RETRIEVING SEMANTICALLY DISTANT ANALOGIES

A DISSERTATION

SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE

AND THE COMMITTEE ON GRADUATE STUDIES

OF STANFORD UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

By

Michael Wolverton

May 1994

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Barbara Hayes-Roth
(Principal Adviser)

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Edward Feigenbaum

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Raymond Levitt

Approved for the University Committee on Graduate Studies:

Abstract

Techniques that have traditionally been useful for retrieving same-domain analogies from small single-use knowledge bases, such as spreading activation and indexing on selected features, are inadequate for retrieving cross-domain analogies from large multi-use knowledge bases. Blind or near-blind search techniques like spreading activation will be overwhelmed by combinatorial explosion as the search goes deeper into the KB. And indexing a large multi-use KB on salient features is impractical, largely because a feature that may be useful for retrieval in one task may be useless for another task. This thesis describes Knowledge-Directed Spreading Activation (KDSA), a method for retrieving analogies in a large semantic network. KDSA uses task-specific knowledge to guide a spreading activation search to a case or concept in memory that meets a desired similarity condition. The thesis also describes a specific instantiation of this method for the task of innovative design.

KDSA has been validated in two ways. First, a theoretical model of knowledge base search demonstrates that KDSA is tractable for retrieving semantically distant analogies under a wide range of knowledge base configurations. Second, an implemented system that uses KDSA to find analogies for innovative design shows that the method is able to retrieve semantically distant analogies for a real task. Experiments with that system show trends as the knowledge base size grows that suggest the theoretical model's prediction of large knowledge base tractability is accurate.

Acknowledgements

First, I would like to thank my advisor, Barbara Hayes-Roth, for all of her support and guidance during my tenure at Stanford. The other members of my reading committee, Ed Feigenbaum and Ray Levitt, provided many useful ideas that contributed to this research. Richard Fikes deserves mention for raising several interesting issues during my thesis defense that ultimately strengthened my understanding of analogy retrieval and improved this thesis. Thanks also to the people who helped me in building my knowledge base of devices—Ray Levitt, Serdar Uckun, Yumi Iwasaki, and Pandu Nayak.

Many people at the Knowledge Systems Lab contributed to this thesis in one way or another. My officemate Rich Washington has been a great colleague and friend throughout my entire stay at Stanford. Whether I needed some new ideas on analogy mapping, a detailed explanation of the Lisp EVAL-WHEN special form, or a diagnosis of my Buick's latest problems, Rich was always there with the right answer. The other members of the BB1 group—especially Janet Murdock, Lee Brownston, David Ash, Philippe Lalanda, Philippe Morignot, John Drakopoulos, Serdar Uckun, and Vlad Dabija—have given me much good advice on my work and on surviving grad school in general. The KSL administrative team—Grace Smith, Michelle Perrie, Peche Turner, and Margaret Timothy—kept the lab running smoothly and made dealing with Stanford red tape almost effortless.

Life at Stanford has been stressful at times, and I might not have maintained my sanity here without great friends in the "real world". My participation with the San Francisco Symphony Chorus has been a wonderful experience, and the friends I've made there—Vance George, Ron Gallman, Greg Cheng, Jody Black, Elizabeth Warden—have really enriched my life while I've been in grad school. Making music with them provided a much-needed source of affirmation at times when research was difficult. Other friends, especially Darrell Vaughn and Frank McDonald, gave me lots of encouragement and good advice during various stages of my grad school career. And thanks to Eric Berglund for sharing a pearl of wisdom that kept me pushing for the degree at the end.

My family has been a terrific source of support. Any success I've had in school over the years is due almost entirely to my parents, Betty and Byron Wolverton, who taught me the right priorities and have always encouraged me to learn. I am thankful to have had my brother Christopher in the Bay Area for most of my time at Stanford. He was able to commiserate with me over the latest Ph.D. program issues, and was also able to help me take my mind off of grad school.

Finally and most importantly, I want to thank my wife Cindy for her love and support, and for putting up with me and my life as a grad student. To her I promise that someday, I *will* get a real job.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Cross-domain analogy is a commonly-used reasoning device, especially among individuals who must exhibit a high level of creativity in their reasoning. Authors, journalists, and political speechwriters often use surprising metaphors in order to enliven their prose; clever teachers use analogies to familiar concepts outside of the domain of discussion to explain unfamiliar concepts; and engineers and inventors often use analogies in order to help them produce a novel design. The concern of this thesis is the retrieval and use of cross-domain analogies, specifically those in which the two analogues are *semantically distant* from one another—that is, they are very different from one another in all but a few key features.

The literature on invention is full of examples of inventions that were guided by semantically distant analogies. Gutenberg invented the printing press after noticing the connection between applying force to impress script on paper and applying force to squeeze grapes in a wine press [Koestler, 1965] . Edison's invention of the quadruplex telegraph was based almost entirely on an analogy to a water system of pumps, pipes, valves, and water wheels [Hughes, 1971]. And actress Hedy Lamarr conceived of a method for coordinating frequencies between sender and receiver in frequency-hopping communication by analogy to a player-piano roll [Simon *et al.*, 1985]. These examples all show the inventor making a connection between two concepts not normally thought of as connected. This type of analogy is by no means the only reasoning method used in the invention process. Many inventions involve no analogies at all. But there is evidence that it is an important technique for many inventors and for other types of advanced human reasoning. This suggests that semantically distant analogy can be an important technique for computer reasoning as well.

From looking at examples of analogies in invention, we can surmise a number of characteristics of semantically distant analogies that present special problems for the development of a computational model:

(1) The domains from which the analogies are drawn are unpredictable. The concepts used to guide novel designs come from a wide range of domains, and it is impossible to predict, given the target design domain, which base domain(s) may prove fruitful for drawing useful analogies. At least one researcher has identified having a wide range of interdisciplinary knowledge as a prerequisite of a good inventor [Kock, 1978].

(2) In the analogies that are made, differences between the analogous concepts are as important as similarities. An inventor's chances of developing a truly novel design by analogy are greatly increased by using a base concept which is unusual or unexpected. This suggests that the base concept used should be as different as possible from the target concept while still being useful for design. That is, the two concepts should share only those features which are necessary to the function of the invention, *and should mismatch on as many extraneous features as possible.* In particular, analogies with a high degree of surface similarity seem unlikely to be useful in producing novel inventions.

(3) Analogous concepts are retrieved in a variety of ways. Some inventors seem to find far-flung analogies through a conscious or unconscious search of their memory. Others more or less "stumble across a solution", noticing a connection between something they encounter in normal activities and the design problem they are working on. Still others encounter or discover an interesting phenomenon, and search for a problem which could be solved by applying this phenomenon to it.

Characteristics (1) and (2) above provide reasons that existing approaches to analogy retrieval are inappropriate for retrieving semantically distant analogies. Most existing approaches to analogy retrieval are based either on task-specific indexing of concepts in a case library or on spreading activation in a semantic network, but neither of these general approaches is well-suited for finding semantically distant analogies. The indexing approach is inappropriate because characteristic (1) above suggests that a successful "case library" for semantically distant analogies would in fact be a large multi-domain multi-use knowledge base, but most successful indices in case-based reasoning are task-specific. To create a new set of indices for each possible task which may be performed in such a KB (and each possible analogical use of a given concept) would require a prohibitive number of organizational links or constructs. The spreading activation approach is inappropriate because characteristic (2) above suggests that most corresponding features involved in the analogues will be far from each other in the semantic network, and an uncontrolled spread of activation throughout the large semantic net will bog down in combinatorial explosion before reaching the semantically distant base concepts it seeks. Characteristic (3) above gives an additional constraint that works against indexing. It suggests that a retrieval mechanism for invention should be flexible, able to retrieve a solution given a problem or a problem given a solution, and able to incorporate sensory information representing an inventor's encounters in normal life; most case retrieval methods are not designed to meet these flexibility requirements.

This thesis describes a method, called *knowledge-directed spreading activation* (KDSA), for retrieving semantically distant analogous concepts from a large diverse knowledge base. This method is based on controlled search in a general semantic network. It uses task-specific knowledge to guide a series of spreading activation searches from the target concept to a semantically distant base concept. This knowledge is applied in the evaluation of intermediate concepts retrieved by a standard spread of activation, and by the modification of weights controlling the spread of activation based on those evaluations. The next section describes this method in more detail.

1.1 Knowledge-directed Spreading Activation

Viewed abstractly, KDSA is an application of general techniques from state-space search (evaluation functions, subgoaling, etc.) to knowledge base search. KDSA finds analogues by a *series* of heuristically-guided spreading activation searches. Each time spreading activation retrieves a concept from the knowledge base, the concept is evaluated as an analogue, and that evaluation is used to direct the next spreading activation search in more promising directions. KDSA uses promising concepts retrieved during these spreading activation searches as "beacons", guiding the search successively closer to a semantically distant base.

This description of KDSA will assume that all world knowledge is represented in a single semantic network. Within that semantic network, small subgraphs of nodes and links which represent aggregate concepts are explicitly grouped together as conceptual graphs [Sowa, 1984][1]. Individual conceptual graphs are treated the same as primitive nodes—i.e., they can be associated with other nodes via links, and they can themselves be parts of larger conceptual graphs. In the discussion below, conceptual graphs will be referred to merely as "concepts".

The basic algorithm of KDSA is shown in Figure 1.1. The low-level search of memory (step 2 in the figure) is conducted by a spreading activation mechanism (see, e.g., [Anderson, 1983]). In this formalism, activation is passed from node to adjacent node via the links that connect them until one concept accumulates enough aggregate activation to be considered retrieved. This basic spreading activation model is a blind knowledge search mechanism. Some method of controlling the search is necessary for the system to retrieve the types of semantically distant base concepts described in the introduction. Anderson and others (while not concentrating on retrieving semantically distant analogies) have used priming methods, where the activation-passing strengths on links are increased each time they are used, to cause the mechanism to prefer some paths over others. KDSA, by contrast, uses feedback from the analogues retrieved so far to focus the search.

[1]The present implementation and discussion use this conceptual graph representation of concepts because of its representational power. However, the general model presented here is also applicable to other representational frameworks.

1. Assign activation to all nodes in the target.
2. Spread activation in semantic network until a new intermediate concept (IC) is retrieved.
3. (GRAPH MATCHER) Find the best mapping between target and IC based only on maximizing isomorphism and minimizing semantic distance between nodes.
4. (MATCH EVALUATION) Evaluate the mapping according to domain-specific similarity metric. If evaluation meets the metric, return IC as base and exit.
5. (SEARCH CONTROL) Based on evaluation, alter the state of the semantic network to guide the next phase of spreading activation in a more promising direction.
6. Go to 2.

Figure 1.1: Knowledge-Directed Spreading Activation

The agent architecture encompassing KDSA begins the retrieval process when some executing task requests an analogy and designates a target concept. This initial request causes some nodes in the semantic network—those representing the target concept plus possibly others representing desired features of the solution, etc.—to be assigned activation, and this assignment begins the spread of activation in memory. When a concept is retrieved by the spread of activation, the *graph matcher* computes a mapping between it and the target concept. The *match evaluation component* then forms an evaluation of the mapping based on a task-specific similarity metric. This evaluation is passed on to the *search control component*, which uses its task-specific heuristics to focus the spreading activation search in directions that are more likely to lead to highly-evaluated analogies for the current task. The process repeats until an analogue which meets the matching component's similarity metric is retrieved.

For simplicity, KDSA has been described so far as a strictly serial algorithm. In fact, it is designed (and implemented) as a collection of independent knowledge sources which execute within a larger intelligent agent architecture[2], and which interact with the agent's other activities. Figure 1.2 shows this interaction. At any time during the cycle of Figure 1.1, other concepts may be activated by the agent's other activities, such as ordinary problem solving or processing sensory input. In this way KDSA can account for an individual

[2] In the computer implementation of KDSA, the agent architecture used was BB1 [Hayes-Roth, 1990].

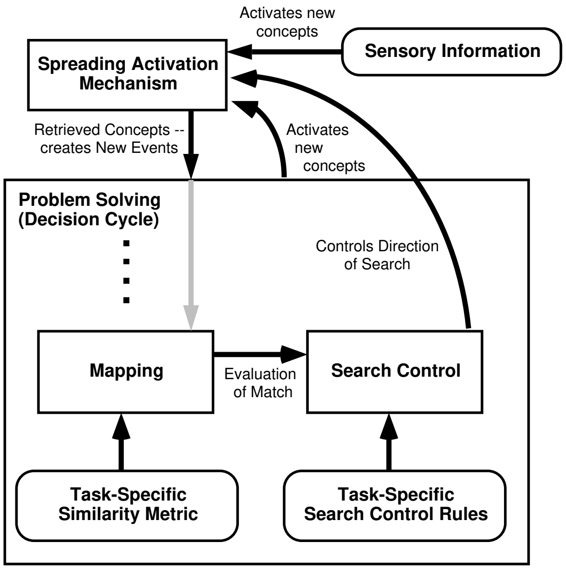


Figure 1.2: Integration of KDSA into problem-solving architecture

possibly “stumbling across a solution”, i.e., being reminded of an analogue by external or internal cues.

The important components of the retrieval system are discussed in more detail below.

Match Evaluation Each time a concept is retrieved by the spreading activation search as a potential base concept, it is passed to the matching component. The matching component first forms the best possible partial mapping between the potential base and the target, and then it evaluates that partial mapping using heuristics which are specific to the task for which the analogy will be used. These heuristics will base their evaluation on three features of the partial mapping: (1) semantic distance between corresponding nodes in the mapping, i.e., the minimum path distance in the type hierarchy between corresponding nodes of the mapping (2) isomorphism between the graphs, i.e., how many nodes and links match between the target and potential base, and (3) the portion of the representation of the target concept matched, and the relevance of that portion to the goal. The evaluation consists of a numeric rating of the mapping, and a description of the shortcoming(s) of the mapping assigned by the heuristics. If the numeric rating is greater than a threshold value, the potential base is accepted as the final analogy, and the KDSA process halts. Otherwise, the evaluation is passed on to the search control component.

Search Control The search control component uses evaluations from the match evaluation component and other information about the state of the search to influence the direction of the spread of activation. It uses heuristics to control the direction of the search in two ways: (1) it can change activation of concepts in the semantic net, particularly the target concept and the retrieved intermediate concept, and (2) it can modify the condition under which spreading activation will retrieve new intermediate concepts. The first of these, changing activation of selected concepts in the KB, is the more important of the two methods of search control. This method includes strengthening the activation of promising intermediate concepts (those which nearly pass the mapping component's similarity metric for being a good final analogy), weakening the activation of unpromising concepts, changing activation of *portions* of the intermediate concept or the target based on evaluations, and clearing the activation of all nodes in the semantic network (to start the search over from a new state).

A simple use of KDSA's search control would have it clearing all activation in the semantic network each time a promising concept is encountered, and then restarting the search by making the promising concept a source of activation. In this way KDSA can use these promising concepts as *beacons* along the way to the final good analogy. This is very similar to the way that promising intermediate states are used in heuristic search techniques such as hill-climbing or best-first search [Pearl and Korf, 1987].

The use of the matching component of the mechanism to provide feedback to the spreading activation search provides a key distinguishing feature of our approach. Most previous approaches to analogy serialize the retrieval and mapping processes: first they retrieve a concept, then they try to map it, then if mapping fails they start at ground zero with retrieval again. By contrast, mapping in KDSA is an integral part of retrieval: mapping (the matching component) provides ongoing information to the retrieval mechanism (spreading activation and search control) throughout the duration of the retrieval process.

1.2 KDSA Applied to Innovative Design

This section describes the particular heuristics used in the implementation of KDSA, called IDA (for Innovative Design by Analogy), to find analogies which are useful for guiding an innovative redesign of the target.

IDA operates in a knowledge base of devices, natural or man-made systems that perform some function. The knowledge base may contain definitions of other concepts as well, but IDA requires that each device be represented by its structure, behavior, and function. Representations of structure consist of the device's parts along with different types of connections among those parts. Representations of behavior and function consist of chains of primitive processes along with the individuals (structural components, substances, etc.) on which those processes act. IDA takes as input an existing device, and returns as output an abstract redesign of that device which satisfies the device's top-level functional requirements, but does so in a different way. This redesign consists of a replacement of one of the target device's top-level behaviors with a behavior from the base device. E.g., a behavior like SPRAYING from the representation of the sprinkler irrigation system may be replaced with DIFFUSION from the circulatory system.

The particular heuristics used in IDA's mapping component attempt to find analogues that satisfy two general requirements:

(1) The base and target devices must have similar functions, but different behaviors and structures.

(2) The base must be adaptable with respect to the target device—that is, the design system must be able to use the base to adapt the target into a new design. The specific test that IDA uses is that it must be able to substitute individual behavioral components of the base for existing behavioral components of the target, creating a new overall behavior for the target device while preserving its function.

The purpose of the first requirement is to find an analogue that will lead to a redesign that is useful ("similar function") and at the same time novel ("different behavior and structure"). The purpose of the second requirement is to ensure that IDA actually will be able to produce a redesign based on the retrieved base concept, i.e., that the mismatch in behavior with the target is not so great that the two devices have nothing to contribute to one another. Thus the second requirement's implementation will depend on the system's mechanism for adapting the retrieved base into a final design.

To implement these two requirements, IDA's mapping component considers separate portions of a device's representation separately. Each device representation is broken down into structure, behavior, and function. The behavior and function representations are broken down further into (1) a sequence of primitive processes that make up the behavior or function, and (2) the individuals (structural components, materials, etc.) on which those processes act. There are separate requirements on the degree of isomorphism and semantic distance required for each of those portions of the representation. For example, IDA prefers the match between nodes in the structures of the target and base devices to be high in semantic distance (to satisfy the dissimilar structure requirement) and prefers a mismatch on only one primitive process in the behaviors of the target and base devices (to satisfy the adaptability requirement).

After the mapping component evaluates devices according to the two requirements, the search control module must focus the spread of activation toward other devices in the KB which meet those requirements. IDA does this by focusing the search based on the strengths of the retrieved beacons encountered so far in the search. The mapping component identifies an intermediate concept as promising if it comes close to meeting the metric for being a final analogy. For each promising concept, the search control component then strengthens the activation of its portions that did meet the mapping component's individual requirement. The rest of the activation in the semantic network is wiped out, and the search is restarted from this new state.

Another way that IDA's search control rules guide the search to a semantically distant analogy is to use abstractions in the knowledge base as "bridges" to other domains. When IDA retrieves a concept that is in the same domain as the target and is a directly-linked example of a *generic abstraction*—a concept that abstractly describes specific concepts from a number of different domains—it strengthens the activation of that abstraction. This will allow activation to be spread into other domains, increasing the likelihood that IDA will find a distant analogy. In this way, IDA can take use previously-generated analogies to retrieve new ones, even if the previously-generated analogy does not directly involve the current target concept.

1.3 Example

This section presents an example demonstrating the execution of KDSA to retrieve an analogy for creative design. The example shows IDA's behavior for the goal of redesigning a blinkered railroad crossing, that is, an intersection of road and railroad tracks where a train's presence on the tracks is indicated by blinking lights signalling drivers on the road to stop. IDA meets this goal by suggesting redesign by analogy to an on-off valve. Specifically, it suggests replacing the FLASHING behavior in the description of the blinkered railroad crossing with the BLOCKAGE behavior in the description of the on-off valve. The retrieval of the on-off valve takes place in the following steps:

1. The nodes contained in the representation of BLINKERED-RR-CROSSING are made sources of activation (i.e., they are tagged with some number), and IDA begins spreading activation.

2. After a few cycles of spreading activation, the device INTERSTATE-HIGHWAY-SYSTEM is retrieved. This device is mapped to BLINKERED-RR-CROSSING, and the mapping is evaluated. The mapping is found to be unpromising—there is a low degree of semantic distance between the structures of the two devices, and the behaviors and functions of the two devices do not correspond in any respect. This is exactly the opposite of what IDA wants for a final analogy. However, IDA notices that INTERSTATE-HIGHWAY-SYSTEM is an instance of a generic abstraction, the FLOW-SYSTEM device. IDA recognizes this abstraction as a possible mechanism for moving the search out of its current domain, and makes FLOW-SYSTEM a source of activation.

All other activation in the semantic network (except the target's) is cleared, and spreading activation starts again.

3. Another of FLOW-SYSTEM's instances, PLUMBING-SYSTEM, is retrieved next, and the mapping between it and BLINKERED-RR-CROSSING is evaluated. This mapping shows high semantic distance between the structures of the devices, and poor matches between the behaviors and functions of the devices. IDA wants high semantic distance in structure, so the structural aspect of the mapping is rated high, but the behavioral and functional aspects of the mapping are rated low. PLUMBING-SYSTEM is evaluated as a promising near-miss. Since the structure of the PLUMBING-SYSTEM is the strongest part of the mapping evaluation, the search control component makes PLUMBING-SYSTEM's structure a source of activation. Since the behavior and function of the PLUMBING-SYSTEM were rated low, the search control component still bases the search on the behavior and function of the target. So the structure of PLUMBING-SYSTEM and the behavior and function of BLINKERED-RR-CROSSING are made sources of activation, and all other activation in the network is cleared.

4. The next concept retrieved is ON-OFF-VALVE. The matching component recognizes that the mapping between ON-OFF-VALVE and BLINKERED-RR-CROSSING is high in semantic distance between the structures, high in isomorphism between the functions, very low in semantic distance between the top-level process sequence of the functions (they both toggle between PREVENTing and ALLOWing another process), and mismatches in a single process in the behavior description (the BLINKING of the rr crossing corresponds to the BLOCKAGE of the valve). With these conditions met, ON-OFF-VALVE meets the similarity metric for being a final analogy for innovative design. It is retrieved, and IDA's simple design module suggests replacing redesigning the BLINKERED-RR-CROSSING by replacing its BLINKING process with ON-OFF-VALVE's BLOCKAGE process.

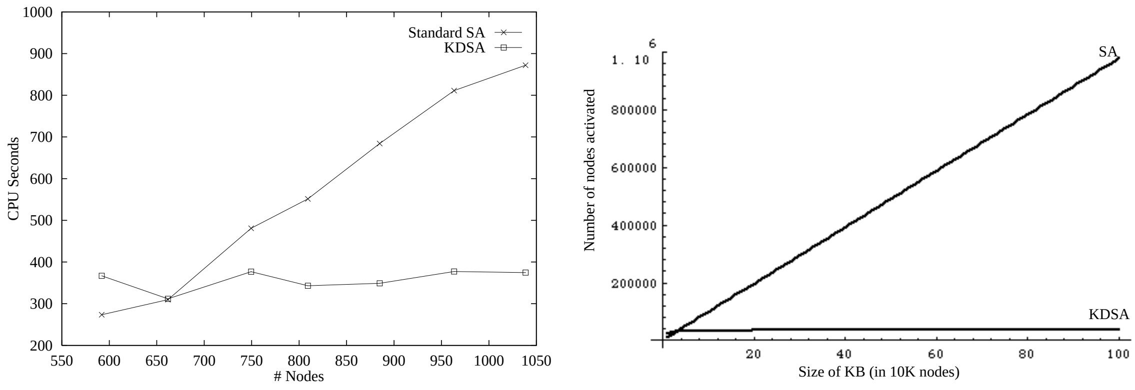
1.4 Results

One of the major questions important in the evaluation of KDSA is: will KDSA retrieve analogies without examining a sizable fraction of the entire knowledge base? In order to answer this question, KDSA was evaluated using two complementary methods.

The first of these methods is to analyze the behavior of a theoretical model. This model predicts KDSA's retrieval time given various parameters such as the size of the knowledge base, the semantic distance required for the analogy, the likelihood of encountering a beacon concept in the knowledge base, and the quality of each beacon concept in terms of the benefit it provides in reaching the ultimate base concept. This model allows us to examine the behavior of KDSA under a wide range of problem and knowledge base characteristics.

The second method of evaluating KDSA is to examine the behavior of the implementation, IDA. This implementation of KDSA demonstrates that KDSA can, in fact, automatically retrieve semantically distant analogies which are useful in solving a real problem. In addition, while it is presently impossible to test IDA with an actual very large knowledge base, we can measure IDA's retrieval time as a function of the KB size for various subsets of IDA's small knowledge base. These experiments allow us to examine KDSA's behavior as the knowledge base grows, and compare that actual behavior to the prediction of the theoretical model.

Figure 1.3 previews some of the results produced by these two validation methods. It shows time taken to retrieve a semantically distant analogy as the size of the knowledge base grows, both for (a) the actual implementation operating in relatively small knowledge bases, and (b) the theoretical model as the knowledge base grows to a size of 1 million nodes. Each graph also shows retrieval time for standard spreading activation (SA) as well. Both methods showed retrieval time for KDSA growing much more slowly than for standard spreading activation as KB size grows. The theoretical model predicts behavior that is roughly logarithmic in the size of the KB.



(a) (b)

Figure 1.3: Retrieval time for KDSA and standard SA as the KB size grows, (a) as observed in the computer implementation, IDA, operating on a small knowledge base and (b) as predicted by the theoretical model in a large knowledge base

Detailed presentations of the theoretical and experimental results are presented in Chapters 5 and 6, respectively. These results can be summarized with the following four qualitative statements, with the first statement being verified by both the theoretical model and experiments, and the remainder being predicted by the theoretical model:

(1) As the knowledge base size grows, retrieval time with KDSA grows much more slowly than does retrieval time with standard SA.

(2) For analogies in which the target and the base are semantically distant, KDSA is far more efficient than standard SA.

(3) KDSA is robust over different distributions and utilities of beacon concepts in the knowledge base. Even when the benefit of each beacon search is low relative to the effort involved in each beacon search, KDSA still shows significant savings over standard SA.

(4) KDSA is robust in the face of bad beacons. When a KDSA search suffers from beacons that direct the search away from, rather than toward, the eventual base, KDSA still

shows substantial savings over standard SA.

1.5 Organization of the Thesis

The remainder of this dissertation presents KDSA and the issues introduced above in more detail. Chapters 2-4 present the general approach: chapter 2 describes the problem of retrieving semantically distant analogies and motivates the need for a new approach to this problem, chapter 3 describes KDSA in detail, and chapter 4 describes the knowledge which allows KDSA to be useful in the task of innovative design. Chapters 5 and 6 present the validation of the approach: chapter 5 details the theoretical model and shows the model's predictions of KDSA's behavior in several interesting situations, and chapter 6 describes the computer implementation of KDSA, IDA, and details IDA's behavior on some example analogy retrieval problems. Chapter 7 relates the work presented in this dissertation to work in other projects in AI and other fields. And chapter 8 summarizes the contributions of this project.

Chapter 2

The Problem

The problem addressed in this thesis is: How can a computer efficiently retrieve semantically distant analogies from a large multi-use knowledge base? This chapter gives a detailed description of this problem and a motivation for the proposed approach to it. The chapter describes the general analogy problem, defines precisely the notion of "semantically distant" analogy (SDA), discusses the ways that humans use SDAs, and presents the difficulties inherent in applying existing retrieval techniques to the SDA problem.

2.1 The General Analogy Problem

Analogy is the process of inferring something about one concept, the *target concept*, based on its similarities to another concept, the *base concept*. An analogical reasoner generally first identifies conditions that hold for both the base and the target, and then infers that some additional condition that holds in the base might also hold in the target. Most researchers divide the analogy process into at least these three steps:

(1) *Retrieval* of a plausibly analogous base concept given the target.

(2) *Mapping* the target to the base by placing in correspondence components of the base concept representation with components of the target concept representation.

(3) *Transfer* of information known about the base concept to the target concept according to the mapping established in step 2. This step may also involve a *validation* that the inference involved in the transfer leads to a useful or sound outcome.

All or nearly all approaches to analogy have a metric which determines whether a relationship between two concepts constitutes an analogy or not. For some approaches (e.g., Greiner's formalism [Greiner, 1988] or Carbonell's Derivational Analogy [Carbonell, 1983a]), this metric more or less measures only whether the transfer step allowed the system to draw a desired inference. Some other approaches contain a (possibly implicit) metric applied *before* the transfer step, either during retrieval or during mapping, which allows the reasoner to forgo the transfer step if the relationship between the two concepts is not considered "analogous" enough for the task at hand. This latter type of metric will be referred to in this thesis as a *similarity metric.*

2.2 Semantically Distant Analogies (SDAs)

Before outlining a method for retrieving SDAs, we must define what "semantically distant" means. Intuitively, we think of two concepts as semantically distant if one very rarely springs to mind when one is thinking of the other. That is, two concepts are semantically distant if they are not normally thought of as connected[1]. In a knowledge base, "not normally thought of as connected" translates roughly to "related to one another only very indirectly". That is, the paths of relations which connect the two concepts through abstractions or other concepts in the KB are long. The precise definition of semantic distance given in Definition 2.1 below is one of several possible specializations of this intuition.

Before defining the notion of semantic distance, let us first lay out some constraints on the type of knowledge representations under which this definition will apply. For the purposes of this discussion, a knowledge base will consist of a collection of definitions of *concepts* or types. Each concept will be defined as a set of relations between *instances*,

[1]Koestler [Koestler, 1965] termed the process of connecting normally unconnected concepts "bisociation"; he believed this process to play a role in all creative thought. We argue shortly that SDAs qualify as one type of bisociation.

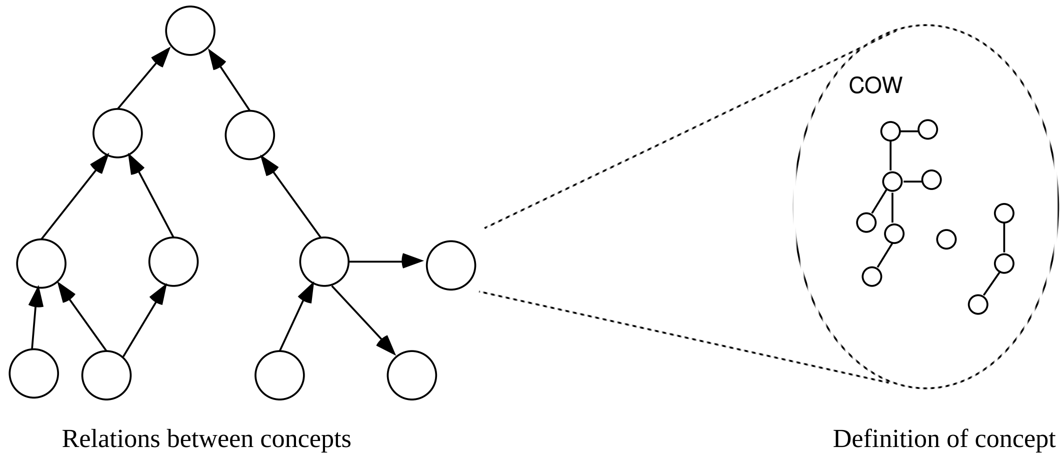


Figure 2.1: Basic assumptions about knowledge representation. Concepts are related in a network, and are defined as a collection of instances and relations between those instances.

where each instance is a member of some concept. These concept definitions may contain true *definitional* information—i.e., conditions which hold for every member of that type—and *prototypical* information—i.e., conditions which would be expected to hold for members of that type. The concepts themselves are also related to one another, through subtype/supertype relationships and other relationships which apply between types irrespective of context.

Figure 2.1 graphically shows the assumptions about knowledge representation. A concept like COW may be defined in terms of relationships between instances of other concepts—MILK, RANCH, and MOO, for example—and may be itself related to other concepts—RUMINANT, for example—through subtype/supertype relationships. Figure 2.1 shows both the inter-concept relationships and the concept definitions as graphs. These representational assumptions are based loosely on John Sowa's Conceptual Graph formalism [Sowa, 1984]. However, it is important to note that the definitions and methods described in this thesis do not depend on representing knowledge as graphs. The concept definitions could just as easily be given as logical sentences, for example, with the variables as instances and predicates as relations. What *is* required here is that there be direct (constant-time) associative access from each concept (or instance) to each related concept (or instance). To summarize, then, this discussion assumes two types of objects in the knowledge base, concepts and instances, and three types of bidirectional relations between objects—concept-to-concept relations, instance-to-instance relations inside a concept definition, and the relation between an instance and its type.

Given this representational framework, let us now present a precise definition of *semantic distance*. Let the *minimum path* between two objects o_a and o_b in the knowledge base be a sequence of objects $o_0, o_1, \ldots, o_{n-1}, o_n$ such that $o_0 = o_a$, $o_n = o_b$, o_i is related to o_{i+1} for all $i < n$, and any other path of related objects connecting o_a and o_b in the knowledge base has length $\geq n$. Also, let the *minimum distance from concept* between an object o and a concept C, $MDC(o, C)$ be the length of the shortest minimum path between o and any instance contained in the definition of C. The semantic distance between two concepts will then be defined as:

Definition 2.1 *The semantic distance between two concepts is defined as the average path distance between the instances contained in the definition of one concept and the closest instance contained in the definition of the other (averaged across both concepts). That is, the semantic distance between two concepts C_1 and C_2, $d(C_1, C_2)$ is given by:*

$$d(C_1, C_2) = \frac{\sum_{o \in C_1} MDC(o, C_2) + \sum_{o \in C_2} MDC(o, C_1)}{|C_1| + |C_2|}$$

Here, $o \in C$ means "o is an instance contained in C's definition", and $|C|$ is the number of instances in C's definition.

Note that this definition skirts several issues which might be important in a more general psychological or linguistic definition of semantic distance. In particular, there is no notion of the context in which the comparison between concepts is being made, e.g., no notion of the purpose toward which the comparison will be applied. There is also no consideration that different types of relations may themselves embody different levels of semantic distance; here all single-level relations between objects are treated as equally distant. This definition is constructed to make explicit the search difficulties in finding semantically distant concepts—if the path distance between concepts in the KB is high, the expense in searching for the connection between the two concepts along the KB's relations is likely to be high as well.

A *semantically distant analogy*, then, is simply an analogy between two concepts whose semantic distance is high relative to semantic distances between other pairs of concepts in the KB. This is the opposite of analogies based on a high degree of surface similarity, where the two concepts have a number of features that match exactly. Since people tend to retrieve analogies based primarily on surface similarity [Holyoak and Koh, 1987], the definition given for an SDA does seem to meet Koestler's criterion for bisociation—it involves two concepts not normally thought of as connected.

2.3 Human Use of SDAs

Despite Holyoak and Koh's (and others') finding that people generally retrieve analogies based on surface similarities, there is evidence that people occasionally are able to generate and use SDAs. This is particularly true among people performing tasks that are commonly classified as *creative*. The literature on creativity, and particularly the literature on scientific discovery and invention, contains many examples of individuals using cross-domain unexpected analogies to guide them to novel results. It seems likely that giving computers the ability to retrieve and use SDAs will bring them one step closer to exhibiting true creativity themselves.

Many separate case histories in invention and discovery point out the usefulness of cross-domain analogy in creativity:

- Gutenberg had worked for years to improve the existing technology for printing, without success. Upon attending a wine festival, he encountered the workings of the wine press. Immediately he recognized that the same mechanism which applied and then removed steady pressure to grapes could be used to apply type to paper and remove it to avoid smudging [Koestler, 1965].

- Many inventors working on a flying machine in the early part of this century were unable to get their inventions off the ground because they tried to restrict the pilot's control to only one axis of motion. The Wright brothers, connecting the building of a flying machine with their work in their bicycle shop, recognized the need to give the pilot control along three different axes of motion (including banking the vehicle during turns), and produced the first plane [Crouch, 1971].

- "Thomas Edison used metaphors extensively. He worked out the quadruplex telegraph, perhaps the most elegant and complex of his inventions, 'almost entirely on the basis of an analogy with a water system including pumps, pipes, valves, and water wheels', according to his son Theodore. Later, thinking metaphorically, Edison conceived of the interaction between existing illuminating-gas distribution systems and the illuminating incandescent-light system he intended to invent." [Hughes, 1971].

- Archimedes, Kekulé, and Poincaré produced important discoveries in the fields of physics, chemistry, and mathematics, respectively, aided by unexpected analogies [Langley and Jones, 1988].

- A new improved design for the high frequency component of audio speakers ("tweeters"), and later for speakers in general, was produced only after the connection between beam width in radar and sound distribution from the speakers was recognized [Kock, 1978].

- An inventor recently noticed how small pebbles and even grains of sand felt larger through a water balloon; using this principle, he invented the sensor pad—a device which facilitates breast cancer lump detection [Clark and Maier, 1988].

- An inventor at a recent idea fair in San Francisco showed off a new umbrella which dries itself by shaking itself off like a dog.

Perhaps more important than the numerous examples in the literature is the agreement among inventors and people who have studied invention that the ability to make connections between problems, solutions, and principles in a wide array of diverse domains is critical to an inventor's success. Koestler's idea of bisociation, which he views as characteristic of *all* creativity, reflects this ability [Koestler, 1965]. Middendorf asserts that "...creative persons usually have ability and willingness to explore tenuous connections between only remotely connectable things" [Middendorf, 1981]. Kock believes in the importance of combining knowledge from multiple disciplines in the invention process, and he points out that a large number of inventions were produced by people who weren't specialists in the field [Kock, 1978]. And successful inventor L. W. Andrews describes his own process of invention as follows [Rossman, 1931]:

> First seek out as many analogies as possible, criticise and eliminate; then examine the small number left by experiment. If this fails begin again with analogies of a quite different type and again follow the process as above. But always analogy is the leading string.

The example analogies given above seem to have been retrieved in a wide variety of ways. Some analogies, such as Gutenberg's, Archimedes', and Kekulé's, seem to have been retrieved when the inventors encountered cues in their interaction with the world, cues that in turn reminded them of the analogue. Some, such as Poincaré's, were retrieved with no external cue at all. Some, such as the water balloon/breast lump detection analogy, were produced by reasoning "backwards", using a solution—or rather a new principle which could be applied as a solution—to retrieve a problem. And some, such as Edison's, seem to involve constructing entirely new concepts to serve as analogues. We would like our computer mechanism to exploit all of the different paths to analogy that different human inventors have followed.

2.4 Difficulties in Computer Retrieval of SDAs

The studies of analogy use among creative individuals have shown that one must be able to draw from a wide body of knowledge of diverse fields to successfully use SDAs. This result in humans suggests that the ability should also exist in computer models of SDA retrieval. In order for a computer program to be able to retrieve and use SDAs to aid its problem solving, it must be able to draw these analogies from a large diverse knowledge base. This sort of knowledge base is the ultimate goal of the CYC project [Lenat and Guha, 1990], among others.

The studies of analogy use in creative people also lead us to another assumption about knowledge representation in computer SDA retrieval models: it is impossible to predict at concept storage time all the different ways a concept may be used in analogy. All the examples of SDAs given in the previous section show concepts being used in surprising and unpredictable ways by inventors.

There are two major classes of analogy retrieval techniques currently in use: search along relationships in the knowledge base and indexing the knowledge base on salient features. Both of these techniques, while useful in retrieving same-domain analogies from small knowledge bases, are poorly suited for retrieving SDAs from a very large, diverse, multi-use knowledge base.

2.4.1 Knowledge Base Search

One major class of methods for analogy retrieval (and knowledge retrieval in general) involves searching the knowledge base, starting at the target, along the defined connections between concepts, e.g., links in a semantic network. This class includes spreading activation approaches (e.g., [Anderson, 1983; Collins and Loftus, 1975; Rau, 1987a]), as well as Winston's Classification-Exploiting Hypothesizing [Winston, 1980] and Quillian's intersection search [Quillian, 1968]. While these approaches replicate many of the characteristics of human associative memory and are useful in retrieving semantically close analogies, they are not tractable for retrieving SDAs *spontaneously*—that is, without any cues from the outside world. The problem is that finding SDAs will require a deep search into the KB from the target (Definition 2.1), and in a large knowledge base the search will face combinatorial explosion well before it is able to reach a semantically distant base.

One possible solution to this problem of combinatorial explosion would be to base the search *only* on those few components of the target definition for which a more exact match is required. Even in a task for which an SDA is desired, there may be a very small number of features which should match exactly or near-exactly. The retrieval mechanism could then search for nearby concepts to these few features so that the depth required of the search would not be as high. But this approach also has tractability problems; in this case, the difficulty is in the high hit rate of retrieved concepts. If the search for an analogue is based on only a few features, in a large KB many concepts are likely to meet the lax retrieval criteria and will be retrieved. These retrieved concepts will in turn have to be mapped and evaluated. Since mapping is a very expensive (NP-complete) operation, retrieving a high number of candidate concepts is not a tractable alternative to a more detailed KB search.

2.4.2 Indexing on Salient Features

The second major class of techniques used for analogy retrieval is indexing the knowledge base on a set of feature tests. These approaches, which are often used in case-based reasoning systems (e.g., [Kolodner, 1984]), usually involve using a series of tests on feature values to guide the retriever down a discrimination network to a matching case. The series of tests ensures that the retrieved base is similar to the target along a set of salient features, where the choice of salient features depends on the specific task for which the case base is being indexed.

Indexing approaches are often very efficient for retrieving same-domain analogies when the task is known ahead of time. However, as we have noted earlier, a given concept can be useful for many different possible tasks in semantically distant analogy, and it is impossible to anticipate all conceivable specific tasks for which a concept may be retrieved. And even if it were possible, the storage costs for separate indexing structures for each possible task would be prohibitively high. These problems with indexing are discussed in more detail in Section 7.1.2.

An additional problem with most indexing approaches is that during the retrieval phase they assume a very simple representation of concepts in the knowledge base, namely, feature vectors (where the value of each feature can be translated into a scalar). When concepts are represented this way, isomorphism essentially becomes a non-issue in retrieval. This representation, while appropriate for a CBR-style task in which every possible concept is essentially the same type, is unrealistically limiting for a large general-purpose knowledge base.

2.5 Desiderata for Solution

To summarize, an analogy mechanism for tractably retrieving SDAs requires three important features that distinguish it from other knowledge retrieval mechanisms:

(1) It should operate in a very large, diverse, multi-use knowledge base.

(2) It should typically examine a relatively small portion of the knowledge base before finding a semantically distant base, and should limit the number of target-base mappings and evaluations it performs during the course of retrieval.

(3) It should assume a task-independent representation and organization of concepts in the KB.

The next chapter presents a general analogy retrieval mechanism which meets these criteria.

Chapter 3

The Approach: Knowledge-Directed Spreading Activation

The previous chapter described the value of retrieving semantically distant analogies and motivated the need for a new approach to computer retrieval of SDAs from a large knowledge base. This chapter introduces such an approach, called Knowledge-Directed Spreading Activation (KDSA). KDSA is able to find SDAs by using task-specific knowledge to direct a spreading activation search from the target concept to a base concept. In particular, KDSA derives a great deal of its power from exploiting promising near-analogies—concepts which nearly meet the system's similarity metric—retrieved at previous stages of the search.

This chapter presents the basic operation of KDSA. Section 3.1 describes spreading activation and its variants. Section 3.2 describes the KDSA framework and algorithm in detail, including its integration into a general agent architecture. And Section 3.3 discusses some additional issues about KDSA, including the conditions under which it will find analogies efficiently.

3.1 Spreading Activation

Spreading activation is a mechanism for associative memory access in semantic networks. There are many variants on the general theme of spreading activation in the literature; the version described here will be a generalization of many of the particular implementations. Spreading activation's general execution is illustrated graphically in Figure 3.1. The process begins when a concept becomes the focus of attention of an agent. This concept's representation (i.e., each of the instances included in the concept's representation) is tagged with some *activation* (1), where the activation can be a boolean marker or a numeric value. In the figure, the activation is assumed to be numeric, with a node's level of activation indicated by shading. This initially activated concept is said to be a *source* of activation. The spread of activation then takes place in a series of cycles (2). During each cycle, each activated object in the KB spreads activation to each of its neighbors. In the figure (2), initially only the node inside *a* is activated; next cycle, activation is spread to the nodes in *b*, then the next cycle to *c*, and so on. The amount of activation assigned to a neighbor object is a function of five values: the level of activation of the activated object, the previous level of activation of the neighbor object, the link which connects the two objects, and the value (i.e., semantic content) of both the activated object and neighbor object. When numeric values are used as activation, the activation level of each concept is often *decayed* by some fraction during each cycle. A new concept in the KB is considered retrieved (3) when the instances which make up its representation accumulate aggregate activation which exceeds some threshold. In the figure, a spreading activation search which starts with COW as the source ends in the retrieval of HORSE and MILK because of the number of features they have in common with COW.

Spreading activation is used most often as a psychological model of associative memory [Holland *et al.*, 1986; Anderson, 1983; Collins and Loftus, 1975], but it has also been used as a method of computer information retrieval in networks without regard to psychological plausibility [Rau, 1987a; Cohen and Kjeldsen, 1987]. It has been used to retrieve concepts according to many different similarity criteria; in particular, it has been used as analogy

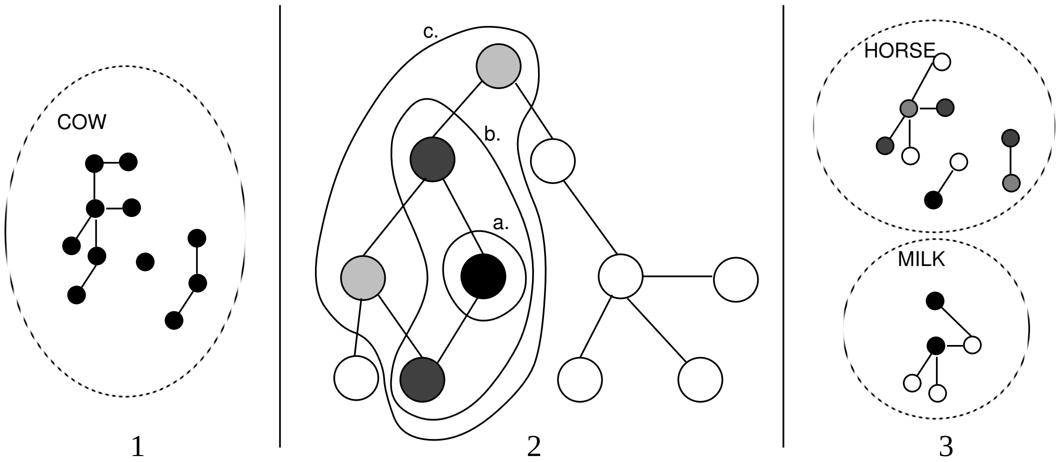


Figure 3.1: Spreading activation. Operates by (1) assigning a concept in the KB some activation, (2) continuously spreading activation from each activated object to all its neighbors, until (3) some other concept(s) in the KB accumulate aggregate activation over some threshold, at which point they are retrieved.

retrieval mechanism [Holland *et al.*, 1986; Anderson and Thompson, 1989; Jones, 1989].

Spreading activation has a number of benefits as an analogy retrieval mechanism. It meets one of the criteria presented in Section 2.5 in that it retrieves concepts without assuming any task-specific indexing or organization of the knowledge base. It exhibits many known characteristics of human recall, so it is psychologically plausible as a retrieval mechanism. And it is flexible in terms of the types of concepts it is able to retrieve; e.g., as discussed for some of the inventors in Chapter 2, it can search from a problem to the solution or from a potential solution to a relevant problem. The major disadvantage of spreading activation as a general analogy retrieval mechanism, also mentioned in Chapter 2, is that it is basically a blind search mechanism. As such, it is susceptible to combinatorial explosion, especially when the concepts for which it is searching are far from the original source of activation.

One method used to control the search in spreading activation is implementing a practice effect in which activation-passing strengths on the links are modified. When this method is used, the strength on links is increased whenever the link leads to a successful retrieval (where "successful retrieval" generally means that the retrieved object leads to a successful result in problem solving) and weakened otherwise. This can lead to a network where much of the spreading activation search occurs along a smaller number of "strong" links, while other links in the knowledge base do not participate in the activation passing process. This sort of practice effect is unlikely to make spreading activation a tractable method for retrieving SDAs for two reasons: (1) at best, it only decreases the branching factor of the spreading activation search and does not remove the problem of combinatorial explosion in large-depth searches, and (2) since SDAs are likely to be retrieved infrequently, and practice effect methods are based on frequency and recency of recall, the weights they learn are not likely to be the right ones to lead future spreading activation searches to SDAs.

3.2 Knowledge-Directed Spreading Activation

Since the problem with spreading activation is the blind nature of the search, it stands to reason that the same general kinds of techniques used to improve the efficiency of blind state space search methods can also be used to allow spreading activation retrieve SDAs. A number of techniques have proven useful in speeding up state space search:

(1) Performing heuristic evaluations of generated states, preferring to expand high-rated states over low-rated ones (used in A* [Hart *et al.*, 1968] and hill-climbing search [Pearl and Korf, 1987]).

(2) Pruning states from the search that are low-rated (used in beam search [Lowerre and Reddy, 1980]).

(3) Restarting search (i.e., wiping out the record of the previously expanded portion of the search space) from an encountered state if that state is known to represent progress toward the goal (used in hierarchical planning [Sacerdoti, 1974]).

Knowledge-Directed Spreading Activation is an application of these abstract ideas to knowledge base search. KDSA uses task-specific knowledge to 1) evaluate concepts retrieved during spreading activation and restart search from those concepts evaluated as promising,

2) prune unpromising areas of the knowledge base from the search, and 3) direct the spread of activation toward more promising areas of the knowledge base by dynamically refining the start and goal states of that search. The next section describes how KDSA uses these procedures to efficiently find SDAs.

3.2.1 Basic Algorithm

Figure 3.2 graphically shows the execution of the KDSA algorithm, which was given in Figure 1.1. KDSA works as follows. When a target concept is given, KDSA makes it a source of activation and begins the spreading activation process. When spreading activation retrieves a concept, the *mapping component* maps the new intermediate concept (the IC) to the target, and evaluates that mapping. If the evaluation determines that the IC meets the similarity metric, it is returned as the retrieved base concept (already mapped to the target) and the process halts. If the IC does not meet the similarity metric, a description of the mapping evaluation is passed on to KDSA's *search control component*. Based on that evaluation, and the history of the search, the search control component modifies the state of the spreading activation mechanism to direct future searches into more promising regions of the knowledge base. The spreading activation process is started again, and the entire KDSA loop repeats until a base is found.

Clearly the sources of power in KDSA are the mapping and search control components. They are described in detail below.

Mapping Component

The mapping component has three major roles in KDSA:

(1) It *generates the mapping* between the IC and the target. That is, it finds a set of correspondences between some subset of the items in the target's representation and some subset of the items in the IC's according to some mapping criterion.

(2) It serves as the *similarity metric* for the analogy mechanism. That is, it decides whether the mapping between the target and the IC constitutes an analogy likely to

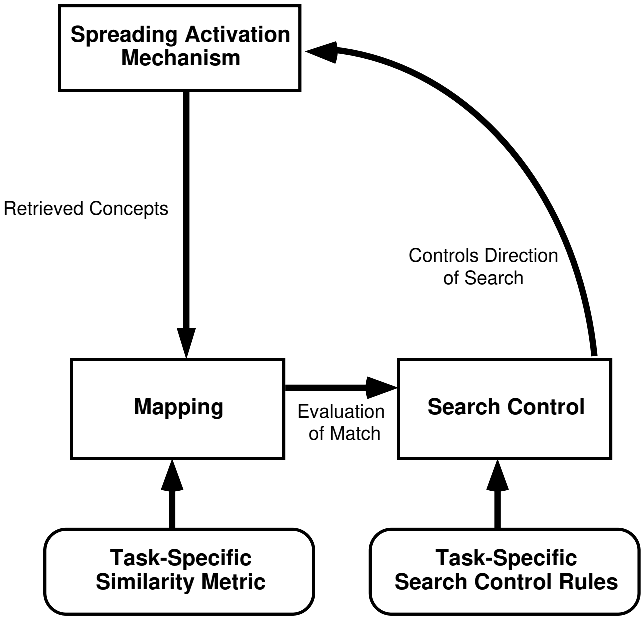


Figure 3.2: Knowledge-Directed Spreading Activation

be useful in solving the task at hand. Those mappings that meet the similarity metric are returned as final analogies by KDSA to be used in the transfer step of the analogy process.

(3) For those ICs that do not meet the similarity metric, it *evaluates the mapping.* This evaluation consists of two separate outputs: first, a rating of the quality of the mapping according to its closeness to meeting the similarity metric; second, a listing of the specific areas of weakness of the mapping, i.e., those conditions of the similarity metric which were not met by the mapping.

After the mapping is generated[1], the mapping component evaluates it according to task-specific rules. This evaluation *step* performs both the evaluation and similarity metric *roles* listed above. The evaluation depends on three distinct aspects of the mapping: 1) the degree of *semantic distance* between target and base, 2) the degree of *isomorphism* between

[1]The specifics of mapping generation are not part of the KDSA formalism; any of a number of techniques can be used. Chapter 6 describes the technique used to generate a mapping in IDA.

target and base, i.e., the degree to which the relationships between corresponding nodes themselves correspond, and 3) the particular *portion* of the target representation which is being evaluated with respect to 1) and 2).

This third aspect—the portion of the representation being considered—needs further explanation. The task-specific mapping rules divide the representations of concepts into separate portions, where the portions are defined according to their importance and role in solving the task. Each portion of the representation has separate conditions on the levels of semantic distance and isomorphism needed for the mapping to meet the similarity metric. For example, we will see in Chapter 4 that, for the task of innovative redesign, salient portions of the target representation are the structure, behavior, and function of a concept; the similarity metric for innovative redesign has separate requirements on semantic distance and isomorphism for each of those portions. This identification of salient portions of the representations is the way that the goal of the analogy influences retrieval in KDSA.

The sequence of steps in the mapping evaluation is as follows. First, using the subdivision of the target representation, the mapping is divided into portions. Second, each submapping is rated according to its level of isomorphism and semantic distance. Third, each rating is compared to the separate metric for that portion of the mapping. If every subportion of the mapping meets its metric, the mapping is returned as an analogy. If not, a description of the mapping evaluation is produced, identifying those areas of the mapping which fall short of the similarity metric, and the degree of the shortcoming(s). This qualitative description is then passed on to the Search Control Component.

Search Control Component

The purpose of the search control component is to modify the direction of the search toward more promising areas of the knowledge base, based on the target, the task, and the current state of the search. Here "current state of the search" consists of the set of all mapping evaluations performed during the course of the search (most importantly the most recent one), and the general state of activation in the network. The direction of the search is modified by altering the spreading activation mechanism, i.e., changing the level of activation of objects in the knowledge base, changing the ability of concepts to receive activation in the future, or changing the retrieval criterion of spreading activation. Earlier descriptions of KDSA [Wolverton and Hayes-Roth, 1993] included the modification of activation-passing strengths on links as one of the allowed mechanisms for modifying search. While this is still theoretically possible, in practice it has been difficult to implement in a principled way, and the other modifications of spreading activation listed have provided more than adequate control over the search direction.

Based on the mapping evaluations performed during KDSA, the Search Control Component can modify activation of objects in the KB in any of a number of ways:

(1) It can strengthen the activation of ICs evaluated as *promising*, i.e., those which failed to meet the similarity metric only in a small number of portions of the overall mapping.

(2) It can weaken the activation of ICs evaluated as unpromising, and can direct future searches away from those areas of the knowledge base by weakening the unpromising concepts' ability to receive future activation.

(3) It can selectively change the activation of *portions* of the target or IC depending upon an evaluation. For example, it can strengthen activation of portions of an IC which meet the similarity metric.

(4) It can clear all activation in the KB (except for the objects activated by methods (1) or (3)), effectively restarting search from a new state.

3.2.2 Integration into Problem Solving Architecture

Figure 3.3 shows how the algorithm of KDSA (Figure 3.2) becomes part of a general agent architecture. Here we assume that the agent's problem solving executes by continuously triggering, evaluating, and executing rules in the manner of BB1 [Hayes-Roth, 1990], Soar [Laird *et al.*, 1987], or any number of other AI architectures, and that the agent receives information from the world through sensors. Spreading activation is the agent's mechanism for associative memory and operates separately from the agent's problem solving.

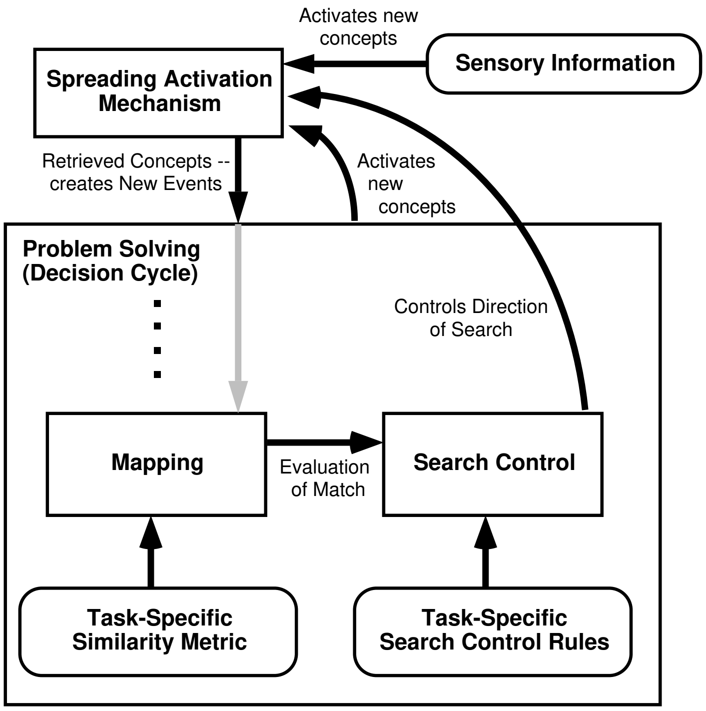


Figure 3.3: Integration of KDSA into problem-solving architecture

The spreading activation mechanism creates memory retrieval *events*, which in turn trigger problem solving rules. The problem solving rules, when executed, can activate new concepts. Sensory information enters the system through the spreading activation mechanism by making sensed concepts sources of activation.

In this model, the evaluation and search control components of KDSA constitute part of the architecture's problem-solving knowledge, and compete with other problem-solving activities for execution. Because of this, KDSA's basic "spread activation/map and evaluate/modify search" loop of Figure 3.2 is effected in three ways: (1) the loop may be interrupted by other problem solving activities that have higher priorities; (2) concepts that are made sources of activation by the agent's problem solving may be retrieved and evaluated by KDSA; and (3) similarly, concepts that are activated by the agent's world sensors may be retrieved and evaluated by KDSA. These last two types of interactions with the architecture allow KDSA to retrieve analogies aided by external or internal cues. That

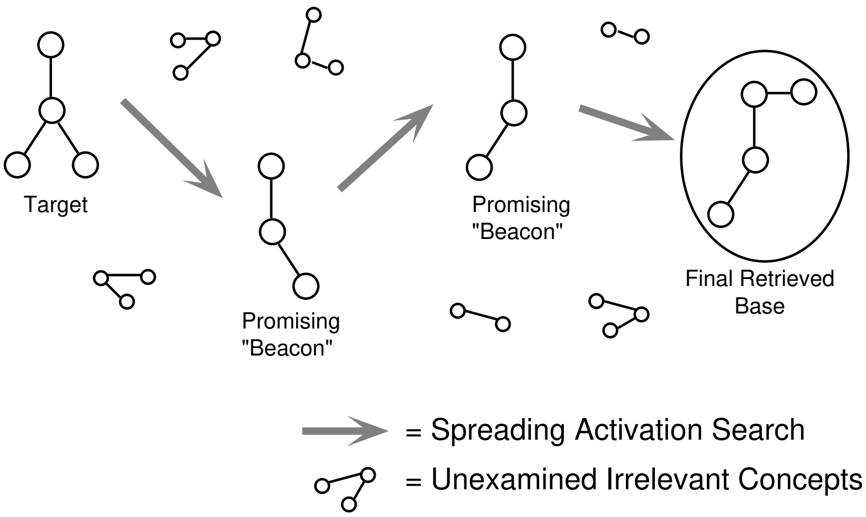


Figure 3.4: Guiding search by noticing beacons

is, concepts sensed in the world or encountered during reasoning, even those that are unrelated to analogy retrieval, may "remind" KDSA of a good analogue to its target or of a promising concept to be used to direct the spread of activation.

3.3 Discussion

A simple way to use KDSA to improve search would be to have only one search control rule: each time an intermediate concept is evaluated as promising, clear all activation in the network, and make the IC a new source of activation. This would produce behavior in KDSA very analogous to the base level search in abstraction planning, where a completely new search is started each time a state from the abstract level plan is encountered at the base level. In this way, KDSA would use the promising concepts encountered as "beacons" leading it to the final base (Figure 3.4)—concepts used in this way will be referred to as *beacons* from here on.

In order for this beacon behavior by KDSA to serve as a source of search efficiency, KDSA must be able to find beacons that successively get the search closer and closer to a concept that will meet the similarity metric. This, in turn, depends on the following informal hypothesis being true:

Hypothesis 3.1 *Beacons—concepts that are close in some respect to meeting the similarity metric—are likely to be semantically close to concepts that do meet the similarity metric or at least to other beacons.*

Whether this hypothesis holds true or not depends on the configuration of the knowledge base and on the particular similarity metric used. But there is reason to suspect it is true for most KBs using similarity metrics based on semantic distance: concepts that are *almost* semantic distance n away from the target, for instance, are going to be semantically close to concepts that *are* semantic distance n away from the target, simply by the definition of semantic distance.

The simple search control heuristic described above provides a useful simplified way to see the benefits of this approach. The analysis in Chapter 5 makes use of these simplifications and models KDSA as a search from (single) beacon to (single) beacon.

KDSA can gain additional search efficiency by using more sophisticated search control heuristics that: (1) base search on the desirable characteristics of *all* of the beacons encountered so far, (2) prune unpromising areas of the knowledge base from the search, and (3) use other means to modify the spreading activation mechanism. The next chapter describes an implemented instantiation of KDSA for innovative design which uses search control in this more sophisticated way.

Chapter 4

KDSA Applied to Design

The previous chapter described an approach to analogy retrieval that is tailored toward tractable retrieval of semantically distant analogies from a large knowledge base. That description presented only the domain-independent formalism, presenting only the modules that make up the approach and constraints on the basic ways they can examine and operate on the knowledge base. This chapter makes the description of KDSA more concrete by presenting the particular heuristic rules used by KDSA to find analogies for innovative design. The rules operate within KDSA's two heuristic modules: the mapping evaluation rules comprise a similarity metric, identifying mappings which are likely to be useful analogies for guiding an innovative design process; and the search control rules direct the search toward other concepts which are likely to meet the similarity metric. The formalism comprised of the KDSA architecture together with these rules will be referred to as $KDSA_{ID}$. Although there are specific differences between $KDSA_{ID}$ and the implementation of it in IDA, the heuristics and examples given below basically reflect the implementation and execution of the program.

This chapter first defines the term "innovative design" and suggests how analogy can be a useful tool for that task. It then describes in detail the mapping evaluation and search control rules in $KDSA_{ID}$, and presents a detailed example illustrating those rules in action. Finally, it discusses some additional issues related to analogical innovative design.

4.1 Innovative Design

Many researchers have classified design along a spectrum of the amount of creativity demonstrated by the designer. Most often, this spectrum is divided into three distinct classes: *routine*, *innovative*, and *creative* design. Some researchers, such as Brown and Chandrasekaran [Brown and Chandrasekaran, 1989] and Gero [Gero, 1990] focus on the *process* of producing the design, making distinctions based on the nature of the design space being searched, the types and quantities used in designing, etc. By contrast, Goel [Goel, 1989] classifies design in a way that focuses more on the *product*, i.e., the design itself[1]. For Goel, a design is routine if it differs from an existing design only in values of parameters; a design is innovative if it involves the same overall topology as an existing design, but changes to individual components; and a design is creative if its overall topology differs from previous designs. The type of design for which $KDSA_{ID}$ retrieves analogies meets Goel's definition of *innovative redesign*, inasmuch as $KDSA_{ID}$ finds a base device that has a top-level behavioral topology that is similar to behavior of the target device it is redesigning, but that has individual behavioral processes within that top-level topology that differ significantly from the corresponding processes in the target's behavior.

Goel's definitions provide a clue as to how analogy can guide the process of design. A new design can be created from an old one by transferring portions of the representation of the old one to the new one. Here the new design can start out as simply a functional specification if the task is to design a new device, or it can be a complete description (function, behavior, and structure) of an old device if the task is to redesign it. In innovative redesign, the transferred portions will be structural, behavioral, or functional components of the base design. The transfer step will not necessarily involve direct copying of a component—it may be modified in some way during the transfer procedure. Also, after transfer there may be some further adaptation of the new design.

Specializing Greiner's definition of useful analogical inference [Greiner, 1988], we will

[1]Goel's definition may more accurately be said to apply to *redesign*—how a new design is produced based on an old one—and redesign is, by definition, a process. The definitions presented here are adapted from Goel's by identifying differences between the new design and existing ones, rather than differences between the new design and the old one it was based on.

say that an analogy is *useful for innovative redesign* of an existing device if it, along with the designer's design knowledge, can be used to produce a novel redesign of the existing device that satisfies the existing device's functional specification. For our purposes, a new device is "novel" if satisfies its functional specification in a way that is different from other devices the designer knows about, not different from all other devices in the world. In this way, Gutenberg's printing press was "novel" even though there is evidence that the Chinese had developed a printing press some 100 years earlier.

4.2 KDSA Heuristics for Innovative Design

This section describes a complete set of heuristics for guiding KDSA toward analogues that are useful for innovative redesign.

4.2.1 Preliminaries

$KDSA_{ID}$ is intended to be the analogy retrieval component for a design system that solves the following problem: given an abstract description of an existing device, produce an innovative redesign of that device. That is, the design system produces a partial description of a device that is based on the existing device and that satisfies the same functional specification as the existing device, but that satisfies that function in a novel way. The system is given no additional requirements for the new design—i.e., no behavioral or structural constraints, failures of the existing device to overcome, etc.; it only knows that the redesign must satisfy the same function as the existing device, and that it must be novel. The design system solves this problem by analogical reasoning: it retrieves from its knowledge base a base device that is analogous to the existing (target) device. The redesign is produced by replacing one or more of the high-level behaviors of the target with high-level behaviors from the base. Since the output of this system produces only a new *behavioral* description of the target device, and does not suggest a new structure, it is only producing a very abstract conceptual-level "design". Obviously much more work would be involved to refine the output of the system into a complete implementable device description.

Representation

The heuristics of $KDSA_{ID}$ assume that each device is represented as a triple, {function, behavior, structure}, where the function describes the device's top-level input-output behavior, the behavior describes the individual processes that produce the function, and the structure describes the device's physical components and the connections between them. Each device representation is assumed to have only one function; a real-world device with more than one function can be represented with multiple device descriptions, one for each function. The behavioral descriptions consist of a network of temporally and causally related primitive behavioral processes (e.g., heating, boiling) and the individuals (structural components or substances, e.g., water, pot) on which they act. The functional descriptions in the device representations are similarly represented as a network of primitive functional processes (e.g., delivery) and the individuals they act on.

Figure 4.1 shows an example device representation using the assumptions above. Borrowing from Sowa's graphical format [Sowa, 1984], we show the objects in the representation as rectangles and the relations (which we will refer to as "links") of the representation as circles. The figure represents a sprinkler irrigation system, i.e., a system which distributes water to plants by spraying it in an area that includes the plants. The structure of the device is represented as a water source (the supertype of lake, reservoir, river, ocean, etc.) connected to a mechanical pump, which is in turn connected to a network of pipes, which are in turn connected to sprinklers. The sprinkers share a location with some plants. The behavior of the device is represented as three processes: (1) the mechanical pump pumps water from the water source, (2) water flows from the pump through the pipe network to the sprinklers, and (3) the sprinklers spray water to the plants' location. The subgraph which includes the three primitive processes themselves along with the temporal or causal connections between them is referred to as the *process sequence* of the behavior; the remainder of the behavior graph constitutes the *individuals* of the behavior. These portions are used separately in the mapping component of KDSA. The function of the sprinkler irrigation system is represented as simply delivering water from the water source to the plants. The

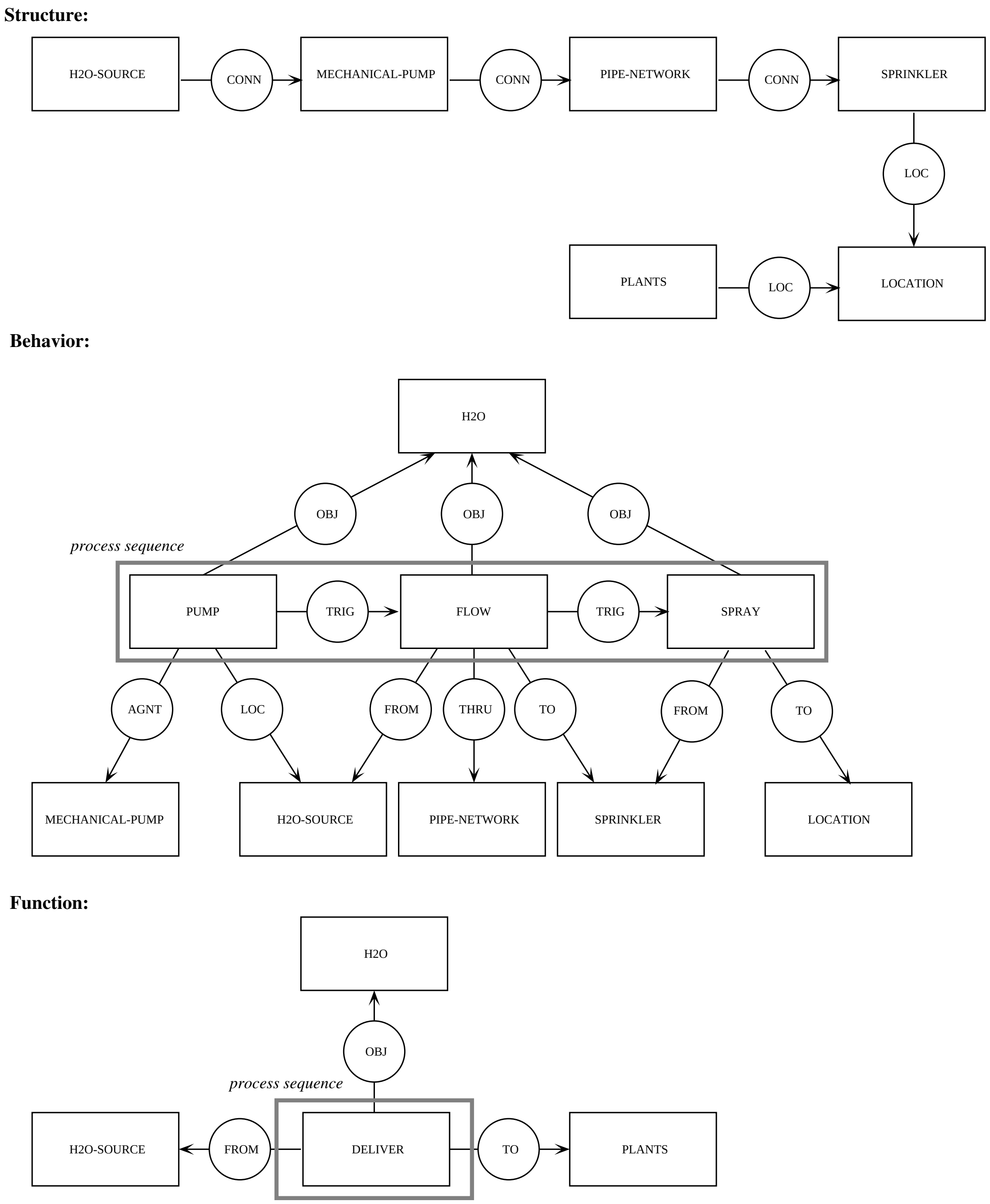


Figure 4.1: Example device representation—sprinkler irrigation system

process sequence of the function is simply the DELIVERY node, and the rest of the function graph constitutes the individuals of the function. Not shown in the figure is the fact that nodes are shared across structure, behavior, and function. For example, the H2O-SOURCE node in the structure graph is actually the same node as the H2O-SOURCE node in the behavior graph and the H2O-SOURCE node in the function graph.

In this representational framework, the terms "process" and "individual" are borrowed from Qualitative Process Theory [Forbus, 1984]. The primitive processes in the behavior of sprinkler irrigation—PUMP, FLOW, and SPRAY—can be thought of as QPT processes. That is, they could be represented as a set of preconditions under which they will become active and effects they will have when made active, where the preconditions and effects may involve some change to qualitative or quantitative variables, and where the effects of one process may cause the preconditions of one or more other processes to be met. The behavior representation shown in Figure 4.1 is intended to be a trace of the process activations in a qualitative simulation of the "normal"—i.e., non-faulty—behavior of a sprinkler irrigation system. That is, it could be derived from the portion of a QPT envisionment representing the normal behavior of sprinkler irrigation by removing the information about the exact conditions on qualitative variables which caused process activations, and replacing that information with simple causal relations (TRIG) between processes which allow one another to become active.

4.2.2 Mapping Evaluation

As mentioned in Chapter 1, the mapping evaluation heuristics of $KDSA_{ID}$ define two requirements for meeting the overall similarity metric:

(1) The base and target devices must have similar functions, but different behaviors and structures.

(2) The base must be adaptable with regard to the target device. In general, this means that the analogy mechanism's transfer step is able to adapt the retrieved base into a redesign which satisfies the target's function. For $KDSA_{ID}$, since it assumes an

analogy mechanism with limited transfer capabilities, this means that it must be able to replace a single process from the target device's behavior with one from the retrieved base.

The first of these requirements increases the likelihood of finding an analogue which will lead to a redesign that is useful and at the same time novel. The "similar function" requirement means that the retrieved base will do abstractly the same thing as the target, so a redesign guided by the base will also meet the target's abstract functional specifications. But the "different behavior and structure" requirement means that the retrieved base will achieve its function in a different way from the target, increasing the likelihood that the redesign guided by the base will be novel or innovative.

The second of these requirements assures that the design system will be able to suggest a redesign based on the retrieved base. The first requirement looks for a base with a different behavior from the target, but the second requirement assures that the difference is not so great that the analogy mechanism cannot perform the final, transfer step in the analogy process. In some sense, this requirement serves as a sanity check for the system. Since the first requirement is looking for a base device which matches the target only abstractly in function, there is a danger that the analogies will be too far-flung—the designs they produce will be innovative but not very likely to be useful. The adaptability requirement provides a greater likelihood that the system will be able to use the retrieved analogue to produce a reasonable redesign of the target.

To implement these two requirements, the mapping evaluation heuristics consider five separate portions of the mapping, based on five separate portions of the target device representation. The portions considered are the leaf nodes of Figure 4.2. *Process sequence* in the figure refers to the subgraph containing the primitive processes in the behavior or function and the links between them. *Individuals* refers to the everything other than the process sequence in the behavior or function representation. Figure 4.1 shows the process sequences for the behavior and function of the sprinkler irrigation system.

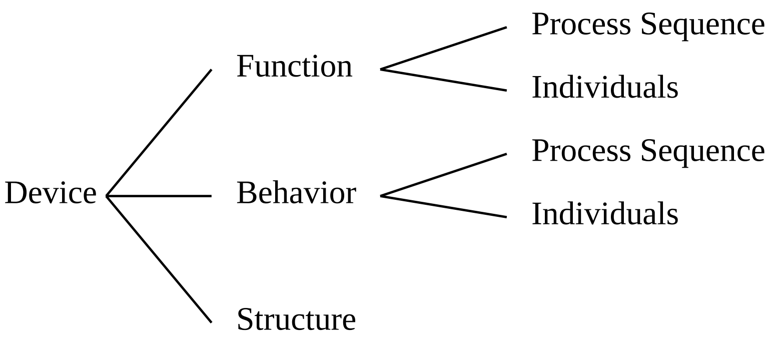


Figure 4.2: Portions of device representation considered by mapping evaluation heuristics

Rule #	*Portion of mapping*	*aspect*	*relation*	*threshold*
1	Function, entire	iso.	$\geq$	T_{FI}
2	Function, process sequence	sem. dist.	$\leq$	T_{FPD}
3	Behavior, process sequence	iso.	$\geq$	T_{BPI}
4a	Behavior, one proc. in proc. seq.	sem. dist.	$\geq$	$T_{BPDmismatch}$
4b	Behavior, rest proc. in proc. seq.	sem. dist.	$\leq$	$T_{BPDmatch}$
5	Behavior, individuals	sem. dist.	$\geq$	T_{BID}
6	Structure	sem. dist.	$\geq$	T_{SD}

Table 4.1: Summary of similarity metric conditions for innovative design

When spreading activation retrieves an intermediate concept, the mapping component first generates a general mapping of the entire target representation to the entire IC representation. Next, the mapping component divides the mapping into five portions based on the portions of the target representation shown in Figure 4.2, and it calculates the degree of isomorphism and semantic distance for each submapping. Semantic distance for a mapping is simply the average path distance between nodes which are mapped. Isomorphism for a mapping is the percentage of nodes and links from the graphs which are mapped to a node or link in the other graph. The mapping component then compares each rating to the applicable *similarity metric condition* for innovative design.

Table 4.1 lists the similarity metric conditions. These conditions can be summarized in English as:

(1) The isomorphism between the functions must be high.

(2) The semantic distance between the process sequences of the functions must be low.

(3) The isomorphism between the process sequences of the behaviors must be high.

(4) The semantic distance between the process sequence of the behaviors must be (a) high for one mismatching process, and (b) low for all of the rest.

(5) The semantic distance between the individuals of the behaviors must be high.

(6) The semantic distance between the structures must be high.

Conditions 1 and 2 enforce the "similar function" requirement by ensuring that the functions are isomorphic overall and match very closely in the processes of their functions. Conditions 5 and 6 enforce the "different structure and behavior" requirement by ensuring that the semantic distances between structure and the individuals of the behaviors is high. And conditions 3 and 4 enforce the adaptability requirement by making sure that only one process in the behavior sequence mismatches badly, and that the behavior representations are shaped similarly enough that replacement of the target's mismatching behavior with the base's is likely to lead to a working design.

The values for the particular thresholds in Table 4.1 can be determined in any number of ways. In IDA, the threshold values are determined empirically—i.e., by noticing what seem to be high and low values for semantic distance and isomorphism. However, one could also imagine simple techniques for learning these thresholds similar to the methods used to learn weights on links for spreading activation.

Mappings which meet the similarity metric are returned from KDSA as final analogies. Mappings which did not meet the similarity metric are passed along with a record of their evaluations to the search control component. The record of an evaluation consists of a list of the mapping's performance with regard to each similarity metric condition—whether it meets the condition, and if not the degree of miss.

Condition:

Newly retrieved IC has some portions $P_1, \ldots, P_n$ that are within some threshold of meeting the applicable condition from the similarity metric.

Action:

Add $P_1, \ldots, P_n$ to the list of promising components L_P;

Clear all activation in the knowledge base;

For each promising component P in L_P, make P a source of activation.

Figure 4.3: Activate Promising Concept Heuristic

4.2.3 Search Control

$KDSA_{ID}$'s search control heuristics guide the search toward likely analogues. They have two major purposes: (1) increasing the likelihood that future spreading activation searches will retrieve devices that score well in the mapping component's similarity metric, and (2) reducing the amount of search unnecessary to accomplishing purpose 1. They work by changing the activation levels of concepts in the KB and retrieval condition of the spreading activation mechanism based on past mapping evaluations.

There are four major search control heuristics in $KDSA_{ID}$—*activate promising concept*, *prune unpromising concept*, *cross-domain bridge*, and *modify retrieval condition*. Each heuristic is described below.

Activate Promising Concept

The applicability condition and action of the *activate promising concept* heuristic are shown in Figure 4.3. This heuristic strengthens the activation levels of portions of concepts which meet (or come close to meeting) conditions in the similarity metric. It also clears all other activation in the semantic network so that the subsequent spread of activation starts a new search.

This heuristic allows KDSA to search based on the *strengths* of near-analogies it has seen before. Each time the mapping component encounters a new IC with portions that look good, it sets up a new *intersection search* [Quillian, 1968] to find the intersection of

Condition:

Newly retrieved IC has some portions $P_1, \ldots, P_n$ which meet a threshold for being considered "unpromising" for the applicable condition from the similarity metric.

Action:

Clear all activation in $P_1, \ldots, P_n$;

Remove $P_1, \ldots, P_n$'s ability to receive activation.

Figure 4.4: Prune Unpromising Concept Heuristic

all promising portions it has seen previously. As the search progresses, and KDSA sees more and more portions of concepts which meet or nearly meet the similarity metric, the spreading activation searches conducted should start with sources which are more and more semantically close to a base that will meet the similarity metric.

This heuristic is task- and domain-independent. Since it says only to start new searches based on the strengths of what has been seen before, it is usable with any similarity metric.

Prune Unpromising Concept

The applicability condition and action of the *prune unpromising concept* heuristic are shown in Figure 4.4. This heuristic not only clears the activation level of unpromising portions of ICs; it also removes their ability to *receive* activation in the future. This of course also means that those unpromising portions will be unable to pass activation in the future, and that in turn will dampen the activation of the area of the KB surrounding the unpromising portions. This heuristic, then, will have two effects: 1) increasing search efficiency by having the search spread less activation in unpromising areas of the KB, and 2) increasing the quality of future retrieved concepts by keeping activation away from concepts semantically close to the unpromising ones. Like the activate promising concept heuristic, this heuristic is task-independent.

Cross-Domain Bridge

Condition:

Semantic distance between the structures of the newly retrieved IC and the target is below some (low) threshold;

Newly retrieved IC is an example of a concept A that is a *generic abstraction*.

Action:

Add A to the list of promising components L_P.

Clear all activation in the knowledge base.

For each promising component P in L_P, make P a source of activation.

Figure 4.5: Cross-Domain Bridge Heuristic

The applicability condition and action of the *cross-domain bridge* heuristic are shown in Figure 4.5. This heuristic provides KDSA with a method of jumping out of the target's domain when it is looking for a semantically distant analogy. It does this by capitalizing on concepts that are known instances of *generic abstractions*—generic descriptions which describe several concepts in different domains. For example, a "flow system" is a generic abstraction which may have as instances an irrigation system, the human circulatory system, a highway, a communications network, and other concepts that are often abstractly or metaphorically described as involving "flow".

Obviously, if the target itself is already an example of a generic abstraction, the KB may already have many built-in cross-domain analogies for the target, and there may be no need to do a long search for an analogy. This heuristic recognizes that the target, which is often poorly understood relative to other concepts in the knowledge base, may not yet be understood in terms of some global abstraction. However, some concepts near the target may be better understood as examples of generic abstractions which span several domains, and these concepts can be used by the search as a bridge into one of these other domains. In this way, KDSA is able to capitalize on previous analogical reasoning and analogical learning in the system, even if that analogical learning did not directly involve the target concept.

The heuristic is triggered when a retrieved IC is in the same domain as the target (here

Condition:

Semantic distance between the structures of the newly retrieved IC and the target is below some (low) threshold.

Action:

Modify the spreading activation retrieval condition to retrieve devices based only on high cumulative activation of the device's *behavior*.

Figure 4.6: Modify Retrieval Condition Heuristic

"same domain" is determined by semantic closeness to the structure of the target) and is known to be an example of a generic abstraction. The generic abstraction in these cases is added to the list of activation sources for KDSA's search, allowing future searches to immediately retrieve other instances of the generic abstraction.

This heuristic is not task-independent, since it focuses specifically retrieving semantically distant analogies. However, it is also not necessarily specific to the task of innovative design. Provided a criterion for "same domain"-ness can be constructed for the task (for innovative design this heuristic uses semantic distance on structure), the heuristic should be applicable in any task in which a semantically distant analogy is required.

Modify Retrieval Condition

The applicability condition and action of the *modify retrieval condition* heuristic are shown in Figure 4.6. This heuristic changes the metric spreading activation uses to retrieve the next concept to one which is based only on the *behavior* of devices. That is, after this heuristic is executed, a device will be retrieved by spreading activation if and only if the representation of its behavior is highly activated. The justification for this heuristic is as follows: if spreading activation is retrieving concepts based primarily on semantic closeness in structure, then structure is playing much too large a role in the retrieval process. Since the function representation usually provides very little on which to base a retrieval—function representations are often very small—the only portion of the device left on which to base the retrieval is the behavior. This heuristic is specific to the task of innovative design.

Sources of activation	Next retrieved concept	Evaluation	Search control heuristic(s)
sprinkler irrigation (entire concept)	plumbing system	no mismatching process, behavior and structure indiv. too semantically close	*activate promising concept, modify retrieval condition*
sprinkler irrigation (entire concept), plumbing system behavior and function	spraying fountain	fails on almost every condition	*prune unpromising concept*
none. (continuation of previous search)	digestive system	isomorphism in behavior process sequence too low	*activate promising concept*
sprinkler irrigation (entire concept), plumbing system behavior and function, digestive system structure	circulatory system	meets similarity metric	none—concept returned as final analogy

Table 4.2: Example of $KDSA_{ID}$ used to retrieve circulatory system as an analogy for the innovative redesign of sprinkler irrigation.

4.3 Example

This section presents a detailed example of a retrieval process guided by the heuristics described in this chapter. In the example, the goal is to redesign the notion of sprinkler irrigation using an analogy. The example is presented in a similar manner to the example in Section 1.3, the difference being that this example refers more specifically to the mapping and search control heuristics applied. This example is based on the behavior of IDA, but differs from IDA in the exact organization and behavior of the heuristics. Table 4.2 summarizes the execution of $KDSA_{ID}$ during the retrieval process in this example. That execution is described in detail below:

1. The nodes contained in the representation of SPRINKLER-IRRIGATION-SYSTEM are made sources of activation, and the spreading activation process begins.

2. After a few cycles of spreading activation, PLUMBING-SYSTEM is retrieved. This represents that portion of a municipal plumbing system which is concerned with the delivery of water from a water source to many homes. This device is mapped to SPRINKLER-IRRIGATION-SYSTEM, and the mapping is evaluated. The mapping passes the similarity metric on conditions 1 (since the abstract functions of the two devices, distributing water to plant roots, and distributing water to homes, have graphs representing them that are exactly the same shape), 2 (since both devices are performing DELIVERY), and 3 (since both devices have three-stage chains of behavior). The mapping fails on condition 4, since the mismatching processes—SPRAYING and POURING—are not semantically distant enough. The mapping also fails on conditions 5 and 6, since many structural components and substances are shared between the two devices (PUMP, PIPE, WATER, etc.).

   The mapping evaluation is passed to the search control component, where PLUMBING-SYSTEM's process sequence in behavior and entire function are evaluated by the *activate promising concept* heuristic as promising. Also, the *modify retrieval condition* heuristic is fired, changing spreading activation's retrieval condition to retrieve only on a device's behavior.

3. A new spreading activation search is started with sources SPRINKLER-IRRIGATION-SYSTEM (the entire concept), PLUMBING-SYSTEM's function, and PLUMBING-SYSTEM's behavior process sequence. The next system retrieved is SPRAYING-FOUNTAIN, which simply represents a fountain which sprays water into the air and collects the water in a pool. This concept fails every similarity metric condition except condition 3—it is dissimilar in function, and too similar in behavior and structure to SPRINKLER-IRRIGATION-SYSTEM to be even close to a useful analogue for redesign. The *prune unpromising concept* heuristic clears the activation of SPRAYING-FOUNTAIN and removes its ability to receive activation for the rest of the search.

4. The spreading activation search continues from where it left off, and the next concept

retrieved is DIGESTIVE-SYSTEM, because of its behavioral and functional similarities to the previously activated PLUMBING-SYSTEM. This concept meets every condition in the similarity metric except 3 and 4—the DIGESTIVE-SYSTEM satisfies the "similar function" and "different structure and behavior" criteria, but the behavior simply does not map well enough for the system to know how to perform the transfer step. The evaluation is passed to the promising concept, where the *activate promising concept* heuristic makes the entire concept of DIGESTIVE-SYSTEM a source of activation. Spreading activation starts a new search.

5. The next concept retrieved is CIRCULATORY-SYSTEM, because of its semantic closeness to DIGESTIVE-SYSTEM. This concept meets the similarity metric: it has a similar function, DISTRIBUTE, a semantically distant structure and behavior, the sequence of processes corresponds well in all but one case, and that one case—the mapping of CIRCULATORY-SYSTEM's DIFFUSION to SPRINKLER-IRRIGATION-SYSTEM's SPRAYING—provides a candidate transfer for the analogical inference. KDSA returns this mapping as the final analogy.

Based on this analogy, the transfer step produces a new behavior graph with the SPRAYING process from SPRINKLER-IRRIGATION-SYSTEM's behavior replaced with DIFFUSION. This "design" is of course extremely abstract and far from complete, but it is conceivable that it could cause a human designer or a sophisticated computer design system to generate the idea of drip irrigation, where water is delivered to plant roots by slowly dripping out of porous "capillary" hoses.

4.4 Discussion

There is a key general issue raised by the model of analogy retrieval for design presented here: the issue of the level of detail needed in the design process. This issue is encountered in two ways in the model presented above. First is the level of detail in the specification of the design. $KDSA_{ID}$'s design system is presented with a very abstract and simple design specification: "Redesign device x in an innovative way". Even with this very high level of specification, the system does get an abstract functional specification for the new device (it must be the same as the function of device x), and open-ended constraints on the behavior and structure of the new device (they must be different from those of device x). But there of course are many other more specific constraints possible in a design specification. A redesign specification can (and often does) include specific shortcomings of the existing device to overcome, specific behavioral or structural constraints, or a functional specification which is different in some way from the functional specification of the existing device. The general way of handling more detailed specifications in this model is clear: make *any* requirements for the new design be a required part of the analogy, in the same sense that the function is a required part of the analogy in $KDSA_{ID}$. So if there is an extra behavioral specification for the redesign, a candidate analogue must match that behavioral specification as well as the function in order to be useful for creative design. This only gives a partial answer, however. There are several specific issues which must be dealt with before we have a general solution to the problem of more detailed and constrained design specifications. For example, what conditions determine whether a base "matches" a behavioral or structural constraint? And, how can the system infer whether the behavioral or structural constraint is analogously satisfied in the base if it is not explicitly represented?

The second way this model encounters the issue of detail is in the representation of devices. The assumptions about representation described earlier in this chapter specify an abstract level of description for devices. This type of representation, while useful for some types of reasoning (explanation, natural language processing), is not useful for other types of reasoning often performed with devices (simulation, diagnosis, detailed design). However, this abstract level of representation is important in the process of innovative design by analogy. It is essential for high-level conceptual design that the high-level features of the device be prominent in the representation. The alternative—representing a device's behavior by a set of quantitative or qualitative parameters and equations—could bog down the analogy process by causing it to focus on unnecessary detail. It is possible to use more detailed representations of devices in the innovative redesign model described in this chapter, but the best way of using them is to automatically derive more abstract representations based on the detailed representations before creating the analogies, i.e., to add the following steps to the analogical design process: (1) abstract the representation of the target, (2) abstract the representation of each candidate base concept as it is retrieved, (3) generate a design by analogy at the abstract level, and (4) specialize the abstract design. This approach allows the analogy to focus on high-level process and configuration issues, rather than conditions on low-level variables.

Chapter 5

Theoretical Analysis

If KDSA is to be a successful mechanism for general analogy retrieval in large knowledge bases, it must overcome the computational difficulties other retrieval methods (e.g., standard spreading activation) have when searching through large amounts of information. That is, it must be able to retrieve semantically distant analogies while only examining a small portion of the total KB. Ideally, the tractability of KDSA finding distant analogies in large KBs would be tested empirically—i.e., by running KDSA with various target concepts in an actual large knowledge base and measuring the amount of time taken to retrieve semantically distant analogies. However, since the development of very large knowledge bases is still in its infancy and actual KBs of this type are not yet ready for such testing, the empirical tests will have to be done in a relatively small KB (see chapter 6). It is possible, though, to analyze the expected performance of KDSA theoretically, examining performance not only in large KBs, but also in different general KB configurations and problem types.

This chapter presents such a theoretical model, and evaluates KDSA's behavior based on the predictions of that model. The model quantitatively predicts the computational effort involved in retrieving analogies based on a number of parameters, including semantic distance required of the analogy, KB size, effort of each of KDSA's beacon searches, and benefit of each of KDSA's beacon searches. The model shows that KDSA's use of beacons to guide search does allow it to avoid examining a large portion of the total KB.

5.1 Assumptions of the Model

Let $S = (V, E)$ be a semantic network knowledge base with nodes V and links E. The size of the knowledge base will be measured as the number of nodes $K = |V|$. The branching factor of the knowledge base is $b = |E|/|V|$, and is assumed to be constant throughout the entire KB, i.e., each node in V is assumed to be adjacent to exactly b links. The links in the knowledge base are assumed to be randomly and uniformly distributed across the nodes. I.e., the semantic network can be thought of as having been constructed with the following algorithm:

> **for** each $v \in V$ **begin**
>
>   pick b nodes randomly from V and create a link from v to each of them;
>
>   add the new links to E;
>
> **end**.

All concepts or types will be defined in the semantic network as subgraphs $(V' \subset V, E' \subset E)$ of S.

5.1.1 Modeling Spreading Activation in a Graph

Standard spreading activation will be modeled as marker passing through S starting at a target graph G_T, and finishing when some designated base graph G_B, which meets the desired similarity metric, is retrieved. The marker passing in this theoretical model will be strictly breadth-first. That is, the activation level of a node is simply a boolean value rather than a numeric value, and each marked node will pass a mark to all of its neighbors in a cycle. There is no notion of decay in this model, and no selective spread of activation along only certain links or link types.

The *spreading activation path distance* between two graphs G_1 and G_2, $d(G_1, G_2)$ will be defined as the number of spreading activation cycles needed to retrieve G_2 after making G_1 (and only G_1) a source.

5.1.2 Modeling KDSA

KDSA between G_T and G_B will be modeled as a sequence of standard SA searches along a sequence of promising beacon graphs, $G_T, G_1, G_2, \ldots, G_{n-1}, G_B$. That is, KDSA will consist of first an SA search from G_T to G_1, followed by an SA search from G_1 to G_2, and so on until G_B is retrieved. The benefit of an individual promising search, $\Delta d(G_i, G_{i+1}, G_B)$, will be defined as how much closer to the base G_{i+1} is than G_i:

$$\Delta d(G_i, G_{i+1}, G_B) = d(G_i, G_B) - d(G_{i+1}, G_B)$$

5.2 Time Cost of Standard Spreading Activation in Graphs

The computational cost of a spreading activation search will be assumed to be the number of nodes marked during that search. This ignores the cost of retrieving and mapping concepts which meet the spreading activation retrieval condition during the course of the search. However, it is reasonable to expect that this cost will be directly proportional to the number of nodes marked, i.e., that there will be one concept retrieved and mapped for every $1/n$ marked nodes in the semantic network, for some n.

Let n_i be the number of previously unmarked nodes that are activated during level i of a spreading activation search. During level i of a search, spreading activation will attempt to mark bn_{i-1} nodes. However, some of these nodes will be nodes that were already marked in the $i-1$ previous levels of the search, and some will also be duplicates within the current level. These two sources of duplication will be handled separately, first by calculating the number of unique nodes marked in level i, and then by calculating the number of those nodes which have not been marked in previous levels of the search.

The number of unique nodes activated by level i will be $(K^{bn_{i-1}} - (K-1)^{bn_{i-1}})/K^{bn_{i-1}-1}$ as derived from the following lemma:

Lemma 5.1 *The expected number of unique nodes marked when n total are marked is* $(K^n - (K-1)^n)/K^{n-1}$.

Proof: Let D_n be a random variable representing the number of unique nodes marked after n total have been marked. Since nodes in the semantic network are assumed to be randomly distributed throughout the graph, the nth node marked will have a D_{n-1}/K probability of being one of the D_{n-1} unique nodes seen so far. Given this probability, the expected value of D_n can be calculated in terms of the expected value of D_{n-1}:

$$\begin{aligned} E(D_n) &= E\left(1 - \frac{D_{n-1}}{K} + D_{n-1}\right) \\ &= E\left(D_{n-1}\left(1 - \frac{1}{K}\right) + 1\right) \\ &= E\left(D_{n-1}\right)\left(1 - \frac{1}{K}\right) + 1 \end{aligned}$$

Using this formula for $E\left(D_n\right)$, we can prove the lemma by induction:

base case: When $n = 1$, $E\left(D_n\right) = \frac{K-(K-1)}{1} = 1$.

induction case: If the induction hypothesis holds for $E\left(D_{n-1}\right)$, then

$$\begin{aligned} E\left(D_n\right) &= E\left(D_{n-1}\right)\left(1 - \frac{1}{K}\right) + 1 \\ &= \frac{K^{n-1}-(K-1)^{n-1}}{K^{n-2}}\left(1 - \frac{1}{K}\right) + 1 \\ &= \frac{K^n-(K-1)^n}{K^{n-1}} \qquad \square \end{aligned}$$

Since there are bn_{i-1} total nodes marked during level i of a spreading activation search, from the lemma we can expect $\left(K^{bn_{i-1}} - (K-1)^{bn_{i-1}}\right)/K^{bn_{i-1}-1}$ of those to be unique.

Of those unique nodes, some percentage will already have been marked in the $i-1$ previous cycles of spreading activation. Since nodes are randomly distributed in the semantic network, this percentage is simply the ratio of nodes already marked to the total number of nodes in the KB, or $\sum_{j=0}^{i-1} n_j/K$. So we can express the number of unique previously unmarked nodes marked in cycle i of a spreading activation search as:

$$\begin{aligned} n_i &= \left(\frac{K^{bn_{i-1}} - (K-1)^{bn_{i-1}}}{K^{bn_{i-1}-1}}\right)\left(1 - \frac{\sum_{j=0}^{i-1} n_j}{K}\right) \\ &= K\left(1 - \left(\frac{K-1}{K}\right)^{bn_{i-1}}\right)\left(1 - \frac{\sum_{j=0}^{i-1} n_j}{K}\right) \end{aligned}$$

Let $N_i = \sum_{j=0}^{i} n_j$ be the *total* number of nodes marked by a spreading activation search up to and including cycle i. Then

$$N_i = N_{i-1} + n_i$$

$$= \quad N_{i-1} + K\left(1 - \left(\frac{K-1}{K}\right)^{b(N_{i-1}-N_{i-2})}\right)\left(1 - \frac{N_{i-1}}{K}\right)$$

So the total number of nodes marked by a spreading activation search up to and including cycle i is given by the recurrence relation:

$$N_i = \begin{cases} 0, & i = 0 \\ |G_T|, & i = 1 \\ N_{i-1} + K\left(1 - \left(\frac{K-1}{K}\right)^{b(N_{i-1}-N_{i-2})}\right)\left(1 - \frac{N_{i-1}}{K}\right), & i > 1 \end{cases} \tag{5.1}$$

(Here $|G_T|$ represents the number of nodes contained in G_T.)

This recurrence equation represents the total computational effort in a spreading activation search of depth i. The computational cost of searching from G_T to G_B by standard spreading activation will then be $N_{d(G_T,G_B)}$. This recurrence equation does not have a known closed form, but even in its present form it allows us to inspect and analyze the behavior of SA and KDSA under a variety of general problem types.

5.3 Time Cost of KDSA

As noted before, KDSA is modeled as a series of standard SA searches. The total cost of KDSA between G_T and G_B will depend on the cost of each beacon subsearch, $N_{d(G_i,G_{i+1})}$, and the benefit gained from each beacon subsearch, $\Delta d(G_i, G_{i+1}, G_B)$. In particular, the cost of a KDSA search from G_T to G_B is given by the rather complex formula:

$$KDSA(G_T, G_B) = \sum_{i=0}^{\sum_{j=0}^{i} \Delta d(G_j, G_{j+1}, G_B) \geq d(G_T, G_B)} N_{d(G_i,G_{i+1})} \tag{5.2}$$

In other words, the cost of KDSA is determined by summing the costs of individual beacon searches between G_i and G_{i+1}, until the sum benefit of the beacon searches so far (in terms of path distance) equals or exceeds the path distance between G_T and G_B.

This formula can be simplified significantly by making the assumption that each beacon search is of a constant depth d_p, and has a constant benefit Δd_p. In this case, the number of searches will be simply $d(G_T, G_B)/\Delta d_p$, and the cost of each search will be N_{d_p}:

$$KDSA(G_T, G_B) = \frac{d(G_T, G_B)}{\Delta d_p} N_{d_p} \tag{5.3}$$

Most of the analysis of the following section (5.4) shows the behavior of KDSA assuming only the simplified formula 5.3. However, the last two subsections relax the assumptions about constant depth and constant benefit, respectively, and show behavior of KDSA based on equation 5.2.

5.4 Theoretical Results

The analysis above allows us to examine the predicted performance of KDSA under a wide variety of circumstances, and compare its performance to standard spreading activation. Each section below describes the behavior of KDSA and SA as a specific parameter of the model—semantic distance of the analogy, size of the KB, etc.—changes. For each section, a graph will demonstrate the behavior of KDSA compared to standard SA. The time cost of standard SA in these graphs is calculated as $N_{d(G_T,G_B)}$ by formula 5.1, and the time cost of KDSA is calculated as $KDSA(G_T, G_B)$ from either formula 5.3 (used unless otherwise stated) or formula 5.2. In each case, the values of other parameters were chosen to reflect a "typical" case of interest to this thesis—i.e., finding a semantically distant analogy in a large knowledge base—and to reflect conservative values for the cost and benefit of each KDSA beacon search—i.e., the cost/benefit ratio of each beacon search was high. In each section, the behavior of KDSA in other, "non-typical" cases is also discussed.

5.4.1 Time cost as search depth grows

Figure 5.1 graphs the costs of SA and KDSA as they vary with $d(G_T, G_B)$, the path distance between the target and the base graphs. For this graph, the knowledge base size was set at 1,000,000 nodes, the branching factor was set at 4, each KDSA beacon search was assumed to be of path length 7, and each beacon search was assumed to lead to a graph which was path distance 2 closer to the base graph. For this graph, both the cost and benefit of a beacon search is assumed to be constant throughout the entire KDSA process. These choices of parameter values correspond to a large KB where the benefit of each beacon search seems low compared to the effort involved.

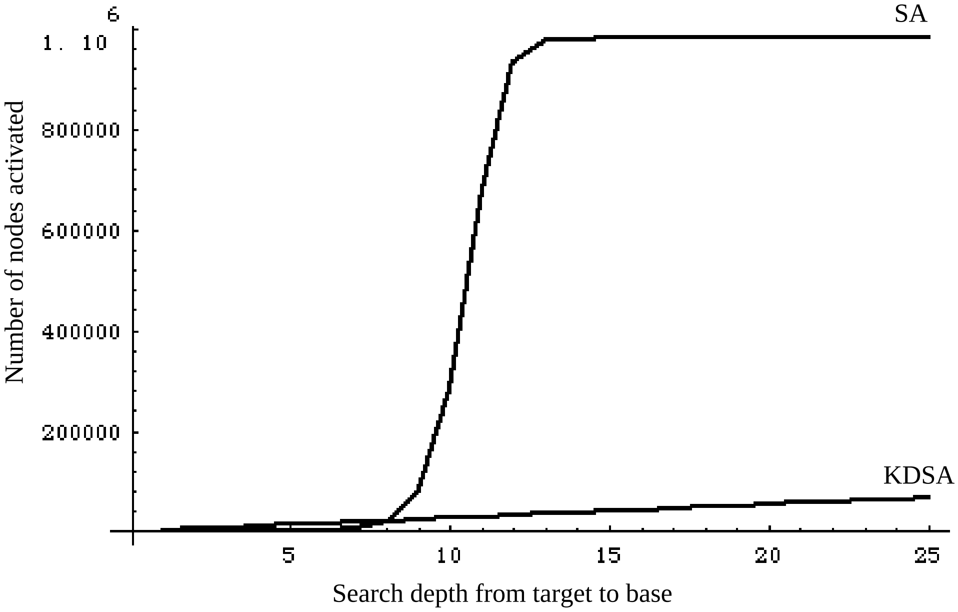


Figure 5.1: Cost of search as search depth from target to base ($d(G_T, G_B)$) grows. $K = 1,000,000$, $b = 4$, $d_p = 7$, $\Delta d_p = 2$, $|G_T| = 1$.

Since KDSA is linear only in the path distance between G_T and G_B, and not the total cost of the SA search between those two graphs, its cost increase is linear with the path distance. The cost of SA, on the other hand, increases roughly exponentially with path distance until the algorithm starts encountering a significant number of nodes it has marked before, at which time the graph asymptotically approaches some number less than the total KB size, K. The conclusion we can draw here is that, when a very high degree of semantic distance between analogies is required, KDSA is likely to be much less computationally expensive than standard SA.

For semantically close analogies, Figure 5.1 shows KDSA as having higher cost than SA. However, this is largely because of the model's unrealistic assumption that KDSA always does $d(G_T, G_B)/\Delta d_p$ beacon searches, even in cases when it would actually encounter the base before it encountered a beacon concept. So, for example, Figure 5.1 should actually show KDSA having equal cost to SA when $d(G_T, G_B)$ is 7 or less, since KDSA would encounter the base concept (which is 7 or fewer spreading activation cycles away) before it encountered the first beacon concept (which is assumed to always be 7 spreading activation cycles away in Figure 5.1).

Even with a correction for bases which are closer than the beacon search distance, it is still possible for KDSA to be more expensive than SA. There are two reasons for this. First, the model assumes (realistically) that KDSA's beacon searches are not along the ideal path to the base graph. Second, in graph search, there is an overhead involved in doing multiple graph searches—each new beacon search visits and marks nodes which were already marked in previous beacon searches. Figure 5.1 shows that, even when the search:benefit ratio of the beacon searches is high (it is 3.5:1 in the graph shown), and therefore the amount of overhead involved in KDSA is also high, KDSA is still able to avoid searching a large portion of the knowledge base which is searched by standard SA. In particular, Figure 5.1 shows that it is almost exactly as expensive to do four SA searches of depth 7 than it is to do one SA search of depth 8, even considering that the search is graph search.

This is because of the overhead inherent in doing many of these separate searches—unlike the abstraction planning model, where you are constantly moving "forward" and expanding new states—because you are double-counting many nodes which you've seen two or three times before.

5.4.2 Time cost as KB size grows

Figure 5.2 graphs the cost of SA and KDSA as K, the number of nodes in the knowledge base, varies. For this graph, most parameters other than KB size were set to the same values as in Figure 5.1: the KB branching factor is 4, each KDSA beacon search is of depth 7, each beacon search gets path distance 2 closer to the base, and the size of the target graph is assumed to be 1 node. The distance between the target and the base, $d(G_T, G_B)$, is set to 13—we can see from Figure 5.1 that this distance corresponds to a semantically distant analogy, in the sense that standard spreading activation will examine a high percentage of the entire knowledge base before retrieving a base of this distance.

Figure 5.2 shows the cost of SA as roughly linear in the size of the knowledge base. Since SA is examining most of the knowledge base before retrieving the analogy, it seems

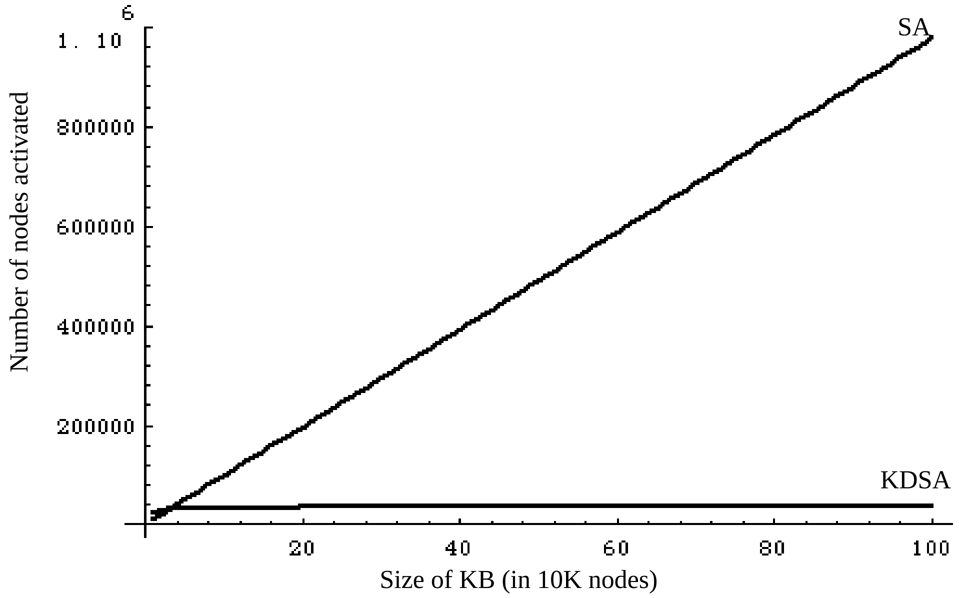


Figure 5.2: Cost of search as knowledge base size (K) grows. $d(G_T, G_B) = 13$, $b = 4$, $d_p = 7$, $\Delta d_p = 2$, $|G_T| = 1$

natural that the cost of SA would be directly proportional to the size of the KB. KDSA, on the other hand, appears roughly logarithmic in the size of the KB. This also makes intuitive sense: since KDSA is conducting only a series of 7-deep beacon searches, and since the branching factor involved in those searches is assumed to remain constant as the KB size grows, KDSA should see roughly the same number of nodes in each of those 7-deep searches, regardless of the size of the KB. In other words, the nodes being added are generally those which will not be reached by the small sub-searches conducted by KDSA.

One key assumption behind Figure 5.2 is that the size of the knowledge base grows without affecting the KB's branching factor. The model assumes that each node added to the KB is adjacent to exactly the same number of links as every other node in the KB. It is conceivable that, as more and more knowledge is entered into a KB, more and more will be known about the relationships between existing concepts in the KB, and thus the number of links per concept in the KB will grow. On the other hand, it is equally plausible that the branching factor will actually shrink as a KB grows: it is possible that the concepts which are best understood and thus most highly connected to other concepts will be entered into the KB first. The actual relationship of branching factor to KB size is one that can be investigated empirically and is beyond the scope of this thesis. It is important to point out, though, that the shape of the graphs in Figure 5.2 is dependent on the assumption of constant branching factor.

Another simplification of the model reflected in Figure 5.2 is the assumption that the added knowledge in the KB will not effect the sequence of beacon concepts used by KDSA to guide its search. That is, the model used to generate Figure 5.2 assumes that KDSA will use the same sequence of beacon concepts $G_1, \ldots, G_{n-1}$ to guide it to the base, regardless of the size of the KB. This simplification probably unrealistically inflates the time cost of KDSA. In reality, we would expect the growth of the knowledge base to add some additional concepts which would be evaluated as promising by KDSA, and some of those would be encountered before a beacon used in the search of a smaller KB, and thus would be used as beacons themselves. It seems reasonable to expect that the benefit of these new beacons would be no worse (or better) than the benefit of the other beacons; the only parameter that new beacons would change is the average distance between beacons in the KDSA search. We could expect, then, that as K grows, d_p will shrink, further decreasing the search time of KDSA.

5.4.3 Time cost as cost of promising searches changes

Figure 5.3 graphs the cost of SA and KDSA as d_p, the semantic distance between beacons in the KDSA search, varies. For this graph, most parameters other than d_p were set to the same values as in the previous graphs: the KB size is $1,000,000$ nodes, the KB branching factor is 4, the path distance between target and base is 13, each beacon search gets path distance 2 closer to the base, and the size of the target graph is assumed to be 1 node. The cost of standard spreading activation of course does not change as d_p is varied, since it is not using search control.

This graph shows that KDSA can be much more efficient than standard SA even when the cost of each beacon search is very high relative to the benefit gained from it. In Figure 5.3, each beacon search gets KDSA only path distance 2 closer to the base, but even when

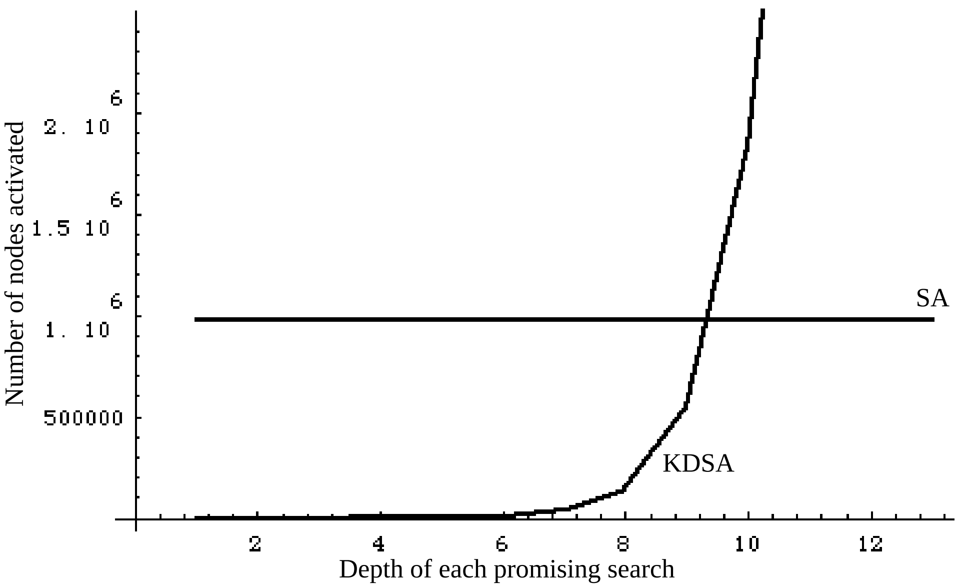


Figure 5.3: Cost of search as beacon search depth (d_p) grows. $K = 1,000,000$, $d(G_T, G_B) = 13$, $b = 4$, $\Delta d_p = 2$, $|G_T| = 1$

it has to search as many as 8 levels to get this benefit of 2 levels for each beacon search, KDSA is still many times more efficient than standard SA.

5.4.4 Time cost as benefit of promising searches changes

Figure 5.4 graphs the cost of SA and KDSA as Δd_p, the benefit (in terms of semantic distance) gained by each beacon search, varies. As before, for this graph, most parameters other than Δd_p were set to the same values as in the previous graphs. The cost of standard spreading activation does not change as Δd_p is varied, since it is not using search control.

This graph shows a similar qualitative result to the last one—KDSA is much more likely to be tractable in finding semantically distant analogies in a large KB, even when the cost/benefit ratio of KDSA's beacon searches is high. Figure 5.4 shows that when a search of depth 7 only shrinks the expected distance to the base by 1 or even 0.5, KDSA is still many times more efficient than standard SA.

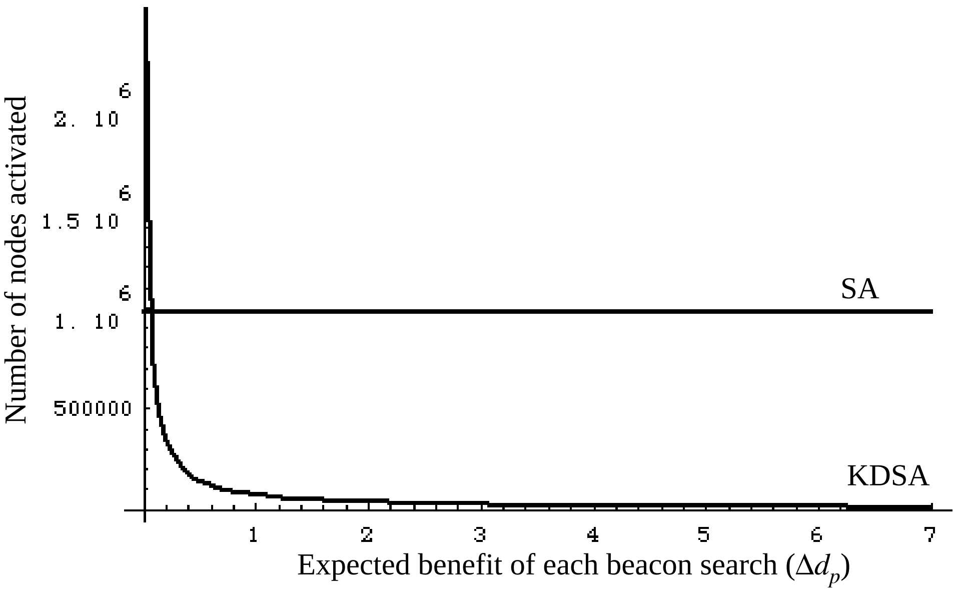


Figure 5.4: Time cost as benefit of beacon search (Δd_p) grows. $K = 1,000,000$, $d(G_T, G_B) = 13$, $b = 4$, $\Delta d_p = 2$, $|G_T| = 1$

5.4.5 Time cost with different distributions of promising concepts

The situations graphed so far have all relied on there being a constant cost and benefit to each of KDSA's beacon searches. This may be a reasonable simplifying assumption for the purposes of a general evaluation of KDSA's tractability, but of course the constant cost and benefit assumption will not hold true in practice. What about different distributions of beacons in the knowledge base? For example, how much will one difficult beacon search during the course of KDSA affect its tractability? The theoretical model allows us to answer questions such as these by using the more general equation describing KDSA's time complexity, equation 5.2.

Figure 5.5 shows the behavior of KDSA under one of these other possible distributions of promising concepts in the KB. It deals with the case where all beacon searches in the KB are of constant depth (7) except one; the cost of KDSA is graphed against the depth of this "anomalous" beacon search. The other parameters for this figure are set to the same values as in the previous graphs. The cost of standard SA remains constant in this graph,

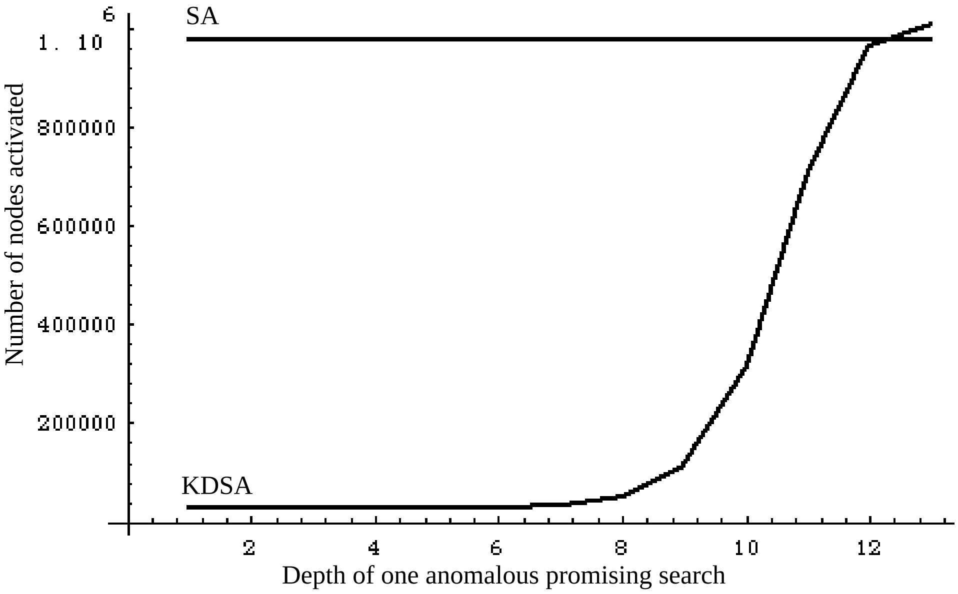


Figure 5.5: Cost of search as depth of one anomalous beacon search ($d(G_i, G_{i+1})$, for some i) grows. $K = 1,000,000$, $d(G_T, G_B) = 13$, $b = 4$, $d(G_j, G_{j+1}) = 7$ for all $j \neq i$, $\Delta d_p = 2$, $|G_T| = 1$

as it did in the previous two, because it does not include search control.

The lesson to be taken from this graph is similar to the lesson of the previous two: KDSA is more tractable than standard SA even when cost of beacon searches is relatively high. In particular, in this graph we can see that KDSA remains more efficient than SA even when the depth of the anomalous beacon search approaches the total semantic distance between the target and the base (13). It seems reasonable to conclude that isolated difficult beacon searches during the course of a KDSA search will not affect the tractability of the algorithm, provided they are not *too* difficult—i.e., as difficult as an SA search from target to base.

5.4.6 Time cost with different distributions of beacon concept benefit

So far the situations considered have all assumed that every concept which KDSA uses as a beacon during the course of its search is actually closer to the base than the previous one. This would be nice if it holds in practice, but a more likely scenario is that KDSA's

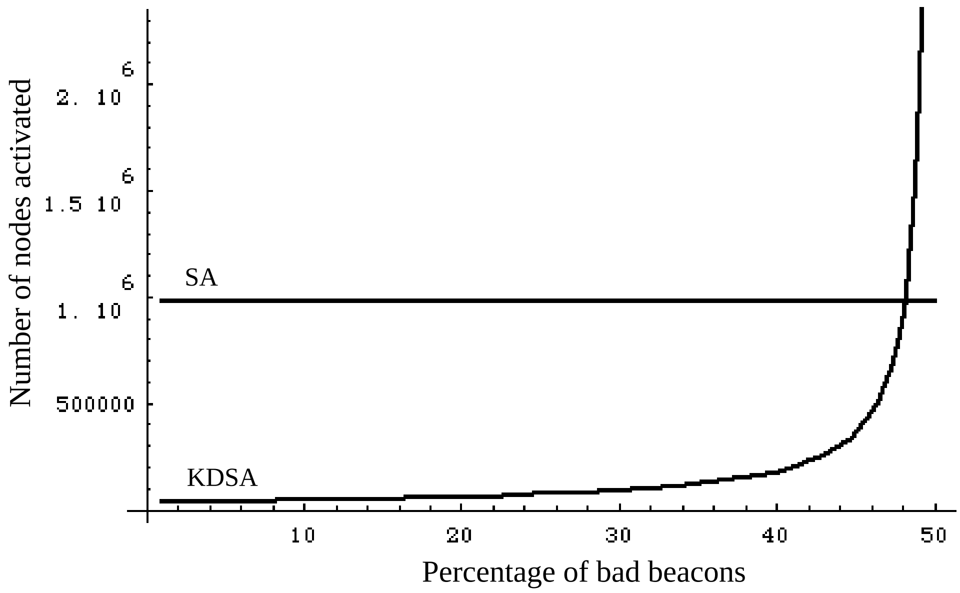


Figure 5.6: Cost of search as percentage of bad beacons ($\frac{100}{m}$) grows. $K = 1,000,000$, $d(G_T, G_B) = 13$, $b = 4$, $d(G_j, G_{j+1}) = 7$, $\Delta d_p = 2$, $\Delta d_{bad} = 2$, $|G_T| = 1$

heuristics will not be perfect—some concepts evaluated as beacons by KDSA will actually place the search further away from the base. The effect of these "bad beacons" can be evaluated by the theoretical model.

Suppose that $\frac{1}{m}$th of the beacons encountered actually take KDSA Δd_{bad} levels *away* from the base. The remainder of the beacons still get the search Δd_p levels closer to the base. Then for every m beacons encountered, KDSA will have gotten on average $(m-1)\Delta d_p - \Delta d_{bad}$ levels closer to the base. In this case, equation 5.2 is modified to give a new cost for KDSA:

$$KDSA(G_T, G_B) = \frac{md(G_T, G_B)}{(m-1)\Delta d_p - \Delta d_{bad}} N_{d_p}$$

Figure 5.6 shows time to retrieve an analogy as the percentage of bad beacons in KDSA's search ($\frac{1}{m}$) varies. In this graph, each good beacon is assumed to take KDSA 2 levels closer to the base, and each bad beacon is assumed to take KDSA 2 levels away from the base ($d_p = d_{bad} = 2$). The other parameters are chosen to be the same as in previous graphs. The cost of standard spreading activation remains constant in this graph because SA uses no beacons.

Figure 5.6 demonstrates that KDSA should be robust in the face of bad beacons, provided that most of the beacons it encounters do in fact bring the search closer to the base. If KDSA's evaluation heuristics are poor enough that the expected benefit of a beacon is very close to 0 (or less than 0), then KDSA can spend its time thrashing between beacons, never finding a final base.

5.5 Discussion

A number of issues raised by the analysis and its assumptions should be discussed in more detail. First is the issue of the knowledge base's branching factor. The branching factor will determine how quickly (i.e., in how few cycles) spreading activation will mark (nearly) all of the nodes in the knowledge base. The higher the branching factor, the faster N_i converges to a value close to the size of the KB, k. If the branching factor is high enough that a spreading activation search to depth d_p will mark the entire KB, then KDSA will take roughly $\frac{d(G_T,G_B)}{\Delta d_p}$ times longer to reach the base concept than standard SA. That is, KDSA will mark the entire KB $\frac{d(G_T,G_B)}{\Delta d_p}$ times to get to the base, while SA will mark the entire KB once. This is what Figure 5.7 shows: with the base semantic distance 13 from the target, and each beacon search getting 2 closer to the base, $KDSA(G_T, G_B)$ converges to a value about 6.5 times larger than SA's cost in its search, N_{13}, as the branching factor grows large.

All this tells us is that if KDSA must search every node in the KB to find a beacon concept, it is much worse than standard SA. That is trivially true. But this sort of scenario is unlikely to occur in practice. The whole point of identifying beacon concepts is to find concepts that can help direct the search *well before* marking the entire KB. If the branching factor is such that a search of d_p depth ends up marking the entire KB, the value used for d_p is probably unrealistically large. In practice, as the knowledge base's branching factor grows and the knowledge represented becomes more densely connected, we would expect the semantic distance between beacon concepts to shrink. Thus, the situation presented in

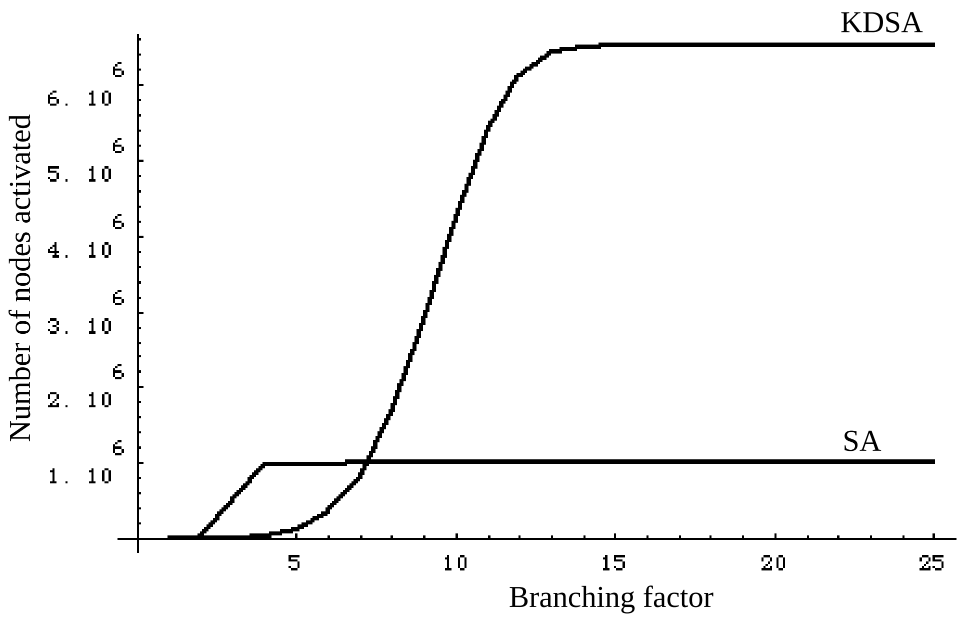


Figure 5.7: Cost of search as branching factor (b) grows. $K = 1,000,000$, $d(G_T, G_B) = 13$, $d(G_j, G_{j+1}) = 7$, $\Delta d_p = 2$, $|G_T| = 1$

Figure 5.7 is unrealistic since it assumes b grows without corresponding drops in the values of d_p and $d(G_T, G_B)$.

A second issue that needs further discussion is the cost of mapping retrieved analogies. Mapping can be a very time-consuming portion of the analogy process. There is no explicit consideration of the time cost of mapping in the analysis presented in this chapter, but it turns out that the model should reflect the relative cost of mapping in the two retrieval approaches well. In particular, since all retrieved concepts are mapped in either KDSA or SA[1],and since the number of retrieved concepts is determined by the number of nodes marked, we can expect the cost of mapping in a search to be proportional to the number of nodes marked. So the graphs in Figures 5.1 through 5.7 can be seen as approximations of mapping time as well as approximations of search time.

A third issue that has not been addressed in the analysis up to now is search control. Spreading activation, both by itself and as a component of KDSA, is modeled as a blind

[1]Even though SA has no search control that is guided by mapping, it needs to map retrieved concepts to determine whether they meet the similarity metric

search along the links of the semantic network. But in practice, SA's search can controlled, both in KDSA (by KDSA's search control component) and in other approaches [Rau, 1987a; Cohen and Kjeldsen, 1987]. However, these search control approaches easily fit within the model presented here of SA as blind search. The search control used in Rau's and Cohen and Kjeldsen's approaches (discussed in detail in Chapter 7) is primarily designed to reduce the branching factor of the search. This can be dealt with in the theoretical model simply by choosing b to be the search-control-reduced branching factor of the SA search rather than the KB's overall branching factor. If there is any inaccuracy of the analysis in ignoring this type of search control, it is probably to KDSA's disadvantage: KDSA will be able to take advantage of the static types of search control developed by Rau and Cohen and Kjeldsen, but it is also able to use *dynamic* search control which depends on the current state of the search.

Finally, it is important to identify differences between this analysis and other similar search analyses, particularly Knoblock's analysis of two-level hierarchical planning [Knoblock, 1991]. Both this analysis and Knoblock's analysis are concerned with evaluating the effort saved by identifying intermediate states in the search which can be taken as new search starting points, i.e., points at which all other active nodes in the search can be erased. But this analysis differs from Knoblock's in four important ways:

- Knoblock deals with search in a tree, while this work analyzes search in a graph. So for this analysis, the effort required to search to depth i is given by the complicated recurrence relation in Equation 5.1; for Knoblock's problem, that effort is b^i.

- Because Knoblock's analysis considers tree search, the issue of revisiting already-visited nodes does not come into play in abstraction-guided search. But it is a cost which must be considered in KDSA.

- In Knoblock's analysis, all intermediate states are along the optimal path to the goal, where the optimal path is the one that would be found by blind search at the base level. There is no possibility considered that the a blind search at the bottom level might be able to find a path to the goal which is much shorter than the path which passes through the intermediate states. This analysis deals with that possibility; in fact, the situations looked at in section 5.4 all assume that the intermediate states are *far* from the optimal path.

- Knoblock considers the amount of work required to generate the set of intermediate states. This analysis assumes that the system can simply recognize them when it encounters them, and assumes that the overhead involved in this recognition is negligible compared to the effort involved in the search.

Chapter 6

Implementation and Experiments

The previous chapter showed that KDSA is much more efficient than standard spreading activation at retrieving semantically distant analogies under a number of theoretical assumptions. While these theoretical results give us reason to expect KDSA to be a tractable analogy retrieval mechanism in an actual large knowledge base, they are still theoretical. A complementary method of evaluating KDSA is to test an actual implementation of the model under a variety of knowledge base conditions. This chapter describes IDA, an implementation of KDSA with heuristics for innovative design. The chapter first describes some important details about the implementation itself. Next it presents the results from a set of experiments run with IDA which agree with the theoretical model's prediction of large knowledge base tractability. Finally, it discusses some of IDA's more surprising behaviors.

6.1 Implementation

IDA is written in BB1 [Hayes-Roth, 1990], an architecture for general intelligent behavior. There are many important facets of BB1 that allow it to support a wide variety of reasoning and interacting with the world. For the purposes of this implementation, the most important are: 1) representation of procedural knowledge as modular *knowledge sources*, 2) a *decision cycle*, in which knowledge sources are chosen for execution based on applicability and importance to the current situation, and 3) an interface to the outside world.

These features of BB1 allow IDA's implementation to look very much like the conceptual framework shown in Figure 1.2.[1]

There are three major components to the software implementing IDA:

(1) The mapping evaluation and search control components, which are implemented as BB1 knowledge sources. Knowledge sources are separate and modular reasoning operations which specify the conditions under which they may legally execute and the action to be taken when they do execute. The mapping and search control knowledge sources compete for execution with any other knowledge sources active in BB1 at the time.

(2) The graph matcher. This is a BB1 library function that produces the best mapping between two graphs given specifications by the user. The specifications determine the relative levels of importance of a) maximizing isomorphism and b) minimizing semantic distance in the mapping. The graph matcher is used by a mapping evaluation knowledge source to produce a mapping between IC and target whenever a new IC is retrieved by spreading activation. The exact behavior of the graph matcher is given in [Wolverton and Brownston, 1994].

(3) The spreading activation mechanism. This is a collection of BB1 library functions which supports many different specific flavors of spreading activation. Its behavior is also described in [Wolverton and Brownston, 1994], and its use in IDA is described in the next section.

6.1.1 Spreading Activation

One difference between the framework of Figure 1.2 and the implementation is that IDA does not have spreading activation as a separate architectural component. BB1 currently does not include spreading activation as part of its memory model, so implementing KDSA within BB1 as it currently stands necessitated calling a spreading activation mechanism from within a knowledge source. This was done by implementing a knowledge source whose action

[1] However, we have not yet examined IDA's behavior when exposed to cues from the environment.

is to spread activation one additional level; this knowledge source is always executable, but is given lowest priority in execution—i.e., it will not run if any other knowledge source is executable. The resulting behavior is more or less equivalent to having a spreading activation process running in the background: it sits idle when there is any other reasoning going on, and executes constantly otherwise.

The particular version of spreading activation used in IDA has these characteristics:

(1) Activation consists of numerical values between 0 and 1.

(2) The only links that spread activation (bidirectionally) are instance-type, subtype-supertype, and part-whole links. Instance-type links spread 100% of a node's activation to a connected node, and the other two links spread 70%.

(3) A node receives activation from another node only if the amount it would receive is greater than its current activation value. In other words, if node A is trying to pass activation value x to node B, B's next activation level will be $MAX(x, current\_activation(B))$. So a concept which is densely connected will not receive more activation because it gets a small amount of activation from many different sources; it will, however, tend to receive more activation per cycle because it has a higher number of potential *single* sources.

(4) A node which is a source of activation will have its activation progressively increased from cycle to cycle until it reaches the maximum value of 1.

(5) There is no decay of activation over time.

(6) Graphs are retrieved when the average activation of every node in the graph is 0.5. A device is considered an intermediate concept and evaluated by the mapping component when one of its constituent graphs (structure, behavior, or function) is retrieved.

Some of the characteristics of the spreading activation used in IDA were determined by IDA's similarity metric. Because semantic distance in a mapping is only dependent on path distance in the type hierarchy, there is no reason to allow general spread of activation by all link types. Similarly, since the mapping component only permits one-to-one mappings, and therefore a given node may only correspond to another node through a single path, there is no reason to allow a single node to receive activation from multiple sources.

The technique of progressively increasing source activation levels came about because of a difficulty with the temporal distribution of concept retrievals. With only a single, static source activation level, spreading activation (after a few cycles) was retrieving many different concepts in a single cycle. The method of progressively increasing the activation over time allows activation to reach deeper into the knowledge base in a given subsearch, activating more nodes but with a lower level of activation, with the effect of spreading out the distribution of concept retrievals over time. This allowed IDA to deal with only one (or few) retrieved ICs at a time, making decisions based on only that small number of concepts and adjusting the search accordingly. This had an overall beneficial effect on KDSA's performances. However, expanding each KDSA sub-search to deeper in the knowledge base had detrimental effects on the amount of effort in spreading activation. These detrimental effects are discussed below.

6.2 Experiments

6.2.1 Experimental Design

Ideally we would run experiments with IDA retrieving analogies from a true very large knowledge base. Since the such KBs do not exist yet, those experiments will have to be saved for future work. For now, we can observe IDA's behavior in small KBs under different configurations and from those observations extrapolate likely behavior in large KBs. In the experiments reported here, we examine the time taken for IDA to retrieve a semantically distant analogy in a small but heterogeneous KB as the knowledge base grows.

Knowledge Base Used

The knowledge base used for this experiment consists of 29 fully represented models of devices. Each device is represented as an abstract {structure, behavior, function} model as

```
AIRSHIP-AIR-BALLONETS-DIVE
AIRSHIP-AIR-BALLONETS-RISE
AQUALUNG
BLINKERED-RR-CROSSING
CIRC-SYS-AS-COOLING-SYS
CIRCULATORY-SYSTEM
DIGESTIVE-SYSTEM
ELECTRIC-GENERATOR
FIRE-EXTINGUISHER
HYDRO-ELECTRIC-TURBINE
INTERSTATE-HIGHWAY-SYSTEM
ON-OFF-VALVE
ONE-WAY-VALVE
PISTON-PUMP
PLUMBING-SYSTEM-AS-WASTE-REMOVAL
PLUMBING-SYSTEM-AS-WATER-DIST
RADIATOR
RADIATOR-PATCH
ROCK-CRUSHER
ROCKET-ENGINE
SPRAYING-FOUNTAIN
SPRINKLER-IRRIGATION-SYSTEM
SUBMARINE-BALLAST-TANKS-DIVE
SUBMARINE-BALLAST-TANKS-SURFACE
TWO-WAY-STOP
VACUUM-CLEANER
WATER-METER-WITH-COUNTER
WATER-METER-WITH-POINTER-DIAL
WINDMILL
```

Figure 6.1: Devices represented in knowledge base for experiments

described in Section 4.2.1. The devices chosen for representation reflect a diverse collection of domains and diverse sources of expertise. Four Stanford researchers provided one or more device descriptions which were converted into models, and many of the other models were based on a more-or-less random traversal of the book *The Way Things Work* [Macaulay, 1988]. The 29 devices represented in the knowledge base are shown in Figure 6.1. The representations of these devices, along with the type hierarchy of concepts used in those representations, comprises about 1100 total objects.

Data collection method

For these experiments, IDA was run on BB1 v3.2 running on top of Lucid Common Lisp on a Sun SparcStation-10. Experiments were run on two separate retrieval examples: the sprinkler irrigation $\Rightarrow$ circulatory system example of Section 4.3, and the blinkered RR crossing $\Rightarrow$ on/off valve example of Section 1.3.

The size of the knowledge base was varied randomly as follows. For each run, a set of n randomly chosen knowledge-base files—where each knowledge base file contains definitions of 1–3 devices from Figure 6.1—was loaded before run time. Those files, along with the

KB files containing knowledge necessary for the example, comprised the knowledge base known to IDA for that run. Only concepts that would be evaluated as "unpromising" by IDA's mapping evaluation heuristics were eligible for exclusion from IDA's knowledge base; the target, the base, and all promising concepts were automatically loaded. The number n of knowledge base files loaded was varied, and for each value of n, some number m of separate runs were performed. For the sprinkler irrigation example, a total of 304 runs were performed; for the railroad crossing example, 37 runs were performed.

For each knowledge base configuration, the analogy retrieval example was run on both KDSA and on standard spreading activation (SA), where SA consisted of running IDA without loading its mapping and search control heuristics. A number of measurements of the level of effort involved in retrieving the analogy were recorded:

(1) CPU time taken to retrieve the analogy, broken down into time spent spreading activation in the network, time spent mapping candidate analogies, and BB1 system time (time spent in BB1's decision cycle).

(2) The total number of unique nodes activated during a search. For this measure, the individual beacon searches of KDSA were each counted as separate searches, so the KDSA measure is the sum over all of the beacon searches performed of the total number of unique nodes activated. This means that nodes will be double-counted in the KDSA measure, but cannot in the SA measure.

(3) The total number of *attempts* to spread activation from one node to another. These attempts include *all* cases where activation is spread, even when the "spread" of activation causes no change in the activation of the destination node.

(4) The total number of BB1 cycles taken to retrieve the analogue.

6.2.2 Results

The timing data for the sprinkler irrigation runs are shown in Table 6.1, and the graph of total CPU time vs. KB size is shown in Figure 6.2. The same information and graph

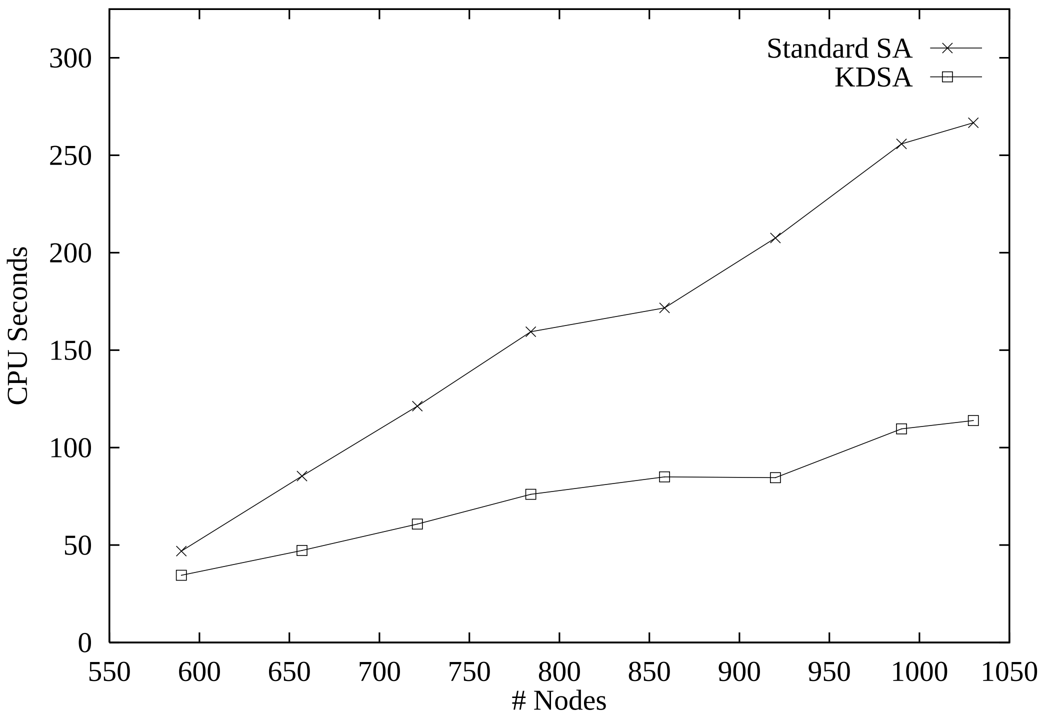


Figure 6.2: CPU time taken to retrieve analogy as KB size grows, sprinkler irrigation example

			KDSA			Standard SA		
# dev	# nodes	# runs	SA tm	map tm	tot tm	SA tm	map tm	tot tm
4–6	590.0	18	3.1	11.4	34.5	3.1	25.7	46.9
7–9	657.0	45	4.3	21.0	47.2	4.3	59.6	85.4
10–12	721.1	41	5.4	34.2	60.8	5.0	94.9	121.3
13–15	784.1	44	5.8	48.6	76.0	5.3	132.3	159.5
16–18	858.4	44	7.4	54.1	85.0	5.9	143.3	171.7
19–21	920.0	42	7.6	54.9	84.6	6.4	178.1	207.6
22–24	990.1	52	10.1	75.3	109.6	7.4	223.1	255.9
25–27	1030.0	18	9.7	81.6	113.9	7.1	237.5	266.7

Table 6.1: Timing measurements of run for sprinkler irrigation example (all times given in CPU seconds)

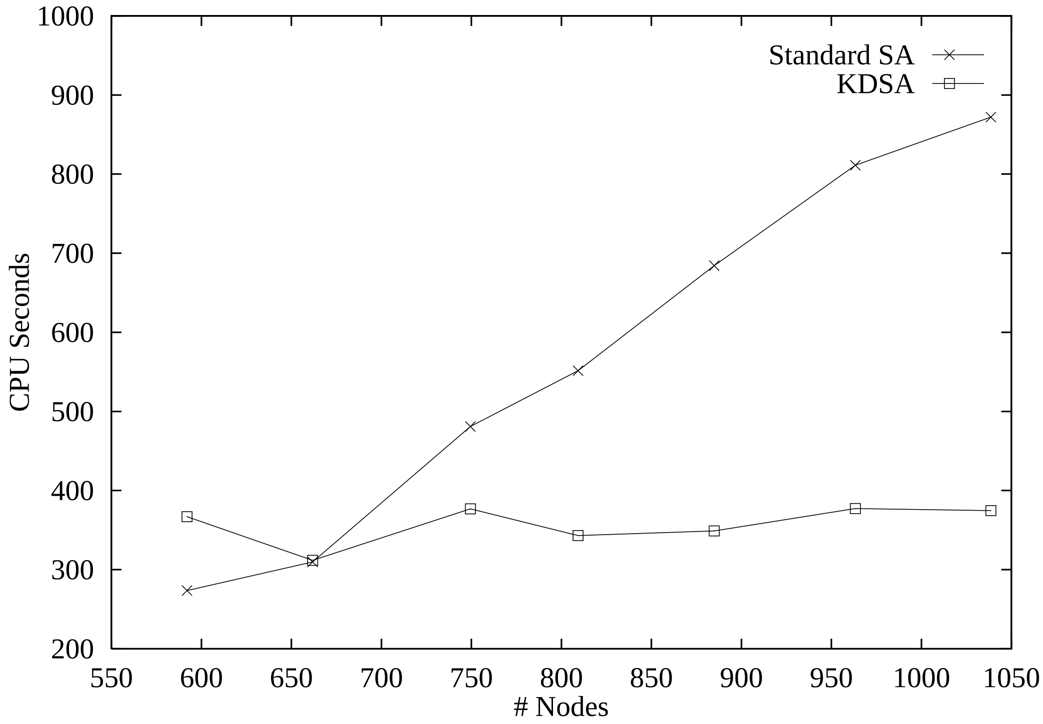


Figure 6.3: CPU time taken to retrieve analogy as KB size grows, railroad crossing example

			KDSA			Standard SA		
# dev	# nodes	# runs	SA tm	map tm	tot tm	SA tm	map tm	tot tm
0–4	592.0	1	2.0	342.8	366.9	1.5	257.0	273.6
5–8	661.8	5	1.8	290.6	311.8	1.7	291.5	309.7
9–12	749.5	8	2.8	349.5	376.9	2.3	456.6	481.0
13–16	809.3	3	2.1	320.2	343.2	2.4	530.1	551.5
17–20	884.9	9	2.6	322.9	348.9	2.7	661.0	684.2
21–24	963.3	6	2.8	350.8	377.2	4.2	781.4	811.3
25–28	1038.6	5	3.7	342.1	374.6	2.6	849.0	872.1

Table 6.2: Timing measurements of run for railroad crossing example (all times given in CPU seconds)

for the railroad crossing example are shown in Table 6.2 and Figure 6.3, respectively. To remove some of the noise, the data is collected and averaged according to ranges of numbers of knowledge base files loaded; e.g., the first row of Table 6.1 summarizes all runs where 4 to 6 KB files were loaded, the next where 7 to 9 files were loaded, etc. For the tables, column 2 contains the average number of nodes in the KB for all the runs of that range; column 3 contains the total number of runs in that range; columns 4 and 5 contain the total time spent by KDSA in spreading activation and mapping, respectively, during the run; column 6 contains the total time KDSA took to retrieve the analogy[2]; and columns 7 through 9 contain the same information for standard SA that 4 through 6 contained for KDSA.

The graphs in Figures 6.2 and 6.3 confirm the general result from the theoretical analysis: that KDSA is affected far less than standard SA by growth in the knowledge base. Figure 6.2 shows both algorithms as roughly linear in the KB size, but shows the slope for KDSA as significantly less than that for SA. Figure 6.3 shows the general result even more dramatically; KDSA's retrieval time was virtually uneffected by KB size in the experiments involving the railroad crossing example. It is unreasonable to expect KDSA to be a constant-time algorithm in the size of the knowledge base, but Figure 6.3 gives reason to believe that the actual behavior of KDSA might approach the logarithm-shaped curve of Figure 5.2.

One interesting result from these experiments is that the amount of time taken to map candidate analogies to the target overwhelms the amount of time taken to spread activation in the network. This points out an important benefit of KDSA not explicitly considered in the analysis in Chapter 5: the ability to avoid performing unpromising mappings. A search focused on promising areas of the knowledge base should eliminate not only much of the combinatorial explosion associated with standard spreading activation, but also much of the corresponding explosion in the number of concepts retrieved by the spreading activation mechanism. Since both search methods—KDSA and standard SA—must map every retrieved concept to the target (standard SA must check if the retrieved concept meets the similarity metric), avoiding the spread of activation in unpromising areas of the KB will

[2]The time not accounted for by spreading activation and mapping is the BB1 system time for the run.

	KDSA		Standard SA	
# nodes	act nodes	act attempts	act nodes	act attempts
590.0	504.0	3954.0	359.0	4065.0
657.0	581.7	4806.2	417.6	4915.5
721.1	657.8	5698.1	474.6	5796.1
784.1	730.5	6538.2	530.3	6636.1
858.4	814.0	7356.5	593.2	7426.8
920.0	884.5	8183.9	645.6	8277.6
990.1	964.8	9113.5	1706.1	9194.8
1030.0	1012.0	9660.0	742.0	9734.0

Table 6.3: Spreading activation statistics for sprinkler irrigation example

result in avoiding many expensive mappings. In fact, nearly all of the benefit from KDSA shown in Figures 6.2 and 6.3 comes from savings in mapping time.

But this brings up the other side of the coin: Tables 6.1 and 6.2 actually show standard SA performing a little better than KDSA in time spent spreading activation during the search. A closer examination of the spreading activation work done during the sprinkler irrigation runs (Table 6.3) confirms this: the total number of unique nodes activated is consistently higher for KDSA than for standard SA, and the number of activation attempts—probably a more accurate measure of total spreading activation work—shows the two algorithms as virtually identical. This, of course, is in conflict with the predictions of the theoretical model. The result is probably explained by the specific spreading activation algorithm used in IDA. The technique, discussed earlier, of progressively increasing the activation of source nodes causes each individual beacon search of KDSA to reach deeper into the knowledge base, and Figure 5.3 shows that the spreading activation cost of KDSA can reach and exceed the spreading activation cost of standard SA if the beacon search depth is high enough. A solution to this problem probably lies in finding a spreading activation technique that allows the retrieval of concepts to be evenly spread out over time while maintaining a smaller average depth than the technique currently used in IDA.

6.3 Other Surprising Analogies

While IDA's behavior in the two examples it runs plausibly represents the behavior of a retrieval algorithm encountering a problem for the first time, we in fact expected the sequence of beacons that IDA used to be more or less what it ended up being in those cases. However, IDA did surprise its builders in a couple of other instances. During experiments with an earlier version of the system, IDA produced a suggested redesign of the sprinkler irrigation system based on a fire extinguisher. The suggested process to replace was not spraying, as it is with the circulatory system analogy, but pumping—IDA's proposed redesign in this case suggested that the sprinkler irrigation system replace its pump with a compressed gas propulsion mechanism like that of the fire extinguisher. This analogy struck human observers as not only surprising, but also useful. A second example occurred even earlier before the adaptability requirement and the redesign step had been added. IDA gave a strong evaluation to an analogy between the irrigation system and a rock crusher, an analogy that suggested to one human observer the notion of transporting water by a conveyor-belt-like mechanism.

These two examples point out that it is sometimes valuable to minimize the number of constraints on the types of allowable analogies. IDA has no conventional notion of a design goal to constrain the analogy—it simply knows it should redesign the existing device *somehow.* In some very important ways, this is a weakness of the system. But IDA would have been unable to retrieve the surprising analogies above with stronger behavioral or functional constraints placed on the redesign specification. (In fact, because of the adaptability requirement, it would not recognize the rock crusher analogy now.) Fewer constraints on the analogy will tend to increase the likelihood of producing surprising analogies—and unfortunately will probably increase the likelihood of flaky, useless analogies as well. The key here is to find the middle ground, where the similarity metric is restrictive enough to eliminate most of the flaky analogies, but flexible enough to give the analogy mechanism the element of surprise.

Chapter 7

Related Work

This chapter discusses a number of other research projects and their relation to this thesis. The major research areas considered are: AI work on retrieval, including analogy retrieval, more general knowledge retrieval, and information retrieval; other work on analogy; and work on design and creativity. In each subfield, only work which is particularly relevant to KDSA is discussed.

7.1 Work on Retrieval

Research on retrieval is divided here into research on spreading activation and research on other retrieval techniques.

7.1.1 Work in Spreading Activation

Work on spreading activation can be roughly divided into two categories: approaches that use spreading activation as general analogy retrieval mechanisms (specifically, use spreading activation to retrieve cross-domain analogies), and approaches that use spreading activation to retrieve concepts which match the target as closely as possible. These latter approaches, which might be termed information retrieval approaches or "literal similarity" approaches, are discussed in the first two sections below. The spreading activation approaches to general analogy are discussed in the subsequent sections.

Rau's SCISOR

Lisa Rau's SCISOR system [Rau, 1987a; Rau, 1987b; Rau, 1989] represents an approach to retrieval based on constrained marker passing in a semantic network. The system performs the task of answering user questions in the domain of corporate takeovers. SCISOR's knowledge base is organized as a collection of episodes which are essentially conceptual graphs representing corporate takeovers, along with a type hierarchy which classify the various nodes that may appear in an episode. SCISOR answers a question by converting the (English) question into a graph, retrieving an episode in its knowledge base which best matches the graphical question, and converting the retrieved episode into a natural language answer to the user's question. The retrieval of episodes takes place in two steps: 1) a coarse search, consisting of passing of markers from nodes in the graphical representation of the question to neighbor nodes according to a set of priming rules, and 2) a fine search, consisting of a graph matching between the most promising episodes retrieved and the graph representing the user's question. The marker passing process in SCISOR continues until all possible nodes in the system's entire type hierarchy are marked; the priming rules, which determine which nodes may be marked, must control the search so that the passing of markers does not extend to every node in the entire knowledge base.

Like IDA, SCISOR uses heuristic search control information to control the spread of activation throughout a semantic network (Rau's marker passing algorithm is exactly like spreading activation without passing numeric values), and uses heuristic information about semantic distance to determine match quality. Furthermore, Rau speculates how a different collection of heuristics would produce different retrieval behaviors. IDA differs from SCISOR, however, in that it uses information from previous match evaluations to dynamically adjust the direction of the spread of activation. IDA in effect runs a series of SCISOR-like searches, starting each sub-search from the near-misses it has encountered in previous sub-searches, and using the evaluations of those near-misses to formulate its search control for the next sub-search.

A second difference between IDA and SCISOR lies in the heuristics themselves. SCISOR's priming rules prevent the spread of activation from reaching above two levels in the type hierarchy from the original source node. That is, when a node in a graph is assigned a marker, that node's type and the parent of that node's type are the only two nodes above the original source node which receive a marker. This rule ensures that the markers from a question will not be passed to episodes whose nodes are semantically distant from the nodes of the question, and makes it feasible for SCISOR to run this marker passing process to exhaustion before starting the matching process. That procedure seems to work in a domain like SCISOR's, where the retrieval process is attempting to maximize similarity between question and episode. However, it is inadequate as a general model of analogy retrieval. In a general analogy retrieval situation, there will often be cases where the minimum common generalization of two corresponding nodes is higher than two levels above those nodes in the type hierarchy. There needs to be a different method of controlling the spread of activation throughout the network. IDA's method of dividing the search into a series of controlled sub-searches provides such a method.

Cohen and Kjeldsen's GRANT

A second system which uses heuristics to guide the search for a matching concept in a semantic network is Cohen and Kjeldsen's GRANT [Cohen and Kjeldsen, 1987; Kjeldsen and Cohen, 1987]. GRANT's task is to take a description of a researcher's interests or project, and return a set of funding agencies who would be likely to fund the researcher's work. The idea behind GRANT is to find funding agencies whose interests match the researcher's by metrics other than a simple keyword match, i.e., by specific semantic relationships between the researcher's interests and the agency's. To do this, GRANT uses a semantic network which classifies concepts in research according to their relationships with other concepts. After the description of the researcher's proposed study is entered, GRANT activates the node(s) in the network which describe the study, and then spreads activation in the network until a set number of funding agencies are retrieved, or until the entire network is exhaustively searched to a certain depth. GRANT constrains the spread of activation in three ways:

(1) Distance: Activation ceases four links from the original source node.

(2) Fan-out: Activation ceases at nodes that have very high connectivity, i.e., are connected through links to a large number of other nodes.

(3) Path endorsements: these are rules of the form

$$request\_funds\_for\_topic(x) \wedge R(x, y) \Rightarrow request\_funds\_for\_topic(y)$$

where $R(x, y)$ specifies that x and y are related by some path of links. Each path endorsement has a weight attached to it indicating how strongly the path should pass activation. These weights may also be negative, indicating that GRANT should prune those paths from the search.

Cohen and Kjeldsen have run a thorough set of experiments with GRANT, comparing the retrieved agencies with ratings by an expert. They measured both its recall (the percentage of agencies highly rated by the expert which GRANT retrieved) and its fallout (the percentage of agencies retrieved by GRANT which were rated poorly by the expert). Experiments showed that GRANT had higher recall than simple keyword lookup, and lower fallout than blind search.

Like IDA (and SCISOR), GRANT is a system which applies knowledge to direct the spread of activation in a semantic network. However, several aspects of GRANT's approach make it inappropriate for the general retrieval of analogies, especially cross-domain analogies. First, it is spreading activation from a very incomplete description of the target problem—the single node representing the main topic of the study—while IDA spreads activation from as complete a description of its target problem as possible. Second, GRANT allows the spread of activation, and the strength of activation assigned by the path endorsements, to serve as the similarity metric for the system. That is, the strength of a funding agency's activation determines whether or not it is considered a good match for the study; there is no graph matching or evaluation process. For the general analogy problem, however, semantic closeness will not suffice as a similarity metric. There must be a process to determine the degree of isomorphism between the target and the base [Thagard *et al.*, 1990]. Third, GRANT's distance constraint is not appropriate for cross-domain analogies, since there are bound to be valid analogies which are greater than four links away from the target. And fourth, as with SCISOR, GRANT only uses its search control knowledge to reduce the branching factor of the search (by specifying what paths are likely to lead to good answers), rather than using the dynamic state of the search to restart search in a more promising direction several times along the way. Again, IDA can abstractly be described as doing a series of GRANT-like constrained searches, and Chapter 5 shows that this sequential search behavior is more likely to be tractable in a large knowledge base than a reduced branching factor approach like GRANT's.

Cohen and Kjeldsen's experience with GRANT does have a great lesson to teach the analogy retrieval community, however. GRANT's performance degraded significantly as they added funding agencies and concepts to the knowledge base without changing the path endorsements. Cohen and Kjeldsen speculated that the source of the performance degradation was that the methodology for defining concepts in the semantic network had changed without corresponding changes in the path endorsements, and they ran some experiments which seemed to validate this hypothesis. This experience shows the importance of maintaining a unified knowledge representation methodology which is consistent with the search control knowledge of the retrieval system. Without a strong agreement between knowledge representation and search control, IDA is likely to have the same kind of performance problems in large knowledge bases.

Holland et. al.'s PI

The PI system by Holland and his colleagues [Holland *et al.*, 1986] is a general cognitive architecture capable of many different methods of reasoning and learning. Holyoak and Thagard have proposed a complete model of analogy within the PI framework. This model retrieves analogies by spreading activation in a network of frame-like objects. The activation is spread automatically from the target problem representation along PI's type hierarchy, and also by rules from the current goal to other nodes related to goal solution. A concept is retrieved according to the summation of the activation of all concepts with which it is stored, and activation is controlled by a decay parameter.

Like IDA, PI retrieves analogies by spreading activation. However, PI has limited ability to control the spreading activation search for analogs. PI's associative rules, which provide one mechanism by which activation is spread, could theoretically be used to prune large sections of the semantic network from the search; however, it seems that the rules are used in practice as a mechanism for introducing a pragmatic constraint into the analogy retrieval—i.e., they push PI toward retrieving an analogy which is related to the particular goal at hand. The rules do this by adding activation to concepts which are relevant to the goal, while other concepts simply receive activation based only on their semantic similarity to the target. This function of directing the search toward analogies useful in goal solution is performed in IDA by the search control knowledge. However, the other function os IDA's search control rules, narrowing the search space to avoid combinatorial explosion as the search moves far from the target, is mostly missing from PI. Since PI was built as a cognitive model, and since psychological studies show that people more easily retrieve analogs which have a great deal of surface similarity to the target, PI is designed largely to facilitate the retrieval of analogs which are near the target in the semantic network. In particular, the method of reducing search by using promising concepts as beacons in IDA is not included in PI's model of analogy retrieval. The problem of control in PI is discussed in [Thagard *et al.*, 1990].

Jones's EUREKA

EUREKA [Jones, 1989] is another general cognitive architecture which uses spreading activation to retrieve concepts from long-term memory. Memory is organized in a semantic network, and retrieval of concepts is accomplished by an ACT*-like spread of activation along the network's links. The spread of activation in EUREKA depends on trace strengths on the links, where the trace strengths are set so that retrieval prefers 1) familiar concepts and 2) those concepts which have led to success in problem solving in the past.

Langley and Jones have proposed a model of scientific discovery based on analogy [Langley and Jones, 1988] which is compatible with the EUREKA architecture. In this model, knowledge of physical situations is stored using Forbus's Qualitative Process Theory [Forbus, 1984], and each situation is indexed by features of the situation's envisionment. When a new situation is encountered, activation spreads from the envisionment of the new situation, retrieving past envisionments which are well-indexed and/or recently studied. These past envisionments are then compared to the new one to test for a possible analogy, and if an analogy is found the process model of the past envisionment can be transferred to suggest a process model of the new situation. Alternately, if no past situations are retrieved when the new one is encountered, the system may wait for a cue which reminds it (through spreading activation) of a past situation, and activation spreads from the past envisionment to retrieve the new poorly understood one. The new one is likely to be retrieved since the trace strengths associated with it have been increased due to its familiarity.

Like IDA, EUREKA provides a general retrieval model which accounts for the use of analogy in creative endeavors. And also like IDA, EUREKA accounts for the utility of external and internal cues in retrieving useful creative analogies. But EUREKA's ability to retrieve these analogies spontaneously, i.e., when the cue is not provided, is very limited. EUREKA's only mechanism of search control in the spreading activation is through the use of learned trace strengths, and the learning heuristics there—preferring familiar concepts, and preferring success in problem solving—will not provide enough search control for a search to a semantically distant concept. Since EUREKA is concerned with cognitive modeling, it, like PI, is concerned primarily with retrieving analogies based on surface similarities. IDA is allowed to retrieve a semantically distant analogy in any manner possible, including spontaneous retrieval through a deep heuristically guided search.

Anderson's PUPS

Another general spreading activation approach to analogy retrieval is implemented in Anderson's PUPS system [Anderson and Thompson, 1989] The PUPS approach to analogy mapping consists of repeated applications of a set of three heuristic rules to elaborate the representation of the target concept until it can be easily matched to the base. The heuristics transform the target concept by repeatedly replacing form (structural) descriptions within the target with the function that the forms fulfill. Retrieval in PUPS is accomplished when an analogy-inducing production is fired. This production requires as a precondition the existence of a target and a base, where the target is pre-selected but any concept in the knowledge base is eligible to be the base. The PUPS architecture narrows down the field of potentially matching bases through its normal memory retrieval procedure—spreading activation. When the target is selected, activation spreads from it throughout the semantic network, and the highest activated structure in memory is the one selected as the base. As with EUREKA, the only method of controlling the spread of activation in PUPS is through learning strengths on the links in the semantic net, and through decay of the activation over time. Like the other spreading activation approaches discussed above, PUPS's retrieval method does not provide the more sophisticated types of control used in KDSA to retrieve semantically distant analogies.

Winston's Classification-Exploiting Hypothesizing

In Winston's project concerning learning from analogies [Winston, 1980; Winston, 1982], he proposes a model for analogy retrieval in a semantic network based on classification-exploiting hypothesizing. This approach involves enumerating all of the nodes comprising the target concept, moving up the type hierarchy from those nodes to the most general type, and having each encountered type "vote" for each potential base concept in the KB based on 1) the importance (measured by number of slots) of the node in the target, 2) whether the base concept involved a node of that type, and 3) the number of other potential bases involving a node of that type. This method of retrieval is essentially computationally equivalent to ACT*-style spreading activation from the target concept along type hierarchy links, with the initial activation of the nodes in the target set proportionately to the number of slots in the node. As such, it will have the same problems scaling up to large knowledge bases already mentioned for basic spreading activation approaches. Winston discusses scalability problems of classification-exploiting hypothesizing in [Winston, 1980].

7.1.2 Other work in retrieval

Several other approaches to analogy and information retrieval which are not spreading activation approaches deserve mention here. These other psychological models, case-based reasoning indexing approaches, and approaches in the field of Information Retrieval concerned with accessing relevant documents.

Thagard, et. al's ARCS

Thagard and his colleagues have proposed an analogy retrieval mechanism based on constraint satisfaction called ARCS [Thagard *et al.*, 1990]. This method retrieves analogies by first constructing a constraint network between the target concept and every other concept in the knowledge base which contains some degree of semantic overlap with the target, and then running a parallel relaxation algorithm on the network to determine the highest rated analog, where the rating is determined by weights in the constraint network. This method allows the retrieval process to be influenced by the degree of isomorphism between target and potential base, a factor which standard spreading activation cannot take into account. Thagard, et. al. show that ARCS accounts for several human retrieval phenomena demonstrated in the psychology literature.

ARCS is an elegant approach which is very interesting as a cognitive model of analogy retrieval. However, it has limited use as a practical analogy retrieval mechanism in large knowledge bases. Thagard, et. al. find that the construction of the constraint network for each target takes $O(n^4k^2)$ time, where k is the number of concepts in the KB and n is the size of the largest concept in the KB. And their experimental results show this worst-case analysis holding true in actual execution, where the size of the constraint network grew with the square of the size of the knowledge base. The constraint that ARCS only considers potential bases which have some degree of semantic overlap with the target helps prune the network to some degree, but in very large knowledge bases the number of potential bases will still be large, and k^2 is too inefficient for a general retrieval mechanism. KDSA is at worst linear in the size of the knowledge base, and there is evidence, presented in Chapters

5 and 6, to expect its performance to be better than that in practice.

Indexing on a Subset of Salient Features

A large number of AI researchers (see, e.g., [Kolodner, 1984; Kolodner and Simpson, 1984; Kolodner and Thau, 1988; Riesbeck and Schank, 1989]) have attacked the problem of retrieval by indexing the knowledge base on a subset of features which are salient for the task at hand. In indexing approaches, commonly used in case-based reasoning, concepts are typically represented not as general graphs, but rather as frames which contain a set of features (slots), each of which contain some value. Many of these approaches are based on creating a discrimination network structure, with the individual cases (i.e., candidate structures for retrieval) at the leaf nodes, and feature tests in the non-leaf nodes. The retrieval algorithm navigates from the network's root node to a leaf by performing the feature test specified by each non-leaf node, choosing one of the node's children based on the outcome of that test, and repeating the procedure until a leaf is reached. The leaf node reached is then considered retrieved. The particular branch to follow at each step is determined by the values of those features in the target case. In this way, the discrimination net approach can retrieve the most similar case to the target along the salient features identified in the discrimination net, and it can do so in time roughly logarithmic in the size of the case base.

The discrimination net approach to retrieval is efficient only if the task and the features salient to that task are known before run-time. A general task-independent discrimination network indexing the entire knowledge base on every possible feature in the KB is not possible, because the retrieval process will take time exponential in the number of irrelevant features. It also seems unlikely that it is possible to index a large knowledge base by building separate discrimination networks for each task for which analogical (case-based) reasoning might be used. In a large multi-use KB, there would be a large number of different tasks for which to construct discrimination networks, and each network will take space which is exponential in the number of features relevant to its task. Therefore, these discrimination network approaches are not built to solve the same problem addressed in this thesis: namely the retrieval of analogies in a large multi-use knowledge base.

One of the major concerns of the case-based reasoning community is selecting an appropriate index (i.e., a set of salient features) for a case at case storage time. Kolodner [Kolodner, 1991] outlines the index selection procedure this way:

> First, determine what the case could be useful for. Second, determine under what circumstances the case would be useful for each of these tasks. Third, massage the circumstances to make them as recognizable and generally applicable as possible.

KDSA, in contrast to case-based reasoning, is concerned with retrieving relevant "cases" when the cases have been stored without regard to future use—or at least have been stored anticipating many different possible uses, some of which cannot be predicted. For this problem, the three steps Kolodner outlines are impossible; "cases" are organized in a more-or-less task-independent fashion. In this situation, the burden for efficient storage and retrieval is shifted to the retrieval mechanism. That is, CBR approaches typically provide efficient retrieval with a sophisticated task-dependent storage mechanism and a relatively simple retrieval mechanism, while KDSA provides efficient retrieval with a relatively simple and general storage mechanism and a sophisticated retrieval mechanism.

One other piece of research in case-based reasoning deserves mention here: Zito-Wolf and Alterman's theoretical analysis of case-based planning [Zito-Wolf and Alterman, 1993]. Zito-Wolf and Alterman present an analysis of their technique for storing and retrieving steps in a plan, called multicases, and show that the multicase method is as efficient or more efficient in case storage and retrieval costs than other existing methods of case-based planning. Like the work presented in this thesis (Chapter 5), Zito-Wolf and Alterman developed a theoretical model of knowledge retrieval and evaluate their method against other existing methods under a set of theoretical assumptions. Also like the work presented in this thesis (Chapter 6), they experimentally validate their theoretical findings by measuring the execution of an implemented system. The difference between Zito-Wolf and Alterman's work and the work presented here is in the type of retrieval studied. Zito-Wolf and Alterman restrict their analysis to the retrieval of plans and plan steps, i.e., their analysis explicitly relies on the retrieved object being an object which can be decomposed into an ordered sequence of smaller subobjects. The analysis of Chapter 5 makes no such assumption about the representation or decomposability of the retrieved objects. Also, Zito-Wolf and Alterman assume that an object is retrieved using the discrimination network approach outlined above, i.e., an object is retrieved by a series of feature tests. The analysis in Chapter 5 models retrieval as a search in a semantic network.

7.2 Work on Design and Creativity

There is a small but growing body of literature in artificial intelligence dealing with creativity or innovation generally, and with creative or innovative design particularly. This section discusses work in this area which is especially relevant to KDSA.

7.2.1 Work on creative design and problem solving

Kolodner and Wills's Case-Based Analysis of Creative Design

Kolodner and Wills have performed an analysis of creative design based largely on protocols from a team project in creative design for a mechanical engineering class [Kolodner and Wills, 1993]. The protocols show that the creative design team made substantial use of analogies, especially cross-contextual analogies, in formulating their new design. Given a design specification, one team member would suggest an analogous design outside the current domain of consideration, which would in turn remind another team member of another analogy along a different set of features, and so on. This general model of team creative design is compatible with the model of analogy retrieval used by IDA, where each retrieved potential base is evaluated according to design goals, and then may in turn remind the system of other promising ideas, and so on until a final analogy is retrieved.

Kolodner and Wills suggest many different possible uses for analogies in creative design which are not used in IDA, including case-based evaluation of proposed designs, and case-based reformulation of design specifications. They acknowledge that anticipatory indexing is not sufficient to explain all analogy retrieval in the creative design process, but they do not as yet have a concrete proposal for an alternative retrieval mechanism.

Bhatta and Goel's IDeAL

Bhatta and Goel address the issue of analogy use in innovative design by learning generic teleological mechanisms (GTMs) in their system, IDeAL [Bhatta and Goel, 1993]. GTMs represent abstract design principles such as cascading, feedback, and feedforward which can be used in design tasks across domains. The system learns the GTM by abstracting across two same-domain design examples, one given by the user and the other retrieved solely on functional similarities with the first. After the GTM is learned, it may be applied in a domain unrelated to the original domain; for example, IDeAL is able to apply the GTM representing cascading generated from circuit design examples to a heat exchanger design problem.

IDeAL can be said to use analogy in two ways: it uses a same-domain analogy between design examples to learn the GTM, and then it uses the GTM as a base for analogies in new innovative designs in other domains. The use of analogy in IDA is different from both of IDeAL's uses. IDA is concerned with retrieving a cross-domain device which can be useful in redesigning an existing device, rather than retrieving a same-domain design which is merely functionally similar to the original. And, while IDA exploits the existence of generic concepts in the knowledge base as a mechanism for guiding the search into another domain (see Chapter 4), it is largely concerned with retrieving analogies when a predefined abstraction between analogs does not exist. The use of generic concepts is an important idea in innovative design, but so are analogies drawn without the assistance of generic concepts. These latter analogies are the ones IDA is looking for.

Turner's MINSTREL

Turner's MINSTREL [Turner, 1992] is another program which addresses analogy's role in the creative process. Like IDA, MINSTREL attempts to recall bases which are different from the target in important ways to facilitate creativity. However, MINSTREL's approach to retrieval is very different from IDA's. MINSTREL 1) uses a collection of heuristics to transform the target problem into a slightly different problem, 2) retrieves a past case which is similar to the transformed problem, and 3) adapts the retrieved case (perhaps using the set of transformations applied in step 1 as a guide) into a solution for the target problem. While MINSTREL looks for differences between target and base by transforming the target, IDA looks for differences by continually searching, retrieving, evaluating, and modifying the search based on what was retrieved, always comparing to the original target. In this way, IDA's search for analogies is guided in part by the actual contents of the knowledge basethe final analog retrieved depends on the concepts encountered by the search along the way. Also, MINSTREL has no evaluation step in its transformation/retrieval sequence.

Lenat's AM

Lenat's AM [Lenat, 1976] is a program which exhibited creativity, in that it discovered concepts in elementary mathematics and set theory that it previously did not know. Like IDA, AM operates by conducting a heuristic search through a space of concepts and evaluating those concepts using heuristic criteria for "interestingness" (AM's Interestingness Heuristics are analogous to IDA's Mapping Heuristics which evaluate the creativity of a newly retrieved concept). One major difference between AM and IDA lies in the method used to generate new concepts. AM creates completely new concepts by modifying slots in existing frames by applying heuristic operators. IDA, on the other hand, retrieves concepts which are already in its knowledge base by searching along the KB's links.

7.3 Other Work on Analogy

Aside from some of the work discussed above, most research on analogy has focussed on the problems of analogy mapping, transfer, and validation, rather than on the problem of analogy retrieval. Since the primary focus of this project has been on retrieval, there is little connection between the bulk of this project and most other projects in the analogy literature. However, there are important roles for mapping and transfer in KDSA and IDA, so it is worth identifying where the mapping approach here fits in with other analogy proposals. Two aspects of mapping will be addressed.

The first aspect of mapping to consider is: how does the system decide which items in the target and base representations to place in correspondence? Gentner's systematicity principle [Gentner, 1983] isolates isomorphism as the primary factor in mapping by preferring to map connected causal substructures of the target and base and leave isolated unconnected features unmapped. Falkenhainer generalizes Gentner's approach in his Structure Mapping Engine program [Falkenhainer, 1988], which allows the user to specify constraints identifying exactly the set of allowable correspondences (solving what Falkenhainer terms the *selection problem*), and a set of rules which assign a numeric evaluation to each correspondence and identify a method for combining the evaluations (solving the *selection problem*). Kedar-Cabelli's Purpose-Directed Analogy [Kedar-Cabelli, 1985] explicitly brings the task to be solved into the equation, by mapping only those features of the base which allow it to satisfy a desired purpose. IDA produces a mapping by searching for a set of correspondences which maximizes a weighted combination of isomorphism and semantic distance. Unlike Kedar-Cabelli's method, IDA does not consider the task to be solved in the generation of the mapping; it generates the best general mapping it can independent of task, and then uses the task to be solved to evaluate that mapping.

The second aspect of mapping to consider is: how does the system evaluate a mapping once it is generated? For example, how does the system decide whether an analogy is worth pursuing further (attempting analogical transfer, etc.)? This is the question of identifying the system's *similarity metric*. Many approaches to analogy do not address this question directly. For the approaches in [Greiner, 1988; Carbonell, 1983b; Carbonell, 1983a], for example, the only method of evaluation seems to be determining whether the analogy allowed the system to solve the problem; the system maps and tries to reason with any analogy given to it. Other analogy models have more explicit mapping evaluation. In Falkenhainer's Structure Mapping Engine, the user-specified constraints also serve as a method of rejecting analogies for which no mapping can be found; if a mapping satisfying the constraints is found, the base meets the similarity metric. Gentner's Structure Mapping Theory provides an implicit mapping evaluation metric for analogy: Gentner suggests that a relationship between two concepts is an analogy to the extent that it has a high number of shared relations but a low number of shared attributes [Gentner, 1987]. KDSA evaluates analogies by a collection of task-specific rules specifying the desired level of semantic distance and isomorphism for separate portions of the representations. Unlike the methods of Greiner and Carbonell, KDSA identifies a method for evaluating the analogy before actually attempting to solve a problem with it. This is especially important in the more creative uses of analogies since attempting creative problem solving is likely to be computationally expensive. And unlike Gentner's metric for "analogy-ness", KDSA uses pragmatic (i.e., goal-solving) considerations in mapping evaluation by allowing different isomorphism and semantic distance requirements on different portions of the target and base representations.

Chapter 8

Conclusion

This thesis has presented Knowledge-Directed Spreading Activation, a general model of analogy retrieval which is well-suited for retrieving semantically distant analogues from a large knowledge base. The method has been implemented in IDA, a computer program which retrieves analogies that are useful for performing the task of innovative redesign. IDA's execution of example analogy retrievals provides a "proof of concept" for the model, demonstrating that KDSA is a workable mechanism which can be used to retrieve analogies in a real knowledge base, and showing that the search for those analogies can be guided by intermediate concepts retrieved along the way. The tractability of KDSA for retrieval in large knowledge bases was tested in two complementary ways. First, experiments with IDA using small knowledge bases showed that KDSA is much less sensitive to the size of the knowledge base than standard spreading activation. Second, a theoretical model approximating the cost of KDSA under a set of assumptions predicts that KDSA will be much more efficient than standard SA when retrieving semantically distant analogies from a large knowledge base, confirming the trend observed in the empirical study. The theoretical model also predicts that KDSA will be robust in the face of different knowledge base configurations—i.e., that KDSA searches more efficiently than standard spreading activation even when the cost-benefit ratio of its individual subsearches is relatively high.

The work reported here provides a number of important steps toward the development of a general analogy retrieval mechanism for computer knowledge bases. Not surprisingly, though, this thesis is far from the last word on analogy retrieval. This chapter summarizes some of the open issues in analogy retrieval and the ways that this research can be extended to address them, and it identifies the important research contributions of the thesis.

8.1 Future Work

There are a number of natural extensions to this work which would expand our understanding of the analogy process. Here we focus on four—performing experiments in a large knowledge base, investigating the role of the analogical transfer step in KDSA and in analogical innovative design, adding learning to the KDSA model, and reasoning with different types of representations.

8.1.1 Additional Experiments

Since the development of KDSA was motivated by the difficulties of retrieving analogies from a large knowledge base, the most obvious extension to this work would be to analyze the behavior of KDSA in an actual large KB. The major requirements on such a knowledge base are that (1) it is much larger than most KBs in use today, and (2) it represents a diverse collection of concepts from many domains. As of this writing, there are no knowledge bases which meet these criteria, but there are projects, most notably the CYC project [Lenat and Guha, 1990], which are endeavoring to build KBs that do meet them.

There are a number of questions which can be answered by experimenting with KDSA in a different knowledge base from IDA's, and in particular a very large one:

(1) How does the organization of the knowledge base affect KDSA's ability to search it? Are there particular features of the KB which must be present in order for KDSA to be effective?

(2) Are there ways KDSA can exploit the new KB's configuration to improve its search? I.e., can we develop new heuristics that utilize aspects of the new KB which were not

aspects of IDA's KB?

8.1.2 Examination of the Transfer Step

The third step in the analogy process, transfer, is not the focus of this thesis. The transfer step for innovative design presented in Chapter 4 is simple and needs to be extended in order for IDA to produce detailed designs. This extension of a transfer step for cross-domain analogies for design would encounter difficulties that are usually not found with more routine same-domain analogies. When the analogy is cross-domain, often there are aspects of the base's structure which are impossible to produce directly in the target domain. Thus, the only thing *directly* transferred from base to target is an abstract idea, and the reasoner must refine this abstract idea by dealing with such issues as material selection and configuration of structure. These problems usually do not arise in more routine analogical design, where structural components can often be copied directly.

Another important aspect of the transfer step to investigate in this context is: how can the process of validating the analogy (e.g., in IDA's case, producing a design) be used to improve the retrieval process? In this thesis, we show how a detailed evaluation of the mapping step in analogy can be fed back into the retrieval step to guide it to more useful analogies. It seems likely that the same is true of the transfer step: a detailed evaluation of the output of analogical transfer can be fed back into the retrieval step to guide KDSA toward better analogues. The reason for IDA's failure to produce a detailed design based on a particular analogy, for example, can be used by search control heuristics to guide the search toward analogues which are unlikely to lead to that same failure.

8.1.3 Learning

There are two major ways that learning can be added to the KDSA model. One is to include the traditional analogical learning step at the end of the analogical reasoning process, learning abstract concepts which are generalizations of the two analogues. IDA's performance would be enhanced by the system learning these abstractions, not only by giving direct access to analogues when the target is a direct example of such an abstraction, but also by using those abstractions as "bridges" to other domains when the target is not a direct example of one (Chapter 4).

The other way learning can be incorporated into KDSA is in the learning of heuristics for KDSA's mapping and search control components. A number of aspects of the heuristics can be learned: the values of thresholds used in the similarity metric, the particular features of the mapping tested in the similarity metric, and new search control rules which modify the activation of concepts and strengths on links in different situations. The learning of similarity metric thresholds is discussed in Section 4.2.2. The learning of the relevant mapping features could be accomplished with an inductive classification algorithm like ID3 [Quinlan, 1983] and its descendants. And search control rules, in the form of "Given search situation x, perform activation and strength modifications y", might be learned with statistical learning techniques.

8.1.4 Other Representations

In Chapter 4, we outline a method of representing devices very abstractly, argue that the method is important for recognizing useful analogies for innovative design, and argue that the device models presented are derivable from more specific representations, such as Forbus's QPT. Unfortunately, most device models in real knowledge bases are not represented so abstractly as in Chapter 4, and we do not specify an exact method for deriving these abstract models. Automatically generating a model at a desired level of abstraction for a particular task is an interesting open question, one which has been addressed in [Falkenhainer and Forbus, 1988].

8.2 Contributions

The major research contributions of this thesis can be summarized as follows:

- It identifies a class of analogies, called *semantically distant analogies*, and demonstrates their importance in creativity. It also provides a search-theoretic definition of semantic distance and semantically distant analogy. These definitions are motivated by the recognition that some analogies will be harder to find in a knowledge base than others, and they attempt to isolate the features of the difficult-to-find analogies which make search for them difficult.

- It describes an analogy retrieval mechanism, KDSA, which is demonstrated to be a tractable mechanism for retrieving semantically distant analogues from a large knowledge base. Two methods are used in the demonstration of tractability: a theoretical model of the search cost of KDSA, and experiments showing KDSA's retrieval effort in different knowledge bases.

- It describes a similarity metric for identifying useful analogues for the task of innovative design. This metric is demonstrated to be useful in efficiently guiding a search through a knowledge base toward a device which meets it.

- It provides a theoretical framework for evaluating different methods of knowledge base search. This framework is used to analyze the cost of two different KB search methods: standard spreading activation and KDSA.

- It identifies the theoretical cost of breadth-first search in a graph under the assumptions of the theoretical model.

Bibliography

[Anderson and Thompson, 1989] J. R. Anderson and R. Thompson. Use of analogy in a production system architecture. In S. Vosniadou and A. Ortony, editors, *Similarity and Analogical Reasoning*, pages 267–297. Cambridge University Press, Cambridge, 1989.

[Anderson, 1983] J. R. Anderson. *The Architecture of Cognition.* Harvard University Press, 1983.

[Bhatta and Goel, 1993] S. R. Bhatta and A. K. Goel. Learning generic mechanisms from experiences for analogical reasoning. In *Proceeedings of the Fifteenth Annual Conference of the Cognitive Science Society*, Boulder, CO, 1993.

[Brown and Chandrasekaran, 1989] D. C. Brown and B. Chandrasekaran. *Design Problem Solving: Knowledge Structures and Control Strategies.* Morgan Kaufmann Publishers, Inc., 1989.

[Carbonell, 1983a] J. G. Carbonell. Derivational analogy and its role in problem solving. In *Proceedings AAAI-83*, pages 64–69, Washington, D.C., August 1983.

[Carbonell, 1983b] J. G. Carbonell. Learning by analogy: Formulating and generalizing plans from past experience. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 137–162. Tioga Press, Palo Alto, 1983.

[Clark and Maier, 1988] M. Clark and F. Maier. Looking for lumps. *Newsweek*, page 77, October 31 1988.

[Cohen and Kjeldsen, 1987] P. R. Cohen and R. Kjeldsen. Information retrieval by constrained spreading activation in semantic networks. *Information Processing and Management*, 23(4):255–268, 1987.

[Collins and Loftus, 1975] A. M. Collins and E. F. Loftus. A spreading-activation theory of semantic processing. *Psychological Review*, 82:407–428, 1975.

[Crouch, 1971] T. D. Crouch. How the bicycle took wing. *American Heritage of Invention and Technology*, 2(1):15–16, Summer 1971.

[Falkenhainer and Forbus, 1988] B. Falkenhainer and K. D. Forbus. Setting up large-scale qualitative models. In *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI-88)*, pages 301–306, St. Paul, Minnesota, August 1988.

[Falkenhainer, 1988] B. Falkenhainer. *Learning from Physical Analogies: A Study in Analogy and the Explanation Process.* PhD thesis, University of Illinois at Urbana-Champaign, 1988.

[Forbus, 1984] K. D. Forbus. Qualitative process theory. *Artificial Intelligence*, 24:85–168, 1984.

[Gentner, 1983] D. Gentner. Structure mapping: A theoretical framework for analogy. *Cognitive Science*, 7(2):155–170, April–June 1983.

[Gentner, 1987] D. Gentner. Analogical inference and analogical access. In *Analogica: Proceedings of the First Workshop on Analogical Reasoning.* Pitman Publishing Co., 1987.

[Gero, 1990] J. S. Gero. Design prototypes: A knowledge representation schema for design. *AI Magazine*, 11(4):26–36, Winter 1990.

[Goel, 1989] A. Goel. *Integration of Case-Based Reasoning and Model-Based Reasoning for Adaptive Design Problem Solving.* PhD thesis, The Ohio State University, 1989.

[Greiner, 1988] R. Greiner. Learning by understanding analogies. *Artificial Intelligence*, 35:81–125, 1988.

[Hart *et al.*, 1968] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determinations of minimal cost paths. *IEEE Transactions on SSC*, SSC-4:100–107, 1968.

[Hayes-Roth, 1990] B. Hayes-Roth. Architectural foundations for real-time performance in intelligent agents. *Journal of Real-Time Systems*, 2:99–125, 1990.

[Holland *et al.*, 1986] J. H. Holland, K. J. Holyoak, R. E. Nisbett, and P. R. Thagard. *Induction: Processes of Inference, learning, and Discovery.* MIT Press, Cambridge, Massachusetts, 1986.

[Holyoak and Koh, 1987] K. J. Holyoak and K. Koh. Surface and structural similarity in analogical transfer. *Memory and Cognition*, 15:332–340, 1987.

[Hughes, 1971] T. P. Hughes. How did the heroic inventors do it? *American Heritage of Invention and Technology*, 1(2):22–23, Fall 1971.

[Jones, 1989] R. Jones. Learning to retrieve useful information for problem solving. In *Proceedings of the Sixth International Workshop on Machine Learning*, pages 212–214, Cornell University, 1989.

[Kedar-Cabelli, 1985] S. T. Kedar-Cabelli. Purpose-directed analogy. In *Proceedings of the Seventh Meeting of the Cognitive Science Society*, August 1985.

[Kjeldsen and Cohen, 1987] R. Kjeldsen and P. R. Cohen. The evolution and performance of the grant system. *IEEE Expert*, 2(2):73–79, 1987.

[Knoblock, 1991] C. A. Knoblock. *Automatically Generating Abstractions for Problem Solving.* PhD thesis, Carnegie Mellon University, 1991.

[Kock, 1978] W. Kock. *The Creative Engineer.* Plenum Press, 1978.

[Koestler, 1965] A. Koestler. *The Act of Creation.* Macmillan, 1965.

[Kolodner and Simpson, 1984] Janet L. Kolodner and Robert L. Simpson. Problem solving and dynamic memory. Technical Report GIT-ICS-84/24, Georgia Institute of Technology, November 1984. To be published in *Memory, Experience, and Reasoning.*

[Kolodner and Thau, 1988] Janet L. Kolodner and Robert Thau. Design and implementation of a case memory. Technical Report GIT-ICS-88/34, Georgia Institute of Technology, October 1988.

[Kolodner and Wills, 1993] J. L. Kolodner and L. M. Wills. Paying attention to the right things: Issues of focus in case-based creative design. In *Papers from the 1993 Workshop on Case-Based Reasoning*, pages 19–25, Washington, D. C., 1993.

[Kolodner, 1984] Janet L. Kolodner. Memory for experience. Technical Report GIT-ICS-84/23, Georgia Institute of Technology, November 1984.

[Kolodner, 1991] J. L. Kolodner. Improving human decision making through case-based decision aiding. *AI Magazine*, 12(2):52–68, 1991.

[Laird *et al.*, 1987] J. E. Laird, A. Newell, and P. S. Rosenbloom. Soar: An architecture for general intelligence. *Artificial Intelligence*, 33, 1987.

[Langley and Jones, 1988] P. Langley and R. Jones. A computational model of scientific insight. In R .J. Sternberg, editor, *The Nature of Creativity: Contemporary Psychological Perspectives*, pages 177–201. Cambridge University Press, Cambridge, 1988.

[Lenat and Guha, 1990] D. B. Lenat and R. V. Guha. *Building Large Knowledge-Based Systems: Representation and Inference in the CYC Project.* Addison-Wesley, 1990.

[Lenat, 1976] D. B. Lenat. *AM: An Artificial Intelligence Approach to Discovery in Mathematics as Heuristic Search.* PhD thesis, Stanford University, 1976.

[Lowerre and Reddy, 1980] B. Lowerre and R. Reddy. The HARPY speech understanding system. In W. Lea, editor, *Trends in Speech Recognition*, pages 340–360. Prentice-Hall, Englewood Cliffs, N.J., 1980.

[Macaulay, 1988] D. Macaulay. *The Way Things Work.* Houghton Mifflin, 1988.

[Middendorf, 1981] W. H. Middendorf. *What Every Engineer Should Know About Inventing.* Marcel Dekker, Inc., 1981.

[Pearl and Korf, 1987] J. Pearl and R. E. Korf. Search techniques. *Annual Review of Computer Science*, 2:451–467, 1987.

[Quillian, 1968] M. R. Quillian. Semantic memory. In M. Minsky, editor, *Semantic Information Processing*, pages 227–270. MIT Press, Cambridge, Mass., 1968.

[Quinlan, 1983] J. R. Quinlan. Learning efficient classification procedures and their application to chess end games. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 463–482. Tioga Press, Palo Alto, 1983.

[Rau, 1987a] L. F. Rau. Knowledge organization and access in a conceptual information system. *Information Processing and Management*, 23(4):269–283, 1987.

[Rau, 1987b] Lisa F. Rau. Information retrieval from never-ending stories. In *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, pages 317–321, July 1987.

[Rau, 1989] L. F. Rau. Exploiting the semantics of conceptual graphs for efficient graph matching. In *Proceedings of the Third Annual Workshop on Conceptual Graphs*, 1989.

[Riesbeck and Schank, 1989] Christopher K. Riesbeck and Roger C. Schank. *Inside Case-Based Reasoning*. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1989.

[Rossman, 1931] J. Rossman. *The Psychology of the Inventor*. The Inventors, 1931.

[Sacerdoti, 1974] E. D. Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, 5:115–135, 1974.

[Simon *et al.*, 1985] M. K. Simon, Omura, Scholtz, and Levitt. *Spread Spectrum Communications, Vol. 1*. Computer Science Press, 1985.

[Sowa, 1984] J. F. Sowa. *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley, 1984.

[Thagard *et al.*, 1990] P. Thagard, K. J. Holyoak, G. Nelson, and D. Gochfeld. Analog retrieval by constraint satisfaction. *Artificial Intelligence*, 46:259–310, 1990.

[Turner, 1992] S. Turner. *MINSTREL: A Model of Storytelling and Creativity*. PhD thesis, University of California, Los Angeles, 1992.

[Winston, 1980] P. H. Winston. Learning and reasoning by analogy. *Communications of the ACM*, 23(12):689–703, 1980.

[Winston, 1982] P. H. Winston. Learning new principles from precedents and exercises. *Artificial Intelligence*, 19(3):321–350, November 1982.

[Wolverton and Brownston, 1994] M. Wolverton and L. Brownston. BB1 v3.2 manual. Technical report, Knowledge Systems Laboratory, Dept. of Computer Science, Stanford University, 1994. forthcoming.

[Wolverton and Hayes-Roth, 1993] M. Wolverton and B. Hayes-Roth. Using controlled knowledge search to retrieve cross-contextual information. In *Working Notes of the AAAI Spring Symposium on Case-Based Reasoning and Information Retrieval—Exploring the Opportunities for Technology Sharing*, pages 133–139, March 1993.

[Zito-Wolf and Alterman, 1993] R. Zito-Wolf and R. Alterman. A framework and an analysis of current proposals for the case-based organization and representation of procedural knowledge. In *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-93)*, pages 73–78, July 1993.