Well, I spent an hour working on this message this pm, only to have the system crash on me and lose it. So, once more into the breech....

As an overview, I'm assuming the entire analogy system will include the following modules:

1) standard rule-firing problem solver
2) EBL-massaged storage of solved problems
3) similarity-based retrieval
4) mapping engine
5) transfer
6) generalization by intersection (schema formation)

A truly staggering package, if it could really be done!

With respect to (3), you are quite right to question the degree to which retrieval is systematic. It is *not* the case that reminding quarantees transfer. The retrieved analog may be superficial and hence useless, or the mapping may be too complex ("Gee, these problems seem alike, but I'm not sure exactly how"--we get that sort of thing with Laura Novick's math analogies). If I have time I'll come back to thoughts about (3). In brief, I like your idea of deriving similarities in part from similar rules. Indeed, for some functional categories that may prove crucial.

For now, however, let's assume (3) is worked out, and the system has found two analogs to compare. As we'll see, similarity of elements can be used to guide mapping, but ACME actually doesn't require it.

A. WHY ACME?

Why not? More seriously, mapping has all the hallmarks of the class of problems that can be neatly solved by constraint satisfaction. There are multiple constraints, such as the following (all embodied in ACME):

CONSTRAINT 1: Only logically comparable elements can map (constants with constants, n-place predicates with n-place predicates).

CONSTRAINT 2: Elements of initial-state descriptors map onto initial-state descriptors, goal-state descriptors to goal-state descriptors.

CONSTRAINT 3: Each element maps to just one element (or to nothing).

CONSTRAINT 4: Argument structure is preserved under mapping.

CONSTRAINT 5: The more similar the elements, the more likely they map.

But none of these constraints are absolute. Dissimilar elements may play analogous roles; argument mapping may be imperfect (the army is like the rays, but also like the flesh in being endangered); some elements may not map at all. In your coherence algorithm, mapping starts by assuming the two most similar elements map. This is a reasonable heuristic, but the serial nature of the process can lead to arbitrary decisions (what if 2 pairs of elements are tied for most similar?). ACME is really quite similar in spirit to your algorithm, I think, except it

sets up all constraints and then lets them all influence mapping in parallel, like a set of simultaneous equations.

B. PRELIMINARY ASSUMPTIONS

I'll describe the algorithm as if we were drawing analogs between problem representations with 2 components: initial state and goal state. This seems to mean avoiding predicates like "causes" that take propositions as arguments, which I understand we introduce as rules rather than state descriptions. However, I think in the end we may want to derive mappings such as the following:

cause (apply (high-intensity (ray)), tumor), (destroyed (tumor)))

cause (apply (high-intensity (laser)), filament), (fused (filament)))

I think the description below is extensible to more complex relational structures, but what to do hinges on how we represent operators in problems.

ACME has two major parts:

1) Procedure for setting up a constraint-satisfaction network. This is specific to analogy, and has nothing to do with connectionism.

2) Relaxing the network into a "best fit" mapping. This is just the Rumelhart/McClelland equation on p. 11 of PDP Vol. 2.

C. SETTING UP THE NETWORK

Let's take a simple abstract example to play with. Let capital letters represent predicates, small represent constants, early letters the source analog, later letters the target. Here is a possible analogy:

SOURCE	TARGET
A(a)	R(r)
B(b)	S(s)
C(c)	T(t)
D(a,b)	U(r,s)
E(b,c)	V(s,t)
F(a,c)	

Our goal is to find the "best" correspondences between predicates and between constants, which obviously are:

A=R B=S C=T D=U E=V F=0
a=r b=s c=t

Note that source has a predicate F with no map to target (0 means null map). Later transfer stage may do something with that. ACME just finds there is no map. More generally, the system will (hopefully) prove robust in the face of less-than-isomorphic analogs.

We let hypothesized mappings be units; weights on connections represent constraints; activation levels on units represent degree of confidence in hypothesized mappings. (Just like Necker cube figure on p. 10 of PDP Vol. 2.). Network is built as follows:

STEP 1: BUILD UNITS

For each element of source and each element of target, form unit for each possible hypothesis. Here we build in Constraint 1: only logically comparable elements can map (constants with constants, n-place predicates with n-place predicates). This may be unduly restrictive, but seems like a desirable simplification for now. (It is equivalent to having 0 weights on all links leading to "illegal" mappings.) So the following units are built:

```
A=R  A=S  A=T  A=0
B=R  B=S  B=T  B=0
C=R  C=S  C=T  C=0

D=U  D=V       D=0
E=U  E=V       E=0
F=U  F=V       F=0

a=r  a=s  a=t  a=0
b=r  b=s  b=t  b=0
c=r  c=s  c=t  c=0
```

Thus we consider all possible pairings of 1-place predicates with 1-place predicates, 2-place with 2-place, and constants with constants, plus in each case the possibility of a null map.

In a further initial restriction (not illustrated in example), we can limit hypotheses to mappings between elements of initial-state descriptors and other initial-state descriptors, and goal-state to goal state (Constraint 2). Thus for ray problem the predicate DESTROYED will only generate units mapping it with 1-place predicates of *goal* states in the analog.

Finally, we add one special unit representing the entire situation (see below).

STEP 2: ADDING CONNECTION WEIGHTS

NB: All connections are symmetical.

a) For all units with a particular predicate or constant, put in inhibitory links to all other units that mention that element. This embodies Constraint 3: each element maps to just one element (or to nothing). In above diagram, this amounts to having inhibition between all units within each of the three sets (2-place predicates, 1-place predicates, constants) that are in either the same row or same column. (Exception: units with the element 0 do not inhibit its other occurrences, since more than one element can have a null map.)

b) For each hypothesis about a predicate map, put in excitatory links with the units for corresponding mapped arguments. This is Constraint 4: Argument structure is preserved under mapping. For example, the unit for D=V will have excitatory links to a=s and b=t. Likewise, a=s will excite D=V and b=t; b=t will excite D=V and a=s.

NB: the hypotheses about the element 0 (e.g., D=0) receive inhibition from rival hypotheses (like D=V), but have no excitatory links. Thus if every possible mapping for an element is ruled out, the element=0 unit will weakly win (I think).

Finally, if I follow the book, we should have one special

unit representing the whole situation. It should have excitatory connections to all other nodes, reflecting the fact that all hypotheses are at least possible.

D. RUNNING THE NETWORK

Now we're about ready to let 'er rip. First, though, we need to set the bushel of free parameters us slippery bandwagon-connectionists get to play with. Trial-and-error (and/or Charlie) will be needed here. I guesstimate as follows, from numbers used in McClelland's TRACE model (p.76):

1) Activation levels of units range from 0-1. All start at 0, except for the special unit, which is clamped to 1.

2) All inhibitory links: .02

3) All excitatory links other than those from special unit: .04

4) The trickiest and most interesting are the excitatory weights from the special unit to the hypothesis units. Two versions (at least) are possible:

a) All weights are some low positive value, like .01. This means the system has no a priori beliefs about the possible mappings.

b) The weights reflect the computed similarity of the mapped elements, embodying Constraint 5: The more similar the elements, the more likely they map. Under this version, which we should surely use, the output of retrieval (or very 1st step in mapping) is to calculate the similarity of predicates. The higher the similarity, the higher the weight from the special unit (perhaps in range .01 to .05). Since constants are by definition semantically empty, their units all still get low weight of .01.

Now, off we go, following the recipe for updating given on p.11. The system will stabilize with the units at activation levels reflecting the plausibility of each possible mapping. For each element, one map (possibly 0) will be most active, hence "best". Importantly, the Hopfield equation (p. 13) will describe the overall "goodness" of the analogy. (Technical question for Charlie: is the value of G really comparable across different analogies, which may have network structures with different relational structures and different sizes, depending on how many propositions represent each analog?)

Something else to note: in the Necker cube runs presented on pp.12-13, the third run gives an impossible figure. Text says this is because stimulus gave big excitatory weights to certain hypotheses. In our terms, if person thinks two predicates are highly similar, hence puts large weight on excitatory link from special unit to that hypothesis, but in fact the two elements are not analogous, the system may get trapped--thus we have a model of how misleading surface similarity could interfere with finding correct mapping. Note our system, under option (a) above, can be directed to place little or no weight on predicate similarity. Mapping can then be found even if similarity is misleading. Thus ACME can both model human heuristic use of similarity in mapping, and also "rise above" such mundane considerations. Indeed, in the purely abstract analogy used as the example above, obviously there is no similarity of elements at all--just a similar relational structure. This is like an abstract mathematical isomorphism. ACME can handle such cases; unlike Gentner (or any other mapping procedure I know of), mapping does *not* *require*

identical elements (although similarity/identity will usually play a role in making analogies easy or hard to process).

Well, in the interests of modularity I'll end here. What do you think? --Keith

\001\001
Date: Fri, 15 May 87 13:48:31 PDT
From: K Holyoak <holyoak>
To: pault@mind.princeton.EDU
CC: holyoak
Subject: Re: Canuckshionism.
In-reply-to: Message of Fri, 15 May 87 15:12:23 EDT
from "pault@Princeton.EDU"
<8705151917.AA25868@mind.Princeton.EDU>
Message-ID: <870515.204831z.06679.holyoak@PEGASUS.CS.UCLA.EDU>

Gee, I'm glad you think ACME/NADIR has some hope. I too am worried that we are carrying our famous eclecticism to dizzying heights: imagine, a program that integrates (concatenates?) Mitchellian EBLism, Winstonian intersectionism, Andersonian activationism, Rumelhartian neurologism, Simonesque productionism, with a dash of Hollandaze, and manages to solve the ray problem! Far freakin' out! This is canuckshionism at its NADIR.

I really think, though, however monstrous the connectionist piece of the model is, it behooves us to do it and see what happens. Indeed, now that I've thought about it, doing mapping by constraint satisfaction fairly screams to be tried, and I'm amazed that none of the evangelicals have done it yet. Rumelhart and Lakoff will surely drool if we beat them to this proposal.

On your specific points:

1) Yes, the network has to be built each time mapping takes place. I agree its more plausible at mapping than retrieval. There are some ways I think the model could be eventually psychologized a bit more. For example, maybe people only form a subset of the possible hypotheses. But that can wait.

2) The similarity calculations do not have to be as much of a bugaboo as you may think. If I'm right, the system will actually converge on the best relational structure even if initial similarities of predicates don't vary. Indeed, here is what may be a plausible procedure for using similarity:

a) The output of retrieval is to report that certain predicates are quite similar, like laser--x-rays, because these triggered reminding. Give these relatively high weights from the special unit.

b) All other weights to mapping hypotheses (e.g., from the special unit to tumor-lightbulb) are set at some uniform low positive value, like .01. Thus there is no special computation of similarity required for mapping. The predicates that were similar enough to aid in retrieval are given high values, others a low value. In fact, given that predicate similarity is only a rough guide to appropriate mappings, this crude use of similarity may be at least as good as one that requires explicit calculation of all predicate similarities.

3) I think G may distinguish the good vs bad version of lightbulb analogs. The good version has the goal proposition, intact(bulb), whereas the bad version has the initial condition, lacking(high-I(laser)). The former will map onto the ray goal, alive(patient) {or maybe uninjured(flesh)}, whereas the latter will have no map into the ray problem. Actually I'm not at all sure G will have all the right properties to be a general yardstick of analogical goodness. The more crucial thing is that bad analogs should give less evidence that the useful operators should be transferred from source to target.

One thing that may bear on the above is how we treat the 0 map. In yesterday's proposal, hypotheses such as predicate=0 are given low excitatoru

weights from the special unit, just like any other low-similarity mapping. Intuitively, though, the mapping should be worse if it is necessary to assume some elements don't map. An alternative way to handle null maps might be:

a) special unit has weak *inhibitory* links to all null-map hypotheses, creating pressure to try to map all elements onto some real element.

b) Consider the following set of units:

A=R A=S A=T A=0
B=R B=S B=T B=0
C=R C=S C=T C=0
D=R D=S D=T D=0

In the story as of yesterday, we put inhibitory links between all units that share a common element (e.g., A=R inhibits A=T because these are competing possible mappings). This amounts to inhibiting all units in the same row or column (except the rightmost X=0 column doesn't mutually inhibit).

Now add the following addition: each unit is given a weak excitatory link to all units that it doesn't inhibit. That is, each unit excites those units that do *not* share a common element; i.e., those not on the same row or column. (Again the rightmost X=0 column is an exception; the 0 maps neither inhibit nor excite each other.) The rationale is that if there is evidence that A=R, for example, then there is evidence that B does not =R; accordingly B must = something else (B=S, B=T, or B=0). The effect of this will be that if all other alternatives have been ruled out, the X=0 unit will get enough positive excitation to outweigh the inhibition from the special unit. E.g., suppose A=R, B=S, and C=T are highly active. They will then strongly inhibit D=R, D=S, and D=T, which therefore can't inhibit D=0. Furthermore, each of the three highly active units will weakly excite D=0. Thus D=0 will weakly win. But because the initial constraint represnted ny the inhibition from the special unit to D=0 will have been violated, the G measure will indicate this analogy is worse than one in which there was some real target predicate U such that D=U.

Note this means that if new predicates are invented for target as part of transfer, the effect will be to *improve* the G measure, because the mapping will be extended by removing a null map and substituting a real one. At least, that's my hope.

4) On a related note, you are absolutely right to include operators in mapping from the start. Transfer rule may be something like, if a source operator that participates in solution has a null map to target, propose an analogous source operator.

5) You say we have syntax and semantcs, but where did the pragmatics go? It's really still there I think. Note the constraint I put in that Goal elements only form units hypothesizing mappings to Goal units, Initial elements to Initial elements (and now, Operator elements to Operator elements). Also, remember that EBL is serving to ensure that only causally relevant aspects of source ever get stored, so as to possibly be mapped later. (The physicist's white lab coat is out right there.)

6) Your suggestion that we reduce Gentner's examples to NADIR is dead on. Indeed, in my current ACME-induced hubris the malicious idea of presenting our mapping of the solar-system--atom at Psych & Phil has crossed my mind. The extension required, I think, is to add propositions to the list of elements being mapped. Yesterday we had constants and predicates forming units; add that propositions map onto propositions. Then we can handle things like

cause (R (x,y)), (S (u,w)))

which would result in setting up maps between the propositional arguments like R (x,y) and propositions in the analog (e.g., we would have units like R(x,y)=C(a,b). Each proposition mapping then excites each corresponding predicate mapping, and each corresponding constant mapping, and all combinations thereof.

I will continue to ruminate over retrieval. --Keith

\001\001

Paul:

Well. I finally was able to read all your messages after our computer got straightened out. A version of this message was written several days ago but I was unable to send it. It is now edited somewhat in light of your recent achievements. Further messages will discuss:

1) I have a suggestion for an empirical test of the adequacy of the mapping-transfer model we're working on. It involves data from studies of the good old missionaries and cannibals problem, which shows that for homomorphs, you get good transfer from more constrained to less constrained problem, little in the reverse direction. This may be a fairly simple test of whether our model is doing sensible things with less-than-isomorphic analogs.

2) I think I have thought of an algorithm for doing structured retrieval of the sort Ziva demonstrated. It is an extension of an idea involving marker passing, which I believe I was babbling about several messages ago ("color-coding" activation traces). Now I think I see how to make it work. Needless to say, this insight may dissolve overnight. We'll see.

In this message, I will just cover details relevant to finalising acme.

FINALIZING ACME

Given the rapidity of your progress last week, it looks like we are close to being able to put acme to a series of (hopefully publishable) tests. I think we should pause a moment to make sure we're in agreement about exactly how the model is working, so when we get into systematic tests we're able to minimize backtracking later on. Incidentally, are you still planning to do SACME? And are we going to get Gentner's SME into the picture? I believe you are exactly right in your assessment of how SME can be viewed as a special case of acme, by the way. (As an aside, my intuition is that the best psychological version of acme will not be a serial approximation, but a quasi-parallel one--QUacme?--that is basically acme except that only a subset of units get established, with some degree of seriality, due to working-memory limitations. But for now I believe our best bet is to continue to focus on acme as an idealized computational-level description of mapping. If it proves to be superhuman in some respects, I can cheerfully live with that.)

I'm sure the viability of acme as a psychological model will be a matter of debate. But its claim for merit is a certain elegance that lends credence to the demonstrations that it really does compute interesting mappings. For this reason I think we should be compulsive in ensuring we make the program as above reproach on the "elegance" criterion as possible. (None of that "look ma no hands" stuff; rather, two hands on the handle-bars at all times.)

A. CONSTRAINTS

First, let's check whether acme has all the constraints originally planned. These are (from original acme message):

CONSTRAINT 1: Only logically comparable elements can map (constants with constants, n-place predicates with n-place predicates).

CONSTRAINT 2: Elements of initial-state descriptors map onto initial-state descriptors, goal-state descriptors to goal-state descriptors. **[And by extension, operators to operators, constraints to constraints, if we have such a category.]**

CONSTRAINT 3: Each element maps to just one element (or to nothing).

CONSTRAINT 4: Argument structure is preserved under mapping.

CONSTRAINT 5: The more similar the elements, the more likely they map.

From your messages, you clearly have (5), and also (4). [The latter was Step 2b under "ADDING CONNECTION WEIGHTS" in the original acme message.] Do you have (3)? This I couldn't tell for sure, as you didn't explicitly mention building inhibitory units. See Step 2a in original message: For all units with a particular predicate or constant, put in inhibitory links to all other units that mention that element.

As stated above, Constraints 3-5 are explicit and matters of degree, as they are explicitly represented by weights on links. Constraints 1 and 2 are absolute, in that units that violate them are never even built. They implicitly correspond to units with all 0 weights on links leading to them (thus guaranteed to have activation levels that remain at 0). Are (1) and (2) reflected in program? (1) is the one I have least theoretical committment to, but seems a useful convenience; (2) is the clearest pragmatic constraint on the mapping process--it uses problem structure to cut down on the number of possible maps considered.

Another thing I wanted to check: did you include units representing null maps (X=0), to accommodate elements that have no map? I was a bit unsure how these would work out, and could imagine doing without explicit null maps, as a null map could also be found indirectly if all real maps for an element are ruled out by competing alternatives. The latter approach may be preferable, since I think it will ensure that absence of a map for an element will reduce G.

B. TREATMENT OF SIMILARITY

I see from one of your messages that you indeed are modeling the similarity effect by setting inital activation levels in proportion to similarity. Although this is certainly a reasonable first approximation, I would like to continue to urge that this be changed to the "special unit" idea: a special unit has links to all hypotheses about predicate mappings, with weights proportional to similarity. That is, a process of similarity calculation feeds into a su with weights to all hypothesis units proportional to the similarities of the mapped predicates. There are several reasons for this, as noted in previous message. Chiefly:

(1) Elegance criterion: I would like to be able to say that all *constraints* on mappings are represented by *weights* on links, whereas *degree of confidence* in a mapping is represented by *activations* on units. The former, of course, determine the latter. The present implementation violates this generalization.

(2) Potential empirical implications: I think there are some tricky things about the correct treatment of similarity in the

model, and the su idea will allow us to deal with them better. Of the results you've already got, the one that decreasing similarity of laser and ray diminishes G, hence analogical goodness, is the most empirically questionable. Gentner, I believe, has evidence that people think *less* similar objects make better analogies. I guess I'm enough of a syntactician to feel that the "bottom line" on analogy is relational structure (Constraints 3 and 4): if the mapping is consistent, its a good analogy. My intuitiion about similarity is: It's a heuristic for *starting* the mapping, which will be found more readily if mapped elements are similar, and more slowly (or even not at all) if it turns out similar elements don't map. But ultimately, it's the internal relational constraints that should determine the asymptotic goodness of the analogy.

There are two ways we can potentially deal with this, not mutually exclusive, and both best formulated with the su units coding similarities in terms of weights:

(a) Under this coding system, similarities are exactly analogous to "perceptual input" in the necker cube example. (As an aside, the development of acme is itself a clear example of the use of analogy, as I literally worked it out from the necker cube case. When done, perhaps we can get acme to discover itself! [AI hubris again....]) Note that Hopfield's G measure decomposes into 2 parts, degree to which internal constraints are satisfied, and degree to which constraints of perceptual inputs are satisfied. We can easily compute these two components separately. Then we have measures of satisfaction of relational constraints (G1) and of similarity constraints (G2), as well as global G (G = G1+G2). The current version does not allow calculation of G2, the specific contribution of similarity. (I believe, in fact, it must not be giving a theoretically interpretable "degree of contribution" of similarity to the overall G.)

(b) We can make similarity constraints higher at first, then "fade them out". (I think your current version is doing something like this, but not in a sufficiently principled way.) It makes sense that as mapping proceeds, the impact of the prior process of similarity assessment should gradually be reduced. Increasingly, the mapping should depend on the internal consistency of the mapping of arguments, rather than on a priori similarity of predicates. This can be naturally modeled as follows: The su is initialized at activation = 1. It is then allowed to decay away to 0 (or to some parameter that sets its minimal activation). The result will be that similarity of predicates will have less and less impact as mapping proceeds. (See next section for a refinement of this idea.)

C. DECAY PARAMETERS

In the present simulation, as in the necker cube example in Ch. 14, at asymptote each unit goes to either 0 or 1--no intermediate values. This may lose valuable information, like what was the "second best" mapping (army is mainly like rays, but a bit like flesh). I think it may be useful to include a decay parameter in the activation-updating equation, which I believe will yield intermediate asymptotic activation levels. That is, on each update you add the net input to the unit after reducing its previous activation level by some fraction. This is in fact the general form of the Rumelhart/McClelland activation rule, which is given as an equation on p. 72 of Vol. 1 (the theta parameter). According to my math intuition, if activation tends

to decay, then asymptotic level will be the value at which the net input to the unit is exactly offset by the amount of decay. The higher the net input, the higher the asymptote (assuming decay is a constant for all units).

The potential advantage of the above is that in addition to getting an overall measure of the goodness of the analogy, we can also get a continuous measure of the strength of individual mappings at asymptote (e.g., army=rays).

Note that the suggestion of allowing the su to decay (see above) can be partially or perhaps completely assimilated to the general use of decay. The su, which starts at activation=1, will not get any further inputs. If we simply update it like other units, it will gradually decay. I believe, however, we may want to make the decay rate differ (be slower?) for the su. So I recommend establishing 2 distinct decay parameters, one for the su and one for all other units. We can always set them equal, or indeed set both to zero, if this decay idea doesn't prove useful after all. But for purposes of exploration, it would be nice to have the program written to include decay parameters, which we can always set to 0 if we want to ignore either or both of them.

I just realized that the last para. contains an important error: If all connections are symmetrical, as we assume, then the su will receive inputs back from the units it supports (e.g., laser=rays). So as long as these units are in fact consistent with the relational structure, and hence remain active, they will tend to keep the su active. But if the actual mapping violates the similarities, it should be possible for the su to finally get turned off. One of the tests I'd like to do is to model a Gentner expt. in which similarity *mismatches* analogical structure to some degree.

I therefore arrive at the following tentative recommendation: Try using a single decay parameter that applies to the su and all other units. Then the similarity effect will not, in general, fade away (I think), unless similarity actually doesn't predict the mapping. I also think we will want to have the weights based on argument structure to be higher than those based on similarity. To account for the apparent fact that people can judge goodness of analogy separately from similarity, we can simply assume they have access to both G1 and G2, and base analogical goodness judgments on G1.

D. RATE OF CONVERGENCE ON ASYMPTOTE

An important implication of the view of similarity I'm pushing is that it primarily affects the *rate* at which acme converges on asymptotic G, rather than asymptotic G itself. It will be useful to have the program produce a plot of G over iterations, so we can inspect the rate at which different analogies converge. I expect this may also imply that you should lower all weights so that convergence is slower, thus making differences in rate easier to see.

In sum, I'm suggesting the following measures should be tracked as we do our simulations:

(a) G1 and G2, which add to G
(b) asymptotic activation of the "best" maps, which with decay in the system will not always be 1
(c) rate of change in the above over iterations

E. ASYNCHRONOUS UPDATES

Finally, a purely technical matter. I assume you're now doing the most obvious updating procedure: updating activations of all units in simulated parallel fashion (synchronous updates). The PDP books emphasize that Hopfield's proof that the system will tend to minimize G depend on two assumptions:

(a) all connections are symmetrical (which I assume we've got)
(b) updating is asynchronous. That is, rather than updating all units in parallel on each "tick", you randomly select one unit to update, then randomly select again (with replacement).

In the interests of compulsive purity, I therefore recommend we shift to an asynchronous updating procedure. For purposes of plotting rate of convergence, we can look at system each time n units have been updated, where n is the number of units. It will be the case that in each such time step, by chance some units will have been updated more often than others, but that should only be a minor source of noise.

F. PROBLEM REPRESENTATIONS

I thought the representations you suggested for missionaries and cannibals in your last message looked good. I think I may have misunderstood the problem. It seems it doesn't have to do with representing higher-order relations in Gentner's sense of predicates that take propositional arguments, but rather representing propositions about variables rather than constants. I believe your suggestion is the same thing that Mitchell and the EBL folks do when they are generalizing from an example.

Here's the point I made on phone, as originally written:

Thinking about the missionaries and cannibals problems reopened for me the question of whether "constraints" on problem solutions should be treated just like goals. In the current plan, the fundamental problem schema is: initial state, goal state, operators. In the ray problem, the constraint that patient must not die is listed as a proposition in the "goal state" slot: alive (patient). That seemed sort of ok.

But in M&C, we have constraint: cannibals must never outnumber missionaries on same side of river. This seems different from the goal: all Ms and Cs on far side of river. The latter must only be true in final state, but the constraint must hold at every step on path to goal. Yet more clearly, consider the constraint: find a solution in minimum number of moves. This is not a description of the goal state per se, but rather of the solution as a whole. Based on these considerations, I suggest you consider moving to a 4-slot problem representation: initial state, goal state, constraints, operators. This is what Jaime C assumes I believe. I presume nothing much hinges on this as far as mapping goes, but it may be useful for other purposes. For example, a projection should be stopped whenever a constraint is violated, whereas of course it should not be stopped just because the goal state has not yet been realized.

I fear this message is filled with tedious trivia. Nonetheless, I think we should nail down details of this sort as much as possible now, before investing lots of time in fitting acme to data. If you diagree with any of the above

recommendations, let's thrash it out on phone.

On a more general level, here is a tentative list of computational experiments for a paper:

1) Simulate H&Koh (especially total solutions: retrieval part more properly belongs in the *next* paper).

2) Show sensitivity of both G and our transfer process (let's not forget the pragmatic computation of the operator(s) to pass from source to target) to:

(a) similarity (which may be complex, as discussed above)
(b) order of arguments (violating structure)
(c) dropping or adding unmappable propositions to source and/or target.

All of (2) can be done with lightbulb and ray problems.

(3) Model Gentner's solar system/atom examples, just to show we can do anything SME can do.

(4) Model Gentner's results on negative effect of object similarity mismatching relational structure. (I'll send you a message explaining this at some point--it will be easy I think).

(5) Modeling asymmetries of transfer between Miss & Cann, Farmer's Dilemmna problems.

(6) (Maybe): show acme can do mappings between some formal analogs (e.g., arithmetic-series problems and constant-acceleration problems, studied by Miriam Bassok and me). I'm still thinking about this one.

Also, if you can't readily get your hands on SME, I think we can take the tack of arguing it can't even be applied to our examples, which is basically true. Remember, SME requires *identities* of relations in order to do mapping at all. No way it gets fused=destroyed, or even laser=x-ray. Thus in a sense SME is entirely dependent on similarity of predicates, given that identity is just the extreme of high similarity. In contrast, while acme can nicely use similarity to guide mapping, a crucial point is that in the limit it can dispense with similarity altogether, and find mappings in which *nothing* is identical. Thus we can dismiss SME as beneath serious consideration as an alternative.

If we get even half of the above tests to produce reasonable results, which I'm certain of, we will have the ingredients for a whiz-bang Cog Sci article.

Geez, I just noticed how long this message got--I hope the more important ones to follow can be shorter!

Here is my idea for ARM, the Analogical Retrieval Module (an acronym for every function, I always say). I'm less sure of this than I was for acme, especially in detail, but see if you can make anything of it.

A. THE PROBLEM

Our current simple summation of activation (or more neutrally, summation of similarity) for analogical retrieval lacks sensitivity to relational structure. Given the source proposition "The dog chased the car", it will be reminded of it equally often by the target propositions "The cat chased the truck" and "The truck chased the cat". This is surely false; if we need empirical evidence, the literature on configural effects on retrieval provides it, not to mention Ziva's anecdote. Indeed, making retrieval sensitive to relational parallels is crucial for getting "deep" analogs, because these typically have only weak similarity in terms of the individual concepts. So while we do not want a retrieval module as powerful as acme, we want it to provide greater analysis of relational structure than we now have.

B. THE PROBLEM REFORMULATED

All we need, I think, is for the similarity-based retrieval process to have access to information about local variable bindings in the target and in potential sources. For our example:

SOURCE	GOOD TARGET	BAD TARGET
chase (x, y)	chase (a, b)	chase (a, b)
dog (x)	cat (a)	truck (a)
car (y)	truck (b)	cat (b)

Clearly, the sum of similarities of individual predicates is equal for GT and Source and for BT and Source. The difference lies in the preservation of local variable bindings: GT does, BT doesn't. Here is a rough algorithm that will give the right result for this simple example. Assume the similarity of "chase" to "chase" is 1, and of "dog" to "cat" and "car" to "truck" is .8. If we just added up predicate similarities, then both GT and BT would have a total similarity of 1+ .8 + .8 = 2.6 to Source.

C. THE BASIC SOLUTION

Instead, we calculate as follows:

For GT and Source:

(1) First, increment similarity counter for each of the "raw" predicate similarities (i.e., 1.0 + .8 + .8).

(2) Second, check for each argument whether it is consistent in terms of how is is activated by source. This is best done from the "top down", beginning with higher-order predicates. Here the highest-order predicate is chase (x, y). Therefore, for each argument of chase (x, y), check activation levels of all other predicates with arguments x and/or y **and the origin of the activation**. For each such predicate **activated by a target predicate with the same argument(s) as that which activated chase (x, y)**, add the predicate similarities of the two relevant predicates. I.e.,

(i) Given chase (x, y) is activated by chase (a, b), check dog (x). This is receiving .8 activation from cat (a). Since a also was first argument of the target predicate that activated chase (x, y), the criterion is met; hence increment similarity by 1.0 (for chase (x, y)) + .8 (for dog (x)).

(ii) Since there are no other predicates of x, go on to y, the second argument of chase (x, y). Checking car (y), we find it is receiving .8 of activation from truck (b). Since b also was the second argument of the predicate that activated chase (x, y), the criterion is again met; similarity is incremented by 1.0 + .8.

(3) Since there are no more predicates of y, and no more arguments of chase (x, y), we are done processing chase (x, y). Since the two lower-order predicates dog (x) and car (y) were checked in the process of checking chase (x, y), we are done altogether.

The result: total similarity of GT to Source is:

1.0 + .8 + .8 for the three active predicates, +
1.0 + .8 for the matching binding of x in chase (x, y) and dog (x), +
1.0 + .8 for the matching binding of y in chase (x, y) and car (y).
= 6.2

By contrast, this is what happens for BT and Source:

(1) First, increment similarity counter for each of the "raw" predicate similarities (i.e., 1.0 + .8 + .8).

(2i) Checking propositions of the first argument of chase (x, y), we find dog (x) is activated .8 by cat (b). Since b was not the first argument of the predicate that activated chase (x, y), we add nothing.

2ii) A check of car (y) also fails for the same reason.

(3) So we are done.

So total similarity is just

1.0 for chase (x, y) + .8 for dog (x) + .8 for car (y) = 2.6.

The general idea, then, is simply to give extra "points" for consistent local variable bindings across activated predicates.

In our previous proposal for retrieval, we assumed each predicate (concept) is represented by a set of properties (features, superordinates, predicates in rules, or whatever). This is still fine. There are just two crucial changes added in ARM:

(1) When a predicate activates its properties, and hence all things that share those properties, identifiers representing the arguments of the predicate are passed along.

(2) When a predicate in a source is activated by properties shared with a predicate in the target, the identifiers are used to perform a check for consistency of variables.

So in the above example, if the target is "The cat chased the truck", then activation sent to "chase" properties is marked with (a,b), "cat" activation is marked (a), and "truck" activation is marked (b). Then when activation is received at

the predicates of the source "The dog chased the car", the above checks for consistent variable bindings can be made.

More generally, the algorithm might be:

(1) Activation spreads via shared properties from each predicate in target to all similar predicates in potential source analogs. Each activation trace carries identifiers for the arguments of the source predicate.

(2) In a given source in memory, each predicate notes which source predicate(s), if any, are sending it activation.

(3) We collect all propositions in source with predicates that are receiving activation above some threshold from predicate(s) of target. This step serves to simplify things by dropping out all source propositions that are not getting significant activation. Thus suppose our source is "The dog chased the car" and the target is "the cat chased the boy". The source proposition car (y) would be ignored because nothing activates it significantly.

(4) In the least elegant part of algorithm (I hope), we must allow for the possibility that a single source proposition will receive significant activation from multiple target propositions. Thus suppose target were "The cat chased the wolf". Both "cat" and "wolf" might activate "dog" significantly. In order to do argument checking, we need to select one match. There are two obvious possibilities:

(i) Select the most similar matching predicate, breaking near-ties randomly if necessary; or

(ii) Perform relation checking with each candidate in turn, finally using whichever yields greater consistency to increment similarity counter. In this example, "cat" would end up being preferred to "wolf" because the cat is a chaser, just like dog.

Whichever we do, we should keep all significant similarities between predicates around, as these will be passed to ACME to guide mapping. Note that even if ARM errs by matching "dog" with "wolf", ACME should recover if it is given a chance. In calculating activation levels and similarities in ARM, however, each source predicate is allowed to be activated by just one source predicate (at a time at least).

(5) All of the above is preliminary to the process illustrated in the above numerical example. We begin by incrementing counter by the activation level of each of the significantly active source predicates.

(6) Then we do argument checking. This is best done hierarchically, beginning with highest-order predicates.

```
For each predicate in source active above some threshold

    For each argument of predicate

        For each other active predicate of that argument

            If activation came from source predicate with
               consistent argument

            Then increment counter by activations of the
```

two consistently-activated predicates

And mark that predicate pair as "checked" for that argument

(7) When all predicate pairs have been checked for consistency of all arguments, the counter gives total similarity score. If the score (transformed to reflect Tversky, as specified below) is above a threshold, ARM calls ACME and reports (a) the entire representation of the source, and (b) a list from step (3) above of all similar source-target predicate pairs, with degree of similarity. This list is used to set ACME's similarity weights from the special unit to predicate mappings. All mapping units involving pairs not on list of "similars" are initialized with some uniform low positive weight from the su. Thus ARM computes significant predicate similarities, and ACME doesn't have to do it again. This preserves the essence of PI.1's "activation tracing" to start mapping.

D. EXTENSION TO TVERSKY

As a useful refinement, let me suggest a way to extend the above to compute a Tversky-type similarity measure. The similarity of a target to a source should be an increasing function of the degree to which the source is matched by the target, and a decreasing function of the degree to which the source is not matched, and a decreasing function of the degree to which the target is not matched. What is calculated above is actually the degree to which target matches source. I have been calling this "total similarity", but it is more properly called "total overlap" between source and target, the first of the three contributors to similarity. To properly calculate overall similarity, observe that the overlap measure has a definite maximum value (obtained if every predicate was fully activated with consistent variable assignments; i.e., if target were actually identical to source).

The maximum would be 7.0 for the above example. By induction, I believe the maximum overlap score for any set of propositions is:

1 point for each individual proposition being active, +
2 points for each pair of shared arguments that are activated consistently.

This maximal value can be calculated for any representation (both source and target). Call maximum for source Smax, maximum for target Tmax. Let O be the "overlap score" calculated by above algorithm. Then overall Tversky similarity is:

a (O) - b (Smax - O) - c (Tmax - O),

where a, b, and c are free parameters representing the relative weights of Overlap, unique aspects of Source, and unique aspects of Target, respectively. Given that the implicit task is to evaluate how similar the Target is to various possible Sources, it will be most reasonable if a > b > c.

For the time being, however, I'll neglect this extension, and continue to refer to the "overlap score" as "similarity".

E. HIGHER-ORDER RELATIONS

In the simple examples so far, all predicates have constants as arguments. We will want to extend the basic algorithm to deal with higher-order relations of the Gentnerian variety that take propositions (or variables, I suppose) as arguments, e.g.,

"The dog's chasing the car caused the man to curse the dog"

cause (chase (dog, car), curse (man, dog)))

In our problem representations, it seems our slots function like higher-order relations that embed propositions. Thus in the ray problem we have:

Goal: destroyed (tumor)

which might be written as

Goal (destroyed (tumor))

In the following I will skate on thin notational ice; I hope you can get the basic idea and invent more approprate predicate calculus if necessary. I'll assume we have these kinds of logical things:

propositions
predicates
constants (and variables? I'll ignore the latter)

Here is a simple case:

Source	Target
goal (P30)	goal (P1)
P30 = fused (x)	P1 = destroyed (a)
filament (x)	tumor (a)

Let's suppose we have predicate similarities: goal/goal = 1.0
fused/destroyed = .2
filament/tumor = 0

When the target predicates send out activation, "goal" will be marked (P1), "destroyed" will be marked (a). The computed similarity to source (assuming a threshold below .2) will be:

(1) 1.0 for goal (P30) + .2 for fused (x).

(2) Checking consistency, starting with goal (P30):
(i) Note P30 is equivalent to fused (x)
(ii) Checking fused (x), it is activated .2 by destroyed (a). Since destroyed (a) is P1, which activated goal (P30), this is consistent. Add 1.0 + .2.

(3) Filament (x) is ignored because it is not active.
Accordingly, all arguments of active predicates have been checked and we are done. Total overlap score is:

1.0 for goal (P30) + .2 for fused (x) +
1.0 + .2 because the same proposition, destroyed (a), led to activation of both goal (P30) and fused (x) =
2.4.

Note that if fused (x) had not been in goal of source, but rather in initial state, similarity would only have been 1.2.

In this example, Smax and Tmax are both 7.0. For source, this is because:

1.0 for each of the three individual propositions = 3.0, +
1.0 + 1.0 because P30 and fused (x) are activated consistently, +
1.0 + 1.0 because fused (x) and filament (x) could be activated consistently =
7.0.

Accordingly, Tversky similarity is

a (O) - b (Smax - O) - c (Tmax - O) =

a (2.4) - b (7.0 - 2.4) - c (7.0 - 2.4)

Here is a more complex example of the Gentnerian sort:

Source: The dog's chasing the car caused the man to curse the dog.
Target: The cat's chasing the shadow caused the girl to laugh.

Source	Target
cause (P11, P12)	cause (P1, P2)
P11 = chase (x, y)	P1 = chase (a, b)
P12 = curse (z, x)	P2 = laugh (c)
dog (x)	cat (a)
car (y)	shadow (b)
man (z)	girl (c)

Let predicate similarities be:

identities = 1.0
curse/laugh = .3
dog/cat = .8
car/shadow = 0
man/girl = .5

Here goes calculation:

(1) car (x) is ignored because it is not sufficiently active.

(2) The counter is incremented by 1.0 for cause (P11, P12) +
1.0 for chase (x, y) +
.3 for curse (z, x) +
.8 for dog (x) +
.5 for man (z)

(3) We start checking argument consistency, beginning with the highest-order proposition, goal (P11, P12), which is activated by goal (P1, P2) in target.

(i) Process P11. It is equal to chase (x, y). This is activated 1.0 by chase (a, b). Since this is P1, linked to P11, the result is consistent. Increment counter by 1.0 + 1.0.

(ii) Process chase (x, y). Start with x. Find dog (x). This is activated .8 by cat (a). This is consistent, so increment by 1.0 + .8.

(iii) Continue with x. Find curse (z, x). This is activated .3 by laugh (c). Since c doesn't equal a, this is inconsistent. Add nothing. No more propositions with x.

(iv) Process y in chase (x, y). Since car (y) is not active, there is nothing to check. This completes processing of P11.

(v) Now pop up and consider P12. This is equivalent to curse (z, x). This is activated .3 by laugh (c). Since this is P2, which is linked to P12 by being second argument of goal proposition, this is consistent. Increment by 1.0 + .3. [NB: I'm assuming at the level of propositions, as opposed to predicates, the number of predicate arguments doesn't neccesarily have to be equal.]

(vi) Process curse (z, x). This is linked to laugh (c). Since the number of arguments mismatch, it is impossible for the arguments to be consistent; hence the processing of curse (z, x) can cease immediately at this point. This completes processing of P12, and hence of goal (P11, P12).

(4) The only source proposition not yet marked as checked is man (z). There is no other proposition of z left to compare it with, so we are done.

The result:

3.6 total for the 5 similar predicates, +
1.0 + 1.0 because chase (x, y) was activated by appropriate causer in "cause" proposition, +
1.0 + .8 because dog (x) was activated by the chaser in target, +
1.0 + .3 because curse (z, x) was activated by an appropriate outcome in "cause" proposition =
8.7.

To check mutual understanding of algorithm: I calculate Smax in this case to be:

6 for raw similarity, +
12 for argument agreement =
18.

Well--can it (or should it) be done? --Keith

Paul:

Well here's the scoop on transfer betwixt farmer's dilemma and missionaries & cannibals. It looks hard, I tell you. If anything brings arrogant acme to its knees, this will be it. Part of the problem is surely problem representation, which I hope you can work out. Let me start by giving you the relevant details from the paper this is taken from so you know what we're modeling.

METHOD

Subjects were 3rd and 4th grade kids. In acquisition each group is given isomorphs of either FD or MC; then they transfer to new isomorphs of FD or MC. (Another group did Tower of Hanoi during acquisition; wierdly, they also transfer to MC, although I can't see any analogy there at all. I'll ignore that condition.) A Control group does transfer but has no acquisition phase. There are actually three versions of each isomorph. During acquisition kids alternate between two versions (allowing schema formation); the third version is used at transfer.

Acquisition. Kids hear problem as a little story, which includes a 7-move (the minimum number) solution. They then repeatedly try to recall the solution steps in order. This continues until they make 2 consecutive error-free trials. This usually took no more than 5 trials. Thus subjects never really solved problems during acquisition, but rather memorized solution.

Transfer. Now they really have to solve a transfer problem. When child made an error, experimenter corrected it: said what constraint was violated and returned the object to where it was prior to error. Importantly: at transfer any legal 7-move sequence was counted as correct. There are only 2 possible solution paths for FD. For MC I think there are 48 if every missionary and cannibal is considered individually. This is the key, I think, as to why FD can transfer to MC (there are many possible mappings, any of which work), but not vice versa.

RESULTS

	Transfer	
Acquisition	FD	MC
FD	7.08 (43)	9.83 (136)
MC	18.17 (349)	7.58 (55)
Control	25.58 (529)	19.33 (411)

The first number is moves to correct (minimum is 7); the number in parentheses is solution time in seconds. Thus there is excellent transfer between isomorphs (FD to FD and MC to MC), very good transfer from FD to MC, and no transfer from MC to FD. (Remember that isomorphs have different cover stories than used during acquisition.)

MATERIALS

Farmer's Dilemma. Here is a precis of the "Fox, Goose, and Corn" version:

A man had a fox, a goose, and some corn in a valley. He wanted to take them to his house on other side of mountain. He had wagon, which would carry man plus one thing. If fox is alone with goose, it eats goose; if goose is alone with corn, it eats corn. How to get fox, goose, and corn across mountain to house without anything being eaten?

The learned solution is:

Trip 1: man, goose
Trip 2: man
Trip 3: man, fox
Trip 4: man, goose
Trip 5: man, corn
Trip 6: man
Trip 7: man, goose

There is only one other 7-move solution, which differs only as follows:

Trip 3: man, corn
Trip 5: man, fox

The three isomorphs are identical except for objects:

Version 1: fox, goose, corn
Version 2: lion, pony, oats
Version 3: wolf, rabbit, carrots

Missionaries & Cannibals

Precis:

Three mssionaries and two cannibals were on one side of a river. They all wanted to get to other side. They have a boat that takes a rower and one passenger. If cannibals ever outnumber missionaries on either side, cannibals eat missionaries. How to get everyone to other side of river without anyone being eaten?

The learned solution is:

Trip 1: M, C
Trip 2: M
Trip 3: M, M
Trip 4: M
Trip 5: M, M
Trip 6: M
Trip 7: M, C

All legal 7-move solutions have the same basic form. I think there are 48 versions that differ only in terms of which particular character(s) is involved at each step.

NB: Note there are just 2 cannibals. This is easier than usual version with 3 cannibals (because subjects were kids).

The three isomorphs are:

2 cannibals, 3 missionaries
2 orcs, 3 hobbits
2 Indians, 3 pilgrims

WHAT TO DO

I suggest trying to get transfer between 2 isomorphs of each type, and between one of each (FD to MC or vice versa). I assume transfer between isomorphs should be doable, assuming you can figure out how to represent problems. (I see there are nasty things about variables, specificity of objects, and so on.) Getting transfer from FD to MC will be the trickiest thing. I don't know of any analogy program that ever handled such an imperfect mapping, do you?

I made a quick stab at converting problems to predicate calculus, but I doubt what I did will be anything you'll find useful, so I'll skip it. Good luck!