Physica D 42 (1990) 322–334
North-Holland

THE EMERGENCE OF UNDERSTANDING IN A COMPUTER MODEL OF CONCEPTS AND ANALOGY-MAKING

Melanie MITCHELL and Douglas R. HOFSTADTER
Center for Research on Concepts and Cognition, Indiana University, 510 North Fess, Bloomington, IN 47408, USA

This paper describes Copycat, a computer model of the mental mechanisms underlying the fluidity and adaptability of the human conceptual system in the context of analogy-making. Copycat creates analogies between idealized situations in a microworld that has been designed to capture and isolate many of the central issues of analogy-making. In Copycat, an understanding of the essence of a situation and the recognition of deep similarity between two superficially different situations emerge from the interaction of a large number of perceptual agents with an associative, overlapping, and context-sensitive network of concepts. Central features of the model are: a high degree of parallelism; competition and cooperation among a large number of small, locally acting agents that together create a global understanding of the situation at hand; and a computational temperature that measures the amount of perceptual organization as processing proceeds and that in turn controls the degree of randomness with which decisions are made in the system.

1. Introduction

How, when one is faced with a new situation, does understanding emerge in the mind? How are we guided by a multitude of initially unconnected and novel perceptions to a coherent and familiar mental representation, such as "a coffee cup", "the letter 'A"', "French Baroque style", or "another Vietnam"? And how are such representations structured so that they are flexible, fluid, and thus adaptable to many different situations, rather than brittle, rigid, and inextensible? In our research, we are investigating these questions by building a computer model of what we think are the mental mechanisms underlying the fluid and adaptable nature of human concepts. As can be seen from the above examples, the mental phenomena we are studying range over the spectrum of recognition, categorization, and analogy-making: all are instances of high-level, abstract, or "deep", perception [1] as opposed to low-level modality-specific perception. The essential philosophy behind our model is that high-level perception emerges from a system of many independent processes running in parallel. These compete with and support each other by creating and destroying temporary perceptual constructs and by changing the activation levels and degrees of overlap in an associative network of permanent concepts with blurry conceptual boundaries. Such a system has no global executive deciding which processes should run next and what each process should do; rather, all processing is done by many small independent agents that make their decisions probabilistically. The system is self-organizing, with coherent and focused behavior being a statistically emergent property of the system as a whole [1]. Our computer model, called "Copycat", is an attempt to implement and test such a system in the realm of analogy-making, a realm in which the necessity of constructing fluid and adaptable mental representations is particularly apparent.

2. A microworld for analogy-making

In order to isolate and model the mechanisms underlying perception and analogy-making, we have developed a microworld in which analogies can be made between idealized situations consist-

0167-2789/90/$03.50

ing of strings of letters. A simple analogy problem in this domain is the following: If the string ***abc*** changes to ***abd***, what is the analogous change for ***ijk?*** In the microworld, knowledge about letters and strings is quite limited. The 26 letters are known, but only as abstract categories; shapes, sounds, words, and all other linguistic and graphic facts are unknown. The only relations explicitly known are predecessor and successor relations between immediate neighbors in the alphabet. Ordinal positions in the alphabet (e.g. the fact that **S** is the 19th letter) are not known. (Notational note: Boldface capitals (e.g. **S**) denote the 26 abstract ***categories*** of the alphabet, and never appear in strings; boldface smalls (e.g. ***abc***) denote ***instances*** of these categories, and appear only in strings.) The strings represent idealized situations containing objects, relations, and events; these small situations serve as ***metaphors*** for more complex, real-world situations. In the above problem, the ***initial string abc*** and the ***target string ijk*** are two situations, each with its own objects and relationships. The change to ***abd*** highlights a fragment of the first situation, and the challenge is to find the "same" fragment in ***ijk*** and to highlight and modify it in "the same way". What has been highlighted, how it has been highlighted, and what "the same way" means in the second situation are all up to the analogy-maker to decide.

For the sample problem given above, 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, other problems are not so simple. For instance, given the same initial change and an 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*** of this sort – 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 [2], 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.

Consider now the same initial change but with target ***srqp.*** Here a rigid application of the original rule would yield ***srqq,*** which again ignores a more abstract similarity between ***abc*** and ***srqp.*** One of the two answers people tend to prefer is ***trqp*** ("Replace the ***leftmost*** letter by its successor"), which is based on seeing ***abc*** as a left-to-right string and ***srqp*** as a right-to-left string (where each string increases alphabetically); here there is a slippage from the concept ***right*** to the concept ***left,*** which in turn gives rise to the slippage ***rightmost*** ***leftmost.*** The other answer given by many people is ***srqo*** ("Replace the rightmost letter by its ***predecessor"),*** for which ***abc*** is seen as increasing and ***srqp*** as decreasing (both viewed as moving rightwards), yielding a slippage from ***successor*** to ***predecessor.***

The letter-string microworld was designed to capture the essence of the issues of concepts and high-level 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. Many people answer ***xya***, but in our microworld the alphabet is not circular; this answer is intentionally excluded in order to force analogy-makers 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 *xyz* is "wedged" against the far end of the alphabet, and *abc* is similarly wedged against the beginning of the alphabet. Thus the *Z* in *xyz* and the *A* in *abc* can be seen to correspond, and then one naturally feels that the *X* and the *C* correspond as well. Underlying these object correspondences is a set of concept slippages that are conceptually parallel: ***alphabetic-first ⇒ alphabetic-last; right ⇒ left,*** and ***successor ⇒ predecessor.*** Taken together, these concept slippages convert the original rule into a rule adapted to the target string *xyz*: "Replace the *leftmost* letter by its ***predecessor",*** which yields an insightful answer: ***wyz.***

These examples illustrate how solving problems in the microworld can require many of the mental abilities needed for analogy-making and high-level perception in general. In the process of understanding a situation, the mind must permit competition among the 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. As mental processing proceeds, exploration of each of these various possibilities should be given the cognitive resources it seems to deserve as determined moment to moment, and the partial and uncertain information obtained as this exploration proceeds must be used to narrow the range of possibilities. One's already-existing concepts should direct perception in a top-down manner, and whatever is perceived should affect one's concepts in return, by activating concepts that seem relevant and by changing the perceived similarity between concepts in order to reflect the current situation. One must decide what is salient and what can be ignored, how abstractly to describe objects, relations, and events, which descriptions to take literally and which to allow to slip, etc. And if these choices lead to an impasse that seems to block progress towards understanding, then one may be required to restructure one's original perceptions, perhaps shifting one's view in unexpected ways, in order to arrive at a deeper, more essential understanding of the situation. (For a detailed discussion of the microworld and a large number of sample analogy problems, see ref. [2].)

3. The architecture of Copycat

The Copycat program solves analogy problems in the letter-string microworld, and we believe it embodies many of the mechanisms underlying analogy-making in general. The architecture of the model consists of three parts: (a) the ***Slipnet:*** a network of nodes and links representing the permanent concepts of the system; (b) a working area in which perceptual structures representing the system's current understanding of the problem are built and modified in statistically emergent processes arising from large numbers of various special-purpose agents and running in asynchronous parallel: this area can be thought of as corresponding to the locus of creation and modification of mental representations that occur in the mind as one formulates a coherent understanding of a situation; and (c) a pool of various perceptual and higher-level structuring agents ("codelets") waiting to run. We will explain how understanding in Copycat emerges from the interaction of these three components.

3.1. The Slipnet

Copycat's ***Slipnet,*** a small part of which is shown in fig. **1,** contains the permanent concepts of the system. Here a node represents the "core" of a

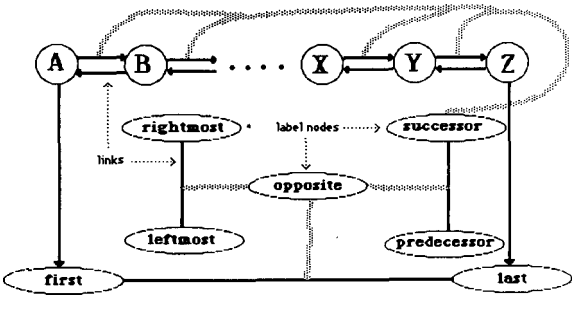


Fig. 1. A small part of Copycat's Slipnet, showing nodes, links (solid lines), and labels on **links** (thickly dotted lines).

concept (e.g. the node labeled "first" represents the core of the concept *first)* and a link simultaneously represents an ***association*** 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 these concepts are similar and one can be slipped to the other. Each link has a ***label*** that roughly classifies the type of association the link encodes. Each type of label is itself represented by a node. For instance, the link connecting nodes ***first*** and ***last*** has label ***opposite,*** which is itself a node; likewise, the links connecting nodes ***A*** and ***B*** have labels ***successor*** and ***predecessor.*** During a run of the program, nodes become activated when perceived to be relevant (by structure-building agents; see section 3.3), and decay when no longer perceived as relevant. Nodes also spread activation to their neighbors, and thus concepts closely associated with relevant concepts also become relevant. The amount of similarity encoded by a link can vary during a run. Since the plausibility of slippage between two concepts depends on context (e.g. ***right ⇒ left*** is plausible in ***"abc ⇒ abd, srqp ⇒*** ?", 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 given problem, which is determined by the level of activation of the node representing the label. For example, the level of activation of the node ***opposite*** affects the degree of similarity between nodes linked by an ***opposite*** link.

Decisions about whether or not a slippage can be made from a given node – say, ***successor*** – to a neighboring node – say, ***predecessor*** – are made ***probabilistically,*** as a function of the degree of similarity between the two nodes. (Such decisions are made by codelets, as described in section **3.3** below.) Thus, in our model, a concept like ***successorship*** is identified not with a single node but rather with a region in the Slipnet, centered on a particular node, and having blurry rather than sharp boundaries: neighboring nodes can be seen as being included in the concept probabilistically, as a function of their degree of similarity to the core node of the concept. Since the degree of similarity between two nodes is context-dependent, concepts in the Slipnet are emergent rather than explicitly defined. They are associative and dynamically overlapping (here, overlap is modeled by **links),** and their time-varying behavior (through dynamic activation and degree of similarity) reflects the essential properties of the situations encountered. Thus concepts are able to adapt (in terms of relevance and similarity to one another) to different situations. Note that we are not modeling ***learning*** in the usual sense: the program neither retains changes in the network from run to run nor creates new permanent concepts; however, our work does involve learning if that term is taken to include the generalization from experience that humans perform in novel contexts.

3.2. Perceptual structures

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, rightmost, and middle in each string. In order to formulate a solution, the program must perceptually organize each situation, as well as the various relationships among situations. To accomplish this, the program gradually builds various kinds of structures that represent its high-level perception of the problem. (This process is similar to the way the Hearsay II speech-understanding system built layers of increasingly abstract perceptual structures on top of raw representations of sounds; see ref. [3].) These structures correspond to 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 on the fact that the Slipnet itself is not rigid but fluid, adjusting itself (via dynamic activation and degrees of similarity) to fit the

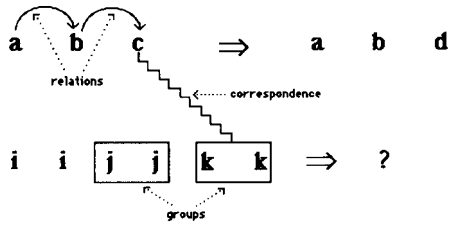


Fig. 2. Perceptual structures, including relations, groups, and a correspondence.

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.

Fig. 2 shows examples of perceptual structures that could be built in the process of solving the problem *"abc ⇒ abd, iijjkk ⇒ ?"*. The types of structures built by the program include ***descriptions*** of objects (e.g. the ***C*** in ***abc*** is the string's ***rightmost*** letter), ***relations*** between objects (e.g. the ***B*** in ***abc*** is the ***successor*** of its left neighbor, the ***A***), ***groups*** of objects (e.g., ***jj*** is a group of adjacent identical letters; the entire string ***abc*** could be seen as a group of adjacent letters that increase alphabetically), and ***correspondences*** between objects (e.g. the ***C*** in ***abc*** corresponds to the group ***kk*** in ***iijjkk***). (See section 5 for examples of these structures in a run of the program.)

3.3. Codelets, randomness, and the parallel terraced scan

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 local task that is part of the process of building a structure. For example, one codelet might estimate how important it is to describe the ***A*** in ***abc*** as "alphabetic-first" (taking into account the level of activation of that node in the Slipnet), another might notice that the ***B*** in ***abc*** is the alphabetic successor of its left neighbor in the string, and another might build the structure representing that fact. The program has several different types of codelets, each corresponding to one of the different types of structures that can be built (descriptions, relations, groups, and correspondences). Codelets are able to calculate the ***strength*** of a potential structure at a given time, based on a number of factors. For example, the strength of a potential successor relation between the ***A*** and the ***B*** in ***abc*** at a given time is a function of the degree of association encoded by the successor **link** from node ***A*** to node ***B*** in the Slipnet, the number of successor relations that have already been built nearby in the same string, and the level of activation (i.e. the currently perceived *relevance*) of the node ***successor.*** Since all these factors can change as processing proceeds, a structure's strength at a given time is only an estimate of how useful that structure will be in the long run.

For each type of structure there are two kinds of codelets: those that investigate the possibility of building that structure at a probabilistically chosen location ("musing" codelets), and those that actually build the structure ("builder" codelets). A structure is built by a series of codelets running in turn, each deciding probabilistically on the basis of its estimation of some aspect of the structure's strength whether to continue by generating one or more follow-up codelets, or to abandon the effort at that point. If the decision is made to continue, the running codelet assigns an "urgency" value (based on the strength of the structure) to each follow-up codelet. This value helps determine how long each follow-up codelet will have to wait before it can run and continue the evaluation of that particular structure. If the effort is not abandoned, the series ends with a builder codelet that tries to build the structure, possibly having to compete with already existing, incompatible structures, which may prevent it from completing its goal. The outcome of such a fight is decided probabilistically on the basis of the strengths 01 the competing structures.

Attempts at building many different structures are interleaved, as follows. All codelets waiting to run are placed in a single pool, and at each time step, the system probabilistically chooses one codelet to run. The choice is based on the relative urgencies of all codelets in the pool. When a codelet is chosen to run, it is removed from the pool. At a given time, the codelet pool generally contains many different types of codelets, but can also contain multiple copies of the same type of codelet – say, one that tries to make a grouping of objects already determined to be related. At the beginning of a run of the program, the pool contains a standard initial population of codelets, and as codelets run and are removed from the pool, new codelets are added both by codelets that run and by active nodes in the Slipnet. Thus the pool's population changes, as processing proceeds, in response to the needs of the system as judged by previously run codelets and by activation patterns in the Slipnet.

The fine-grained breakup of acts of structure building in our model has two purposes: (1) it allows many such acts to be carried out in parallel, by having their components interleaved; and (2) it allows the speed of and computational resources allocated to each such act to be dynamically regulated by moment-to-moment evaluations of the structure being constructed. It is important to understand that in this system, such processes, each of which consists of many codelets running in a series, are themselves emergent entities: rather than being predetermined and then broken up into small components, they are instead "post-determined", being the pathways visible, after the fact, leading to some given macroscopic act of construction or destruction of perceptual or organizational structure. In other words, only the codelets themselves are predetermined; the macroscopic ***processes*** of the system are emergent.

Since these processes are interleaved, attempts at building many different structures proceed 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, since active nodes (e.g. ***successor)*** are able to add codelets to the pool (e.g. a codelet that tries to find a successor relation between some pair of objects). 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. In a cell, all activity is carried out by large numbers of widely distributed enzymes of various sorts. These enzymes depend on random motion in the cell's cytoplasm in order to encounter substrates (molecules such as amino acids) from which to build up structures. Complex structures are built up through long chains of enzymatic actions, and separate chains proceed independently and asynchronously in different spatial locations throughout the cytoplasm. Moreover, the enzyme population in the cell is itself regulated by the products of the enzymatic activity, and is thus sensitive to the moment-to-moment needs of the cell. In Copycat, codelets roughly act the part of enzymes. All activity is carried out by large numbers of codelets, which choose objects in a probabilistic, biased way for use in building structures. As in a cell, the processes by which complex structures are built are not explicitly programmed, but are emergent outcomes of chains of codelets working in asynchronous parallel throughout Copycat's structure-building area (its "cytoplasm"). And just as

in a cell, the population of codelets in the codelet pool is self-regulating and sensitive to the moment-to-moment needs of the system. And to carry this analogy further, the Slipnet could be said to play the role of DNA, with active nodes in the Slipnet corresponding to genes currently being expressed in the cell, controlling the production of enzymes. The purpose of this metaphor was to draw inspiration from the mechanisms of self-organization in a fairly well-understood natural system, and to use these ideas in thinking about the mechanisms of abstract perception. The mechanisms of enzymes and DNA in a cell are not to be taken literally as a model of perception; rather, general principles can be abstracted and carried over from the worlungs of cells to the worlungs of perception. Distributed asynchronous parallelism, emergent processes, the building-up of coherent complex structures from initially unconnected parts, self-organization, self-regulation, and sensitivity to the ongoing needs of the system are all central to our model of perception, and thinking about the workings of the cell has helped us to devise mechanisms underlying these principles in Copycat. (For further discussion of the cell metaphor, see ref. [1].)

In summary, in Copycat, 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 (for example, whether to continue building a given structure, whether to destroy a competing structure, etc.) and the high-level behavior of the system emerges from the combination of hundreds of these very small choices. The result is a ***parallel terraced scan*** [1], 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. (Note that since the program arrives nondeterministically at a solution, different answers are possible on different runs, and even runs leading to the same answer follow very different pathways on a microscopic level.) The above discussion of Copycat will be made clearer by the sample run of the program given in section 5.

4. Computational temperature

A fourth element of Copycat's architecture is a ***temperature*** variable, which ***measures the amount of disorganization*** ("entropy") in the system's understanding of the problem: the value of the temperature at a given time is a function of the amount and quality of perceptual or organizing structure that has been built so far. In particular, the temperature at a given time is an inverse function of the sum of the current strengths of all existing structures. Thus temperature starts high, falls as structures are built, and rises again if structures are destroyed or if their strengths decrease. In turn, the value of temperature ***controls the degree of randomness*** used both by codelets in making decisions (such as whether to add follow-up codelets to the pool, whether to break a competing structure, whether to allow a particular slippage, etc.) and by the system in choosing which codelet to run next. 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 structure deemed to be good 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 strategy of dividing one's play between the two arms is optimal for profit-making?) is an elegant mathematical verification of these intuitions [4]. The solution states that the optimal strategy is to sample both arms but with probabilities that di-

verge 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 – in fact, a "bandit" with a dynamically changing number of arms, where each arm represents a potential path of exploration. (This is similar to the search through schemata in a genetic algorithm; see ref. [4].) 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*'. The role of temperature is to cause the exponential increase in the 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 probabilistic choices and thus to speed up convergence toward some resolution, but to never ***absolutely*** rule out any path of exploration, is optimal in any situation in which there is a limited amount of time to explore an intractable number of paths. Such a strategy appears to be a central principle in many kinds of adaptive systems [4], which supports our belief that the temperature-controlled parallel terraced scan is a plausible description of how perception takes place in humans.

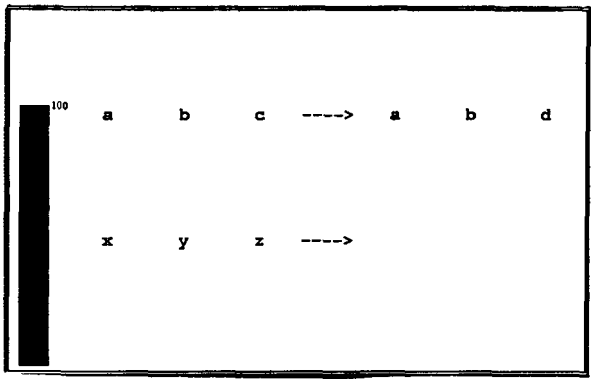
Temperature allows Copycat to close in on a good solution quickly, once parts of it have been discovered. In addition, since high temperature means more randomness, temporarily raising the temperature gives Copycat a way to get out of ruts or to deal with snags; it can allow old structures to break and restructuring to occur so that a better solution can be found. That is, when the system faces an impasse, the temperature can go up in spite of the fact that seemingly good organizing structures exist. Such a use of temperature is illustrated in the run of the program given in section 5. Note that the role of temperature in Copycat differs from that in simulated annealing, a technique used in some connectionist networks for finding optimal solutions [5–7], in which temperature is used exclusively as a top-down randomness-controlling factor, its value falling monotonically according to a predetermined, rigid "annealing schedule". By contrast, in Copycat, the value of temperature reflects the current quality of the system's understanding, so that temperature acts as a feedback mechanism that determines the degree of randomness used by the system. Ideas about such a role for temperature were first presented by Hofstadter [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 problem 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 evaluations 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.

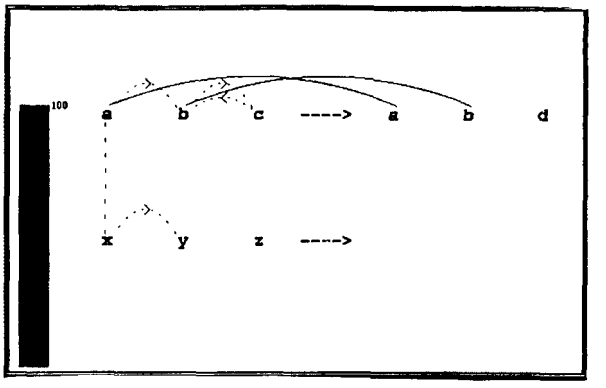
5. A run of the program

The following screen dumps are from a run of Copycat on *"abc ⇒ abd, xyz ⇒ ?"*. This run produced the desired answer *wyz*. Since the program is nondeterministic, different answers are possible on different runs. The program currently produces

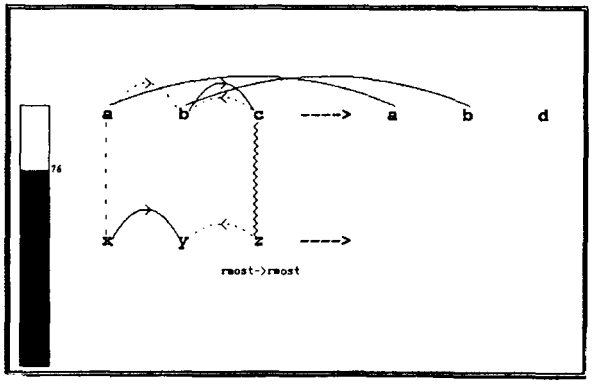
this answer infrequently; it more commonly produces ***xyd*** (as a consequence of generating the rule "Replace the rightmost letter by ***D"), xyz*** (for which the rule is "Replace all ***C's*** by ***D's"),*** and ***yyz*** (in which the original rule is translated for the target string as "Replace the *leftmost* letter by its successor"). The relative merits of these and several other possible answers are discussed by Hofstadter [2]. The current version of Copycat can solve all the problems discussed in this paper; that is, for each problem, the program is able on some runs to find the answer or answers that most people agree are the best. Plans for extending the program are given in ref. [8].



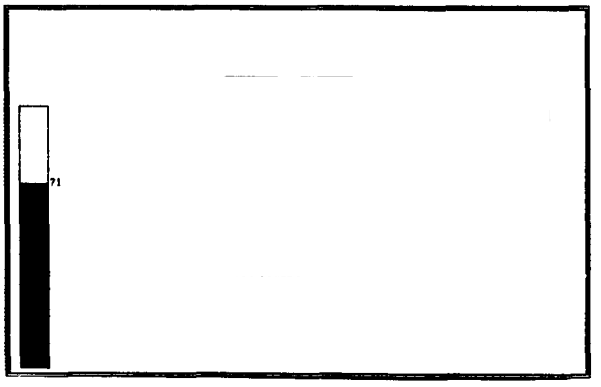
1. The program is presented with the three strings. The temperature, initially at its maximum of 100, is represented by a "thermometer" at the left. As structures are built, their representations will be drawn on the screen. Dashed lines and arcs represent structures in the process of being built, and 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.



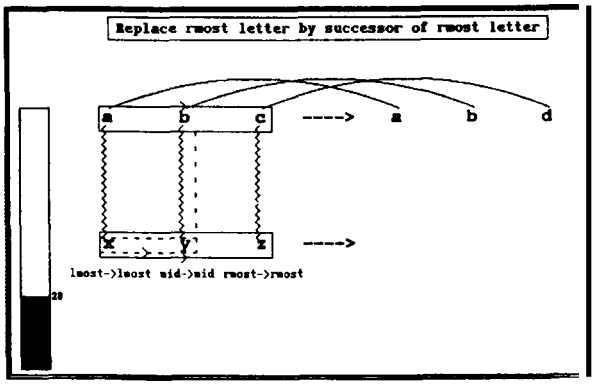
2. Codelets begin to build up structures. The two solid arcs across the top line represent correspondences from the ***A*** and ***B*** in ***abc*** to their counterparts in ***abd***. The dashed arcs inside each string represent potential ***successor*** and ***predecessor*** relations in the process of being built, and the vertical dashed line represents a potential correspondence between the ***A*** and the ***X***.



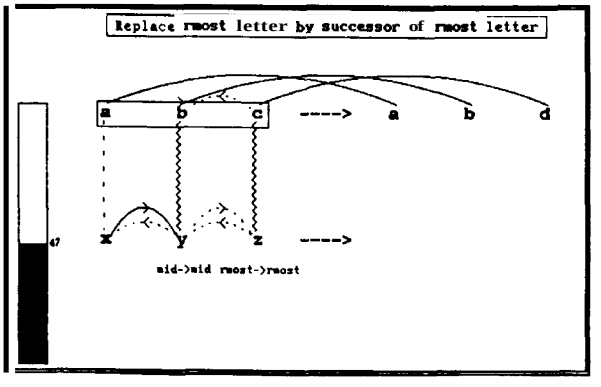
3. Some relations between letters within each string have been built and others continue to be considered. Copycat, unlike people, has no left-to-right or alphabetic-forwards biases, and in general is equally likely to perceive relations in either direction, although here, ***successor*** tends to be 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*** and the ***Z*** (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 the growing amount of structure, the temperature has dropped to 76.



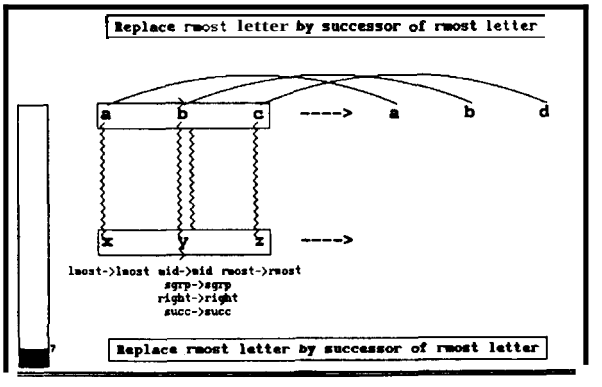
4. More relations have been built. Note that the potential predecessor relation between the Z and the Y shown in the previous screen has fizzled, and a potential successor relation has taken its place. This demonstrates the top-down pressure on the system to perceive the situation in terms of concepts 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 ***abc*** and ***xyz***. The temperature has dropped to 71.



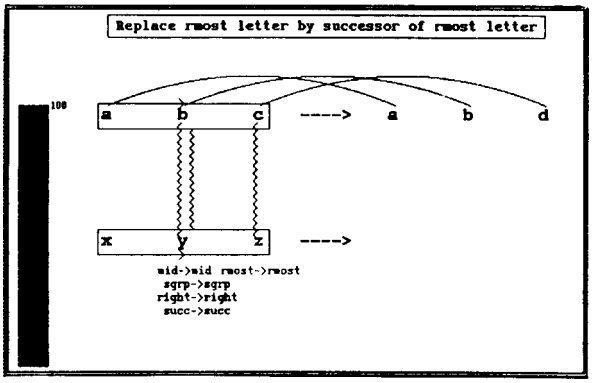
6. *xyz* has now been described, like ***abc,*** as a left-to-right successor group. (The direction of a group is indicated by an arrow at the top or bottom of the rectangle representing the group.) A rival grouping of just the ***X*** and Y is also being considered (dashed rectangle), but will have little chance of defeating the stronger already-existing group. A strong set of correspondences has been made between the letters in ***abc*** and *xyz*, and a correspondence between the two groups *as* wholes (dashed vertical line) is being considered. The temperature has fallen to the low value of 28, reflecting the high degree of perceptual organization, and virtually ensuring that this highly coherent way of seeing the situations will win out.



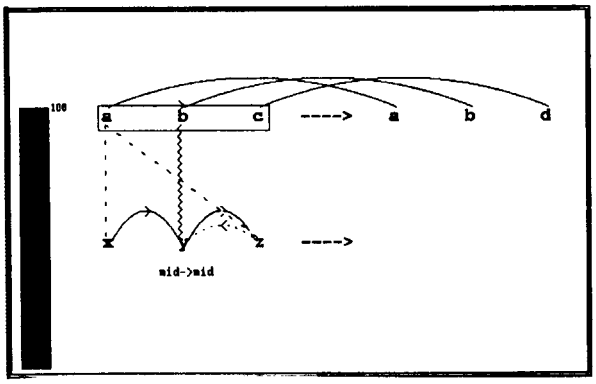
5. abc has been identified as a ***successor*** group, increasing alphabetically to the right (the relations between the letters still exist, but are not displayed). A ***B-Y*** correspondence has been built, and a rule (top of screen) has been constructed to describe the *abc-abd* change. Note there 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 indicated at the beginning of section 5, 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.



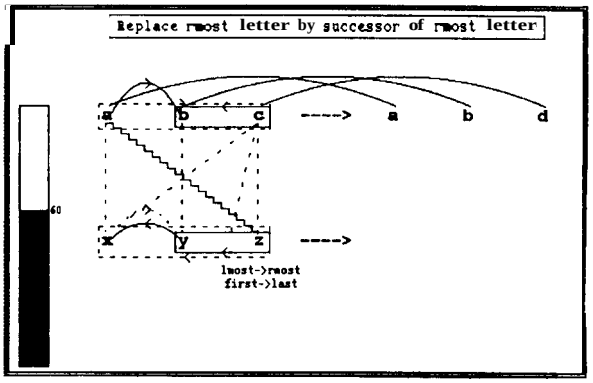
7. All the correspondences have been made. The group-level correspondence is supported by concept mappings expressing the facts that both are successor groups (displayed as "sgrp ⇒ sgrp") based on successor relations ("succ ⇒ succ") and both are increasing alphabetically toward the right. The concept 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 all 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 as an answer-building codelet attempts to carry out the translated rule, it hits a snag: Z has no successor.



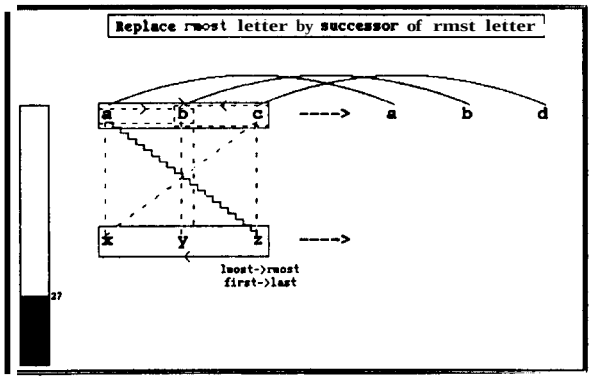
8. Being unable to take the successor of Z, the program hits an impasse, which causes the temperature to go up to 100. This causes competitions between structures to be decided much more randomly, and allows structures to be destroyed more easily (as can be seen, the *A – X* correspondence has been broken).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



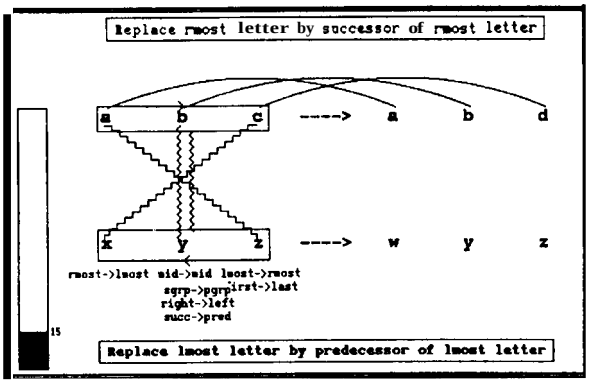
9. After breaking more structures and making other ineffectual attempts at restructuring (not shown), the program has noticed the relationship between the letters *A* and Z, and is trying to build a correspondence between them. Underlying it are two *slippages:* "leftmost ⇒ rightmost" and "first ⇒ last". 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 makes 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, survives.



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). The new structures have caused the temperature to fall to 60, indicating that this new way of structuring the problem seems promising to the program.



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 complete 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.

6. The architecture of Copycat as a style of emergent computation

Forrest [9] has outlined a paradigm for the phenomenon of emergent computation. According to this paradigm, emergent computation occurs when explicit local instructions interact to form implicit, meaningful global patterns. Information processing is done at the level of the implicit global patterns rather than at the level of the explicit instructions. In addition, the global patterns have the possibility of looping back down and influencing the behavior of the local instructions. This account has much in common with Hofstadter's [10] discussion of emergent ***symbols*** in the brain, in which the top level (the ***symbolic*** level) reaches back down towards the lower levels (the ***subsymbolic*** levels) and influences them, while at the same time being itself determined by the lower levels. Since the Copycat project itself is an attempt to model the mechanisms underlying emergent symbols in the brain (i.e., concepts), it is not surprising that the features of emergent computation posited by Forrest can be identified in our model. In Copycat, individual nodes and links in the Slipnet and individual codelets can be thought of as constituting explicit local instructions. The interaction among codelets, nodes, and links gives rise to two types of implicit global patterns: **(1)** ***concepts,*** which are identified not with any one node but rather with a core node surrounded by a probabilistic "halo" of neighboring nodes, where inclusion of a node in a concept is a probabilistic function of the context-dependent conceptual distance to the core node; and ***(2)*** the ***parallel terraced scan,*** in which many possible ways of construing the problem are considered simultaneously, each at a rate and to a depth proportional to moment-to-moment estimates of its promise. These implicit global patterns in turn reach back and influence the lower levels (and thus each other): the emergent structure of concepts influences how codelets will evaluate possible perceptual structures (since the evaluation of a structure takes into account activations and conceptual distances in the Slipnet) and affects the population of codelets and their urgencies; and the emergent parallel terraced scan, by determining how the search through possible structurings proceeds, affects the activation of nodes and thus the conceptual distances encoded by links in the Slipnet. It is through this complex interaction of explicit and emergent levels that we are attempting to capture the fluidity and adaptability of concepts and perception.

Acknowledgements

We thank David Chalmers, Robert French, Liane Gabora, Jim Levenick, Gary McGraw, David Moser, and Peter Suber for ongoing contributions to this project and for many helpful comments on this paper. The paper was also improved by helpful suggestions from two anonymous reviewers. 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] D.R. Hofstadter, The architecture of Jumbo, in: Proceedings of the International Machine Learning Workshop, Monticello, Illinois **(1983).**

[2] D.R. Hofstadter, Analogies and roles in human and machine thinking, in: Metamagical Themas (Basic Books, New York, **1985)** p. **547.**

[3] L.D. Erman, F. Hayes-Roth, V.R. Lesser and D. Raj Reddy, The Hearsay-I1 speed-understanding system: Integrating knowledge to resolve uncertainty, Computing Surveys **12 (1980) 213.**

[4] J.H. Holland, Adaptation in Natural and Artificial Systems (University of Michigan Press, Ann Arbor, MI, **1975).**

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

[6] G.E. Hinton and T.J. Sejnowski, Learning and relearning in Boltzmann machines, in: Parallel Distributed Processing, eds. J. McClelland and D. Rumelhart, (Bradford/MIT Press, Cambridge, MA, **1986)** p. **282.**

[7] P. Smolensky, Information processing in dynamical systems: Foundations of harmony theory, in: Parallel Distributed Processing, eds. J. McClelland and D. Rumelhart, (Bradford/MIT Press, Cambridge, MA, **1986)** p. **194.**

[8] D.R. Hofstadter, M. Mitchell and R.M. French, Fluid concepts and creative analogies: A theory and its computer implementation, Technical Report **10,** Cognitive Science and Machine Intelligence Laboratory, University of Michigan, Ann Arbor, MI **(1987).**

[9] S. Forrest, Physica D **42 (1990) 1–11,** these Proceedings.

[10] D.R. Hofstadter, Gödel, Escher, Bach: an Eternal Golden Braid (Basic Books, New York, **1979).**