Computational Approaches to Analogical Reasoning: A Comparative Analysis

Rogers **P. Hall**
Department of Information and Computer Science, University of California, Irvine, CA 92717, U.S.A.

Recommended by Jaime Carbonell

ABSTRACT

Analogical reasoning has a long history in artificial intelligence research, primarily because of its promise for Ike acquisition unit effective use of knowledge. Defined as a representational mapping from a known "source" domain into a novel "target" domain, analogy provides a basic mechanism for effectively connecting a reasoner's past and present experience. Using a four-component process model of analogical reasoning, this paper reviews sixteen computational studies of analogy. These studies are organized chronologically within broadly defined task domains of automated deduction, problem solving and planning, natural language comprehension, and machine learning. Drawing on these detailed reviews, a comparative analysis of diverse contributions to basic analogy processes identifies recurrent problems for studies of analogy and common approaches to their solution. The paper concludes by arguing that computational studies of analogy are in a slate of adolescence: looking to more mature research areas in artificial intelligence for robust accounts of basic reasoning processes and drawing upon a long tradition of research in other disciplines.

1. Introduction

Over the last two decades, metaphor and analogy have grown to occupy a central position in artificial intelligence research, for both practical and theoretical reasons. In problem solving and learning, analogical reasoning promises to overcome the explosive search complexity of finding solutions to novel problems or inducing generalized knowledge from experience. In natural language understanding, metaphor and analogy set practical goals for communication interfaces and theoretical goals for understanding figurative thinking and expression. Analogy presents a basic and challenging epistcinological question: when are two representational descriptions, for some purpose, alike? Interest in these problems has produced a sizable literature of computational approaches to analogy.

Artificial Intelligence 39 (1989) 39-120
0004-3702/H9/$3.50 © 19K9· Elscvicr Science Publishers B.V. (North-Holland)

This paper proposes an abstract process model of analogical reasoning and uses the model to frame a comparative review of studies from the computational literature. Early influential studies of analogy are examined first, followed by more recent studies organized chronologically within traditional artificial intelligence problems: automated deduction, problem solving and planning, natural language processing, and learning. These reviews are summarized in Appendices A and B. The latter half of the paper uses the results of these studies to focus on process model components and considers how computational proposals for analogy solve problems within each component. The detailed reviews and comparative analysis can be read separately. The paper concludes with an overview of problems, proposed solutions, and a prospectus for further computational research on analogy.

The review covers sixteen different projects but is not exhaustive, instead reflecting the author's interests and access to various literatures. One notable omission is connectionist approaches to classification and learning (e.g., see Rumelhart et al. [98] or Hofstadter et al. [53]), where concepts like schematic blending offer a novel approach to analogy. An experienced reader may find other more or less glaring omissions. Reviews of these and other significant studies will undoubtedly appear as computational study of analogy continues.

1.1. Terminology: Materials for analogical reasoning

Polya's studies of heuristics in mathematical problem solving have been a primary inspiration for computational studies of analogy. Central among his heuristics is to use what one already knows:

> Many of these questions and suggestions aim directly at the *mobilization* of our formerly acquired knowledge: *Have you seen it before! Or have you seen the same problem in a slightly different form! Do you know a related problem!* (emphasis in original) [94, p. 146]

Analogy to existing (or newly generated) problems gives a heuristic problem solving strategy in which the result or method of the related problem can be carried into the new problem. The analogy itself consists of a "community of relations" in common between corresponding parts of the problems. As described by Newell [87], these and other aspects of Polya's work provide a rich program for research in artificial intelligence. Among other things, the studies reviewed in this paper attempt to operationalize the concept of analogy as heuristic inference.

Before reviewing computational studies of analogy, a common terminology will be introduced and maintained (as possible) throughout the paper. Analogy will be described as a *mapping*[1] between elements of a *source* domain and a *target* domain. Abstracting over different representational schemes, each do-

[1] See Herstein [49] for a concise mathematical description of mappings.

main simply contains related knowledge or facts (e.g., about the card game of spades). Each fact is a description (e.g., what to bid given a certain hand) consisting of related elements (e.g., the concept of a bid). The analogy mapping associates or maps elements and descriptions from the source domain into the target domain. Analogy becomes useful in some context when a reasoner is familiar with the source, and can map familiar elements or relations from the source into unfamiliar (or unknown) elements or relations in the target. These mapped elements are *analogical inferences* and receive varying levels of support from other mapped elements, from knowledge about the target domain, or from more elaborate confirmatory schemes.[2]

Treating analogy as a mapping introduces organizing terminology for parts of analogy, its processes, and its results. Formal properties of mappings (e.g., special classes of mappings or inverse images) suggest properties that analogies, as used by humans or machines, may or may not have. For example, Centner [33, 34] argues that humans prefer "systematic" analogies that map higher-order relations (e.g., cause or revolves-around) in a one-to-one fashion from source to target: a theoretical assertion supported by a broad spectrum of empirical evidence. Likewise, computational studies of analogy choose particular classes of mappings. For example, Kling [66] allows one-to-many mappings from source to target clauses in an automatic theorem proving task, but restricts predicate mappings to be one-to-one. Since clauses arc composed of predicates, the predicate mapping restriction constrains the space of possible clause mappings. For both Centner and Kling, the analogical mapping promotes analogical inferences and provides a mechanism for transferring relevant information from source to target. What gets transferred is quite varied in the reviewed studies, ranging from selective constraints on target inferences to justified plans for complete target solutions.

Source, target, mapping, analogical inference, and confirmatory *support* are the basic materials of analogy. Computational studies of analogy must adequately represent these materials, specify processes for accessing, manipulating, and creating these representations, and then integrate these representations and processes into existing computational accounts of reasoning and learning. Recognizing analogous sources, elaborating and extending analogical mappings, evaluating the support of analogical inferences, and consolidating confirmed inferences are the basic process components of computational research on analogy.

1.2. Contributions from other disciplines

Artificial intelligence has been on the scene for three decades, and researchers have been publishing studies on analogy for the past two decades. However,

[1] IIesse [50] gives an accessible and revealing philosophical treatment of analogical inference in scientific discovery and argumentation.

research traditions in other disciplines are longer-lived and more prolific. Psychology, linguistics, philosophy and related disciplines provide a diversity of views on figurative thought, its development, use, and significance in human affairs. A detailed review of these literatures here is simply impossible,[3] although references are included wherever appropriate in the following sections.

These literatures support a number of general conclusions about metaphor and analogy which may be useful for computational approaches. Four issues are particularly important: the prevalence of analogy in human reasoning, likely representational structures supporting analogy, corresponding process models of analogy, and the role of analogy in learning. By most accounts, metaphor and analogy are ubiquitous within our cognitive experience. However, estimates of their importance vary widely: from analogy as misleading ornamentation [80] to an indispensable epistemological bridge for acquiring new concepts [91]. Second, these disciplines have shifted towards representations which make explicit the "higher-level" structure of similarity between the source and target of an analogy [89]. Thus "schematic" or "structured" knowledge representations are often preferred. Third, process models of analogical reasoning generally make a distinction between access to an analogical source and its use [35, 38, 88]. These may be governed by considerably different constraints, leading to a growing appreciation for the necessity of studying analogy within a context of use (e.g., analogical problem solving) and in a manner which is sensitive to characteristics of the reasoner (e.g. expertise). Fourth, although "learning by analogy" is generally accepted within the artificial intelligence community, other disciplines do not differentiate it as explicitly from more general forms of learning. For example, some psychological studies of problem solving treat analogy as a reasoning or problem-modeling mechanism, where learning can facilitate spontaneous access to an analogy (e.g., schema induction in Gick and Holyoak [39]).

While these literatures do not provide clear-cut answers to the underlying structures or processes of analogical reasoning, they do provide conceptual frameworks which pose interesting questions and propose partial answers. Computational research should consider these frameworks in detail, building upon and contributing to an increasingly coherent theoretical and empirical treatment of analogy and metaphor. We have much to gain from this rich tradition of research, and much to contribute as well.

1.3. A descriptive framework for computational studies of analogy

Computational studies of analogy and metaphor are difficult to understand as an integrated whole. Areas of commonality are difficult to extract both because

[3] The interested leader is referred to a number of more extensive reviews or collections from these literatures [10, 46, 56, 60, 71, 88-90].

of the recency of work in artificial intelligence and the variety of task domains in which computational approaches have been attempted. There are few comprehensive reviews of computational studies of analogy [63, 79], a problem which this paper addresses. Before considering the body of computational approaches to analogy, however, a simple process framework for analogical reasoning will be presented, drawing from conceptual frameworks advanced in other literatures mentioned above. This framework gives abstract process components that a relatively complete picture of analogical reasoning, computational or otherwise, would include:

(1) *recognition* of a candidate analogous source, given a target description,
(2) *elaboration* of an analogical mapping between source and target domains, possibly including a set of analogical inferences,
(3) *evaluation* of the mapping and inferences in some context of use, including justification, repair, or extension of the mapping,
(4) and *consolidation* of the outcome of the analogy so that its results can be usefully reinstated in other contexts.

These components provide a conceptual organization for comparing computational approaches to analogy. To make detailed comparisons of individual studies possible, concrete descriptions of representation, system inputs, and system outputs are included, along with examples of implemented or proposed processing wherever possible. More evaluative criteria include the completeness of process specification (across components described above), specificity of these descriptions at an implementational level, plausibility of described performance (cither as a result of implementation or optimism), and the generality or extensibility of a model to wider or different domains. As always, critique comes easier than creative solutions to clearly stated and important problems. Critical comments are presented in the hope of identifying these problems, existing solutions, and general contributions of computational studies of analogy.

2. Early Computational Paradigms for Analogical Reasoning

Ringle [96] describes early work in artificial intelligence as attempts to get machines to do something (anything) intelligent as an existence proof for computational intelligence. To some extent, the studies discussed in this section represent such an effort. However, these studies also play an ancestral role that shapes approaches to analogical reasoning in subsequent studies. Evans' [28] system for solving proportional analogies focuses on elaborating an analogical mapping between source and target descriptions. Becker's [9] model of analogical processes in induction embeds analogical comparison in a more general problem solving framework, and addresses each of the process components mentioned earlier.

2.1. Evans [28]: Solution of geometric analogies

Evans' ANALOGY system solves geometric analogy problems like the one shown in Fig. 1. A problem consists of eight figures, each composed of one or more objects. The task is to:

> Find the rule by which figure A has been changed to make figure B. Apply the rule to figure C. Select the resulting figure from figures 1 to 5 [20, p. 272]

These problems are written more compactly as A:B::C:{choices 1, 2, 3, 4, or 5} where the A:B pair and relations between their objects serve as a given source. A correct C:choice pair with corresponding relations serves as the target. These problems have a long history in psychological studies of intelligence, where solution processes are hypothesized to be central components of reasoning ability. While Evans compares his model with gross human performance (e.g., the number of problems correctly answered), others study this task from a more rigorous empirical perspective (e.g., see Grudin [44], Hunt [57], Rumelhart and Abrahamson [97], and Sternberg [104]).

As input, ANALOGY takes primitive prepositional descriptions of figures and decomposes each into constituent objects, a process that Evans describes at length. The resulting representation includes objects and binary relations over those objects within each figure. The relations INSIDE, ABOVE, and LEFT provide a vocabulary for typical geometric analogies. Given this higher-level description, ANALOGY solves problems in four steps:

Step 1. Generate rules that express how figure A can be altered to give figure B.
Step 2. Generate rules that express how figure C can be altered to give each of the choice figures.

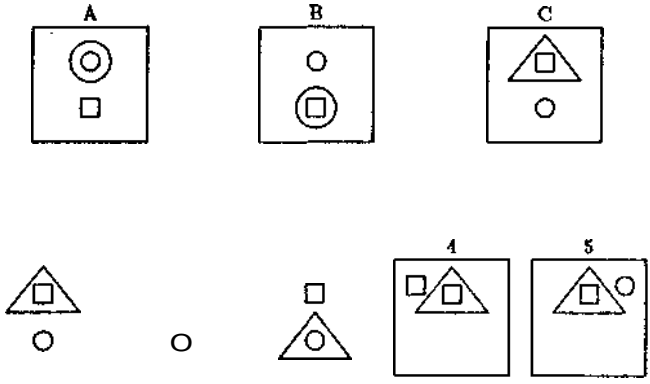

Fig. 1. A geometric analogy problem solved by ANALOGY (Evans [28, p. 330]).

Step 3. Compare each A:B rule with each O.choice rule. For each admissible comparison, find more general candidate rules that cover both of the original rules.

Step 4. Select a single candidate rule that takes C into a choice but still preserves the most information about the A:B rule.

As output, the system returns a single best choice if one can be found.

Rules are generated by considering a restricted class of object-level mappings between two figures. An object in one figure maps to an object in another figure only if both are the same geometric type (i.e., they have the same shape). Differences in objects' size or orientation are accommodated by selecting from a set of Euclidean transformations including scale change, rotation, and horizontal or vertical reflection. In Fig. 1, the large circle of A could be mapped to either the large or small circle in B. The latter mapping requires a scale reduction for the large circle in A. If objects in one figure fail to map with any object in the other figure, they are either deleted or added, as in a rule taking C into choice 2. The small square in C maps with nothing in choice 2 and must be deleted, while one of the small circles in choice 2 maps with nothing in C and must be added. Deleting or adding an object which can be mapped is forbidden, resulting in maximal object correspondence. For example, an A:B rule which deletes all objects in A and then adds all objects in B is not considered.

A rule has three components: a set of objects which are deleted, a set which are added, and a set which are mapped from one figure to another. Mapped objects also record Euclidean transformations. The sense of the analogy (e.g., moving the big circle "down" in A:B of Fig. 1) is carried by the binary relations between objects within each figure. Given an object-level correspondence, these relations are simply associated with their respective objects and deleted, added, or mapped by a rule. For example, one rule taking A into B maps large circles, small circles, and small squares without Euclidean transformations. Since all objects in each figure are mapped, none are deleted or added. The rule specifies that relations describing objects in figure A be replaced by relations describing objects in figure B:

iNSiDE(small circle, large circle)		ABOVE(small circle, large circle)
ABOVE(small circle, small square)	Φ	ADOVE(small circle, small square)
ABOVE(large circle, small square)		lNSIDE(small square, large circle)

Thus, mapped objects serve as a substitution list when the rule produces a new relational description. If unmapped objects are deleted or added, any relation they participate in is also deleted or added. The result is a rule that can be applied to one figure to yield another. Objects can be mapped and transformed in different ways, so multiple rules can be generated for a pair of figures.

There are also many ways of comparing two rules, leading to a sizable space

of candidate answers. Rule comparisons are admissible only if they have the same number of object mappings, object additions, and object deletions. Each admissible comparison is used to generate all pairings of elements from within these rule components, regardless of object type. This yields a pool of candidates whose size is roughly exponential in the size of the largest rule component. Returning to Fig. 1, each admissible comparison yields six candidates since there must be three mapped objects, no additions, and no deletions. Each candidate is generalized by dropping A:B relations or transformations which are not found in the Qchoice rule. ANALOGY ranks candidates by using a similarity metric which rewards candidate length (discriminability) and prefers simple over complex Euclidean transformations. The system chooses the candidate with highest rank, in effect having found a target relationship for a single Qchoice that best preserves source relationships from A:B.

For Fig. 1, the system selects answer three. The chosen candidate has no additions, deletions, or transformations of mapped objects. During rule comparison, object mappings in A:B are paired with mappings in C:choice as follows:

-large circles::large triangles,
-small squares::small circles,
-small circles::small squares.

Given this correspondence, the relations which move the large circle (triangle) enclosure from above to below are common to both rules and retained. According to Evans, this problem is the most difficult for his system, generating 36 candidates for evaluation. On a set of problems drawn from American Council on Education examinations, ANALOGY scores at a level roughly comparable to median performance in a population of high school students.

In overview, Evans concentrates on elaborating an analogy between two representations drawn from the same problem domain. Elaboration maps elements at two levels: direct object mapping during rule generation, and object role mapping during rule comparison. Requiring mapped objects to be the same type (e.g., both are squares) constrains elaboration but still allows for differences in size and orientation. The second level is constrained by requiring that mapped objects play a similar role (i.e., deletions, additions, or mappings) in source and target rules, without any type restriction. During elaboration of an analogical mapping between figure pairs, relations between objects arc included but only used when generalizing and ranking candidates. Making object-level comparisons alone, ANALOGY cannot find solutions to problems requiring transformations of relations as well as objects. For example, if an ABOVE relation in A:B should correspond to a LEFT relation in the preferred Qchoice, the system would drop these relations during generalization and might not be able to discriminate correctly among ranked candidates. Evans

describes an extension to ANALOGY that uses a fixed set of relation substitutions (e.g., substitute LEFT for ABOVE) to generate variations of A:B rules when a "best" object-level candidate cannot be found. This only allows the system to find solutions covered by the set of relation substitutions. Some aspects of ANALOGY incorporate specific knowledge of geometry, although mechanisms which find rules to alter figures, generalize those rules, and choose among candidates are independent of domain-specific knowledge. These mechanisms could apply to other problem domains given an adequately descriptive set of objects and relational predicates.

Evans briefly discusses issues of recognition, evaluation, and consolidation by proposing a combination of ANALOGY mechanisms with a general problem solving system (Newell et al. [86]). The system would use analogical comparison to select applicable rules for a target problem and include two learning mechanisms: "packaging" instantiations of successfully applied rules for later use and constructing "least common generalizations" over multiple uses of the same rule.' In each case, the performance capacity of the problem solver could improve as analogical comparisons extend its repertoire of methods.

2.2. Becker [8, 9]: Analogical processes in induction

Becker describes JCM, a model of analogical reasoning and schema induction. Functional components of the model are described with examples from two task domains: interpretation of observations in a world of animate characters [8] and planning action sequences for a simple robotic arm [9]. JCM is proposed as a cognitive model of basic sensorimotor processes needed for learning to recognize common objects or navigate in a familiar environment. Analogy serves two purposes: to use existing knowledge when interpreting novel situations, and to organize related information prior to inductive learning.

The basic representational unit in JCM is the *schema,* composed of *nodes, kernels,* and *events.* Nodes represent atomic concepts in the world (e.g., "fireman" as a class of objects); kernels represent relational predicates over nodes (e.g., (member Wilfred fireman)); and events represent a conjunction of predicates which make an assertion about the world:

```
{(wears Wilfred AA)
 (member AA suspenders)
 (property AA red)
 (time @ Thursday)}
```

The symbol, @, in the last kernel refers to the enclosing event. Events are combined to form schemata.

[4]Each suggestion appears in later sluciies of machine learning. For example, see Anderson [5] for a discussion of *[>rc>ce<luralization* and Winston |i()7] for an early treatment of learning from examples.

A schema is a rule-like structure with antecedent and consequent events. When interpreted, the left side (antecedent) gives a procedure for achieving the right side (consequence). Each schema includes a numeric estimate of confidence that records its relative frequency of successful use. When confidence drops below threshold, a schema is expunged from memory. Inside a schema, the *criteriality* of nodes within kernels and kernels within events arc represented by integer weights—e.g., {(member[4] Wilfred" fireman[4]):4}. A higher weight means higher salience for the node (or kernel) within its enclosing kernel (or event). Changing criteriality values as a result of experience allows a continuous form of generalization by turning constants into variables (node criteriality goes to zero) or dropping conditions (kernel criteriality goes to zero).

Becker defines analogy as a one-to-one mapping between kernels of two events, requiring consistent node-node pairings. Thus, analogy is an invcrtiblc relation which ranges from minimal to identical correspondence between events. An analogy allows for unmapped source or target kernels, differences in kernel ordering within events, and non-identical nodes within mapped kernels. A heuristic measure of mapping quality between two events accumulates over corresponding kernels and nodes in proportion to their criteriality values. To maximize mapping quality, highly criteria! kernels or nodes in the source should map to target components of the same type. Low criteriality source components can map with nothing (i.e., an unmapped kernel) or anything (i.e., dissimilar types) without seriously affecting mapping quality.

Becker uses the symmetric analogy relation to propose a *prediction paradigm* (see Table 1) in which a target observation, *Event\,* is mapped into the left side of a source schema, *Event,*. The schema's right side, *Event* ^ is then inverse-mapped into a target prediction, *(Evenl^).* Alternately, a desired target event (or goal) could be mapped into the right side of a source schema. Inverse-mapping the left side of the source generates a set of target events (subgoals) necessary for achieving the desired event. In either case, an existing schema is the source for the analogy, the novel event is the target, and a predicted event (analogical inference) results from elaborating the mapping between source and target. Forward and backward chaining *schema application* allow JCM to interpret world events or plan simple action sequences.

Given a target event in short-term memory (STM), recognition of a source schema occurs in two phases. Choosing a target kernel as a retrieval cue (e.g.,

Table 1
A prediction paradigm for analogical schema application (adapted from Becker [9, p. 420]

Target events:	*Event',*		*(Eveni'$_2$)*
			A"'
Source schema:	[Even/,	Φ	*Event$_2$]*

(wears Cyrus BB)), JCM first collects all schemata stored in long-term memory (LTM) that have a kernel with an identical predicate symbol (e.g., wears). Mapped kernels in source schemata must also be in an appropriate position (e.g., in the left side of a schema when the system is forward chaining). Schemata in LTM arc stored with an associative organization so that each generic node is directly linked to all schemata which include an instance of that node in some kernel. A large set of candidate source schemata may be retrieved in this first phase of recognition. In the second phase, candidate sources are ranked by a heuristic evaluation of analogical mapping (as described above), confidence in the source, and estimated cost of applying the source.

A "best" source schema is selected and unmapped kernels in its left side (if any) are pursued to further justify the analogy, ordered by their crilcriality in the schema. Justifying kernels may already be in STM or may require further search in LTM for schemata that can produce them. When a schema's left side kernels arc satisfied by target kernels in STM, the kernels in the schema's right side are inverse-mapped to produce a set of target predictions. STM is monitored for arrival of target instances which corroborate predicted kernels, signaling that the source schema has been successfully applied. Results of schema application provide evidence for *schema refinement,* including changes in confidence, cost estimate, and kernel or node criteriality.

Given a successfully applied schema and the analogical mapping between source and target, JCM forms a generalization by adjusting criteriality weights on nodes and kernels. For example, assume that the observation of Wilfred shown earlier is encoded as the left side of a schema in which descriptive kernels (e.g., (wears Wilfred AA)) predict class membership (i.e., (member Wilfred fireman)). Now assume the system observes Cyrus wearing red suspenders in the city of Peoria:

```
{(wears Cyrus BB)
 (member BB suspenders)
 (property BB red)
 (location @ Peoria)}
```

This target event can be interpreted by analogy to the existing LTM schema describing Wilfred. After successfully predicting that Cyrus is a fireman, source and target events and the mapping between them can be used to create a more general schema as follows:

$$\{(\text{wears}^4\ \text{Cyrus}^2\ \text{BB}^2):4\ (\text{member}^4\ \text{BB}^2\ \text{suspendcrs}^4):3\ (\text{property}^4\ \text{BB}^2\ \text{red}"):3\ (\text{location}^4\ @^4\ \text{Peoria}^4):!\ (\text{time}^4\ @^4\ \text{Thursday}^4):!\}\quad \varphi\quad \{(\text{member}^4\ \text{Cyrus}^2\ \text{fireman}^4):4\}$$

Low node criterialities for Cyrus and the suspenders token (BB) allow treating these nodes as variables in future uses of the schema. Low kernel criterialities for location and time eventually allow generalization by dropping these conditions. Essentially the generalized schema slates: "if a man wears red suspenders, then he is likely to be a fireman." Becker cautions that schemata formed in this manner can be meaningless or over-general. Through repeated failure, confidence in a meaningless schema drops below the threshold for retention in LTM. Confidence in an over-general schema converges on a value lower than 1 but above the retention threshold. Becker briefly describes a *differentiation* mechanism which detects this condition and adds discriminating conditions to the left side of the over-general schema.

Becker's work covers aspects of all four process components mentioned in the beginning of this survey. Given a new event to be interpreted, source schemata are recognized by search in an associatively structured LTM. Candidate source schemata are preferred if they have high confidence, low estimated cost, and a high scoring analogy mapping over their criterial components. Unmapped source kernels are inverse-mapped to give analogical inferences which are sought as further justification for the analogy. When these are confirmed, the schema fires and generates target predictions. Predictions may be verified in the external environment, leading to source schema refinement by adjusting criteriality and confidence weights. Target interpretations and refined source schemata are consolidated into LTM for later use and refinement. Although JCM is a relatively complete model of analogy and seeks to integrate analogy within a basic reasoning architecture, many of its components have never been implemented (see [9] for a final description).

3. Analogical Reasoning in Automated Deduction

This section presents three approaches to automated deduction. Kling [65, 66] proposes a mechanism for constraining irrelevant inferences when using a uniform theorem proving approach; Munyer [85J exploits a conceptual parallelism between analogy and unification to reuse portions of known proofs; and Greiner [41-43] introduces a series of heuristic constraints for finding abstraction-based useful analogical inferences.

3.1. Kiing [65, 66]: Restricting starting clauses for automated deduction

Kling describes ZORBA-I, an implemented system that assists an automated proof of a target theorem by elaborating an analogy with a source proof supplied by the user. The system uses the supplied theorem and clauses in its proof (either axioms or proved theorems) to select a near-optimal set of starting clauses to be used in a proof of the target theorem. As shown in Fig. 2, the source theorem, P, includes a known set of clauses, D,, used in its proof. For a new target theorem, P_A, which is analogous to the source, ZORBA-I

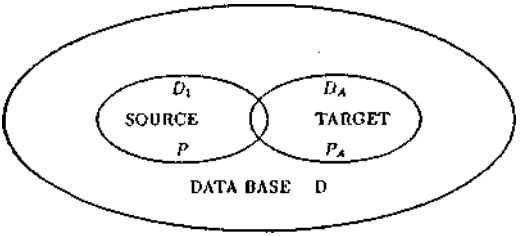

Fig. 2. ZORBA-I restricts starting clauses for a target proof (adapted from Kling [66, p. 149]).

identifies a set of clauses, D_A, for proving the target theorem. These are a strict subset from the database of all available clauses, D. This strategy greatly reduces the number of irrevelant inferences that an automated theorem prover would generate while attempting a proof of P_A from the total database. ZORBA-I is tested on pairs of theorems in abstract algebra like the following:

> The intersection of two abelian groups is an abelian group.
> The intersection of two commutative rings is a commutative ring.

Inputs to ZORBA-I include target and source theorems, clauses used in a proof of the source theorem, and *semantic templates* for each predicate in the domain representation. A semantic template specifies the type of a predicate and each of its arguments. For example, the semantic template for *group\a\ *]* specifies that the predicate is a structure, its first argument is a set, and its second argument is an operator. Descriptive properties also augment individual clauses by recording the role of each predicate within the clause (e.g., positive or negated occurrence). Finally, Kling provides a database of 239 clauses as an axiomatization of abstract algebra. ZORBA-I attempts to select a subset of these clauses for use in proving the target theorem. The target theorem and selected clauses arc then presented to a resolution-based theorem prover, QA3 [40].

Kling focuses entirely on elaborating an analogy between source and target theorems, leaving recognition to the user of ZORBA-I, evaluation as an exercise for OA3, and consolidation for future work. A target set of clauses is selected by incrementally extending an analogical mapping over variables, predicates, and clauses. ZORBA-I first finds a mapping between predicates and variables in each theorem, restricted so that mapped predicates (e.g. group—> ring) and their arguments have the same type. The system then attempts to extend the mapping so that clauses used in the source proof map to target clauses selected from the database. Incremental elaboration of the mapping stops when every clause in the source proof has one or more corresponding target clauses.

ZORBA-I incrementally elaborates an analogical mapping by searching in a space of candidate extensions. A state in this space is a partial analogical mapping that takes some source clauses into corresponding target clauses. An extension is a successor state which maps an additional source clause. Given a

partial analogy map, an extension operator finds a mapping extension in four steps:

Step 1. Select a source, clause with some variables and predicates in the current mapping.

Step 2. Generate a target "image" of the clause using the current mapping.

Step 3. Select a database clause whose description matches the image description.

Step 4. Extend the current mapping to cover this new clause.

By starting with a type-restricted mapping between source and target theorems before considering clauses used in the source proof, ZORBA-I reduces an exponential space of candidate analogical mappings by several orders of magnitude. Choosing extensions where source and target clauses have maximal overlap, the system performs a best-first heuristic search through the space of candidate analogical mappings. Although not guaranteed to find a target proof, this approach is effective in practice.

ZORBA-l does not construct analogies at the level of a proof plan, although Kling [66] finds this approach "attractively elegant" (p. 148). Instead, Kling uses analogy to modify the problem solving *environment* by selecting a restricted set of starting clauses for a completely separate proof procedure. He argues that this avoids problems with diagnosing and repairing plan failures and notes that planning-oriented problem solvers were not available when ZORBA-i was conceived. Thus Kling's approach to reasoning by analogy is considerably different than Becker's [9] prediction paradigm. As noted by Miller [79], Kling's use of analogy is,

> Rather as though, possessing a recipe for lamb casserole, and wishing to cook a beef stew, we noted that we were likely to need beef, onions, potatoes, carrots, stock, salt, an oven, a knife, a dish and a work-surface, and then threw away the recipe book without reading the method of preparation. [79, p. 33]

Paradigmatic reservations aside, when ZORBA-I is applied to pairs of nontrivial problems in algebraic group theory, the results are encouraging. The system selects starting clause sets only slightly larger than optimal. Without restricting the starting database of clauses, these proofs would exhaust computational resources before termination.

3.2. Munyer [85]: Analogy by implicit and explicit planning

Munyer describes two methods for using analogies to assist automated deduction: implicit and explicit planning. In each case, an analogy mechanism is grafted onto a search-based problem solving framework for making deductive inferences, and its performance is compared with weaker methods. Munyer's

task is to prove two mathematical expressions equivalent, for example:

$$\ddot{Y} \quad , \_ \; \underline{nR^{\ldots} - (/; + 1)\ldots + R}$$

An *implicit planning method* finds a sequence of operators (e.g., a change of variable) that transform one expression into the other. While searching for a solution to the target problem, this method uses an analogical mapping with the proof of a similar problem to heuristically evaluate candidate expressions and operators in the target solution. For the problem shown above, the implicit planning method examines all known derivations and selects

$$\ddot{y} \; R' = \; 1 \quad p^{\ldots}$$

as an analogical source. Candidate steps in the target problem space are compared with their counterparts in this source derivation. At each step in the target derivation, the target candidate that most closely resembles an analogous step in the source derivation is selected.

The implicit planning method does not use the source derivation directly as a plan, but instead uses the analogy as a heuristic evaluation function to guide an otherwise uninformed search for a target solution. In the example shown above, a 23-step solution sequence is found using a database of axioms with an approximate branching factor of 10. Weak methods alone (e.g., breadth-first search) would make search to this depth infeasible, but implicit planning reduces the branching factor and brings solutions within range. Although alternative heuristic search methods might give a more challenging comparison, Munyer reports that an implementation of the implicit planning method finds solutions to several problems like the example shown above.

Using analogy as a search heuristic, the implicit planning method does not use sequence information contained in source derivations. Most important, the method cannot use intermediate steps in the source derivation to "chunk" the search for a target solution sequence. If k intermediate steps were available at little cost, a search to depth d could be reduced to k searches of depth d/k. Although an exhaustive search with branching factor /; and depth d considers roughly b' states, effective selection and use of k intermediate steps would consider only $kb^{(\ldots)}$ states, giving an exponential improvement with respect to the depth of search. Using a source derivation as a plan outline in this way contrasts with both the implicit planning method and Kling's use of analogy to select a set of starting axioms. These strategics restrict the search branching factor and at best result in a polynomial reduction in search effort.

Ocorgeff [37] discusses both approaches under tlie common theme of discovering strategies in heuristic search.

Munyer's *explicit planning method* directly uses selected source derivations as plan outlines. Although not implemented, this method proposes several interesting mechanisms for anahs&ical problem solving and learning. The central idea is to replace the unification algorithm in a traditional deductive problem solver with an analogical mapping algorithm, and then to treat source derivations as macro-operators when searching for a target solution sequence. Single-step deductive inferences in the target problem space compete with retrieved source derivations in a bidirectional, best-first search paradigm. Candidate search steps, whether target inferences or suggested by analogy, are ordered by their degree of match with a current target problem state. Search terminates when a logically valid target derivation is found.

Given a formula that is a current state in the target problem space, analogous source formulas are recognized by collecting and then filtering a set of candidates indexed in an associative database. Each source formula is indexed by instances of functional containment for nonvariable symbols within the formula. For example, the formula/(a, *b)* would have indices (/, *a)* and (/, *b).* Inference links between formulas are also stored, allowing an entire source derivation to be retrieved through any of its constituent formulas. Using functional containment pairs in the target formula as retrieval cues, source formulas with a high percentage of identical pairs are returned *as* candidates. Overlap in functional containment estimates the quality of an analogical mapping between source and target formulas, bypassing source candidates with relatively little promise. A mapping is then calculated between the target formula and each candidate formula, and candidates that overlap poorly with the target are pruned. Recognition finds a source derivation by examining the derivations rooted at each candidate. A candidate source derivation must contain an operator sequence which terminates in a source formula that consistently maps to a known goal formula in the target search space. By insisting that consistent analogy mappings exist at both ends of the source derivation, Munyer further restricts the number of analogical sources considered.

Munyer finds analogical mappings by extending first-order unification to allow associations between unlike constants or predicates, many-to-one bindings, commutative or associative reordering of arguments within predicates, and erasure of terms.[6] For example, extended unification would map *f(a , g(b))* to *h(g(c), a)* by associating/-*/!, *a—>a, g—>g,* and *b-+c.* This correspondence includes unlike predicates and a commutative reordering of arguments. Elaboration of the mapping between source and target formulas proceeds bottom-up, starting with all possible local maps between like symbols or variables. Each local map is reinforced by the highest-valued maps that it

'Siekmann [102] surveys approaches to a general unification theory that covers these and further extensions to first order unification.

directly dominates (i.e., maps involving its argument symbols), and local maps which reinforce nothing are deleted. In effect, local maps compete to reinforce their parent maps and many consistent but poorly aligned analogies are discarded. Taking the surviving local maps, associations between unmapped symbols with mapped parents are added, and the algorithm returns a single best mapping. A "degree of certainty" (DOC) for the mapping is calculated by finding the percentage of symbols in source and target formulas that correspond. The DOC value ranges from unity if the formulas are first-order unifiable to lesser values when extended associations (mentioned above) are present.

Evaluation of a selected source derivation insures that successive operators produce formulas with consistent analogical mappings to target states. According to Munyer, this condition rarely holds. Instead, mappings often change during plan application, leading to a "skewed" plan. Skewed plans are repaired by inserting plan correction steps to bypass a troublesome operator and to preserve a consistent mapping between source and target derivations. Corrections are found by suspending the plan and using blind search to generate target successor states which are added to the search frontier. The plan is "corrected" when a new target state allows the source plan to be resumed. Just as the original target problem could invoke analogical derivations, so too can plan repairs be effected by analogy. Thus, the explicit planning method is recursively organized around finding and using analogical derivations, but can fall back on single-step, deductive problem solving as necessary.

Consolidation creates a least common generalization over mapped source and target formulas and then assembles successive formulas into a generalized derivation. A formula generalization algorithm introduces variables for mapped but differing symbols and also accommodates merged, permuted, or unmapped symbols. In the latter cases, Munyer augments the generalization with attributes that allow the mapping algorithm to correctly interpret a generalized formula. A derivation generalization algorithm assembles contiguous generalized formulas by storing inference links between them. Generalized formulas and derivations are integrated into the source database, along with the instantiated target derivation. These may influence future problem solving performance when recognized and applied by the explicit planning method. The utility of each stored component is recorded as a ratio of successful to attempted uses, allowing deletion of stored components with a poor performance record. Munyer argues that this provides a plausible model for acquiring problem solving expertise.

In summary, Munyer makes an ambitious proposal for incorporating analogical planning and problem solving within a traditional deductive paradigm. His model is similar to Evans' suggestion for adding analogical comparison to a search-based problem solver, but is presented in more detail. Recognition of source derivations is supported by an associative database, using functional containment as an indexing scheme. Elaboration of the analogical mapping is a

bottom-up, competitive process with output ranging from first-order unification (if possible) to less certain associations between target and source formulas. Analogies between source and target derivations are evaluated by the consistency and DOC of mapped plan steps. Skewed plans can be repaired by resorting to weak methods. Successful target derivations, as well as their generalization with supporting source derivations, are consolidated within the associative database for future problem solving. Although the proposed mechanisms are interesting, an implementation of the explicit planning method could encounter difficulties. Using functional containment to restrict an initial set of candidate source formulas, Munyer would map each into the target formula, and prune this set of candidates only after finding that a subsequent formula in the source derivation could not be consistently mapped to a target goal state. There could be an enormous amount of effort expended before resorting to a weak, search-based approach. Without empirical demonstration or stronger analytic arguments, Munyer's proposal might trade the cost of weak, search-based methods off against an even more difficult problem.

3.3. Greiner [41-43]: Abstraction-based useful analogical inference

Greiner argues that unconstrained analogical inference, defined as conjecturing facts about a target domain from the existence of similar facts in a source domain, is intractable. Too many legal conjectures are available, even when given a hint that a target concept is like a source concept. As a solution, Greiner offers a set of heuristics for constraining the space of generated conjectures. Primary among these heuristics is a preference for existing *abstractions* that give "'coherent' clusters of facts . . . which encode solution methods to past problems." [41, p. 1]. Greiner presents empirical results with NLAG, an implementation that proposes answers to problems in analogically paired domains of electricity and hydraulics.

Within a deductive problem solving framework, Greiner defines an *abstraction-based useful analogical inference* operator as:

$$\mathrm{Th},\ A \sim B\ \ \mathrm{K}\tau\ \ \varphi(<I, \ldots A, \ldots \alpha_{\mu})$$

where
- Unknown: $\mathrm{Th}\hat{}<\mathrm{p}(\ a_{\rho}. \ldots, A, \ldots, a_{n})$
- Consistent: $\mathrm{Th}\hat{}\sim\backslash\varphi(\ \alpha,, \ldots, A, \ldots, \beta,,)$
- Common: $Th\backslash = ip(7bi, \ldots, B, \ldots, lb_{n})$
- Useful: $\mathrm{Th} + \varphi(\alpha_{,}, \ldots, /\mathrm{i}, \ldots \ll,,)\mathbf{M}^{>\mathrm{T}}$
- Abstraction: $\mathrm{Th}|=\mathrm{AbstForm}(<\mathrm{p})$

Given axioms in a theory (Th) for target and source domains, a target problem statement (PT), and a hint that the target concept is like the source concept *(A~ B),* Greiner's analogy operator (|~) yields a conjecture (analogical inference) about the target concept: $\varphi(\alpha_{,}, \ldots, A, \ldots, a_{n})$. "Concepts" are sym-

bols in the theory (e.g., FlowRate in a fluid system), while "conjectures" are propositions containing these symbols. The system conjectures a target instantiation of an existing abstraction.

Application of the analogy operator is constrained so that:

(1) The target conjecture cannot be inferred from the starting theory—i.e., it is **Unknown.** Thus analogy is a plausible but nondeductive inference strategy.

(2) The target conjecture is **Consistent** with the starting theory—i.e., the negation of the target conjecture cannot be logically inferred from the theory.

(3) The conjecture must be **Common** to both concepts, requiring that its source instantiation, $<p(fr,,. . . , B,. . . , b_n)$, is already known or can be inferred from the starting theory.

(4) When added to the starting theory, the target conjecture is **Useful**—i.e., it allows a deductive solution to the target problem, FT.

(5) The conjecture is "tagged" as an **Abstraction** in the starting theory. Only relations which occur a priori in abstractions are considered when pursuing an analogy.

NLAG (see Fig. 3) solves hydraulics problems like: in a fluid system where a total flowrate is diverted through two parallel pipes, find the flowrate through one of the pipes, given information about the total flowrate and characteristics of the two pipes. The unknown flowrate (target solution) is found by analogy to familiar relations about electrical circuits (source domain). As input, NLAG receives a hint that the concept of Current in electricity is like the concept of FlowRate in hydraulics. The system is also given the target problem to solve and a starting theory (Th_{CF}) which includes extensive information about

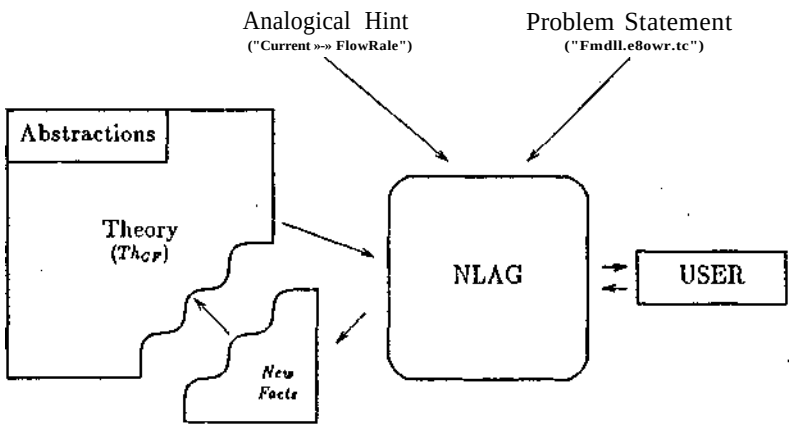

Fig. 3. Overall structure of tlie NLAG system (adapted from Greiner [41, p. 109]).

electrical circuits and few facts about fluid systems. The theory includes a set of domain-independent abstractions that organize general clusters of related facts. For the flowrate problem, four different linear systems abstractions are included in the starting theory. The following abstraction eventually enables a solution to the flowrate problem:

Abstraction(RKK)
RKK(t, *c, r, 1)* « Kirchoffl(<) & Kirchoff2(r) & Ohms(/, *c, r, I),*

Facts in the starting theory define each term in the abstraction (e.g., Kirchoffl). While **RKK(<,** *c, r,* /) is domain-independent, the starting theory supports a source instantiation, RKK(Current, VoltageDrop, Resistance, Resistors). In addition, the starting theory contains instantiations for some of the abstraction's predicates within the target domain (e.g., Kirchoffl(FIowRate)). Finally, the NLAG user screens useful and consistent target conjectures before accepting a solution. As output, the system gives a general abstraction, its instantiation in the target domain (i.e., RKK(FlowRale, PressureDrop, PipeCharacter, Pipes)), and any conjectures necessary to support the target instantiation.

NLAG uses an iterative generate and test control structure: the generator finds a common abstraction for source and target, and a test process determines whether the abstraction is useful for solving the target problem and acceptable to a user of the system. Finding a common abstraction requires three steps:

Step 1. *Find-Kernel.* The source concept is lexically substituted into the target problem, and a backward chaining search process finds rules in the starting theory that contribute to proving the source problem statement. The result is a set of atomic "kernel facts" necessary for deducing the source problem statement.

Step 2. Instantiate-Source. Using forward chaining breadth-first search from the kernel facts, deduce a source instance of an abstraction which includes the concept (e.g., RKK(Currcnt, VoltageDrop, Resistance, Resistors)). This abstraction satisfies the common criterion described above and requires least effort for instantiation vis-a-vis competing abstractions.

Step 3. *Instantiate-Target.* Find a target instance of the abstraction in which the target concept replaces the source concept, and the remaining variables are consistently bound to target concepts from the starting theory. This requires "residual" conjectures which, when added to the starting theory, support a proof of the target problem.

The first and second steps solve the source problem by finding an applicable abstraction. The third step proposes a target solution which re-uses that

abstraction. Residual supporting conjectures for the proposed target solution which cannot be deduced (e.g., Kirchoff2(PressureDrop)) may simply be asserted.

Target instantiations of abstractions are generated, as above, until a proposed solution can be verified:

Step 4. *Useful!* Add the target conjectures to the starting theory and attempt to solve the target problem. Proceed if the proof is successful; otherwise choose another abstraction or instantiation.

Step 5. *Consistent'?* Verify the consistency of each conjecture in a useful abstraction by attempting to prove its negation from the starting theory. Each consistency test includes all other target conjectures in the theory. Proceed if all conjectures are consistent; otherwise consider another abstraction.

Step 6. *Correct!* Ask the user to approve each new conjecture. If any conjecture is rejected, choose a different abstraction.

Consistent and useful target instantiations are evaluated by the user because they can give wrong answers (e.g., RKK(FlowRate, PressureDrop, CrossSection, Pipes). Accepting residual conjectures without deductive certainty, NLAG could not detect these errors unassisted.

A variety of heuristics prune and order the candidate abstractions and instantiations which NLAG generates: consider only abstractions, prefer abstractions which closely match a source solution, consider only source instantiations which bind variables to concepts from a common domain (e.g., hydraulics), prefer instantiations which require fewest supporting conjectures, prefer more general abstractions, and prefer instantiations which bind variables to familiar domain concepts. These heuristic constraints implement the intuition that coherent clusters of facts with demonstrated usefulness are preferred a priori when pursuing a posteriori useful analogies.

To test his model, Greincr contrasts the performance of different versions of the NLAG system, using the number of deductions required to find a target solution as a dependent measure. Adding heuristic constraints on the analogy generator creates successively stronger versions of the system. At one extreme, deducing a target solution with no analogy mechanism exceeds memory limits before terminating. Successive additions of heuristics for selecting abstractions and pruning their instantiations give increasingly efficient performance. For example, when the starting theory contains many highly specific abstractions, adding a heuristic to prefer a maximally general abstraction results in a 15-fold reduction in the number of necessary deductions. Other results are less clear-cut. For example, removing explicit abstraction labels from the starting theory causes only minor degradation. Hence, the starting theory relations are sufficient as sources of analogies, without segregating them as explicit abstractions. Greiner interprets this result positively, characterizing Ihese relations as *"pre-defined* relations — which the ancient scholars, and others, have defined

and named earlier" [41, p. 161]. He does not discuss situations in which the starting theory is missing these relations, contains a very large number of them, or contains an assortment of plausible but competing relations.

Greiner's work addresses each process component considered in this survey. A given point of correspondence (the hint) between target and source domains initiates recognition of a source analog. Lexical substitution generates a source problem, that problem is solved, and kernel facts in the solution are used to identify candidate abstractions. Elaboration instantiates a candidate in the target domain, using the analogical hint as an initial map. Unlike Kling [66] or Munyer [85], however, Greiner further elaborates the analogy by searching for a consistent target instantiation without reference to the source domain or solution. Evaluation is a proof process that verifies a consistent and useful instantiation of the abstraction within the target domain. User acceptance screens any conjectures asserted without deductive certainty within the existing theory. Consolidation adds new and possibly nondeductive instantiations of known abstractions to the starting theory. Additions to the theory could influence later problem solving, although Greiner does not explore this possibility. In overview, the contents of the starting theory exert a powerful influence on analogical reasoning. Based on abstractions or reified relations, a model-driven focus is given to processes of recognition, elaboration and evaluation.

4. Analogical Reasoning in Problem Solving and Planning

Projects reviewed in this section focus primarily on problem solving and planning: analogical reasoning uses solution plans for familiar problems to help solve novel problems. Four approaches are discussed in detail. Brown [11] uses an incrementally extended reduction analogy to transfer problem solving expertise between domains; McDermott [74, 75] transfers and repairs specific problem solving methods to solve new problems in a reactive environment; Carbonell [18, 20] transforms solution paths or re-plays derivational histories in a reconstructive problem solver; and Simpson [69, 103] uses successful or failed problem solving cases to plan a solution for a new problem. Each study also examines learning by analogy.

4.1. Brown [11]: Use of analogy to achieve new expertise

Brown explores a sense of analogy as *reduction,* where a hard problem in one domain is reduced to a familiar and more easily solved problem in another domain. Reduction analogies, in Brown's view, contrast with analogies that use a shared abstraction or refer to a familiar domain during exploration of a novel domain. He describes an unimplemcnted model in which reduction analogies transfer problem solving expertise across domains.

Expertise, what makes some problems easier in Brown's view, consists of a

domain *description* written as axioms in first-order logic, problem solving *plans* that determine predicates or functions in the description, and LISP *programs* that carry out portions of plans. Since an analogy between domains may be inexact, *justifications* for the sufficiency of plans with respect to descriptions help to evaluate transferred knowledge. There are multiple levels of representation for domain knowledge, and analogical reasoning consists of "lifting" material at each level from a source domain of existing expertise into a target domain about which little is known. Brown describes his model in several task domains, but gives primary attention to transfer from plane to solid geometry.

Brown presents a three stage analogy process: *map, solve,* and *lift.* In brief, these stages map a target problem into a source problem, find a source solution, and then inverse-map the solution and its justification back into the target domain for evaluation. The problem solving context gives a source domain, and the mapping process generates a particular source problem. Hence, Brown's model focuses primarily on elaboration, evaluation, and consolidation.

The analogical reasoncr starts with extensive source domain knowledge (e.g., axiomatic descriptions, plans, code, and justifications for plane geometry), is given limited target domain knowledge (e.g., several axioms for solid geometry), and then receives a carefully selected sequence of target problems (theorems to prove). Each problem requires an incremental extension of the analogical mapping between domains. Plans, justifications, and code are transferred as needed over the extended mapping, and the reasoner becomes increasingly capable of solving target domain problems. As output, tjie system not only solves the presented problems but also acquires an effective representation of expertise for the target domain.

Brown abstractly describes process components of his proposal and relies on an extended series of examples to demonstrate techniques for mapping, solving, and lifting problems. We examine what he calls a "trivial" example of reduction from solid to plane geometry. The target problem in solid geometry is: *Show that the intersection of distinct lines containing a point, C, is that point C.*

```
(for-all (K L C) (implies (and (In L K) (pt C)
                               (in-ln K C) (in-ln L C))
                          (equal C (intersect K L))))
```

The initial representation of line intersection in the source domain of plane geometry is relatively complete and, most important for this problem, includes an axiom for line intersection, a plan for finding the intersection of two lines, code for implementing this plan, and a justification that the execution of the plan would find the intersection of these lines (see Table 2). Brown develops these representation schemes in some detail, but most important for his model

Table 2
Axiom, plan, and justification for line intersection in plane geometry (adapted from Brown [Ilj)

```
(determines (intersect A B)                          ; function form
            ((In A) (In B) (distinct A B))           ; argument restrictions
            ((in-In A (intersect A B))               ; restrictions on returned value
             (in-In B (intersect A B))))

(to-lind find-intersect (intersect LI L2)
            (bind P)                                 ; find an object, P. such that
:FI1        (condition (In LI))                      ; the following conditions hold
:FI2        (condition (In L2))
:FI3         (condition (distinct LI L2)
:FI4        (pattern LI In (* ?P *))                 ; and P is in both lines
:FI5        (pattern L2 In (* P *))
            (return P))

(plan-justification flnd-inlcrsect
            (LI (In LI) %condition FI1)              ; refers to plan step label
            (L2 (In L2) %ccmdition FI2)
            (L3 (distinct LI L2) %condition FI3)
            (L4 (in-In LI P) RC1 FI4)                ; a representation claim
            (L5 (in-In L2 P) RC1 FI5)
            (L6 (in-In LI (intersect LI L2)) P-DEF1 LI L2 L3)
            (L7 (in-In L2 (intersect LI L2)) P-DEFI LI L2 L3)
            (L8 (equal P (intersect LI L2)) P-THM22 LI L2 L3 L4 L5 L6 L7))
```

of reduction analogies is that plan justification entries organize a set of axiomatic supports for the correctness of plan steps. For example, justification entry L4 asserts that object P is in line LI if representation claim, RC1, is true. This supports plan step :FI4 which finds a point in line LI. We return to plan step justifications in a moment.

Since the initial representation of solid geometry contains no method for determining line intersections, Brown's problem solver is unable to evaluate the intersection function in (equal *C* (intersect *K* L)) when attempting to solve the target problem. This impasse invokes reduction analogy processes. First, the *map* process finds an initial mapping between the target solids problem and the source domain of plane geometry as follows:

lιτ-»lη ,
pti->pt,
in-ln 冲 in-ln ,
intersect»-»intersect.

From the initial mapping, the reasoner generates a related problem in the source domain.

To initiate or extend an analogical mapping, selected elements from the target domain are placed in correspondence with elements in the source

domain description. Target elements are selected by summarizing the target problem statement and a set of closely related facts. The summary includes type information carried by "semantic templates" for predicates and functions in the domain description, much as in Kling [66]. The analogy mapping is found in two stages:

(1) Find an identical mapping between types in the target summarization and types in the source domain description. These initial mappings must preserve the compatibility of source domain predicates and their arguments.

(2) Extend the mapping using a depth-first search to include other elements (e.g., objects and predicates) of the target summarization and source description.

The search first considers source elements with most restrictive types (i.e., fewest members in the class) and backtracks over mappings which arc inconsistent with the source domain description. Constraints on elaboration come entirely from source domain knowledge, which is consistent with Brown's focus on transferring domain expertise. The final mapping is applied to the target summarization to generate an analogous source problem.

If existing source plans and code can be applied, the source problem is solved directly. When a source solution cannot be found, extended or alternative mappings arc sought and used to transfer more information about the target problem. Once a source solution is found, Brown inverts the analogy mapping to lift the solution (instantiated plans and code) into the target domain. The lifted solution is only a candidate, however, since the analogy may still be incomplete or incorrect. To verify the candidate solution, Brown also inverse-maps the source justification showing that plan steps (and attendant code) satisfy a goal statement in the domain description. The resulting target justification is a tentative explanation of why the candidate solution should be correct. If components of the mapped justification are true of the target domain description, then lifted plans and code succeed, and the problem solver acquires new expertise. If the target justification contains unprovable or extraneous components ("bugs"), these must be "patched" to salvage the candidate solution. An unprovable justification may be replaced by some other deductive inference in the target domain, while an extraneous justification component and its plan step can be ignored. In either case, patching bugs usually extends the existing analogy mapping, and patches must be propagated through candidate plans and code. Finally, if the target justification is shown to be false in the target domain description, Brown returns for another analogy.

In the intersection example, existing axioms and plans of plane geometry solve the source problem directly. The *find-intersection* plan generates a source solution which satisfies (equal C (intersect *K* L)) in the problem statement and permits the reasoner to lift the definition, plan, and justification for finding line intersections into the domain of solid geometry. Axiomatic components of plan

justification steps (i.e., P-DEF1, P-THM22, and RCI) are lifted into the target domain by inverse-mapping. P-DEF1 is the axiom defining line intersection, P-THM22 is an axiom which determines a distinct line from two points, and RCI is a "representation claim" which asserts that a point in the LISP representation of a line (i.e., an element in a list) must also be "in" that line at the axiomatic level. If each target justification holds, then the plan for finding line intersection will produce a valid result in solid geometry. Brown confirms the target analog of P-THM22 with a single-step deductive process, leaving target analogs of I'-DEFl and RCI for more complex proof mechanisms. All three justification components are confirmed without bugs, giving an answer to the target problem and an extension to the analogy mapping:

S-THM22 >-> P-THM22
s-find-intersect ·-» find-intersect
S-DEF1-»P-DEF1
S-RC! ·-» RCI

The reasoner acquires a definition and justified method for determining line intersection in solid geometry as a result of a successful reduction analogy.

In summary, Brown's work is interesting in several respects. First, generating source problems by a reduction analogy simplifies the problem of recognizing analogies. Using an ongoing tutorial context and extensive source expertise, Brown need not maintain a store of prior solutions to provide an analogical source. Instead, the reasoner is a source domain expert and can generate source solutions as needed.[7] Second, although Brown's mapping process adds little to the notion of incremental extension demonstrated by Kling [66], it does provide a rationale for relying heavily on source domain constraints (i.e., type compatibility) when pruning inconsistent mappings. Third, he gives a complex description of transferring knowledge at several levels of representation (descriptions, plans, and code), including the evaluation of analogical inferences by confirming their justifications. These act as proofs for the sufficiency of plan steps with respect to goal descriptions, and can be repaired (patched) under some circumstances. Finally, Brown describes a common context for analogical learning: a tutorial dialogue in which a carefully chosen sequence of examples can be solved by reduction analogies to a given source domain. Problem solving methods arc incrementally transferred between domains as the analogical learner climbs a scaffold of successively more difficult problems.

'Although Munyer's source expert uses reduction analogies, the general problem of finding and using "auxiliary problems" and connecting them up is examined in detail by Newell [87j.

4.2. McDermott [74, 75]: Learning to use analogies

McDermott describes ANA, an implemented production system that uses analogies when solving problems in a simulated environment. Analogical reasoning "assimilates" new problems to a given set of methods, while a simple rote learning mechanism "accommodates" problem solving results to improve performance. Assimilation covers recognition, elaboration, and evaluation of analogies, while accommodation corresponds to consolidation. The problem solving framework is a traditional production system, where a recognize/act cycle applies productions whose condition elements match working memory (WM) elements. Special purpose productions support each analogy process by manipulating a mapping between WM elements from the target problem statement and selected elements of an "almost adequate" source method. This mapping allows type discrepancies for corresponding objects or actions in target and source. The success or failure of an applied method is recorded by building productions which bypass analogy when the target problem is seen again.

ANA performs in a simulated paint shop environment where a set of recognizable objects can be found, moved, and sprayed with paint or water (see Fig. 4). Inputs include object locations, six methods for accomplishing

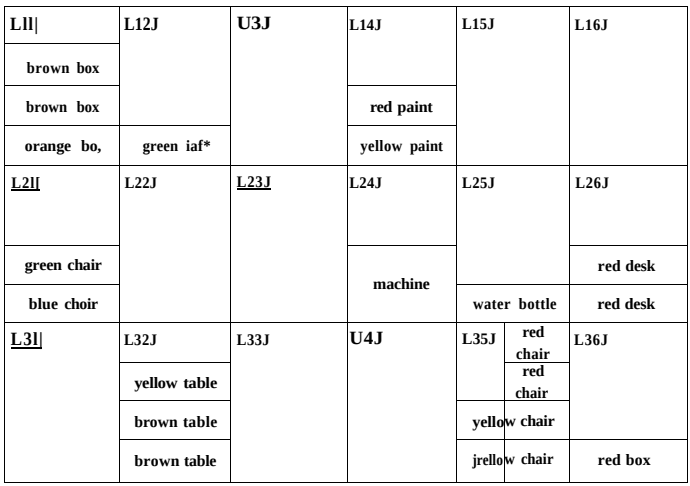

Fig. 4. ANA's initial paint shop (McDermotl [75], p. 570).

The concepts of assimilation and accommodation have a rich history in developmental psychology. See Indurkhya [59] for a discussion of these concepts from a computational perspective.

specific goals (e.g., productions which *paint a table fit L32 red),* and supporting knowledge sources encoded as production rules. These rules define object and action type hierarchies, recognize analogies between problems and methods, manage element mappings for analogies, patch recoverable errors in method application, and record problem solving experience. When ANA'S knowledge sources are insufficient, the user is asked to diagnose and correct errors in method application. As output, the system solves tasks by analogy and acquires productions for solving these tasks more quickly on a second presentation.

Asked to perform a task (e.g., *wash the safe in L12)* for which no existing method directly applies, ANA attempts to recognize an analogy with one of these methods. McDermott uses two mechanisms for "making contact" between the new task and an analogous method. First, he indexes each method with a "method description production" that will fire whenever similar types of actions or objects occur in a target task. Second, he adds a cue extraction process which re-expresses actions and objects in the target problem statement by generalizing or specializing them with respect to known type hierarchies. These manipulations insure that method indices are eventually triggered. For example, a method for *painting a table in L32 red* can be recognized as a source method for *washing a safe in L12* by generalizing and then specializing the target action (i.e., *wash I spray I paint)* and the target object (i.e., *safe/thing/ table).*

When a method index fires, it asserts "possible" elements which map into corresponding target problem elements. The target elements are given in the problem statement, detected by scanning the paint shop environment, or "stipulated" as expected elements. Thus, every element asserted by the method description production maps into a target element, either known or expected to exist. In the washing by analogy to painting example above, the initial analogical mapping includes:

paint >-» *wash* ,
table >^> *safe* ,

red i-> *clean-surface* .

These will be extended if the source method is selected for application.

During problem solving, ANA records method failures by building productions which restrict the offending method's preconditions. These productions fire whenever the same action is attempted under circumstances that would again lead to failure. As a result, the system is able to anticipate possible failures before selecting and applying an analogical source. Two kinds of problems may be anticipated: mutable violations (e.g., the position of an object in a stack prevents carrying it) and more serious immutable violations (e.g., the weight of an object is too great to carry it). Candidates with the least

serious precondition violations are selected, and can be further discriminated by preferring more specialized methods, methods with a high proportion of identity mappings to the target, and methods which alter the target problem least during cue extraction. Finally, ANA selects the single most promising source method.

Application of the selected method first copies "possible" source elements and marks these copies as "given" in WM. Marked copies enable some of the source method's productions, and ANA builds result-monitoring productions to watch for violations of their expected results. Finally, the enabled productions are applied. Each source action is mapped into a target action and executed in the environment. As method productions are applied, ANA must do two things: map resulting target elements into preconditions of source method productions and extend the mapping to consistently cover subgoals. A mapping production detects when elements added to WM satisfy "stipulated" target elements, confirming that the method is behaving as expected. When a subgoal is generated (e.g., moving an object in order to wash it), a different set of productions extends the mapping to consistently cover additional methods selected for achieving subgoals. These productions watch for identical objects shared by two methods and check with the user whenever new objects are added to the mapping.

ANA encounters difficulties when a method is either under or over-specified. An under-specified method is detected when an action fails in the paint shop environment or when a result-monitoring production fires. In the first case, the environment deposits an clement in WM that identifies the failed action and a troublesome attribute. In the second case, ANA asks the user for assistance in diagnosing and repairing the error. Detecting a mutable error (e.g., trying to carry an object that is underneath others in a stack), the system builds a production to anticipate this error during later applications and generates a subgoal to repair the violated condition (e.g., clear the top of the desired object). Detecting an immutable error (e.g., trying to carry a heavy safe), the system builds two productions: one to restrict the failed method by checking for a proper attribute value, and another to recommend an alternative method when the troublesome value is found (e.g., using a cart to move the safe). An over-specified method is detected when a goal element is satisfied by an unexpected value (e.g., an object is painted an appropriate color but ends up in an unexpected location). In these cases, ANA checks the unexpected value with the user. The solution is successful if the violation is unimportant, but the solution attempt continues if the violation is unacceptable.

ANA consolidates successful methods by building a production which fires whenever that goal context recurs. This production encodes the mapping from method to problem, and is added to the store of productions when the source method succeeds. Since subgoals can be embedded in a successful method, the system has some capacity for assembling subgoals and their expected results

into the conditions of an acquired production. Although not described in detail, this gives a limited form of chunking[9] where acquired productions can span several operations. It is not clear whether acquired chunks can be composed during subsequent learning. The ANA implementation solves paint shop tasks by analogy and achieves a three to six-fold speedup when solving these tasks a second time.

In summary, ANA demonstrates mechanisms for each idealized component of analogical reasoning. An analogy between the target problem and a source method is recognized by manipulating target cues until a method index (description production) is satisfied. McDcrmott cautions that as the knowledge base of method instantiations grows, recognition on the basis of object and action types might not discriminate between competing methods. An analogical mapping associates WM elements from the target problem and retrieved source. These associations allow the underlying production system architecture to tentatively accept "stipulated" target values, which are evaluated during method application. A selected method minimizes anticipated failures, and is evaluated by comparing prior expectations with feedback from a reactive environment (including the user). Error recovery strategies repair faulty solution approaches, when possible, and record productions which can avoid similar errors on later applications of the same method. Finally, successful applications of existing methods are consolidated as context-specific productions which support a solution without analogy. McDermott points out that ANA is an archetypical "hacker," building productions for everything from successful method application to restrictions on single preconditions. This simple learning mechanism proves effective on a small sample of tasks in a constrained experimental environment.

4.3. CarboncII [18, 20]: Reconstructive problem solving

Carboneil embeds analogical reasoning within a traditional problem solving framework that uses "weak methods" to search for solution paths or applies general plans to reduce the task into easier subproblems. Two distinct analogical methods are proposed. The first uses a second-order problem space to transform a source solution path into a target solution path, while the second reconstructs an entire problem solving attempt from a prior source solution. Both methods extend the traditional problem solving framework and provide a variety of opportunities for learning.

The *transformational analogy* method [18] transforms a solution path from a previous problem (the source) into a solution path for the target problem. A traditional means-ends-analysis (MEA) problem solver is extended into an "analogy transform problem space" (T-space, see Fig. 5). T-space is a second-order problem space in which states encapsulate a sequence of operator

[9] See Laird et al. [70] for a discussion of chunking in a general problem solving architeclure.

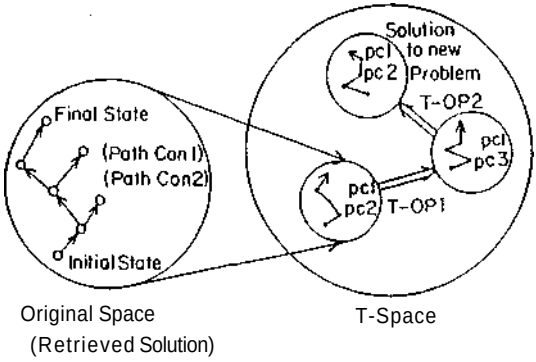

Fig. 5. Analogy as search in a transformation space (Carboncll [18, p. 143]).

invocations in the original space. Each T-space operator (T-operator) modifies a sequence to give a new solution sequence. The central idea is that analogical problem solving can occur in T-space just as routine problem solving occurs in the traditional problem space. Enabling conditions for T-operators are satisfied, T-operator application gives a new sequence, and a resulting sequence eventually satisfies the target goal specification.

The ARIES system is an initial implementation of the transformational method. Inputs include a first-order problem space (state representation and operators), a similarity metric used to rank candidate mappings between target and source problems, a second-order problem space including T-operators, and a collection of previous problem solving cases. The similarity metric is sensitive to the comparability of initial states, goal states, and path constraints (e.g., cost), as well as the overall applicability of a candidate source. T-operators are procedures for inserting, deleting, or reordering operators in an existing sequence, concatenating new subsequences of operators, or merging multiple operator sequences. The implementation contains a subset of T-operators organized in a difference table (see Newell and Simon [86]) by the sequence differences they reduce. Memory for prior solutions (sources) can be dynamically organized around MOP-like structures [99] that discriminate among previous solutions on the basis of similarities in problem states, goals, constraints, and operators.[1"] Generalized plans are learned for solving classes of problems related by having a common transformational source; successes and failures in problem solving arc used to refine the similarity metric as well as difference table entries for original and T-space operators; and new T-space operators can be discovered by using conceptual clustering [31, 78] methods over problem solving instances where T-space MEA fails.

"'See Kolodner [67, 68| for a detailed implementation using MOP .structures, Simpson [103] as discussed later in this section, and Dyer [26] as discussed in Section 5 on natural language processing.

Given a target problem described in the original state space (initial state, goal state, and constraints), ARIES selects a source solution with similar state descriptions and constraints. For example, the system proves that *the product of two odd numbers is tin odd number* by analogy to a source proof that *the product of two even numbers is an even number.* The sequence of steps in the source proof is an initial state in T-space, and a sequence which proves the target theorem is a goal state. The analogy is elaborated in T-spacc by a means-ends search for a sequence of transformations that yield a target solution sequence. For example, in the odd product proof several additional algebraic operations must be spliced into the source solution sequence.

To summarize Carbonell's transformational method, a second-order problem space (T-space) is integrated with a traditional problem solving framework (MEA), and a variety of possible learning opportunities are discussed. Recognition occurs on the basis of a partial mapping between the target problem statement and source episodes indexed in a hierarchically organized memory. The most promising source solution is incrementally transformed into a sequence of operators that gives a correct target solution. Beyond learning generalized solution sequences, the method allows the possibility of tuning various components of the analogy framework. Carbonell reports that an initial ARIES implementation was "effective when tested in various domains, including algebra problems and route-planning tasks" [20, p. 376], but was unable to take advantage of more abstract planning information that might be stored with a solution sequence (e.g., the reasons for problem solving decisions or failures).

The *derivational analogy* method [19, 20] extends the transformational approach to include complete derivational traces for previously solved problems (see Fig. 6). The derivation includes subgoals, alternative operator choices at each step along with reasons for decisions among them, beginning and terminal nodes for unsuccessful paths with reasons for failure, interdependencies between successive decisions, pointers to peripheral knowledge structures used during problem solving, and reasons and/or assumptions supporting successful or unsuccessful problem solving attempts. Carbonell argues that this information is necessary to successfully transform a source solution into a solution for a novel problem. Particularly in complex domains like design (e.g., coding quicksort in LISP, having previously coded quicksort in PASCAL), useful analogical transfer must occur at an abstract level and include a reconstruction of decisions made when solving the source problem.

Given a target problem, reconstructing a solution from a source derivation is somewhat complex. As with the transformational method, derivational analogy is cast against a background of weaker problem solving methods. If direct plan instantiation is not possible, the system uses weak methods (e.g., MEA) to search for a solution in the target problem space. When initial search-based problem solving steps resemble the initial segments of a derivational trace in

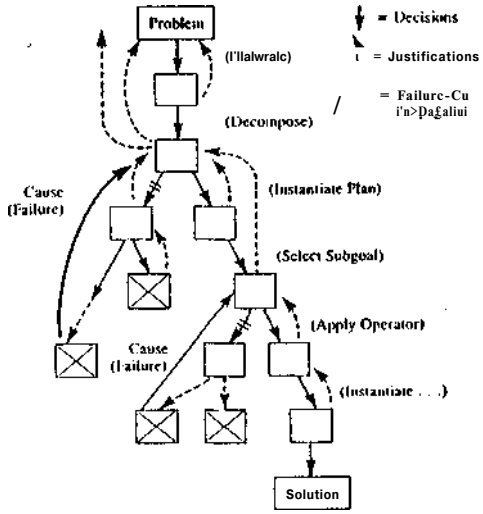

Fig. 6. Derivational truce for *a* solved problem (Carbonell [20, p. 380]).

dynamic memory, the system retrieves and attempts to reconstruct the derivational trace as follows:

Step 1. For each step in the trace, check its justification in the target problem description. If the justification holds, the step is applied directly.

Step 2. If the source justification is violated but an alternative target justification is found, the step is applied and a record of this change to the retrieved derivation is stored for later learning.

Step 3. If attempts to justify a derivation step fail:

(a) attempt to justify alternative steps stored at this level in the derivation,
(b) post the unsatisfied justification as a subgoal for further problem solving,
(c) try failed derivation paths if the reasons for those failures do not hold in the target problem,
(d) give up the current analogical derivation in favor of another source derivation or return to weak methods.

Minor deviations from derivational traces are not sufficient to abandon the derivation, since subsequent derivational decisions and steps may be independent and still apply to the target problem. A "perseverance threshold" interrupts fruitless derivational efforts. Carbonell argues that despite the apparent complexity of the derivational trace, the processes sketched above can be implemented efficiently because space requirements arc proportional to trace depth rather than to the number of nodes visited while solving the source. In addition, few dependency links point outside the local structure of the derivational trace.

Opportunities for learning are again varied. First, problem solving experience results in a richer database of case derivations. Second, stripped-down solution traces for successful and unsuccessful problem solving efforts can be passed to a general-purpose induction engine" to form generalized plans, a process similar to the primary learning mechanism described in the transformational method. Third, individual decision points in derivational reconstruction can be used as positive and negative instances for learning. Justifications for these decision points can focus the learning process on functionally relevant aspects of the derivational trace, resulting in domain-specific search heuristics. Finally, a decomposition process can extract frequently occurring subgoal subsequences out of derivational traces. Subgoal justifications are examined to find their context of occurrence (e.g., the instantiated goals they satisfy), and a rule is formed which proposes the subgoal sequence whenever that goal context recurs. Since justifications refer to necessary preconditions among a variety of contextual details, rules formed by decomposition provide more general plan components.

The derivational method augments transformational analogy in two respects. First, the materials of an analogical source are extended considerably. Justification linkages guide the reconstruction process by allowing the analogy mechanism to evaluate not only the actual solution path (i.e., a series of operators), but also a record of why particular steps were taken and succeeded or failed. This provides an advantage over the relatively less informed method of difference-driven transformations. These justifications also provide explanatory focus to a variety of learning mechanisms. Second, the derivational approach integrates recognition of analogies directly into an ongoing problem solving context. Initial problem solving activities may trigger retrieval and reconstruction of a source derivation, or the problem solver may continue with a weaker, search-based approach. In summary, Carbonell's description of reconstructive problem solving charts an ambitious program for computational studies of analogy, opening up a variety of challenging research problems.

4.4. Simpson [69, 103]: Case-based dispute mediation

Simpson describes MEDIATOR, a system that suggests resolutions to physical^ economic, or political disputes by using analogies to past cases. Analogous cases are used to classify a target dispute, derive plans for its resolution, predict plan consequences, and recover from planning failures. This approach combines aspects of Kolodner's [67, 68] model of maintaining a long-term episodic memory with mechanisms for analogical problem solving. Given a memory of prior dispute cases, the system is tested on novel disputes. For example, MEDIATOR suggests a plan for resolving the Sinai dispute between

"See Michalski'[76], Milchell [82], or Quinliin (95) for domain-independent induction mechanisms.

Israel and Egypt by analogy to source cases of land dispute in the Korean War, children quarreling over possession of an orange, and land dispute over the Panama Canal.

A target problem is solved by classifying it in a taxonomy of problem types, constructing a solution plan, confirming plan predictions, and remediating plan failures if necessary (see Fig. 7). Planning failures are resolved by a recursive application of the problem solver.[12] Classification, planning and recovery are attempted by analogy to prior cases, but can be achieved by default reasoning (e.g., assembling and instantiating general plan fragments) if necessary. The system learns by updating its episodic memory for successful and failed solution attempts.

The basic unit in MEDIATOR'S case memory is the *generalized episode,* which can either be a specific dispute case or a summary of multiple cases. The episode is a frame structure containing slots for normative features, a set of indices to more specific episodes, and a pointer to a specific precedent case. Episodes in long-term memory are organized as a discrimination network around indices which test specific feature values. These restrictive tests allow "locked" [67] traversal of the network when attempting to classify a new episode. MEDIATOR starts with an episodic memory for several dispute resolutions rooted by a hierarchy of generic episode types. These include problem

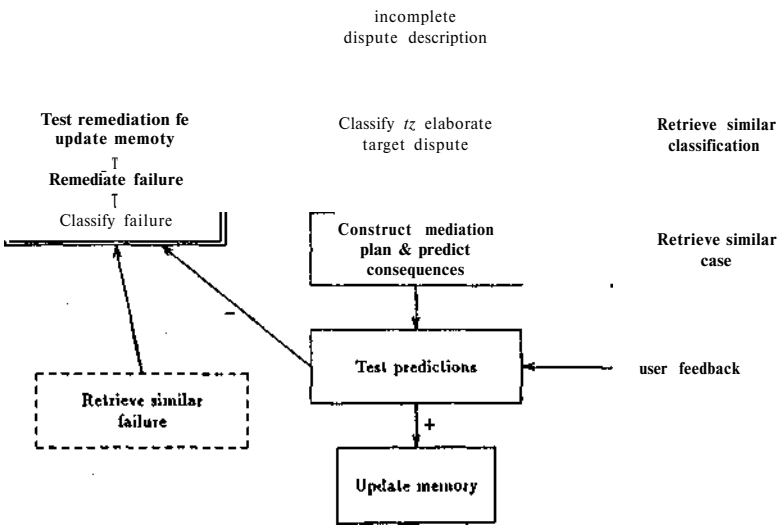

Fig. 7. Retrieved cases assist dispute mediation planning (adapted from Simpson [103, p. 17]).

[12]See Hammond [47] for an approach that uses failures and their explanations for memory organization and retrieval.

classes (e.g., physical or economic disputes) and their mediation tactics (e.g., divide the disputed object equally), as well as failure types (e.g., goal misunderstanding) and their remediation tactics (e.g., infer an alternative goal). Taxonomic hierarchies for disputed objects and disputants are also given as input. These provide the system with a "blueprint" to bias case comparison and memory update. As output, the system recommends dispute resolutions and updates its episodic memory for successes or failures.

An analogous case is recognized by traversing episodic memory with target components as retrieval cues. In the Sinai dispute, for example, MEDIATOR recalls the Korea dispute because both involve land and use military force to achieve disputants' goals. Starting with the most general component structures in episodic memory, target cues (e.g., Israel, Egypt, and the Sinai) are compared with index tests to guide traversal to an episodic structure that contains a specific case (e.g., the Korea dispute). Colliding with this episodic structure, the system is spontaneously "reminded" of the source case. MEDIATOR uses multiple target components to retrieve a set of candidate case remindings. These are ranked according to how well they preserve a "relative invariance hierarchy" [16, 17] between target and source. Candidates that violate target goal descriptions (e.g., concordant versus competitive) are eliminated, and remaining candidates are ranked by the taxonomic similarity of argument style (e.g., persuasive versus adversarial), disputed object, and disputant. The top-ranking source case is selected and, depending upon the current stage in problem solving, can be used to classify the target problem, suggest a dispute resolution plan, classify the target failure, or suggest a recovery.

Elaboration of the analogy between a retrieved case and the target problem is distributed across a traversal path in episodic memory. Successful index tests during traversal establish points of correspondence and successively constrain calculation of the analogical mapping. Arriving at an analogous case in memory (a reminding), the system aligns information recorded in the source norms with its current representation of the target problem. Episode slots (e.g., the disputants in each case) map identically, and ambiguities in mapping slot values are resolved by peripheral constraints in the source frame (e.g., that the older of two disputants divides a disputed object). Depending on the problem solving context, elaboration with the source case can map problem classification, plan steps, failure classification, or remediation steps.

MEDIATOR evaluates information provided by an analogous case at two levels. First, frame components which will be directly transferred are tested for consistency with the target problem description. For example, the "divide equally" plan suggested by analogy to the Korea case has several preconditions (e.g., the disputed object is spliltable) which are verified in the Sinai dispute before the plan is transferred. Analogical inference of dispute classification, plan consequences, or failure recovery tactics are evaluated in a similar

fashion. If this level of evaluation fails, the system falls back on default reasoning. Second, predicted consequences of a transferred plan are evaluated by soliciting results from the user. If user feedback differs from plan expectations, MEDIATOR marks the attempted plan as a failure, collects explanatory remarks from the user (e.g., Israel and Egypt have unexpected goals), and recurses with a new problem of resolving the planning failure. Since prior failures, their classification, and resolution are also stored in episodic memory, the system can resolve the current failure by analogy to past cases.

Just as remindings occur when traversing episodic memory, the traversal process also identifies locations in the memory structure where MEDIATOR can consolidate problem solving successes and failures. Since reminding is the result of two specific cases colliding in memory, update can be accomplished by making local changes to the contexts and indices of surrounding episodic structures. Simpson describes five conditions under which memory update occurs, each depending on the taxonomic relation of target and source when traversal terminates:

(1) If the source is more specific than the target, *insert* the target case as a parent.

(2) If the source is more general than the target, *install* the target case as a specialization.

(3) If source and target are instances of the same concept, *generalize* non-identical slot values, *insert* a generalized episode in place of the source, and *install* target and source as specializations.

(4) If the source and target are siblings in a taxonomic hierarchy, find their "most specific common parent," *insert* the parent with generalized norms, and *install* each sibling as a specialization of the parent.

(5) If source and target are unrelated in any taxonomy, then report an update failure.

Simpson uses taxonomic hierarchies over objects, goals, and actions to find generalized descriptors for the general episode's norms, and uses source/target differences as specializing indices. Since MEDIATOR resolves problem solving failures in much the same fashion as routine problem solving efforts, these can be consolidated into the episodic memory as well. Simpson reports that the system is capable of managing nine dispute cases in its episodic long-term memory before exceeding machine limitations. Although adding cases extends the problem solving capacity of MEDIATOR, there is no explicit evaluation of whether the system's performance improves in accuracy or efficiency.

In summary, Simpson explores each process component of analogy. Recognition and consolidation occur while traversing a dynamic memory for problem solving episodes. Promising source cases are selected to the extent that they follow an invariance hierarchy for analogical transfer. An analogical mapping between target and source cases is elaborated incrementally across a retrieval

path, so that much of the mapping appears to be established before any specific source case is selected. Having found a source case, a frame-level mapping is constrained by enforcing identical slot alignment. Information stored in the norms of a source case is evaluated for consistency with the target problem before transfer, and serious violations cause the system to fall back on default problem solving inferences. When failure of transferred material is signaled by external feedback, the system recursively applies itself to the failure as a subproblem. When problems are solved (or failures resolved), consolidation processes install, insert, or generalize memory episodes at appropriate memory locations. As a result, MEDIATOR acquires specific, analogically derived solutions to target problems as well as generalized episodic structures.

5. Metaphor and Analogy in Natural Language Processing

Given the bulk of related research in linguistics, philosophy and psychology, it is not surprising that attention has been given to metaphor and analogy in computational studies of language understanding. In this section, three studies are reviewed: Winston [108] describes a system that comprehends similes in an instructional context; Hobbs [51, 52] extends a model of selective infcrencing to resolve metaphorical expressions during routine discourse analysis; and Dyer [26, 27] describes systems that use analogies to previous cases of planning failure to understand narrative text. Each study presents specific mechanisms for analogical comparison and inference, but each also uses the surrounding context to constrain these mechanisms.

5.1. Winston [108]: Concept learning by creatifying transfer frames

Winston focuses on learning through "absorb(ing) simile-like instruction" [108, p. 151]. Although he examines a highly constrained form of learning, the approach is equally interesting as a computational model of making appropriate inferences when presented with simile-like statements (e.g., Robbie is like a fox). In overview, Winston's model constructs "transfer frames" that allow the projection of properties (slot/value pairs) from a source frame (e.g., fox) into a less well-understood target frame (e.g., Robbie). Central to this transfer process are constraints provided by the *salience* of information in the source frame and its conceptual siblings, the *prototypicality* of information in the target frame and its siblings, and the *instructional context* given by a tutor. A simple frame language describes objects and their properties in microworlds of blocks and animals.

Winston describes an implementation with four stages of processing:

Step 1. *Hypothesize* which of many source properties should transfer to the target, based on their salience in the source frame.

Step 2. Filter candidate "transfer frames" to prefer properties which are

consistent with prototypic knowledge of the target or the ongoing instructional context (i.e., properties involved in recent similes).

Step 3. *Justify* a selected transfer frame by finding supporting similarities between source and target (e.g., properties supporting the purpose of objects) or by appealing to a tutor.

Step 4. *Conjecture* further transferred material through the learner's "curiosity" or encouragement from the tutor.

Hypothesizing and filtering transfer frames correspond to elaboration, justification of these frames to evaluation, and conjecturing further transfer to some aspects of recognition and consolidation. A collection of strategies is described for each stage.

To hypothesize transferable properties, Winston first considers existing transfer frames constructed earlier when reasoning from the same source object. An existing transfer frame can be used again if the justification (stored as distinguished properties in another frame) for its previous transfer is true of the current target frame (i.e., similar distinguished properties exist). When unable to identify existing transfer and justification frames, the system selects salient source properties (slots and/or values). Salient source properties have extreme values, are globally important (i.e., a property is marked as important in an absolute sense), are atypical in comparison with properties stored with conceptual siblings of the source frame, or have atypical values in comparison with those siblings. If existing transfer frames or salient properties cannot be identified, all properties of the source are considered. Candidate properties for transfer are grouped by property category (e.g., size, weight, height, width and depth are all instances of the size-property category) and considered independently by the filtering process.

Filtering strategies use prototypical aspects of the target frame to select which properties are transferred (e.g., which transfer frame to select). Prototypical information is statistically determined for a class of related frames (e.g., a typical cylinder has a color slot which takes on varied values in over 65% of cylinder instances) and stored in "typical-instance frames." Transfer frames arc preferred if they promote slot/value pairs that occur in typical-instance frames which are conceptually related to the current target frame. When prototypical expectations are not available, transfer frames which promote properties contained by any conceptual siblings of the existing target frame are preferred. Finally, in the absence of any knowledge of what is typical of the target frame, transfer frames which continue the instructional context (i.e., the type of properties recently transferred on the basis of instruction) are preferred.

Justification strategies are somewhat less ambitious. The learner either asks the tutor about the appropriateness of a candidate transfer frame or checks for violations of known restrictions on transfer frame slots. As a third strategy, the

learner consults a record of shared properties which were important in previous cases of successful transfer. For example, the tutor might present the simile, "CUBE-I is like TABLE." The property of "purpose" being a platform for eating or writing is transferred from table to cube, and the tutor suggests creating a "justification frame" which records common characteristics of the cube and table that enable the transferred purpose. In this case, the properties medium size and flat, level top justify the purpose of being a platform for eating or writing. Later tutor-initiated transfer involving TABLE can use this justification frame to confirm that a new target object's properties support the transfer of information regarding purpose.

Winston briefly describes strategies for making independent conjectures. The tutor sometimes directs the system to extend recently acquired information in some target frame. Three mechanisms are proposed for this exploration. Since the system records frames for slot types (e.g., TOP-FLATNESS) and notes extreme values encountered while absorbing instructor-generated similes (e.g., VERY-HIGH for flatness when TABLE is used as a source), recurring extreme values can trigger system-generated similes (e.g., a new object with VERY-HIGH TOP-FLATNESS is like a TABLE). Alternately, when adding a new slot/value pair to a current target frame, the system can look for a conceptual sibling that has a similar property, and then generate a new simile with that sibling. Finally, the learner can inherit properties for a target frame when a typical-instance frame is available.

In summary, Winston explores each process component of analogical reasoning. Similes between objects are presented by a tutor or recognized when the system detects extreme or shared values for two objects. These objects are conjectured as the source and target of a new simile. The properties of hypothesized transfer frames are elaborated by preferring existing transfer frames, salient properties of the source, properties that are consistent with prototypical knowledge of the target, or properties that continue the instructional context. Transferred properties are evaluated by checking for violations of known target properties, finding support from existing justification frames, or asking the tutor. Similes are consolidated by recording successfully transferred properties and then building transfer, justification, and typical-instance frames which can be used in subsequent learning. As a study of metaphor comprehension, Winston's model explicitly uses salient characteristics of the source domain and prototypical qualities of the target domain. These strategies parallel psychological studies of metaphor comprehension as a feature comparison process (e.g., see Malgady and Johnson [73]).

5.2. **Hobbs** [51, 52]: **Metaphor as selective inferencing**

Hobbs argues that textual metaphors like "Mary is graceful, but John is an elephant" can be understood by selective inferencing during routine discourse analysis. Assuming the hearer knows that elephants are large, have trunks,

have excellent memories, are thick-skinned, and are clumsy, the problem of metaphor comprehension is to select an appropriate inference about John from among the many possible inferences. Metaphorical inference is appropriate when it recovers the intended meaning of the speaker. Hobbs describes a natural language understanding system, DIANA, that selects appropriate metaphorical inferences while resolving traditional discourse problems like recognizing coherent relations among text concepts, finding specific interpretations of predicates, or determining implicit relations between compound nouns. As input, the system takes predicate calculus formulas generated by a syntactic analysis of text. A knowledge base of axioms describing commonsense and linguistic knowledge is also given, including detailed schematic representations of source domains. As output, the system delivers a coherent representation of the target text with appropriate metaphorical inferences.

DIANA interprets metaphors like the one shown above, but Hobbs [52] devotes his attention to a more complicated metaphorical complaint by a democratic member of the Congress about President Gerald Ford's vetoes of congressional bills:

> We insist on serving up these veto pitches that come over the plate the size of a pumpkin. [52, p. 126]

Although the system does not interpret this example, Hobbs develops its representation and processing in detail. As a source for the metaphor, Hobbs gives a description of baseball using universally quantified variables and a notation for "conditions" which identify instances of predicates:

pitcher'(c.x,χ),
ball'(cy, y),
batter'(cz, ζ),
p'Hch'(p,x, y,z),
miss'(«i, z, y),
hit'(/i,z, y),
or'(omh, in, /?,),
then'(//z, p, omh) .'

Variables *x, y,* and ζ are a pitcher, ball, and batter, respectively. The condition, /;, in the pitch' predicate represents an instance of a pitcher pitching a ball to a batter. Conditions *in* and Λ represent disjoint possibilities (i.e., the batter missing or hitting the ball) after the pitch occurs. These axioms, sharing predicates and variables, gives a baseball schema that must be recognized as the source of the metaphor, instantiated by a mapping into the target (i.e., congressional bills and vetoes), and then used to generate appropriate metaphorical inferences.

The appearance of predicates in the target input triggers recognition of the source schema (e.g., the word "pitch" invokes knowledge of baseball). A

mapping from the congressional domain to the baseball domain is incrementally elaborated as discourse problems are resolved. For example, since the target sentence is uttered by a Congressman, the pronoun "we" resolves with the concept of Congress, allowing the role of pitcher in the "pitch" action to be mapped to Congress. Interpretation of the compound nominal "veto pitch" requires finding a coherent relational constraint over the contained concepts. Assuming a hearer knows that Congress must "send" a bill to the President for signature or veto, Hobbs argues that the system should detect the parallelism between pitching a ball to a batter and sending a bill to the President. This extends the mapping by associating pitching/sending, ball/bill, and batter/President. The compound nominal is resolved by considering how the batter's actions (i.e., to miss or hit the ball) can be mapped into the President's choices (i.e., to sign or veto the bill). According to Hobbs, the remainder of the target utterance (". . . that come over the plate the size of a pumpkin") must be resolved so that the President/batter would find the bill/ball easy to veto/hit. He does not discuss how "easy to hit" is recognized or transferred.

Evaluation of the metaphor's interpretation, beyond achieving contextual coherence and satisfying pragmatic constraints, is not discussed. However, Hobbs argues that the hearer should be able to suppress irrelevant or inappropriate inferences (e.g., that a bill is a thing or is a spherical object with stitching) but remain open to the powerful role which more neutral inferences, neither contextually intended nor inappropriate, can play in augmenting knowledge of the target domain. For example, knowledge that baseball is an adversarial game can be metaphorically extended to interactions between Congress and the President. Having interpreted and possibly extended a speaker's metaphor, the hearer consolidates metaphorically inferred knowledge in several stages, described by Hobbs as the "life story" of metaphor as a linguistic phenomena. Originally, the metaphor must be explicitly interpreted, mapping source and target concepts with considerable effort. Later, the familiar metaphor is comprehended quickly since appropriate inferences are expected. The metaphor becomes "tired" when earlier metaphorical inferences are directly available as concepts in the target domain, allowing comprehension without explicit metaphorical inference. Finally, inferences are so well integrated into the target domain that the metaphor is no longer recognized (i.e., it becomes "dead" [12]).

Hobb's account of metaphor comprehension is clear but open to several computational problems. Recognizing source axioms solely on the basis of concepts appearing in the input (e.g., "pitch") might not constrain candidate interpretations, particularly if the hearer's knowledge of the source domain is extensive. The recognition mechanism must choose among multiple senses available for a single term (e.g., a sales pitch or a tar-like substance) and filter inappropriate sources activated by nonsalient concepts (e.g., "serving up" a "pumpkin" over a "plate" as a holiday meal). Once an appropriate source

domain is recognized, context or discourse constraints must suppress a variety of extraneous metaphorical inferences. For example, inferences from the baseball domain involving ticket sales, umpires, or hotdogs are not distinguished from the more appropriate inference involving adversarial games. Contextual relevance, the most likely evaluative filter, is not detailed. Finally, Ilobbs sketches the consolidation of novel metaphors without describing how a knowledge base of axioms could be updated so that metaphors eventually become frozen.

5.3. Dyer [26, 27]: Adages and understanding planning failures

Dyer describes an implemented system, BORRIS [26], and its successor, MORRIS [27], as cognitive models of in-depth understanding of short but intricate narrative stories. Since these stories often involve expectation failures when characters use faulty plans, Dyer introduces "thematic abstraction units" (TADS) to organize narratives involving similar planning failures. He hypothesizes that this memory organization accounts for a class of cross-contextual "remindings" in which people spontaneously remember an adage summarizing a planning failure in the target narrative (e.g., being caught "red-handed" or remarking that "every cloud has a silver lining"), lie also describes experiments in which human subjects are given narratives designed to reflect a particular TAU, and are later able to generate superficially dissimilar stories which reflect the same planning failure. When these stones are sorted by a second group of subjects, categories preserve abstract planning similarities despite wide variations in story content. Although Dyer's primary intent is not to study metaphor, storage and retrieval of narratives based on abstract planning information gives a plausible account for some forms of metaphorical reasoning. Sharing plan information between a target narrative and a previously encountered story is largely consistent with Centner's mapping of structural relations [34] or Carbonell's invariant transfer of semantic categories [16, 17]. For example, August and Dyer [6] describe preliminary work on understanding analogies in editorial arguments. Analogies are recognized using textual clues (e.g., "is similar to") or similarity of textually contiguous conceptual structures. Causal structures for target and source are mapped and guide inferences during a question answering task.

Dyer's focus on memory organization and retrieval of analogous narratives implements one approach to Schank's description of remindings within a dynamically organized memory [99]. Schank proposes "thematic organization packets" (TOPS) which organize memory around domain-independent goal interactions. For example, a memory for choosing a compromise restaurant when two parties prefer different foods might be associated with a "competing goal; compromise solution" TOP. Later, while scheduling an appointment with someone who can only meet at an inconvenient lime, one might be reminded

of the restaurant compromise. Dyer's TAUs organize memory around less abstract planning difficulties. In an anecdotal example, Dyer [27] describes watching a friend grow increasingly upset about the possibility that automobile repairs had not been completed. Dyer told the friend to "quit bleeding before he'd even been cut" and was spontaneously reminded of a story in which a distressed and apprehensive motorist seeks help at a farmhouse along the roadside. Before even explaining his situation, the motorist angrily refuses help from the startled farmer. According to Dyer, both events are instances of a "TAU-EMOT-ANTICII'ATE" where anticipation of a negative situation disrupts effective planning.

Generalizing from the task of understanding complex narratives, Dyer's work can be described as analogical reasoning between experienced events. Given a memory organized around planning failures, recognition uses planning mistakes in a target narrative as indices into an appropriate TAD. The TAU is a generalized index for memories involving similar planning errors. Planning information stored with the TAU and the specific memories it organizes can be mapped into the target narrative to generate a coherent interpretation of the text and to make predictions about what might happen next. Differences between a retrieved source and the target narrative prompt a reorganization of existing memory structures so that the target can be discriminated during subsequent retrieval. Similar forms of recognition and consolidation in long-term episodic memory are discussed by Lebowitz *[72],* Kolodner [67, 68], and Simpson [103] (as reviewed in Section 4.4).

6. Analogical Reasoning in Machine Learning

Many of the reviewed studies include some form of learning, but concentrate on other tasks for which analogies may be useful or necessary. For example, Carbonell's methods for transformational and derivational analogy [18, 20] include learning, but they also provide an exemplary view of analogical reasoning as a problem solving process. In this section, four studies of learning by analogy are discussed. Winston [109, 110, 114] describes mechanisms for learning principles from precedents and exercises; Burstein [13-15] presents a cognitive model of integrating multiple analogies drawn from different levels of abstraction while learning simple concepts in a programming language; Pirolli et al. [92, 93] describe a cognitive model of learning to write recursive LISP programs; and Kedar-Cabelli [61, 62] uses purpose-directed analogies to focus attention during a concept learning task.

6.1. Winston [109, 110, 114]: Learning and reasoning from analogous cases

Winston describes an implemented model of analogical reasoning in which constraints from a source situation (a precedent) are transferred into a target situation (an exercise). Shared constraints support inferences about the target,

and if these inferences are confirmed, the model forms rules which generalize over source and target situations. The implementation has been applied in several domains, including short narratives, medical cases, and physical systems (e.g., electrical circuits). For example, Winston [110] starts with plot summaries of Shakespearean plays, gives partial information about a target case and asks the system to answer questions about the target (e.g., show that a man is weak). A source plot summary is recognized (e.g., *Macbeth)* and used by analogy to infer an answer to the target question. The system delivers this solution, induces a general rule over target and source cases, and then stores the rule for later use.

Source cases arc represented as object-oriented (rather than event-oriented) prepositional networks, augmented with supplementary constraint descriptions (e.g., causality). Figure 8 shows network representation for the source (precedent), target (exercise), and rule described in the preceding example. The target problem (? MAN-i EVIL) can be answered by importing relations from the source. In this case, causal relations enabled by MAN-1 being weak and being married to a greedy spouse support the analogical inference that MAN-l is also evil. Inputs include structured natural language descriptions of situations which the system translates into the network representation, target questions about relations among objects, answers to system-generated questions concerning values or relations in the situation, and instructions to consider alternative source cases or to form rules. Outputs include justified answers to questions

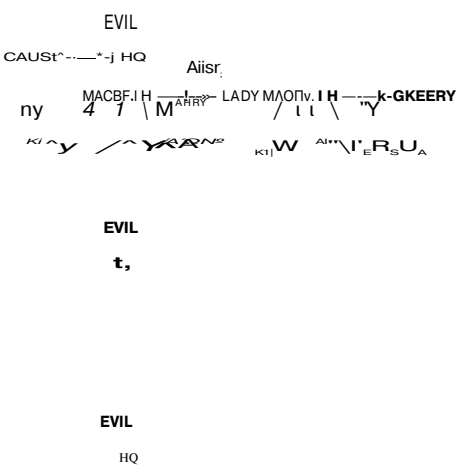

Fig. 8. A precedent, exercise and resulting rule (Winston [110, p. 331]).

posed by a tutor, augmented network representations for target situations, and general rules.

A process of "classification-exploiting hypothesizing" [109] recognizes analogous source situations. This is essentially a bottom-up retrieval scheme that exploits exhaustive indexing of candidate sources through a hierarchy of object types. "Votes" for a source are collected by traversing a-kind-of (AKO) links originating at instance nodes in the target representation. Each node in the type hierarchy has an APPEARS-IN slot that records the situations in which that node and its descendants participate. Each type node visited during this AKO traversal casts votes for sources appearing in its APPEARS-IN slot. A vote counts by inverse proportion to the number of candidate sources in the slot but in direct proportion to the salience of the target concept. After a complete AKO traversal, the source situation with the highest number of votes is retrieved.

Although pathways in the AKO traversal implicitly contain mapping information about the target, Winston uses a separate process for elaboration. With an exhaustive search through the space of possible network mappings, he reports [109] that his system is confined to problems of 100 or fewer possible pairings. Instead, he carefully chooses initial points of correspondence, preferring a priori important relations (e.g., cause) that can be compatibly aligned. Thus the space of all possible mappings between source and target is expanded using a heuristic search for plausible mappings (see also [113]). Winston [109] scores each of these with a summative metric that rewards similarity in the correspondence of object properties, relations among objects, and constraint relations. A variety of similarity metrics are empirically tested with plot summaries, and results suggest that scoring only constraint relations known to be important can be as effective as more exhaustive scoring schemes.

After finding a best analogical mapping between target and source, the system proceeds to answer a question (i.e., an unknown but desired relation) about the target:

Step 1. If the relation is available in the target, simply use it to answer the question.

Step 2. If the relation is missing in the target but caused by other relations in the source, try to establish these causal antecedents in the target situation. This generates additional questions as subgoals to be answered by direct inference or another analogy.

Step 3. If the relation is missing in the target and has no known cause in the source, look for another analogous source.

Winston [110] extends this strategy to allow abductive analogical inference: a desired relation in the target may be inferred if its known consequences, as suggested by analogy with the source, are true. Thus the desired relation is "covered" by its consequences. When the system reaches an impasse at Step 3

(e.g., no analogous source can be found), the tutor can directly confirm or disconfirm desired relations.

In general, questions about unknown target relations can be answered by analogically inferring complex causal chains imported from one or more source situations. In the network representation, these causal chains are expressed as AND trees, with the desired relation at the root and sufficient facts at the leaves. Target and source AND trees are passed to an inductive process that extracts common implicational structure from the trees. The result is also an AND tree where leaves make up the rule conditions, and the root node is the action or consequence. Generalizations are obtained by dropping intermediate nodes in the implicational network [110, 114]. Just as with a source case, a rule can be used by analogy to promote inferences in another problem solving task. The rule learned in the (? MAN-i EVIL) problem can be expressed as follows:

Rule RULE-l
if [MAN-4 HAS-OUALITY WEAK]
[MAN-4 MARRY WOMAN-2]
[WOMAN-2 HAS-QUALITY GREEDY]
then [MAN-4 HAS-OUALITY EVIL]
case MACBETH

The source of this rule, MACBETH, is retained so that a case comparison is possible if the rule fails during use in another context. New rules can be learned from an analogy between situations as well as when applying an old rule in a new situation.

In his example, marriage between MAN-4 and WOMAN-2 enables the wife to influence her husband. However, if a new case suggests that the wife cannot influence her husband (e.g., they are estranged), then RULE-l would apply inappropriately. Winston [111, 114] resolves the problem of over-general learned rules by including "unless" conditions which block the rule when exceptional conditions are known. Augmented rules take the form:

if [preconditions . . .] **then** [conclusion] **unless** [exceptions . . .].

Exceptions are simply the negations of relations between the leaves (conditions) and root (conclusion) in the AND trees from which rules are synthesized. Thus, exceptions are restricted to relations of known causal relevance. When an augmented rule is considered for use, a limited amount of effort is put into showing that no unless conditions hold. If an exception holds, the rule is blocked because its supporting causal structure is violated. Rules which trigger exception conditions are called "censors," have the same form as other augmented rules, and can be learned accordingly. To facilitate later retrieval,

rules are indexed through the classes of actors, relations, and objects found in their consequences. In this order (i.e., actors before relations), indices are stored in multi-level association lists residing in frame structures that describe each class. Given a new target question, components of the question are used as retrieval cues to index into sets of candidate rules. These candidates must further match the target situation before being used to find an answer.

Winston's model of reasoning and learning by analogy covers each of the process components used in this survey. Candidate sources are recognized through a bottom-up voting mechanism which uses exhaustive type indexing. Although Winston does not set out to recognize novel analogies across domains, it is interesting to consider whether his recognition scheme could "scale up" to this task. For example, a psychodynamic analogy between current in electrical circuits (another of Winston's test domains) and depression in Hamlet's psychological state might not be easily recognized, since the only common ancestor of current and depression might be THING in the ΛK.O hierarchy. Analogical mappings are elaborated by a heuristic search process that prefers a priori important relations in source and target situations. A static scoring function ranks the resulting mappings, and a best candidate is selected. Analogical inferences are treated as target subgoals and can be confirmed by direct inference, further analogies, abductive inference, or questioning the tutor. Rules with exception conditions are constructed and stored under indices reflecting actors, relations, and objects. Acquired rules both summarize analogous cases and provide a mechanism for falling back on source cases when a rule fails. Winston does not describe this recovery mechanism in detail.

6.2. Burstcin [13-15]: Learning assignment statements in BASIC

Programming concepts of *variable* and *assignment* are often taught by analogy to more familiar domains like placing physical objects in boxes. Burstein [14] presents a computational model of using didactic analogies to learn about simple assignment statements in BASIC. The model is based on verbal tutorial protocols from students learning to program and is intended as a plausible psychological account of learning. An implemented system, CARL, engages a tutor (Burstein) in a tutorial dialogue about BASIC assignment statements. Knowledge of source and target concepts is represented by interconnected networks of frame structures, organized in a fashion similar to Schank's [99] description of dynamic memory. Central in Burstein's model is a "top-down" analogical mapping of causal abstractions from a well understood source domain into a poorly understood target domain. This is contrasted with other's approaches (e.g., Evans [28] and Winston [109]) which attempt to find an object-level mapping that supports consistent relational structure. Burstein argues that these mapping strategies cannot be sufficient when little is known

of the target domain or when irrelevant constraint relations (e.g., cause in Winston's MACBETH) are commonplace.

The tutor gives CARL a didactic analogy, an example of the target concept to be learned, and a target problem:

> The computer remembers a number by putting it in a variable.
> A variable is like a box.
> To put the number 5 in the variable named *X,* type *"X=* 5."
> Mow do you get the computer to remember the number 9? [14, p. 356]

With support from the tutor, the system incrementally maps causal abstractions for source domain actions (e.g., preconditions and results of putting an object in a box) into the target domain of BASIC assignment statements. As output, CARL answers the tutor's questions and learns how to interpret and use common assignment statements.

Burstein proposes several constraints on a top-down analogy mapping process. First, he uses the tutorial context to recognize a particular relational structure from the source domain. The target problem statement (" . . . get the computer to remember the number 9"), coupled with the explicit analogical hint ("A variable is like a box"), allows CARL to recognize a useful causal abstraction for boxes: a human actor puts a physical object in a box. Second, Burstein maps relations in this abstraction before considering object-level correspondence. Target and source objects arc mapped only when they play similar relational roles. He points out that the relational correspondence need not remain at a literal level. For example, since numbers are usually not physical objects which can be placed within boxes, a new sense of the "INSIDE" relation must be found. CARL finds the new relation (e.g., INSIDE-VAR) by selecting an ancestor or sibling in a relation type hierarchy. This differs from others' accounts of relational invariance [16, 34] by allowing non-identical relational mappings. Third, Burstein discards source relations which have no known support in the target domain. For example, a precondition for putting an object in a box is that the object be smaller than the box. Since CARL knows of no comparable scale for fitting numbers in variables, this precondition is dropped during transfer.

In protocols collected while tutoring novice BASIC programmers, Burstein finds that multiple analogies are regularly used to acquire programming concepts. The following analogies often occur when learning how to use the assignment operator: a variable is like a box, a computer "remembers" numbers after an assignment statement, and assignment is like mathematical equality. The CARL system models these findings by using multiple analogies to incrementally extend and repair knowledge in the target domain. For example, while the box analogy may be acceptable for simple assignment statements with

a single assigned entity, more complex arithmetic assignments like *"X = B + 1"* reveal errors in a student's conception of assignment. A common misinterpretation places *"B + 1"* inside the variable, *X.* A study by Bayeman and Mayer [7] of novice BASIC programmers reports similar errors. Burstein argues that this misconception can be overcome by combining algebraic knowledge (i.e., equality of value and algebraic operations like addition) with the physical sense of placing objects (a value in this case) within containers (a variable). In addition, subjects appear to incorporate a third analogy, that of a computer as a "human information processor" capable of interpreting algebraic expressions and carrying out storage commands. As the tutor presents a variety of analogies, the learner incrementally acquires an effective conceptualization of the assignment operator.

In summary, CARL provides a model of learning by using multiple, overlapping analogies to build a causal representation of a target domain. Given a didactic analogy and an example of its use, the system recognizes a specific relational abstraction for use in the analogy process. A top-down relational mapping mechanism considers object-level mappings only as needed. If mapped relations violate target domain restrictions, CARL considers non-identical relational matches by choosing ancestors or siblings in a relation type hierarchy. Analogical inferences without support in the target domain are discarded. The tutor presents a series of problems that force CARL to incrementally extend existing analogies. Feedback from the tutor is used to evaluate each extended analogy, and the system uses alternative analogical sources to repair an inappropriate causal understanding of the target. Eventually, CARL consolidates these multiple analogies into an integrated causal model of assignment statements that includes: semantic representations, parsing rules, expected effects, and the ability to use assignment in simple program plans.

6.3. Pirelli et al. [92, 93]: Learning recursion by analogy to worked examples

Pirolli describes learning by *structural analogy* where a novice uses previously worked examples when learning to write recursive programs. Verbal protocol studies of novice and expert programming behavior are modeled by constructing simulations in the GRAPES production system architecture [3]. Productions are organized around AND/OR goal trees that reflect task decomposition and focus the system on achieving one goal at a time. Two learning mechanisms, *composition* and *proceduralization* [5], restrict analogy to a particular role in acquiring programming expertise. Novices attempt analogies to worked examples when their existing solution procedures (productions) fail. If successful, the analogy allows the novice to learn a new procedure or to extend the applicability of an existing procedure. As the learner's repertoire of problem solving procedures expands, the need for analogy to examples decreases.

Eventually, the learner exhibits expert-level performance of a practiced skill without using structural analogies.

For example, Pirolli and Anderson [93J present a GRAPES simulation of a protocol in which a novice college student studies a textbook example of a set intersection function and uses the example as the source of an analogy when asked to write a set difference function. Inputs to the novice simulation include a detailed representation of the source function, a specification for the target function, relevant set theory facts, and any idiosyncratic relations which the subject mentions in the verbal protocol. The target specification includes an example function call and correct result, (SETDIFF *(ABC)* (/?CD)) = (/l), and the simulation is given the goal to code SETDIFF. The simulation generates a function for set difference and records productions which bypass analogy if the target problem or a similar problem is encountered again. Table 3 shows the set intersection example and the set difference function produced by the novice simulation.

In the verbal protocol, the novice programmer spontaneously considers the definition of INTERSECTION 1 as the source of an analogy and begins elaborating a mapping between that code and the target specification. Since recognition of analogies in subject's protocols appears to be impasse-driven, the simulation recognizes an analogy when no existing productions allow progress towards a solution. This capacity is encoded by the following production:

IF the goal is to write a function
and there is a previous example
THEN set as subgoals
(1) to compare the example to the function
(2) to map the example's solution onto the current problem

Pirolli does not describe mechanisms which recognize analogous examples, but

Table 3
A LISP definition of set intersection given in a worked example and a definition of set difference found by structural analogy (adapted from Pirolli [93])

```
(DEFUN INTERSECTION I (SET1 SET2)
  (COND ((NULL SETI) ())
        ((NULL SET2) ())
        ((MEMSET (CAR SETI) SET2)
         (CONS (CAR SETI) (INTEKSECTIONl (CDR SETI) SET2)))
        (T (INTERSECTION! (CDR SETI) SET2))))

(DEFUN SETDIFF (SETI SET2)
  (COND ((NULL SETI) NIL)
        ((NULL SET2) SETI)
        ((MEMBER (CAR SETI) SET2)
         (SETDIFF (CDR SETI) SET2))
        (T (CONS (CAR SETI) (SETDIFF (CDR SETI) SET2)))))
```

Anderson [2] presents supporting details for this class of production system architectures. Anderson's framework distinguishes between two long-term memories: procedural and declarative. Procedural memory contains productions for carrying out some task (e.g., writing LISP functions), while declarative memory contains an associative network of factual knowledge and example problem specifications (e.g., an abstract description of INTERSECTION and its function definition). Retrieval of source examples from declarative memory occurs by spreading activation through declarative memory elements, each having a trace strength which filters activation. Facts enter working memory by reaching a high state of activation. In the production shown above, the condition that "there is a previous example" could simply detect a working memory clement retrieved by spreading activation from the target problem statement.

If an initial comparison of problem features (e.g., argument types) shows sufficient similarity, structure mapping productions elaborate a mapping from the source solution to the target problem. Pirolli shows three of these productions:

IF the goal is to map an example structure onto the current problem and that structure has known components
THEN map those components from the example to the current solution

IF the goal is to map a conditional clause
THEN map the conditional of that clause
and set as subgoals
(1) to determine the action in the current case given the mapped condition
(2) to code the new condition-action clause

IF the goal is to code a relation
and a code template exists for the relation
THEN map the code template

The first production posts subgoals to map each component of the source code into the target definition; the second provides a subgoal decomposition for mapping conditional clauses; and the third allows retrieval and use of knowledge about LISP (e.g. CDR-ing through the elements of a list). The domain-specific content of the second and third mapping productions minor statements made during the novice's protocol and suggest that mapping productions can be learned during problem solving. Although Pirolli does not describe a general-purpose mapping facility, Anderson [2, pp. 209-214] shows more detailed productions for elaborating target material by using a retrieved declarative schema (source).

Evaluation is a goal-directed activity invoked by subgoals contained in the actions of mapping productions. A variety of knowledge sources are used to evaluate the mapping from source to target, including background knowledge about LISP functions and set theory. For example, the second condition of INTERSECTION1 asserts that a call with an empty second argument should return the empty set. Using the second mapping production (above), the simulation transfers the condition element and then posts a subgoal to evaluate the proposed action in the current (target) problem specification. Mapping the empty set result into the SETD1FF specification, the simulation detects a violation, apparently by consulting background knowledge about set theory. When the second argument is an empty set, SETDIFF should return its first argument as the correct set difference. This correction is made by replacing the condition's action with the variable name of the first argument, SETl. Mapping and evaluation alternate until each component of the source definition has a corresponding component in the target definition. Although Pirolli does not explicitly show how background knowledge (e.g., an encoding of facts about sets) and the target specification can be combined to generate difficult mappings, he summarizes the protocol evidence by arguing that analogy mapping "... was never a mindless symbol-for-symbol mapping." [93, p. 258]

After successfully completing the component level mapping, the simulation writes an effective target function and consolidates the results of analogical problem solving into partially generalized productions. A *proccduralization* learning mechanism directly compiles facts which were originally retrieved from declarative memory into production conditions, while a *composition* mechanism combines two productions which were fired in sequence to yield a single production. Generalizations are obtained by deleting unnecessary conditions and retaining intermediate variables in a series of composed production instantiations or turning unlike constants from target and source into variables.[13] These learning mechanisms allow the GRAPES novice to acquire the following productions:

IF the goal is to code a relation on two sets SETl and SET2
and the relation is recursive
THEN code a conditional
and set as subgoals to
(1) refine and code a clause to deal with the case when SETl is NIL
(2) refine and code a clause to deal with the case when SET2 is NIL
(3) refine and code a clause to deal with the case when the first element of SETl is a member of SET2

"Anderson [5] makes the broader argument that all generalization arises through compilation of deliberate analogical comparisons.

(4) refine and code a clause to deal with the else case

IF the goal is to code a relation causing a function to repeat on the rest of a list and this occurs in the context of writing a function that codes the relation on the list
THEN insert a recursive call of a function with the argument the CDR of the list

The first production directly encodes aspects of the analogical mapping of conditional clauses between INTERSECTION! and SETDIFF. The second production compiles a successful mapping and repair of the third condition of INTERSECTIONI into a rule which encodes a typical sense of CDR-recursion. While neither production gives a completely accurate account of recursive function definitions, they do provide coverage for a class of functions which the simulation could solve directly without resorting to further analogies. For example, the simulation codes a SUBSET function as its next task without resorting to analogy, but later reaches an impasse when attempting to code a POWERSET function.

In summary, Pirolli builds on a sizable body of work in adaptive production system architectures to present a model of learning by analogy. Novice learners recognize analogies to prior examples when they reach a problem solving impasse. Access to prior examples is not completely described, but an initial comparison and similarity threshold mechanism allow a simulation to select a single appropriate source problem. An analogical mapping between source example and target problem specification is elaborated by structure mapping productions which may (with experience) reflect task-specific information. Pirolli does not discuss the complexity of this mapping process. Source elements transferred across the analogy mapping are evaluated against the target specification, using background material encoded as productions or declarative facts. These knowledge sources may allow the simulation to repair inappropriately transferred material, although detailed mechanisms for repair are not discussed. Knowledge compilation operators consolidate successful analogies by directly encoding mapped elements and repairs as productions. During later attempts of the same or similar problems, these productions are applied directly. Pirolli argues that structural analogy to worked examples is important for early learning in a task domain, is quite sensitive to the quality of source material in a worked example, and decreases in prevalence as the learner becomes more skilled in the domain.

6.4. Kedar-Cabelli [61, 62]: Purpose-directed analogical reasoning

Kedar-Cabelli argues that existing approaches to analogical reasoning provide insufficient constraints on *which* aspects of the source should be extended to the target. As noted by others in the psychological and computational litera-

tares (e.g., Ilolyoak [55] or Greiner [41]), contextually-relevant analogical inferences may be a small subset of all possible inferences. Many studies of analogical reasoning represent *only* relevant causal structure in advance of making analogical inferences or give explicit hints about which source structures to consider for elaboration. Kedar-Cabelli proposes a model of analogical reasoning which uses explicit knowledge of the *purpose* for which an analogy is being constructed to constrain the process of elaboration. In addition, the model uses techniques for explanation-based learning [64, 83].

In overview, Kedar-Cabelli describes a method for learning concepts about common objects (e.g., a cup or a vehicle) by using an analogy between two concept instances: a given target instance and a retrieved source instance. For example, Kcdur-Cabclli [62] uses a problem in which the concept of a HOT-CUP used for drinking hot liquids needs to be refined after finding a new target instance, a styrofoam cup. Using a ceramic cup as the source, what gets transferred through the analogy is an explanatory structure which justifies using the ceramic cup to drink hot liquids. In an earlier paper [61], Kedar-Cabelli describes acquisition of legal concepts by re-using justifications in a similar fashion. Purpose-directed analogy requires several important pieces of information as input. First, the concept being refined is known: it is sufficient for instances seen thus far, but must be refined to cover the target instance. Second, the concept's purpose is given. In the example case, the purpose of a HOT-CUP is to enable drinking hot liquids. Third, a *domain theory* is given with axioms for explaining how physical attributes of objects relate to their functional roles. In the example, an object's handle enables grasping. As final input, structural attributes of the target instance are given.

Learning concepts by purpose-directed analogy proceeds in five stages:

Step 1. *Retrieve* a typical source instance of the goal concept. As a simplification, the source is given.

Step 2. Explain why the source instance is considered a member of the goal concept for the given purpose. A derivation of the explanation requires several steps:

(a) Find a plan that achieves the stated purpose. For drinking hot liquids, a plan might be: to put the liquid in the container, to keep the liquid in the container and hot long enough to drink it, to grasp the container, to pick it up, and to drink the liquid.

(b) Extract *object preconditions* from the plan (e.g., that the container object can be grasped). These arc functional requirements which objects must satisfy if the plan from Step 2(a) is to be used.

(c) Construct a network of "explanatory" inferences which show how structural characteristics of the source object enable functional requirements of the plan (sec Fig. 9). This network is deduced from domain theory axioms—e.g., having a handle and being constructed of ceramic material

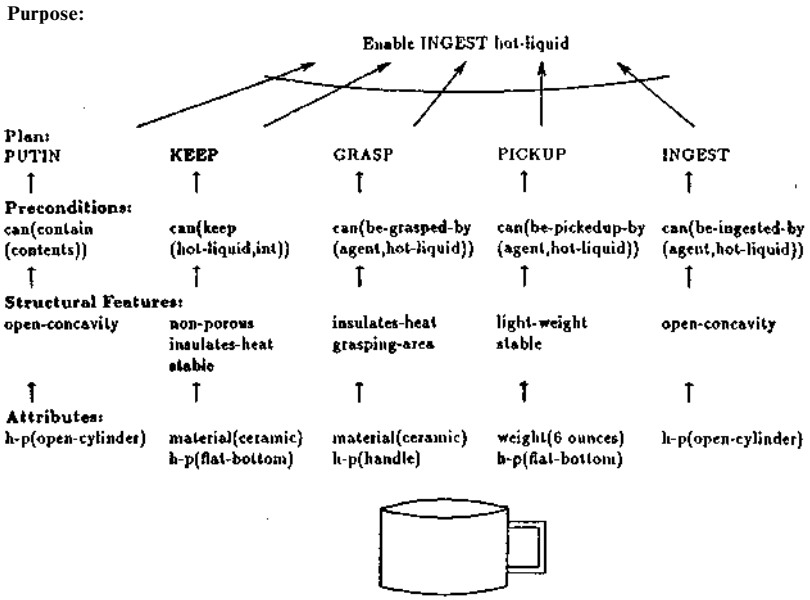

Fig. 9. Explanatory inferences satisfy plan preconditions (adapted from Kedar-Cabelli [62, ρ 154]).

provides an insulated grasping area which an agent can use to grasp the cup. An adequate explanation enables the plan provided in Step 2(a) above.

Step 3. *Map* the explanation found in Step 2 into the target.

Step 4. *Justify* the explanatory inferences of Step 2(c) for attributes of the target instance. Justification starts with structural attributes of the target, and may require finding alternative attributes or even plan steps which are effective for satisfying, the stated purpose for the target object. For example, the styrofoam cup is graspable because it is conical, rather than having a handle.

Step 5. *Learn* or refine the goal concept. This can be accomplished by recording justifications for source and target objects, or by inducing a characterization over both objects treated as positive instances.

The resulting HOT-CUP concept specifies that an object "can have an open concavity, can be made of nonporous, insulating material, can be stable, lightweight, and can be graspable." [62, p. 154]. This is a sufficient characterization of HOT-CUPs formed by finding a common network of explanatory inferences within each of the instance explanations. Note that the "graspable" attribute is a functional rather than structural term, which results from the absence of a handle in the target instance (styrofoam cup).

Kedar-Cabelli's model primarily focuses on processes of elaboration and

consolidation. Rather than considering any plausible analogical mapping between source and target, attention is given only to information which provides an explanation for why the source instance satisfies the learning purpose. In the HOT-CUP example, only structural features which are salient for using the ceramic cup to drink hot liquids are considered when elaborating and evaluating the analogy. Analogical inferences about where the cup was purchased or whether the cup was given as a gift are never considered. However, in order to use purpose-relevant features of the source, the retrieval and plan finding processes (Steps 1 and 2(a) above) must first provide a source instance and plan. Consolidation provides the surrounding reasoning context in this model: analogies are used to refine existing concepts with respect to a stated purpose. A concept description is refined by finding a common inference network between explanations for source and target instances. The model is less clear on how learned concepts are stored or subsequently used.

Purpose-directed analogy treats recognition and evaluation less fully. Recognition of the source instance is given, while recognition of an appropriate plan through which the source object can be used to achieve the given purpose (e.g., drinking hot liquids) is not described. The importance of this latter step is crucial since minor planning variations can lead to different explanatory structures. Evaluation of an explanatory network mapped from source to target depends strongly upon the given domain theory. Axioms of the domain theory must be a priori sufficient not only for deducing a source explanation but also for justifying that explanation when mapped into the target. Justification is an evaluative process which, as described, must either deduce an appropriate target explanation (as mapped or newly created) or find alternative plan steps. Thus, the results of analogical inference could also be inferred from the existing domain theory. At a pragmatic level, the utility of analogy for learning becomes an empirical question. Learning efficiency or concept quality might be compared with other learning techniques. At a theoretical level, there is an issue of whether learning occurs since every inference (analogical or otherwise) is derivable from existing knowledge (e.g., compare with Greiner's [41] "unknown" restriction).

7. Analogical Reasoning in Review

The studies reviewed in preceding sections each make contributions to a computational account of analogical reasoning. In this section, these contributions are examined within each component of the abstract process model used as a guide for detailed reviews. These include:

(1) *recognition* of an analogous source,
(2) *elaboration* of an analogical mapping between source and target,
(3) *evaluation* of the elaborated analogy,
(4) *consolidation* of information generated while using an analogy.

From a computational perspective, each process presents a number of problems. Comparisons drawn in this section examine the range of solutions offered by computational studies and identify common approaches to basic problems in analogical reasoning. Appendices A and B summarize detailed reviews within each process component and provide a reference for comparative sections.

7.1. Recognition of a candidate analogy

Given an unfamiliar situation (the *target),* how does a reasoner connect this new situation with one or more familiar situations *(sources)* contained in a store of previous experience? From a computational perspective, search is implicated and an organization is usually imposed on the store of previous experience to help constrain search for a candidate source. From a cognitive perspective, a reasoner attends to familiar aspects of the target and uses these aspects to retrieve appropriate experiences from memory. By allowing partial similarity between target and candidate sources, a central problem of recognition is to impose constraints on the retrieval process but still allow recognition of analogically related sources. For example, strict organizational criteria that suppress tenuously related candidates might not allow recognition of relatively abstract inter-domain analogies. In the following comparison, computational approaches to recognition are organized around increasingly elaborate organizational constraints.

The most effective but least ambitious solution to constraining recognition is to give the reasoner a source analog. Some studies do this as a simplifying assumption (e.g., Evans [28], Kling [66], or Pirolli and Anderson [93]), while others give a hint about the source and rely on supporting mechanisms to complete recognition. For example, Kedar-Cabelli [62] gives the learning purpose, the to-be-learned concept, and a target instance. The system then selects a prototypical source instance and a plan for using that instance to achieve the given purpose. Using a similar approach, Greiner [41] gives an initial mapping (a hint) between source and target concepts, and then finds a source instantiation and an abstraction for solving it. Both approaches use the relation of an abstraction (or plan) to a source instance during later stages of analogical reasoning. Brown's [11] reduction analogies, Winston's [108] simile-based instruction, and Burstein's [15] integration of multiple analogical models each place analogy in a tutorial context. In all three cases, the reasoner uses a hint and the ongoing tutoring context to recognize salient aspects of the source. For example, Burstein gives the analogy (e.g., a variable is like a box) and examples of its use, and CARL retrieves a source abstraction (e.g., a causal model of containment) to extend the analogy.

Without giving the analogy directly, other source candidates compete for attention and require an organization that restricts their number. This organization is generally an indexing scheme that enforces selective retrieval. The

reviewed studies use three general approaches: nonselective indexing, task-specific indexing, and task-independent indexing. In each, the question is what to choose as indices into the store of candidate sources. Choosing an indexing scheme makes an explicit commitment to the kinds of analogies that can be recognized.

Nonselective indexing schemes approximate an associative memory for candidate sources. For example, Becker [9] indexes candidate schemata through generic concept nodes, while Munyer [85] indexes formulas around instances of functional containment. Each approach promises extensive search in a memory with relatively primitive organization. In practice, each applies additional constraints to the search process: Becker insists that mapped kernels occur in an appropriate position in the schema, and Munyer requires consistent formula mappings at either end of a candidate derivation. In both cases, recognition returns a set of candidate source analogs, and one (or several) are selected during elaboration and evaluation. For example, Munyer suggests an agenda control mechanism prioritized by the "degree of certainty" for competing analogical views.

Task-specific indexing schemes select distinguished elements of the representation and make an a priori commitment that these elements predict future contexts of use for the source. Winston's [109] "classification-exploiting hypothesizing" resembles this scheme, although he mentions using relational indexing in the bottom-up voting mechanism. His later work [110] indexes acquires rules by the types of actors, acts, and objects found in their right-hand sides. Extracting type cues from the target problem, Winston retrieves sources which make predictions about those types. McDermott [74] uses a similar strategy when indexing source method productions by types of objects and actions. As with nonselective approaches, both studies include further constraints on recognition. Winston [109] weights his voting scheme in negative proportion to source concept prevalence and in positive proportion to the contextual salience of the target concept. McDermott, on the other hand, generates taxonomic variants of the target cue to make contact with method indices. In each case, differential focus on target elements refines cue extraction, allowing the reasoner to influence the recognition process by manipulating elements of the target description. Although task-specific indexing and cue extraction prove effective for the problems solved in these studies (e.g., painting or washing tables), these methods may not extend across more heterogeneous tasks and may not recognize more abstract analogical similarity between target and source domains.

Task-independent indexing schemes select more abstract representational elements for indices that organize memory. Carbonell's [16, 17] "invariance hierarchy" over semantic categories is an example of this approach. Examining metaphors and analogies in different domains, Carbonell ranks semantic categories by decreasing order of invariant transfer. The resulting hierarchy

specifies that goals, plans, and causal structure are usually preserved in an analogical mapping. Carbonell argues that this invariant hierarchy is important for recognizing analogies since memory can be organized around (i.e., indices are based on) precisely the knowledge structures that are likely to transfer without variation when reasoning by analogy. Thus, recognition proceeds by extracting goals, plans, or causal structure from the target and using these as indices into a memory for candidate sources. For example, Carbonell [18] organizes a memory for solution sequences in ARIES around state descriptions (initial and goal states) and constraints, and then uses a similarity metric based on the same information to select among recognized candidates. Dyer also [261 uses this indexing scheme to organize memory around instances of planning failures (TAUs). Planning difficulties in a target narrative serve as retrieval cues for recognizing adages and analogous narrative episodes. Likewise, Simpson [103] recognizes analogous cases by comparing a target description with generalized episodes in a memory organized hierarchically around problem types and planning information. Traversing indices in episodic memory structures guides the recognition process through increasingly specific comparisons ending with retrieval of candidate cases. By indexing and retrieving over task-independent semantic categories, these approaches can support recognition and retrieval of genuinely novel metaphors or analogies.

Of these three indexing schemes, the task-independent approach might be preferred since it anticipates retrieval of useful source candidates and clearly allows recognition of analogies where target and source content are markedly different (e.g., choosing a restaurant and scheduling a meeting, as discussed in the section on Dyer's work). Also, as argued by Schank [99], Dyer [26], and Kolodner [67, 68], these approaches appear consistent with human studies of episodic memory organization and retrieval. On the other hand, task-independent indexing schemes could make overly strong a priori commitments to the utility of source situations, preventing access in some useful but unexpected contexts. This is especially true when the target is completely novel, since the reasoncr may -not be able to extract cues required for recognition of a useful analogy from a memory organized around abstract semantic categories. Evidence from studies of human analogical access [35] suggests that recognizing an analogical source may depend on different principles than those that determine elaboration and evaluation. At present, it seems likely that analogical retrieval depends on interactions between several factors: what the reasoner attends to in the target situation, what is available in the store of source experiences, and the degree to which the reasoning context during recognition matches the encoding context for a stored source. These tradeoffs are open research questions for computational studies. As psychological models of memory organization and retrieval become more explicit, computational approaches to recognition may benefit; the converse may also be true.

7.2. **Elaboration of an analogical mapping**

Among the four processing components of analogical reasoning reviewed in this paper, elaboration receives the largest share of attention. As with recognition of a candidate source, elaboration addresses a general problem: finding an analogical mapping between elements of target and source domains. The mapping shapes and, in some cases is shaped by, the specific kinds of information that can be transferred between domains. For example, when McDermutt [75] tentatively identifies a method for achieving a current goal (e.g., using a method for painting to wash an object), the mapping between these situations (washing and painting) depends on the mutability of antecedent conditions for painting against the conditions of washing. Thus, elaboration and evaluation processes are often intertwined. Elaboration incrementally extends a mapping between target and source domains that supports transfer of analogical inferences, and the effectiveness of these extensions are subject to evaluation. As possible, however, evaluation will be discussed separately in a later section.

Elaboration of a mapping between domains must start somewhere. For example, Winston [113] begins by mapping compatible relations and then scores the fit of the surrounding structures according to their relative importance. In contrast, Dyer [26] starts with abstract relational information about planning failures and conducts a top-down elaboration of the mapping between narrative episodes. In each case, elaboration starts with a partial mapping uncovered during recognition. This initial mapping is established when an effective larget retrieval cue contacts elements in a candidate source.

7.2.1. *Constraints on analogical mapping*

Constraints on an analogical mapping can either be independent of interaction between source and target domains or come about by an evaluative comparison of these domains. Comparative constraints are considered later under the evaluation of an analogical mapping. In this section, approaches to defining and using independent constraints are compared as alternative preferences for what to preserve in the analogical mapping. The general problem of elaboration is to restrict the enormous space of possible mappings between source and target to a smaller space of plausible or useful mappings.

One obvious constraint is to use an existing analogy mapping or a consistent extension of an existing mapping. Winston's [108] preference for existing transfer frames is an example of the former, while Brown's [11] extension of a reduction analogy is an example of the latter. Extensions are consistent when existing associations are not violated and added associations obey any a priori restrictions on the kinds of mapping allowed (e.g., predicates are mapped one-to-one in Kling's ZORBA-I [66]). To re-use or extend a mapping,

it must be found in the first place. This leads to independent mapping constraints.

In the reviewed studies, independent constraints can be organized into three classes: preserve the relational structure of the source description, preserve semantic categories determined a priori, and preserve material relevant within a surrounding reasoning context. Each constraint class introduces preferences that restrict the elaboration of an analogical mapping.

The first class of preferences considers representations of the source and target, asking what aspects of those representations should be preserved. The most general approach, as evident in many studies, is to preserve the *relational structure* of the source representation. For example, Brown [11] maps predicates from source to target domains only if those predicates have the same type and their arguments have a type-compatible mapping. With similar effect, Munyer [85] uses a bottom-up approach in which local maps between arguments compete to reinforce predicate mappings higher in the representational network. In both cases, the relational structure of a source representation is preserved in the analogy mapping if a corresponding structure can be found in the target representation. This approach is also found in algorithms for computing inductive summaries over instances (e.g., Hayes-Roth [48]) and has been studied systematically by Falkenhainer [29, 30][u] as a computational realization of Centner's structure-mapping theory [34].

The second class of preferences focuses on semantic categories of source and target knowledge, asking what semantic structures are commonly preserved in analogies and metaphors. These preferences range from task-specific restrictions to preserving more general informational categories in the analogical mapping. Evans' [28] restriction of a one-to-one mapping of rule components and Pirolli and Anderson's [93] compilation of mapping rules are examples of task-specific semantic preferences. Winston's promotion of salient source properties when comprehending similes [108] or his preference for salient relations (e.g., cause or enablement relations) in importance-dominated matching [109] are intermediate along this continuum. Carbonell's invariance hierarchy [16, 17] and Simpson's use of that hierarchy to organize memory and direct elaboration [103] are examples of the most general preference for semantic categories. Whereas the first class of mapping preferences preserve relational structure in source and target descriptions, this class promotes semantic categories deemed important for the analogy a priori. Relational structure and important semantic categories may overlap, and we return to this issue in a moment.

The third class of preferences focuses on the *contextual relevance* of mapped material, asking which relational or semantic structures to preserve within the

[u] Their structure-mapping algorithm also rewards candidate mappings whicli support analogical inferences.

current reasoning context. Since an arbitrarily large collection of facts might be known of the source or target, some mechanism must focus on those facts which are important at the moment. Contextual relevance is a broad concept, and takes different forms in the studies reviewed here. For example, Burstein [15] uses a tutorial context to select among alternative relational abstractions in the source domain. Also arguing for contextual constraints, Keclar-Cabelli [62] uses a to-be-learned concept and its stated purpose (e.g., drinking hot liquids) to focus elaboration on explanatory inferences used with a source instance of the concept. Perhaps the strongest adherent to contextual relevance, Hobbs [51] argues that resolving discourse problems in context finds a coherent metaphorical interpretation.

In isolation, these three preference classes for elaborating an analogical mapping may seem incompatible. For example, relying solely on a preference for preserving semantic categories, a reasoner might attempt to map isolated and potentially irrelevant source goals, plans, or causal relations. These could be suppressed by a mapping strategy that preferred maximally coherent (or "systematic" [34]) relational structures. In contrast, relying solely on a preference for relational structure, a reasoner might fail to map attribute-level information that is critical for achieving some goal. These and other arguments are levelled in detail by Holyoak [55] and Gentner [36] and are relevant for computational research.

From an integrative viewpoint, constraints provided by all three preference classes contribute to processes of analogical reasoning. When recognition and evaluation are considered as pre- and post-processes to elaboration, many of the more strident contrasts bctsvecn these approaches fall away. For example, contextual constraints on recognition help to restrict the relational structures available for mapping, while evaluation processes give a posteriori force to a preference for semantic categories. Since these categories tend to be represented as higher-order relational structures, the more parsimonious preference for preserving relational structure within elaboration may be a tenable approach, provided that contextual and semantic constraints surround the mapping process.

7.2.2. *Varieties of analogical inference*

Comparing different approaches to analogy, elaboration of a mapping between target and source domains is clearly a process of varying complexity. In some studies, finding a mapping between target and source descriptions directly achieves the purpose of the analogy. For example, Evans' [28] ANALOGY system generates a set of generalized rule candidates, choosing the one that best preserves a one-to-one, type-consistent mapping between source figures. Similar descriptions apply to most psychological studies of proportional analogies (e.g., [104]) and comparison-based theories of metaphor comprehension

(e.g., [73]). In contrast, other studies describe elaboration as an active, incremental process. For example, Carbonell [18, 20] starts with a partial mapping over problem specifications (e.g., states and constraints) and then enters a complicated search space of plan transformations or replayed derivational steps to find a solution for the target problem. The repairs described by McDermott [75] and Burstein [15] or the justification for a new case described by Kedar-Cabelli [62] suggest similar complexity when elaborating an effective analogy.

Simple, relatively homogeneous correspondence as an end in itself supports a limited view of analogy: analogical comparisons finds a mapping which renders two superficially dissimilar situations virtually identical. In this view, the real work of analogy is in elaborating a consistent mapping, and analogical inference is either missing or given a limited role. In contrast, more complex views of elaboration see analogy as an open-ended, experimental process. An elaborated mapping supports analogical inferences from a well-understood source domain into a less familiar target domain. These inferences are hypotheses that must be verified in the target domain, giving rise to an experimental interplay between elaboration and evaluation.

7.3. **Evaluation of the analogy**

As in elaboration, selecting and using constraints provides the central story line for evaluation. Assuming that elaboration incrementally proposes extensions to a mapping between source and target domains, evaluation examines the plausibility of these extensions in the target domain. From a cognitive perspective, Centner's [34] description of the *soundness* of an analogical mapping and Holyoak's [54] distinction between *structure-preserving* and *structure-violating differences* address the same evaluative concerns. Map extensions between known elements of source and target domains are largely covered by constraints discussed in the previous section. However, map extensions that propose unknown target elements (i.e., analogical inferences) are verified by evaluative processes discussed here. In addition, global evaluative feedback is needed to determine whether resources used for analogical reasoning might better be devoted to alternative problem solving techniques. For example, Carbonell [20] describes a perseverance threshold that detects a fruitless analogy and directs the problem solver to fall back on alternative techniques (e.g., weak search methods). Munyer [85] argues for similar global evaluation by estimating the degree of certainty in the current analogical mapping. In contrast, Simpson [103] recursively invokes case-based reasoning to remediate global failures in analogical problem solving.

As described in the previous section, constraints when target and source are considered independently are distinct from constraints that arise through domain "interaction." Source and target domains interact when the analogical

mapping is used to transfer inferences and supporting material from source to target. In evaluation processes, interactive constraints are of central importance. As with other analogy processes, there are diverse proposals for identifying and using interactive constraints. This diversity focuses on two problems: how to *confirm* inferences extended from source to target and how to *repair* inappropriate extensions. Obviously, these problems are not easily separable as exclusive concerns of evaluation. Instead, elaboration and evaluation are interdependent processes that both operate on an analogical mapping. In a figurative sense, this mapping provides a bridge between source and target domains [91]. Confirmation of information transferred over this bridge and repair of its structure in the face of an incomplete or inappropriate mapping are managed by elaborative and evaluative processes working together. What results should be relevant within the reasoning context.

7.3.1. *Confirming analogical inferences*

In the reviewed studies, two approaches are used to evaluate analogical inferences extended from the source to the target domain. Inferences are tested for validly or usefulness in the target domain, or the reasoner attempts to establish a justification for inferences about the target domain which mirrors their justification in the source domain. These approaches are identical in their attempt to validly instantiate knowledge extended across domains, but re-using a justification provides added constraints on *what* to consider as supporting source information.

The plausibility of analogical inferences can be confirmed by consulting prototypical expectations of the target domain or verifying the usefulness of inferences in some ongoing reasoning process. In either case, evaluation *tests predictions* about the target domain. As an example of confirmation using target expectations, Winston [108J "filters" inferred target properties by checking that they fill slots or have values found in a "typical" target instance. In Winston's later work [110], abductive reasoning verifies an analogical inference when its consequences are known in the target domain or provided by a tutor. In both cases, existing knowledge of the target is used to confirm predictions from the source domain.

More ambitious evaluative strategies weigh the problem solving *utility* of analogical inferences. For example, Carbonell's [18] transformational analogy mechanism uses a similarity metric to select T-operators which incrementally transform a source solution sequence into a target solution sequence. Features used in this metric (e.g., comparisons of states or path constraints) encode knowledge of desirable or undesirable solution forms in the target domain. In a more general deductive framework, Greiner's [41] NLAG must prove that an analogical conjecture is useful for solving the target problem. As an alternative to task-specific knowledge of the target domain, Burstein [15] uses critical interactions between CARL and a tutor to collect feedback on analogy-driven

solutions that includes corrections for wrong answers. In each approach, the success of an analogical inference in reaching a target solution is used to evaluate the analogy.

Taken in isolation, a fact or action suggested by an analogical inference may be plausible, but the reasons supporting that fact in the source domain may not be plausible when evaluated in the target domain. A common solution is to map source justifications for analogical inferences into the target domain and then to establish their validity. A *justification* gives a representational description of the "reasons" which support an inference or action in some domain. Becker [9] gives an early example of this approach by collecting facts which justify a "motivated" analogical mapping. The motivation is to apply a schema in his prediction paradigm, and justifying facts are unmapped source kernels in either side of the schema (e.g., antecedent kernels in a forward application). Somewhat more direct, Winston's [108] justification frames explicitly capture those aspects of a target description which must be present for a known analogy (i.e., a transfer frame) to be useful. For example, a justification frame for an analogy between a table and a cube to be used for a common purpose (e.g., to eat or write) might record that both target and source objects must be of medium size, have a flat top, and be level. Using functional justifications is extended by Winston et al. [112] and used to good purpose by Kedar-Cabelli [62]. In purpose-directed analogy, an explanatory justification generated in the source domain (e.g., the structural reasons why a ceramic cup can be used to drink hot liquids) both confirms and constrains analogous reasoning in the target.

Replaying justifications is central to some computational studies of analogy. For example, Brown [11] represents plan justifications as collections of assertions which relate steps in a solution plan to facts about the task domain found in a goal description. After generating a justified source solution, these assertions must be confirmed when the candidate solution is "inverse-mapped" into the target domain. If justifications cannot be confirmed, further elaboration of the existing analogy or introduction of a new analogy are attempted. Carbonell's derivational analogy method [20] also replays justifications, stored as part of a derivational trace of decisions made when solving a source problem (e.g., programming quicksort in PASCAL). Given an analogous target problem (e.g., programming quicksort in LISP), the reasons for choosing among actions in the source derivation must be confirmed or replaced by alternative reasons for the derivational analogy to succeed."

"Replaying derivational traces (including justifications for decision points) is also a going concern in software engineering [100, 106J, where the goal is to re-use a history of program development. Mostow [84] describes problems with this approach from a machine learning perspective, and Dershowitz [24] explores program modification and abstraction by analogy.

7.3.2. *Repairing a faulty analogical extension*

In addition to confirming or justifying analogical inferences, many computational approaches consider how to recover when an extension of the mapping between domains fails. The specific nature of recovery depends, of course, on the seriousness of failure. Improperly instantiating transferred information generally requires less extensive repair, while inappropriately transferred information requires further reasoning or abandoning the foundering analogy.

In McDermott's ANA [75], repairing mutable precondition failures helps to instantiate transferred planning information. ANA has trouble using an analogy when it generates inappropriate subgoals, transfers insufficient constraints, or transfers unnecessary constraints. Problems are detected when actions are attempted in the paint shop environment and fail or produce unexpected results. Repair depends on a combination of background knowledge about what types of entities can participate in known methods and direct feedback from the user. For example, loading a spraying machine with paint when the goal is to wash an object is repaired by substituting water for paint. Underspecification of transferred methods can lead to planning failures like stacking too many objects in a constrained area. These failures are repaired by generating additional subgoals which transform problematic but mutable aspects of the plan through known methods (e.g., removing an object from a crowded area) or advice from the task master. Finally, difficulties with overspecified plans (e.g., moving an obstructing object to a distant location when a closer location suffices) are repaired by asking the task master which constraints are unnecessary. In some respects, McDermott's approach to repair is a task-specific precursor to Carbonell's transformational analogy method [18].

A number of studies use multiple analogies to repair inappropriate analogical inferences. For example, Burstein's [15] CARL integrates multiple analogical models (e.g., physical containment and human memory) to repair incorrect predictions about simple assignment statements. Among the variety of studies using GRAPES simulations, Anderson et al. [3] also model problem solving sessions in which the tutor presents a simplifying example to help repair incorrectly transferred LISP code. In both cases, errors from inappropriate analogical inferences are repaired by introducing additional analogies. These must be integrated with the original analogy. In related psychological studies, Clement [23] describes how expert problem solvers in physics use intermediate "bridging analogies" to help elaborate an analogical mapping between a target problem and a troublesome analogical source. Each approach is computationally relevant and psychologically plausible, but integrating multiple analogies may introduce other difficulties. Multiple analogies, possibly at differing levels of abstraction, must be combined into a usable concept, avoiding what Halasz and Moran characterize as a "baroque collection of special-purpose models" [45, p. 34j.

In summary, analogical inferences must be treated, at best, as tentative hypotheses supported by a partial mapping between source and target domains. Domain interactions during evaluation confirm and repair analogical inferences extended during elaboration. Evaluation occurs at many levels: testing analogical predictions against expectations of what is typical of the target domain, verifying the utility of analogical inferences in some reasoning context, replaying justifications for analogical inferences in the target domain, and repairing inappropriate analogical inferences. As a result of the evaluation process, parts of the analogical mapping may be changed or deleted, multiple analogies may be combined to suggest new hypotheses about the target domain, or the original analogy may be abandoned altogether in favor of an alternate line of reasoning.

7.4. Consolidation of analogical reasoning

Recognition, elaboration and evaluation result in a mapping between target and source domains that supports verified analogical inferences. "Learning by analogy" considers how to consolidate these materials to improve future reasoning performance in these or similar domains. Most studies consider learning when analogies succeed, although learning opportunities also arise when analogies fail.

7.4.1. *Varieties of analogical learning*

The simplest form of consolidation directly records information successfully transferred from source to target domain. Of the reviewed studies that address consolidation, most perform this simple form of learning. For example, McDermott [75] and Pirolli and Anderson [93] record specific target productions; Hobbs [52] creates and extends a target schema; Winston [109, 110] records successful target cases; and Greiner [41] augments the starting theory with useful target conjectures. In each approach, learned material is strongly context-specific with little or no generalization. When facing a new task which is identical to an earlier success, the earlier solution is applied directly without resorting to more costly inference mechanisms. Although this simple learning scheme might seem limited, when coupled with powerful recognition and elaboration processes, it could achieve incremental performance improvements as the collection of source candidates provides wider domain coverage.

A more ambitious form of consolidation stores a successful analogical mapping for later use under similar circumstances. This strategy stores the process of analogical reasoning in the hope that elaboration and evaluation can be reused or extended without further effort. For example, Winston [108] stores transfer and justification frames. When reasoning about new similes, recognition first attempts to reuse an acquired transfer frame if related justification frames can be verified for the target. Using a similar approach,

Pirolli and Anderson [93] acquire task-specific mapping productions which supplant portions of later elaboration attempts. Other studies save the analogical mapping for the duration of an instructional context. For example, Brown [11] and Burstein [15] incrementally extend and repair a mapping as new tasks or feedback are given by a tutor. Although this may seem a matter of technical convenience, analogical reasoning is often an explicit component of tutorial interactions, and computational techniques for managing analogical comparisons (e.g., diagnosis or direct manipulation) can provide useful instructional or experimental tools.

To acquire knowledge with wider applicability, many studies form inductive summaries over target and source materials. Becker [9], Winston [110, 114], and Pirolli and Anderson [93] acquire generalized rules which consolidate inferences common to target and source. Becker's learner refines schemata through experience with a reactive environment; Winston's learner forms rules and censors from a series of predictive tasks presented by a tutor; and Pirolli's learner compiles analogical comparisons of declarative material into productions. Other studies use inductive mechanisms to form more complex plans or problem solving derivations common to target and source domains. For example, Carbonell [18, 20] consolidates successful transformational analogies into generalized solution sequences and derivational analogies into generalized plans and search heuristics. Similarly, Burstein [15] argues for concept formation through analogies supported at varying levels of abstraction (e.g., causal inferences and plan steps) but includes learning from multiple analogical sources that cover different aspects of the target problem.

While consolidation processes discussed above record materials uncovered during successful analogical reasoning, learning opportunities also exist when analogy fails. For example, McDermott [75] and Simpson [103] record unsuccessful analogical inferences (e.g., inaccurate problem classification or inappropriate subgoals) and their resolution, and later recognize these materials in similar problem solving contexts. McDermott uses failures to help select among candidate source analogies and to divert ANA from pursuing inappropriate subgoals, while Simpson treats failure remediation as another case-based problem solving task. Rather than changing or updating domains, Carbonell [18] changes recognition and evaluation mechanisms by using information obtained during failures. When recognition fails to suggest any candidate analogies, alternate problem solving methods may reach a solution. A postmortem analysis can suggest that the new solution is similar to a previously known problem solution. In this case, Carbonell generalizes the similarity metric so that source solutions will be recognized in the future. Likewise, if a recognized analogy fails to produce a solution, the similarity metric is specialized to suppress false recognition in the future. Similar learning strategies tune the enablement conditions of transformation operators which are not recognized as applicable or fail once they are recognized.

7.4.2. *Analogy in machine learning*

The reviewed studies present a variety of mechanisms for "learning by analogy." These include recording target materials imported by analogical inference, recording the structure of the analogy, recording generalized rules or plans which capture common aspects of target and source domains, and altering various constraints on analogy processes. However, taking a wider comparative perspective we might ask how learning by analogy compares with existing approaches to machine learning?

Overviews of machine learning usually differentiate learning strategies along inversely related dimensions: the complexity of inferences required of the learner versus the amount of guidance supplied by the environment [21, 25, 77]. Since the learner is knowledgeable about the source and analogically infers target information, learning by analogy is generally thought to require a less complex inference mechanism than learning from examples. In this view, the analogy generates a target example, and an inductive summary over target and source is found using techniques for learning from examples. On the other hand, when the learner recognizes the analogical source without assistance, learning by analogy may require less external guidance than learning from examples [25]. In this view, recognition of an analogous source provides an attentional function common to learning from observation or discovery. The learner actively selects which concept is being learned, the instances used to refine its summarization, and interpretations of those instances. The actual inductive summary for a chosen concept (in isolation) could again be achieved using techniques appropriate for learning from examples, although storing the refined concept with other concepts requires a strategy for memory organization and update.

In the reviewed studies, consolidation mechanisms do not introduce new techniques for inducing instance summaries. Instead, analogy provides an organizational framework for existing machine learning techniques. Recognizing analogies in a reasoning context (e.g., planning in McDermott's paint shop [75]) overlaps with issues in observational learning; elaborating an analogical mapping between source and target anticipates (e.g., as suggested by Munyer [85]) inductive summarization techniques in learning from examples; evaluating analogical inferences about the target domain resembles experimental aspects of discovery learning; and consolidation of confirmed or disconfirmed analogical inferences uses inductive summarization and memory integration techniques common to various machine learning strategies.

7.4.3. *The role of analogy in learning*

Approaches to learning by analogy mirror the techniques and difficulties of existing machine learning strategies. However, a more general question is what

role docs analogy play in learning? When the question is asked of machine learners, the answer might draw on empirical or theoretical demonstration. When the question is asked of human learners, however, the answer rests upon demonstrations of what learners can do and explanations of those capacities. In practice, analogy in machine and human learning are closely related when designed artifacts reflect common analogies (e.g., iconic realizations of the desktop metaphor [22,58]) or actively attempt to teach human learners [4,105]. An exploration of this relationship is beyond the scope of this paper. However, the reviewed studies provide contrasting perspectives on *when* analogies are important for learning and *what kind* of analogies are either accessible or useful.

Studying spontaneously recognized analogies to past examples, Pirolli and Anderson [93] report that analogy is frequent in the early stages of learning about a domain, is limited by students' generally poor memory for previous examples, and is highly sensitive to the contents of the analogous source (also see Anderson [1]). Analogical comparisons with recently encountered material (e.g., a worked example) are quickly compiled into generalized productions, and these productions mirror appropriate or irrelevant content in the source material. In combination, these observations lead to a cautionary pedagogical approach to learning by analogy: abstract models of the target competence are directly taught (e.g., a strategic description of writing recursive functions), while specific examples alone arc avoided (e.g., a recursive definition of factorial).

Burstein's [15] study of using multiple analogical models in early skill acquisition contrasts with this view in several ways. Although analogy in early learning is frequent in both accounts, Burstein reports that novices integrate analogical inferences from source abstractions in widely different domains. He also assumes that the learner understands and remembers relevant information about several analogical sources, and that the target problem and surrounding tutorial context will focus reasoning on appropriate analogical inferences. Finally, Burstein's work suggests a somewhat longer-lived role for analogy in learning. Each analogy persists across an ongoing tutoring session, is extended as neeHcd, and can be repaired when analogical predictions fail.

These viewpoints on the role of analogy in learning may not be incompatible. For example, Pirolli and Anderson's suggestion for teaching abstract models which become sources for analogical comparison may correspond to Burstein's focus on existing domain abstractions (e.g., causal abstractions for containment). Conversely, Burstein's extended elaboration and repair of the analogical mapping could be modeled by repeating structural analogies with the same source and using discrimination learning mechanisms to refine compiled productions [5]. Whether analogy is limited to early stages of learning is also open to debate. Kolodner et al. [69] argue that case-based reasoning is

common to both novices and experts across task domains. In contrast, Forbus and Gentner [32] argue that early learning about physical domains is dominated by surface-level similarity. True analogies appear during later stages of learning when the relational structure of the domain is more easily analyzed.

8. Conclusion

After reviewing computational studies of analogy by task domain and then comparing their contributions to basic process components of analogy, we can conclude by asking what progress has been achieved towards these ends in the past twenty years. Reviewing Evans' thesis shortly after completion, Minsky [81] writes:

> ... it is becoming clear that analogical reasoning itself can be an important tool for expanding artificial intelligence. I believe it will eventually be possible for programs, by resorting to analogical reasoning, to apply the experience they have gained from solving one kind of problem to the solution of quite different problems. [81, p. 251]

Eventually, robust computational mechanisms for reasoning by analogy may appear. Over twenty years after Evans' thesis, however, reasoning about a new situation by effectively using a previous experience is still a difficult problem. Computational studies of analogy have reached a sort of adolescence: exploring the structure of the problem, offering partial solutions in some areas, and identifying difficult questions in others. The research reviewed in this paper expands our appreciation for the complexities of analogy by posing the "problem" as a diversified set of interrelated problems, each providing its own challenge.

8.1. Problems and solutions for analogical reasoning

Process components of recognition, elaboration, evaluation and consolidation developed and discussed in preceding sections not only reflect the academic topography of work done in artificial intelligence and related disciplines, but these components also organize continuing research problems and proposed solutions. Table 4 presents problems, proposed solutions, and citations to exemplary studies drawn from the comparative analysis of the preceding section.

Most work on analogical reasoning from a computational perspective addresses elaboration and evaluation, and alternative approaches to these problems can be clearly distinguished. Preferences for analogical mappings that preserve distinguished representational classes or contextually-relevant material constrain elaboration, while interactive constraints on confirmation, repair, and global monitoring of analogical inferences guide evaluation. As described in

Table 4
Problems and solutions across process components of analogical reasoning

Recognition
Problem: Given a target and a store of *sources,* find a manageable but promising set of candidates.
Solutions: (1) Give the source as a simplification [66] or in a tutorial context [15].
(2) Organize the store of sources around an indexing scheme.
(a) nonselective indexing [9],
(b) task-specific indexing [HO],
(c) task-independent indexing [26].

Elaboration
Problem: Given *target, source,* and mapping *preferences,* find a mapping and analogical inferences.
Solutions: (1) Use an existing analogy map [108].
(2) Prefer analogical mappings which:
(a) preserve relational structure [85],
(b) preserve semantic categories [103],
(c) preserve contextual relevance [62].

Evaluation
Problem: Given a mapping, analogical inferences, and a reasoning context, evaluate the analogy.
Solutions: (1) Confirm analogical inferences:
(a) test predictions against target domain knowledge [108],
(b) weigh the utility of inferences in context [41],
(c) replay a justification in the target domain [2(1].
(2) Repair faulty analogical inferences:
(a) post failures as subgoals [75],
(b) integrate multiple analogies [15].
(3) Monitor global progress:
(a) heuristic thresholding [85],
(b) treat failure as new problem [103].

Consolidation
Problem: Given a *target, source,* and evaluated analogical inferences, consolidate these to improve future performance.
Solutions: (1) Record the target and outcome [75].
(2) Record the analogical mapping [108].
(3) Record an inductive summary:
(a) induce rules [114],
(b) induce plan schemata [18].
(4) Learn from failures:
(a) record the failure to anticipate it later [75].
(b) record the failure remediation [103],
(c) refine analogy mechanisms [18].

the preceding comparative analysis, these processes are strongly interdependent. In contrast, recognizing candidate analogs and consolidating information generated during their use have received less attention. However, differing approaches are also evident: alternative indexing schemes for organizing candidate sources constrain recognition, while a variety of analogically derived materials are learned during consolidation. Juxtaposed, these processes present a basic tension: recognizing analogies anticipates plausible inductive summari-

zation, while consolidating confirmed analogical mappings attempts to predict future contexts of use.

Rather than solving basic problems in analogical reasoning, these approaches offer partial solutions, proposals, or refinements of larger problems. For example, replaying justifications for plan-level analogical inferences [10, 20, 62] refines the problem of elaborating and evaluating inferences at one level into a comparable problem at a lower level. For each analogy process component, the most ambitious and possibly most promising computational approaches have yet to be fully developed, implemented, or tested. Instead, implementations usually demonstrate carefully crafted solutions to isolated problems. Problems of scale and generality apply almost uniformly across the reviewed studies. This is less a criticism than an invitation to further analytical and empirical work.

8.2. Prospectus: Analogy in a computational adolescence

One explanation for the adolescence of computational research on analogy is that analogical processes are second-order phenomena which depend on a detailed understanding of first-order phenomena for a comparative demonstration of progress. For example, it is difficult to adequately specify a theory of analogical problem solving without having a robust theory of the problem solving activities that analogy will use. Many of the reviewed studies adopt this view, embedding analogy processes within more traditional problem solving frameworks [18, 20, 41, 66, 74, 85, 93]. To the extent thai these first-order frameworks provide a plausible basis for comparison, we can ask whether analogy provides solutions to problems that could not otherwise be solved or that could not be solved efficiently within the first order framework. Although these kinds of comparisons are sensitive to how source and target domains are encoded, some initial results are encouraging. For example, Greiner [41] demonstrates that analogically reusing abstractions (or constituent relations) outperforms conventional proof techniques and that a variety of additional heuristic constraints on analogy further improve performance.

Another explanation is that computational approaches to analogy are simply in the middle of a healthy adolescence: solutions to difficult problems mature slowly. As mentioned in the introduction, studies of analogy have a long history in other disciplines. Recognition of analogies poses challenging questions about memory organization and use, whether by machines or human reasoners. Elaboration and evaluation of a candidate analogy must focus attention among the myriad details of stored experience and control problem solving resources that use this experience. From both pragmatic and theoretical perspectives, it is important to avoid trading the costs of "weak methods" against a comparable or greater burden of determining the applicability of everything a reasoner knows when facing a new problem. Making this tradeoff is the essential problem of analogy: ". . . the means by which we bring our past to bear on our future so as to permit us to profit by our experience without

being saddled with it" [101, p. 111]. Consolidating the results of analogical reasoning returns to problems of memory organization and questions how we can learn anything that is genuinely new, short of rote-memorization [91], These are difficult but exciting problems in the study of intelligence. Although computational approaches to analogy are relative newcomers, they offer a fresh empirical methodology to existing perspectives and promise both theoretical and practical contributions of their own.

Appendix A. Contributions of individual studies to processes of recognition and elaboration

Study	Recognition	Elaboration
Evans [28] ANALOGY	given	object type and rule components restrict object and rule mappings; relational mapping enabled by a fixed vocabulary of substitutions
Becker [9] JCM	associative memory of schemata indexed by generic concepts	kernel predicates map identically; node mismatch weighted by a linear combination of salience estimates; best candidate maximizes matcli score and schema confidence, minimizes cost
Kling [66] ZORBA	given	partial mapping over theorem statements is extended by a type-restricted best-first search
Munyer [85]	source formulas and derivations indexed by functional containment	extends unification through bottom-up, competitive reinforcement of a global mapping; requires consistent mappings at boundaries of the source derivation
Greiner [43] NLAO	given a hint, generate a source problem and find an abstraction for solving it	find a target instance of the abstraction, inferring residual conjectures as needed
Brown [11]	source domain given by tutorial context	incrementally extend a type-restricted mapping; map target problem into a source problem and solve it
McDermott ANA *[IS]*	manipulate target cues to trigger method indices; anticipate method failures	map source into existing or "stipulated" target objects; extend mapping to cover method subgoals
Carboncll [18] ARIES	solution paths are indexed by states and constraints; a similarity metric screens candidate source sequences	map source and target sequences
Carboncll [2()|	similar initial reasoning triggers retrieval of a source derivation from dynamic memory	map source and target derivations
Simpson [103] MEDIATOR	collect source rcmindings by traversing an episodic memory for plans and failures; select a source candidate that preserves an invariance hierarchy	mapping is distributed across index tests; align identical norm slols

Study	Recognition	Elaboration
Winston 1108) FOX	a simile is given by a tutor or conjectured by the learner	prefer existing transfer frames, salient properties of the source, properties that are prototypical of the target class, or properties that continue the instructional context
Hobbs [51] DIANA	a source schema is triggered by the appearance of target predicates	resolution of discourse problems aligns source and target concepts
Dyer [26] BORRIS MORRIS	target planning failures trigger retrieval of a source narrative from dynamic memory	map narrative elements through a common planning abstraction (TAU)
Winston [114] MACBETH	recognize source by botlom-up voting through exhaustive type indexing; index rules by actors, acts, and objects	map a priori important relations before lower level relations
Burstein [15] CARL	given source domain, a target example and reasoning context triggers retrieval of a source abstraction from dynamic memory	top-down relational mapping includes objects only as needed; can map non-identical relations
Pirolli [93]	selects prior example under production control; spreading activation retrieves declarative memory structures	mapping productions (some are learned) align code template components
Kedar-Cabelli [62]	select a source instance and plan (not described); explain why the source instance satisfies the purpose	map the source explanation and plan to the target instance

Appendix 15. Contributions of individual studies to processes of evaluation and consolidation

Study	Evaluation	Consolidation
Evans [28] ANALOGY	drop unmatched relations and prefer candidates that preserve most source relations (A:B)	proposes rule proceduralization and generalization
Becker [9] JCM	unmapped source kernels are treated as subgoals to be confirmed	store (he target interpretation; weight adjustments introduce variables, drop conditions, and estimate a schema's worth
Kling [66] ZORBA	pass target clauses to a separate resolution theorem prover	none
Munyer [85]	implicit planning uses analogy as an evaluation function; explicit planning detects skewed analogies and finds plan repair steps; both succeed when a logically valid derivation is found	record the target derivation; generalize formulas and derivational sequences; delete redundant or unsuccessful derivations
Greincr [43] NLAG	verify that inferred facts solve the target problem, are consistent with existing knowledge, and are acceptable to the user	add inferred facts to the domain theory
Brown [11]	lift the source solution into the target domain; confirm transferred justifications, patching "bugs" as necessary	add successful descriptions, plans, justifications, and code to the target domain repertoire

Study	Evaluation	Consolidation
McDermolt ΛΝΛ[75]	environment and user feedback confirms method expectations or signals method errors; repair by subgoaling on failure or selecting an alternate method	builds error detection and recovery productions; adds target instantiation of successful method productions
Carbonell [18] ARIES	MEA in T-space reduces differences between source and target sequences	store target solution sequence; generalize operator sequences; tune similarity metric and difference table; cluster over T-space failures
Carbonell [20]	check justifications for source steps in target description; reconsider alternative source steps or prior failures; monitor a perseverance threshold	update case memory with target solution; generalize target and source traces; justification points identify instances for learning search heuristics; decompose traces into general plan components
Simpson [103] MEDIATOR	verify the source classification and plan preconditions in target; confirm plan predictions wilh user; remediate failures	memory update installs, inserts, and generalizes episodic structures
Winston [108] FOX	check for violations of known target properties, confirm existing justification frames, or ask the tutor	transfer properties; construct transfer, justification, and typical-instance frames; conjecture additional properties or new similes
Hobbs [51] DIANA	preserve contextual coherence and satisfy pragmatic constraints	target schemata arc extended; metaphors tire and die with repeated use
Dyer [26] find HORRIS, MORRIS	a coherent interpretation of the; target narrative; confirm plan-based predictions	augment the target interpretation; reorganize existing memory structures to maintain discriminable access to the target narrative
Winston [114] MACBETH	analogical inferences are confirmed directly, by further analogies, by abductive inference, or by the teacher	record the target case; build general inference rules; augment rules to censor exceptions; index original case with rule
Burstein [15] CARL	discard unsupported inferences; tutor gives feedback and corrections; multiple analogies correct misconceptions	integrate multiple causal abstractions in the target domain
Pirolli [93]	check inferred components against the target specification; test target code in the LISP environment; repair or abandon the current analogy	proceduralization and composition build general productions which supplant structural analogies
Kedar-Cabelli [62]	justify explanatory inferences for target; replace structural attributes or plan steps as necessary	find a common explanatory structure for the refined concept

ACKNOWLEDGMENT

This paper has improved through the editorial and intellectual challenges of many people. Dennis Kibler, Eticnne Wenger, Dan Easterlin, David Aha, Chris Truxaw, and Doug Fisher of the Irvine Computational Intelligence Project all commented on earlier drafts. Karen Wieckert, also at Irvine, patiently helped to clarify pages of unnecessarily complicated text and to sharpen many comparative distinctions. Members of the Cognition and Language Laboratory at the University of Illinois at Urbana-Champaign provided hours of fruitful discussion and include: Dedre Centner, Bob Schuinnkcr, Mike Jeziorski, Janice Skorstad, Brian Falkenhainer, and Mary Jo Ratterman.

Two journal reviewers also suggested critical reorganizations and extensions. I remain responsible for any residual confusion or inaccuracy. This work was supported by the Personnel Training and Research Division of the Office of Naval Research, contract N00014-85-K-0373, a gift from the Hughes Artificial Intelligence Research Laboratory, and a Regent's Dissertation Fellowship from the University of California.

REFERENCES

1. Anderson, J.R., Acquisition of proof skills in geometry, in: R.S. Michalski, J.G. Carbonell and T.M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach* (Tioga, Palo Alto, CA, 1983) 191-220.
2. Anderson, J.R., *The Architecture of Cognition* (Harvard University Press, Cambridge, MA, 1983).
3. Anderson, J.R., Farrell, R. and Sauers, R., Learning to program in LISP, *Cognitive Sci.* 8 (1984) 87-129.
4. Anderson, J.R., Boyle, C.F. and Reiser, B.J., Intelligent tutoring systems, *Science* 228 (1985) 456-462.
5. Anderson, J.R., Knowledge composition: The general learning mechanism, in: R.S. Michalski, J.G. Carbonell and T.M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach* (Morgan Kaufmann, Los Altos, CA, 1986) 289-310.
6. August, S.E. and Dyer, M.G. Understanding analogies in editorials, in: *Proceedings Seventh Annual Conference of the Cognitive Science Society,* Irvine, CA (1985) 845-847.
7. Bayman, P. and Mayer, R.E., A diagnosis of beginning programmers' misconceptions of BASIC programming statements, *Cominun. ACM* 26 (9) (1983) 677-679.
8. Becker, J.D., The modeling of simple analogic and inductive processes in a semantic memory system, in: *Proceedings UCA1-69,* Washington, DC (1969) 655-668.
9. Becker, J.D., A model for the encoding of experiential information, in: R.C. Schank and K.M. Colby (Eds.), *Computer Models of Thought and Language* (Freeman, San Francisco, CA, 1973) 396-435.
10. Brown, A.L. and Campione, J.C., Three faces of transfer: Implications for early competence, individual differences, and instruction, in: M. Lamb, A.L. Brown and B. Rogoff (Eds.), *Advances in Developmental Psychology* (Erlbaum, Hillsdale, NJ, 1985).
11. Brown, R., Use of analogy to achieve new expertise, Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA (1977).
J2. Brown, R., *Words and Things* (Free Press, Glencoe, IL, 1958).
13. Burstein, M.H., Concept formation through the interaction of multiple models, in: *Proceedings Third Annual Conference of the Cognitive Science Society,* Berkeley, CA (1981) 271-273.
14. Burstein, M.H., Concept formation by incremental analogical reasoning and debugging, in: *Proceedings International Machine Learning Workshop,* Monticello, IL (1983) 19-25.
15. Burstein, M.H., Concept formation by incremental analogical reasoning and debugging, in: R.S. Michalski, J.G. Carbonell and T.M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach* (Morgan Kaufmann, Los Altos, CA, 1986) 351-370.
16. Carbonell, J.G., Invariancc hierarchies in metaphor interpretation, *Proceedings Third Annual Meeting of the Cognitive Science Society,* Berkeley, CA (1981) 292-295.
17. Carbonell, J.G., Metaphor: An inescapable phenomenon in natural-language comprehension, in: W.G. Lehnert and M.H. Ringle (Eds.), *Siategies for Natural Language Processing* (Erlbaum, Hilldale, NJ, 1982) 415-435.
18. Carbonell, J.G., Learning by analogy: Formulating and generalizing plans from past experience, in: R.S. Michalski, J.G. Carbonell and T.M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach* (Tioga, Palo Alto, CA, 1983) 137-162.
19. Carbonell, J.G., Derivational analogy in problem solving and knowledge acquisition, in: *Proceedings International Machine Learning Workshop,* Monticello, IL (1983) 12-18.

20. Carbonell, J.G., Derivational analogy: A theory of reconstructive problem solving and expertise acquisition, in: R.S. Michalski, J.G. Carbonell and T.M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach* (Morgan Kaufmann, Los Altos, CA, 1986) 371-392.
21. Carbonell, J.G., Michalski, R.S. and Mitchell, T.M., An overview of machine learning, in: R.S. Michalski, J.G. Carbonell and T.M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach* (Tioga, Palo Alto, CA, 1983) 3-24.
.22. Carrol, .l.M. and Mack, R.L., Metaphor, computing systems, and active learning, RC-9636(42545), IBM Watson Research Center, Yorktown Heights, NY (1982).
23. Clemen', J., Observed methods for generating analogies in scientific problem solving, in: *Proceedings Annual Meeting of lite American Educational Research Association,* Montreal, One. (1983).
24. Dershowitz, N., Programming by analogy, in: R.S. Michalski, J.G. Carbonell and T.M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach* (Morgan Kaufmann, Los Altos, CA, 1986) 395-423.
25. Dietteiich, T.G., Learning and inductive inference, in: P.R. Cohen and E.A. Feigenhaum (Eds.), *The Handbook of Artificial Intelligence* VIII (William Kaufmann, Los Altos, CA, 1982) 323-512.
26. Dyer, M.G., *In-Dcplli Understanding* (MIT Press, Cambridge, MA, 1983).
27. Dyer, M.G., Understanding stories through morals and remindings, in: *Proceedings IJCAI-83,* Karlsruhe, F.R.G. (1983) 75-77.
28. Evans, T.G., A program for the solution of a class of geometric analogy intelligence test questions, in: M. Minsky (Ed.), *Semantic Information Processing* (MIT Press, Cambridge, MA, 1968) 271-353.
29. Falkenhainer, B., Forbus, K.D. and Centner, D., The structure-mapping engine, in: *Proceedings AAAI-86,* Philadelphia, PA (1986) 272-277.
30. Falkenhainer, B., Forbus, K.D. and Centner, D., The structure-mapping engine: Algorithm and examples, Tech. Rcpt. UIUCDCS-R-87-1361, Department of Computer Science, University of Illinois at Urbana-Cliampaign, IL (1987).
31. Fisher, D. and Langley, P., Approaches to conceptual clustering, in: *Proceedings UCAI-S5,* Los Angeles, CA (1985) 691-697.
32. Forbus, K.D. and Centner, D., Learning physical domains: Toward a theoretical framework, in: R.S. Michalski, J.G. Carbonell and T.M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach* (Morgan Kaufmann, Los Altos, CA, 1986) 311-348.
33. Gentner, D., Are scientific analogies metaphors? in: D. Miall (Ed.), *Metaphor: Problems and Perspectives* (Harvester, Brighton, England, 1982) 106-132.
34. Centner, D., Structure-mapping: A theoretical framework for analogy, *Cognitive Sci.* 7 (1983) 155-170.
35. Gentner, D., Analogical inference and analogical access, in: A. Priedilis (Ed.), *Analogica: The first Workshop on Analogical Reasoning* (Pitman, London, 1987).
36. Gentner, D., Mechanisms of analogical learning, Tech. Rept. U1UCDCS-R-87-1381, Department of Computer Science, University of Illinois at Urbana-Champaign, IL (1987) also in: S. Vosniadou and A. Ortony (Eds.), *Similarity and Analogical Reasoning* (Cambridge University Press, London, to appear).
37. Georgcff, M.P., Strategies in heuristic search, *Artificial Intelligence* 20 (1983) 393-425.
38. Gick, M.L. and Holyoak, K.J., Analogical problem solving, *Cognitive Psycho!.* 12 (1980) 306-355.
39. Gick, M.L. and Holyoak, K.J., Schema induction and analogical transfer, *Cognitive Psychol.* 15 (1983) 1-38.
4(1. Green, C., Theorem proving by resolution as a basis for question answering systems, in: D. Michie and B. Meltzer (Eds.), *Machine Intelligence* 4 (Edinburgh University Press, Edinburgh, 1969) 183-205.

41. Greiner, R., Learning by understanding analogies, Ph.D. Thesis, Tech. Kept. STAN-CS-85-1071, Stanford University, Stanford, CA (1985).
42. Greiner, R., Learning by understanding analogies, *Artificial Intelligence* 35 (1988) 81-125.
43. Greiner, R., Abstraction-based analogical inference, in: D.I1. Helinan (Ed.), *Analogical Reasoning: Perspectives of Artificial Intelligence, Cognitive Science, anil Philosophy* (Reidcl, Dordrecht, The Netherlands, 1988).
44. Grudin, J., Processes in verbal analogy solution, *J. Experimental Psycho/. Unman Perception and Performance 6* (1980) 67-74.
45. Halasz, F. and Moran, T.P. Analogy considered harmful, in: *Proceedings Conference on Human Factors in Computer Systems,* Gaithersburg, MD (1982) 33-36.
46. Hall, R.P., Understanding analogical reasoning: Viewpoints from psychology and related disciplines, UC1-TR-86-10, Department of Information and Computer Science, University of California at Irvine, CA (1986).
47. Hammond, K.J., CHEF: A model of case-based planning, in: *Proceedings AAAI-86,* Philadelphia, PA (1986) 267-271.
48. Hayes-Roth, F., The role of partial and best matches in knowledge systems, in: D.A. Waterman and F. Hayes-Roth (Eds.), *Pattern-Directed Inference Systems* (Academic Press, New York, 1978) 557-576.
49. Herstein, I.N., *Topics in algebra* (Blaisdell, Waltham, MA, 1964).
50. Hesse, M.B., Models and analogies in science, in: M.A. Hoskin (Ed.), *Newman History and Philosophy of Science Series* 14 (Sheed and Ward, London, 1963).
51. Hobbs, J.R., Metaphor interpretation as selective inferencing: Cognitive processes in understanding metaphor (Part 1), *Empirical Studies of the Arts* 1(1) (1983) 17-33.
52. Hobbs, J.R., Metaphor interpretation as selective inferencing: Cognitive processes in understanding metaphor (Part 2), *Empirical Studies of the Arts* 1 (2) (1983) 125-142.
53. Hofstadter, D.R., Mitchell, M. and French, R.M , Fluid concepts and creative analogies: A theory and its computer implementation, Tech. Rept. CSM1L-H), Fluid Analogies Research Group, University of Michigan at Ann Arbor, Ml (1987).
54. Holyoak, K.J., Analogical thinking and human intelligence, in: R.J. Steinberg (Ed.), *Advances in the Psychology of Human Intelligence* 2 (Erlbaum, Hillsdale, NJ, 1984) 199-230.
55. Holyoak, K.J., The pragmatics of analogical transfer, *Psychol. Learning and Motivation* 19 (1985) 59-87.
56. Honeck, R.P. and Hoffman, R.R. (Eds.), *Cognition and Figurative Language* (Erlbaum, Hillsdale, NJ, 1980) 25-46.
57. Hunt, E., Quote the raven? Nevermore! in: L.W. Gregg (Ed.), *Knowledge and Cognition* (Erlbaum, Hillsdale, NJ, 1973).
58. Hutchins, E., Metaphors for interface design, Rept. 8703, Institute for Cognitive Science, University of California at San Diego, CA (1987).
59. Indurkhya, B., *Analogies and Metaphors: A Computational Theory* (MIT Press, Cambridge, MA, 1988).
60. Johnson, M. (Ed.), *Philosophical Perspectives on Metaphor* (University of Minnesota Press, Minneapolis, MN, 1981).
61. Kedar-Cabelli, S., Analogy with purpose in legal reasoning from precedents: A dissertation proposal, LRP-TR-17, Department of Computer Science, Rutgers University, New Brunswick, NJ (1984).
62. Kcdar-Cabelli, S., Purpose-directed analogy, in: *Proceedings Seventh Annual Conference of the Cognitive Science Society,* Irvine, CA (1985) 150-159.
63. Kedar-Cabelli, S.T., Analogy: From a unified perspective, in: D.I I. Hclman (Ed.), *Analogical Reasoning: Perspectives of Artificial Intelligence, Cognitive Science, and Philosophy* (Reidel, Dordrecht, The Netherlands, 1988).
64. Kedar-Cabelli, S.T. and McCarly, L.T., Explanation-based generalization as resolution theorem proving, in: *Proceedings Fourth International Workshop on Machine Learning,* Irvine, CA (1987) 383-389.

65. Kling, R.E., Reasoning by analogy wilh applications to heuristic problem solving: A case study, I'h.D. Thesis, Tech. Rept. CS-216, Stanford University, CA (1971).
66. Kling, R.E., A paradigm for reasoning by analogy. *Artificial Intelligence 2* (1971) 147-178.
67. Kolodner, J.L., Maintaining organization in a dynamic long-term memory, *Cognitive Sci.* 7 (1983) 243-280.
68. Kolodner, J.L., Reconstructive memory: A computer model, *Cognitive Sci.* 7 (1983) 281-328.
69. Kolodner, J.L., Simpson, R.L., Sycara-Cyranski, K., A process model of cased-bascd reasoning in problem solving, in: *Proceedings IJCAI-85,* Los Angeles, CA (1985) 284-290.
70. Laird, J.E., Newell, A. and Rosenbloom, P.S., SOAR: An architecture for general intelligence, *Artificial Intelligence* 33 (1987) 1-64.
71. Lakcoff, G. and Johnson, M., *Metaphors We Live By* (Chicago University Press, Chicago, IL, 1980).
72. Lebowitz, M., Correcting erroneous generalizations. *Cognition anil Drain Theory S* (4) (1982) 367-381.
73. Malgady, R.G. and Johnson, M.G., Measurement of figurative language: Semantic feature models of comprehension and appreciation, R.P. Honeck and R.R. Hoffman (Eds.), *Cognition and Figurative Language* (Erlbaum, Hillsdale, NJ, 1980) 239-258.
74. McDermott, J., ANA: An assimilating and accommodating production system, CMU-CS-78-156, Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA (1978).
75. McDermotl, J., Learning to use analogies, in: *Proceedings IJCAI-79,* Tokyo (1979) 568-576.
76. Michalski, R.S., A theory and methodology of inductive learning, in: R.S. Michalski, J.G. Carbonell and T.M. Mitchell (Eels.), *Machine Learning: An Artificial Intelligence Approach* (Tioga, Palo Alto, CA, 1983) 83-134.
77. Michalski, R.S., Understanding the nature of learning: Issues and research directions, in: R.S. Michalski, J.G. Carbonell and T.M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach* (Morgan Kaufmann, Los Altos, CA, 1986) 3-26.
78. Michalski, R.S. and Stcpp, R.E., Conceptual clustering: Inventing goal-oriented classifications of structured objects, in: R.S. Michalski, J.G. Carbonell and T.M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach* (Morgan Kaufmann, Los Allos, CA, 1986) 471-498.
79. Miller, C., Analogy and mathematical reasoning: A survey, PB84-I44401, Department of Computer Studies, Leeds University, England (1983).
80. Miller, R.M., The dubious case for metaphors in educational writing, *Educational Theory 26* (1976) 174-181.
81. Minsky, M., Artificial Intelligence, *Sci. Am.* 215 (1966) 246-263.
82. Mitchell. T.M., Generalization as search, *Artificial Intelligence* 18 (1982) 203-226.
83. Mitchell, T.M., Keller, R.M. and Kedar-Cabelli, S.T., Explanation-based generalization: A unifying view, *Mach. Learning I* (1986) 47-80.
84. Mostow, J., Some requirements for effective replay of derivations, in: *Proceedings of the Third International Machine Learning Workshop,* Skytop, PA (1985) 129-132.
85. Munyer, J.C., Analogy as a means of discovery in problem solving and learning, Ph.D. Thesis, University of California at Santa Cruz, CA (1981).
86. Newell. A. and Simon, II.A., *Human Problem Solving* (Prentice-Hall, Englewood Cliffs, NJ, 1972).
87. Newell, A., The heuristic of George Polya and its relation to Al, in: R. Groncr, M. Groncr and W.I·. Bischof (Eds.), *Methods of Heuristics* (Erlbaum, Hillsdale, NJ, 1983) 195-243.
88. Novick, L.R., Analogical transfer: Processes and individual differences, in: D.H. Helman, (Ed.), *Analogical Reasoning: Perspectives of Artificial Intelligence, Cognitive Science, and Philosophy* (Reidel, Dordrecht, The Netherlands, 1988).
89. Ortony, A., Reynolds, R.E. and Arter, J.A., Metaphor: Theoretical and empirical research, *Psych'ol. Bull.* 85 (1978) 919-943.
90. Ortony, A., *Metaphor and Thought* (Cambridge University Press, London, 1979).

91. Petrie, H.G., Metaphor and learning, in: A. Ortony (Ed.), *Metaphor unit Thought* (Cambridge University Press, London, 1979) 438-461.
92. Pirelli, P.L., Anderson, J.R. and Farrell, R., Learning to program recursion, in: *Proceedings ofllie Sixth Annual Conference of the Cognitive Science Society,* Boulder, CO (1984) 277-280.
93. Pirolli, P.L. and Anderson, J.R., The role of learning from examples in the acquisition of recursive programming skills, *Can. J. Psychol.* 39 (2) (1985) 240-272.
94. Polya, O., *How to Solve It* (Princeton University Press, Princeton, NJ, 1945).
95. Ouinlan, J.R., Learning efficient classification procedures and their application to chess end games, in: R.S. Michalski, J.G. Carhonell and T.M. Mitchell (Eds.), *Machine Learning: An Artificiiil Intelligence Approach* (Tioga, Palo Alto, CA, 1983) 463-482.
96. Ringle, M., Philosophy and artificial intelligence, in: M. Ringle (Ed.), *Philosophical Perspectives in Artificial Intelligence* (Humanities Press, Atlantic Heights. NJ, 1979) 1-20.
97. Rumelhart, D.E. and Ahrahamson, A.A., A model for analogical reasoning. *Cognitive Psychol.* 5 (1973) 1-28.
98. Rumelhart, D.E., McClelland, J.L. and the POP Research Group, *Parallel Distributed Processing: Explorations in the Microstntcture of Cognition* (Bradford Books, Cambridge, MA, 1986).
99. Schank, R.C., *Dynamic Memory: A Theory of Reminding and Learning in Computers and People* (Cambridge Univesity Press, London, 1982).
100. Scherlis, W.L. and Scott, D.S., First steps towards inferential programming, in: R.E.A. Mason (Ed.), *Information Processing 83* (Elsevier North-Holland, New York, 1983) 199-212.
101. Schon, D.A., *Displacement of concepts* (Tavistock Publications, London, 1963).
102. Siekmann, J.H., Unification theory, in: *Proceedings ECAI-8fi,* Brighton, England (1986) vi-xxxv.
103. Simpson, R.L., A compuler model of case-based reasoning in problem solving: An investigation in the domain of dispute mediation, Ph.D. Thesis, Tech. Kept. G1T-ICS-85/18, Georgia Institute of Technology, Atlanta, GA (1985).
104. Sternberg, R.J., *Intelligence, Information Processing and Analogical Reasoning: The Coinponential Analysis of Human Abilities* (Erlbaum, Hillsdale, NJ, 1977).
105. Wengcr, E., *Artificial Intelligence and Tutoring Systems: Computational and Cognitive Approaches to the Communication of Knowledge* (Morgan Kaufmann, Los Altos, CA, 1987).
106. Wile, D.S., Program developments: Formal explanations of implementations, *Commun. ACM 26* (11) (1983) 902-911.
107. Winston, P.H., Learning structural descriptions from examples, in: P.M. Winston (Ed.), *The Psychology of Compuler Vision* (McGraw-Hill, New York, 1975).
108. Winston, P.H., Learning by creating and justifying transfer frames. *Artificial Intelligence* 10 (2) (1978) 147-172.
109. Winston, PH., Learning and reasoning by analogy, *Commun. ACM* 23 (12) (1980) 6f{9-703.
110. Winston, P.H., Learning new principles from precedents and exercises, *Artificial Intelligence* **19** (1982) 321-350.
111. Winston, P.H. Learning by augmenting rules and accumulating censors, in: *Proceedings International Machine Learning Workshop,* Monlicello, IL (1983) 2-11.
112. Winston, PH., Binford, T.O., Katz, B. and Lowry, M., Learning physical descriptions from functional definitions, in: *Proceedings AAAI-83,* Washington, DC (1983) 433-439.
113. Winston, P.M., *Artificial Intelligence* (Addison-Wesley, Reading, MΛ, 1984).
114. Winston, P.H. Learning by augmenting rules and accumulating censors, in: R.S. Micliul.ski, J.G. Carbonell and T.M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach* (Morgan Kaufmann, Los Altos, CA, 1986) 45-61.

Received November 1986; revised version received December 1987