Conceptual Slippage and Analogy-Making: A Report on the Copycat Project

Douglas R. Hofstadter and Melanie Mitchell
Department of Psychology and Department of Computer Science
University of Michigan

In our research we are investigating the mechanisms underlying human analogy-making. We are developing a theory of these mechanisms, which centers on the interaction of perception with the associative, overlapping, and context-sensitive human conceptual system, and on how this interaction gives rise to "conceptual slippage" (the flexible translation of ideas from one framework to another), which is required for creative analogy-making. To test this theory, we are building a computer model called "Copycat", which is able to make analogies in a microdomain. Although the microdomain appears small and simple, it is surprisingly rich; extremely subtle analogies requiring great flexibility and creativity can be made in it, and we believe it is an excellent testbed for computer models of analogy-making. This paper describes the current state of our research, and shows in detail (using a series of screen printouts from two runs of the program) how Copycat's perceptual mechanisms interact with its conceptual system and allow it to describe situations and make analogies. Previous work on the Copycat project has been reported by Hofstadter (1984b, 1985), Hofstadter, Mitchell, & French (1987), Hofstadter & Mitchell (1988), and Mitchell (1988).

Copycat's microworld consists of the 26 letters of the alphabet and associated concepts; in it we construct analogy problems involving letter-strings. A simple problem is: If **abc** changes to **abd**, what is the analogous change to **pqrs**? Most people answer **pqrt**, using the rule "Replace the rightmost letter by its successor". However, if the target string were **ppqqrrss** rather than **pqrs**, that rule would yield **ppqqrrst**, which almost all people see as too rigid. The rule thus has to be "translated" to the new situation. But a different translation is needed for target **ssrrqqpp**, and still other translations for targets **mrrjjj** (in which numerical successorship plays the role of alphabetic successorship), **aababc** (extending the notion of successorship if the string is parsed **a-ab-abc**), **ace** (double successorship), and **xyz** (**Z** has no successor). A vast number of interesting problems can be constructed in this domain (see Hofstadter, Mitchell, & French (1987) and Hofstadter & Mitchell (1988) for collections of such problems, and see Hofstadter (1985) and Mitchell (1988) for discussions of how these problems relate to "real-world" analogy-making).

We will explain the workings of the current version of the Copycat program by presenting two series of annotated screen printouts from actual runs on two problems. First, we discuss some of the ideas behind the model. The first idea is nondeterminism, which permeates the workings of Copycat. The program accomplishes its goals by executing a very large number of small pieces of code, called "codelets", chosen probabilistically from a constantly changing pool. Thus not only does each run differ from every other run, but also many different answers can be reached for a given problem. Thus to show just one run for a given problem (as we have done) is somewhat misleading. We have chosen two fairly typical runs for the two problems, but readers should bear in mind that other answers are often produced, and many other routes to the shown answers exist.

Copycat's nondeterministic nature is based on the idea that analogy-making, like perception, is highly and asynchronously parallel. In perception and in Copycat, many processes take place concurrently. In Copycat, each process consists of many small codelets, and codelets of different processes are interleaved probabilistically. Each process has a dynamically evaluated importance, so that favored processes can run faster. This is accomplished by giving each codelet an "urgency" -- a number that determines its probability of being chosen from the pool of codelets waiting to run.

The second idea is the building-up of a coherent view. It is up to the program to build up an understanding of each letter-string on its own, and also of how strings are related. This is very similar to a perceptual process. At the outset, each codelet picked has a small region of a string as its focus, and it looks for any local structure of interest there. If so, it suggests that that structure

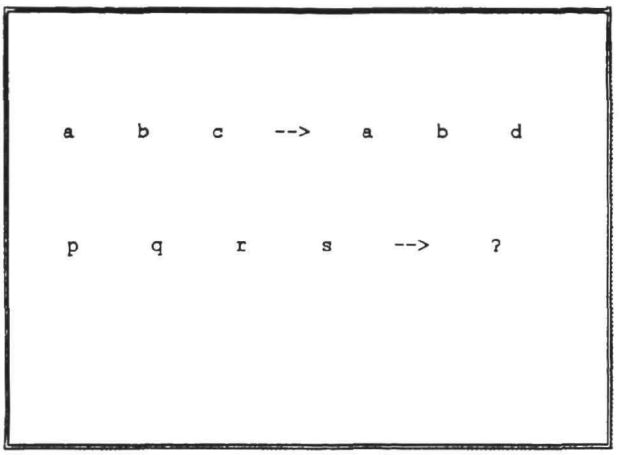
be officially recognized by another codelet. As many such codelets run, they gradually "annotate" individual letters and letter-strings, converting them from raw data into coherently understood structures. This process is very similar to the operation of the Hearsay II speech-understanding program (Erman et al., 1980), which took a raw speech waveform and allowed many processes to build higher-level hypotheses about it, upon which yet higher-level hypotheses could be built, at the top level of which emerged a totally semantic understanding of the utterance. The aim of Copycat is similar: to convert a raw letter-string into a totally understood situation.

Moreover, not only must Copycat knit together each separate string, it must also construct a coherent network of correspondences between the three given strings. These correspondences express Copycat's view of how certain parts of one structure map onto parts of another, without there necessarily being any one-to-one mapping involving all the parts. The build-up of local structure inside a given string tends to precede the build-up of correspondences between strings, but this is not an ironclad order; the nondeterminism allows these types of processes to take place concurrently. At the outset, intra-string processes are given higher urgencies, but as coherent views of individual strings gradually get built up, the urgencies of inter-string processes rise and so those processes become predominant. Thus activity gradually shifts from a local to a global scale.

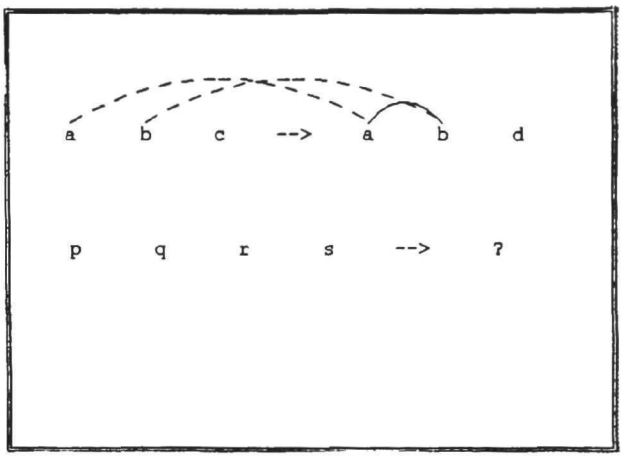
The third idea is that of conceptual distance and slippage. In any analogy worth the name, there are "conceptual slippages": mental correspondences made between things that are not identical. In Copycat, the plausibility of any such correspondence is determined by referring to a network of concepts called the "Slipnet", one of whose main functions is to define a "distance" between any chosen pair of concepts. The smaller the distance between the two concepts, the more plausible is a mapping in which they are considered counterparts (i.e., in which the one "slips" into the other). Of course, the smallest possible distance is zero -- when a concept is mapped onto itself. But an analogy in which all conceptual distances are zero would be a total identity. Thus non-trivial slippage is an essential ingredient of interesting analogy-making. As its name would imply, the Slipnet is the measure of all slippages.

The Slipnet is a dynamically changing network, in which conceptual distances change as a function of processing (thus as a function of context). There is a default setting of the Slipnet, in which concepts have "neutral" distances, but as certain slippages take place, they modify the distances between similarly-related concepts. For example, if a slippage between two concepts considered to be "opposites" is incorporated into an analogy, that shortens the distances between all pairs of "opposites" in the Slipnet, which tends to increase the likelihood of similar slippages. Each concept has a time-varying activation level: a function of the importance of the role the concept has been perceived as playing. When a concept is activated, its use in forming descriptions is encouraged. For example, if a group of any sort has been perceived, codelets attempting to create other groups of that sort will henceforth tend be more successful. Another crucial function of activation is that of determining the salience of all the objects in the situations at hand. Each object (letter, group of letters, entire string) has a number of descriptions, each of which consists of names of certain concepts in the Slipnet. To each description is attached a time-varying number that reflects how active those concepts are at the moment, and the object's salience is a simple function of those numbers for all its descriptions. Thus an initially unremarkable object can become strikingly salient if one or more of its descriptions involve highly activated concepts. The reason this matters is that codelets are highly biased towards acting upon salient objects. Thus there is an interesting reciprocal influence of the Slipnet and the perceptual processing of strings: the Slipnet determines what objects are most "interesting" as foci of processing, and the results of processing determine the level of activation of Slipnet concepts, which feeds back into the processing. Finally, activation spreads from a concept to neighboring concepts in the Slipnet, so that even if a concept is not directly involved in the situations, its conceptual closeness to concepts that are directly involved may cause it to be brought in. This allows unexpected associations to be brought in, even though they are not on the surface at all.

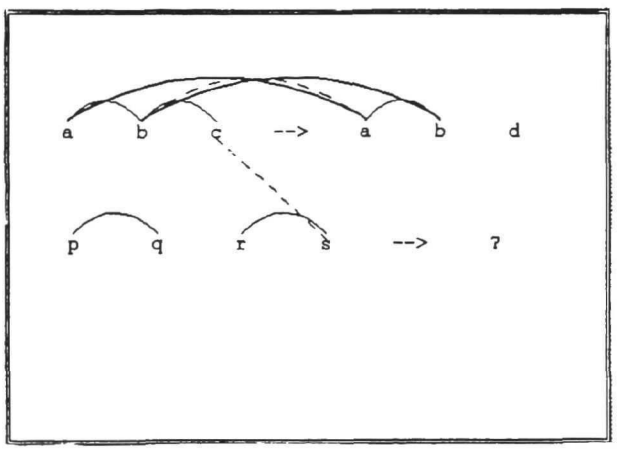
The following is a run of the program on the problem "If **abc** ---> **abd**, then **pqrs** ---> ?" This run produced the answer **pqrt**. This answer is almost always produced by the program, although on rare occasions it produces the rigid answer **pqrd**.



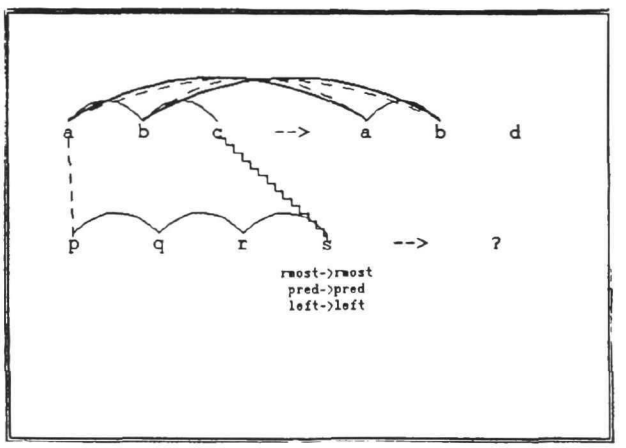
1. The program is presented with the three strings.



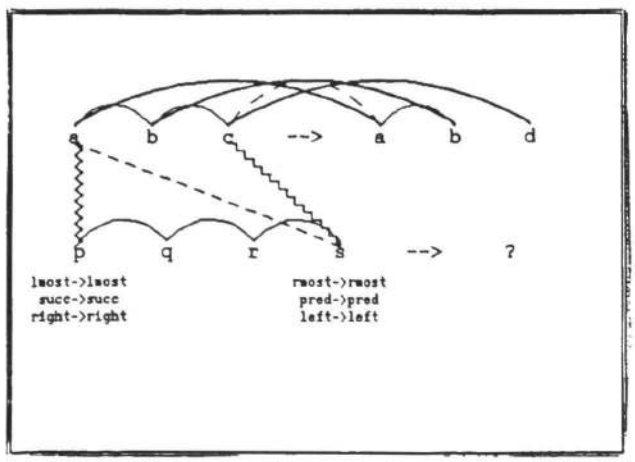
2. Tentative correspondences between letters in the two top strings are being considered (dashed arcs). A successor/predecessor relation has been noticed between the **A** and the **B** in **abd** (solid arc).



3. Correspondences between the two **A**'s and between the two **B**'s have been built (solid arcs). A competing correspondence between the **B** in **abc** and the **A** in **abd** is tentatively being considered. More successor/predecessor relations have been noticed. A tentative correspondence between the **C** in **abc** and the **S** in **pqrs** is being considered.

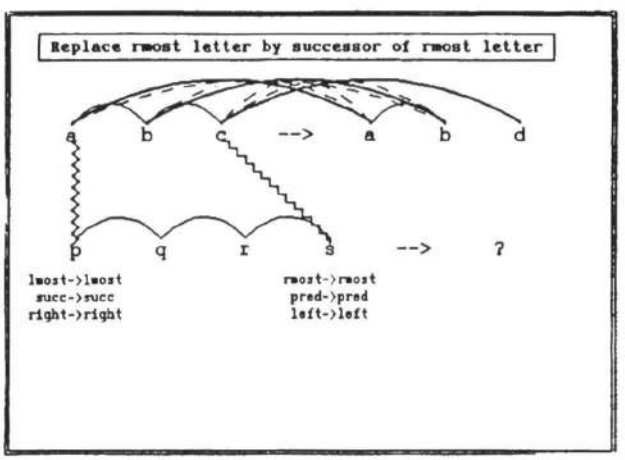


4. The **C**-**S** correspondence has been built (jagged line) and at the bottom are listed the three trivial slippages underlying it. The slippage "rmost --> rmost" means that both letters are rightmost in their respective strings. The slippages "pred --> pred" and "left --> left" indicate that each letter's left neighbor is its predecessor. A correspondence between the **A** and the **P** is being considered, which would be compatible with the **C**-**S** correspondence. Meanwhile, alternative correspondences on the top line are being considered.

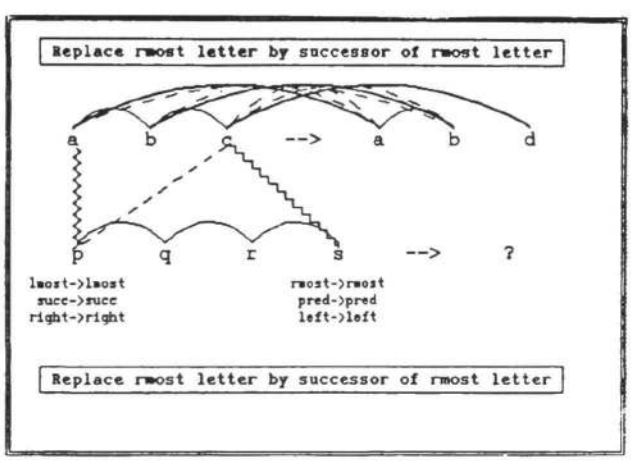


5. The **A**-**P** correspondence has been built, and there is a competing tentative correspondence between **A** and **S**, based on the notion that the **A** is leftmost and the **S** rightmost, two concepts close enough in the Slipnet to allow a correspondence to be considered, but, it turns out, not close enough in this context for that correspondence to compete. Incompatible tentative correspondences can coexist, but incompatible genuine correspondences cannot, so various incompatible sets of correspondences compete on the basis of strength.

The mapping between **abc** and **abd** is complete, but an alternative correspondence is being considered.

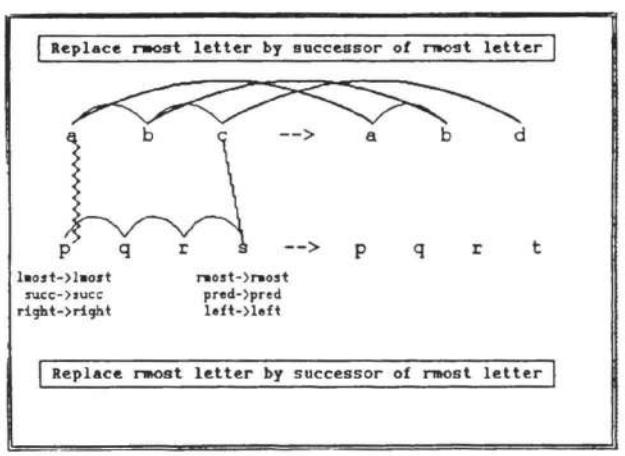


6. The tentative **A**-**P** correspondence was weak, and died. A rule describing the change from **abc** to **abd** has been written at the top. Copycat currently is limited to situations where just one object is changed, so rules are made by filling in a template of form "Replace _____ by _____ ". The rule-building codelet finds the changed object, and probabilistically chooses a description of it, preferring salient and abstract ones. E.g., "rightmost letter" is more abstract than "instance of **C**", though the latter is occasionally chosen. Likewise, a description of the corresponding object in the second string is probabilistically chosen.



7. The rule has been "translated" for use on the target string, and appears at the bottom. The slippages underlying the correspondences are used as translation rules; in this problem there is nothing to translate, since "rightmost", "successor", and "letter" play the same role in **pqrs** as in **abc**. As will be seen, however, this is not so when the target is **ssrrqqpp**.

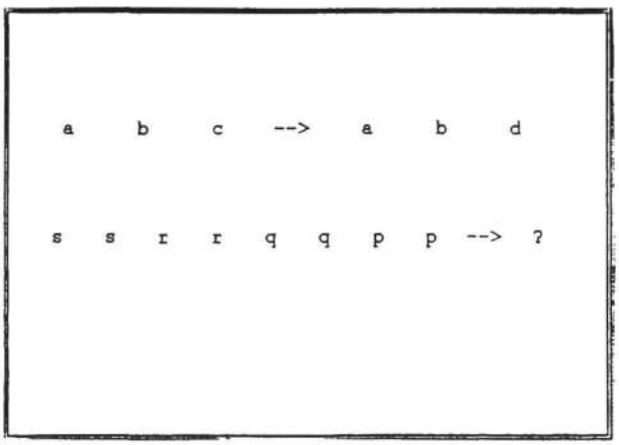
Even though the rule has been translated, alternatives to the mappings are still being considered. If any of these alternatives were to succeed, then the current rule would be discarded and a new one would have to be constructed.



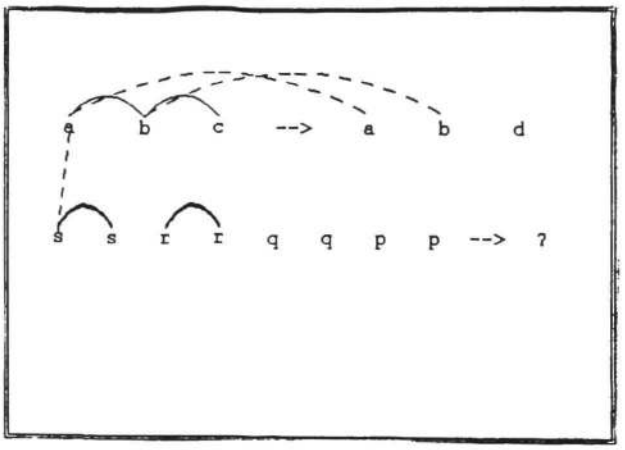
8. The program has used the translated rule to create an answer: **pqrt**.

Notice that no correspondence was ever made between the **B** in **abc** and anything in **pqrs**. This reflects the fact that in an analogy between two situations, not every aspect of each situation has to be mapped. In the case of these miniature situations, there is no good counterpart for **B** in **pqrs**.

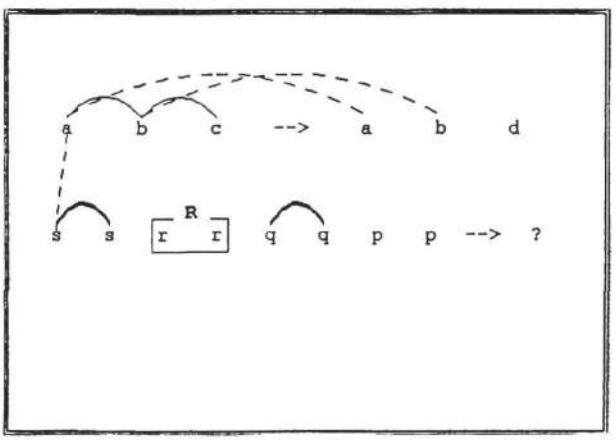
Next, the program is run on the problem "If **abc** ---> **abd**, then **ssrrqqpp** ---> ?" This run produced the answer **ssrrqqoo**, but note that the answer **ttrrqqpp** is also produced quite often, and on rare occasions rigid answers such as **ssrrqqpq** and **ssrrqqpd** are produced.



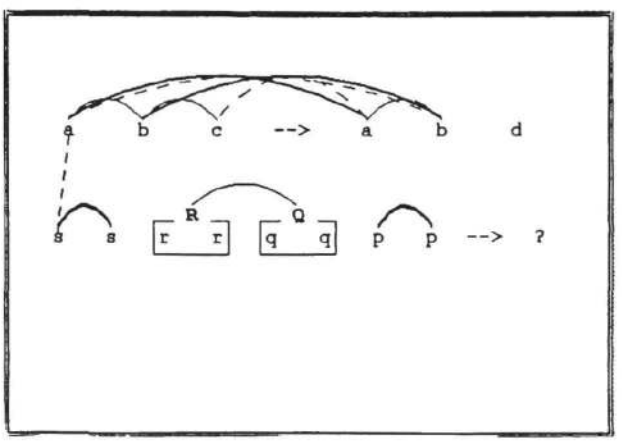
1. The program is presented with the three strings.



2. Successor/predecessor relations (light solid arcs) and sameness relations (dark solid arcs) between letters are beginning to be noticed, and some tentative correspondences have been set up between **abc** and **abd**. In addition, a tentative correspondence has been made between the **A** in **abc** and the leftmost **S** in **ssrrqqpp**.

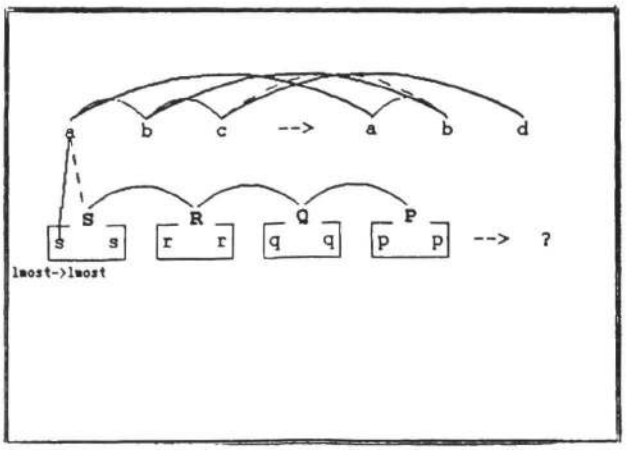


3. A group is formed out of the two **R**'s bonded by a sameness relation. The group is represented by a parameter-letter **R** (the boldface **R** appearing above the group). A parameter-letter acts much like a letter, but exists at a more abstract level.

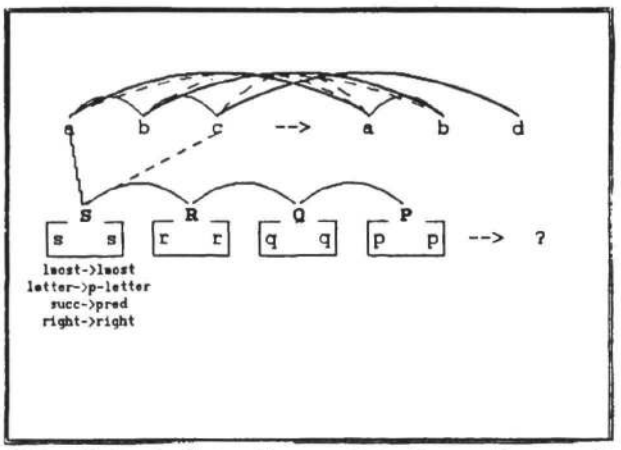


4. The group of **Q**'s has been perceived, and is characterized by the parameter-letter **Q**. This allows a successor/predecessor relation to be noticed between the parameter-letters **R** and **Q**.

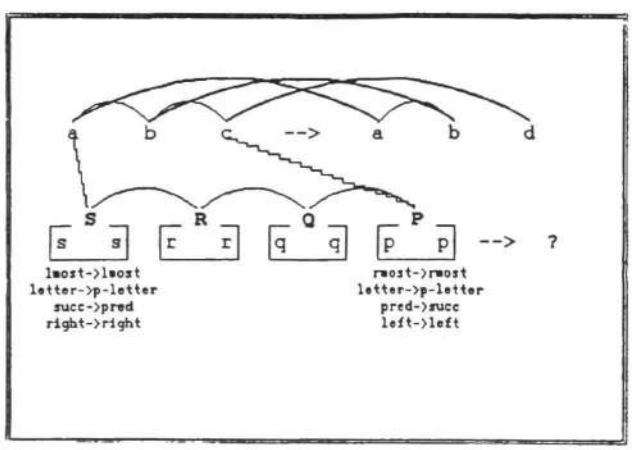
Some correspondences between **abc** and **abd** have been built, and others are being considered.



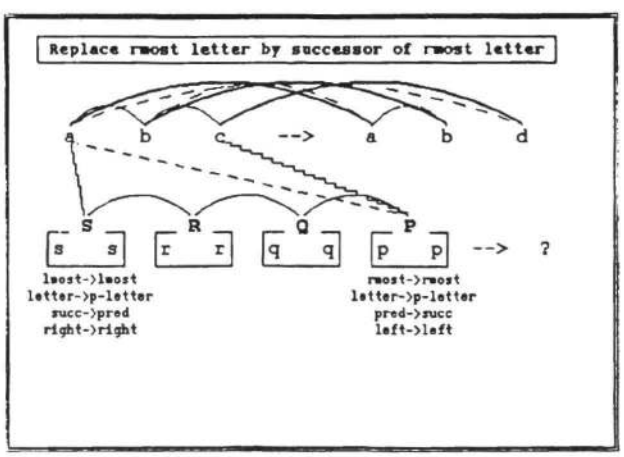
5. All the groups in **ssrrqqpp** have been noticed, and all the successor/predecessor relations at the parameter-letter level have been noticed. (The already-perceived successor/predecessor and sameness relations at the letter level are still present, but are suppressed from the graphics.) Also, a mapping between **abc** and **abd** has been completed. In addition, a correspondence has been built between the **A** in **abc** and the leftmost letter **S** in **ssrrqqpp** (jagged line). Its only (trivial) slippage is "leftmost --> leftmost". However, a rival correspondence is being considered between the **A** and the parameter-letter **S**.



6. In a fight, the latter destroyed the former, because it is supported by more slippages (4 vs. 1), meaning more similarities are being taken into account, and because they involve more abstract concepts (like "parameter-letter"), meaning deeper similarities are being taken into account. Its slippages tell us: "leftmost" plays the same role in both strings; "letter" in **abc** corresponds to "parameter-letter" in **ssrrqqpp**; "successor" to "predecessor", and "right" to "right", since **abc** increases alphabetically to the right, while **ssrrqqpp** decreases (at the parameter-letter level). Meanwhile, "diagonal" competition has appeared.

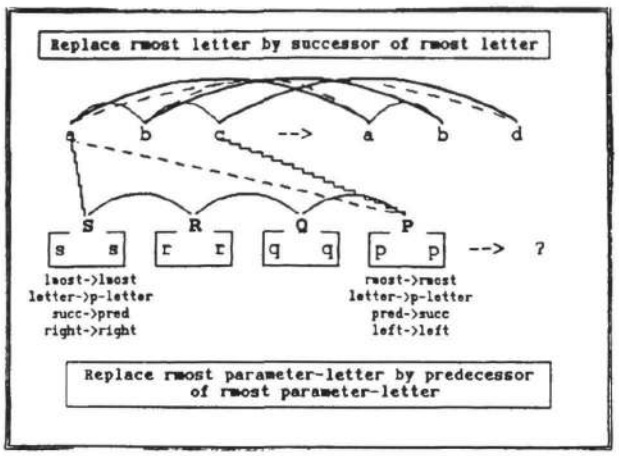


7. The diagonal competition is gone. In contrast to the previous problem, diagonal correspondences (representing the view that the strings have the same alphabetic order, but opposite spatial directions) are as strong as the vertical correspondences (representing the view that the strings are read in the same spatial direction, but with alphabetic direction reversed). Both these views are reasonable, though only one can exist at a time. The diagonal mapping lost only because of an unlucky throw of the dice. The compatible correspondence (from **C** to parameter-letter **P**) has been built, completing the vertical mapping.

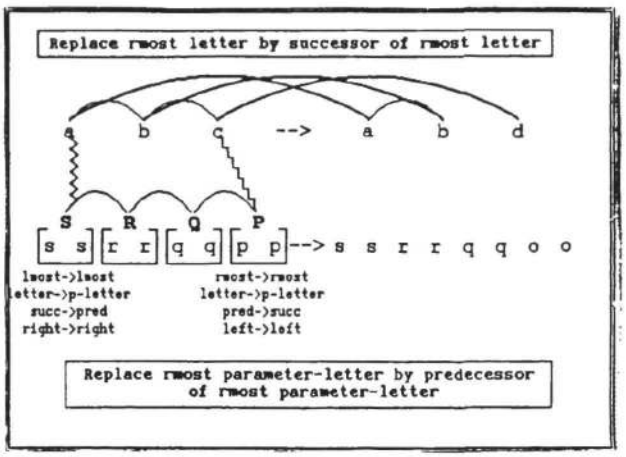


8. A rule expressing the change in the first line has been constructed (the very same one as was constructed in the first problem). But note that if it were applied directly to **ssrrqqpp**, it would yield the rigid and unappealing answer **ssrrqqpq**; therefore, it will have to be translated.

Another attempt is being made to construct a diagonal mapping (here between the **A** and the parameter-letter **P**). If it were successful, then the two existing correspondences would be destroyed.



9. The rule has been translated according to the translation recipes embodied in the slippages underlying the correspondences.



10. The answer **ssrrqqoo** has been created according to the translated rule. If the diagonal correspondences had won, the rule would have been "Replace the leftmost parameter-letter by the successor of the leftmost parameter-letter", yielding answer **ttrrqqpp**.

These runs demonstrate Copycat's current capabilities; solving the subtler analogy problems mentioned above will require some additions to the architecture. Our plans for future work are discussed in Hofstadter, Mitchell, & French (1987) and in Mitchell (1988). We are also testing the generality of our approach by using similar architectures in different domains (Hofstadter, Mitchell, & French, 1987; Meredith, 1986). Comparisons between Copycat and other models of analogy-making (especially work by Gentner, 1983 and Falkenhainer et al., 1986, and by Holyoak and Thagard, 1987) are given in Hofstadter, Mitchell, & French (1987) and in Mitchell (1988).

Acknowledgements

We thank Robert French for many important contributions to this project. This research has been supported by a grant from the University of Michigan, a grant from Mitchell Kapor, Ellen Poss, and the Lotus Development Corporation, a grant from Apple Computer, Inc., and grant DCR 8410409 from the National Science Foundation.

References

[1] Erman, L.D., F. Hayes-Roth, V. R. Lesser, & D. Raj Reddy (1980). The Hearsay-II speech-understanding system: Integrating knowledge to resolve uncertainty. Computing Surveys, 12(2), 213-253.

[2] Falkenhainer, Brian, Kenneth D. Forbus, & Dedre Gentner (1986). The structure-mapping engine. In Proceedings of the American Association for Artificial Intelligence. Los Altos, CA: Morgan Kaufmann.

[3] Gentner, Dedre (1983). Structure-mapping: A theoretical framework for analogy. Cognitive Science, 7(2).

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

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

[6] Hofstadter, Douglas R., Melanie Mitchell, & Robert French (1987). 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, Michigan.

[7] Hofstadter, Douglas & Melanie Mitchell (1988). Concepts, analogies, and creativity. To appear in Proceedings of the Canadian Society for Computational Studies of Intelligence. Edmonton: Univ. of Alberta.

[8] Holyoak, Keith J. & Paul Thagard (1987). Analogical mapping by constraint satisfaction. Manuscript submitted for publication.

[9] Meredith, Marsha J. (1986). Seek-Whence: A model of pattern perception. Unpublished doctoral dissertation, Indiana University, Computer Science Department, Bloomington, Indiana.

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