general: in order to understand a situation, one must choose from a large number of possible ways in which the objects in the situation can be described and related to one another, and in which similarity to other situations can be perceived. It must be decided which concepts are relevant to the situation at hand, what is salient and what can be ignored, at what level of abstraction to describe objects, relations, and events, which descriptions to take literally and which to allow to slip, and so on. And if these choices lead to a impasse that seems to block progress towards understanding, then one may be required to fluidly restructure one's original perceptions, to shift one's view in unexpected ways, and hopefully to arrive at a deeper, more essential understanding of the situation. We are developing a computer model of the mental mechanisms we believe underlie these abilities, in which a notion of *temperature* has a central role.

2. THE ARCHITECTURE OF COPYCAT

Our computer model, called "Copycat", solves analogy problems in the microworld. (Earlier versions of the program have been described by Hofstadter, 1984, and Hofstadter & Mitchell, 1988a and 1988b.) In Copycat, concepts are modeled using what we call a "Slipnet": a network in which a node represents the "core" of a concept (e.g., *first*) and a link simultaneously represents a resemblance or relationship between two nodes and a potential slippage from one to the other. For example, *first* is the opposite of *last*, and thus in some circumstances they are similar and one can be slipped to the other. Each link has a label that roughly classifies the resemblance or relationship the link encodes. Each type of label is itself represented by a node. Thus, the nodes *first* and *last* are connected by a link with label *opposite*. During a run of the program, nodes become activated when perceived to be relevant, and decay when no longer perceived as relevant. Nodes also spread activation to their neighbors. The amount of similarity encoded by a link also can vary during a run of the program. Since the plausibility of slippage between two concepts depends on context (e.g., *right* ⇒ *left* is plausible in "**abc** ⇒ **abd**, **xyz** ⇒ ?" but not in "**abc** ⇒ **abd**, **ijk** ⇒ ?"), the degree of similarity encoded by a link depends on the relevance of the link's label to the problem at hand, which is measured by the activation of the node representing the label (e.g., the activation of the node *opposite* determines the degree of similarity between concepts linked in the Slipnet by an *opposite* link).

In our model, a concept is a region in the Slipnet, centered on a particular node (its core), having blurry rather than sharp boundaries: any other node is included in the concept *probabilistically*, to the degree that it resembles (or can be reached by a slippage from) the core node of the concept (Hofstadter & Mitchell, 1988a). The result is a network in which concepts are associative and dynamically overlapping (in Copycat, overlap is modeled by links), and in which the time-varying behavior of concepts (through dynamic activation and degree of similarity) reflects the essential properties of the situations encountered.

At the beginning of a run, Copycat is given the three strings of letters; it initially knows only the category membership of each letter (e.g., **a** is an instance of category **A**), which letters are spatially adjacent to one another, and which letters are leftmost and rightmost in each string. In

order to formulate a solution, the program must perceive what is going on in the problem. To accomplish this, the program builds various kinds of structures that represent its high-level perception of the problem. (This is similar to the way the Hearsay-II speech-understanding system built perceptual structures on top of raw representations of sounds; see Erman et al., 1980.) These structures represent Slipnet concepts of various degrees of generality being brought to bear on the problem, and accordingly, each of these structures is built of parts copied from the Slipnet. The flexibility of the program rests on the fact that concepts from the Slipnet can be "borrowed" for use in perceiving situations, and that the Slipnet itself is not rigid but fluid, adjusting itself (via dynamic activation and degrees of similarity) to fit the situation at hand. An essential part of our model is this interaction of top-down and bottom-up processing: while the program's perception of a given problem is guided by the properties of concepts in the Slipnet, those properties themselves are influenced by what the program perceives.

The types of perceptual structures built by the program include descriptions of objects (e.g., the **Z** in **xyz** is the "alphabetic-last" letter), relations between objects (e.g., the **Z** in **xyz** is the successor of its left neighbor, the **Y**), groups of objects (e.g., **abc** is a group increasing in the alphabet), and correspondences between objects (e.g., the **A** in **abc** corresponds to the **Z** in **xyz**). (See section 4 for examples of these structures in a run of the program.) The actual building (and sometimes destroying) of perceptual structures is carried out by large numbers of simple agents we call "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 estimate how important it is to describe the **A** in **abc** as "alphabetic-first", another codelet might notice that the **B** in **abc** is the alphabetic successor of its left neighbor in the string, and another codelet might build a data structure corresponding to that fact). Each perceptual structure is built by a series of codelets running in turn, each deciding on the basis of some local evaluation of the structure being built whether to continue by allowing the next codelet in the series to proceed, or to give up the effort at that point. If the decision is made to continue, an "urgency" value is assigned to the next codelet in the series. This value helps determine how long the codelet has to wait before it can run and continue the building-up of that particular structure.

All codelets waiting to run are placed in a single pool, and the system interleaves the building of many different structures by probabilistically choosing the next codelet to run. The choice is based on the relative urgencies of all codelets in the pool. Thus many different structures are built up simultaneously, but at different speeds. The speed of such a process emerges dynamically from the urgencies of its component codelets. Since those urgencies are determined by moment-to-moment estimates of the promise of the structure being built, the result is that structures of greater promise will tend to be built more quickly than less promising ones. There is no top-level executive directing processing here; all processing is carried out by codelets. Codelets that take part in the process of building a structure send activation to the areas in the Slipnet that represent the concepts associated with that structure. These activations in turn affect the makeup of the codelet population (for details, see Mitchell, 1988). (Note that though Copycat runs on a serial computer and thus only one codelet runs at a time, the system is roughly equivalent to one in which many activities are taking place in parallel at different spatial locations, since codelets work locally and to a large degree independently. Copycat's distributed asynchronous parallelism was inspired by the similar sort of self-organizing activity that takes place in a biological cell; see Hofstadter, 1984.) In summary, processes that build up structures are interleaved, and many such processes — some mutually supporting, some competing — progress in parallel at different rates, the rate of each being set by the urgencies of its component codelets. Almost all codelets make one or more probabilistic decisions, and the high-level behavior of the system emerges from the combination of thousands of these very small choices. The result is a *parallel terraced scan* (Hofstadter, 1983): many possible courses of action are explored simultaneously, each at a speed and to a depth proportional to moment-to-moment estimates of its promise. (Note that since the program uses

nondeterminism to arrive at a solution, different answers are possible on different runs.)

3. THE ROLE OF TEMPERATURE

In addition to the Slipnet and codelets, an essential element of Copycat's architecture is a *temperature* variable, which plays two roles. It measures the amount of disorganization (or entropy) in the system: its value at a given time is a function of the amount and quality of structure that has been built so far. Thus temperature starts high, and falls as more structure gets built, rising again if structure gets destroyed. Temperature's other role is to control the degree of randomness used in making decisions (such as which codelet should run next, which structure should win a competition, etc.). The idea is that when there is little perceptual organization (and thus high temperature), the information on which decisions are based (such as the urgency of a codelet or the strength of a particular structure) is not very reliable, and decisions should be more random than would seem to be indicated by this information. When a large amount of good structure has been built (and thus temperature is low), the information is considered to be more reliable, and decisions based on this information should be more deterministic.

The solution to the well-known "two-armed bandit" problem (Given a slot machine with two arms, each with an unknown payoff rate, what is the optimal strategy for profit-making?) is an elegant mathematical verification of these intuitions (Holland, 1975). The solution states that the optimal strategy is to sample both arms but with probabilities that diverge increasingly fast as time progresses. In particular, as more and more information is gained through sampling, the optimal strategy is to exponentially increase the probability of sampling the "better" arm relative to the probability of sampling the "worse" arm (note that one never knows with certainty which is the better arm, since all information gained is merely statistical evidence). Copycat's parallel terraced scan can be likened to such a strategy extrapolated to a many-armed bandit, where each potential path of exploration corresponds to an arm. (This is similar to the search through schemata in a genetic algorithm; see Holland, 1975). There are far too many possible paths to do an exhaustive search, so in order to guarantee that in principle every path has a non-zero chance of being explored, paths have to be chosen and explored probabilistically. Each step in exploring a path is like sampling an arm, in that information is obtained that can be used to decide the rate at which that path should be sampled in the near future.[1] The role of temperature is to cause the exponential increase in speed at which promising paths are explored as contrasted with unpromising ones; as temperature decreases, the degree of randomness with which decisions are made decreases exponentially, so the speed at which good paths crowd out bad ones grows exponentially as more information is obtained. This strategy, in which information is used as it is obtained in order to bias randomness and thus to speed up convergence toward some resolution, but to never *absolutely* rule out any path, is an optimal strategy in any situation in which there is a limited amount of time in which to explore an intractable number of paths. This appears to be an ubiquitous principle in adaptive systems of all kinds (Holland, 1975), which supports our belief that the temperature-controlled parallel terraced scan is a plausible description of how perception takes place in humans.

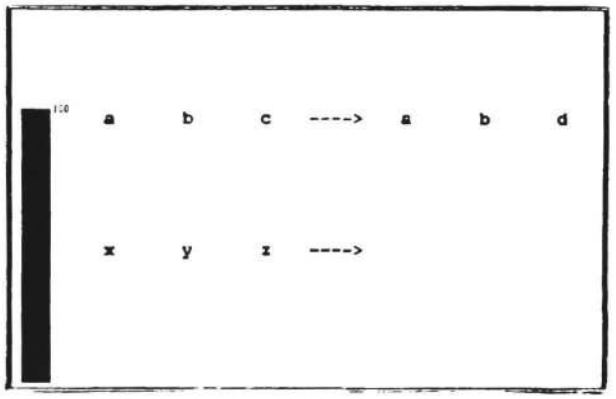
---

[1] It should be made clear that in Copycat, "paths of exploration" are defined as any of the possible ways in which the program could structure its perceptions of the situation in order to construct an analogy. Thus possible paths are not laid out in advance for the program to search, but rather are constructed by the program as its processing proceeds, just as in a game of chess, where paths through the tree of possible moves are constructed as the game is played. The evaluation of a given move in a game of chess blurs together the evaluation of many possible look-ahead paths that include that move. Similarly, any given action in building a structure by a codelet in Copycat is a step included in a large number of possible paths toward a solution, and an evaluation obtained by a codelet of a proposed structure blurs together the estimated promise of all these paths.

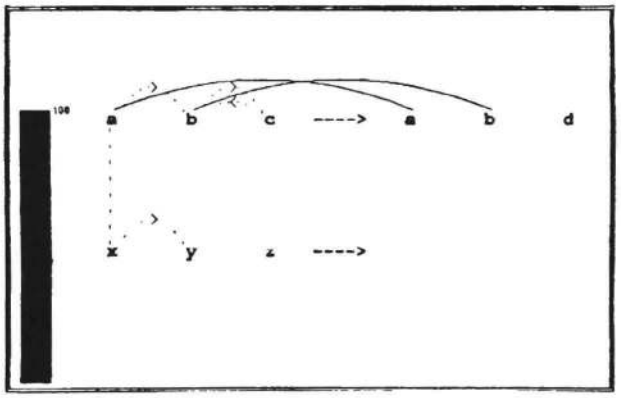
Temperature allows Copycat to close in on a good solution quickly, once parts of it have been discovered. In addition, since high temperature means more randomness, raising the temperature gives Copycat a way to get out of ruts or to deal with snags; it can allow old structures to break and restructuring to occur so that a better solution can be found. This idea is similar to the use of temperature in simulated annealing, a technique used in some connectionist networks for finding optimal solutions (Kirkpatrick et al., 1983; Hinton & Sejnowski, 1986; Smolensky, 1986). Note, however, that the role of temperature in Copycat differs from that in simulated annealing; in the latter, temperature is used exclusively as a top-down randomness-controlling factor, its value being set by a rigid "annealing schedule", not by the state of the network, whereas in Copycat, the value of temperature reflects the current quality of the system's understanding, and is used as a feedback mechanism to determine the degree of randomness used by the system. Ideas about such a role for temperature were originally presented in Hofstadter (1983, 1984).

4. A RUN OF THE PROGRAM

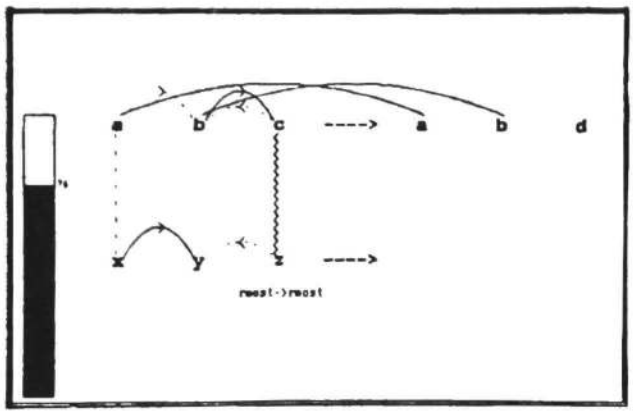
The following set of screen dumps shows the role of temperature in a run of Copycat on the problem "**abc** ⇒ **abd**, **xyz** ⇒ ?", initially helping the system to quickly arrive at a seemingly good solution that unfortunately has a snag, and then helping it to get out of that "local minimum" to create a deeper understanding of the situation and allow a more insightful answer (**wyz**) to emerge from that understanding. Note that since the program is nondeterministic, different answers are possible on different runs. At present the program produces this answer rarely; it more commonly produces **xyd** (using the rule "Replace the rightmost letter by **D**"), **xyz** ("Replace all **C**'s by **D**'s"), and **yyz** ("Replace the *leftmost* letter by its successor"). These answers, along with several other possibilities, are discussed in Hofstadter (1985).



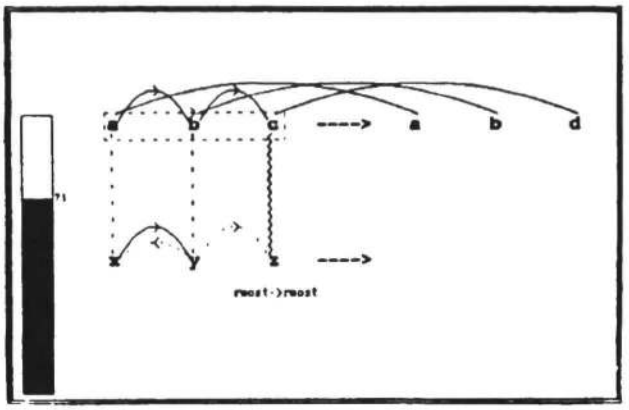
1. The program is presented with the three strings. The temperature, initially at its maximum of 100, is represented by a "thermometer" at the left.



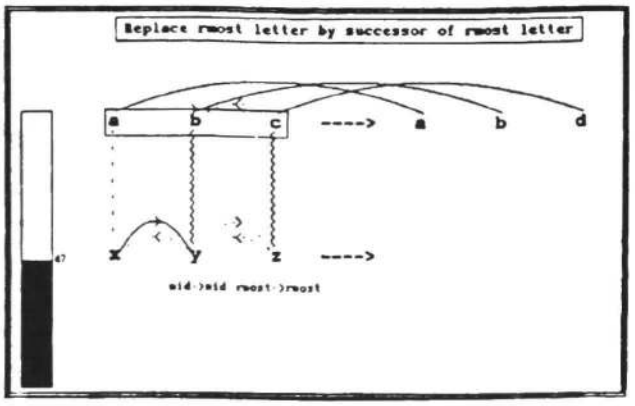
2. Codelets begin to build up perceptual structures. Dashed lines and arcs represent structures in the process of being built, and solid lines and arcs represent fully built structures. Once fully built, a structure is able to influence the building of other structures and the temperature. A fully built structure is not necessarily permanent; it may be knocked down by competing structures. Here the two solid arcs across the top line represent correspondences from the **A** and **B** in **abc** to their counterparts in **abd**. The shorter dashed arcs inside each string represent potential *successor* and *predecessor* relations in the process of being built, and the vertical dashed line represents a potential correspondence between the **A** and the **X**.



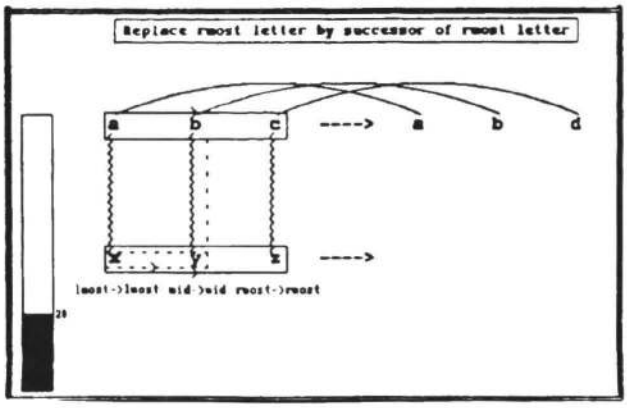
3. Some relations between letters within each string have been built and others continue to be considered. Copycat, unlike people, has no left-to-right or alphabetic-first-to-last biases, and in general is equally likely to perceive relations in either direction, although here, *successor* tends to be activated early when the C-to-D change is noticed, causing the system to tend to perceive the letters as having left-to-right successor relations rather than right-to-left predecessor relations. A correspondence between the C in abc and the Z in xyz (jagged vertical line) has been built. Both letters are *rightmost* in their respective strings: this underlying concept mapping is displayed beneath the correspondence. In response to these structures, the temperature has dropped to 76.



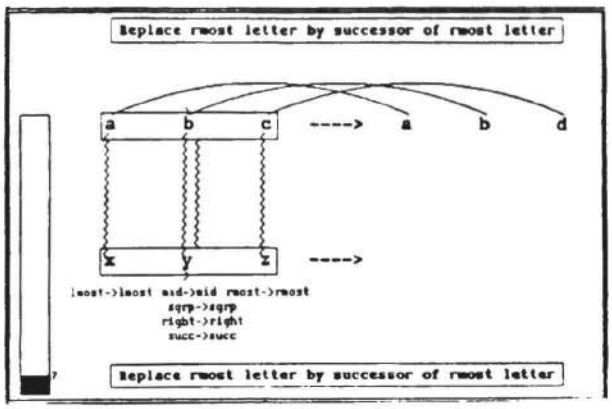
4. More relations have been built. Note that the potential predecessor relation between the Z and the Y shown in the previous screen has fizzled, and a potential successor relation has taken its place. This demonstrates the top-down pressure on the system to perceive the situation in terms of concepts it has already identified as relevant: since successor relations have been built elsewhere, the node *successor* in the Slipnet has become active, causing the system to more easily notice new successor relations. The program is also considering a left-to-right grouping of the letters in abc (represented by a dashed rectangle with a right arrow at the top), and other correspondences between the letters in abc and in xyz. The temperature has dropped to 71.



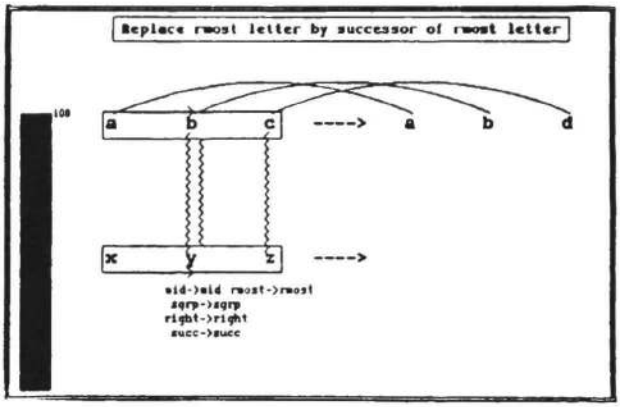
5. abc has been identified as a *successor-group*, increasing alphabetically to the right (the relations between the letters still exist, but are not displayed). A B-Y correspondence has been built, and a rule (top of screen) has been constructed to describe the abc–abd change. Note there is no internal structuring of abd. Copycat currently expects the change from the *initial string* (here abc) to the *modified string* (here abd) to consist of exactly one letter being replaced. Thus no structures are built in the modified string except to identify what has changed and what has stayed the same. The program constructs the rule by filling in the template "Replace ____ by ____". As was mentioned at the beginning of section 4, there are several possible rules for describing this change. Note that a right-to-left predecessor relation between the B and the C in abc is being considered (dashed arc), and will have to compete against the already built left-to-right successor group. The latter, being much stronger than the former, will survive, especially since the temperature is now fairly low, reflecting that a high-quality mutually consistent set of structures is taking over.



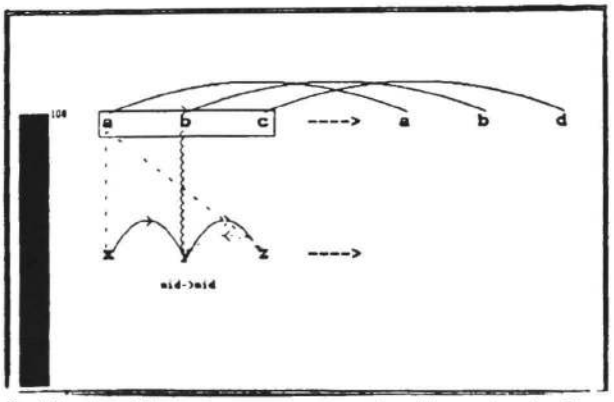
6. xyz has now been described, like abc, as a left-to-right successor group. (The direction of a group is indicated by an arrow at the top or bottom of the rectangle representing the group.) A strong set of correspondences has been made between the letters in abc and xyz, and a correspondence between the two groups (dashed vertical line) is being considered. The temperature has fallen very low, reflecting the high degree of perceptual organization, and virtually ensuring that this point of view will win out.



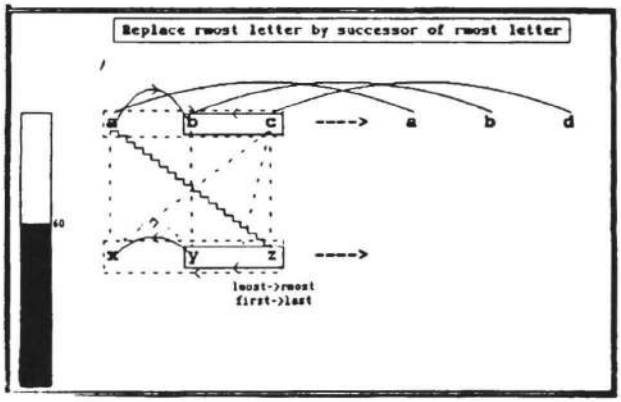
7. All the correspondences have been made. The correspondence between the two groups is supported by concept mappings expressing the facts that both are successor groups (displayed as "sgrp -> sgrp") based on successor relations ("succ -> succ") and both are increasing alphabetically toward the right. The concept mappings listed below the correspondences can be interpreted as instructions on how to translate the rule describing the initial change so it can be used on the target string. Here the concept mappings are identities, so the translated rule (appearing at the bottom of the screen) is the same as the original rule: "Replace rightmost letter by its successor". The temperature is almost at zero, indicating the program's satisfaction in its understanding of the situation. But then it hits a snag: it is unable to construct an answer according to the translated rule, since **Z** has no successor.



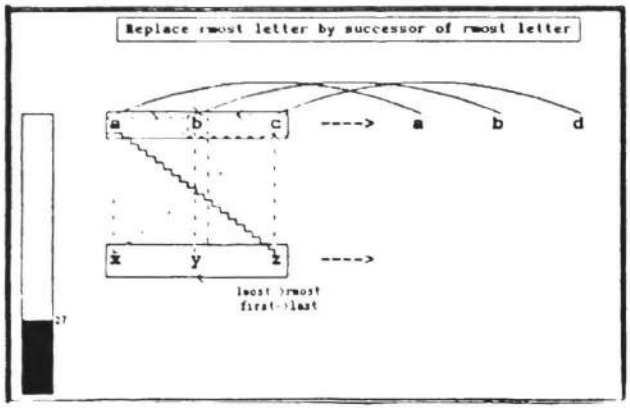
8. Being unable to take the successor of **Z**, the program has hit an impasse, which causes the temperature to go up to 100. This causes competitions between structures to be decided more randomly, and allows structures to be destroyed more easily (as can be seen, the **A** **X** correspondence has been broken). In addition, since the **Z** was identified as the cause of the impasse, the node **Z** in the Slipnet becomes highly activated, which spreads activation to *alphabetic-last*, making this concept relevant to the problem. In turn, *alphabetic-last* spreads activation to *alphabetic-first*.



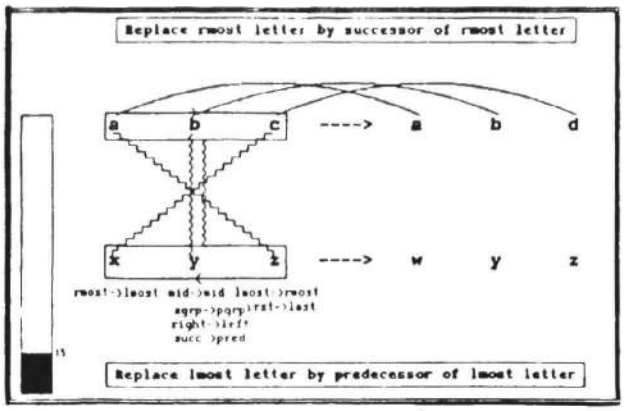
9. After breaking more structures and making other ineffectual attempts at restructuring (not shown), the program has noticed the relationship between the letters **A** and **Z**, and is trying to build a correspondence between them. Underlying it are two *slippages*: "leftmost -> rightmost" and "first -> last". Before the impasse was reached, the descriptions *first* and *last* were neither seen as relevant nor considered conceptually close enough to be the basis for a correspondence. But the combination of high temperature and the focus on the **Z** make this mapping possible, though still not easy, to make. In fact, on most runs of program on this problem, this mapping is either never made, or quickly destroyed once made. But in this run, this correspondence, once made, is perceived to be strong.



10. Many possible ways of restructuring the situation are being considered simultaneously, but the program is beginning to develop an understanding of the situation based on the **A**-**Z** correspondence. Under pressure from this correspondence, the program is now beginning to perceive **xyz** as a right-to-left *predecessor* group (yz has already been perceived as such, and the direction of the relation between the **X** and the **Y** has reversed). This new way of structuring the problem seems promising; the new structures have caused the temperature to fall to 60.



11. The "first -> last" slippage has engendered a complete restructuring of the program's perception of **xyz** (which is now understood as a right-to-left predecessor group, opposite in direction from the group **abc**) and the program is closing in on a solution. Alternative ways of structuring the situation are still being considered, but the low temperature reflects the program's satisfaction with its current understanding, and will make it hard for any alternatives to compete at this point.



12. The mapping is complete and all attempts at building rival structures have ceased. The concept mappings listed underneath the correspondences give the slippages needed to translate the rule. The translated rule ("Replace *leftmost* letter by *predecessor* of *leftmost* letter") appears at the bottom of the screen, and the answer **wyz** appears at the right.

ACKNOWLEDGEMENTS

We thank David Chalmers, Robert French, Liane Gabora, Kevin Kinnell, David Moser, and Peter Suber for their ongoing contributions to this project and for many helpful comments on this paper. 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] Erman, L.D., F. Hayes-Roth, V. R. Lesser, and D. Raj Reddy (1980). The Hearsay-II speech-understanding system: Integrating knowledge to resolve uncertainty. *Computing Surveys*, **12** (2), 213-253.

[2] Kirkpatrick, S., C.D. Gelatt Jr., and M. P. Vecchi (1983). Optimization by simulated annealing. *Science*, **220** (4598), 671-680.

[3] Hinton, G.E. and T.J. Sejnowski (1986). Learning and relearning in Boltzmann machines. In McClelland, J. and D. Rumelhart (1986) (Eds.). *Parallel distributed processing* (pp. 282-317). Cambridge, MA: Bradford/MIT Press.

[4] Hofstadter, Douglas R. (1983). The architecture of Jumbo. *Proceedings of the International Machine Learning Workshop*. Monticello, Il.

[5] Hofstadter, Douglas R. (1984). The Copycat project: An experiment in nondeterminism and creative analogies (AI Memo #755). Cambridge, MA: MIT AI Laboratory.

[6] Hofstadter, Douglas R. (1985). Analogies and roles in human and machine thinking. In *Metamagical Themas* (pp. 547-603). New York: Basic Books.

[7] Hofstadter, Douglas R. and Melanie Mitchell (1988a). Concepts, analogies, and creativity. In *Proceedings of the Canadian Society for Computational Studies of Intelligence*. Edmonton, Alberta: Univ. of Alberta.

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

[9] Holland, John (1975). *Adaptation in natural and artificial systems*. Ann Arbor, MI: Univ. of Michigan Press.

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

[11] Smolensky, P. (1986). Information processing in dynamical systems: Foundations of harmony theory. In McClelland, J. and D. Rumelhart (1986) (Eds.). *Parallel distributed processing* (pp. 194-281). Cambridge, MA: Bradford/MIT Press.