Analog Retrieval by Constraint Satisfaction

Paul Thagard
Cognitive Science Laboratory, Princeton University, 221 Nassau St., Princeton, NJ 08542, USA

Keith J. Holyoak
Psychology Department, University of California, Los Angeles, CA 90024, USA

Greg Nelson
Cognitive Science Laboratory, Princeton University, 221 Nassau St., Princeton, NJ 08542, USA

David Gochfeld
Oberlin College, Oberlin, OH 44074, USA

ABSTRACT

We describe a computational model of how analogs are retrieved from memory using simultaneous satisfaction of a set of semantic, structural, and pragmatic constraints. The model is based on psychological evidence suggesting that human memory retrieval tends to favor analogs that have several kinds of correspondences with the structure that prompts retrieval: semantic similarity, isomorphism, and pragmatic relevance. We describe ARCS, a program that demonstrates how these constraints can be used to select relevant analogs by forming a network of hypotheses and attempting to satisfy the constraints simultaneously. ARCS has been tested on several data bases that display both its psychological plausibility and computational power.

1. Introduction

Analogy is ubiquitous in human thinking, in diverse areas that range from practical problem solving to scientific explanation to literary embellishment. This paper addresses one of the central questions related to understanding analogy: How can relevant analogs be retrieved from memory for potential use? Numerous computational models have been proposed for analog retrieval (sometimes also called *access* or *recognition*) using mechanisms such as similari-

Artificial Intelligence **46** (1990) 259–310
0004-3702/90/$03.50

ty matching, causal indexing, and spreading activation. We see merit in all of these proposals, but contend that each has only partially captured how relevant analogs can be efficiently retrieved from memory. We propose instead a theory according to which analogs are retrieved from memory using a combination of semantic (meaning-related), structural (configural), and pragmatic (goal-related) constraints. Simultaneous satisfaction of multiple constraints can be naturally implemented using connectionist networks: ARCS (analog retrieval by constraint satisfaction) is a program that constructs and uses a localist connectionist network to retrieve relevant analogs. ARCS has been tested on several data bases designed to test it as a psychological model and to evaluate its computational power.

We begin this paper with a review of the uses of analogy and a description of the role of retrieval in these uses, and then present a constraint-satisfaction theory of retrieval and describe its implementation in ARCS. We compare ARCS with other computational models of analogical retrieval and describe its applications. Our conclusion is that analog retrieval can plausibly be viewed as a process of parallel satisfaction of multiple constraints.

1.1. The ubiquity of analogy

A theory of analog retrieval should be broad enough to cover the rich diversity of kinds of analogical thinking. In cognitive science, most attention has been paid to analogical problem solving, in which a stored *source* analog is used to suggest a solution to a posed *target* problem. Most psychological experiments have concerned *cross-domain* analogical problem solving, in which the source and target derive from different domains (e.g. [18, 21, 22, 33]). In contrast, much AI research has been restricted to the use of analogies within a single domain, often under the heading of *case-based* problem solving and planning [26, 39].

But problem solving is not the only purpose of analogy. For example, analogies are often used in explanations, when we use a source analog to provide understanding of a target phenomenon [62]. Sometimes the source generates understanding without much modification, but in other cases the source is used to form new explanatory hypotheses, a process that Thagard [60] calls *analogical abduction*. Analogies can be used, not only to form hypotheses, but also to help evaluate them [61]. Analogies are often used in political, historical, and legal arguments, functioning to convince someone that a particular conclusion is warranted. For example, arguments that Nicaragua is in danger of becoming another Cuba are used to support US intervention, while arguments that it is in danger of becoming another Viet Nam are used to support a hands-off policy. Finally, literary analogies can have an evocative function, calling forth relevant emotional responses to past events or situations with established emotional content.

1.2. The problem of retrieval

In common with many theorists, we decompose analogy into:

(1) retrieval or selection of a plausibly useful source analog,
(2) mapping between the source and the target,
(3) transfer of information in the source for appropriate use with the target,
(4) subsequent learning.

Sometimes an analog is directly given, as when a student is told to think of a chemical bond as a kind of tug-of-war [62]. In such cases the use of analogy primarily requires *mapping* one analog onto the other. Our direct concern in this paper is only with (1), retrieval of an analog in the absence of any external guidance as to what information in memory is relevant. We have previously proposed ACME, a constraint-satisfaction model of analogical mapping [35]. Because there is evidence that the constraints on analogical access overlap in important ways with the constraints on analogical mapping, the ARCS model of retrieval proposed in the present paper shares many features with the ACME model of mapping, as will be discussed below. The output of the ARCS retrieval process can in fact serve as input to ACME simulations.

Computationally, retrieval of analogs is a difficult problem. A computational system must have at least the following capabilities:

(1) It must efficiently find relevant analogs. In a small data base, this can be trivial, since exhaustive search can be applied. But in a large data base, exhaustive search will be too slow.
(2) The system must be able to screen out analogs that are less relevant, so that it does not get swamped by retrieving unusable numbers of cases. Some sort of comparative mechanism must ensure that the most relevant analogs are selected over others.
(3) If the system is intended to be a psychological model, the analogs it retrieves and the processes by which it does so should correspond to human performance as exhibited in controlled psychological experiments. Our project is to model human cognition, but we acknowledge the possibility of AI models aimed more at engineering adequacy than cognitive modeling. At least in the present state of AI, cognitive and engineering aims seem highly convergent, as a system that successfully emulated human analog retrieval would clearly surpass the performance of current AI models of analogy.

To put it simply, the problem of analog retrieval is how to get what you want from memory without getting more than you can use. Indeed, this is the core problem of memory retrieval in general. The present proposal by no means constitutes a complete model of human memory; many crucial aspects of the human memory system, such as the effects of processing limitations in working

memory, have been set aside. Nonetheless, the present model shares some basic general features with most current conceptions of human long-term memory. In particular, the model assumes an organized store of concepts to which representations of particular episodes are linked. Representations of episodes are viewed as decomposable into more elementary elements, which provide multiple potential retrieval cues. The memory system is content addressable, in that any element of a stored representation can potentially serve as a basis for access to the entire representation. Retrieval is an essentially parallel process that involves comparison of the target analog to representations of potential source analogs in memory.

As we will see, however, our model of analog retrieval goes beyond current theories of general memory retrieval in its sensitivity to configural information concerning the relational structure of the analogs. The distinctive characteristic of the retrieval of analogs, as opposed to simple associative retrieval, is that the basis for retrieval is not simply shared elements, but also comparable *patterns* of elements. The nature of these relational patterns will be described below. The present model provides a mechanism for integrating Gestalt-like sensitivity to relational patterns into an associative memory.

Whereas the process of analog retrieval is viewed as involving parallel comparisons to multiple representations in memory, mapping and transfer are apparently more restricted, in that people generally use one analog at a time when engaged in the mapping and transfer stages of problem solving, explanation, and argument. It is therefore important that the retrieval process be designed so as to tend to select only the most promising potential analogs as candidates for more detailed subsequent analyses.

2. A Constraint-Satisfaction Theory

In our paper on mapping [35], we identified three major kinds of constraints that have been proposed to govern how parts of two analogs can be placed in correspondence with each other: semantic similarity, isomorphism, and pragmatic centrality. These constraints were treated not as absolute requirements on successful mappings, but rather as *pressures* that operate to some degree [29]. We argued that all three types of constraints are involved in analogical mapping. Here we will briefly review the distinctions among these three classes of constraints and the psychological evidence regarding their impact on the retrieval of analogs.

2.1. Semantic similarity

Two analogs are semantically similar to the extent that the predicates used in the representations of the two analogs are semantically similar. We use the term "semantics" here in the lexical sense [10], not in the formal, model-

theoretic sense. Two predicates are semantically similar if they are identical or if they participate in lexical relations such as synonymy, hyponymy (representing things of the same kind) and meronymy (representing things similar with respect to part-whole relations). Section 3.1 describes our use of semantics modeled after WordNet, an electronic lexical reference system based on psycholinguistic theories of the organization of human lexical memory [43]. For example, "dog" is similar to "cat," because dogs and cats are both kinds of animals and kinds of pets; and "hand" is similar to "foot" because they are both parts of the body.

Numerous psychological experiments indicate that retrieval of analogs by humans is very sensitive to the degree of semantic overlap between the target analog that provides retrieval cues and the source analog to be found in memory [19, 21–23, 33, 48, 50–52]. Such overlap depends on the semantic similarity of the concepts used to represent the analogs. For example, Holyoak and Koh [33] found that subjects given a problem of finding a way to use an X-ray to destroy a tumor were much more likely to retrieve an analogous problem concerning using a laser to fuse a filament in a light bulb than they were to retrieve an analogous problem concerning using ultrasound to fuse a filament. This difference appeared to reflect the fact that X-ray devices are more like lasers than like ultrasound devices. Other similar studies using analogs that lacked *any* apparent similar elements (e.g., a source analog involving use of an army to capture a fortress) have found that people are often unable to retrieve dissimilar analogs, even though such analogs could readily be used to aid in problem solving once the person was reminded of their relevance by the experimenter [8, 21, 22, 58].

It is important to distinguish semantic similarity, as we use the term in our models of mapping and retrieval, from conceptions of "surface," "literal" or "superficial" similarity that have been employed in the analogy literature (e.g. [33, 51]). Surface similarity has been characterized as similarity based on one-place predicates, or attributes [16]; another sense of surface similarity involves similarity based on features of the analogs unrelated to goal achievement [33]. In contrast, semantic similarity is simply overlap in meaning. Concepts can be semantically similar regardless of whether they involve attributes (e.g., "dog" and "cat"), or relations of any level of internal complexity (e.g., "push" and "shove," or "know" and "understand"). Semantic similarity also has no necessary relationship to goal relevance, which we treat as a separate constraint of pragmatic centrality.

We view the role of semantic similarity in analog retrieval as simply a special case of its dominant role in general memory retrieval processes in humans. Because semantic links typically provide fundamental retrieval pathways, we would expect that positive similarity between at least one pair of elements in a target and source analog will be a necessary (but not sufficient) precondition for retrieval. We know of no cases of human analog retrieval in which the

analogs had no semantic similarity. In contrast there are special cases of mapping between pairs of analogs that depend only on isomorphism, for example mapping abstract relations of number addition with abstract relations of set union [35]. The problem of mapping two given analogs is different from the problem of selecting an analog for a single given target from a large memory that contains numerous potential analogs. Our model treats semantic similarity as the dominant constraint on retrieval, but not the only one.

2.2. Isomorphism

The overwhelming evidence that semantic similarity plays a major role in analog retrieval has contributed to the relative neglect of configural effects on reminding. We contend that an analog is more likely to be retrieved the greater the degree of isomorphism it has with the structure that initiates the retrieval. Informally, isomorphism is a matter of having the same configuration. A more exact notion can be developed by assuming that structures can be represented by propositions consisting of predicates and arguments. Then two structures are isomorphic if there is a one-to-one correspondence between them that preserves structural consistency, where structural consistency requires that if two propositions are mapped, then their constituent predicates and arguments should also map [12, 13, 16]. For example, let $F(a, b)$ be a sample proposition of one structure asserting that the relation F holds between a and b, and let $G(c, d)$ be a sample proposition of another structure. A condition on the two structures being isomorphic is that if these two propositions correspond to each other, then F corresponds to G, a corresponds to c, and b corresponds to d. Isomorphism is conceptually distinct from semantic similarity, in that two structures can be perfectly isomorphic even though they share no identical or similar elements [35, 45]. Two structures can fail to be isomorphic because the best correspondence between them is not one-to-one or does not preserve relational structure. Mathematically, isomorphism is an all-or-none matter, but we speak of degree of isomorphism informally as the extent to which the best correspondence between two structures is one-to-one and structurally consistent. Like Falkenhainer et al. [13], our notion of structural consistency includes higher-order relations such as "cause" that can take propositions as arguments. For example, if a proposition *cause*(p, q) stating that p causes q is to be placed in correspondence with *cause*(r, s), then the predicates and arguments of p and q should correspond, respectively, to the predicates and arguments of r and s.

Sensitivity to isomorphism has been a crucial component of virtually all AI models of analogical mapping (e.g. [6, 12, 13, 35, 39, 66]). It is clear that people are indeed sensitive to isomorphsim of analogs in mapping. For example, Gentner and Toupin [20] found that children had difficulty mapping two situations if similar characters played different roles in the two analogs. Such cross-mapping leads to conflicting interpretations. Studies by Holyoak

and Koh [33] and Ross [51, 52] have also provided evidence that structural consistency affects ease of mapping.

But what about the retrieval stage? Theorists have often failed to clearly address the question of whether retrieval is sensitive to isomorphism. Consider the following mini-story:

The dog bit the boy and the boy ran away from the dog.

We conjecture that this story will be more likely to remind a person of past stories of dogs biting boys who run away than of past stories of boys biting dogs or of dogs running away. The structural issues become clearer when such stories are represented in predicate calculus:

Analog 1:
dog (Fido),
boy (John),
bite (Fido, John),
run-away-from (John, Fido).

Analog 2:
dog (Rover),
boy (Fred),
bite (Rover, Fred),
run-away-from (Fred, Rover).

Analog 3:
dog (Rover),
boy (Fred),
bite (Fred, Rover),
run-away-from (Rover, Fred).

Analog 4:
dog (Rover),
boy (Fred),
bite (Rover, Fred),
run-away-from (Rover, Fred).

Notice that analogs 1 and 2 are isomorphic, and we conjecture that this will make analog 2 more easily retrieved given analog 1 as a cue than are either analogs 3 or 4, which are just as semantically similar but can be made isomorphic to 1 only by cross-mapping dog to boy and boy to dog.

There is some evidence from studies of analogy that retrieval is indeed sensitive to isomorphism. Experiments reported by Gentner [17, 19] primarily showed the effects of semantic similarity on retrieval, but they found that higher-order relational commonalities also promote access. Holyoak and Koh [33] found evidence that *both* structural consistency and semantic similarity

affect the probability that a source analog will be spontaneously retrieved and used. The materials they used were various versions of the lightbulb and tumor analogs discussed earlier. One variation in the source lightbulb stories involved the reason a single large force could not be used to fuse the filament: either because the force would break the delicate glass surrounding the filament, or because no single large force was available. In both versions the successful solution was to use multiple converging weak forces. The former "similar constraint" version was more structurally consistent with the tumor problem, in which a single strong laser could not be used because it would damage healthy tissue surrounding the tumor. Holyoak and Koh found that subjects were more likely to spontaneously apply the similar-constraint version of the source analog. (The term "constraint" here refers to the specific constraint on the problem solution in the experimental materials, and should not be confused with the general analogy constraint of isomorphism.)

Ross [52] performed a series of experiments in which subjects had to apply probability principles that were initially illustrated by a single concrete word problem. Subjects were then asked to solve transfer word problems. Ross varied whether the overall cover stories of the source and target problems were highly similar (e.g., two problems involving the IBM motor pool) or dissimilar (e.g., a motor-pool problem and a nursery-school problem). This manipulation of semantic similarity of the analogs was crossed with a variation in degree of structural consistency. In the consistent conditions, similar types of entities mapped onto corresponding variables in the relevant equations (e.g., people mapped to people, artifacts to artifacts), whereas in the inconsistent conditions the required mappings were crossed (people to artifacts and artifacts to people). Ross found that inconsistent mappings impaired access to the source, but only when the overall cover story was similar. Thus relatively high semantic similarity was a necessary condition for obtaining an effect of structural consistency. As we will see, this type of interaction between semantic and structural constraints on retrieval is predicted by the ARCS model.

Note that structural consistency, like semantic similarity, must be distinguished from other similar terms that have been used in the analogy literature. In particular, Holyoak and Koh [33, 32] and Ross [51], following the usage in the problem-solving literature, used the term "structural" features to refer to those that are relevant to the goals of a problem. In contrast, structural consistency is a purely formal constraint on the *relations* between structures; we use the term in the same sense as Falkenhainer et al. [12]. The goal-related sense of "structural" corresponds to our third type of constraint, pragmatic centrality.

2.3. Pragmatic centrality

As we suggested in Section 1.1, analogies have various purposes. The purpose of an analogy in problem solving is to help accomplish the goals of the

problem. Clearly, a retrieval system attuned to increase the retrieval of analogs relevant to goal accomplishment would contribute more to problem-solving effectiveness than a retrieval system that lacked sensitivity to goals. We therefore postulate that one of the constraints on analog retrieval is pragmatic centrality: stored structures that are potentially important to a system's goals are more likely to be retrieved than irrelevant ones. Numerous AI theorists have argued that causal relevance to goal accomplishment should influence retrieval [6, 7, 16, 39, 55, 66, 67]. Many of these proposals make the claim that causal indexing is the main way in which analogs are stored and retrieved. A major hypothesis is that failures of goal achievement are especially likely to be indexed, so they can play a role in avoiding similar future failures.

Psychologists have only begun to establish experimentally that people's analogical access is sensitive to pragmatic constraints. Part of the problem is that it is difficult to distinguish effects of goals that reflect a special pragmatic constraint from effects that can be interpreted as consequences of other general constraints involving semantic similarity and structural consistency. In general, if a representation of a structure is augmented with goal information, the augmented structure will therefore be more semantically similar to, and more structurally consistent with, other structures containing similar goal information. Nevertheless, some studies have demonstrated the importance of task goals in eliciting remindings. Seifert, McKoon, Abelson and Ratcliff [56] and Faries and Reiser [14] have shown that people are sensitive to the solution-relevant aspects of previously solved problems.

Although empirical evidence is still limited, the computational argument that pragmatic constraints on retrieval are useful for narrowing the search for analogs leads us to include such constraints in the ARCS model. We view relevance to the purposes of the analogy (including explanation and argumentation, not just problem solving) as an important factor in retrieval, although not the dominant factor suggested by some AI theorists. Problems and plans have explicit purposes that should play a role in retrieval; similar pragmatic constraints could also operate in the case of stories and other structures with less specific purposes. As in the ACME mapping model, we treat pragmatic centrality as an additional pressure to semantic similarity and structural consistency. If an element of the target is relevant to the goal, the pressure of pragmatic centrality will favor retrieval of analogs that allow the important target element to be mapped.

2.4. Parallel constraint satisfaction

We propose, therefore, that retrieval of analogs from memory is determined by simultaneous satisfaction of the constraints of semantic similarity, structural consistency, and pragmatic centrality. When a target analog is presented in the form of a problem to be solved, an explanation to be given, or a conclusion to be reached, search for potentially useful source analogs in memory proceeds by

searching memory for analogs that look promising for semantic, structural, and pragmatic reasons. Search is massively parallel, so that many different analogs can be simultaneously examined for potential relevance. But to prevent the system from being swamped by too many analogs of lesser relevance, search is competitive, in the sense that the retrieval of one analog tends to suppress the retrieval of other analogs. We will now describe how these ideas can be implemented computationally. Later we will provide more detailed comparisons with other computational models that implement only some of the constraints.

3. ARCS: A Connectionist Program for Analogical Retrieval

ARCS (analog retrieval by constraint satisfaction) is a COMMON LISP program that retrieves analogs using multiple constraints. In brief, the program operates as follows. Retrieval is initiated from a probe structure and is intended to find structures in memory that are analogous to the probe. Search for analogs of the probe proceeds initially using the predicates in the probe, looking for predicates that are in some degree semantically similar to them.[1] If a predicate that is semantically similar to a predicate in the probe structure occurs in a stored structure, then there is a possibility that the stored structure is analogous to the probe structure. Many stored structures, however, are likely to have some semantic overlap with the probe structure, and the principle task of ARCS is to pick out the ones that are most relevant according to the constraints of semantic similarity, structural consistency, and pragmatic centrality. As potentially analogous structures are noticed, ARCS sets up a constraint network to compare their relevance to the probe structure. Once this constraint network has been built, ARCS uses a standard parallel connectionist relaxation algorithm to settle into a state that indicates the relative correspondence of the various stored structures to the probe structure. The memory structure underlying these processes is described in the next section.

Figure 1 provides a rough picture of how the retrieval process works. The probe structure at the bottom contains five dots corresponding to predicates used in representing the probe. Each of these predicates is linked to an associated conceptual structure, indicated by the dots outside the probe structure. Each of these concepts in turn has access to numerous other concepts that are semantically similar to it, and predicates corresponding to these concepts occur in various stored analogs S1–S5. Of these, S2 and S5 have greater semantic overlap than the others with the probe by virtue of multiple semantic links, so these two stored structures are prime candidates for analogical retrieval. However, determination of the optimal apparent analog will

[1]The term "probe" is used rather than "target" because when discussing retrieval of stored structures from memory it is confusingly natural to think of them as targets, although in our standard terminology for analogs they are potential sources for use with a target structure.

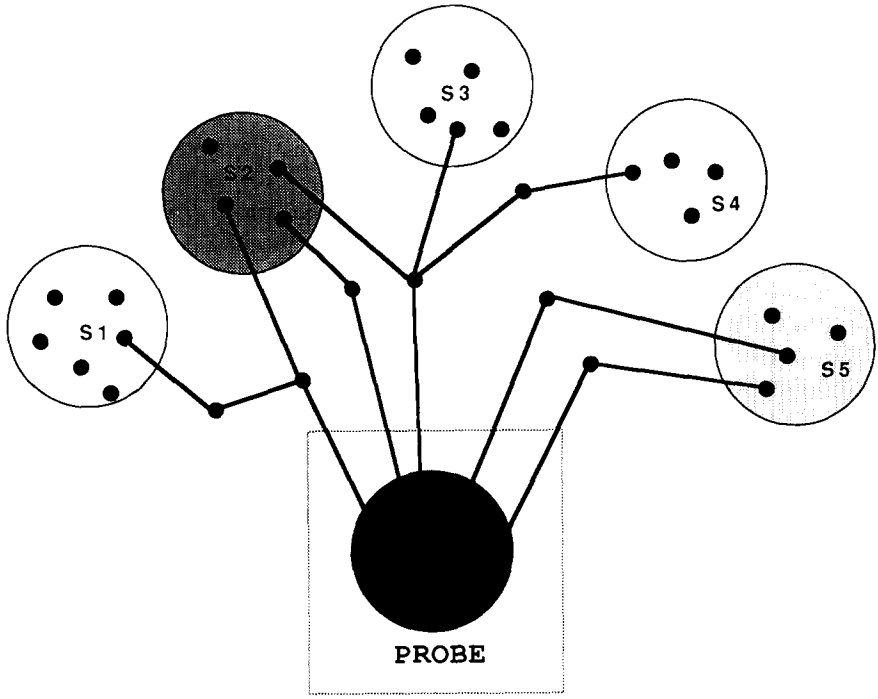

Fig. 1. Probing from a structure into long-term memory via conceptual links. The large circles represent structures stored in memory. The dots represent concepts; the lines connect semantically similar concepts.

depend on structural and pragmatic constraints in addition to the semantic links that initially suggested possible relevance. Thus retrieval has two broad steps:

(1) finding semantically associated structures in memory,
(2) assessing these initial candidate structures in terms of the full set of constraints.

Now let us examine in greater detail how the retrieval process operates.

3.1. Input to ARCS

Three kinds of knowledge representation are used in ARCS for different purposes:

(1) Structures representing probe and stored analogs consist of propositions in predicate calculus.
(2) The semantic information used in making judgments of semantic similarity is stored in frame-like structures attached to predicates. We call

these structures "concepts." We thus distinguish the predicates used to represent analogs from concepts, the more complex structures that carry semantic information. In our implementation, predicates are LISP atoms, whereas concepts are LISP atoms with their property lists. The concepts are organized into hierarchies by virtue of kind-relations and part-relations.

(3) The constraint network used to select the most relevant analog or analogs consists of units representing correspondences between structures and parts of structures.

The first two kinds of representation are used for long-term memory, whereas the third is created during a retrieval task. The first corresponds roughly to what psychologists call *episodic* memory, while the second corresponds roughly to what they call *semantic* memory [64]. The constraint network itself can be viewed, not as a memory store, but as a temporary computational representation of the pattern of correspondences between a retrieval cue and information in long-term memory. We shall now describe each kind of representation in greater detail.

Figure 2 displays a sample analog structure, representing the problem of using an X-ray to destroy a tumor, to be discussed in Section 5.1 below. The structure is divided into two fields, representing the starting conditions and the goals of the problems. If the problem were solved, it would have a third field describing its solution. Other kinds of structures have different fields; for example, the fables discussed below consist of a story and a moral, while the plays consist of a list of characters and a story. Although fields are used in the mapping program ACME to restrict possible mappings, ARCS does not disqualify any source on the basis of field organization. ARCS does, however, use fields in its implementation of the pragmatic constraint described in the next section.

Each field consists of a set of propositions of the form:

(predicate arguments truth-value proposition-name)

Incorporating the truth value into the proposition makes it unnecessary to use a predicate NOT. For predicates like IF and CAUSE, the arguments can be propositions, as in the following proposition S1-3, which states that proposition S1-1 refers to an event that causes the event referred to by proposition S1-2:

(cause (S1-1 S1-2) true S1-3)

Representation of if-then statements is a bit more complicated, since we need to be able to say, for example, that if S1-1 is true then S1-2 is false:

(if ((S1-1 true) (S1-2 false)) true S1-3)

The extra truth value embedded in IF statements is necessary to allow the expression of counterfactual conditionals, such as "If Dukakis had won the

Starting conditions:
(doctor (obj-doctor) true tp-1)
(patient (obj-patient) true tp-2)
(tumor (obj-tumor) true tp-3)
(malignant (obj-tumor) true tp-4)
(have (obj-patient obj-tumor) true tp-5)
(treat (obj-doctor obj-tumor) unknown tp-6)
(must (obj-doctor (tp-6 true)) true tp-7)
(die (obj-patient) unknown tp-8)
(if ((tp-7 false) (tp-8 true)) true tp-9)
(desire (obj-doctor (tp-8 false)) true tp-10)
(remove (obj-tumor) false tp-11)
(cannot (tp-11 true) true tp-12)
(ray (obj-ray) true tp-13)
(ray-source (obj-ray-source) true tp-39)
(produce (obj-ray-source obj-ray) true tp-40)
(radiation (obj-ray) true tp-14)
(dosage (obj-dosage obj-ray) true tp-15)
(high (obj-dosage) unknown tp-16)
(if ((tp-16 true) (tp-18 true)) true tp-17)
(destroy (obj-ray obj-tumor) unknown tp-18)
(tissue (obj-tissue) true tp-19)
(healthy (obj-tissue) true tp-20)
(surround (obj-tissue obj-tumor) true tp-21)
(destroy (obj-ray obj-tissue) unknown tp-22)
(if ((tp-16 true) (tp-22 true)) true tp-23)
(if ((tp-22 true) (tp-8 true)) true tp-24)
(low (obj-dosage) unknown tp-25)
(if ((tp-25 true) (tp-18 false)) true tp-26)
(if ((tp-18 false) (tp-8 true)) true tp-27)

Goals:
(become-true (tp-18) true tp-36)
(survive (obj-patient) true tp-37)
(become-false (tp-22) true tp-38)

Fig. 2. Predicate-calculus representation of the radiation problem.

election, then taxes would have been raised," where the antecedent is false. Moreover, in fables and plays it is often necessary to represent *propositional attitudes* of characters, for example that Hamlet believes proposition S2-1 or that Macbeth orders Lady Macbeth to do an act represented by proposition S3-1:

(believes (Hamlet S2-1) true S2-2)
(orders (Macbeth Lady-Macbeth S3-1) true S3-2)

ARCS' data bases now include, respectively, 5 radiation problems, 5 Karla the hawk stories, 100 fables, and synopses of 25 plays.

In order to find concepts that are semantically similar to the concepts in a probe structure, we need to encode semantic information. ARCS' semantic structures are modeled after WordNet, an electronic lexical reference system based on psycholinguistic theories of the organization of human lexical memory [43, 44]. In WordNet, a concept is represented by a set of synonyms, and synonym sets are organized by means of kind, part–whole, and antonymy relations. Kind and part–whole relations are fundamental to the organization of the lexicon because they generate hierarchies. For example, a whale is a kind of cetacean, which is a kind of mammal, which is a kind of animal, which is a kind of living thing; a toe is part of a foot, which is part of a leg, which is part of a body. WordNet now includes more than 60,000 entries, including verbs and adjectives as well as nouns. One advantage of working on such a large scale is that the differences between the kinds of lexical items become readily apparent. Kind and part–whole hierarchies apply well to nouns, but adjectives are primarily organized into antonymic clusters, such as that posed by the extremes *wet–dry*. According to the WordNet researchers, verbs have entailment hierarchies rather than part–whole hierarchies, and their kind hierarchies seem to differ from the kind hierarchies of nouns in ways that are still under investigation. The input to ARCS includes semantic information for each predicate in each structure. ARCS' semantics now includes entries for more than 1200 words.

A major advantage of using WordNet is that it allows us to establish semantic structures in relative independence from the particular analogs we wish the program to retrieve, thus providing a stronger test of the model. ARCS uses the same semantic information for all its data bases. Approximately two thirds of the entries derive directly from WordNet. The WordNet entries themselves were generated from many different sources, especially dictionaries and thesauruses. Our own supplemental entries derive largely from *Roget's International Thesaurus* (4th ed.), and *Webster's Ninth New Collegiate Dictionary*.

Here are some samples of our lexical information, for a noun, an adjective, and a verb.

ANIMAL
SUPERORDINATES: organism living-thing
SUBORDINATES: prey person child mammal primate reptile amphibian fish bird insect vertebrate invertebrate game

PARTS: voice tooth tail claw-of-claw antler
PLURAL: animals
SYNONYMS: beast creature fauna
ANTONYMS: plant flora

AFRAID
SYNONYMS: afraid-of fearful dreading alarmed frightened apprehensive anxious uneasy apprehensive scared terrified worried
ANTONYMS: unafraid confident secure

ABDICATE
TENSES: abdicates abdicated abdicating
SYNONYMS: resign vacate relinquish cede renounce

Because a mammal is a kind of animal, the predicate ANIMAL has MAMMAL as a subordinate and MAMMAL has ANIMAL as a superordinate. (In our terminology, if A is a kind of B, then A is a subordinate of B, and B is a superordinate of A.) WordNet is still under development, and the 1988 version we used did not include verb subordinates and superordinates that have been added to later versions. Moreover, the lists of synonyms, superordinates, and subordinates have been pruned considerably in recent versions. Ideally, the lists of superordinates and subordinates should include only immediate links, so that ANIMAL would be directly linked only to MAMMAL and not to PERSON. A full theory of semantic decomposition might make it possible to derive synonyms, rather than taking them as given, but we follow WordNet in simply listing them. We have added plurals and tenses to this data base so that structures that use variants of the same predicates can be retrieved.

The WordNet-style semantic information enables ARCS to make judgments of the semantic similarity of any two predicates based on the kinds of semantic relations between them. The greatest degree of semantic similarity holds between identical predicates, with synonymy constituting a lesser degree. Still lower degrees of similarity derive from kind relations (subordinate and superordinate) and part relations. The algorithm used by ARCS to compute semantic similarity is stated below.

3.2. Algorithms of ARCS

Given the permanent memory store consisting of analog structures and hierarchically organized concepts, ARCS executes retrieval from a probe structure in four stages:

Stage 1. Using information about the semantic similarity of predicates, the program creates a constraint network representing possible correspondences between objects, predicates, propositions, and structures. Because of their

ubiquity and context independence, the following predicates are not used as retrieval cues: CAUSE, IF, CONJOIN-OBJECT, CONJOIN-EVENT, BECOME-TRUE, BECOME-FALSE. If a stored structure that is a potential source contains one or more other predicates that are semantically similar to the predicates in the probe, then it is considered as a potential analog. Units representing correspondences are created and links between units are set up to indicate correspondences between the probe and source that support each other. The most important units are the ones that hypothesize that a source structure in memory is analogous to the probe structure. Such units receive names of the form PROBE=SOURCE. (Here "=" means "corresponds to," not identity.) If the probe is P1 and the stored source is S1, then the unit created to represent a correspondence between them will be P1=S1. If P1-1 is a proposition in P1 that corresponds to proposition S1-1 in source S1, then the unit P1-1=S1-1 that hypothesizes a correspondence between the propositions will have an excitatory link with the unit P1=S1. Moreover, units are created putting in correspondence the predicate and arguments of P1-1 with the predicate and arguments of S1-1, and these units receive excitatory links with the unit P1-1=S1-1. Excitatory links are also set up from a special semantic unit to predicate-predicate units based on the degree of semantic similarity of the predicates. The special semantic unit, like the special pragmatic unit of Stage 3, is a unit whose activation level is always kept at the maximum value of 1. Hence it serves to pump activation to all units that are linked to it.

Stage 2. Inhibitory links are constructed between units representing incompatible hypotheses, for example, between P1=S1 and P1=S2. These links make retrieval competitive, in that the retrieval of one structure will tend to suppress the retrieval of an alternative.

Stage 3. Pragmatic constraints are implemented by noting that certain elements (predicates, objects, or propositions) are IMPORTANT and that certain correspondences are PRESUMED to hold. Information about presumed correspondences is provided by the programmer, just as a teacher trying to get a student to retrieve a problem similar to a given one might explicitly provide a hint relating part of the new problem to part of the old. Excitatory links are set up from the special pragmatic unit to all units involving IMPORTANT elements, and to all units representing PRESUMED correspondences. Depending on the purpose of the analog, elements can be marked as IMPORTANT automatically. For example, if the probe structure is a problem to be solved, its fields include starting conditions and goals. Each of the predicates in the goals is marked as IMPORTANT, as is each predicate in the starting conditions occurring in a proposition that is determined to be relevant to the goals by virtue of CAUSE or IF relations or by propositional attitudes.

Stage 4. The network is run by setting the activation of all units to a minimal initial level, except for the semantic and pragmatic units for which activation is clamped at 1. Then the activation of each unit is updated by considering the

activations of those units to which it has links. Cycles of activation adjustment continue until all units have reached asymptotic activation, which typically takes fewer than 150 cycles.

Figure 3 provides a very simple illustration of how this process works, using a probe analog P1 consisting of only two propositions, P1-1 and P1-2, and two

Probe analog	Stored analogs	
P1	S1	S2
P1-1 A(a,b)	S1-1 M(m,n)	S2-1 M(n,m)
P1-2 B(b,a)	S1-2 N(n,m)	S2-2 R(n,m)

A and M are semantically similar; B and N are semantically similar.

A is important.

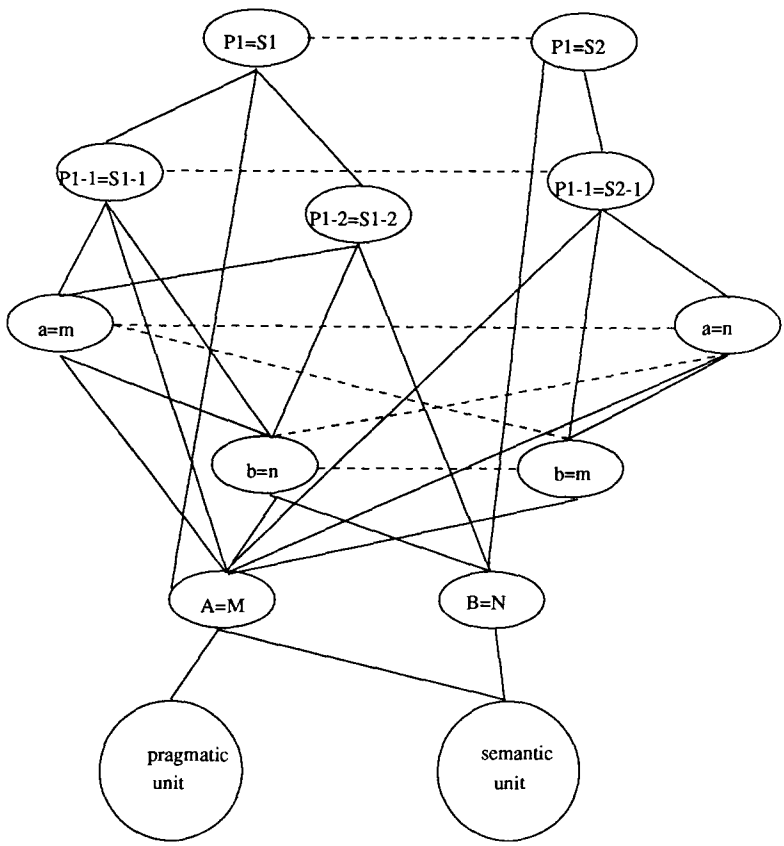

Fig. 3. An example of a network constructed by ARCS. Ellipses are units representing possible correspondences. Solid lines indicate excitatory links while dotted lines indicate inhibitory ones.

stored analogs S1 and S2 each of which consists of only two propositions. Figure 3 depicts these propositions and the constraint network that is created by probing from P. The units P1=S1 and P1=S2 indicate the possibility of retrieving both S1 and S2 from P1. These possibilities arise because the semantics for the predicates A and B in P indicated that A is semantically similar to M and B to N. A structure S3 that did not contain any predicates semantically similar to M or N would not be considered. The semantic similarity of M to N also gives rise to the possible proposition correspondence P1-1=S1-1 and the possible object correspondences $a=m$ and $b=n$. These correspondence hypotheses all support each other, as is shown by the solid lines, which indicate symmetric excitatory links between pairs of units. Also shown are the excitatory links from $A=M$ to P1=S1 and P1=S2, and from $B=N$ to P1=S1. The dotted lines indicate inhibitory links between pairs of incompatible units. Running the network consists of adjusting the activation of all the units based on their inputs from the units to which they are linked. In this case, we would expect the unit P1=S1 to become active and suppress the activation of P1=S2, because the former is supported by two proposition correspondences and indirectly by two predicate correspondences, whereas the latter is supported by only one. This example network has only 13 units and 26 links, but below we will describe ARCS networks with many hundreds of units and many thousands of links. Note that Fig. 3 displays a constraint network of hypotheses about correspondences, not a network of concepts such as A and M; it therefore differs from Fig. 1 whose nodes represent concepts.

For full generality, the algorithms used by ARCS are outlined in Figs. 4–8. Figure 4 describes the algorithm for finding the predicates semantically similar to a given predicate, and Fig. 5(B) shows how ARCS computes the degree of semantic similarity between two predicates. Identical predicates are judged to be more similar than synonyms, which in turn are more similar than predicates that have superordinate and subordinate relations. The numerical values used for the various types of relations reflect our judgment about the comparative similarity of predicates; the specific values have no empirical justification, but sensitivity analyses reported in Section 6.3 show that program behavior is not dependent on these particular values.

Figure 5 also outlines the algorithm for setting up the network using the set of search predicates determined to be semantically similar to the predicates in the probe structure. Figure 5(D) describes how units and excitatory links between them are created to implement the part of the isomorphism constraint concerned with structural consistency. A unit postulating a correspondence between a proposition in the probe and a proposition in the target is created and linked to the unit postulating a correspondence between the predicates in those propositions. Similarly, units representing correspondences between the arguments of the two propositions are created and linked to the units representing proposition and predicate correspondences. Finally, ARCS creates a

P is the probe structure that initiates the retrieval process. Get the list of search predicates **search-preds** by considering each proposition $\text{prop}_{P,i}$ in P, consisting of a predicate $\text{pred}_{P,i}$ and arguments $(\text{arg}_{P,i,1}, \ldots, \text{arg}_{P,i,n})$. There may be more than one occurrence of $\text{pred}_{P,i}$ in P, each associated with different arguments, but **search-preds** lists a predicate only once.

(A) If the predicate $\text{pred}_{P,i}$ is one of a select set of unmapped predicates (CAUSE, CONJOIN-EVENT, CONJOIN-OBJECT, IF, BECOME-TRUE, BECOME-FALSE), discard it and do not continue the search.
(B) If the predicate is a plural or an unusual tense, probe from the singular or the standard tense (first person singular) instead.
(C) Collect all of the predicate's own superordinates, subordinates, parts, part-ofs, subordinates of direct superordinates, and either synonyms (if proposition has a "true" truth value) or antonyms (if it is "false"). (This takes only immediate associates, not the transitive closure.)

Fig. 4. Algorithm for selecting semantically similar predicates from the predicates in a probe structure

unit hypothesizing that the probe structure corresponds to the source structure containing the proposition and links this unit to the units representing proposition and predicate correspondences.

One-to-one correspondences are encouraged by the construction of inhibitory links described in Fig. 6. If two units represent incompatible hypotheses about which objects, predicates, propositions, or objects correspond to each other, then an inhibitory link is created between the units. The pressure toward one-to-one correspondences may be dominated by others. As in our mapping program ACME, it is possible that the best match will not be one-to-one; in contrast, the structure-mapping engine (SME) insists that all matches be one-to-one [13]. Excitatory links are set up with weight *excit*, whereas inhibitory links have weight *inhib*.

To implement the constraint of pragmatic centrality, links to the pragmatic unit are set up in accord with the instructions in Fig. 7. For example, if a predicate occurs in the statement of a goal, it is marked as important, as are predicates in starting conditions whose relevance to the goal is evident because of higher-order IF or CAUSE relations. Links from the pragmatic unit to units involving important predicates will encourage correspondences involving those predicates over correspondences involving less relevant predicates. Sometimes, certain correspondences are indicated in advance, for example by a teacher

Start with the list of predicates **search-preds** found by the algorithm described in Fig. 4. For each member search-pred$_j$ of **search-preds**:

(A) Create the unit pred$_{P,i}$=search-pred$_j$.
(Here pred$_{P,i}$ is from the probe structure: see Fig. 4).

(B) Connect this unit to the semantic unit with a weight based on the semantic relation between the two predicates. The weight is the result of multiplying a parameter for maximum degree of excitation (typically 0.01) times:
for identical predicates, 1;
for synonyms, 0.6;
for superordinates, 0.3;
for coordinates, 0.25;
(Note. If *A* and *B* are both kinds of *C*, then *A* and *B* are coordinates, i.e. subordinates of superordinates.)
for subordinates, 0.2;
for superordinates of superordinates, 0.2;
for holonyms (part-ofs), 0.1;
for parts, 0;
for antonyms, −0.4.
(See section 6.3 for discussion of these weights.)

(C) Get the list **search-props** of propositions in source structures that contain search-pred$_j$. (When the memory of analog structures is loaded, ARCS notes for each predicate the names of the propositions that contain it, and notes for each proposition the structure that contains it.)

(D) For each proposition prop$_{P,i}$ in the probe P containing pred$_{P,j}$, and for each member search-prop$_k$ of **search-props**;
(1) Create the unit prop$_{P,i}$=search-prop$_k$.
(2) Link this unit to pred$_{P,i}$=search-pred$_j$.
(3) For each argument of the proposition, create units for arg$_{P,i,m}$=the *m*th argument of search-prop$_k$:
(a) Link each of these units to prop$_{P,i}$=search-prop$_k$.
(b) Link each of these units to pred$_{P,i}$=search-pred$_j$ (weight = excit * number of times this object correspondence occurs with these predicates),

Fig. 5. Algorithm for creating a constraint network. In (D.2), as in the ACME mapping program, a link is increased in strength for each proposition that supports it. Thus if the probe contains two occurrences of a predicate, then links to units involving that predicate will double in strength. Except as indicated in (B), all excitatory links have weight set by the parameter *excit*.

(4) Find the source structure $struc_n$ which contains search-$prop_k$. (This is trivial, since when memory is loaded, the information about the structure that contains it is stored with each proposition name.)
(5) Create the unit P=$struc_n$, hypothesizing that the source structure is relevant to the probe structure.
(6) Link this unit to the unit $prop_{P,i}$=search-$prop_k$ and to $pred_{P,i}$=search-$pred_j$.

Fig. 5. *Continued.*

After semantic similarity has been used to create the network of units, set up inhibitory links between pairs of incompatible hypotheses, considering:
The competing source structure hypotheses;
the competing predicate hypotheses;
the competing proposition hypotheses; and
the competing object hypotheses.
Every hypothesis of the form $A=B$ gets an inhibitory link to every hypothesis of the form $A=C$ or $C=B$.

Fig. 6. Algorithm for setting up inhibitory links.

giving a hint, so some mappings can be presumed and the units representing them are linked to the pragmatic unit.

Finally, Fig. 8 describes the algorithm for synchronously updating the activations of the different units. Note that this algorithm is fully parallel: once the network is set up, the adjustment of the activation of each unit can be done simultaneously. This parallelism has both conceptual and practical advantages. Practically, this means that on a computer with as many processors as the number of units and links, the amount of time it takes to update the entire network should be roughly constant despite dramatic increases in the size of the data base of stored analogs. Conceptually, the algorithm has the advantage of simultaneously determining the best way to satisfy as much as possible all of the constraints that underlie the construction of the network. The equation used for updating activation is that suggested by Grossberg [24]. The activation level of unit j on cycle $t+1$ is given by:

$$a_j(t+1) = a_j(t)(1-d) + enet_j(max - a_j(t)) + inet_j(a_j(t) - min). \quad (1)$$

Here d is a decay parameter, $enet_j$ is the net excitatory input, and $inet_j$ is the

(A) Note elements that are considered IMPORTANT, either from input statements, or as determined by the following analysis:
  (1) If the probe structure P is a problem, mark every predicate in a goal proposition as important, as well as every predicate in a starting condition proposition that is determined to be goal-relevant by considering CAUSE, IF and SATISFIES relations. For example, if P1 is a start proposition in the probe P, and P1 causes P2 which is a goal proposition in P, then P1 is important.
  (2) If the probe structure P is an explanation, mark every predicate in a proposition to be explained as important, as well as every predicate in a starting condition, using the same determination of relevance used in (1).
  (3) If the probe structure P is an argument, mark every predicate in the conclusion as important, as well as every predicate in a premise proposition that has direct relevance to the conclusion as indicated by IF relations.

(B) For each element E considered IMPORTANT, connect all units which contain hypotheses about E to the pragmatic unit with a weight prag1, typically 0.01, a parameter for pragmatic excitation of hypotheses with important elements.

(C) If any mappings are considered PRESUMED, connect the units representing the specified mappings to the pragmatic unit with a weight prag2, typically 0.1, a parameter for pragmatic excitation of presumed hypotheses.

Fig. 7. Algorithm for implementing pragmatic centrality.

net inhibitory input (a negative number), with minimum activation $min = -1$ and maximum activation $max = 1$. Inputs are determined by the equations:

$$enet_j = \sum_i w_{ij} o_i(t) \quad \text{for } w_{ij} > 0 ; \tag{2}$$

$$inet_j = \sum_i w_{ij} o_i(t) \quad \text{for } w_{ij} < 0 . \tag{3}$$

Here $o_i(t)$ is the output of unit i on cycle t, set by:

$$o_i(t) = \max(a_i(t), 0) . \tag{4}$$

In ARCS, updating continues until all units have reached asymptote, that is, a cycle is reached at which the activation change of each unit is less than a specified value, typically 0.001. As shown by the experiments described below, these algorithms for setting up and adjusting networks are very effective for

(A) Running the network:

Set all unit activations to an initial starting value (typically 0.01), except that the special semantic and pragmatic units are clamped at 1.
Update activations in accordance with (B) below.
If no unit has changed activation more than a specified amount (usually 0.001), or if a specified number of cycles of updating have occurred, then stop.

(B) Synchronous activation updating at each cycle:

For each unit u:
calculate the new activation of u in accord with equations (1)–(4) in the text, considering the old activation of each unit u' linked to u.

Fig. 8. Algorithms for network operation.

selecting relevant analogs. Activation levels become stable in a reasonable number of cycles—typically less than 150—and ARCS works well over a wide range of parameter values (see Section 6.3). ARCS currently runs serially on Sun workstations, and real run-times on a Sun 4 for the simulations described in Sections 5 and 6 range from half a minute to almost an hour for the largest network with over 60,000 links.

To sum up, let us review how the algorithms of ARCS implement the three constraints discussed in Section 2. Semantic similarity is implemented by the algorithms described in Figs. 4 and 5(B). Isomorphism is implemented by the algorithms in Figs. 5 and 6: see Fig. 5(D) for structural consistency, and Fig. 6 for one-to-one mapping. Figure 7 shows how pragmatic centrality is implemented in ARCS. The network of units and links embodies all three kinds of constraints. Finally, Fig. 8 shows how simultaneous satisfaction of all three constraints can be computed by updating activations of units in the network.

It is important to be clear about what ARCS is *not* doing. It does not compare the probe with every structure stored in memory, but considers only those that have semantically similar predicates. Nor does it do a complete match between the probe and the source analogs whose potential relevance is indicated by semantic similarity. Unlike our mapping program ACME and Falkenhainer, Forbus and Gentner's SME, ARCS does not calculate a global map of the probe to the source, but only considers those propositions containing semantically similar predicates. Typically, in comparing the probe and the source, ARCS looks at much less of each structure than does ACME, which attempts to map predicates that are not semantically similar. In two of our retrieval runs, we found that ARCS created fewer than 5% of the units needed for a full ACME

comparison of two structures. Doing a full mapping between the probe and each possible source would increase the chance that the selected source is really the best possible, but the added computational load is not worth it if a simpler, semantics-driven retrieval system yields plausible potential sources, as seems to be the case for human psychology. Whereas ACME and SME compute a full set of candidate mappings, ARCS is restricted to correspondences suggested by semantic similarity. Finally, ARCS does not have a two-stage process in which the overall match between the probe and each source is computed and then compared. Rather, the relative value of the source analogs is computed in parallel through the relaxation of the network of units.

3.3. Computational complexity

Holyoak and Thagard [35] showed that the space complexity of ACME, which produces a mapping between two structures, is at worst $O(n^4)$, where n is the number of propositions in the larger structure. This followed from the calculation that the number of units was at worst on the order of n^2, and there cannot be more links than there are units squared. Since ARCS, unlike ACME, only puts in correspondence concepts that are semantically similar, ARCS can do no worse than ACME when only the relation between the probe structure and a single stored structure is considered. Hence the number of units for ARCS can be no worse than k times the number for ACME, where k is the number of structures stored in memory. The maximum number of units in an ARCS run is therefore $O(kn^2)$. Hence the maximum number of links and the overall space complexity of ARCS is $O(k^2n^4)$. Experiments reported in Section 6 show that the numbers tend to be much smaller, since ARCS does not put in correspondence propositions whose predicates are not semantically similar.

Calculation of the time complexity of ARCS requires taking into account both the time to create the constraint network and the time for the network to settle. The complexity of network creation is the same as the space complexity, $O(k^2n^4)$, assuming that a constant number of steps is required to create each unit and link. Unfortunately, it is not possible to determine analytically whether c, the number of cycles it takes the network to settle, depends on the number and size of structures in memory. Fortunately, however, our experiments with both ACME and ARCS suggest that for larger networks the number of cycles required by a network to settle is roughly constant and independent of the size of the network (see Section 6). In principle, it is possible to implement all the units and links on separate processors, so that a parallel machine could reduce the time complexity of settling in ARCS to $O(c)$.

4. Comparison with Other Computational Models

ARCS can usefully be compared with two sorts of computational models: other models of analog retrieval and memory, and other connectionist models. A

fully comprehensive review is beyond the scope of this article, but we will contrast ARCS with models that are similar and different in interesting ways.

4.1. Models of analog retrieval

Many computational models of analogy have been proposed, as Hall [25] has comprehensively reviewed. Here we will primarily compare ARCS with those models that are most similar in being concerned with retrieval of analogs. We will not discuss programs whose major function is mapping between two given analogs [13].

The most distinctive features of ARCS are: it applies all of the constraints of (1) semantic similarity, (2) structural consistency, and (3) pragmatic centrality; it (4) uses a parallel algorithm for determining the stored analogs that best satisfy these constraints; and it (5) uses competitive interactions among alternatives to screen out candidates. We will therefore consider whether other computational models of retrieval use each of the three kinds of constraints, parallel algorithms, and competition. This determination is not always straightforward, as previous models have seldom been explicitly described in terms of the five features that we view as basic to ARCS. Nonetheless, such comparisons are at least suggestive of the similarities and differences among extant models.

We judge a model to use the semantic similarity constraint if it retrieves analogs that are related to the probe analog by having elements that are related semantically via synonyms, antonyms, kind relations, and/or part–whole relations. Using identical predicates as matches can be understood as a limiting case of semantic similarity. We judge a model to use isomorphism if it is sensitive to argument order differences, such as the difference between "dog bites boy" and "boy bites dog." A model is viewed as using the constraint of pragmatic centrality if it takes into account the goals and purposes for which the analog is intended to be used. We count an algorithm as parallel if it simultaneously evaluates the potential relevance of any number of stored analogs. Finally, we count a model as involving competitive retrieval if evidence favoring one potential source analog can be used to reduce the likelihood of retrieving another. By competitive retrieval we mean more than simply that some candidates are selected as better than others. Any retrieval program will be comparative; for example, a program might simply calculate a degree of match between the probe and each source structure and report the highest. By competitive retrieval we mean that *during the process of retrieval* the different source structures vie with each other to be the best match, with the claims of one tending to suppress the claims of the others. ARCS is competitive in this sense since the activation of one PROBE=SOURCE unit suppresses by means of inhibitory links the activation of all other PROBE=SOURCE units. After the network has settled, the supremacy of the

most active PROBE=SOURCE unit reflects its having beaten out all the others. Competitive retrieval is most naturally done as part of a parallel process, but there are many ways of doing parallel retrieval that are not competitive because structures are retrieved completely independently of each other. Psychological studies of interference effects in human long-term memory provide evidence that human memory retrieval is in fact competitive (e.g. [2, 5]).

Table 1 summarizes the results of our comparisons. In the table, "yes" means the model has the relevant feature; "+" and "−" signs indicate the model makes notably greater or lesser, respectively, use of the feature than does ARCS. Keep in mind that these comparisons solely involve retrieval mechanisms; many of these models have other features when they are applied to analogical mapping.

Let us first compare the ARCS model with our own earlier work. Holyoak and Thagard [34] proposed an account of analogical problem solving within the context of the PI system [31, 60, 61]. PI solves problems by firing of production rules and spreading activation through a network of frame-like concepts. The program uses semantic similarity in a weaker way than does ARCS because activation spreads from a concept only to other concepts that are its superordinates and subordinates. Otherwise, activation spreads by rule firing, which involves both forward chaining and backward chaining from goals to be accomplished. This last feature shows that PI does use pragmatic centrality to a limited extent. PI differs from ARCS in that it does not consider structural consistency at all, and it considers semantic and pragmatic constraints in a much more limited way. Analogs are retrieved in PI by means of summation of activation of the concepts with which they are stored, a parallel process, but PI lacks a means for competitively selecting the best match in memory. That is, PI can pick the most active analog, but the analogs and their components do not inhibit each other, so that there is no competition for activation as there is in ARCS. Another major difference is that the spreading activation mechanism in

Table 1
Comparison of ARCS with other computational models of analogical retrieval

	Semantic similarity	Isomorphism	Pragmatic centrality	Parallel algorithm	Competitive retrieval
ARCS	yes	yes	yes	yes	yes
PI	yes−	no	yes−	yes	no
Anderson	yes−	no	yes	yes	yes−
Carbonell	yes−	yes	yes+	no	no
Hammond	yes−	no	yes+	no	no
Kolodner	yes−	no	yes+	yes	no
Winston	yes−	no	no	no	yes
Lehnert	no	yes	no	yes	yes

Note. yes+ indicates that a constraint is used to a greater degree than in ARCS, while yes− indicates that it is used to a lesser degree. See the text for references and names of programs.

PI is controlled only by a decay parameter, so that large parts of the data base could eventually be activated through a wide variety of inferences and activations. ARCS does a much narrower search into memory, looking only at stored analogs that have some semantic overlap with the probe analog.

Anderson and his colleagues [3, 4, 46] have used spreading activation as a mechanism for retrieval. In the PUPS system [4] stored analogs are representations of LISP functions describing their form and function. Neither PUPS nor ACT* [3] give semantic relations a special role, although they may be indirectly present in the propositions that are the sources of connections between nodes. In ACT*, any proposition involving two concepts C1 and C2 will give rise to a link between them. Since some of these propositions will encode semantic relations, such as the fact that a dog is a kind of animal, these relations will be among those that provide lines of spreading activation, although with no greater effect than a proposition stating that dogs have fleas. Structural considerations are not directly used in ACT*, although the form and function specifications in PUPS do have a structural role in mapping. ACT* allows goal elements to affect the spread of activation, giving it a pragmatic dimension. Spreading activation is a fully parallel process; however, ACT* does not make use of inhibitory connections to produce competitive retrieval. Retrieval in ACT* is competitive in a weaker sense, however: for any node, outgoing activation along a link is proportional to the strength of that link relative to summed strengths of all links. Hence a receiving node will get less activation from a sending node if there are many other nodes linked to the sending node.

It is important to distinguish the notion of activation used for retrieval in ACT* and PI from the very different notion of activation used in connectionist systems such as ARCS. In ARCS, the activation of a unit represents the plausibility of a hypothesis concerning a correspondence between a probe structure and a stored structure. Retrieval in ARCS works fundamentally by semantic elaboration of the probe and construction of a network of units representing correspondence hypotheses: activation of units is used simply to provide a simultaneous evaluation of the hypotheses. It is possible that a full model of human memory may need to include a looser process of spreading activation in addition to the tightly constrained retrieval process used by ARCS. Models influenced by Schank's [55] theory of reminding emphasize indexing of structures in memory on the basis of their causal relevance to the accomplishment of goals and avoidance of failures. Carbonell's [6, 7] analogy systems, implemented in his ARIES program, use a similarity metric that heavily weights relevance to solution, so it seems to be primarily concerned with pragmatic centrality, although structural relations are also taken into account. The MEDIATOR system of Kolodner and Simpson [39] uses classifications of disputes as indexing cues. For example, given the Sinai dispute as a probe, it retrieves the Korean conflict because the objects of dispute are classified as being of the same type and because they used the same type of force to achieve

a similar goal. This system therefore appears to use both semantic and pragmatic constraints; it does not appear to use a parallel algorithm for retrieval. Kolodner's [37] earlier memory model CYRUS, which used a sort of discrimination-net approach to selecting cases from a memory organized in terms of E-MOPs (episodic memory organization packets), has a natural parallel implementation. The hierarchical organization of memory used by ARCS is limited to its lexical semantics, in contrast to CYRUS which has all the E-MOPs arranged in a network. In ARCS, the only connections between source structures are provided by the lexical semantics represented in individual concepts. In her most recent system, PARADYME, Kolodner [38] uses a set of preference heuristics to choose the best match from a set of partially matching cases retrieved by a parallel process. Unlike ARCS, which simultaneously satisfies its three constraints, PARADYME's six heuristics are ordered and applied sequentially, with preference given first to cases that can help address the reasoner's current reasoning goal.

The CHEF system of Hammond [26] uses different kinds of memory organization for different purposes: successful plans, failures, modifications, and repair strategies. The memory for successful plans appears to be entirely organized in terms of goal accomplishment, and so is primarily pragmatic. Embedded in the goal hierarchies, however, are some semantic relations disguised as goal relations, such as the information that if you want to cook seafood you can cook shrimp. Hammond contends that work on memory has to consider the function of memory; however, memory serves many functions besides problem solving, and therefore needs a more flexible organization than is found in CHEF. Neither CHEF nor the MEDIATOR system involves competitive retrieval.

Winston's [66, 67] model of analogy is primarily concerned with mapping between analogs and learning, but he also discusses finding analogies by "classification-exploiting hypothesizing." His retrieval mechanism operates by moving down an annotated A-KIND-OF hierarchy of frames representing situations, with the slots in the frames "voting" concerning their relevance to the probe situation. The voting mechanism appears to make retrieval competitive, since the degree to which the slots of a frame count in favor of the relevance of the potential source situation depends on slots in other frames that are also voting. The use of the A-KIND-OF hierarchy implies that retrieval is largely semantic, although Winston's mapping program also uses the constraints of isomorphism and pragmatic centrality. As implemented, the algorithm is serial, but traversal of the A-KIND-OF hierarchy would presumably not be hard to do in parallel.

Finally, a comparison can be made with Lehnert's [40] system for retrieving word pronunciations. Like ARCS, her PRO system uses a parallel relaxation algorithm to select from memory the pronunciations most similar to a probe, and it does so in part by keeping track of structural relations. However, since this domain is phonological, there is no relevant semantic or pragmatic

information to be used. The analogs that have been used to test the ARCS model are far more complicated both structurally and semantically than the strings of phonemes that constitute the cases in the PRO system. Stanfill and Waltz [59] also do parallel retrieval of word pronunciations from large data bases, but their approach is very different from Lehnert's or ours: they use an explicit metric to measure the similarity between a probe record and a stored record, then pick the most similar. Retrieval in their system is thus comparative, but not competitive.

It is interesting to consider how it might happen in different systems that no structure gets retrieved, simulating the occasions in human thought when nothing comes to mind. In ARCS, a probe can fail to retrieve any structures if there are no structures in memory with any semantic overlap with the probe. An extreme example would be a probe consisting of nonsense words. More naturalistically, retrieval failures can arise if only a small degree of overlap exists between the probe and a structure stored in memory. For example, if a story about computers is used as a probe into the data base of fables described in Section 6.1, no unit representing a computer story to fable correspondence acquires activation much above 0, even if the story happens to mention a fox. The predicate FOX appears in numerous fables, but the different units representing the relevant PROBE=FABLE mappings suppress each other and none become very active. Retrieval systems that use explicit metrics would presumably judge that nothing is retrieved if no stored structure surpasses some minimum threshold for similarity to the probe.

To sum up, all the above models assume that analog retrieval involves some means of doing a partial similarity match against a host of analogs, but they differ primarily in what aspects they consider most relevant to similarity, and in how the comparison is actually executed. ARCS embodies the hypothesis that semantic, structural, and pragmatic similarity are all important, and shows how to give an integrated parallel calculation of overall similarity. ARCS does not require an explicit similarity metric that is applied to various structures in memory, judging each independently of other potential analogs. Rather, a judgment about the relative similarity of potentially relevant stored analogs emerges from parallel competition for activation. ARCS differs from ACT*-style spreading activation in that the competition for activation is among hypotheses concerning correspondences, not among structures representing analogs, so that ARCS can gracefully handle structural pressures as well as semantic and pragmatic ones.

4.2. Connectionist and related models

The ARCS model uses a connectionist relaxation algorithm to select analogs from memory, but we do *not* want to draw a strong distinction between our approach and traditional "symbolic" approaches. Our basic memory repre-

sentations of analog structures and concepts are quite traditional, being similar to predicate-calculus and frame-like representations. What is more novel in ARCS (and its sibling, ACME) is the use of a constraint network of units representing hypotheses about correspondences to select the best analogs. Feldman and Ballard [59] provide an introduction to the general approach. The correspondence units in ARCS are conceptually related to the "binding nodes" used in the parsing system developed by Cottrell [9].

Our approach should be distinguished from the more radical "PDP" brand of connectionism that uses distributed representations. Common to connectionist approaches is the use of units with excitatory and inhibitory links, but the interpretation of the units varies substantially. A unit can be used to represent a symbolic entry in memory such as a concept [57] or a proposition [61]. It is a mistake to describe such *localist* models as "nonsymbolic." In contrast, PDP (parallel distributed processing) theorists recommend that concepts and propositions be distributed across numerous units, so that a symbolic entity corresponds to patterns of activation across numerous units [28, 54]. Among localist models, we can distinguish ones in which the units represent parts of permanent memory from ones in which the units and the connections between them are more ephemeral, created on the fly for a particular task. The networks ARCS creates are ephemeral in this sense. Another example of the use of ephemeral networks is in the frequently discussed example of the Necker cube [15], in which the units represent hypotheses concerning which of the parts of the cube are in front and which are in back. No one would claim that every hypothesis about reversing cubes is part of permanent memory. Kintsch [36] has proposed a connectionist model of discourse comprehension that includes ephemeral units constructed by a parser, as well as permanent lexical ones.

To sum up, connectionists models can be local or distributed, and the local ones can use permanent or ephemeral units; the networks in ARCS are local and ephemeral. Although ARCS constructs a new network for every retrieval, we are currently building a cognitive architecture in which ARCS-style retrieval proceeds in an incremental fashion so that the constraint network, while still not permanent, can persist and grow through a problem-solving episode. The ARCS constraint network should not be thought of as "working memory" as this is normally understood by psychologists, since it represents processing at a much more unconscious level. ARCS's networks have roughly the same theoretical status as the RETE nets used to match the conditions of rules in production system models of cognition, although it should be noted that ARCS is adept at finding *partial* matches. Indeed, we are experimenting with a new cognitive architecture that smoothly integrates analog retrieval with rule-based reasoning, using ARCS-style networks to perform partial matches of the conditions of rules.

Thagard [61] distinguishes eight different approaches to understanding the nature of mind and intelligence:

(1) straight neuroscience, studying neurons or sections of the brain;
(2) computational modeling of actual neurons in the brain;
(3) connectionist models using distributed representations, so that a concept or hypothesis is a pattern of activation over multiple units;
(4) connectionist models using localist representations, in which a single unit represents a concept or proposition;
(5) traditional artificial intelligence models using data structures such as logical expressions, frames, and production rules;
(6) cognitive-level psychological experimentation;
(7) mathematical analysis;
(8) theoretical speculation about the general nature of mind.

ARCS is a hybrid model embodying approaches (4) and (5) (see Hendler [27] for a discussion of the advantages of hybrid approaches). A stark connectionist versus symbolist distinction is an impediment to conceptual progress for at least two reasons. First, a sharp distinction overlooks the existence of many interesting localist connectionist models and the presence in distributed models of symbolic inputs and outputs. Second, it threatens to block exploration of interconnecting aspects of what we see as compatible computational methods.

We are not aware of any other connectionist models of analogy, but what such a model might be like is suggested by the model of distributed memory advocated by McClelland and Rumelhart [42, 53]. In their system, a mental state is a pattern of activation over the units that comprise permanent memory. Memory traces are changes in the weights between units. An experience produces a particular pattern of activation and an adjustment of the weights between units. Later similar experiences produce a similar activation pattern, amounting to retrieval of the earlier experience. This distributed approach is potentially an elegant implementation of semantic similarity, since experiences with a set of properties will be stored as patterns of activation that become activated by experiences with similar properties. We predict, however, that it will be much harder to implement the constraint of isomorphism, since it is not clear how a distributed representation can keep track of argument structure. In analogical reasoning, it is crucial to distinguish "boy bites dog" from "dog bites boy," since structural consistency is a key influence on the extent to which two structures correspond. (See Ratcliff and McKoon [47] for discussion of additional difficulties encountered by models based solely on vector representations in accounting for human retrieval of rational information.) ARCS' implementation of structural consistency depends on the use of units representing hypothetical correspondences involving propositions and parts of propositions, not just features.

The Copycat project of Hofstadter and Mitchell [30] investigates analogy in a spirit similar to connectionist approaches. Using many small "codelets" interacting in parallel, Copycat solves analogy problems using letter strings. Given

the example that *abc* goes to *abd*, it comes up with a number of answers to the question of what *pqr* goes to. Copycat uses a network of concepts, called a *Slipnet*, to find correspondences between non-identical objects; this is roughly similar to ARCS' use of lexical semantics to find similar predicates. The processes by which Copycat finds correspondences differ, however, in two important respects from ARCS. First, Copycat is nondeterministic, since the codelets are chosen nondeterministically from a constantly changing pool. Second, Copycat's parallel search for correspondences varies in speed determined dynamically by moment-to-moment evaluations of the promise of each match. Copycat is better described as *creating* analogies rather than as retrieving them, and nondeterminism arguably increases the possibilities of creativity. Its use of the Slipnet can be construed as an implementation of the constraint of semantic similarity, but Copycat does not appear to implement the constraints of isomorphism or pragmatic centrality.

5. Psychological Evaluation

ARCS is intended to be a model of human cognition, so it is essential that it be consistent with the results of psychological experiments on analog retrieval. We will accordingly describe its performance in simulating the results of two sets of experiments done by Holyoak and his colleagues and by Gentner and her colleagues. Section 6 describes computational experiments concerning how well ARCS scales to larger data bases.

5.1. Convergence problems

As reviewed briefly above, Holyoak and his colleagues have done a series of experiments in which the target problem to be solved concerns using an X-ray to destroy a tumor deep inside a patient [21, 22, 33] (see Duncker [11] for the original use of the tumor problem). Simply shooting the X-ray beams at the tumor is not an acceptable solution, because the beams, if shot at sufficient intensity to destroy the tumor, will also destroy intervening healthy tissue and kill the patient. A "convergence" solution is appropriate: using multiple beams weak enough not to hurt the patient, but aimed from several directions so that they converge on the tumor with enough intensity to destroy it. Very few subjects discover this strategy on their own, but they are more likely to formulate the convergence solution if they have first been provided with an analog. In the various experiments, analogs have been used with markedly different degrees of similarity to the tumor problem, and subjects' ability to retrieve them from memory and notice their relevance, without any hint from the experimenter, has varied accordingly.

We cannot simulate the relevant experiments directly, since that would require constructing the full knowledge base of the various subjects and incorporating different stored analogs. However, since we are interested in the

relative retrievability of the different analogs, we can contrive a data base consisting of all of the analogs and see which ones ARCS prefers to retrieve. Our data base accordingly consists of the following five structures:

- the laser problem (version 1), in which a laser beam is used to fuse a filament in a lightbulb by convergence from several directions, avoiding breaking the bulb (high semantic similarity, high structural consistency);
- the laser problem (version 2), in which a laser beam is used to fuse a filament in a lightbulb, but where convergence is required because the original beam is too weak, not because of the danger of destroying the bulb (high semantic similarity, low structural consistency);
- the ultrasound problem (version 1), in which an ultrasound device is used to split apart a filament in a lightbulb by convergence from several directions, avoiding breaking the bulb (low semantic similarity, high structural consistency);
- the ultrasound problem (version 2), in which an ultrasound device is used to split apart a filament in a lightbulb, but where convergence is required because the original beam is too weak, not because of the danger of destroying the bulb (low semantic similarity, low structural consistency);
- the fortress problem, in which an army succeeds in attacking a fortress by dispersing its troops and attacking it from multiple directions; here the convergence solution is necessary because a frontal assault will lead to the destruction of the army.

The first four of the above problems provide a factorial manipulation of degree of semantic similarity and degree of structural consistency, and their relative accessibility was assessed in a single experiment by Holyoak and Koh [33]. The fifth problem has been used in many other experiments [8, 21, 22, 58]. To test the relative retrievability of these analogs, we translated the materials used by Holyoak and his colleagues into predicate calculus and added the relevant semantic information concerning the predicates used. An English version of the tumor problem is the following:

> Suppose you are a doctor faced with a patient who has a malignant tumor in his stomach. It is impossible to operate on the patient, but unless the tumor is destroyed the patient will die. There is a kind of ray that can be used to destroy the tumor. If the rays reach the tumor all at once at a sufficiently high intensity, the tumor will be destroyed. Unfortunately, at this intensity the healthy tissue that the rays pass through on the way to the tumor will also be destroyed. At lower intensities the rays are harmless to healthy tissue, but they will not affect the tumor either. What type of procedure might be used to destroy the tumor with the rays, and at the same time avoid destroying the healthy tissue?

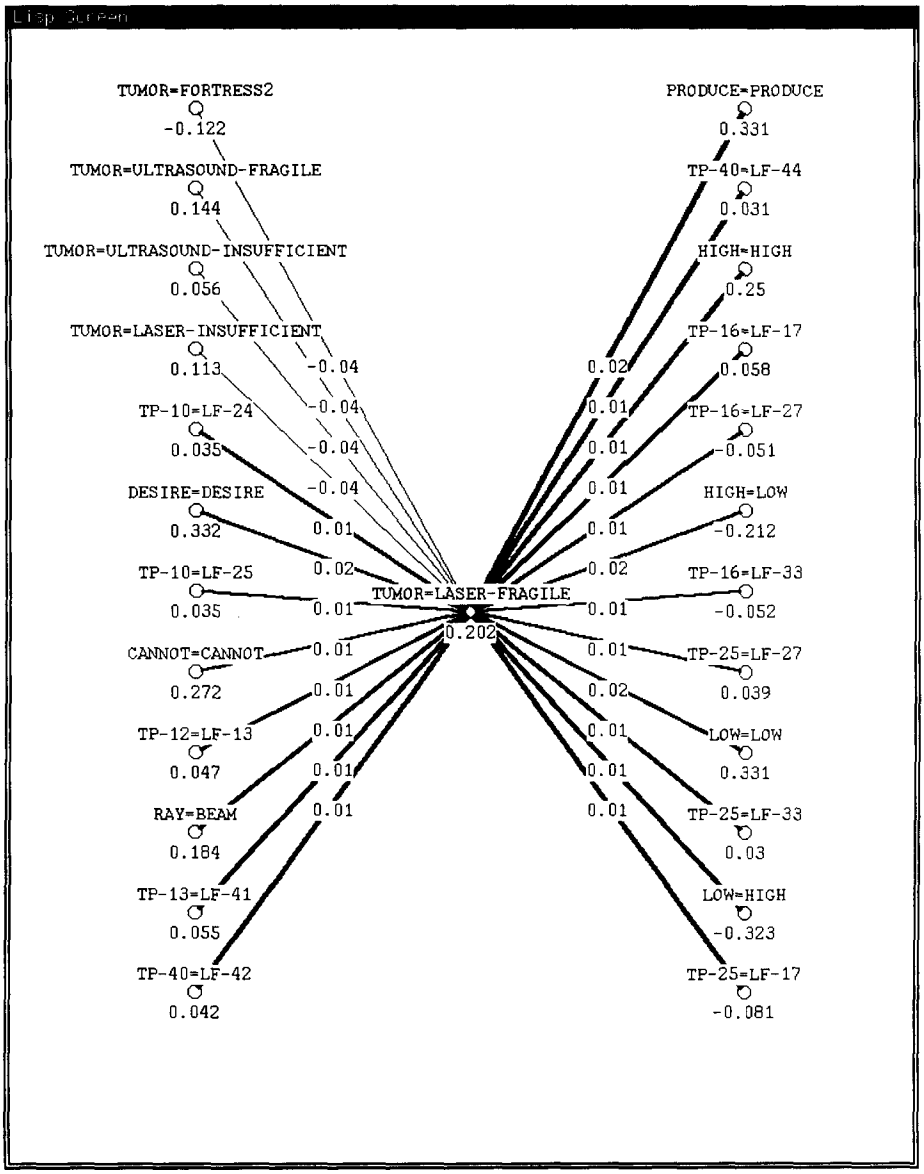

Fig. 9. Connectivity of a sample unit. Thick lines indicate excitatory links, while thin lines indicate inhibitory links. Numbers on lines indicate the weights of the links. Numbers under unit names are the truncated asymptotic activations of the units.

Our predicate calculus translation used as the probe was given in Fig. 2. WordNet-style concepts were constructed for the 72 predicates used in the probe problem and in the five stored structures.

We then used the tumor problem as a probe into the data base containing the other five problems. ARCS created a network of 100 units and 1104 links, and took 95 cycles to settle. Figure 9, derived from a graphics program that runs concurrently with ARCS, shows the connectivity of the unit TUMOR=LASER-FRAGILE, representing the hypothesis that the tumor problem corresponds most closely with the laser problem concerning not breaking the glass bulb. This laser problem is in fact most comparable to the probe tumor problem because it has greater structural consistency with it than does the laser problem in which the constraint is that the laser beam is not strong enough, and greater semantic similarity with it than do the ultrasound and fortress problems. It should also be noted that on the basis of the representation shown in Fig. 2, ARCS marks the propositions tp-18 and tp-22, along with their predicate DESTROY, as IMPORTANT since they are mentioned in the goals of the problem. These propositions do not, however, appear in Fig. 9, because they do not correspond to any propositions in the representation of the laser-fragile problem. Figure 10 depicts a graph, also produced by ARCS, of the activations of the five units representing hypotheses about the appropriateness of retrieving each of the stored structures. Each graph charts the activation of a unit (the range of the y-axis is from 1 to -1, with the horizontal line indicating 0) over 100 cycles of updating.

The results correspond quite well with the degree of retrievability observed in the psychological experiments. Table 2 compares (1) the percent of subjects in the experiments who were able to generate the convergence solution without any hint with (2) the activation of the unit representing the correspondence between the probe tumor problem and the stored analog. For the four versions of the lightbulb problems, the order of activations corresponds to the empirical

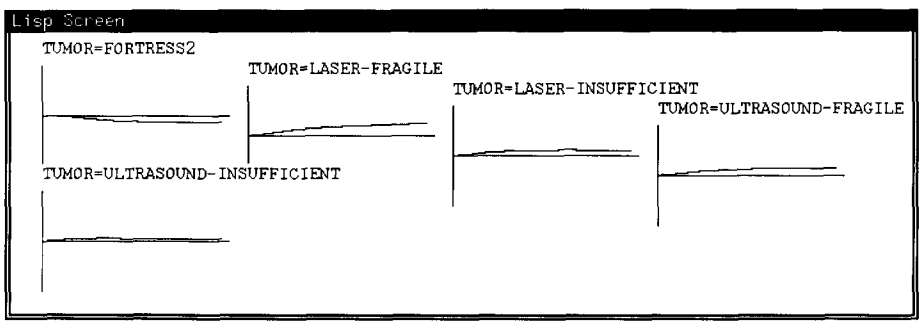

Fig. 10. Activation histories of selected units in convergence-problems simulation. Each graph shows activation on a scale of 1 to -1, with the horizontal line indicating the starting activation of 0.

Table 2
Comparison of ARCS results with retrievability of radiation problem analogs

	Percentage of subjects achieving solution	Asymptotic activation of ARCS unit
Laser problem (fragile glass)	69	0.22
Ultrasound problem (fragile glass)	38	0.15
Laser problem (insufficient intensity)	33	0.11
Ultrasound problem (insufficient intensity)	13	0.05
Fortress problem	30	−0.12

Note. Percentages for first four versions are taken from Holyoak and Koh [33]; percentage for fifth version is an approximate average based on several experiments reported by Gick and Holyoak [21, 22].

measure of accessibility in the Holyoak and Koh [33] experiment. Similarly, Holyoak and Thagard [35] used the ACME program to simulate post-retrieval mapping performance using data from this same experiment.

The relative activation of the fortress problem is worse than the empirical measure would apparently predict. However, this measure was obtained with different subjects in different experiments than was the data for the other four problems, so precise comparisons are unwarranted. Also, subjects in the actual experiments using the fortress problem had only one of the lightbulb problems stored in memory, whereas ARCS had all of them in competition.

The above results were obtained with excitation set at 0.01, inhibition at −0.04, and decay at 0.04. Sensitivity analyses to be described in Section 6.3 show that the results do not depend on these particular values.

5.2. "Karla the Hawk" stories

Gentner and her colleagues have performed a series of experiments concerning the retrievability of a number of similar stories, and ARCS has also been used to simulate these results. Figure 11 gives the English versions of one set of stories from Rattermann and Gentner [48], while Fig. 12 provides our predicate-calculus translation of the first story.[2] Rattermann and Gentner first had subjects read a list of 32 stories, including 20 crucial stories such as "Karla the Hawk." Then subjects were given a series of probe stories, including one of the

[2]Because of its use of WordNet semantics, ARCS is able to handle more accurate and detailed representations of the stories than those used by Falkenhainer, Forbus and Gentner [13] to test their SME program. In their versions, for example, the propositions that Karla is a hawk and that Zerdia is an eagle were both represented using the predicate "bird": (bird karla), (bird zerdia). ARCS' semantics enables it to make the appropriate correspondence between (hawk karla) and (eagle zerdia), since hawks and eagles are both kinds of bird.

Base Story

Karla, an old hawk, lived at the top of a tall oak tree. One afternoon, she saw a hunter on the ground with a bow and some crude arrows that had no feathers. The hunter took aim and shot at the hawk but missed. Karla knew the hunter wanted her feathers so she glided down to the hunter and offered to give him a few. The hunter was so grateful that he pledged never to shoot at a hawk again. He went off and shot deer instead.

Literal Similarity

Once there was an eagle named Zerdia who nested on a rocky cliff. One day she saw a sportsman coming with a crossbow and some bolts that had no feathers. The sportsman attacked but the bolts missed. Zerdia realized that the sportsman wanted her tailfeathers so she flew down and donated a few of her tailfeathers to the sportsman. The sportsman was pleased. He promised never to attack eagles again.

True Analogy

Once there was a small country called Zerdia that learned to make the world's smartest computer.

One day Zerdia was attacked by its warlike neighbor, Gagrach. But the missiles were badly aimed and the attack failed. The Zerdian government realized that Gagrach wanted Zerdian computers so it offered to sell some of its computers to the country. The government of Gagrach was very pleased. It promised never to attack Zerdia again.

Mere Appearance

Once there was an eagle named Zerdia who donated a few of her tailfeathers to a sportsman so he would promise never to attack eagles.

One day Zerdia was nesting high on a rocky cliff when she saw the sportsman coming with a crossbow. Zerdia flew down to meet the man, but he attacked and felled her with a single bolt. As she fluttered to the ground Zerdia realized that the bolt had her own tailfeathers on it.

False Analogy

Once there was a small country called Zerdia that learned to make the world's smartest computer. Zerdia sold one of its supercomputers to its neighbor, Gagach, so Gagrach promised never to attack Zerdia.

But one day Zerdia was overwhelmed by a surprise attack from Gagrach. As it capitulated the crippled government of Zerdia realized that the attacker's missiles had been guided by Zerdian supercomputers.

Fig. 11. English versions of "Karla the Hawk" stories (from Ratterman and Gentner [48]).

(hawk (obj-karla) true bsf-1)
(old (obj-karla) true bsf-2)
(oak (obj-tree) true bsf-3)
(live-in (obj-karla obj-tree) true bsf-4)
(afternoon (obj-afternoon) true bsf-5)
(hunter (obj-hunter) true bsf-6)
(see (obj-karla obj-hunter) true bsf-7)
(when (obj-afternoon bsf-7) true bsf-8)
(bow (obj-bow) true bsf-9)
(arrows (obj-arrows) true bsf-10)
(crude (obj-arrows) true bsf-11)
(feathers (obj-feathers) true bsf-12)
(have (obj-karla obj-feathers) true bsf-13)
(have (obj-arrows obj-feathers) false bsf-14)
(see-that (obj-karla bsf-14) true bsf-15)
(shoot-at (obj-hunter obj-karla) true bsf-16)
(miss (obj-hunter obj-karla) true bsf-17)
(want (obj-hunter obj-feathers) true bsf-18)
(know-that (obj-karla bsf-18) true bsf-19)
(glide-to (obj-karla obj-hunter) true bsf-20)
(offer-to (obj-karla obj-feathers obj-hunter) true bsf-21)
(accept-from (obj-hunter obj-feathers obj-karla) true bsf-22)
(grateful-to (obj-hunter obj-karla) true bsf-23)
(pledge-that (obj-hunter (bsf-26 false)) true bsf-24)
(hawks (obj-hawks) true bsf-25)
(shoot-at (obj-hunter obj-hawks) false bsf-26)
(deer (obj-deer) true bsf-27)
(shoot (obj-hunter obj-deer) true bsf-28)
(cause (bsf-14 bsf-11) true bsf-29)
(cause (bsf-18 bsf-16) true bsf-30)
(cause (bsf-14 bsf-17) true bsf-31)
(cause (bsf-19 bsf-20) true bsf-32)
(cause (bsf-19 bsf-21) true bsf-33)
(cause (bsf-21 bsf-22) true bsf-34)
(cause (bsf-21 bsf-23) true bsf-35)
(cause (bsf-23 bsf-24) true bsf-36)
(cause (bsf-24 bsf-26) true bsf-37)
(cause (bsf-26 bsf-28) true bsf-38)

Fig. 12. Predicate-calculus translation of "Karla the Hawk" probe.

four alternative stories listed in Fig. 11, and were asked to write down any story they were reminded of by the probe. (In Gentner's terminology, "target" corresponds to our "probe," and "base" corresponds to our "source.") The four versions manipulated the similarity of the probe and source story. The "literal-similarity" and "mere-appearance" conditions were high in overall semantic similarity, whereas the "true-analogy" and "false-analogy" conditions were low in overall similarity. The literal-similarity and true-analogy conditions preserve the pattern of causal relations in the probe, whereas the mere-appearance and false-analogy conditions did not. Thus the design provides a factorial manipulation of degree of semantic similarity and degree of structural consistency at the level of causal relations.

As in the case of the convergence problems, it is impractical to simulate the actual knowledge of subjects in Rattermann and Gentner's experiment. As a surrogate for general memory, we used the fables data base described in Section 6.1. Because many of the 100 fables concern animals, as do the Rattermann and Gentner stories, they serve as excellent distractors for the probe story. We did four retrieval simulations, using the same parameters as in the simulations involving the tumor problem. The largest network consisted of 570 units, 26,330 links, and did not settle within the 200-cycle limit we imposed. As Table 3 indicates, ARCS' simulation corresponded well with the results of the psychological experiments. Each probe story retrieved the "Karla the Hawk" story, but the retrievability of that story as measured by the number of fables that were better retrieved varied inversely with the number of subjects recalling the story in the Rattermann and Gentner experiment. That is, the greater the percentage of the subjects retrieving the "Karla the Hawk" story using a probe of a particular type, the better the story did in relation to the fables also stored in memory. The ordering of activations of the units shows a small reversal for the true-analogy and false-analogy case relative to ordering of the retrieval percentages obtained by Rattermann and Gentner, but we view

Table 3
Comparisons of ARCS simulation with "Karla the Hawk" experiment (Ratterman and Gentner [48])

	Percentage of subjects retrieving story	Rank of "Karla" unit	Asymptotic activation of "Karla" unit
Literal similarity	58	1 of 68	0.67
Mere appearance	45	9 of 59	−0.17
True analogy	18	9 of 61	−0.27
False analogy	8	16 of 29	−0.11

Note. The percentages are inferred from a graph; the actual numbers were not reported by Ratterman and Gentner. By "Karla" unit we mean the unit representing the correspondence between the probe story and the "Karla the Hawk" story. In the rank column, "1 of 68" means that there were 67 fables activated in addition to the Karla story and the "Karla" unit had the highest activation.

this as an artifact of there being so many more fables activated in the true-analogy case than in the false-analogy case.

The results of Rattermann and Gentner [48] appear to show effects of both semantic similarity and structural consistency. The latter effect reveals itself in the apparent advantage of the literal-similarity condition over the mere-appearance condition, and the advantage of the true-analogy over the false-analogy condition. Although these differences fell short of statistical significance, each of these two trends were replicated in two separate experiments [19, 48], and in one experiment the advantage of the true analogy relative to the false analogy was significant [19]. Note that pragmatic centrality plays no role in the "Karla the Hawk" simulations, since the stories are not problems, explanations, or arguments. We agree with Gentner [17] (in contrast, for example, to Hammond [26]) that semantic similarity is the most important constraint on analog retrieval, although we maintain that structural and pragmatic constraints are also used when such information is available.

6. Computational Evaluation

ARCS is thus consistent with some major psychological experiments on analog retrieval, and therefore gains some credibility as a cognitive model. However, it is also important to do computational experiments to determine what happens in data bases in which the presence of many potential source analogs raises the danger of either retrieving too many not-so-relevant analogs or requiring too much computation to select out the more relevant ones. Do the networks required to do constraint satisfaction explode as the size of the data base increases, rendering the model intractable? We showed in Section 3.3 that the space complexity of ARCS is no worse than $O(k^2n^4)$, but realistic cases provide more encouraging results.

To provide experimental answers to questions concerning how well ARCS scales up, we constructed two large data bases consisting of 100 fables and 24 synopses of Shakespeare's plays. Tests on the first data base show that ARCS is capable of screening the wheat from the chaff in a data base of many complex structures, whereas tests on the second data base show that it is capable of handling even more complex structures: the fables average about 22 propositions per structure whereas the plays average about 55.

6.1. The fables data base

Aesop's fables [1] are a series of engaging stories intended to provide moral and social lessons. Most readers will be familiar with the story of the fox who decides that grapes that he cannot reach are probably too sour to be wanted anyway, illustrating how people sometimes decide that unattainable things are not desirable. The fables contain a wealth of social wisdom and opinion, but our reason for choosing them as a test data base is that they are semantically

rich, structurally complex, and highly interconnected with each other via similar characters and events. In this data base, the problem of selecting relevant analogs while not being swamped by less relevant ones is acute.

Figure 13 presents the text of the fable about sour grapes, while Fig. 14 provides our predicate-calculus representation of it. The fables data base also includes 99 similar structures. The semantic information for all our data bases has WordNet-style entries for over a thousand predicates like FOX and GRAPES. Probing from fable 3, "Sour Grapes," yielded connections to 71 of the 99 stored fables (71%), largely because of the preponderance of animal characters in them; the winner was a fable [1, number 15] that also concerned foxes and eating. To check whether ARCS' performance on this data base was plausible, we in addition created a story about humans that we thought was analogous to the fox and sour grapes story. As expected, probing from this story into the entire data base retrieved the sour grapes fable more than any other.

Sour Grapes

A hungry fox tried to reach some clusters of grapes which he saw hanging from a vine trained on a tree, but they were too high. So he went off and comforted himself by saying: "They weren't ripe anyhow."

In the same way some men, when they fail through their own incapacity, blame circumstances.

Fig. 13. Sample fable "Sour Grapes."

```
Story:
  (fox (obj-fox) true f3-1)
  (grapes (obj-grapes) true f3-2)
  (want (obj-fox obj-grapes) true f3-3)
  (get (obj-fox obj-grapes) false f3-4)
  (decide (obj-fox (f3-6 true)) true f3-5)
  (sour (obj-grapes) false f3-6)
  (cause (f3-4 f3-5) true f3-7)

Moral:
  (men (obj-men) true m3-1)
  (circumstances (conc-circumstances) true m3-2)
  (fail (obj-men) true m3-3)
  (incapable (obj-men) true m3-4)
  (cause (m3-4 m3-3) true m3-5)
  (blame-for (obj-men conc-circumstances m3-3) true m3-6)
```

Fig. 14. Predicate-calculus version of "Sour Grapes" fable.

We tested ARCS' capacity for efficient retrieval with the fables data base by performing a series of computational experiments. The results are shown in Fig. 15(A)–(D), which provides graphs of various performance measures against data bases of varying sizes. We randomly selected 10 fables and used them as probes into data bases of 10, 20, . . . , 99 fables, also randomly selected from the total data base. For each run, we noted the size of the networks created and the settling time. Figure 15 show the results averaged over the 10 fables selected as probes. The graphs show that the numbers of units and links increase manageably as the data base grows. The number of propositions (Fig. 15(A)) and units (Fig. 15(B)) increase linearly with the number of fables, while the number of links (Fig. 15(C)) increases at a rate on the order of the *square* of the number of fables. This increase is far more manageable than the $O(k^2n^4)$ result (for k fables, of maximum n propositions) result in Section 3.3. If the worst-case result obtained generally, parallelism would not be of much help, since a very large number of processors would be required for retrieval of a relatively small number of analogs. Fortunately, ARCS uses semantic similarity information to prune dramatically the set of possible correspondences, resulting in the $O(k^2)$ experimental result for the fables data base.

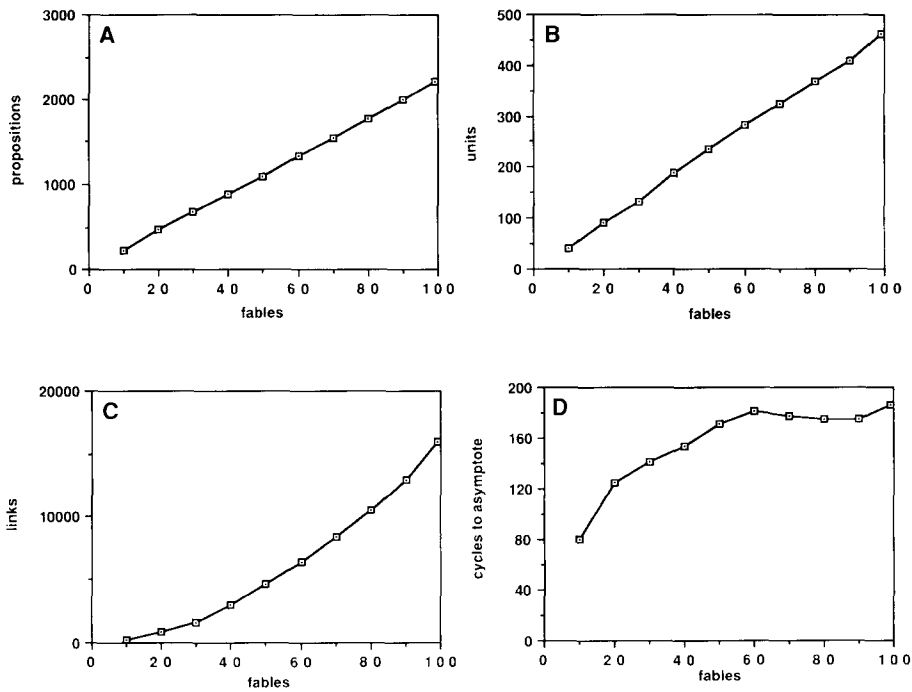

Fig. 15. (A) Number of propositions, (B) number of units, (C) number of links, and (D) number of cycles to asymptotic settling, as a function of number of fables.

Most encouragingly, the number of cycles required for the fable networks to settle tends to level off. Figure 15(D) graphs the average number of cycles to settle in networks produced by probing into data bases of different sizes. Even networks created by probing into the largest data bases tend to settle in under 200 cycles. The maximum number of units created was 587, and the maximum number of links was 25,170.

6.2. The plays data base

Winston [66] tested his ideas about analogy using very complex structures representing synopses of Shakespeare's plays. Although such structures are larger than people can probably handle in an entirely parallel manner, they provide a useful test of the computational power of a model of retrieval. Probing from a play into a data base of plays requires the comparison of a very large number of characters and elements. Our plays data base includes structures for 24 synopses based largely on summaries of the plays taken from Magill [41]. The plays averaged about 55 propositions per play, and every predicate was provided with WordNet-style semantic information. Figure 16 presents our predicate-calculus version of the synopsis of the play *Hamlet*.

Because the plays are such large structures, and because they overlap with each other a great deal, the networks produced using this data base are large. Probing from *Hamlet* into a data base of the remaining 23 plays produces a network with 1,160 units and 67,140 links. This exceeds the rate of growth we found for links versus the size of the fables data base, because each play is on average twice the size of each fable. It is less important, however, that a model of human analog retrieval scale well for size of structure than for size of data base, since it is more plausible that human memory contains a very great number of cases than that these cases are very large.

Figure 17 shows the concurrently produced activation graphs of the units representing which play is most retrievable from *Hamlet*. The graph over 200 cycles shows the stability of the network. With the default values for excitation and inhibition, the network settles in 268 cycles. Notice that six tragedies, *Romeo and Juliet*, *King Lear*, *Othello*, *Cymbeline*, *Macbeth*, and *Julius Caesar*, were judged to be more similar to *Hamlet* than the other plays, most of which had units with activation values below 0. As a further check on the plausibility of ARCS' performance on this data base, we formalized a plot outline of *West Side Story*, expecting that this structure would retrieve *Romeo and Juliet*, which it does.

6.3. Sensitivity analyses

How sensitive is ARCS' performance to changes in representation and parameter values? Our structure representations were constructed independently of the tests described above, using predicate calculus, WordNet-style semantics,

and a preponderance of semantic entries drawn from WordNet. Performance does not depend on the internal organization of analogs, for example of problems into starting conditions and goals: ARCS uses such information only for marking some propositions and concepts as pragmatically important. The common representation scheme for all data bases makes it possible to do experiments on amalgamated data bases. When the tumor problem is used as a probe into a data base consisting of *both* convergence problems and "Karla the Hawk" stories, it turns out that the stories are candidates for retrieval along with the problems. But the stories are quickly rejected as plausible analogs of the tumor problem and the ordering of activation of the radiation problems remains as described in Section 5.1.

All of the runs described above, on all the data bases, used the default parameter values of 0.01 for excitation, 0.04 for decay, and −0.04 for inhibition. The default values were selected because they tend to lead to networks settling in fewer than 100 cycles; other parameter values increased settling time to as many as 150 cycles. We have conducted systematic sensitivity tests on the Holyoak and Koh data base, and found that ARCS obtains the desired results for a reasonable range of parameter values. Decay values between 0.01 and 0.05 work fine, except that at the higher levels the final resting values of the units approach 0, so that there is not as much spread between the activations of the units representing the retrievability of the different structures. The orderings, however, are preserved. Higher levels of decay significantly reduce the settling time, with the Holyoak and Koh experiment settling in fewer than 30 cycles at high (>0.1) decay values. Holding inhibition constant at the default value, we varied excitation and determined that acceptable performance resulted from excitation values within the range from 0.007 to 0.0145. The lower values tended, however, to reduce

Characters:
(king (obj-old-hamlet) true hac-1)
(prince (obj-hamlet) true hac-2)
(queen (obj-gertrude) true hac-3)
(man (obj-claudius) true hac-4)
(man (obj-polonius) true hac-5)
(man (obj-laertes) true hac-6)
(woman (obj-ophelia) true hac-7)
(daughter (obj-ophelia obj-polonius) true hac-8)
(son (obj-laertes obj-polonius) true hac-9)
(siblings (obj-old-hamlet obj-claudius) true hac-10)
(son (obj-hamlet obj-old-hamlet) true hac-11)

Fig. 16. Predicate-calculus formalization of Hamlet synopsis.

Plot:
(kill (obj-claudius obj-old-hamlet) true ha-1)
(marry (obj-claudius obj-gertrude) true ha-2)
(conjoin-event (ha-1 ha-2) true ha-3)
(king (obj-claudius) true ha-4)
(cause (ha-3 ha-4) true ha-5)
(upset (obj-hamlet) true ha-6)
(cause (ha-3 ha-6) true ha-7)
(distress (obj-hamlet obj-claudius) true ha-8)
(dead (obj-hamlet) unknown ha-9)
(desire (obj-claudius (ha-9 true)) true ha-10)
(cause (ha-8 ha-10) true ha-11)
(curtain (obj-arras) true ha-12)
(behind (obj-polonius obj-arras) true ha-13)
(behind (obj-claudius obj-arras) false ha-14)
(believe (obj-hamlet (ha-14 true)) true ha-15)
(kill (obj-hamlet obj-polonius) true ha-16)
(cause (ha-6 ha-15) true ha-17)
(cause (ha-15 ha-16) true ha-18)
(upset (obj-ophelia) true ha-19)
(cause (ha-16 ha-19) true ha-20)
(kill (obj-ophelia obj-ophelia) true ha-21)
(cause (ha-19 ha-21) true ha-22)
(upset (obj-laertes) true ha-23)
(cause (ha-21 ha-23) true ha-24)
(desire (obj-laertes (ha-9 true)) true ha-25)
(cause (ha-23 ha-25) true ha-26)
(conjoin-event (ha-10 ha-25) true ha-27)
(duel (obj-hamlet obj-laertes) true ha-28)
(cause (ha-27 ha-28) true ha-29)
(kill (obj-hamlet obj-laertes) true ha-30)
(kill (obj-laertes obj-hamlet) true ha-31)
(conjoin-event (ha-30 ha-31) true ha-32)
(cause (ha-28 ha-32) true ha-33)
(kill (obj-claudius obj-gertrude) true ha-34)
(accident (ha-34) true ha-35)
(cause (ha-10 ha-34) true ha-36)
(kill (obj-hamlet obj-claudius) true ha-37)
(conjoin-event (ha-6 ha-10) true ha-38)
(conjoin-event (ha-38 ha-34) true ha-39)
(cause (ha-39 ha-37) true ha-40)

Fig. 16. *Continued.*

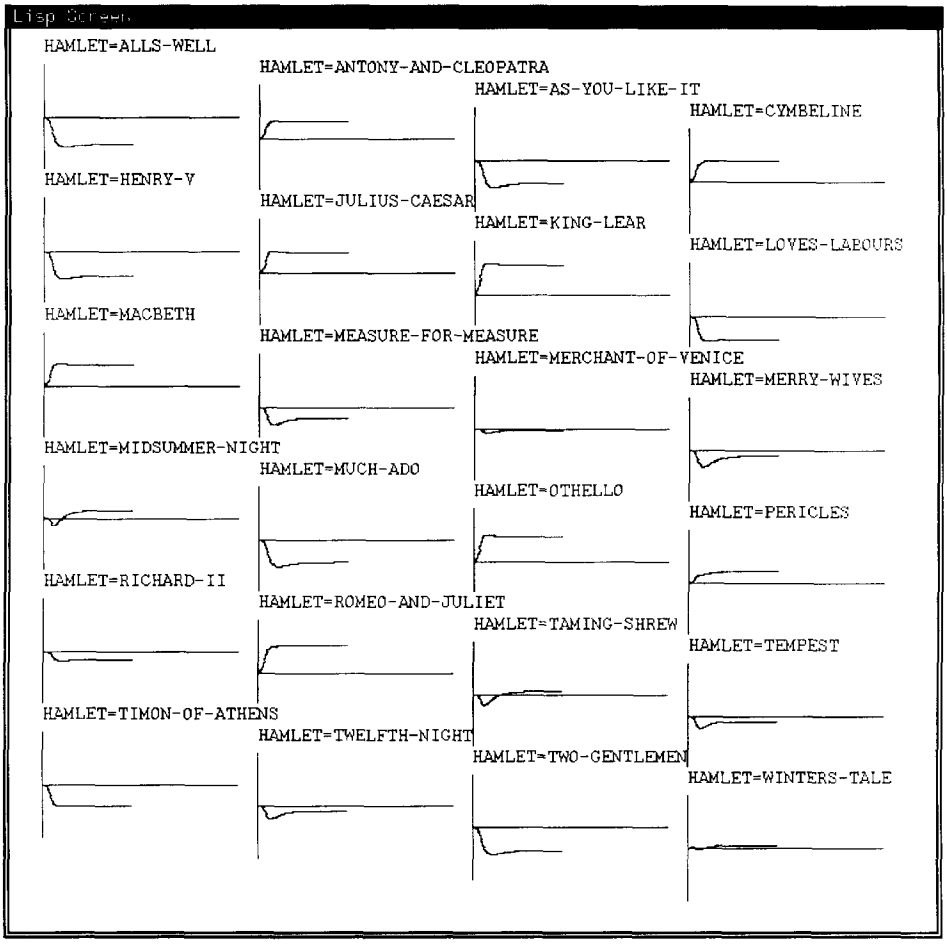

Fig. 17. Activation histories of selected units in "plays" simulation. Each graph shows activation on a scale of 1 to −1, with the horizontal line indicating the starting activation of 0.

the spread among the final activation levels of competing units. More important, when excitation values exceeded 0.0145 the networks became unstable with units undergoing oscillations that delayed or precluded settling. Holding excitation constant at 0.01, we varied inhibition systematically and found that performance was maintained for the range of −0.016 to −0.2.

In testing whether ARCS performance is sensitive to the values for semantic similarity stated in Fig. 5(B), we found that the parameter values could all be made the same without interfering with the simulation of the Holyoak and Koh [33] data, except for the parameter for antonyms which could vary but had to remain negative. Otherwise, ARCS found a mapping of HIGH to LOW as good as

a mapping from HIGH to HIGH, preventing it from distinguishing between the fragile-lightbulb and the insufficient-intensity versions of the problems.

Since graceful degradation is often cited as a strength of connectionist networks, we tested ARCS' immunity to deletion of randomly chosen units. Deletion of a few units from networks for the Holyoak and Koh simulation caused little problem, but in experiments where 25% of the units were randomly deleted, the results failed to fit the psychological data 75% of the time. Random deletion of percentages of concepts from permanent memory has a more severe effect on performance: on average, deleting 10% of the concepts eliminated the psychological fit 60% of the time.

Finally, we did selective sensitivity tests on the fable data base, probing from the sour grapes fable using extreme values for excitation and inhibition, and found that performance was fine over roughly the same ranges of values as for the two much smaller data bases. In particular, the same fable was favored in each case. The fable runs take just over two minutes on a Sun 4 workstation.

6.4. Connection machine implementation

Because the algorithm used by ARCS for updating the network is fully parallel, it can theoretically operate in constant time if a processor is assigned to each unit and link. To begin to take advantage of this inherent parallelism, a version of ARCS has been implemented in *LISP on a 16384-processor CM2 Connection Machine. To implement the settling algorithm, CM2 processors are divided into two sets, corresponding to units and links, respectively. Each processor representing a unit contains slots for the unit's current activation, its activation on the prior cycle, and for the net excitatory and net inhibitory inputs to that unit. Each processor representing a link contains slots for the link's weight, for the processor index corresponding to the unit on the source end of the link, and for the processor index corresponding to the unit on the receiving end of the link. When the number of units and links exceeds the number of physical processors on the CM2, the memory associated with each processor is divided to create multiple "virtual processors" representing several units or links. On each cycle, each processor representing a link operates in parallel to calculate and transmit its input to the processor representing the receiving unit, where all inputs are summed; all units then update their activations in parallel. For the largest network so far tested (probing with *Hamlet* into the plays data base), the CM2 version of ARCS runs at a speed of approximately 1.6 seconds per cycle, a rate over nine times faster than that obtained with the serial version running on a Sun 4.

7. Conclusion

We conclude on the basis of the above evaluation that ARCS provides a psychologically plausible and computationally powerful model of memory for

analogs. People retrieve complex structures by simultaneously applying semantic, structural, and pragmatic constraints, with the semantic constraints being most crucial for initiating the probe process.

ARCS is only part of a full model of analogy. Retrieval should blend naturally into mapping, so that when the retrieval system has selected a possible analog, the mapping system can go to work to determine in more detail how well the two structures correspond. Our mapping program ACME [35] uses similar principles to ARCS, and preliminary tests show that passing ARCS results directly to ACME greatly facilitates ACME's ability to fill in a complete mapping. ACME is not so dependent on semantic similarity as ARCS and can fill in the mappings of elements with no semantic overlap.

After an analogy system has performed retrieval and mapping, it still needs to transfer the results of the analogy for the appropriate problem-solving, explanatory, argumentative, or evocative purpose. At this point, the goals of the system should be more important than anything else, although semantics and structural consistency may also contribute. Thus in our view all three constraints—semantic, structural, *and* pragmatic—are important to all three stages of analogy: retrieval, mapping, and transfer. As Table 4 summarizes, however, the importance of the different constraints to the different stages is hypothesized to vary, with semantics paramount for retrieval, structural constraints paramount for mapping, and pragmatics paramount for transfer.

A fourth stage of analogy use is learning: if an analogy proves to be useful, various strategies can be applied to learn from the success (e.g., by schematizing the two analogs and by forming rules about how to use such schemas). Our earlier system PI [34] has such capabilities. Another attractive feature of PI is that it does both rule-based and analogical problem solving within the same cognitive architecture. We see no point in making a sharp contrast between rule-based and analogical (case-based) problem solving, since a system that approximates the power of human performance should gracefully incorporate both mechanisms. We are currently developing a cognitive architecture that

Table 4
Summary of hypothesized importance of different constraints to different stages of analogical thinking

	Semantic	Isomorphism	Pragmatic
Retrieval (ARCS)	Very	Yes	Yes
Mapping (ACME)	Yes	Very	Yes
Transfer and learning	Yes	Yes	Very

Note. Here Yes means that the constraint is important, and Very means that the constraint is very important.

integrates the constraint-satisfaction methods of ARCS and ACME with a rule-based problem solver. The amalgamation of a rule-based problem solver and our constraint-satisfaction programs reflects our view that connectionist and traditional AI approaches belong to a continuum of complementary computational methods.

Thus ARCS is only one component of a much larger system for analogical thinking. The system demonstrates, however, that an important part of that system, retrieval of relevant analogs from memory, can naturally be understood in terms of the satisfaction of multiple semantic, structural, and pragmatic constraints.

ACKNOWLEDGEMENT

The research reported in this paper was supported by Contract MDA903-86-0297 from the Basic Research Office of the Army Research Institute for the Behavioral and Social Sciences, and by a grant from the McDonnell foundation to Princeton University. The Connection Machine implementation of the program was supported by NSF Biological Facilities Award BBS 87-1420, and by a grant from the Keck Foundation. We are grateful to George Miller for providing access to WordNet and to Brian Reiser, Jerry Faries, and Michael Ranney for helpful conversations. Eric Melz reimplemented the program in *LISP and performed the Connection Machine runs. We thank two anonymous reviewers for extensive suggestions that greatly improved the exposition.

REFERENCES

1. Aesop, *Fables of Aesop*, translated by S. Handford (Penguin, Harmondsworth, 1954).
2. J.R. Anderson, Retrieval of propositional information from long-term memory, *Cogn. Psychol.* **5** (1974) 451–474.
3. J.R. Anderson, *The Architecture of Cognition* (Harvard University Press, Cambridge, MA, 1983).
4. J.R. Anderson and R. Thompson, Use of analogy in a production system architecture, in: S. Vosniadou and A. Ortony, eds., *Similarity and Analogical Reasoning* (Cambridge University Press, Cambridge, 1989) 267–297.
5. J.M. Barnes and B.J. Underwood, 'Fate' of first-list associations in transfer theory, *J. Experimental Psychol.* **58** (1959) 97–105.
6. J.G. Carbonell, Learning by analogy: Formulating and generalizing plans from past experience, in: R.S. Michalski, J.G. Carbonell and T.M. Mitchell, eds., *Machine Learning: An Artificial Intelligence Approach* (Tioga, Palo Alto, CA, 1983) 137–161.
7. J.G. Carbonell, Derivational analogy: A theory of reconstructive problem solving and expertise acquisition, in: R.S. Michalski, J.G. Carbonell and T.M. Mitchell, eds., *Machine Learning: An Artifical Intelligence Approach* **2** (Morgan Kaufman, Los Altos, CA, 1986) 371–392.
8. R. Catrambone and K.J. Holyoak, Overcoming contextual limitations on problem-solving transfer, *J. Experimental Psychol. Learning Memory Cognition* **15** (1990) 1147–1156.
9. G.W. Cottrell, Connectionist parsing, in: *Proceedings Seventh Annual Conference of the Cognitive Science Society*, Irvine, CA (1985) 201–211.
10. D. Cruse, *Lexical Semantics* (Cambridge University Press, Cambridge, 1986).
11. K. Duncker, On problem solving, *Psychol. Monographs* **58** (No. 270) (1945).
12. B. Falkenhainer, K.D. Forbus and D. Gentner, The structure-mapping engine, in: *Proceedings AAAI-86*, Philadelphia, PA (1986).

13. B. Falkenhainer, K.D. Forbus and D. Gentner, The structure-mapping engine: Algorithms and examples, *Artif. Intell.* **41** (1989/90) 1–63.
14. J. Faries and B. Reiser, Access and use of previous solutions in a problem solving situation, in: *Proceedings Tenth Annual Conference of the Cognitive Science Society*, Montréal, Qué. (1988) 433–439.
15. J. Feldman and D. Ballard, Connectionist models and their properties, *Cogn. Sci.* **6** (1982) 205–254.
16. D. Gentner, Structure-mapping: A theoretical framework for analogy, *Cogn. Sci.* **7** (1983) 155–170.
17. D. Gentner, The mechanisms of analogical learning, in: S. Vosniadou and A. Ortony, eds., *Similarity and Analogical Reasoning* (Cambridge University Press, Cambridge, 1989) 199–241.
18. D. Gentner and D.R. Gentner, Flowing waters or teeming crowds: Mental models of electricity, in: D. Gentner and A.L. Stevens, eds., *Mental Models* (Erlbaum, Hillsdale, NJ, 1983) 99–129.
19. D. Gentner and R. Landers, Analogical reminding: A good match is hard to find, in: *Proceedings International Conference on Systems, Man, and Cybernetics*, Tucson, AZ (1985).
20. D. Gentner and C. Toupin, Systematicity and surface similarity in the development of analogy, *Cogn. Sci.* **10** (1986) 277–300.
21. M.L. Gick and K.J. Holyoak, Analogical problem solving, *Cogn. Psychol.* **12** (1980) 306–355.
22. M.L. Gick and K.J. Holyoak, Schema induction and analogical transfer, *Cogn. Psychol.* **15** (1983) 1–38.
23. T. Gilovich, Seeing the past in the present: The effect of associations to familar events on judgments and decisions, *J. Personality Social Psychol.* **40** (1981) 797–808.
24. S. Grossberg, A theory of visual coding, memory, and development, in: E. Leeuwenberg and J. Buffart, eds., *Formal Theories of Visual Perception* (Wiley, New York, 1978).
25. R. Hall, Computational approaches to analogical reasoning: A comparative analysis, *Artif. Intell.* **39** (1989) 39–120.
26. K. Hammond, The use of remindings in planning, in: *Proceedings Eighth Annual Conference of the Cognitive Science Society*, Amherst, MA (1986) 442–451.
27. J. Hendler, Marker-passing over microfeatures: Towards a hybrid symbolic/connectionist model, *Cogn. Sci.* **13** (1989) 79–106.
28. G.E. Hinton and J.R. Anderson, eds., *Parallel Models of Associative Memory* (Erlbaum, Hillsdale, NJ, 1981).
29. D. Hofstadter, The Copycat project: An experiment in nondeterminism and creative analogies, AI Memo 755, MIT Artificial Intelligence Laboratory, Cambridge, MA (1984).
30. D. Hofstadter and M. Mitchell, Conceptual slippage and analogy making: A report on the Copycat project, in: *Proceedings Tenth Annual Conference of the Cognitive Science Society*, Montréal, Qué. (1988) 601–607.
31. J. Holland, K.J. Holyoak, R. Nisbett and P. Thagard, *Induction: Processes of Inference, Learning, and Discovery* (MIT Press/Bradford Books, Cambridge, MA, 1986).
32. K.J. Holyoak, The pragmatics of analogical transfer, in: G.H. Bower, ed., *The Psychology of Learning and Motivation* **19** (Academic Press, New York, 1985).
33. K.J. Holyoak and K. Koh, Surface and structural similarity in analogical transfer, *Memory Cogn.* **15** (1987) 332–340.
34. K.J. Holyoak and P. Thagard, A computational model of analogical problem solving, in: S. Vosniadou and A. Ortony, eds., *Similarity and Analogical Reasoning* (Cambridge University Press, Cambridge, 1989) 242–266.
35. K.J. Holyoak and P. Thagard, Analogical mapping by constraint satisfaction, *Cogn. Sci.* **13** (1989) 295–355.
36. W. Kintsch, The role of knowledge in discourse comprehension: A construction-integration model, *Psychol. Rev.* **95** (1988) 163–182.

37. J. Kolodner, Maintaining organization in a dynamic long-term memory, *Cogn. Sci.* **7** (1983) 243–280.
38. J. Kolodner, Selecting the best case for a case-based reasoner, in: *Proceedings Eleventh Annual Conference of the Cognitive Science Society*, Ann Arbor, MI (1989) 155–162.
39. J. Kolodner and R. Simpson, The MEDIATOR: A case study of a case-based problem solver. Tech. Rep. GIT-ICS-88/11, School of Information and Computer Science, Georgia Institute of Technology, Atlanta, GA (1988).
40. W. Lehnert, Case-based problem solving with a large knowledge base of learned cases, in: *Proceedings AAAI-87*, Seattle, WA (1987) 301–306.
41. F. Magill, *Masterplots*, Revised Edition, edited by Frank N. Magill (Salem Press, Englewood Cliffs, NJ, 1976).
42. J. McClelland and D. Rumelhart, Distributed memory and the representation of general and specific information, *J. Experimental Psychol. General* **114** (1985) 159–188.
43. G.A. Miller, C. Fellbaum, J. Kegl and K. Miller, WORDNET: An electronic lexical reference system based on theories of lexical memory, *Revue Québécoise Linguistique* **17** (1988) 181–213.
44. G. Miller and P. Johnson-Laird, *Language and Perception* (Harvard University Press, Cambridge, MA, 1976).
45. S.E. Palmer, Levels of description in information processing theories of analogy, in: S. Vosniadou and A. Ortony, eds., *Similarity and Analogical Reasoning* (Cambridge University Press, 1989) 332–345.
46. P.L. Pirolli and J.R. Anderson, The role of learning from examples in the acquisition of recursive programming skills, *Can. J. Psychol.* **39** (1985) 240–272.
47. R. Ratcliff and G. McKoon, Similarity information versus relational information: Differences in the time course of retrieval, *Cogn. Psychol.* **21** (1989) 139–155.
48. M. Rattermann and D. Gentner, Analogy and similarity: determinants of accessibility and inferential soundness, in: *Proceedings Ninth Annual Meeting of the Cognitive Science Society*, Seattle, WA (1987) 23–35.
49. *Roget's International Thesaurus* (Harper and Row, New York, 4th ed, 1977).
50. B. Ross, Remindings and their effects in learning a cognitive skill, *Cogn. Psychol.* **16** (1984) 371–416.
51. B. Ross, This is like that: The use of earlier problems and the separation of similarity effects, *J. Experimental Psychol. Learning Memory Cognition* **13** (1987) 371–416.
52. B. Ross, Distinguishing types of superficial similarities: Different effects on the access and use of earlier problems, *J. Experimental Psychol. Learning Memory Cognition* **15** (1989) 456–468.
53. D. Rumelhart, Toward a microstructural account of human reasoning, in: S. Vosniadou and A. Ortony, eds., *Similarity and Analogical Reasoning* (Cambridge University Press, Cambridge, 1989) 298–312.
54. D. Rumelhart, J. McClelland and the PDP Research Group, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, 2 volumes (MIT Press, Cambridge, MA, 1986).
55. R.C. Schank, *Dynamic Memory* (Cambridge University Press, Cambridge, 1982).
56. C. Seifert, G. McKoon, R. Abelson and R. Ratcliff, Memory connections between thematically similar episodes, *J. Experimental Psychol. Learning Memory Cognition* **12** (1986) 220–231.
57. L. Shastri, A connectionist approach to knowledge representation and limited inference, *Cogn. Sci.* **12** (1988) 331–392.
58. R.M. Spencer and R.W. Weisberg, Context-dependent effects on analogical transfer, *Memory Cogn.* **14** (1986) 442–449.
59. C. Stanfill and D. Waltz, Toward memory-based reasoning, *Commun. ACM* **29** (1986) 1213–1228.

60. P. Thagard, *Computational Philosophy of Science* (MIT Press/Bradford Books, Cambridge, MA, 1988).
61. P. Thagard, Explanatory coherence, *Behav. Brain Sci.* **12** (1989) 435–467.
62. P. Thagard, D. Cohen and K.J. Holyoak, Chemical analogies: Two kinds of explanation, in: *Proceedings IJCAI-89*, Detroit, MI (1989) 819–824.
63. P. Thagard and K.J. Holyoak, Discovering the wave theory of sound: Induction in the context of problem solving, in: *Proceedings IJCAI-85*, Los Angeles, CA (1985) 610–612.
64. E. Tulving, Episodic and semantic memory, in: E. Tulving and W. Donaldson, eds., *Organization and Memory* (Academic Press, New York, 1972).
65. *Webster's Ninth New Collegiate Dictionary* (Merriam-Webster, Springfield, MA, 1986).
66. P.H. Winston, Learning and reasoning by analogy, *Commun. ACM* **23** (1980) 689–703.
67. P.H. Winston, Learning new principles from precedents and exercises, *Artif. Intell.* **19** (1982) 321–350.

Received February 1989; revised version received January 1990