Analogy in Problem Solving

Erica Melis* Manuela Veloso†

1 Introduction

When Konrad Lorenz was awarded the Nobel Prize for medicine in 1973 he delivered the lecture "Analogy as a Source of Knowledge" and acknowledged that "...this procedure (analogical reasoning) has led me to the discovery which I personally consider to be my own most important contribution to science" [90]. So it is worth to ask what is this powerful procedure called analogy?

A first answer is: analogical reasoning infers an aspect of a *target* object from knowledge about a *source* object that is somehow similar to the target. In particular, problem solving by analogy infers the solution of a target problem from knowledge about the solution to a source that is similar to the target.

Analogy seems rampant in all domains of human affairs, political, legal, engineering, and so on (for examples see, e.g., [64, 76]). Analogical reasoning is an important source of practical reasoning and human reasoning is replete of analogies: From their use in everyday life to analogies in mathematics and scientific discovery. Here are some examples.

Everyday life: If one wants to buy a used car (the target) and to estimate its price it may well happen that the price of another car (a source) with a similar make and age guides one's estimation. Many examples can be found in [123].

Engineering: A new technical object (the target) is to be designed such that it can perform a given physical function. Its physical structure is analogically inferred from a biological object (a source) whose function is similar to the technical function aimed at. As experience shows, there is a general connection between the function and the structure of a system. The analogical inference is thus based on this connection and works by searching for biological analogs that are similar to the target w.r.t. their function and then inferring the physical structure of the target from the structure of the biological analogs. See [20] for the more detailed *bionics* procedure.

Avionics engineers who were analyzing components for a new aircraft explicitly employed analogies to functionally similar components of older aircrafts [77]. More examples for the use of analogy in engineering can be found, for instance, in [124, 128].

Mathematics: Polya [112] saw analogical thinking as an absolute necessity for mathematical creativity. One of Polya's well known heuristics is: *Have you seen a similar problem before? Try to use its solution.* More reports of analogical reasoning in maths can be found in [113, 51].

*Universität des Saarlandes, Fachbereich Informatik, D-66041 Saarbrücken, Germany
†School of Computer Science, Carnegie Mellon University, Pittsburgh, U.S.A.

Science: Rutherford's famous solar system model of the atom (see, e.g.,[56]) was used to make predictions about the allowable energy levels of atoms. When Lavoisier developed his oxygen theory of combustion[1], his thinking employed the similarity between the combustion of inorganic substances and the air transformation in lungs. Both of these processes involve a change of oxygen into carbon dioxide and a provision of heat. Franklin's analogical reasoning from a Leyden Bottle phenomenon to lightning led him to the hypothesis that lightning was a form of electricity. More examples for the success of analogical reasoning in science can be found in [10, 56, 64].

Analogical reasoning is central to problem solving, learning, and creativity (see, e.g., [40, 80, 88]). Since it has been thought to be essential to creative human behavior, many IQ tests monitor the ability to draw analogies. Indeed, human reasoning by analogy is particularly valuable in ill-defined, complex, or new situations because often it is easier to adapt a source such that it fits the conditions of the target than to solve a problem from first principles. For complex problems the search space of solutions can be huge and without background knowledge restricting the search considerably a complete search would require too much time or memory resources. For ill-defined and new situations some knowledge is even missing for a solution from first principles. Problem solving by analogy is a heuristic that responds to resource limitations by solving a target problem guided by the solution of a source problem.

Given the wide range of situations in which humans use reasoning by analogy successfully, it is not surprising that many psychological investigations have been conducted about how humans reason or solve problems by analogy. Similarly, it has been a dream from the early days of Artificial Intelligence (AI) to build systems able to reason by analogy. Evan's ANALOGY program [35], for instance, dealt with geometric proportional analogy problems such as the one in Figure 1.

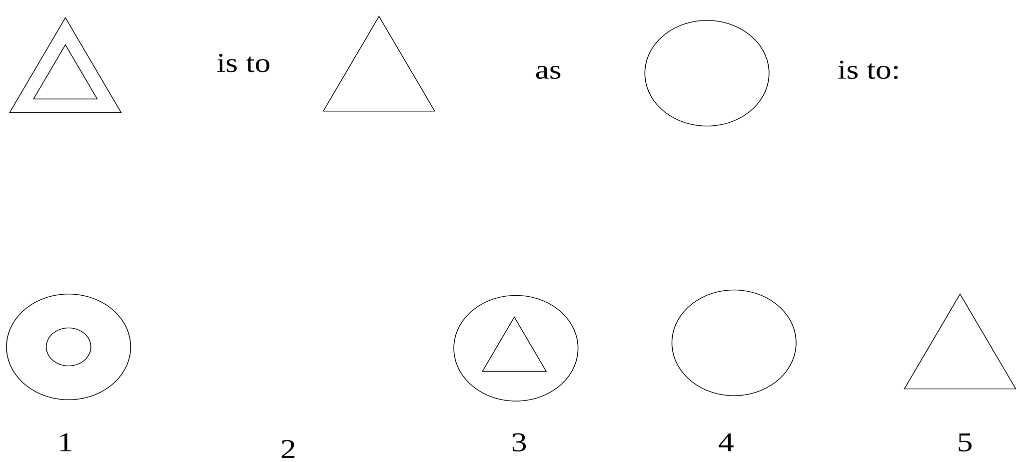


Figure 1: A problem solved by Evan's ANALOGY

Simulating human "intelligent" behavior greatly motivated much of the analogy research in AI with the ultimate goal of flexible and powerful computational systems. To understand how human analogical reasoning works was another motivation for building computer

[1] as opposed to the Phlogiston theory

programs that simulate human analogical reasoning. Recently, the use of computational analogy in real world applications has become more and more important and has the objective to reduce the search in *complex* problem solving, to propose solutions in domains that are ill-defined and not well understood in terms of rules and domain theories, and to avoid past mistakes to be repeated in problem solving. The Artificial Intelligence (AI) systems for analogy vary considerably in the employed mechanisms. Some systems emphasize an elaborate memory design and retrieval, some stress the construction of mappings, some allow for multiple sources for the analogical replay, some focus on justification of analogical reasoning. Some work with connectionist representations, some rule-based, some by looking up case-bases. We shall present several classes of systems in this article.

Computational analogy is, in general, far from reaching human performance and it still is extremely dependent on the actual representation of the source and the target problem. To date analogy remained a research area with many open problems and no commonly agreed theory. Even the term "analogy" occurs in different contexts, with different meanings, and even as a synonym for "similarity" or for "metaphor."

The tasks to be solved by analogy systems are manifold and range from problem solving and conjecturing[2] to learning and teaching, or natural language understanding. We shall concentrate on approaches to problem solving by analogy which can be roughly defined as using a known solution to a source problem to aid in the generation of a solution to a similar target problem, or using previous problem solving experience for a target problem.

Reader's guide

The article has four more sections dealing with a broad view of analogy, with the definition of different types of analogy, with computational accounts to problem solving by analogy, and with theorem proving by analogy.

The next section, analogy at large, includes a very brief review of some contributions from philosophy and a glance at empirical results and models of analogy in cognitive psychology that may influence the research in Artificial Intelligence.

Section 3 provides a theoretical framework for characterizing different types of analogical inference. As opposed to other overviews of analogy research (e.g., [52]), we emphasize relevance and similarity as key ingredients of justified analogical reasoning rather than focusing on the subprocesses of retrieval, mapping, elaboration, and evaluation here. This emphasis gives rise to several types of analogical reasoning. Few attempts have been made to formalize such an inference framework and, in particular, actually implemented systems do not explicitly refer to such a framework. We think, however, that this can be important for future systems that employ *justified* analogy.

Section 4 presents different paradigms of computational problem solving by analogy and their realization along with the purpose of the analogy approach, the domains tackled, the knowledge representation used, the top level algorithms, and particular computational mechanisms employed. We do not consider approaches to proportional analogy, metaphor, learning, and exact reuse. Finally, an approach to theorem proving by analogy is given in more detail.

Each section is quite self contained which allows the focused reader to select the topic of his or her interest.

[2]Conjecturing can include proportional analogies problems which have the form A : B = C : ?.

2 Analogy at Large

While AI has participated in the analogy research during the last decades, philosophy, linguistics, and most notably psychology have a longer tradition in investigating analogy. B Therefore, before examining computational approaches to analogy, it is appropriate to lean back and ask what did other disciplines contribute to analogy approaches of AI. Some answers are:

- cognitive memory models underlying analogical reasoning,
- models of subprocesses establishing analogical reasoning,
- studies of the importance and prevalence of analogical reasoning in human thinking, particularly in science, and
- investigations of the role that analogical reasoning plays in human learning and problem solving.

2.1 Contributions from Philosophy

Philosophical arguments tried to provide *conceptual frameworks*, to create definitions for analogical reasoning and to find justifications for analogical inferences. Philosophers have particularly analyzed analogical reasoning in science and have shown the importance of analogy in the history of science.

From the time of the ancient Greeks through the twentieth century, analogy has been subject to philosophical arguments. Throughout the history, analogy was taken by rhetoric and linguistic analysis and, since analogical inference was actually used in science, formulators of the concept of scientific method, such as Bacon [5], regarded analogy as a rational mode. Aristotle [4] informally defined reasoning by analogy. He divided the analogy process into two steps: Single-instance induction yielding a rule and the deductive application of the resulting rule to the target. It is beyond the scope of this article to enumerate all the contributions from philosophy. So we focus on some recent ones.

Mill [101] proposed a similarity measure as a probabilistic justification of his analogy rule. He suggests that analogical inferences are likely to be true if source and target agree in some properties and if the inferred fact is caused by some of these properties.

Ujomow [137] characterizes reasoning by analogy as a transfer of information from a source to a target in case a reason for the transfer exists. He postulates several conditions for the plausibility and reliability of analogical inferences, among them the relevance of the transferred properties. Another result of his work is a classification of analogies into structural – functional, functional – structural, cause – effect analogies which means that the analogical reasoning infers similar functions of two objects from similar structures of the objects etc. He also describes differences between reasoning by analogy and by induction.

Weitzenfeld [150] assumes that the existing similarities between a source and a target determine the inferred similarities and offers "determining structures" as background knowledge justifying analogical inferences. He considers a thermodynamic system a paradigmatic case of a determining structure because the thermodynamic properties will always be related in a known way. Weitzenfeld describes analogical inference as a deductive process that assumes the premise of the determining structure. Consequently, the 'extra-logical' force of that inference comes from this assumed premise that cannot be deduced from a background theory.

Indurkhya [69] analyzes metaphors and several modes of analogical reasoning: analogy by rendition, proportional analogy, and predictive analogy. In [68] he introduces the constraint semantic transference which restricts analogical inferences to those consistent with a given set of constraints (including the already known facts). His analogical mappings are defined as homomorphisms between algebras.

Hesse [56] investigated the analogies between scientific models with their explicandum. She differentiates the properties common to the model and its explicandum (positive analogy), the properties not common (negative analogy), and neutral properties for which we do not know yet whether they are common or not. She calls the problem of establishing the conditions under which analogical inference is warranted the "logical problem of analogy". Hesse suggests that in a system that is to be used as a model for an explicandum, we do not generally know the causal relations of the properties A_i with the properties B_j, we only know that the A_i are related to the and B_j as they occur together in the model. Leatherdale [89], Biela [10], and Thagard [134] also list some the important analogies in the history of biological and physical science.

2.2 Contributions from Psychology

A great deal of cognitive research has been conducted on analogy. Since the 1980s considerable cognitive research has been carried out to elucidate this form of thought. For reviews see [39, 64, 45]. Analogy is generally assumed to be a basic process in learning [110], cognitive development, and critical in problem solving [43]. The importance of analogy to mature reasoning and expertise is underscored by numerous studies of problem solving, e.g., by [28].

The contributions of psychologists to the analogy research are manifold. Different theories of analogical mapping and retrieval of analogs have been proposed that are supported by experimental data of human analogical reasoning and by computational models. Three well known cognitive models for analogical mapping are the Structure-Mapping-Engine (SME) [37], the Analogical Constraint Mapping Engine (ACME) [63], and the Incremental Analogy Machine (IAM) [72]. These models have mainly been used to explain human performance and to compare it with a system's performance.

In the following we review some selected empirical results and psychological models of analogical mapping and of memory organization. The latter widely influenced the computational approaches to source retrieval.

2.2.1 Empirical Results

Empirical investigations have proved that analogical reasoning is central to human thinking. The empirical research also established empirical evidence for certain subprocesses of analogical reasoning and showed the differences in experts' and novices' use of analogy.

Subprocesses

The earliest psychological models for analogy as well as more recent ones echo psychological theories stemming from research on analogy comprehension and problem solving, e.g., in [77, 132]. Several theories, e.g. [52, 71, 107], suggest that the following subprocesses constitute analogical reasoning:

- *Representation* of the source and the target,
- *Retrieval* of a potentially useful source,
- (Partial) *mapping* between the source and the target,
- *Elaboration* of the mapping,
- *Evaluation and adaptation* of the resulted target description,
- *Learning* new generalizations from the analogical transfer.

EXAMPLE: Faced with the problem of understanding the structure of the atom one might retrieve the solar system as a source, and then perform an analogical mapping between the two objects – the solar system and the atom [46]. By analogy to energy levels in the solar system, the energy levels of the atom can be inferred analogically. This analogous solution may not be correct, but needs to be tested and to be adapted to the reality of the target problem. Finally, as a result of the analogy some generalized knowledge, e.g., a schema encoding the general concept of central force systems might be formed (as, e.g., described in [47]). ■

As shown recently in [34], these subprocesses need not to be performed in a strict serial or independent fashion and some of the subprocesses can be missing in certain situations.[3]

Retrieval

Analogical problem solving can be initiated by an explicit hint to use a particular case as a source [47], by a reasoner's explicit decision to try to solve a difficult problem by analogy and constructing various sources by systematic transformations [25], or by spontaneous retrieval of a source from the reasoner's memory [62].

Some results in the psychological literature show that people spontaneously access sources cued mainly by the so-called "surface" features of the target [44, 62]. But to some extent the similarity of structural components also influences the spontaneous selection of an analog. Structural features have a greater impact than do surface features on a problem solver's ability to *use* a retrieved analog. Investigations that took contextual features into account have shown that the goal of the target problem plays a role in retrieval as well [126].

Gick and Holyoak [47] empirically proved that people form an "inductive schema" when they are given two or more analogous stories about similar problems and solutions (Duncker's Radiation problem [33] served as a target. [4]). They demonstrated that the availability of a

[3]Variations of the set of subprocesses have been proposed, e.g., by Sternberg [132] and Clement [28]. Sternberg suggested an early component-process model for proportional analogies (A : B = C : D) that consists of (1) encoding attributes and values for the terms of an analogy, (2) inferring rules relating A and B terms, (3) incrementally mapping, i.e., finding rules relating A and C terms, and (4) applying these rules to attributes of A, C, and D terms until a unique choice can be discriminated.

[4]In this problem a doctor must find a way to use X-rays to destroy an inoperable stomach tumor, without damaging the healthy tissue that the rays must pass through on the way to the tumor. One source problem was provided by a story in which a general needed to get a large army to a fortress located in the center of a country, in order to capture it. Because the entire army could not pass safely along any one road, the general divided the army into small groups and had each group travel along one of several roads that each led to the fortress. The groups converged simultaneously on the fortress, where the combined forces succeeded in capturing it.

schema eases the retrieval of a source.

Mapping

The subprocess analogical mapping ranges from simple role substitution by variable instantiations to very elaborate mappings. A well-known example is the mapping in the solar system–atom analogy, where *revolves* in the solar system representation is mapped to *revolves* in the atom representation as well as *planet* to *electron*. The mapping problem can be difficult if there is a large number of alternative matches to choose among. Several constraints have been proposed that reduce the complexity: structural constraints, similarity constraints, pragmatic constraints, and behavioral constraints.

Gentner [39] found that in using metaphors or analogies people prefer to map connected systems of relations governed by higher-order relations, such as *causes* or *implies*, rather than isolated predicates. Empirical results in [26] support the role of the so-called systematicity as a constraint of analogical mapping (see below). Other structural constraints experimented with are the restriction to mappings between entities of the same type in [40, 63] and structural consistency in [37]. The latter means that if the proposition $revolves(A, B)$ is mapped to $revolves(C, D)$, then the corresponding arguments of both should also be mapped, i.e., A to C and B to D. As for similarity constraints, Gentner [40] found that identical concepts are put into correspondence whereas [63] prefers the looser mapping constraint of user-supplied semantical similarity. Holyoak's and Burns' empirical results [60, 19] indicate that the information mapped in analogical reasoning is goal-dependent. Behavioral constraints, such as working memory limits and effects of background knowledge, have been supported by empirical results in [72].

We mainly reviewed the psychological work about the subprocesses retrieval and mapping because only few investigations elaborate the representation, elaboration, evaluation, and learning subprocesses. T However, the other subprocesses are included into the computational approaches presented in section 4.

Miscellaneous

Several investigations have been concerned with differences in the use of analogy depending on the expertise and on the actual situation. They indicate that people perform better in analogical reasoning when they are given explicit instruction and feedback [132]. Stein et al. [131] found that the probability that a person reasons by analogy is dependent on the problem solving situation. As for individual differences, VanLehn and Jones [140] found that ineffective learners use analogical problem solving in place of ordinary problem solving, whereas good problem solvers use analogy sparingly in order to detect and fill knowledge gaps. Schwarz [125] and Klix [79] investigate individual differences in human analogy performance and suggest that differences in the information preprocessing (in the definition of relevant features and relations) determine different analogy performances. Hickman and Larkin [57] state that subjects with a high level of expertise tend to use compiled knowledge rather than analogical reasoning in situations where compiled knowledge exists. Similarly, VanLehn and Jones [140] suggests that even good problem solvers effectively use analogy for new tasks or if no principles knowledge is available.

Several reports show that inferences to change the representation of a target may be critical for the ability to use analogies [127, 27]. An example of such a target situation has been documented in psychological studies of Duncker [33], where subjects were presented with problems which at first seem to be unsolvable given the current target representation. Only after changing the representation the subjects reasoned by analogy.

Some investigations have shed light on reasons for false analogies: misleading surface similarities can access false analogies [116] and particularly novices' reasoning by analogy is subject to false analogies because of the lack of "deep" knowledge [2].

2.2.2 Models

Models have been developed in cognitive psychology, most notably models for analogical mapping and for memory organization. As for the analogical mapping, the model of SME rivals that of ACME and of IAM which mainly differ in their mapping constraints. Gentner's SME [40] is based on an emphasis of a structural constraint. Holyoak and Thagard's ACME [63] focuses on pragmatic constraint and similarity (explained in section 4.2), and Keane's IAM [73] adds behavioral constraints.

In recent years it has become clear that all constraints are important at least for some steps in the analogical reasoning process. Hence, to some extent Gentner [41] included pragmatic constraints into her system while Holyoak and Thagard [63, 135] included structural constraints both on mapping and retrieval.

A brief review of the system SME based on Gentner's ideas is given in this section, whereas the approach of Holyoak and Thagard is discussed in more detail in section 4.2. Different cognitive memory models that have influenced AI research, in particular in case-based reasoning, are presented at the end of this section.

SME

The Structure Mapping Engine (SME) [37] computes mappings between the given predicate calculus representations of a source and a target. Higher-order relations, such as *cause*, are used in the formalization of the source and the target. Since there can be many mappings between two given representations, SME focuses only on those mappings that follow the systematicity principle. Gentner's systematicity principle is a salient part of the Structure Mapping Theory [40] widely influenced by her psychological studies on the human understanding of metaphors where she extracted features characterizing good metaphors [39]. It states that among the common relations, the relations that are mapped are just those that participate in a common system of relations: i.e., relations connected by higher-order relations. Therefore, the systematicity prefers to map sets of relations governed by higher-order relations such as *cause* rather than isolated predicates. It provides a syntactic characteristic of a mapping between two fixed and specifically tailored representations, thus it requires representations of the source and the target. This and other problems have been the core of an ongoing argument about Gentner's account.

For the mapping, only relational commonalities (of n-ary predicates with $n > 1$) count and similarities in attributes (predicates with $n = 1$) are irrelevant unless they are governed by a common higher-order relation. Furthermore, SME prefers consistent one-to-one object mappings. The SME algorithm is outlined in Table 1 which is taken from [42].

Figure 2 gives an example of analogical mappings between the heat and water domains that was produced by SME [41].

The Structure-Mapping approach has been extended and implemented in PHINEAS [36]. PHINEAS includes contextual constraints into the mapping and transfer process, it allevi-

- *Local Match Construction*: For each pair of expressions in a source and target, if their predicates are identical, create a match hypothesis (MH) expressing the possibility that they match. Corresponding arguments of items in a MH are also matched if either
    - their predicates are identical,
    - the predicates are n-ary relations with $n > 1$, or
    - they are entities.
- *Kernel Construction*: Find kernel mappings by starting at MH's which are not themselves the argument of any other. For each such MH_i, if it is structurally consistent, then it and every MH below it comprise a kernel. Otherwise, recurse on the MH's that are its arguments.
- *Structural Evaluation*: Propagate scores through the network of MH's, using argument connections to pass activation downward. This implements the systematicity preference. The score of a mapping is the sum of the scores of its MH's.
- *Merge*: Construct up to n global interpretations of a match by using a greedy algorithm to find structurally consistent combinations of kernels that maximize structural evaluation scores.

Table 1: Outline of the SME algorithm

ates the one-to-one mapping requirement of SME, and allows for mapping non-identical predicates.

```
Water domain:
CAUSE ((GREATER (PRESSURE(beaker)),
PRESSURE(vial)),
(FLOW (water, pipe, beaker, vial)))

Heat domain:
CAUSE ((GREATER (TEMP (coffee)), TEMP(ice)),
(FLOW(heat,bar, coffee, ice)))


Despite of other relations in the water domain such as
GREATER(DIAM(beaker), DIAM(vial)) the only
correspondences constructed by SME are:
heat/water,
beaker/coffee,
vial/ice,
pipe/bar,
PRESSURE/TEMP,
GREATER/GREATER,
FLOW/FLOW.
```

Figure 2: Heat-flow example for a mapping in SME

ACME

As opposed to Gentner's theory of structure mapping, Holyoak and Thagard argue for the necessary influence of the reasoner's goal in exploring analogy. They introduce similarity and pragmatic constraints. Their system ACME [63] has the components: (1) dynamical construction of a mapping network for a particular analogy problem, where the units represent mapping hypotheses and the links represent specific instantiations of the constraints, and (2) activation algorithm for settling the network in order to identify the best mapping. Like SME, ACME tries to find mappings between symbols of the source and the target representations. The information about relevance for the goal is provided by the user. ACME is explained in more detail in section 4.2.

IAM

Keane and Brayshaw describe the model of IAM that implements structural, similarity, pragmatic, and behavioral constraints using serial constraint satisfaction [73]. IAM generates a single interpretation based on a small subset of possible mappings between the two domains. The system builds up this mapping incrementally by selecting a small portion of a source domain for mapping, and then moving on to map another portion.

Memory Models

Two memory models considerably influenced the knowledge representation particularly in case-based reasoning systems: The *dynamic memory model* [122] and the *exemplar model* [117].

The basic idea of Schank's dynamic memory model is to organize cases which share similar properties under a more general structure, a memory organization packet (MOP)[5]. Kolodner further developed this model by proposing that E-MOPs, episodic MOPs, and cases being arranged together in hierarchies [82]. E-MOPs store general knowledge about a type of cases and they organize individual cases using a network of indices. The generalized episodes, such as **Diplomatic Meeting**, contain norms, cases, and indices, as shown in Figure 3. The box represents norms, that is, features common to all cases under the MOP. Triangles indicate features, e.g. topic, which discriminate between the cases of a generalized episode. Indices represent these features. An index is composed of an index name, e.g. "topic", and an index value, e.g. SALT. The values are pointers to either a specialized MOP or to cases. The memory structure is dynamic in the sense that similar parts of two case descriptions are dynamically generalized in a MOP and the cases are indexed under this MOP by their differences. Similarity is defined in terms of common features, and a source case is retrieved by pushing down the discrimination network to a case with most features in common with the target case. The entire memory is a discrimination network with generalized episodes-, index name-, index value-, and case-nodes.

The *exemplar model* was developed in the mid 1970s when the discovery of typicality effects led to a rejection of the classical view that instances of a category share necessary and sufficient attributes [117]. Instead, categories are defined extensionally by their instances and family resemblance. According to the exemplar (or family resemblance) view certain objects are thought to be categorized together because they were sufficiently similar, either

[5]Another structure, the TOPs is not discussed here.

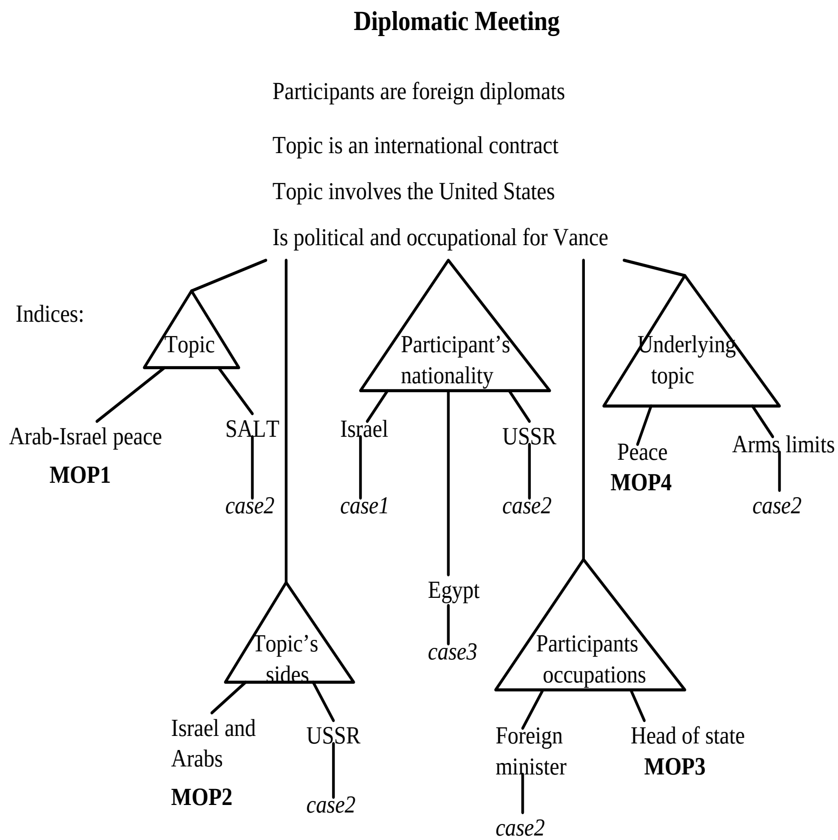


Figure 3: Structure of an E-MOP: Diplomatic Meeting

to one another or to some abstract specification of a central tendency. The categories are understood as a mental representation of central tendencies (*family resemblance* view), of instances (*exemplar* view), or combinations of the two [17, 129].

The typicality of instances of categories has been empirically investigated in psychology, e.g. in [100, 115], and reproducible typicality ratings that distinguish typical instances have been found. Some experimental methods for the extraction of this typicality rating are the direct rating of representativity, the examination of the reaction time to decide whether an instance belongs to a category, the test of the reproduction of instances, and the use of instances in generalizations and in analogical reasoning. The typicality ratings give rise to concept structures that are sets of instances partially ordered by the typicality relations $\sqsubseteq_{\mathcal{C}}$ that compare for each category $\mathcal{C}$ the degrees of typicality of similar instances (for example, a hammer is commonly considered a more typical tool than a compass saw). $\sqsubseteq_{\mathcal{C}}$ is a partial order in the set of instances, and typical instances of a concept can be defined by[6]:

DEFINITION: An instance $c \in \mathcal{C}$ is called a *typical instance* of $\mathcal{C}$ if it is maximal (that is, there is no $c' \neq c$ with $c \sqsubseteq_{\mathcal{C}} c'$), written as $\mathrm{typex}(c)$[7]. ■

With the 1980s came the growing awareness of the need to additionally include into category models information about their relations to one another and about the relevance

[6]Note that this definition does not yet consider typicality w.r.t. a certain aspect or context.
[7]The index $\mathcal{C}$ for typex will be omitted if this does not cause any confusion

relations among the attributes displayed by their instances. This led to several variants of an explanation-based view [85]. Despite these changes of the view of categories, typicality ratings are still accounted for the explanation of typicality effects and of similarities.

As for analogical reasoning, between a source and target belonging to the same category, typical instances are of some interest because they were shown to be preferred sources in analogical reasoning [115, 118].

3 An Attempt to Define Analogical Inference

In this section we explain several patterns of analogical inference that can be used in the "elaboration" subprocess. Most of the definitions in the relevant literature agree in that analogical inference transfers information from a sufficiently known *source* to an at least partially unknown *target* in the following general way:

> Analogical inference allows to infer *new* similarities between the source S and the target T, from *known* similarities between S and T.

The actual analogical inference has two steps[8], which we formulate in terms of common sense notions defined more precisely later:

(1) Inferring the new similarity of S and T w.r.t. an aspect Q (written as *similar*(S, T, Q)) from the known similarity w.r.t. an aspect P (written as *similar(S,T,P)*).

(2) Inferring the actual description of the aspect Q for the target from the known description of Q for the source and the result of step (1).

The first inference(1) is the actual analogical inference and (2) can be considered a deductive inference.

EXAMPLE: : (1) If you know that car_S and car_T are similar w.r.t. their make and their year of production, then you may infer analogically that car_S and car_T are also similar w.r.t. their price.
(2) Additionally knowing the price of car_S you can infer a likely price of car_T. ■

Analogical inference can be employed to provide an approximate description of an aspect Q of the target T, if the source and the target are somehow similar and the description of the aspect Q is known for the source S. This leads us to a first attempt towards a formalization by the following inference rules:

$$\textbf{(1)}\quad \frac{\text{similar}(S,T,P)}{\text{similar}(S,T,Q)} \qquad\qquad \textbf{(2)}\quad \frac{\text{similar}(S,T,Q),\, Q(S)}{\text{probably } Q(T)}$$

or combined

$$\frac{\dfrac{\text{similar}(S,T,P)}{\text{similar}(S,T,Q),\, Q(S)}}{\text{probably } Q(T)}$$

[8] Similarly to what Aristotle suggested.

where the form of the inference rules means that it is allowed to conclude the lower fact from the upper facts.

Additionally, the result of the analogical inference should be confirmed only if it does not contradict the knowledge inferable for the target by a deductive inference, which is silently assumed from now on. This assumption belongs to the constraints commonly accepted for analogical inference. There is also a consensus about another constraint of analogical inferences: the inference should not be possible without knowing a source. Some approaches additionally require that the results of analogical inferences must not be achievable by deductive inference [50]. We skip this constraint and allow for analogical inferences that may provide an abbreviation of a deductive inference.

Up to now it is still vague, what is meant by aspects, by similarity, and which additional similarities should be inferred from which known similarities. It is also not substantiated yet, *why* those similarities can be inferred. In the following we attempt to make these notions and reasons as precise as possible. We shall keep the very general definition of analogical reasoning, which includes cross-domain analogies, i.e., analogies for which S and T belong to different domains. Sources and targets will be called *objects* which one can imagine to be things, situations, systems, etc.

Aspects

An *aspect of an object* is a part of the object's description. one can imagine, e.g., a set of feature-value pairs, or a first order predicate calculus formula to represent an aspect for a certain object. Although analogical inference is not restricted to sources and targets represented in a formal language, we rely on this representation in order to use inference rules.[9]

Let L be a formal language in which the objects W_i are described and assume a fixed interpretation of L that interpretes the constants $w_i \in L$ by the objects W_i. An aspect is a function that takes as an argument an object W_i and returns a formula in L. An aspect function can be partially defined, i.e., may not have a value for some objects W_j. Remember that $[x/w_i]$ means that the variable x is instantiated by the constant w_i.

Definition (Aspect): For a fixed interpretation of L, an *aspect* A is defined as a partial function, mapping objects W_i to a non-tautological formula $A\langle W_i\rangle$ of L with at most one free variable[10] x, such that $A\langle W_i\rangle[x/w_i]$ is `true`. ■

Informally speaking, for an object W_i and an aspect A, $A\langle W_i\rangle$ is a formula describing A in W_i. Because of the free variable we can express that two objects W_1 and W_2 are identical w.r.t. an aspect A by writing $A\langle W_1\rangle = A\langle W_2\rangle$.

Example:
Consider the aspect safety, a car (an object) W, a constant w from L for W, and a bicycle B. Let $age(w) < 10 \wedge airbag(w) \wedge antiblock(w) \wedge max\_speed(w) < 100$ be `true` under the fixed interpretation. Then the value of $\mathsf{safety}\langle W\rangle$ could be the formula

$$age(x) < 10 \wedge airbag(x) \wedge antiblock(x) \wedge max\_speed(x) < 100$$

[9] It is well-known in AI that many other representations can be translated into some logic.

[10] Free variables are variables in a formula that are not bound by a quantifier such as $\forall$ or $\exists$. As opposed to a logical definition a common sense definition considers a "free variable" as a "generalization" from an example

with the free variable x.
For another car W_1 that has no airbag $\mathsf{safety}\langle W_1\rangle$ could be the formula

$$age(x) < 10 \wedge antiblock(x) \wedge max\_speed(x) < 100$$

because

$$airbag(w_1)$$

is not `true`.
The value of $\mathsf{safety}\langle B\rangle$ for the bicycle B could be the formula

$$frame\_diameter(x, 3) \wedge age(x) < 5$$

The examples show that the values of the aspect **safety** for different objects, say, cars, bicycles, trains, etc. can be formulas that can differ considerably.

■

The given definition of aspects is very general and has to be operationalized for computational accounts by explicitly defining the function A or by restricting aspects, e.g., to functions whose values are atomic formulas with fixed predicates. The use of aspects rather than simple features is in accordance with research that shows that analogy is not just the laying out of similarities based on shared features (e.g. [40]).

The given aspect definition is well suited for problem solving. An aspect *solution* can describe solutions that may vary in their syntactical structure and in the involved operators. The aspect *problem* that describes initial states and goals may differ in the used predicates and in the syntactic structure.

3.1 Similarity

We have to specify what should be considered as similarity between source and target. In computational systems, frequently a global numeric similarity measure is calculated based on local similarities between features from a given set of features. Different local similarity measures have been employed, among them

$$sim(a, b) = \frac{card(a \cup b) - card(a \cap b)}{card(a \cup b)}$$

where *card* is the size of a set and a, b are sets of feature values. Other measures are

$$sim(a, b) = \begin{cases} 0 & : \quad a \cap b = \emptyset \\ 1 & : \quad a \cap b \neq \emptyset \end{cases}$$

primarily for sets of symbolic feature values, and

$$sim(a, b) = \frac{|a - b|}{ul}$$

for numeric feature values a, b, where $|a - b|$ is the absolute value of the difference between a and b, and ul is the absolute value between the upper and the lower bounds of the interval of possible values of the feature.

Once the set of local similarities has been defined for each feature, they have to be combined in a global similarity measure for the objects A, B. Some well-known global measures are:

City-Block:

$$SIM(A,B) = \frac{1}{p}\sum_{i=1}^{p} sim_i(a_i, b_i),$$

where p is the number of features, sim_i the local similarity measure for the i-th feature a_i and b_i.

Euclidean:

$$SIM(A,B) = \sqrt[2]{\frac{1}{p}\sum_{i=1}^{p}(sim_i(a_i, b_i))^2}$$

Maximum:

$$SIM(A,B) = max_i \omega_i sim_i(a_i, b_i)$$

where $0 \leq \omega_i$ are relevance weights with $\sum_{i=1}^{p} \omega_i = 1$

Simple Matching Coefficient:

$$\frac{\alpha + \delta}{\alpha + \beta + \gamma + \delta}$$

where α is the number of agreeing, positive features, β is the number of inconsistent, positive features, γ the number of inconsistent negative features, and δ the number of agreeing negative features. Similarly to the latter, Tversky [136] defines similarity as a weighted sum of weighted agreeing features, of weighted features only in A, and of weighted features only in B. Gentner [40] argues, however, that this measure is too simplistic.

As opposed to the quantitative similarity measures, many approaches to analogy agree in defining similarities qualitatively as a partial identity of the source S and the target T [48, 54, 119, 144]. For the purpose of analogical inference based on aspects we define the "similarity w.r.t. an aspect P" here as

Definition (Similarity w.r.t. an aspect): For the aspect P, two objects S and T are similar w.r.t. P if $P\langle S\rangle = P\langle T\rangle$[11]. ■

As the following example shows, however, this definition of similarity alone does not sufficiently restrict the analogical inference in order to produce justified analogies.

Example: If two devices S and T agree in their `physical structure`, then it can be inferred analogically that presumably their `function` is the same as well. In combination with knowing what the function of S is like, the approximate function of T can be inferred. On the other hand, analogically inferring the `function` from a device with the same the `color` seems to be an illegitimate inference. ■

This example clearly shows that we need another ingredient for analogical reasoning, namely some kind of relevance knowledge.

[11] Remember, the truth of $P\langle W\rangle[x/w]$ under some fixed interpretation already belongs to the definition of P.

3.2 Relevance

The problem is still open, on defining for which P and Q can the additional similarity $similar(S,T,Q)$ be derived from the known similarity $similar(S,T,P)$. This problem was addressed by Hesse [56] and emphasized by Russell and others [137, 150, 30, 120, 93, 56, 24, 36, 63]. They all investigate the conditions under which an analogical inference is likely to be *justified*, the conditions that provide at least a certain probability for the analogical conclusion to be true. These conditions belong to the background knowledge used in analogical reasoning and, hence, justified analogical inference is considered knowledge-based opposed to blind similarity-based analogical inference.

Most attempts to pin down and formalize justification for analogical inferences are centered around some notion of relevance. For instance, Schank developed a theory of explanation patterns to capture the relevance for the mapping process [123]. Russell [120] presumes determinations between aspects to be given explicitly in a domain theory; Carbonell [23] assumes so-called justifications, relevant for a problem solving process that have to be checked in order to justify a problem solving by analogy. In computational systems for analogical reasoning this relevance knowledge has to be built in (implicitly), given in a domain theory, or to be explicitly provided in the problem representation or problem solving episodes. Before we discuss particular computational systems in the next section in more detail, we glance at types of knowledge justifying analogy that yield different types of analogical inference.

1. **Connection-based analogical inference**
   Connections is knowledge that encodes *some* dependency of an aspect Q from an aspect P denoted by $[P,Q]$. Connections justify analogical inferences to some extent. [12]

   If a connection $[P,Q]$ is known, then the inference schema for *justified* analogy becomes:

   $$\textbf{(CONN)} \quad \frac{\dfrac{P\langle S\rangle = P\langle T\rangle, [P,Q]}{probably(Q\langle S\rangle = Q\langle T\rangle), \quad Q\langle S\rangle(s)}}{probably(Q\langle T\rangle(t))}$$

   The degree of reliability of the justification is responsible for the probability of the agreement of $Q\langle S\rangle = Q\langle T\rangle$ and, thus, for the reliability of the result $Q\langle T\rangle(t)$. For any aspect P, $[P,P]$ and $[SYS,P]$ are trivial connections, where the latter is the connection between the overall description SYS of an object and the aspect P. These trivial connections do not carry any justifying information, and hence we exclude them in the following.

   Employing this schema for finding the description of a given goal aspect G for a target T leads to the following procedure:

   - Get a connection $[P,G]$.

[12] Some computational systems work with domain theories that explicitly contain connections, e.g., [120, 66]. Some systems, e.g., [23, 78] employ *implicitly* assumed connections, others, e.g., [37, 36, 151] use explicitly formulated connections such as RELEVANT(P,Q). Some systems [143] even generate connections at run time, e.g., by foot-printing the features relevant for a certain goal. (See section 4.) Very differently, a connection between two aspects that are represented by two sets P_N and Q_N of nodes in a neural net could be represented by exciting links between the nodes of P_N and the nodes of Q_N. But this is beyond the realm of this chapter.

- Retrieve a source S for which $P\langle S\rangle = P\langle T\rangle$. For this S it is likely that $G\langle S\rangle = G\langle T\rangle$.
- Get $G\langle S\rangle(s)$ (which is known).
- Take $G\langle S\rangle(s)$ as the approximate $G\langle T\rangle(t)$.

EXAMPLE: If the car car_S was produced in 1990, that its make is Honda-Civic, and that its price Q is 13,000.00 DM you would infer analogically that a car car_T which was produced in 1990 and the make of which is Honda-Civic would probably cost about 13,000.00 DM. The relevance knowledge that justifies this particular analogical inference is the connection $[P, Q]$ between P (the make and year of a car) and its price Q. ■

Total connections

If P includes *all* the information relevant to Q, then the connection $[P, Q]$ fully justifies the analogical inference. Then the analogy can be expected to yield a correct value $Q\langle T\rangle(t)$. Therefore, the schema is then

$$\frac{\dfrac{P\langle S\rangle = P\langle T\rangle,\ [P,Q]}{Q\langle S\rangle = Q\langle T\rangle, Q\langle S\rangle(s)}}{Q\langle T\rangle(t)}$$

EXAMPLE: Functional dependencies in data bases [138] are total connections. For $P(x, a_1, a_2, a_3) = lastname(x, a_1) \wedge birthday(x, a_2) \wedge mother(x, a_3)$ and $Q(x, b) = passport\_no(x, b)$ $[P, Q]$ is a total connection. An analogical inference using this connection concludes for persons W_1 and W_2 which have the same last name, birthday, and mother to have the same passport number. ■

For a restricted aspect representation by atomic formulas $P(x, a_1, \ldots, a_n)$ with a fixed predicate P, Russell [120] defines total connections, called total determinations, by:

DEFINITION (TOTAL DETERMINATION): $PdeterminesQ$ iff $\forall \overline{w}, \overline{y}, \overline{z}(P(\overline{w}, \overline{y}) \wedge Q(\overline{w}, \overline{z}) \rightarrow \forall \overline{x}(P(\overline{x}, \overline{y}) \rightarrow Q(\overline{x}, \overline{z})))$, where $\overline{w}, \overline{y}, \overline{z}, \overline{x}$ are tuples of variables. ■

Russell [120] argues that determinations are weaker knowledge than implications and represents the weakness by defining:
$PdeterminesQ$ iff $\exists R \forall \overline{x}, \overline{y}, \overline{z}(P(\overline{x}, \overline{y}) \rightarrow (Q(\overline{x}, \overline{z}) \leftrightarrow R(\overline{y}, \overline{z})))$ for a relation R which relates the values of R to the values of Q. Since R is not specified and, thus, not operationalized, this definition of *determines* is weaker than that of the implication $\forall \overline{x}, \overline{y}, \overline{z}(P(\overline{x}, \overline{y}) \rightarrow Q(\overline{x}, \overline{z}))$.

The definition of a total determination is quite strong, since it does not allow for exceptions and requires to know all the information that determines Q rather than only some influences[13]. Since explicit total determinations are rarely available in domain theories,

[13] Which, of course, could require a modification of the outcome of the analogical inference.

Russell's approach has been subject to criticism mainly because of its limitations for the scope of analogical reasoning.

The strongest special case of total connections are implicational connections defined as: $[P,Q]$ iff $\vdash_{Th} \forall\overline{x}(P(\overline{x}) \rightarrow Q(\overline{x}))$ for a domain theory Th.

Kedar-Cabelli [74] extracts implicational connections (explanations) by Explanation Based Generalization from a domain theory Th. The analogical inferences based on implicational connections can be described by the schema:

$$\frac{P(\overline{t}), [\forall\overline{x}(P(\overline{x}) \rightarrow Q(\overline{x}))]}{Q(\overline{t})}$$

Note that implicational analogical reasoning provides knowledge that could be inferred deductively as well, which is sometimes not regarded a proper analogical reasoning. Implicational analogy, however, may provide abbreviations of deductive reasoning procedures and thus satisfies one of the advantageous features of analogy: increasing efficiency.

Incomplete connections

Two kinds of non-total, that is, incomplete connections exist, uncertain and overgeneral connections. The analogical inference for *incomplete* connections $[P,Q]$ usually requires subsequent modification of the outcome. Many connections in human reasoning arise from experience and, therefore, they express just a tendency only. Hence, these connections have a certain degree of probability that causes an uncertainty for the corresponding analogical inference. The schema for analogical inference with uncertain connections is:

$$\frac{\dfrac{P\langle S\rangle = P\langle T\rangle, [P,Q]}{probably(Q\langle S\rangle = Q\langle T\rangle), \quad Q\langle S\rangle(s)}}{probably Q\langle T\rangle(t)}$$

EXAMPLE: The connection of month and weather is uncertain. Hence for the same month in 1980 and 1997 the same weather can be inferred analogically with some probability only. ∎

A connection $[P,Q]$ is *overgeneral* for an analogical inference if only a similarity of S and T w.r.t. P_1 can be stated and $P subsumes P_1$ [14]. Using an overgeneral connection $[P,Q]$ makes sense if a more specific connection $[P_1,Q]$ is not known. Then $[P,Q]$ still provides some justification.

The identity of S and T w.r.t. an aspect P_1 subsumed by P does guarantee an identity of S and T w.r.t. some aspect Q_1 that is subsumed by Q only rather than w.r.t. Q. Thus, analogical inferences based on overgeneral connections have the drawback to

[14] Aspects can be hierarchically organized by the relation *subsumes*.

DEFINITION: $P subsumes P'$ if $P\langle W\rangle \rightarrow P'\langle W\rangle$ is a tautology for all W. ∎

Example: The aspect function *subsumes* the aspect shock_absorbing_function be less reliable than those based on total connections. For an overgeneral connection $[P, Q]$, for $P subsumes P_1$, and for some Q_1 (that might not even be explicitly known) with $Q subsumes Q_1$ the analogical inference can yield $Q_1\langle T\rangle(t)$ only. This means it can yield an approximate $Q\langle T\rangle(t)$ only. The inference schema for overgeneral connections is:

$$\frac{\dfrac{P_1\langle S\rangle = P_1\langle T\rangle,\ [P,Q]}{Q_1\langle S\rangle = Q_1\langle T\rangle,} \qquad Q\langle S\rangle(s)}{probably Q\langle T\rangle(t)}$$

EXAMPLE: In the bionic example from section 1 we have the connection $[P, Q]$ for P = `function` and Q = `structure`. S and T agree in their `shock_absorbing_function` (P_1) which is subsumed by `function`. An agreement of S and T can only be inferred for some description Q_1 subsumed by `structure` rather than for the whole `structure`. ■

2. **Typical-instance-based analogical inference**
Psychological investigations provided evidence that typical instances are the preferred sources for an analogical reasoning where source and target are instances of the same category We cannot explain this preference by interpreting those analogical inferences as connection-based inferences, with P = "belonging to the same class" for a connection $[P, Q]$. The connection-based analogical inference defined above does not explain why a typical instance may preferably serve as the source. Hence, we need a specific analogical inference based on typical instances. An attempt [75] to formalize analogical inference based on typical instances uses a context-independent typicality rating $\sqsubseteq_{\mathcal{C}}$ introduced in section 2.2 as a partial order relation in concept structures $\mathcal{C}$ and the relevance knowledge "Q is relevant for the typical instance S" as defined next.

DEFINITION: An aspect A is called *relevant* for an instance W (written as $relevant(A, W)$) iff $A\langle W\rangle[x/w]$ is true and for all instances W' with $W' \sqsubseteq_{\mathcal{C}} W$ holds that $A\langle W\rangle[x/w']$ is either `true` or `undefined`. ■

The relevance information used here differs from the relevance knowledge assumed for the connection-based analogical inferences. This knowledge permits to analogically infer Q for a target T and a source S if $T \sqsubseteq_{\mathcal{C}} S$ and $typex(S)$. This gives rise to the inference schema

$$\boxed{\textbf{(TYP)} \quad \frac{\dfrac{typex(S), T \sqsubseteq_{\mathcal{C}} S, relevant(Q, S)}{Q\langle S\rangle = Q\langle T\rangle,} \qquad Q\langle S\rangle(s)}{Q\langle T\rangle(t)}}$$

where $T \sqsubseteq_{\mathcal{C}} S$ for a category $\mathcal{C}$ entails a similarity between the typical instance S and the instance T. Remember as usually, an analogical inference based on typical instances is confirmed only if its result does not contradict the knowledge inferable for the target by deduction.

EXAMPLE: An example for the typical instance-based analogical inference is: If Berlin is a typical city, and its transportation system includes an underground, busses, and taxis, and if Berlin is the typical example for Rome in the categories of cities (i.e., Rome is similar to Berlin), then Rome's transportation system includes an underground, busses, and taxis too.

■

Employing the **TYP** schema to find the description of a goal aspect G for a target T that is an instance of a category $\mathcal{C}$ yields the following procedure:

- Get a typical instance S with $T \sqsubseteq_{\mathcal{C}} S$.
- Check whether G is relevant for S.
- Take $G\langle S\rangle$ as $G\langle T\rangle$ which gives $G\langle T\rangle(t)$.

3. **Similarity-based analogical inference**
If an analogical inference is only based on the amount of existing similarity of the overall descriptions of the source, denoted by the aspect SYS, and a target, then *no justification* is included. Interpreting this case as one with an unidentified relevant aspect leads to the view that some aspect could be relevant to Q even if the connection is not known explicitly. This means that a commonality of large parts of SYS serves to increase the likelihood that the relevant aspect will be included into the commonality [119, 78], and this gives the schema:.

$$\textbf{(SIM)} \quad \frac{\dfrac{P\langle S\rangle = P\langle T\rangle, (P \text{ close\_to } SYS)}{probably(Q\langle S\rangle = Q\langle T\rangle),\ Q\langle S\rangle(s)}}{probably Q\langle T\rangle(t)}$$

Indeed, similarity-based analogical inference has often been implemented although it is the least justified type of analogy.

3.3 Reformulation

The notion of similarity needs to be extended because the similarity of S and T w.r.t. the aspect P is often found by mapping constants (including functions and relations) of S to constants of T. Accordingly, much of the work on analogy has focused on finding *symbol-to-symbol* mappings between *fixed* representations of the source and the target (see section 4.2). For instance, [50] supports the mapping by an "analogical hint" and constraints it to be a part of certain "abstractions". Winston [151] stressed the mapping of "important" relations between descriptions, whereas Gentner [40] emphasized the mapping between relations related by "higher-order relations" such as "implies" and "causes". Owen's analogy in theorem proving [109] employs extensions of mappings of symbols occurring in the source theorem to those occurring in the target theorem and their assumptions.

It turns out that the symbol mapping, generally accepted for analogical inference, is only a special case of reformulations, that should be included into the analogical inference. The fact that the commonalities of the source and the target are not necessarily explicitly represented, but have to be revealed, suggests the introduction of a change of representation of objects into reformulation. Examples for how different representations may hide a commonality are:

- A time description can be given in terms of hours or in terms of morning, noon, afternoon, evening, and night;
- Velocity can be described in terms of miles per hour as well as in terms of kilometer per hour.

Since we are interested in justified analogical inferences, we consider those reformulations only that preserve the connection $[P, Q]$ at hand. A change of representation by mapping a formula to a logically equivalent formula as well as to a formula equivalent with respect to a domain theory Th[15] and certain abstractions belong to the admissible reformulations defined by:

DEFINITION ($[P,Q]$-REFORMULATION): $[P,Q]-$reformulations are mappings ρ of object descriptions to object representations which preserve the connection $[P,Q]$. ■

EXAMPLE:

1. Abstraction
   Let P be the aspect describing time (in terms of hours and minutes) and day_of_week. The aspect Q describes whether the stores are open in Germany. Then each reformulation ρ mapping a time between 9.00 and 11.00 to "morning", a time between 11.01 and 13.00 to "noon", a time between 13.01 and 18.30 to "afternoon", a time between 18.31 and 22.00 to "evening", or a time between 22.01 and 9.00 to "night" is a $[P,Q]$-reformulation because stores in Germany are open (usually) from 9.00 through 18.30.

2. Employing domain equivalences
   Given a domain theory that contains the formula $(\forall x (x\ mh^{-1} = 1{,}6x\ kmh^{-1}))$, a mapping $\rho : x\ mph \mapsto 1.6x\ kmh^{-1}$ is a $[P,Q]$-reformulation for any P and Q.

3. Alteration (Knowing how changes of P change Q)
   Let the objects S and T be families, the aspect P describe the number of children and the aspect Q describes the number of beds needed. Each mapping increasing the number of children by x *and* increasing the number of beds by x as well is a $[P,Q]$-reformulation. Here certain changes of P cause certain known changes of Q. Other reformulations of this type are known in theorem proving by analogy, for example, the `Add-Argument` reformulation (see section 5).

■

For an operationalization we assume a set of $[P,Q]$-reformulations to be given for a domain at hand. The reformulations must at least heuristically preserve the employed connection for the domain, hence reformulations are domain-dependent. [108, 14] use reformulations that heuristically preserve the employed connection in a geometric domain, for instance, certain abstractions, reflection, rotation, and translation. [15] uses sets of reformulations that have been generated to preserve a given connection for restricted finite sets of objects. Several papers address abstracting reformulations [15, 95, 147].

[15] $\vdash_{Th} F \leftrightarrow \rho F$ for formulas F of the object description and a reformulation ρ.

The necessity to reveal similarities leads to an extended definition of similarity that considers an object S as similar to T w.r.t. P if $P\langle S\rangle$ can be reformulated to $P\langle T\rangle$ by an admissible reformulation ρ:

DEFINITION (EXTENDED SIMILARITY): S and T are PQ-similar with respect to an aspect P if there exists a $[P, Q]$-reformulation ρ of the description of S such that $\rho P\langle S\rangle = P\langle T\rangle$. ■

Including admissible reformulation extends the notion of similarity and is a means to make analogical inferences less dependent on the actual representation of the source and the target. Hence, including reformulation into analogical inferences provides more elaborate analogies that are within the scope of human analogy capabilities. The existence of a $[P, Q]$-reformulation ρ for a total connection $[P, Q]$ gives rise to the analogical inference schema:[16]

$$\boxed{\textbf{(REF)} \quad \dfrac{\dfrac{\rho P\langle S\rangle = P\langle T\rangle, [P, Q]}{\rho Q\langle S\rangle = Q\langle T\rangle,} \qquad Q\langle S\rangle(s)}{Q\langle T\rangle(t)}}$$

Employing this schema to infer analogically a goal description G for a target T yields the following procedure:

- Get a connection $[P, G]$.
- Retrieve a source S for which $P\langle S\rangle$ can be reformulated by a $[P, Q]$-reformulation ρ to $P\langle T\rangle$. (The retrieval can be done by searching for the ρ^{-1} with $\rho^{-1}P\langle T\rangle = P\langle S\rangle$.)
- Reformulate $G\langle S\rangle$ by ρ.
- Take $\rho G\langle S\rangle(s)$ as $G\langle T\rangle(t)$.

Summary

We incrementally introduced the notions of similarity and relevance which are basic for analogical reasoning. Several justifications of analogical inferences by different kinds of relevance knowledge were considered and led to different schemas for analogical inference. The resulting inference schemas describe analogical inference for connection-based, similarity-based, and typical-instance-based analogical inferences. Furthermore, it turned out that a reasonable and widely applicable definition of similarity for analogical inference needs to include reformulations of the source which preserve the justification.

3.4 Non-monotonicity of Analogical Inference

For an inference relation to be non-monotonic means that an inference based on some knowledge can be invalidated by an inference based on additional knowledge. As opposed to the classical inference relation in predicate logic, analogical inference is commonly considered to be non-deductive, tentative, and non-monotonic. These features of analogical reasoning have been addressed explicitly in [48, 50, 120, 93] and implicitly presupposed in all the approaches to reasoning by analogy.

[16] This schema would need to be modified for incomplete connections.

Let us look at an example: Knowing that the car car_S was produced in 1990, that its make is Honda-Civic, and that its price is 13,000.00 DM you would infer analogically that a car car_T which was produced in 1990 and the make of which is Honda-Civic would probably cost about 13,000.00 DM. The relevance knowledge that justifies this particular analogical inference is the determination of the price of a car by its make and year of construction. But what happens if you gain the additional knowledge that car_T is rusty and was involved in many severe accidents? Then you would no longer infer the price of 13,000.00 DM for car_T. This example provides a clue for the non-monotonicity of connection-based analogical inferences which stems from the default character of the justifying connections.

For understanding the non-monotonicity of analogical reasoning with typical instances, remember the classical Tweety story of non-monotonic reasoning: All birds fly. Tweety is a bird. Tweety is a penguin and, therefore, Tweety cannot fly. In this story the non-monotonicity has a lot to do with the break down of an inheritance in the category "bird". Correspondingly, the non-monotonicity of analogical inference based on typical instances is caused by the default character of the knowledge about the relevance of an aspect for a typical instance of a category which is assumed to hold for other instances as well.

As a result, the non-monotonicity of analogical inference may require the additional check of whether the result of the analogical inference contradicts any deductively derived knowledge, if it is necessary to guarantee soundness.

4 Computational Problem Solving by Analogy

The term "problem solving" has been used widely in AI for intelligent actions to achieve well-defined goals. Newell and Simon [105] characterized problem solving as selecting a sequence of operators that transforms an initial state into a desired goal state. Many researchers adopted this view. (e.g., [38, 121, 106]). A different approach has been proposed by McDermott [92] that views problem solving as a solution or plan instantiation by identifying and instantiating existing suitable plans. In this sense, the problem solving methods developed in AI can be organized in a problem solving reasoning continuum, ranging from *search intensive* to *knowledge intensive* methods.

Pure search intensive methods search exhaustively for a solution from first principles, i.e., individual steps that model atomic actions in the task domain and may be chained to form a solution to a problem. Pure knowledge intensive methods for problem solving presuppose the existence of a collection of instance or generalized solutions from where the problem solver may retrieve and eventually instantiate the adequate solution. Variations from these two extreme approaches extend the search intensive paradigm to heuristic searching, while the knowledge intensive extreme extends to case-based reasoning approaches in which the retrieved solution may be adapted after being retrieved and instantiated.

A compromising approach between the two extremes is to solve a new (target) problem *by analogy* to a previously solved similar (source) problem. This process entails searching for similar past problems and transforming their solutions into solutions potentially applicable to the new problem. Such a method was advocated in [22]. According to Carbonell, analogical problem solving can be defined as a problem solving strategy that transfers knowledge from past problem solutions or from past problem solving episodes to a target problem that shares significant aspects with the corresponding past experience and employs the transferred knowledge to construct solutions for the target problem. "Significant aspects shared by source and target" can be interpreted as aspects relevant to the goal to be achieved by the problem solving. That is, the analogical problem solving can be based on connections between all "significant aspects" and the goal.

Some variations of the general analogical process capture different levels of problem solving experience. Transformational analogy [21] uses simply the final source solution to guide the target solution, whereas derivational analogy [23] uses the knowledge relevant in the source problem solving process (e.g., control knowledge) to guide the target problem solving process. We focus on presenting the computational aspects of derivational analogy, as it has shown to be a powerful computational implementation of problem solving by analogy.

4.1 Derivational Analogy

Derivational analogy denotes a process that draws analogies from the experiences of the past reasoning process. The underlying key insight is that useful experience is encoded in the reasoning process used to derive solutions to similar problems, rather than just in the final solution. Therefore, derivational analogy is a *reconstructive* method by which lines of reasoning are transferred and adapted to a new problem as opposed to transformational methods that adapt directly final solutions.

Carbonell [23] discusses an example that shows the advantage of derivational analogy: Suppose you have coded a quicksort routine in Pascal, and then you are asked to recode the routine in LISP. Although the problem-solving process may preserve much of the inherent similarity, the resultant solutions may be hardly similar. A line-by-line transfer is clearly not appropriate, but a reuse of major structural and control decisions required to construct the Pascal program is possible. Therefore, the analogy must be guided by a reconsideration of the key decisions in light of the new situation. In particular, the derivation of the LISP quicksort program starts from the same specification, retaining the same divide-and-conquer strategy, but it may diverge in the selection of data structures (list vs. arrays) or in the method of choosing the comparison element. However, future decisions that do not depend on earlier divergent decisions can still be transferred to the new domain rather than recomputed.

The derivational analogy framework has been instantiated by several computational systems, including BOGARD [103], REMAID [13], PRIAR [70], APU [9], and by Prodigy/Analogy [143]. These systems apply to a variety of problem solving activities, including software reuse in a UNIX programming domain [9], the design of human computer interfaces [13], and multiple planning applications [70, 143].

We shall concentrate our presentation of computational implementations of analogy in problem solving on the Prodigy/Analogy system, because it implements all subprocesses of analogy, it is a domain-independent system embedded into a general problem solving architecture (a non-linear planner) rather than a stand-alone tool, it allows for multiple sources to be replayed for a target, and because it handles large case bases (over 1000 cases) and addressing the indexing and organization of planning cases.

Purpose, Domains, Knowledge Representation

The main purpose of derivational analogy is to solve problems more efficiently than an original basic problem solver (or planner) by reusing past experience to guide the generation of solutions for new problems and thus avoiding a completely new search effort. Examples of the results obtained by Prodigy/Analogy in terms of increased efficiency obtained for the problems are given for example in [146], where the base-level planning effort is compared with analogical planning guided by source cases. Prodigy/Analogy permits a partial transfer of solutions when a total transfer cannot be justified.

Prodigy/Analogy was and continues to be demonstrated in a variety of domains, For instance a transportation domain,

Prodigy/Analogy is built on top of the planner of the Prodigy architecture. This generative planner is a means-ends analysis backward-chaining nonlinear planner. PRODIGY4.0's nonlinear character stems from its dynamic goal selection which enables the planner to fully interleave plans, exploiting common subgoals and addressing issues of resource contention. PRODIGY4.0 combines state-space search corresponding to a simulation of plan execution of the plan and backward-chaining responsible for goal-directed reasoning [142].

A domain is specified in Prodigy as a class hierarchy of objects, a set of operators, inference rules, and control rules. Each operator has a precondition expression that must be satisfied before the operator can be applied, and an effects list with add- and delete statements. As an example, Figure 4 shows the operator `load-truck` from a logistics transportation domain that specifies that an object can be loaded into a truck if the object and the truck are at the same location. In this domain packages are to be moved among different cities. Packages are carried within the same city in trucks and between cities in airplanes. At each city there are several locations, e.g., post offices and airports.

```
(OPERATOR load-truck
  (params
        ((<obj> OBJECT)
        (<truck> TRUCK)
        (<loc> LOCATION)
        (<city> CITY)))
  (preconds
        (and (loc-at <loc> <city>)
             (part-of <truck> <city>)
             (at-obj <obj> <loc>)
             (at-truck <truck> <loc>)))
  (effects ()
        ((add (inside-truck <obj> <truck>))
         (del (at-obj <obj> <loc>)))))
```

Figure 4: Example of an Operator in Prodigy

A problem consists of an initial state and a goal expression. The initial state is a set of literals describing the initial situation. For example, consider the problem illustrated in Figure 5. In this problem there are two objects, `ob4` and `ob7`, one truck `tr9`, and one airplane `pl1`. There is one city `c3` with a post office `p3` and an airport `a3`. In the initial state, `ob4` is at `p3` and the goal is to have `ob4` inside of `tr9`.

Corresponding to its search cycle, Prodigy's planner produces a plan with decision nodes, i.e., goal, operator, bindings, and apply nodes. Automatic generation of the derivational planning episodes occurs by extending the base-level generative planner with the ability to examine its internal decision cycle, recording the justifications for each decision during its search process. The base-level PRODIGY4.0 reasons about multiple goals and multiple

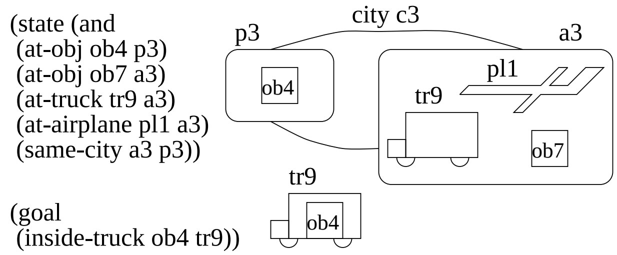


Figure 5: Example: The goal is to load one object into the truck. Initially the truck is not at the object's location.

alternative operators relevant to achieving the goals. This choice of operators amounts to multiple ways of trying to achieve the same goal. Therefore, in addition to searching in the space of multiple goal orderings, as most of the standard nonlinear planners do, our planner searches equally in the space of multiple different approaches to solving a problem. The search in both of these spaces benefits from the analogical guidance provided by the similar planning episodes. Prodigy/Analogy provides a *language* the *justifications* for the decisions made. A planning case to be stored consists therefore of the successful solution trace augmented with the justifications, i.e., the derivational trace. Prodigy/Analogy identified that the key feature of this language is that it needs to be reinterpretable at planning replay time. We will later further comment on this issue. Figure 6 shows the structure of the justification slots at the stored decision nodes.

Goal Node	Chosen Op Node	Applied Op Node
:step	:step	:step
:sibling-goals	:sibling-ops	:sibling-goals
:sibling-appl-ops	:why-this-op	:sibling-appl-ops
:why-subgoal	:relevant-to	:why-apply
:why-this-goal		:why-this-op
:precond-of		:chosen-at

Figure 6: Justification record structure

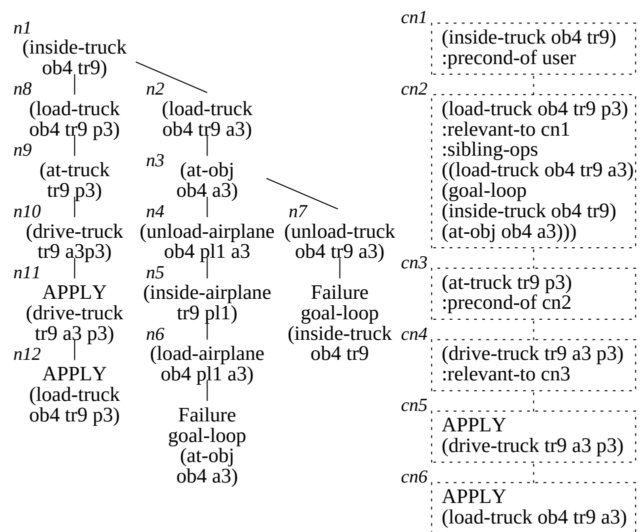
There are mainly three different kinds of justifications: links among choices capturing the subgoaling structure (slots `precond-of` and `relevant-to`), records of explored failed alternatives (the `sibling-` slots), and pointers to any applied guidance (the `why-` slots).

We show an example of the generation and representation of a planning case [17]. Consider the problem shown in Figure 5. There are two operators that are relevant for solving this problem: the operator `load-truck`, as specified in Figure 4, and the operator `drive-truck` that states that a truck can move freely between locations within the same city. Figure 7 (a) shows the decision tree corresponding to Prodigy's search for the solution. Nodes are numbered in the order in which the search space is expanded. The search is a sequence of goal choices followed by operator choices followed occasionally by applying operators to the planner's internal state when their preconditions are true in that state and the decision for immediate application is made. The solution found consists of driving the truck from the

[17]The problems used in the examples are simple for the sake of a clear illustration of the analogical reasoning process.

airport to the post office and then loading the object.

This trace illustrates PRODIGY handling multiple choices of how to instantiate operators. There are two instantiations of the operator `load-truck` that are relevant to the given goal, i.e., the instantiations `(load-truck ob4 tr9 p3)` and `(load-truck ob4 tr9 a3)` add the goal `(inside-truck ob4 tr9)`. An object can be loaded into a truck at both post office and airport locations. Node `n2` shows that the alternative of loading the truck at the airport `a3` is explored first. This leads to two failed paths. The solution is found after backtracking to the alternative child of node `n1`. Nodes `n8` through `n12` show the final sequence of successful decisions. `n8` shows the correct choice of loading the truck at the post office, where `ob4` is located. The solution corresponds to the two steps applied at nodes `n11` and `n12`: the truck `tr9` is driven from `a3` to `p3`, as chosen at node `n8` and then it is loaded with `ob4`.[18]



(a) (b)

Figure 7: (a) The search tree to solve the problem in Figure 5; (b) The corresponding problem solving episode to be stored.

Figure 7 (b) shows the case generated from the problem solving in Figure 7 (a), where only a subset of the justifications is shown. The entire search tree is not stored in the case, but only the decision nodes of the final successful path. The subgoaling structure and the record of the failures are annotated at these nodes. Each goal is a precondition of some operator and each operator is chosen and applied because it is relevant to some goal that needs to be achieved. The failed alternatives are stored with an attached reason of failure.

As an example, node `cn2` corresponds to the search tree node `n8`. This search node has a sibling alternative `n2` which was explored and failed. The failed subtree rooted at `n2` has two failure leaves, namely at `n6` and `n7`. These failure reasons are annotated at the case node `cn2`.

Top Level Algorithm

Prodigy/Analogy is the first system that achieved an integration of generative planning and analogy [145]. It casts the complete cycle of constructing, storing, retrieving, and reusing

[18]Note that domain-independent methods to try to reduce the search effort [133] in general do not capture domain specific control knowledge, which must be then acquired by learning.

planning experience. Figure 8 shows how Prodigy/Analogy extends the classical generative planning paradigm with the ability to replay possibly multiple past planning episodes that are found by the retrieval procedure to cover jointly a new planning problem.

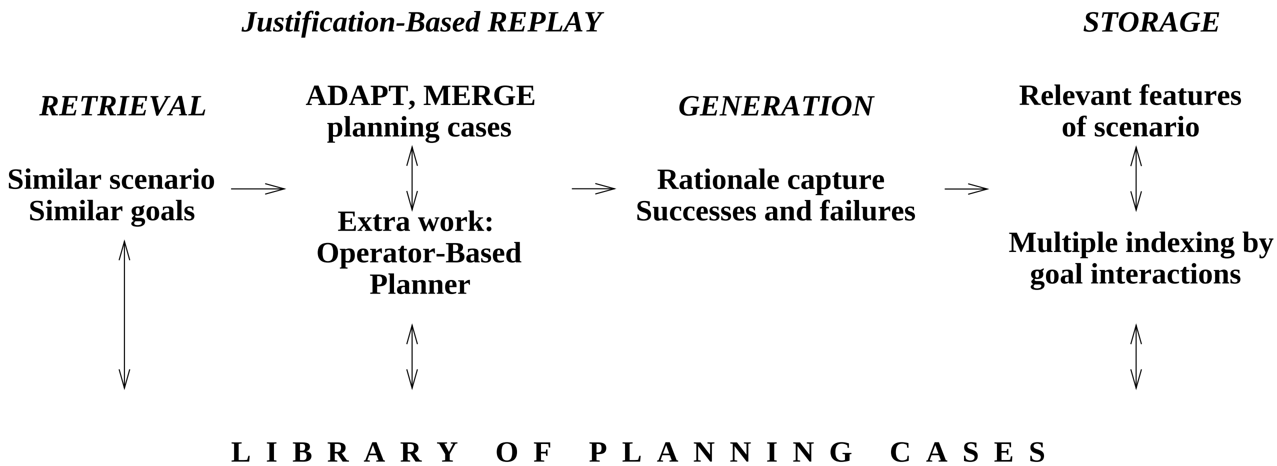


Figure 8: Prodigy/Analogy: Retrieval, replay, generation, and storage of planning episodes.

After a problem solving case is generated, it is stored. From the exploration of the search space and by following the subgoaling links in the derivational trace of the plan, the system identifies, for each goal, the set of *weakest preconditions* necessary to achieve that goal. It recursively creates the *foot-print* of a user-given goal conjunct by doing goal regression [149], i.e. projecting back its weakest preconditions into the literals in the initial state. The literals in the initial state are, therefore, *categorized* according to the goal conjunct that employed them in its solution. Goal regression acts as an explanation of the successful path. Foot-printing is similar to explanation-based indexing techniques [102] and chunking [87] in that it uses an explanation provided by the subgoaling chain supplied by the underlying domain theory.

The system automatically identifies the sets of interacting goals of a plan by partially ordering the totally ordered solution found. The connected components of the partially ordered plan determine the independent fragments of the case each corresponding to a set of interacting goals. Each case is multiply indexed by these different sets of interacting goals.

In Prodigy/Analogy source and target case belong to the same domain. When a new problem is presented to the system, the retrieval procedure must (partially) match the new initial state and goal statement against the indices of the cases in the case library. Table 2 shows the retrieval procedure where the underlying strategy is to get guidance from cases that cover the largest possible set of interacting goals.

The retrieval algorithm focuses on retrieving past cases where the planner experienced equivalent goal interactions and has a reasonable match between initial states because then it can expect to achieve a large reduction in the new planning search effort. The retrieval procedure may return a set of similar planning cases. Each retrieved case provides guidance to a set of interacting goals from the target goal. As an example, Figure 9 shows a new problem and two past cases selected for replay by analogy. The cases are partially instantiated to match the new situation. Further instantiations occur while replaying.

Input : A new problem with goal statement $\mathcal{G} = G_1, G_2, \ldots, G_k$ and initial state $\mathcal{S}$.

Output : A set of similar cases.

procedure **Retrieve_Similar_Cases** ($\mathcal{G}$, $\mathcal{S}$):

1. Initialize to empty the set of matched cases.
2. Initialize to k the number of interacting goals to match, i.e., the total number of goals.
3. Initialize to $\mathcal{G}$ the set of goals not covered yet by some matched case, i.e., the complete goal statement.
4. Initialize to nil the past case to consider.
5. Initialize to nil that a successful match was reached.
6. Terminate if all the goals have been matched successfully or there are no more past cases. Return the set of matched cases.
7. Retrieve another candidate analog as a function of the past case considered, the number of interacting goals to match, and the current set of goals not yet covered.
8. Evaluate the similarity measure for the matched goals.
9. Use the matched goal substitution to evaluate the similarity measure between the new initial state and the foot-printed initial state of the candidate analog.
10. If the total match is above the desired threshold preestablished, then add the current case to the set of matched cases, continue with the other goals to be covered, and goto step 6.
11. If not satisfied with the total match, then go to step 7.
12. If not satisfied with the total match, then decrease the number of interacting goals to match, and goto step 7.

Table 2: Retrieving similar past cases

The replay mechanism involves a complete interpretation of the justification structures annotated in the past cases in the context of the new problem to be solved. Equivalent choices are made when the transformed justifications hold. Otherwise, Prodigy/Analogy plans for the new goals using its domain operators adding new steps to the solution or skipping unnecessary steps from the past cases. Table 3 shows the main flow of control of the replay algorithm.

Past cases

(goal (inside-airplane ob3 pl5))
(relevant-state (at-obj ob3 <ap3>)
(at-airplane pl5 a12))

(goal (inside-truck ob8 tr2))
(relevant-state (at-obj ob8 p4)
(at-truck tr2 <ap7>))

New problem

(goal (inside-airplane ob3 pl5)
(inside-truck ob8 tr2))
(initial-state
(inside-truck ob3 tr2)
(at-truck tr2 p4)
(at-airplane pl5 a12)
(at-obj ob8 p4))

Figure 9: Past cases cover the new goal and partially match the new initial state.

1. Terminate if the goal is satisfied in the state.
2. Choose a step from the set of guiding cases or decide if there is need for additional problem solving work. If a failure is encountered, then backtrack and continue following the guiding cases at the appropriate steps.
3. If a goal from a past case is chosen, then
   - 3.1 Validate the goal justifications. If not validated, go to step 2.
   - 3.2 Create a new goal node; link it to the case node. Advance the case to its next decision step.
   - 3.3 Select the operator chosen in the case.
   - 3.4 Validate the operator and bindings choices. If not validated, base-level plan for the goal. Use justifications and record of failures to make a more informed new selection. Go to step 2.
   - 3.5 Link the new operator node to the case node. Advance the case to its next decision step.
   - 3.6 Go to step 2.
4. If an applicable operator from a past case is chosen, then
   - 4.1 Check if it can be applied also in the current state. If it cannot, go to step 2.
   - 4.2 Link the new applied operator node to the case node. Advance the case to its next decision step.
   - 4.3 Apply the operator.
   - 4.4 Go to step 1.

Table 3: The main flow of control of the replay procedure.

Figure 10 shows the replay episode to generate a solution to the new problem of Figure 9. The new situation is shown at the right hand side of the figure and the two past guiding cases at the left.

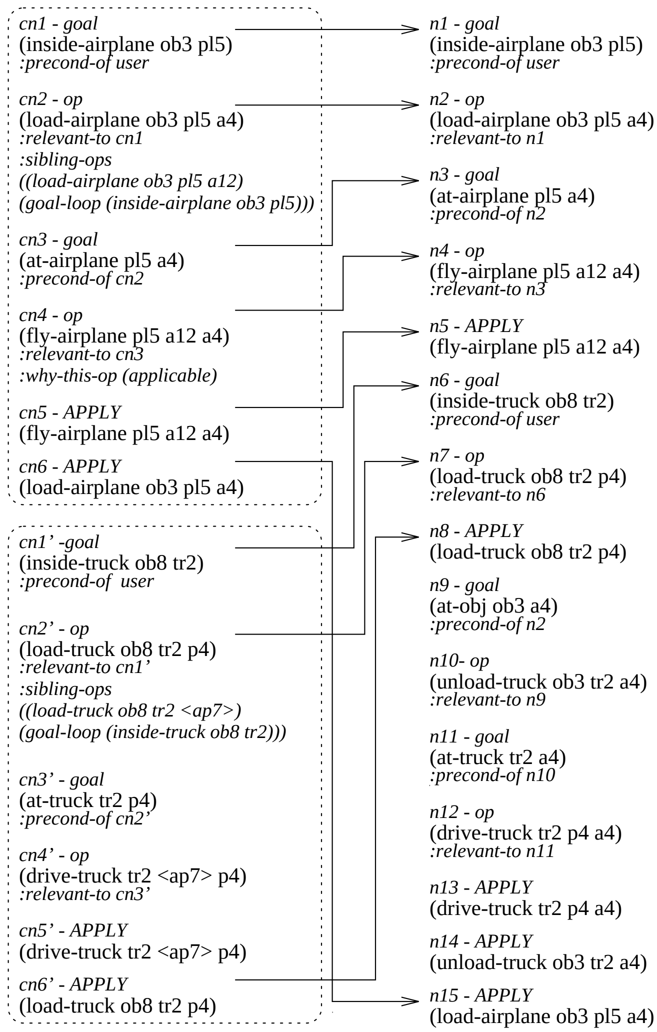


Figure 10: Derivational replay of multiple cases.

The transfer occurs by interleaving the two guiding cases, performing any additional work needed to accomplish remaining subgoals, and skipping past work that does not need to be done. In particular, the case nodes `cn3'` through `cn5'` are not reused, as there is a truck already at the post office in the new problem. The nodes `n9-14` correspond to unguided additional planning done in the new episode.[19] At node `n7`, Prodigy/Analogy prunes out an alternative operator, namely to load the truck at any airport, because of the recorded past failure at the guiding node `cn2'`. The recorded reason for that failure, namely a goal-loop with the `(inside-truck ob8 tr2)`, is validated in the new situation, as that goal is in the current set of open goals, at node `n6`. Note that the two cases are merged using a bias to postpone additional planning needed. Different merges are possible.

The replay functionality transforms the planner, from a module that costly generates possible operators to achieve the goals and searches through the space of alternatives gen-

[19]Note that extra steps may be inserted at any point, interrupting and interleaving the past cases, and not just at the end of the cases.

erated, into a module that tests the validity of the choices proposed by past experience and follows equivalent search directions. The replay procedure, shown in the procedure of Table 3, provides the following benefits:

- Proposal and validation of choices versus generation and search of alternatives (steps 2, 3.1, 3.3, 3.4, and 4.1).
- Reduction of the branching factor – past failed alternatives are pruned by validating the failures recorded in the past cases (step 3.4); if backtracking is needed Prodigy/Analogy backtracks also in the guiding cases – through the links established at steps 3.2, 3.5 and 4.2 – and uses information on failure to make more informed backtracking decisions.
- Subgoaling links identify the subparts of the case to replay – the steps that are not part of the active goals are skipped. The procedure to advance the cases, as called in steps 3.2, 3.5 and 4.2, ignores the goals that are not needed and their corresponding planning steps.

Particular Computational Techniques

Note that the particular computational techniques employed in Prodigy/Analogy are:

- annotating the decision nodes and re-interpretation of the justification structures,
- the foot-printing in the case storage process,
- the calculation of the similarity metric in the retrieval, and
- the guidance by multiple cases in the replay process.

Foot-printing showed to be an interesting technique to find goal-relevance. To illustrate how foot-printing works, consider a domain where a robot moves among rooms pushing boxes. Rooms are connected through doors that can be open, closed, or locked. Figure 11 shows a problem (goals, initial state) and its foot-print for a plan. Note that foot-printing and its notion of goal-relevance is dependent on the particular solution considered. In particular, different plans for the same problem can use different features of the initial and therefore yield different foot-prints.

```
goal:
     (and (inroom box1 rm2) (door-open dr34))

initial state:
     (inroom robot rm1)              (inroom box1 rm1)
     (inroom box2 rm3)               (inroom box3 rm3)
     (inroom box4 rm4)               (inroom key24 rm4)
     (door-open dr12)                (door-open dr13)
     (door-closed dr34)              (door-locked dr24)
     (arm-empty)                     (pushable box1)
     (connects dr12 rm1 rm2)         (connects dr13 rm1 rm3)
     (connects dr24 rm2 rm4)         (connects dr34 rm3 rm4)

plan:
     (GOTO-BOX box1)
     (PUSH-THRU-DOOR box1 dr12)
     (GO-THRU dr12 rm1)
     (GOTO-DOOR dr13)
     (GO-THRU dr13 rm3)
     (GOTO-DOOR dr34)
     (OPEN-DOOR dr34)

foot-print of the initial state for both conjunctive goals:
     (inroom robot rm1)         (inroom box1 rm1)
     (door-open dr12)           (door-open dr13)
     (door-closed dr34)
     (arm-empty)                (pushable box1)
     (connects dr12 rm1 rm2)    (connects dr13 rm1 rm3)
     (connects dr34 rm3 rm4)
```

Figure 11: The foot-printed initial state for a plan and goal

Since not all features in the foot-printed initial state are equally relevant in solving the particular goal, the foot-printed initial state is organized in a discrimination tree. This tree captures different levels of relevance by the distance to the root. A stored case is indexed by the parametrized goal and the foot-printed initial state. The indexing goals are stored in a hash table pointing to the discrimination network of the foot-printed initial state.

The case retrieval identifies similar problems according to the following *similarity metric* that takes into account the conjunct goals matched to the target goals and the degree of match of the foot-printed initial state that corresponds to these conjunct goals.

DEFINITION (GLOBAL FOOT-PRINT MATCH VALUE): Let T be a target problem and S a source problem, respectively with initial states I_S and I_T and goals G_S and G_T. Let δ_G^σ be the number of literals of G_S matched with those of G_T, under substitution σ, and let $G_1, \ldots G_m$ be the matched goals. Let I_{fp}^S be the foot-printed initial state of S for the set of matched goals $G_1, \ldots G_m$. Let δ_I^σ be the number of literals of I_{fp}^S matched with literals of I_T under substitution σ.
Then S foot-print matches T with match value $\delta_{fp}^\sigma = \delta_G^\sigma + \delta_I^\sigma$. ■

The replay includes two main procedures: the validation of the justifications annotated in the case, and the merge of the multiple guiding cases (as illustrated in Figure 10). Prodigy/Analogy can construct a new solution from a set of guiding cases as opposed to a single past case. Complex problems may be solved by resolving minor interactions among simpler past cases. Guidance from several cases can be merged in Prodigy/Analogy in several modes with different degrees of deliberation on the predicted level of interaction.

Reference to the Theory Section

Basically, derivational analogy is connection-based. Derivational analogy implicitly realizes a connection-based analogical inference rather than applying the inference schema **CONN**. In terms of aspects the representation of the problem solution corresponds to the aspect Q and all the solution-relevant facts conjunctively build the aspect P. Derivational analogy follows practical needs by including into P both control knowledge encoded in justifications and goal-relevant features of the initial state. Since usually only a *partial* match can be achieved, the implicit connections $[P, Q]$ are overgeneral.

Prodigy/Analogy dynamically extracts the solution-relevant features of a case that belong to the problem description by foot-printing the initial state. In addition, some information relevant to achieve the goal, that is not encoded in the problem description at all, is acquired during the source planning process and encoded in the justifications annotating the plan decision nodes. These justifications are encoded in a control-language that is different from the object-level language of the problem description. The correspondence of this goal-relevant information in source and target can only be established during the analogical replay and this is done by checking the justifications. A symbol mapping is possible in Prodigy/Analogy by type-preserving instantiations. This amounts to a simple level of reformulation.

4.2 A Neural Network Approach

Several computational analogy systems utilize neural nets and reinforcement learning, among them COPYCAT [59] and ACME.

Holyoak and Thagard [63] designed a system called ACME (Analogical Constraint Mapping Engine) and a system ARCS (Analog Retrieval by Constraint Satisfaction) [135] that both use a connectionist architecture. These systems are based on results from psychological research as well as theoretical convictions emphasizing the role of structural consistency, semantic similarity, and pragmatic relevance. The structural constraint supports mappings that maximize the consistency of relational correspondence between the elements of the two analogs by assuming that two objects are similar to the extent that they serve as arguments of similar predicates and vice versa. Since similarity cannot be reduced to structural consistency, the semantic similarity constraint enforces mappings of predicates which are supposed to have similar meanings. The pragmatic centrality constraint favors mappings of elements relevant to the purpose of the analogical reasoning.

Purpose, Knowledge Representation, Domains

ACME focuses on the subprocess of *mapping elements* of a source with elements of a target based on interacting structural, semantic, and pragmatic constraints. Its purpose is to model human analogical mapping and then to compare the computer model and human analogy performance. ARCS is designed to the same general end and deals with the *retrieval* of sources. That is, only single subprocesses are tackled by the systems.

Figure 12 depicts the input supplied to the system for an example, where ACME was used to simulate an explanation by analogy between the flow of water caused by differential pressure and the flow of heat caused by differential temperature. In Figure 12, w1 - w11 and h1 - h11 are the names of the source and target literals, respectively.

As in the HEAT-FLOW example of Figure 12, ACME generally uses a symbolic representation of source and target in a simplified version of the predicate calculus. The pragmatic constraints come into play with optional inputs of the form (IMPORTANT...) and (PRESUMED...). (IMPORTANT <element>) has to be given explicitly as an input for each "element" (object, relation, formula) that is relevant for the goal of the analogy, e.g., a query. A PRESUMED(...) input represents the bias to a user-intended mapping. Semantic similarity values are supplied to ACME by explicit statements of the form

(SIMILAR <concept-1> <concept-2> <degree>).

In order to represent queries that serve to focus attention on the pragmatically central aspect of a target analog, ACME allows some arguments to be variables with a question mark, as in `value?` for cross-structure queries and as in `?proposition?` for internal queries as in Figure 12. Cross-structure queries indicate that the answer to the query should come from seeing the relevant correspondences in the other analog. For example, if *Ortega* is mapped to *Castro*, then *Castro*'s political views could be transferred to *Ortega*. Internal queries indicate that the answer is to be found within the structure that contains the query, i.e., the answer is some argument already present in the structure. For instance, `cause(?proposition? h10)` suggests to instantiate `?proposition?` with a proposition in the target representation of the example in Figure 12. The query `?proposition?` in proposition `h11` of the target represents the unknown cause of heat flow.

ACME has been applied to examples that include problem mappings (water flow - heat flow explanation, light bulb - radiation problems, fortress - radiation problems), analogical arguments, explanatory analogies, story analogies, mappings between sets of mathematical formulas, and metaphors.

Top Level Algorithm

From the input consisting of the representations of the source and the target and optional information about similarity and pragmatic importance ACME generates a network. The network as schematically displayed in Figure 13 consists of units that represent hypotheses about the pairing of source and target elements, of two special (pragmatic and semantic) units, and of excitory and inhibitory links between the units. Numbered capital letters represent propositions, unnumbered capital letters represent predicates, and lowercase letters represent objects. Solid lines represent excitatory links and dotted lines represent inhibitory links. A network as in Figure 13 consists of units that represent hypotheses about the pairing of source and target elements, of two special units, and of excitory and inhibitory links between the units.

```
WATER-FLOW (source)

(liquid (obj-water) w1)
(flat-top (obj-water) w2)
(clear (obj_beaker) w3)
(diameter (obj-vial obj-val2) w5)
(greater obj-val1 obj-val2) w6)
(pressure (obj-beaker obj-val3) w7)
(pressure (obj-vial obj-val4)w8)
(greater (obj-val3 obj-val4) w9)
;flow from x to y of w via z:
(flow (obj-beaker obj-vial obj-water obj-pipe) w10)
;pressure difference causes flow:
(cause (w9 w10) w11)
(volume (obj_beaker obj_val5) w12)
(volume (obj_vial obj_val6) w13)
(greater (obj_val5 obj_val6) w14)
;diameter difference causes volume difference
(cause (w6 w14) w15)

HEAT-FLOW (target)

(liquid (obj-coffee) h1)
(flat-top (obj-coffee) h2)
(temperature (obj-coffee obj-val13) h7)
(temperature (obj-ice-cube obj-val14) h8)
(greater (obj-val13 obj-val14) h9)
(flow (obj-coffee obj-ice-cube obj-heat obj-bar) h10)
;what causes the flow?
(cause(?proposition? h10) h11)
(volume (obj_coffee obj_val15) h12)
(volume (obj_ice_cube obj_val16) h13)
(greater obj_val15 obj_val16) h14)
```

Figure 12: Representation of the water-flow and heat-flow situations

INPUT

TARGET	SOURCE	Semantic Weights for	Pragmatic Information
T1 A(a)	S1 M(m)	A = M	IMPORTANT D
T2 B(b)	S2 N(n)	C = O	
T3 C(a,b)	S3 O(m,n)		
T4 D(b,a)	S4 P(n,m)		

PARTIAL RESULTING NETWORK

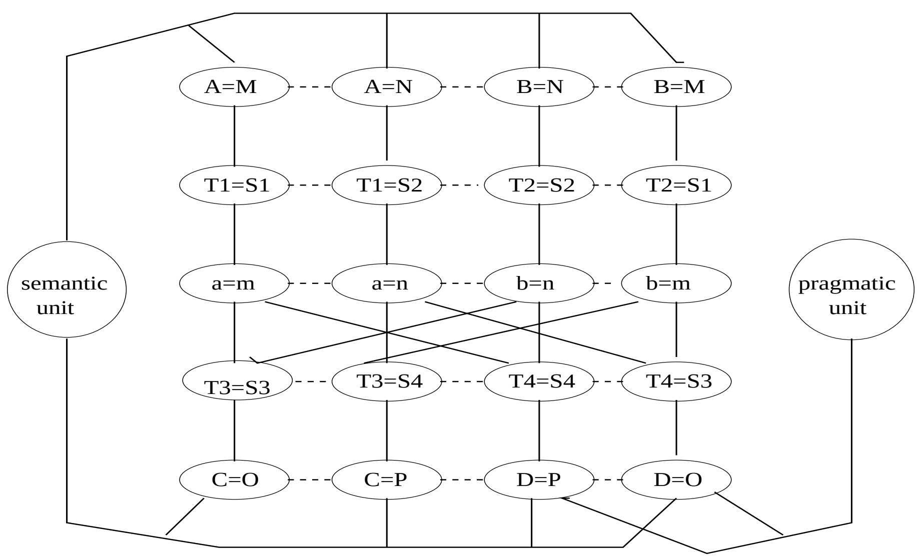


Figure 13: Schematic ACME mapping network

Two restrictions are used to limit the number of units: The type restriction limits mappings to those between elements of the *same type*: object to object, n-ary predicate to n-ary predicate, and proposition to proposition. A part-correspondence restriction applies if the analogs are divided into major parts. The mapping units in Figure 13 represent all possible pairings of elements of the same type, e.g., C = O, A = M, a = m, and T1 = S1.

Symmetric excitatory links are created between mapping hypotheses that are *consistent* with each other. For each proposition mapping, excitatory links are created between the proposition mappings and the corresponding mappings of the proposition's predicates and arguments, e.g., between T1 = S1 and A = M and between T1 = S1 and a = m. For predicates that have more than one argument, the argument mappings support each other by excitory links in order to realize the *structural constraint*, e.g., T3 = S3, a = m supports b = n. Excitatory links are created between predicate-mapping units and their corresponding argument-mapping units. ACME implements the constraint of *semantic similarity* by linking any mapping unit relating two semantically similar predicates to the special semantic unit that has a maximum activation value. Values are provided by SIMILAR-statements. If no value is given, ACME assumes a minimum value. Units relating identical predicates are connected automatically to the semantic unit with high values of excitation. Similarly, the pragmatic unit is linked to mapping units with IMPORTANT-elements. After all units have been formed, symmetric inhibitory links are placed between hypotheses that are inconsistent, that is between all units that represent alternative mappings for the same element, e.g., between b = n and b = m.

A summary of the ACME algorithms for forming a mapping network and learning the probabilities of mappings is given in Table 4 which is taken from [63]. To summarize, 1 and 2 implement the structural constraint, 3 implements the semantic similarity constraint, and 4 and 5 along with the note in 2 implement the pragmatic centrality constraint.

When the network has stabilized, those mapping hypotheses represented by the units with the highest activation jointly determine a mapping of the source representation and the target representation that best satisfies the given interacting constraints.

I Set up a mapping network.

1. For each proposition $P_{Ti} = pred_{Ti}(arg_{Ti1} \ldots arg_{Tin})$ and each proposition $P_{Sj} = pred_{Sj}(arg_{Sj1} \ldots arg_{Sjm})$ are in the same part (e.g.,goal) of their respective structure and have the same number of arguments
   - Construct the units $P_{Ti} = P_{Sj}$, $pred_{Ti} = pred_{Sj}$, and each $arg_{Tik} = arg_{Sjk}$.
   - Construct links between $P_{Ti} = P_{Sj}$ and $pred_{Ti} = pred_{Sj}$.
   - Construct links between $P_{Ti} = P_{Sj}$ and each $arg_{Tik} = arg_{Sjk}$.
   - Construct links between $pred_{Ti} = pred_{Sj}$ and each $arg_{Tik} = arg_{Sjk}$.
   - Construct links between each pair of $arg_{Tik} = arg_{Sjk}$.

2. Construct inhibitory links with a negative weight, set by a default parameter i, between any units that represent incompatible mappings.

   Note: If a mapping unit connects an internal query variable to some other < element>, then construct excitory rather than inhibitory links to other units representing possible mappings to < element>.

3. For each predicate-mapping unit $pred_{Ti} = pred_{Sj}$ construct a link from the semantic unit based on the degree of semantic similarity between the two predicates, with a weight ranging from a minimum value, if there is no similarity, to a maximum value, if the predicates are identical.

4. For each element listed as IMPORTANT, construct a link from the pragmatic unit to each unit concerning a mapping for that element with a weight equal to a certain parameter $p1$.

5. For each unit listed as PRESUMED, construct a link from the pragmatic unit with a weight equal to a parameter $p2$.

II Run the network starting with maximum value 1 for the semantic and the pragmatic unit and with some initial value for all other units. At each cycle $t + 1$ calculate the new activation a_j of all units j according to
$a_i(t + 1) = a_j(t)(1 - d) + enet_j(max - a_j(t)) + inet_j(a_j(t) - min)$, where d is a decay parameter, $enet_j$ is the net excitatory input, $inet_j$ is the net inhibitory input, with $min = -1$ and $max = 1$.

Table 4: Outline of the ACME algorithm

The best mapping outputs of the ACME algorithm for the HEAT-FLOW example is shown in Figure 14.

```
best mappings after 93 cycles:

LIQUID LIQUID           0.52
FLAT-TOP  FLAT-TOP      0.52
CLEAR  LIQUID          -0.004
DIAMETER  TEMPERATURE   0.10
GREATER  GREATER        0.84
PRESSURE  TEMPERATURE   0.74
FLOW  FLOW              0.83
CAUSE  CAUSE            0.76
VOLUME VOLUME           0.84
W6  H9                 -0.14
W14 H14                 0.70
OBJ_VAL6  OBJ_VAL16     0.81
OBJ_VAL5  OBJ_VAL15     0.81
W9  H9                  0.78
mapping W9 ?PROPOSITION? is also possible: 0.78
W10  H10                0.85
OBJ-PIPE  OBJ-BAR       0.80
OBJ-VAL4  OBJ-VAL14     0.75
OBJ-VAL3  OBJ-VAL13     0.75
OBJ-VIAL  OBJ-ICE-CUBE  0.89
OBJ-VAL2  OBJ-VAL14     0.09
OBJ-VAL1  OBJ-VAL13     0.09
OBJ-BEAKER  OBJ-COFFEE  0.89
OBJ-WATER  OHB-HEAT     0.80
```

Figure 14: ACME's best mapping output of the HEAT-FLOW example

Particular Computational Techniques

Running a neural net that was set up by ACME, updates the activations of units until all units have reached an asymptotic activation. The resulting unit activations are interpreted as the probabilities of the particular mapping hypotheses.
$G(t) = \sum_i \sum_j w_{ij} o_i(t) o_j(t)$, where o_i is the output in cycle t of a unit i and w_{ij} is the activation of the link between i and j, is interpreted as a measure of the overall fit of the emerging mapping to the constraints. The choice of minimal and maximum activation values and of the default values of similarity and relevance clearly influences the output of the run.

Running the net yields a synergy of different constraints. For instance, in the HEAT-FLOW example, the information that both water and coffee are liquids and have a flat top tends to encourage water to map to coffee rather than to heat. Despite this misleading similarity information, the structural information from
(flow (obj-beaker obj-vial obj-water obj-pipe) w10) and

(flow (obj-coffee obj-ice-cube obj-heat obj-bar) h10)
enables ACME to credit higher activation to the hypothesis that water maps to heat as shown in Figure 14.

Since ACME's inputs (IMPORTANT ...) differ for different goals of the analogical reasoning, a new net has to be constructed for each new goal.

In case ACME is used to find a problem solution by mapping, the goal is to find an instantiation $F[h_i/x_i]$ of variables x_i in a formula F. The instantiation is suggested by the maximally activated of nodes (w_i, x_i) and (w_i, h_i) for each x_i. In the representation of the HEAT-FLOW example in Figure 12, the purpose of the analogy is captured by the proposition `cause(?proposition?  h10) h11)`, where the internal query variable `?proposition?` represents the unknown cause of the heat flow in the target. Since `w9` has the desired feature by virtue of its appearance in the proposition `(cause(w9 w10) w11)`, whereas `w6` does not, the units for mapping `w9` are preferred. This means the mapping of `w9` to `h9` is preferred to the mapping of `w9` to `h6` and, therefore, the mapping of `pressure` to `temperature` is preferred to the mapping of `diameter` to `temperature`. Since `?proposition?` is treated as a variable that supports rather than competes with specific values such as `h9`, ACME reports both `?proposition?` and `h9` as best maps for `w9`.

$Q\langle T\rangle(t)$ corresponds to an instantiation $F(h_i/x_i)$ of (unknown) in a formula F that is suggested by the maximally activated of nodes (w_i, x_i) and (w_i, h_i) for each x_i. ACME instantiates target variables by mapped elements.

The design decision that the mapping hypotheses of a particular example form the units is somewhat orthogonal to the common connectionist models. Due to some weaknesses of ACME this decision has recently been reconsidered in [67], and there units represent the predicates and objects of the source and the target the links between units represent the mapping hypotheses. Links are constructed between these units in accordance to the mapping constraints. The settled network provides a measure of the analogical correspondence of two units as the conditional probability that a target unit fires given that a particular source unit fires.

Reference to the Theory Section

Because of the different paradigms (connectionist nets vs. rule-based inference) it is difficult to compare ACME in detail with the inference schemas of section 3. At an abstract level, however, the pragmatic constraint of ACME can be considered as leading to a connection, ACME's mapping can be considered a reformulation, and formulas in ACME's knowledge representation – with maybe different instantiations of constant, relation, and proposition symbols – can be considered aspects in the framework presented in section 3. If Q is the goal of an analogical inference in ACME, then the conjunction of IMPORTANT-formulas forms an aspect P_1 that is subsumed by the P from a connection $[P, Q]$. The connection can be overgeneral if some determining information might be missing in the user-given IMPORTANT-formulas.

Several features cannot be captured by the inference schemas **CONN** or **REF**, in particular the interaction of different constraints. Furthermore, ACME is biased in favor of the structural relation/argument dependencies which have no equivalent in section 3. As a result, ACME's output extremely depends on the actual formalization of the problem and domain, as correctly pointed out by Hofstadter [58]. The influence of the pragmatic unit in ACME via inputs PRESUMED(...) merely respects presupposed mappings. This can be considered a technique to reduce the search space for mappings rather than a basic ingredient of analogical inference.

4.3 Analogical Reasoning with Prototypes

Two computational analogy systems, PROTOS [6] and TIBL (Typical-Instance-Based Learning System) [153], employ typical-instance-based analogical reasoning. Their cognitive psychological basis, the family resemblance view, that classifies instances according to their similarity to typical instances rather than to necessary and sufficient features has briefly been reviewed in section 2.2.

Purpose, Domains, Knowledge Representation

Bareiss and Porter built the system PROTOS that exploits analogical reasoning for the interactive assistance of heuristic classification in ill-defined domains. It begins as a novice and is taught explanations of classifications and of mistakes by a "teacher". PROTOS refers to the typicality of instances mainly for the retrieval of instances that serve as sources for the classification. It relies on cognitive psychological assumptions such as (typical-) instance-based categorization and the existence of a partial order on instances of categories induced by the typicality.

Zhang describes another instance-based classification system TIBL that relies on the family resemblance view as well. The key contribution of [153] is a computational method for measuring the typicality of an instance and a classification algorithm which stores only typical instances as potential analogs.

PROTOS has worked in clinical audiology which involves the evaluation and diagnosis of hearing disorders. TIBL has been applied successfully in an artificial domain, in the domain of congressional voting records, for malignant tumor classification, diabetes classification, and in the diagnosis of heart disease. In the following we concentrate on PROTOS.

In PROTOS, each exemplar of a category is represented by features and explanations of each feature's relevance to the category. Figure 15 contains exemplars of the category *chairs* with their explanations. Each node in this category structure is either an instance-containing category or a domain primitive (feature or exemplar). For instance, the node labeled *armrest* might contain exemplary armrests, each described by features and explanations.

There are three types of indexing knowledge: remindings, prototypicality, and exemplar differences. In Figure 16 the indices are shown as links from features to categories (remindings), links from categories to associated exemplars (degree of prototypicality), links from features to exemplars (remindings), and difference links from one exemplar to very similar exemplars.

Remindings estimate the likelihood of particular exemplars or categories being relevant to the processing of a target case. Remindings from a feature to a category, such as *backrest* indexing *chair* in Figure 16, suggest that the instances of the category have the feature. A reminding from a feature to an exemplar, such as *pedestal* indexing *chair1* in Figure 16, suggests that the exemplar will to some extend match cases described by this feature. Each reminding has an associated strength which is used to order the list of retrieved candidate source exemplars.

Prototypicality partially orders the exemplars within a category according to their record of success in previous classifications. Prototypicality estimates the psychological notion of

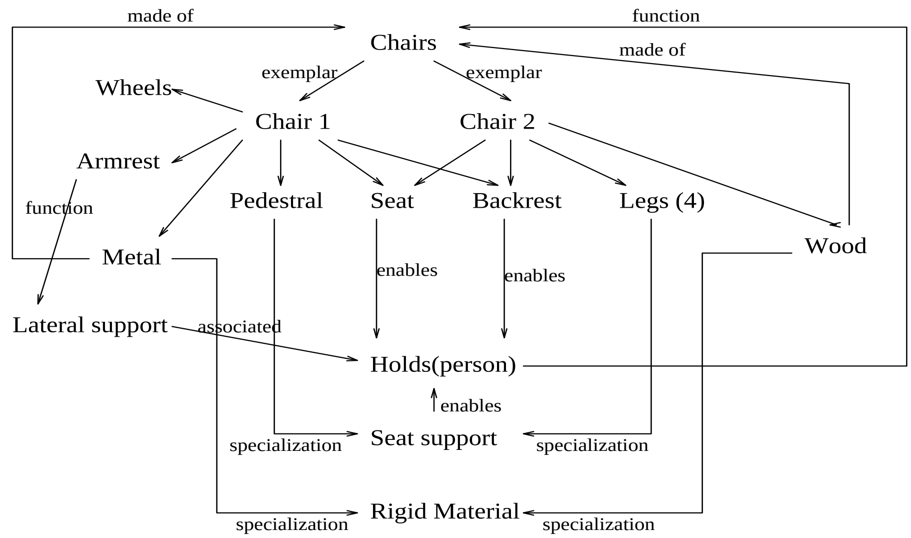


Figure 15: A category structure of PROTOS

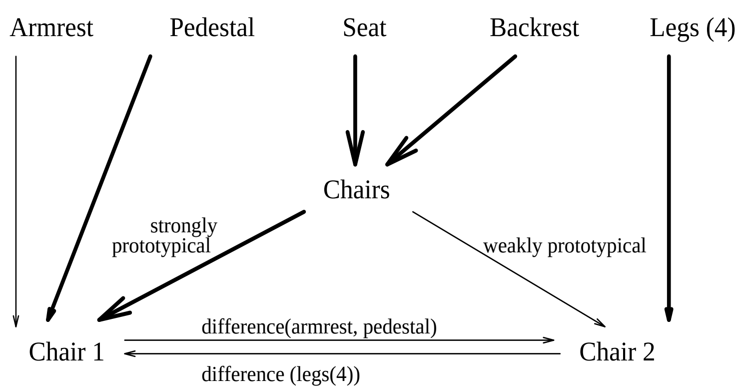


Figure 16: Indexing knowledge for *chair*. Unlabeled links are remindings

family resemblance (see section 2.2), which is the degree to which an exemplar is similar to other category members. In Figure 16 *chair1* is more prototypical than *chair2*.

The exemplar differences index exemplars by the feature that distinguish them from exemplars with similar descriptions. In Figure 16 the distinguishing features *armrest, pedestal* and *legs(4)* respectively, of *chair1* and *chair2* annotate the difference links.

PROTOS uses and communicates explanations which are expressed in a predefined language of definitional, causal, and correlational relations as given in Figure 17:

Definitional:

```
definitionally entails
is equivalent to
requires
if and only if
has generalization
part of
is mutually exclusive with
```

Causal:

```
causes
has function
enables
```

Correlational:

```
co-occurs with
is consistent with
implies
suggests
spurious to
```

Figure 17: PROTOS' explanation language

Top Level Algorithm of PROTOS

The first step of PROTOS' processing a new case is the retrieval, as shown in Table 5 taken from [7]:

---

1 Input a a target exemplar to be classified.
2 Retrieve remindings based on the target features and heuristically combine them.
repeat

> *repeat*
>
> 3 Use the strongest combined reminding to select a category and use prototypicality to select the most prototypical exemplar from the category.
>
> 4 *until* an adequately explained match is found

5 Use difference links to improve the classification by considering neighboring exemplars as alternatives.
6 Present the match and explanation to the teacher.
if the teacher disapproves
7 *then* request new domain knowledge from the teacher
8 Re-assess the remindings which suggested the exemplar
until the teacher approves. *if* the match is very strong
9 *then* merge the target with the exemplar.
10 *else* add the target to the category as a new exemplar. *if* the target was initially misclassified,
11 *then* install difference links to the instances involved.
until training is complete.

Table 5: The PROTOS algorithm

When searching for a source case, PROTOS first collects the remindings to categories. The features of a target case usually yield numerous "raw" remindings which then are heuristically combined to determine the most likely candidate classification for the target case. Related categories are combined by summing the strengths of duplicate remindings and by inheriting remindings to subcategories. Only the strongest combined remindings are returned. For instance, if the features of the target case include *backrest* and *pedestal*, PROTOS is reminded of the category *chair* and the exemplar *chair1*. The heuristic combination then yields a strong reminding to *chair1*. PROTOS can derive remindings (explanations) from the the domain theory. E.g., from the explanation

seat enables *holds person* which is the function of *chair*.

a reminding from *seat* to *chair* is derived.

The next step is to select an exemplar to serve as a source for the analogical classification. PROTOS may use prototypicality (line 3 of the algorithm in Table 5) to select an exemplar from a category in the set of candidate classifications. If there are no remindings to a particular exemplar, the most typical exemplar is selected. For example, if a target case evokes remindings to *chairs* but no remindings to particular exemplars, then PROTOS prefers *chair1* based on typicality.

Finally, PROTOS collects remindings from the target's features to particular exemplars and adds these to the list of exemplars (ordered by reminding strength). The result is a list of candidate exemplars which are tried to match with the target.

So far the remindings and prototypicality are used to retrieve an exemplar on expectation of similarity. If necessary, additional similarity is revealed by "knowledge-based pattern matching" in step 4 of the algorithm (explained below) that provides an explanation justifying the presence of features in the target case. For example, *legs(4)* in the target is pattern-matched to *pedestal* in the exemplar *chair1* using the knowledge

(*) *legs(4)* is a specialization of *seat support* and
pedestal is a specialization of *seat support*,

which suggests that the features *legs(4)* and *pedestal* can be matched because each provides evidence for *seat support*.

The difference indices are used in step 5 of the algorithm after PROTOS found a partial match of the target with a retrieved exemplar and noted mismatches of features. Then neighbors of the retrieved exemplars are considered as candidates for a better match. For instance, if the target case partially matches *chair2* but has the additional feature of *armrest*, then the difference link suggests *chair1*.

The eventually retrieved source exemplar S is used to classify the target T and the match between S and T is used to explain this classification. The classification and explanation is "discussed" with the teacher (steps 6-8 of Table 5) who may explain correct classifications, the *relevance* of features, and may provide domain knowledge for explanations. The teacher's interaction can result in changed remindings and in learning of prototypicality and exemplar differences.

Particular Computational Techniques

Several indexing techniques have been explained above. In addition, PROTOS can learn indexing knowledge for exemplars and concept structures from training by a teacher who provides instances to classify and approves classifications. The prototypicality of an exemplar is increased when it was successfully used for a classification. The more new cases successfully match an exemplar, the more typical it becomes.

PROTOS also acquires and uses matching knowledge, such as importance of features and match strength. PROTOS heuristically estimates the importance of features to a category by analyzing feature-to-category explanations. For instance, from the explanation relating *chair1*'s features in Figure 16 to the category *chairs*, it estimates that *seat* is an essential feature of a *chairs*, that *pedestal* is a moderately important feature of a *chairs*, and that *wheels* is a spurious feature of *chairs*. Internally, feature relevance is represented by a number between 0(spurious) and 1(important). The estimates can be revised if they yield a misclassification or an explanation rejected by the teacher.

PROTOS determines which features of an exemplar are important for a successful match. If an important feature of the retrieved exemplar does not belong to the target description, *knowledge-based pattern matching* attempts to infer it from the target features using knowledge. The knowledge-based matching is a heuristic search that begins from each unmatched exemplar feature and chains through the network of knowledge until reaching either a case feature or the depth bound. Each search step extends the current path with a relation.

A path connecting an exemplar feature with a target feature is an explanation of how the feature is "equivalent", in the sense that the features suggest the same classification.

The example (*) above showed how the features *legs(4)* and *pedestal* can be pattern-matched and, thereby, the presence of an important feature in the target is confirmed. That is, using knowledge, the similarity of the source and the target is more accurately estimated or, put differently, revealed. Knowledge-based pattern matching in PROTOS uses 38 domain-independent heuristics to evaluate the quality of a path with the purpose to find the strongest explanation and to prune weak ones.

Reference to the Theory Section

Corresponding to the typical-instance based analogical inference schema **TYP**, PROTOS may use the most prototypical instance S of a chosen category as a source if no remindings to a specific exemplar were found. The aspect Q is, of course, relevant for S because Q is $\mathsf{classification}$. If the target case reminds PROTOS of a particular exemplar, then the case's important features provide the aspect P which gives rise to a connection-based analogical inference with $Q = \mathsf{classification}$. Knowledge-based pattern matching can be seen as a reformulation that utilizes domain equivalences or implications. In this case, PROTOS employs an analogical inference including reformulation of P that must not effect the aspect $\mathsf{classification}$.

4.4 Case-Based Reasoning

So far we reviewed Prodigy/Analogy, ACME, and PROTOS and covered their purposes, knowledge representations, domains, top level algorithms, and some particular computational techniques. The references to our theoretical presentations in section 3 clearly show the differences of these systems in a systematic way. Prodigy/Analogy uses connection-based analogy with overgeneral connections and justification-based similarity; ACME employs connections explicitly given in the object description and explicitly represented similarity; PROTOS utilizes typical-instance based and connection-based analogy. The three systems address reformulation differently. Prodigy/Analogy finds type preserving symbol mappings, ACME focuses on inter-domain higher-order mappings, and PROTOS even utilizes knowledge-based reformulation.

Along several dimensions, analogical reasoning can be seen and used as a synonym to case-based reasoning (CBR), where "cases" play the role of objects. The term case-based reasoning was introduced by Schank. Frequently, case-based reasoning is considered as intra-domain analogy, which means that the source and the target belong to the same domain, as opposed to analogy across domains that is dealt with by some analogy systems. More importantly, most previous CBR research has a focus that is different from the main issues of analogy systems. Whereas most analogy research stressed the justification of analogical inferences and the mapping, most CBR research emphasized the organization of case bases, the retrieval and storage of cases, and some adaptation of source cases Kolodner [84], for instance, analyzes CBR as a general sequence of steps grouped into three phases: (1) building the case library, (2) retrieving cases, and (3) applying the retrieved cases to the target situation, and according to [1] the standard CBR cycle is Retrieve, Reuse, Revise, Retain.

The first so-called CBR system was CYRUS [82], a question-answering system with knowledge about travels and meetings of the US Secretary of State Cyrus Vance. The last decade has seen a steady grow of interest in CBR and a successful progress in CBR research and systems development [141]. Several commercial systems and CBR development tools emerged at the beginning of the 1990s, such as CLAVIER (Lockheed), CBR2 (Inference Corp.), ESTEEM (Esteem Software Inc.), KATE (Acknosoft), and S^3-CASE (tec Inno GmbH). CBR methodology proved to be most appropriate for classification tasks such as diagnosis, assessment, and help desk applications because classification mainly relies on case retrieval rather than adaptation. Fewer applications work for planning and design tasks which are more difficult in terms of adaptation.

The CBR paradigm covers a range of different methods for organizing, retrieving, and utilizing the knowledge stored in cases. Two different memory models (see section 2.2) are followed by CBR systems. The CYRUS system [82], e.g., is based on Schank's dynamic memory model [122] explained in section 2.2. The PROTOS system [8] is based on the exemplar view of categories that gives rise to a memory organization as presented in 4.3. Different representations of cases are described in a fixed or open vocabulary and are flat (sets of attribute-value pairs) or structured as, e.g., shown in Figure 18, where the bold words indicate objects and the italic indicate features. The features of **Hotel** do not apply to the object **Bungalow**.

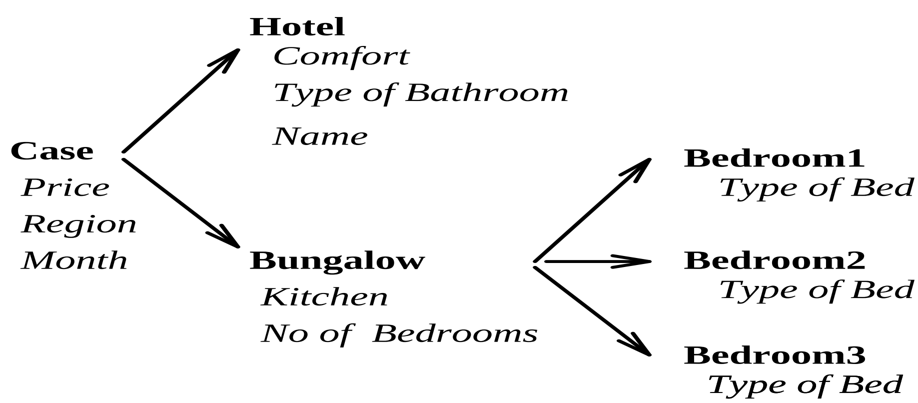


Figure 18: Example of a structured representation

Case indexing is an important issue of CBR applications that use a large case base because it selects a (small) subset of features of cases to speed up the retrieval and, thus, avoids to compare all cases from the case base with the new target case. In some academic systems, complex discrimination networks are used. An example is a discrimination network from CYRUS shown in Figure 19 that contains MOPs (see section 2.2) which store shared features of its cases and index those cases. Nodes in the network either hold single cases or are sub-MOPs that themselves create discrimination networks. In Figure 19, triangles represent questions, and labels on arcs represent answers. Boxes are MOPs.

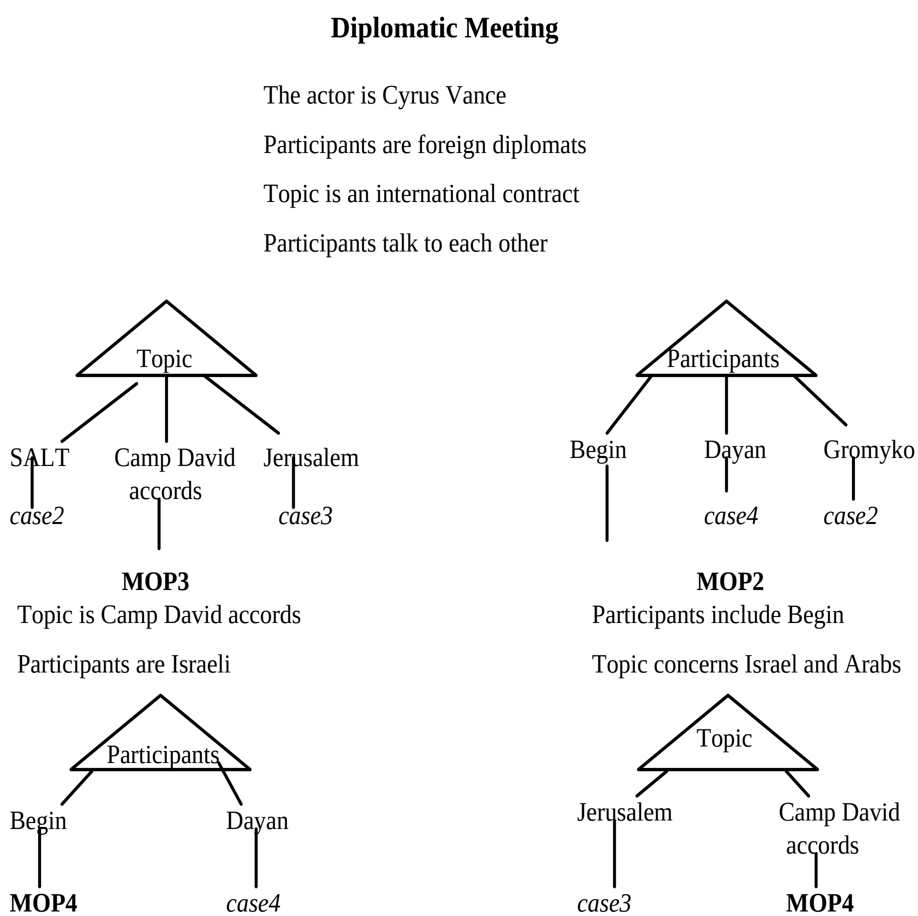


Figure 19: Discrimination Network from CYRUS

In real-world applications several methods developed in Machine Learning are employed for building index trees and retrieving cases, among them ID3 and k-d trees. A k-d tree is built by repeatedly partitioning the case base in a balanced way. Each node in the k-d tree represents a subset of cases in the case base. Every inner node partitions the set of cases represented by the node into two disjoint subsets using a discriminating feature. In Figure 20 (taken from [3]), for instance, A to I are cases in a 2-dimensional feature space (X and Y).

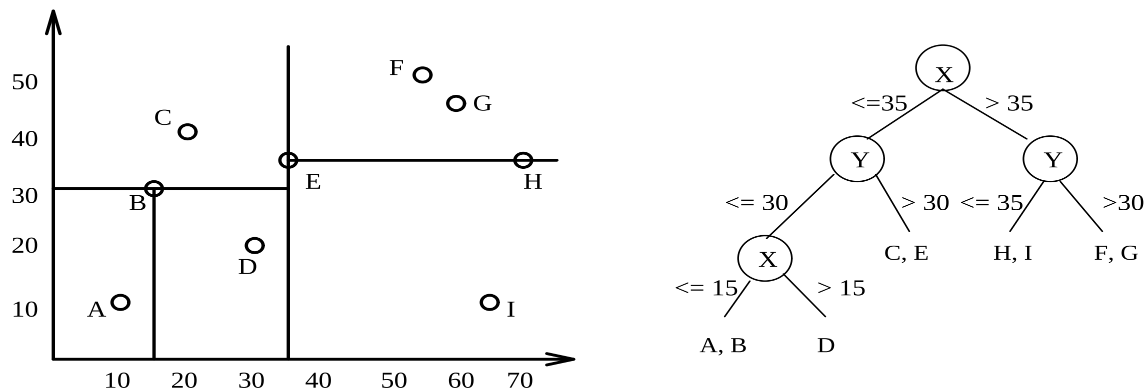


Figure 20: Example of a k-d tree

The bold lines show the partitioning during the creation of the k-d tree. The resulting k-d tree is shown at the right hand side. The method terminates when the partitions are smaller than a given threshold. The k-d tree mechanism can efficiently handle numeric features. It produces a well-balanced binary search tree.

ID3 automatically generates a decision tree from a set of training cases. It uses hill-climbing and a heuristic measure called "information gain" to build the tree. The information gain evaluates and picks the most discriminating relevant features as, e.g., in Figure 21, where in the root node it selects the feature *Month* because it yields the highest increase of the information gain measure which estimates how well a feature discriminates between classes with the same target feature (in the example the hotel name).

	Hotel	**Price**	**Region**	**Month**
Case 1	A	125	India	April
Case 2	B	200	Egypt	June
Case 3	C	175	India	April
Case 4	B	200	Egypt	May

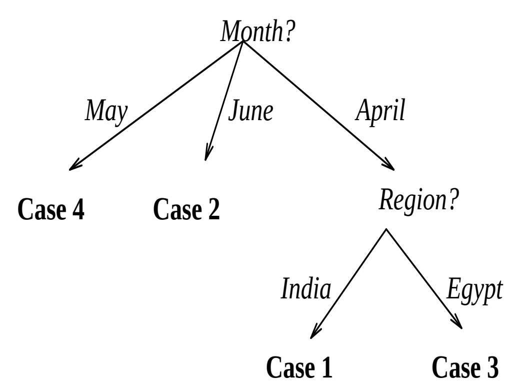


Figure 21: A decision tree build from a case base of hotels

As for the retrieval, various techniques for retrieval have been developed, including iterative, parallel, and constraint-satisfaction techniques. Several techniques for selecting the most relevant case(s) include different ranking schemes and similarity metrics, the overlap of salient features, and the importance of shared features with the target case. Knowledge-intensive selection methods typically generate explanations that support the ranking process. While some CBR systems retrieve a source case based on syntactical similarities of problems (e.g., CYRUS [83], PATDEX [114]), others attempt to include semantic similarity (e.g., PROTOS [8], CASEY [86]). Approaches with "retrieval for adaptation" have been suggested for the retrieval in [14] and [130] and they bear some resemblance to reformulation-based analogy.

A number of systems retrieve cases from indexed representations, for instance by the nearest neighbor technique applied to k-d trees or by ID3, which uses the decision tree built by ID3 from the case base in order to find the most similar cases. On the other hand, some systems scan the entire case base rather than an indexed memory, e.g., by nearest neighbor matching. The nearest neighbor matching computes a degree of match with some global similarity (or distance) function (see section 3) and ranks higher scoring cases before lower scoring ones. This is possible in domains where the costs of answering questions is minimal or the data base is small itself and, thus, the set of features has not to be restricted.

As for the reuse, copying and adaptation are possible subtasks. Recent discussions have shown that an appropriate adaptation is difficult and needs profound domain knowledge. Hence, few systems actually adapt former cases, e.g., CASEY [86].

When the solution generated by the retrieval and reuse is not correct, a revision is necessary. CHEF [53], e.g., utilizes causal knowledge to generate an explanation for a fault which is included into a failure memory. Then these explanations are used to modify the solution by adding steps into the plan. Such a revision is difficult and also relatively rare in CBR systems.

For more details and as a pointer to the comprehensive body of analogy research we refer the interested reader collections of papers in the analogy area [55, 148, 61]. A 1989 overview is given in Hall's [52]. For further details on CBR you might consult Kolodner's book [84], the overview of Aamodt and Plaza [1], the evaluation of commercial CBR tools [3], or the proceedings of the recent first international conference on CBR [141].

5 Analogy in Theorem Proving

Theorem proving by analogy, as sketched in Figure 22, finds a proof for a target theorem guided by a proof of a given source problem which is similar to the target theorem.

Mathematicians have clearly recognized the power of analogical reasoning in mathematical problem solving [51, 113]. However, analogy in theorem proving has received limited attention despite its importance in mathematics and the claims made for its usefulness in theorem proving in [11, 152].

5.1 The Need for a New Model

Kling's work [78] was one of the first attempts in theorem proving by analogy. His system essentially produces mappings between predicate symbols. These mappings are applied to

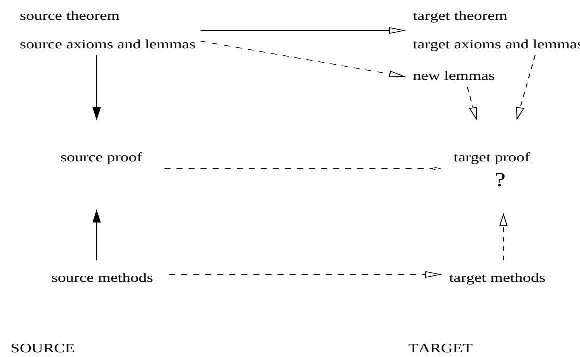


Figure 22: Analogy in theorem proving. Dashed arrows indicate the elaboration process.

the assumptions of the source proof to find the assumptions for the target proof.

Munyer [104] focuses on the formulas derived in each proof step. He applies a symbol mapping, constructed from the source and target theorem, to these formulas in order to obtain the derived formulas in the target proof.

Owen [109] thoroughly analyzed Kling's and Munyer's approaches to theorem proving by analogy and showed them inadequate even for many simple analogies, among others because their mapping was not sufficiently flexible. Owen's account emphasizes the matcher that recursively constructs symbol mappings and argument pairings. He transfers *single* calculus steps, i.e. binary resolution and paramodulation, from the source proof to provide steps in a target proof.

[16] starts with a user-provided initial symbol mapping and pairing of source lemmas with target lemmas and focuses on failed constructions of analogous proofs and on heuristics for patching target proofs. In the same intellectual tradition Bledsoe continued dealing with the debugging of analogies by his precondition prover [12]. More recently, [31] and [29, 81] employ certain second-order matchings in order to establish an analogical mapping and to transfer proofs.

In summary, computational accounts of theorem proving by analogy have been dominated by the idea of mapping symbols of the source theorem to symbols of the target theorem and employing an extended symbol map for transferring single proof steps of the source to the target.[20] Empirical investigations [94, 96], however, provided evidence that this idea does not appropriately cover many analogies drawn by mathematicians, not even all the analogies of a standard textbook [32]. The empirical studies have shown that:

- Methods, rather than just single calculus steps, are transferred analogically in mathematical theorem proving by analogy.
- For many of these analogies, symbol mapping is only insufficient. A change of a problem representation by, e.g., unfolding a definition, can be necessary to reveal the commonality of theorems or assumptions, upon which the analogical transfer is based. Other mappings of problems and proofs that go beyond symbol mapping are necessary as well. For some analogies, first the right level of *abstraction* has to be found before transferring a proof.

[20]Though Bledsoe transfers larger steps calling an automated prover but again uses symbol mapping and Plaisted [111] and Villafiorita [147] try to employ abstractions as a form of analogy.

- Many proofs by analogy result from transferring *parts* of the source proof to *parts* of the target proof, and some proofs by analogy transfer only the proof idea, while others transfer a detailed proof.

Melis [98] developed a new approach to theorem proving by analogy and explained why the *plan* level is an appropriate one for the analogical transfer. As shown in [95] this method can cope with the analogies in the mentioned textbook and is even able to deal with more complicated mathematical theorems [99] such as a non-trivial Heine-Borel theorem. This example has been proposed in [12] as a challenging example for theorem proving by analogy and could not be solved by previous approaches to analogy in theorem proving.

In the remainder of this section the analogy-driven proof-plan construction is presented, then the analogical transfer of the Heine-Borel proof-plan is described step by step.

5.2 Purposes, Knowledge Representation

In automated theorem proving analogy is useful for long and complex proofs that can be found by an automated theorem prover with user assistance only or for proofs that cannot be found automatically at all. In these cases the additional effort of storing and replaying a source pays by avoiding many user interactions or by providing a solution at all, e.g., by suggesting needed target lemmas or overriding search heuristics.

Proof planning has been introduced by Bundy [18]. A proof plan, as shown in Figure 23, is an abstract representation of a proof that is a tree (or a forest) of method nodes which are connected by sequents, called goals.

A *method* is a (partial) specification of a tactic, represented in a meta-level language, where a tactic executes a number of logical inferences [49]. The methods used later in this section have been defined in [65] and they are frame-like structures with pre- and postconditions just as the common planning operators. Similar to planning operators, a method M has the following slots: parameter, preconditions (pre(M)), postcondition (post(M)), constraints, proof schema and procedure. pre(M) is a set of sequents[21] from which the application of the method derives post(M) which is a set of sequents as well. pre(M) and post(M) serve as the input and output of methods in planning. The constraints are formulated in a meta language and serve to restrict the search during planning, e.g., restrictions of pre(M), post(M), or of the parameters. A standard program in the slot procedure executes the application of the proof schema. The proof schema is a declarative schematic representation of proofs in the object logic, relying on the Natural Deduction (ND) calculus and on invoking automated theorem provers such as OTTER [91]. The proof schema lines contain a label, a sequent, and a line-justification. The line-justification consists of the name of an ND-rule, the name of a prover, or LEMMA in case, the sequent is in pre(M). Additionally it may include supporting lines. For instance,

3. $\Delta \vdash F$ (IP;2)

describes that $\Delta \vdash F$ is derived from the sequent in line 2 of the proof schema by the ND-rule IP[22]. An example of a method is the following

[21] *Sequents* $\mathsf{P} = (\Delta \vdash F)$, are pairs of a set Δ of formulas and a formula F in an object language that is extended by meta-variables for functions, relations, formulas, sets of formulas, and terms.

[22] The IP rule is a combination of the ND-rules $\neg$I and $\neg$E that has been proved to be correct.

method: `Refutation`	
parameter	F: formula, Δ: set of formulas
preconditions	$(\Delta \cup \{\neg F\} \vdash \bot)$
postcondition	$(\Delta \vdash F)$
constraints	
proof schema	1. $\neg F \quad \vdash \neg F$ (HYP) 2. $\Delta, \neg F \quad \vdash \bot$ (LEMMA) 3. $\Delta \quad \vdash F$ (IP;2)
procedure	standard schema-interpreter

These methods differ mainly from those in [18] in that the tactic slot is replaced by a declarative proof schema *and* a procedure interpreting this schema[23]. The intention behind this difference is to enable reformulations of methods.

A method is *correct* if, given pre(M), then the method yields a correct proof of post(M) in case the constraints are satisfied. For verifying a line with a call of an automated theorem prover, e.g. OTTER, a time limit is set to prove the sequent.

Since mathematical proofs are constructed top down and bottom up, we consider backward *and* forward search in proof planning and define planning operators to be an f-method or a b-method respectively. In planning, f-methods get their preconditions as input and postconditions as output; b-methods work reversely. [24]. f- and b-methods will be treated differently by the analogy procedure.

Goals and assumptions are sequents, and a *proof-plan* is a forest. Each path in a tree consists of sequent nodes and method nodes that satisfy the *link condition* that says that a method node M follows a sequent node g and precedes the sequent nodes $\mathsf{g}_1,\ldots,\mathsf{g}_n$ if $\mathsf{g} \in \sigma(\text{post(M)})$ and $\sigma(\text{pre(M)}) = \{\mathsf{g}_1,\ldots,\mathsf{g}_n\}$ for a substitution σ of parameters.

Proof planning starts with a goal g and assumptions $(\emptyset \vdash F_i)$, where F_i are proof-assumptions, axioms, definitions, or lemmas. The proof planning proceeds by inserting methods and sequents: A b-method follows a goal and yields new (sub)goals as its successors. An f-method precedes assumptions and yields a new preceding assumption. Planning aims at reducing the gap between leaf goals and assumptions. Leaf goals that are not equal to an assumption are called *open goals.* As soon as a goal g_i equals an assumption, the two nodes collapse. Then g_i is no longer an open goal but *satisfied.* The planning terminates if there are no open goals.

The source proof-plans are trees with the source problem at the root, with no open goals, and with correct methods. The analogy procedure uses *linearized* proof-plans ordered by the sequence in which the nodes have been added to the plan. As in [144] justification structures, used to encode justifications for the decision made, annotate the plan nodes. These *justifications* capture the subgoaling structure of a plan and point to reasons for the choice, such as application conditions of a method, user-given guidance, or pre-programmed control knowledge. The correctness of a method is one of the justifications.

Reformulations are mappings ρ of a proof-plan to a proof-plan which usually, but not necessarily, preserve the correctness of methods in the plan. They encode mathematical

[23]Besides, the slots are renamed, e.g., our preconditions are named input there.

[24]For instance, the method corresponding to the ND-rule $\wedge$-elimination is a typical f-method, whereas the method $\wedge$-introduction is a typical b-method.

heuristics on how a proof-plan changes dependent on certain changes of the problem to be proved. Reformulations may insert subplans or replace methods and may change methods, sequents, and justifications of plan nodes.

For example, the reformulation `Add-Arguments` is applied in the example below. It is applicable if a function f is to be mapped to a function f', where arguments x_i of f are mapped to k arguments of f'; for instance, if the function $[-,-]$ that takes as arguments two real numbers x, y and returns the real interval $[x, y]$ has to be mapped to a function $[-,-,-,-]$ that takes as arguments four real numbers x, y, z, w and returns an interval $[x, y, z, w]$ in R^2, then `Add-Arguments` is applicable. The procedure `Add-Arguments` replaces the function f by the function f' with duplicated (i.e.,$k = 2$) arguments in goals, assumptions, justifications, and methods. It also yields additional related changes in the proof schema of effected methods such as replacing certain subformulas by conjunctions and duplicating certain lines of the proof schema which may also cause the duplication of related preconditions of the method (see [97] for more details). If a method's precondition P became duplicated, then `Add-Arguments` modifies the structure of the proof-plan by duplicating the subplan that yields the goal/assumption P in the original proof-plan.

For example, when `Add-Arguments` is applied to map the function $[-,-]$ to the function $[-,-,-,-]$, a proof-schema line

$$\emptyset \vdash a \in R \wedge b \in R \rightarrow clsdint([a, b])$$

to the line

$$\emptyset \vdash a \in R \wedge b \in R \wedge c \in R \wedge d \in R \rightarrow clsdint([a, b, c, d])$$

and a line

$$\emptyset \vdash \forall x \forall y \forall z \forall w (lf([x, y, z, w]) = x \wedge rt([x, y, z, w]) = y)$$

to

$$\emptyset \vdash \forall x \forall y \forall z \forall w (lf([x, y, z, w]) = x \wedge lf'([x, y, z, w]) = z \wedge rt([x, y, z, w]) = y \wedge rt'([x, y, z, w]) = w)$$

Even if it might seem that `Add-Arguments` was designed exactly for the Heine-Borel example explained in detail later, or that this particular example was chosen according to the reformulation, this is *not* the case. `Add-Arguments` proved to be a fairly frequent and general reformulation that has been used in several examples and was documented in [95, 65]. Moreover, the particular Heine-Borel example was chosen independently in [12].

5.3 Analogy-Driven Proof-Plan Construction

The general idea of analogy-driven proof-plan construction is to use the linearized source proof-plan as a guide for constructing an analogous target proof-plan, to reformulate the source plan, and to transfer methods, goals, and assumptions of a reformulated source proof-plan to the target proof-plan. The analogy-driven proof-plan construction is a derivational analogy, which has not been used for theorem proving before, extended by reformulation and bidirectional planning.

During the analogical transfer it is checked whether the justifications from the source node hold for the corresponding node in the target, e.g. whether the application conditions of a method hold in the target. Checking the justifications makes it possible to consider control information that is not available in the source and target problems, but relevant to the proof process.

The analogy-driven proof-plan construction employs reformulations that aim at matching a source goal with a target goal or as many preconditions of a source f-method with target assumptions respectively. These reformulations change the proof-plan. Table 6 shows the top-level procedure of the analogy-driven proof-plan construction. Given a linearized source proof-plan, target assumptions, and a target goal, the output of the procedure is a target proof-plan.

---

input: linearized source plan, (open) target goal, confidence level m
output: (linearized) target plan

1. **while** there are open target goals **do**
2. **if** source plan is exhausted, **then** base-level plan for the open goals.
3. Get next sequent P from source plan. The sequent is either an assumption or a goal. **if** P is an assumption, **then** go to 5.
4. **if** there is a reformulation ρ, such that $\rho\mathsf{P}$ matches an open target goal g_T for which the justifications hold, **then**
   - reformulate source plan by ρ and transfer g_T to the target plan.
   - Select from the reformulated source the relevant b-method M.
   - Check M's justifications. **if** they hold, **then** transfer M to the target plan and update open goals.
   - **if** justification does not hold, **then** choose suitable action:
     - Advance the source if M is redundant,
     - Try to establish the justifications, or
     - Base-level plan.
5. Select from the reformulated source the relevant f-method M.
6. **if** there is a reformulation ρ' left such that $|missing(\rho'\mathrm{M})| < m$ and that justifications hold for the matched target assumptions **then**
   - reformulate source plan by (best) ρ' and link the matched target assumptions to source plan.
   - Check M's justifications.
   - **if** justification does not hold, **then** choose suitable action as above.

Table 6: Outline of the analogy-driven proof-plan construction

Step 4 is relevant for a planner with backward search only whereas step 5 and 6 apply to the treatment of forward planning. The two branches differ mainly in that ρ aims at matching a source goal to one target goal, whereas ρ' aims at matching as many source assumptions as possible to target assumptions. $|missing(\rho'\mathrm{M})| < m$,[25] means that less than m preconditions of the currently treated reformulated f-method M do not match a target assumption. The sequents of *missing*(ρ'M) become new open goals if ρ' is an acceptable reformulation. m is a procedure parameter which indicates the confidence in an analogical transfer despite missing target lemmas.

The first goal, usually the source problem P_S, is chosen. If P_S can be reformulated by a ρ such that it matches the target problem P_T, then ρ will be applied to the (current) source

[25] where *missing(M)* := set of preconditions of M that do not match a current target assumption,

plan and the method M with post(M) $= \rho \mathsf{P}_S$ is a candidate for the transfer to the target plan. If the original justifications hold for M in the target, then M is transferred and the open goals are updated by removing post(M) and introducing all sequents from pre(M) as open goals. The procedure is repeated if there are open goals and if some source plan node has not yet been visited.

After the reformulation, different actions can be taken, if a justifications does not hold for M in the target. The actions depend on the kind of violated justification. For instance, a reformulated method might not be correct because reformulations do not necessarily preserve correctness. If the correctness-justification does not hold for M, then the procedure tries to *modify* M to a correct method M'. One possible modification technique decomposes M such that a correct *sub*method M' results with post(M)= post(M') and M' replaces M creating an additional open goal. If a precondition of M does not hold in the target, then it can be made an open goal that has to be proved by base-level planning, i.e., planning without guidance by analogy. As in Prodigy/Analogy another action is taken if the current goal is satisfied in the target already. Then M is superfluous in the target and can be skipped.

When all methods of the source plan have been visited, then the remaining open goals have to be proved by base-level planning.

To illustrate the procedure we explain a challenging example next. This example is presented in [97] with all details.

5.4 The Heine-Borel Example

THEOREM: *[Heine-Borel-1 (HB1)] If a closed interval [a,b] of R^1 is covered by a family G of open sets (in R^1), then there is a finite subfamily H of G which covers [a,b].*[26] ■

Formalized: $\{a \in R \wedge b \in R \wedge a \leq b, \forall B(B \in G \rightarrow open(B)), [a,b] \subset \cup G\}$
$\vdash \exists H(H \subseteq G \wedge finite(H) \wedge [a,b] \subset \cup H)$
Abbreviated: $\Delta \vdash Th$

THEOREM: *[Heine-Borel-2 (HB2)] If a closed rectangle [a,b,c,d] of R^2 is covered by a family G of open sets (in R^2), then there is a finite subfamily H of G which covers [a,b,c,d].* ■

Formalized: $\{a \in R \wedge b \in R \wedge c \in R \wedge d \in R \wedge a \leq b \wedge c \leq d, \forall B(B \in G \rightarrow open(B)), [a,b,c,d] \subset \cup G\} \vdash \exists H(H \subseteq G \wedge finite(H) \wedge [a,b,c,d] \subset \cup H)$.
Abbreviated: $\Delta' \vdash Th'$

Figure 23 shows the proof-plan for HB1 which, in fact, yields a proof of HB1 when executed. The digital annotations of the methods indicate the order in which the methods have been introduced into the proof-plan.

One of its (correct) methods is method-111, where the relations $finite, open, clsdint, \subset, \subseteq$ have the obvious meanings (finite, open, closed-interval, covers, subset) and the function symbols $rt, lf, [-,-], \cup$ are interpreted as right-margin-of, left-margin-of, interval-with-upper-and-lower-bound, and union, respectively. L3 and L4 denote proof assumptions (lemmas).

[26] R^1 denotes the set of sets of real numbers and R^2 denotes the set of sets of ordered pairs (x,y) of real numbers x and y.

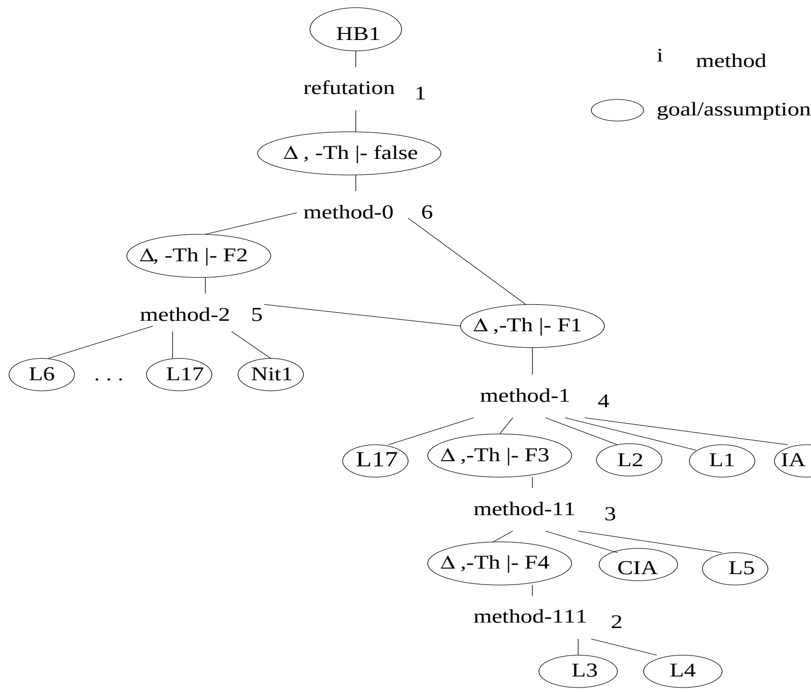


Figure 23: The proof-plan of HB1 with annotated method nodes

method: `-111`	
parameter	$a, b, G, clsdint, open, rt, lf, [-,-], finite$
preconditions	L3,L4
postcondition	$(\Delta, \neg Th \vdash F4)$
constraints	
proof schema	0-1 $\Delta \vdash a \in R \wedge b \in R \wedge a \le b$ (HYP) 1. $\Delta \vdash a \in R \wedge b \in R$ ($\wedge$E,0-1) 2. $\Delta \vdash a \in R \wedge b \in R \rightarrow clsdint([a,b])$ ($\forall$E;L3) 3. $\Delta \vdash clsdint([a,b])$ ($\rightarrow$E,1,2) 4. $\{\neg Th\} \vdash \neg\exists H(H \subseteq G \wedge finite(H) \wedge [a,b] \subset \cup H)$ (HYP) 5. $\Delta \vdash lf([a,b]) \le rt([a,b])$ (OTTER;L4, 1) 6. $\Delta \vdash [a,b] \subset \cup G$ (HYP) 7. $\Delta, \neg Th \vdash F4$ ($\wedge$I,3,4,5,6)
procedure	schema-interpreter

Proving HB2 by Analogy

Among the lemmas L1... L17, Nit1, CIA, IA used in the proof of HB1, several lemmas, namely L1, L2, L6, L8, L10, L12, L13, L14, L17 are potential lemmas for HB2 as well because they contain only symbols not specific for R^1. m is set to a large number because many lemmas are missing for HB2.

The choice of the reformulations is assisted by a user-supplied connection-table `CT`, which contains connections of the kinds, ([-,-] [-,-,-,-]), (*clsdint clsdrect*). (Here [-,-] and [-,-,-,-] are written for the functions $\lambda x \lambda y.[x,y]$ and $\lambda x \lambda y \lambda z \lambda w[x,y,z,w]$). `CT` restricts the search for reformulations and reduces the number of parameters that have to be instantiated to be able to prove suggested target lemmas after the analogy procedure.

The source proof-plan is reformulated stepwise along with a step by step transfer of correct methods:

1. The source goal HB1 has to be reformulated such that it matches HB2. The reformulation consists of the `Symbol-Mapping` instantiating the parameters $a, b, G, open, finite$ to the constants $a, b, G, open, finite$ of HB2 and of `Add-Argument` which inter alia replaces the binary function $[-,-]$ by the 4-ary function $[-,-,-,-]$ and introduces the additional parameters lf', rt' and constants c, d. No duplication of subplans occurs in this application of `Add-Argument`. The `Refutation` method stays unchanged but all other methods are changed by the reformulation.

2. Next, the source assumption L3 (reformulated in step 1) leads the f-method method-111'. No lemmas corresponding to L3, L4 are given for HB2 but, since m is large, this does not matter. The parameter $clsdint, lf, rt, lf', rt'$ occur in the reformulated L3, L4, thus the `Symbol-Mapping` $clsdint \Rightarrow clsdrec$ is forced by `CT` while lf, rt, lf', rt' cannot be instantiated and remain parameters. method-111 is reformulated to the correct method-111' by the reformulations of step 1 and 2:

method: `-111'`	
parameter	rt, lf, rt', lf'
preconditions	L3',L4'
postcondition	$(\Delta', \neg Th' \vdash F4')$
constraints	type(rt)=type(rt'), type(lf)=type(lf')
proof schema	0-1 $\Delta' \vdash a \in R \wedge b \in R \wedge a \leq b \wedge c \in R \wedge d \in R \wedge c \leq d$ (HYP) 1. $\Delta' \vdash a \in R \wedge b \in R \wedge c \in R \wedge d \in R$ ($\wedge$E,0-1) 2. $\Delta' \vdash a \in R \wedge b \in R \wedge c \in R \wedge d \in R \rightarrow clsdrec([a,b,c,d])$ ($\forall$E;L3') 3. $\Delta' \vdash clsdrec([a,b,c,d])$ ($\rightarrow$E,1,2) 4. $\{\neg Th'\} \vdash \neg \exists H(H \subseteq G \wedge finite(H) \wedge [a,b,c,d] \subset \cup H)$ (HYP) 5. $\Delta' \vdash lf([a,b,c,d]) \leq rt([a,b,c,d]) \wedge lf'([a,b,c,d]) \leq rt'([a,b,c,d])$ (OTTER;L4', 1) 6. $\Delta' \vdash [a,b,c,d] \subset \cup G$ (HYP) 7. $\Delta', \neg Th' \vdash F4'$ ($\wedge$I,3,4,5,6)
procedure	schema-interpreter

3. The next (previously reformulated) source sequent is $(\Delta', \neg Th' \vdash F4')$ and the corresponding f-method is method-11'. No lemmas corresponding to L5',CIA' are given in the target which does not matter because of the large m. No reformulations are necessary. method-11' is correct.

4. Essentially the same is true for the next assumption and method-1'.

5. Now it is method-2's turn. *center* occurs in the preconditions of method-2' and remains a parameter since no lemma to match L7 or L9 is given in the target. No reformulation takes place.

- Checking the justifications of method-2' it turns out that method-2' is not correct because line5 of the proof schema is not.
  In order to establish the justification method-2' is decomposed into a plan consisting of the correct method-21' with post(method-2') = post(method-21'), the subgoal $\mathtt{g'}_{5a}$ which is in pre(method-2'), and a not correct method-22' with post(method-22') = $\mathtt{g'}_{5a}$.
- method-21' can be transferred to the target and $\mathtt{g'}_{5a}$ remains an open goal.

6. $(\Delta', \neg Th' \vdash F2)$ is the next source assumption and leads to method-0'. No further reformulation happens in this step and method-0' is correct.

Note that more techniques could be developed for automatically modifying the not correct target methods such as abduction [12] which automatically provides additional preconditions in order to prove target methods correct and which has handled the proof of a difficult completeness theorem.

The Resulting HB2 Proof-Plan

The analogy procedure yields the plan, shown in Figure 24, with reformulated goals, assumptions and methods proved to be correct. The target plan for HB2 looks like that for HB1 except that some new lemmas $L3' \ldots$ replace $L3 \ldots$ and all methods but method-2 are replaced by the corresponding reformulated methods. method-2 is replaced by its reformulated submethod method-21'.

Actually, this proof-plan has been produced by manually choosing the reformulations but applying them automatically. The analogy procedure suggested as open goals the lemmas L3', L4', L5', L7', L9', L11', L17', Nit2, CIA', IA', and g'_{5a} which are left to be proved in the target.

For Bledsoe's original formalization of the Heine-Borel theorems, given above, that contains the functions [-,-] and [-,-,-,-] and that assumes L1, L2, L8, L15, L17 to be target lemmas, symbol mapping is not sufficient for the reformulation of the HB1 proof plan. This forces the arguments of [-,-] to be duplicated and, thus, a reformulation `Add-Arguments` has to be applied that changes the proof plan (and methods). Just symbol mapping [-,-] to [-,-,-,-] without the other changes provided by `Add-Arguments` would yield a target proof-plan (or proof) that lacks necessary features and this would imply many breakdowns of the analogical replay or ad hoc repairs as, e.g., practiced in [139].

Experiments with other Representations of HB

The robustness of the analogy-driven proof-plan construction to different formulations of theorems is an important issue. Actually, we looked at three different formulations when dealing with the Heine-Borel examples[27]:

1. Let the problem formalized without the functions [-,-] and [-,-,-,-], i.e.,
   HB1: $\ldots \vdash clsdint(I) \rightarrow \exists H(H \subseteq G \wedge finite(H) \wedge D \subset \cup H)$ and HB2:
   $\ldots \vdash clsdrec(I) \rightarrow \exists H(H \subseteq G \wedge finite(H) \wedge D \subset \cup H)$,
   and the some source assumption

[27] These examples were not proceeded in detail though.

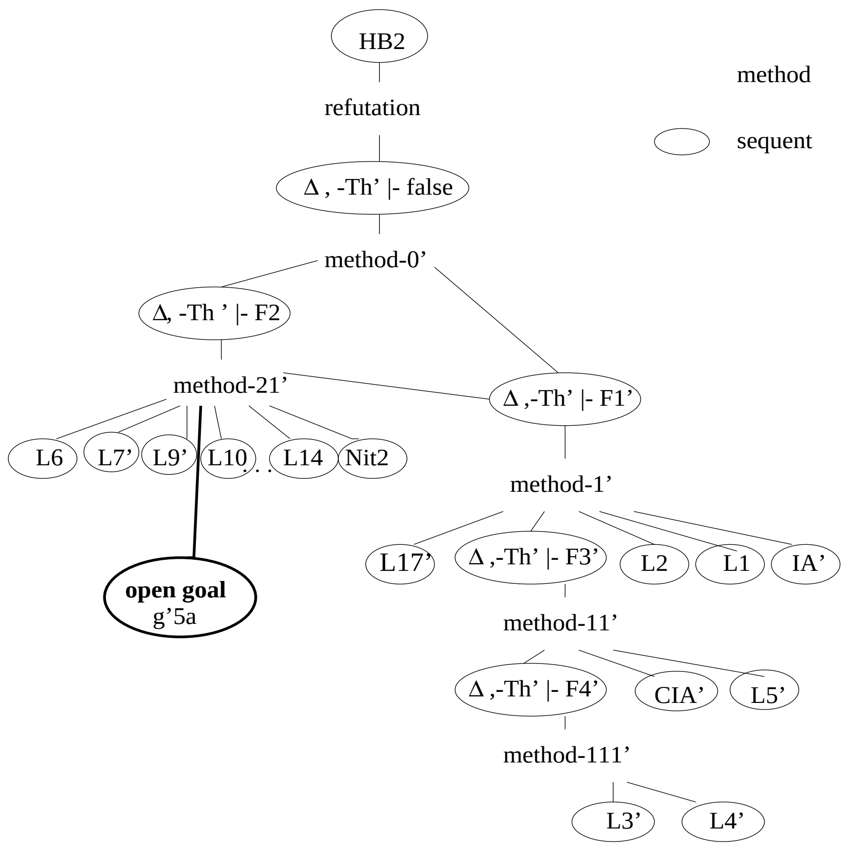


Figure 24: The proof-plan of HB2

$\emptyset \vdash \forall D(clsdint(D) \rightarrow \exists x, y(D = [x, y] \wedge x \in R \wedge y \in R))$ and the target assumption
$\emptyset \vdash \forall D(clsdrec(D) \rightarrow \exists x, y, z, w(D = [x, y, z, w] \wedge x \in R \wedge y \in R)) \wedge z \in R \wedge w \in R$.

When the function $[-,-]$ occurs first in the proof-plan, the connection table forces its mapping to $[-,-,-,-]$ and this triggers the reformulation `Add-Arguments` only later than in the first analogy step. The first occurrence of the function $[-,-]$ depends, however, on the actual proof of HB1.

2. If the formalizations of HB1 and HB2 have superficially different assumptions or conclusions as in
HB1: $\ldots clsdint(I) \vdash \exists H(H \subseteq G \wedge finite(H) \wedge D \subset \cup H)$
HB2: $\ldots \vdash \exists H(H \subseteq G \wedge finite(H) \wedge [a, b, c, d] \subset \cup H)$,
then a normalizing reformulation (not discussed in this paper), has to be applied to the source plan in order to match HB1 with HB2. Such a normalization introduces an additional method at the top of the source proof-plan that applies the definition of $clsdint(X)$ and yields a postcondition that can be matched with HB2.

3. Using $length(D)$ instead of $rt(D) - lf(D)$ in the assumptions caused problems because the analogy procedure suggests a target lemma
L9'= $\emptyset \vdash \forall x \forall u \forall v \forall s \forall t \forall c \forall d \forall e \forall f(u \in R \wedge v \in R \wedge s \in R \wedge t \in R \wedge c \in R \wedge d \in R \wedge e \in R \wedge f \in R \wedge u \leq v \wedge s \leq t \wedge c \leq d \wedge e \leq f \rightarrow (center(x, [u, v, s, t]) \wedge x \in [c, d, e, f] \wedge$
$length([c, d, e, f]) \leq length([u, v, s, t])/2 \wedge length([c, d, e, f]) \leq length([u, v, s, t])/2 \rightarrow [c, d, e, f] \subset$

$[u, v, s, t]))$
No definition of *length* was found that can satisfy L9' in the target for fixed definitions of $\leq$, division, etc. as functions and relations of R^1.

5.5 Reference to the Theory Section

The analogy-driven proof-plan construction employs both the derivational and the transformational paradigm by replaying the decisions of the source planning process and by transforming the source proof plan via reformulations.

Analogy-driven proof-plan construction implicitly realizes a connection-based analogical inference with reformulations, rather than applying the inference schema REF. In terms of aspects, the representation of the proof-plan corresponds to the aspect Q and all the proof-relevant facts conjunctively buid the aspect P.

As for derivational analogy in general, the proof-relevant information that is not part of the problem description, but acquired during the source solving process, is encoded in the justifications annotating the plan nodes. The correspondence of the justifications in source and target can only be established during the analogical replay and is done by checking the justifications. Compared with general derivational analogy the analogy-driven proof-plan construction additionally utilizes reformulations which heuristically preserve the connection between (problem + justification) and proof-plan.

6 Acknowledgement

This work was supported in part by the NATO grant ***, by the Max Kade Foundation, and by ***.

References

[1] A. Aamodt and E. Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications*, 7:39–59, 1994.

[2] C.M. Allwood and M. Eliasson. Analogy and other sources of difficulty in novices' very first text-editing. *Int.J.Man-Machine Studies*, 27:1–22, 1987.

[3] K-D. Althoff, E. Auriol, R. Barletta, and M. Manago. *A Review of Industrial Case-Based Reasoning Tools.* AI Intelligence, Oxford, 1995.

[4] Aristotle. *Prior Analytics. In: The Organon I.* Harvard University Press, Cambridge, MA, 1838.

[5] F. Bacon. The great instauration: Novum organum, or directions concerning the interpretation of nature. In E.A. Burtt, editor, *English Philosophers from Bacon to Mill*, pages 25–132. Cambridge University Press, 1994 orig. 1620.

[6] E.R. Bareiss, B.W. Porter, and C.C. Wier. Protos: An exemplar-based learning apprentice. In P. Langley, editor, *Proceedings of the fourth International Workshop on Machine Learning*, pages 12–23. Morgan Kaufmann, 1987.

[7] E.R. Bareiss, B.W. Porter, and C.C. Wier. Protos: An exemplar-based learning apprentice. *International Journal of Man-Machine Studies*, 29:549–561, 1988.

[8] R. Bareiss. *Exemplar-based Knowledge Acquisition: A Unified Approach to Concept Representation, Classification, and Learning.* Academic Press, Boston, 1989.

[9] S. Bhansali and M.T. Harandi. Synthesis of UNIX programs using derivational analogy. *Machine Learning*, 10, 1993.

[10] A. Biela. "good" and "surfaces" analogies in scientific inquiries. *Science of Science*, 5(3-4):327–346, 1985.

[11] W.W. Bledsoe. The use of analogy in automatic proof discovery. Tech.Rep. AI-158-86, Microelectronics and Computer Technology Corporation, Austin, TX, 1986.

[12] W.W. Bledsoe. A precondition prover for analogy. *BioSystems*, 34:225–247, 1995.

[13] B. Blumenthal. *Replaying Episodes of a Metaphor Application Interface Designer.* PhD thesis, University of Texas, Artificial Intelligence Lab, Austin, 1990.

[14] K. Börner. Structural similarity as guidance in case-based design. In *First European Workshop on Case-Based Reasoning 1993*, volume LNAI, Berlin, 1994. Springer Verlag.

[15] K. Börner. Towards formalization in case-based reasoning for synthesis. In D. Aha, editor, *AAAI-94 Workshop on Case-Based Reasoning*, Seattle, 1994.

[16] B. Brook, S. Cooper, and W. Pierce. Analogical reasoning and proof discovery. In E. Lusk and R. Overbeek, editors, *Proc. 9th International Conference on Automated Deduction (CADE)*, volume 310 of *Lecture Notes in Computer Science*, pages 454–468, Argonne, 1988. Springer.

[17] L.R. Brooks. Nonanalytic concept formation and memory instances. In E. Rosch and B.B. Lloyd, editors, *Cognition and Categorization*, pages 169–215. Erlbaum, Hillsdale NJ, 1978.

[18] A. Bundy. The use of explicit plans to guide inductive proofs. In E. Lusk and R. Overbeek, editors, *Proc. 9th International Conference on Automated Deduction (CADE)*, volume 310 of *Lecture Notes in Computer Science*, pages 111–120, Argonne, 1988. Springer.

[19] B.D. Burns. Goal similarity in analogical problem solving. In *Proceedings of the Twelvth Annual Conference of the Cognitive Science Society*, Hillsdale, NJ, 1990.

[20] K. Busch, C. Christmann, J. Kurth, and C. Lankow. Bionik liefert Problemlösung. *Wissenschaft und Fortschritt*, 34(8):216–219, 1984.

[21] J.G. Carbonell. Learning by analogy: Formulating and generalizing plans from past experience. In R.S. Michalsky, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 371–392. TiogaPubl., Palo Alto, 1983.

[22] J.G. Carbonell. Learning by analogy: Formulating and generalizing plans from past experience. In R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 137–161. Springer, 1984.

[23] J.G. Carbonell. Derivational analogy: A theory of reconstructive problem solving and expertise acquisition. In R.S. Michalsky, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 371–392. Morgan Kaufmann Publ., Los Altos, 1986.

[24] J.R. Carbonell and A.M. Collins. Natural semantics ini artificial intelligence. In *Proceedings of the Third International Joint Conference on Artificial Intelligence*, Stanford, CA, 1973.

[25] C.A. Clement. Observed methods for generating analogies in scientific problem solving. *Cognitive Science*, 12:563–586, 1988.

[26] C.A. Clement and D. Gentner. Systematicity as a selection constraint in analogical mapping. *Cognitive Science*, 15:89–132, 1991.

[27] J. Clement. Analogy generation in scientific problem-solving. In *Proceedings of the Third Annual Conference of the Cognitive Science Society*, 1981.

[28] J. Clement. Analogical reasoning patterns in expert problem solving. In *Proceedings of the Forth Annual Conference of the Cognitive Science Society*, pages 79–81, 1982.

[29] R. Curien. *Outils pour la Preuve par Analogie.* PhD thesis, Universite Henri Poincare - Nancy, 1995.

[30] T.R. Davis and S.J. Russell. A logical approach to reasoning by analogy. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, pages 264–270, Milan Italy, 1987. Morgan Kaufmann.

[31] T. Boy de la Tour and R. Caferra. Proof analogy in interactive theorem proving: A method to express and use it via second order pattern matching. In *Proceedings of the AAAI-87*, pages 95–99, 1987.

[32] P. Deussen. *Halbgruppen und Automaten*, volume 99 of *Heidelberger Taschenbücher*. Springer, 1971.

[33] K. Duncker. On problem solving. *Psychological Monographs*, 58, 1945.

[34] T.C. Eskridge. A hybrid model of continuous analogical reasoning. In K.J. Holyoak, editor, *Analogical Connections*, pages 207–246. Ablex, NJ, 1994.

[35] T.G. Evans. A program for the solution of geometric analogy intelligence test questions. In M.Minsky, editor, *Semantic Information Processing.* 1986.

[36] B. Falkenhainer. *Learning from Physical Analogies: A Study in Analogy and the Explanation Process.* PhD thesis, University of Illinois, Urbana-Champaign, IL, 1989.

[37] B. Falkenhainer, K.D. Forbus, and D. Gentner. The Structure-Mapping Engine. In *Proceedings of the AAAI-86*, Philadelphia, PA, 1986.

[38] R.E. Fikes and N.J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[39] D. Gentner. Studies of metaphor and complex analogies. In *Symposium on Metaphor as Process, American Psychological Society*, Montreal, 1980.

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

[41] D. Gentner. The mechanisms of analogical learning. In S. Vosniadou and A. Ortony, editors, *Similarity and Analogical Reasoning*, pages 199–241. Cambridge University Press, 1989.

[42] D. Gentner, K.D. Forbus, and R.W. Ferguson. Incremental structure-mapping. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, pages 313–318, Hillsdale, NJ, 1994.

[43] D. Gentner and D.R. Gentner. Flowing waters or teeming crowds: Mental models of electricity. In D. Gentner and A. Stevens, editors, *Mental Models*, pages 95–130. Lawrence Erlbaum, 1983.

[44] D. Gentner and R. Landers. Analogical reminding: A good match is hard to find. In *Proceedings of the International Conference on Systems, Man, and Cybernetics*, Tucson AZ, 1985.

[45] R.W. Gibbs. *The Poetics of Mind: Figurative Thought, Language, and Understanding.* Cambidge University Press, Cambridge, 1994.

[46] M.L. Gick and K.J. Holyoak. Analogical problem solving. *Cognitive Psychology*, 12:306–335, 1980.

[47] M.L. Gick and K.J. Holyoak. Schema induction and analogical transfer. *Cognitive Psychology*, 15(1):1–38, 1983.

[48] R. Goebel. A sketch of analogy as reasoning with equality hypotheses. In K.P. Jantke, editor, *Analogical and Inductive Inference*, volume 397 of *Lecture Notes on Computer Science*, pages 243–253. Springer, 1989.

[49] M. Gordon, R. Milner, and C.P. Wadsworth. *Edinburgh LCF: A Mechanized Logic of Computation.* Lecture Notes in Computer Science 78. Springer, Berlin, 1979.

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

[51] J. Hadamard. *The Psychology of Invention in the Mathematical Field.* Princeton Univ. Press, Princeton, 1945.

[52] R.P. Hall. Computational approaches to analogical reasoning: A comparative analysis. *Artificial Intelligence*, 39(1):39–120, 1989.

[53] K. Hammond. CHEF: A model of case-based planning. In *Proceedings Fifth National Conference on Artificial Intelligence*, pages 267–271, Los Altos, CA, 1986. Morgan Kaufmann.

[54] M. Haraguchi and S. Arikawa. Reasoning by analogy as a partial identity between models. In K.P. Jantke, editor, *Analogical and Inductive Reasoning*, volume 265 of *Lecture Notes of Computer Science*, pages 61–87. Springer, 1987.

[55] D. Helman. *Analogical Reasoning.* Kluwer Academic Publisher, 1988.

[56] M. Hesse. *Models and Analogies in Science.* Indiana: University of Notre Dame Press, Notre Dame, 1966.

[57] Angela K. Hickman and Jill H. Larkin. Internal analogy: A model of transfer within problems. In *The 12th Annual Conference of The Cognitive Science Society*, pages 53–60, Hillsdale, NJ, 1990. Lawrence Erlbaum Associates.

[58] D. Hofstadter. A review of mental leaps: Analogy in creative thought. *AI Magazine*, 16(3):75–80, 1995.

[59] D.R. Hofstadter and M. Mitchell. An overview of the copycat project. Tech Report CRCC-52-1991, Indiana University Bloomington, Indiana, 1991.

[60] K.J. Holyoak. The pragmatics of analogical transfer. In G.H. Bower, editor, *The Psychology of Learning and Motivation.* Academic Press, 1985.

[61] K.J. Holyoak. *Analogical Connections, Vol II.* Ablex, NJ, 1994.

[62] K.J. Holyoak and K. Koh. Surface and structural similarity in analogical transfer. *Memory & Cognition*, 15(4):332–340, 1987.

[63] K.J. Holyoak and P. Thagard. Analogical mapping by constraint satisfaction. *Cognitive Science*, 13:295–355, 1989.

[64] K.J. Holyoak and P. Thagard. *Mental Leaps: Analogy in Creative Thought.* Bradford, Cambridge, MA, 1995.

[65] X. Huang, M. Kerber, M. Kohlhase, and J. Richts. Methods - the basic units for planning and verifying proofs. In *Proceedings of Jahrestagung für Künstliche Intelligenz*, Saarbrücken, 1994. Springer.

[66] Ye Huang. *Relevancy Knowledge and Analogical Reasoning.* PhD thesis, UMIST, UK, 1993.

[67] J.E. Hummel, B. Burns, and K.J. Holyoak. Analogical mapping by dynamic binding: Preliminary investigations. In K.J. Holyoak, editor, *Analogical Connections*, pages 416–445. Ablex, NJ, 1994.

[68] B. Indurkhya. Approximate semantic transference: A computational theory of metaphors and analogies. *Cognitive Science*, 11:445–480, 1987.

[69] Bipin Indurkhya. Modes of analogy. In K.P. Jantke, editor, *Proceedings of the Workshop on Analogical and Inductive Inference 1989*, pages 217–230, Reinhardsbrunn, 1989.

[70] S. Kambhampati and J.A. Hendler. A validation based theory of plan modification and reuse. *Artificial Intelligence*, 55:193–253, 1992.

[71] M. Keane. *Analogical Problem Solving.* Ellis Horwood (Wiley), Chichester, England, 1988.

[72] M.T. Keane. Incremental analogizing: Theory and model. In K.J. Gilhooly, M.T. Keane, R. Logie, and G. Erdos, editors, *Lines of Thinking: Reflections on the Psychology of Thought, Vol I.* Wiley, 1990.

[73] M.T. Keane and M. Brayshaw. The incremental analogy machine. In D. Sleeman, editor, *Third European Working Session on Machine Learning*, London, 1988.

[74] S.T. Kedar-Cabelli. Purpose-directed analogy. In *Proc. of the Cognitive Science Society Conference*, pages 150–159, Irvine, CA, 1985.

[75] M. Kerber and E. Melis. Using exemplary knowledge for justified analogical reasoning. In Michel De Glas and Zdzislaw Pawlak, editors, *WOCFAI '95 – Proceedings of the Second World Conference on the Fundamentals of Artificial Intelligence*, pages 157–168, Paris, France, 1995. Angkor.

[76] G.A. Klein. Applications of analogical reasoning. *Metaphor and Symbolic Activity*, 2:201–218, 1987.

[77] G.A. Klein and J. Weitzenfeld. The use of analogies in comparability analysis. *Applied Ergonomics*, 13:99–104, 1982.

[78] R.E. Kling. A paradigm for reasoning by analogy. *Artificial Intelligence*, 2:147–178, 1971.

[79] F. Klix. Begabungsforschung- ein neuer Weg in der kognitiven Intelligenzdiagnostik. *Zeitschrift für Psychol.*, 191(4):360–387, 1983.

[80] A. Koestler. *The Act of Creation.* Picador, London, 1964.

[81] Th. Kolbe and Ch. Walther. Second-order matching modulo evaluation – a technique for reusing proofs. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, Montreal, 1995. Morgan Kaufmann.

[82] J. Kolodner. Reconstructive memory: A computer model. *Cognitive Science*, 7:281–328, 1983.

[83] J. Kolodner. Retrieving events from case memory: A parallel implementation. In *Proceedings of the Case-Based Reasoning Workshop, DARPA*, pages 233–249, Clearwater Beach, 1988.

[84] J. Kolodner. *Case-Based Reasoning.* Morgan Kaufmann Publishers, 1993.

[85] L. Komatsu. Recent views of conceptual structure. *Psychological Bulletin*, 122:500–526, 1992.

[86] P. Koton. *Using Experience in Learning and Problem Solving.* PhD thesis, Massachusetts Institute of Technology, Laboratory of Computer Science, 1989.

[87] John E. Laird, Paul S. Rosenbloom, and Allen Newell. Chunking in SOAR: The anatomy of a general learning mechanism. *Machine Learning*, 1:11–46, 1986.

[88] G. Lakoff and M. Johnson. *Metaphors We Live By.* University of Chicago Press, Chicago, IL, 1980.

[89] W.H. Leatherdale. *The Role of Analogy, Model, and Metaphor.* North Holland, Amsterdam, 1974.

[90] K. Lorenz. Analogy as a source of science. *Science*, 185:229–234, 1974.

[91] W.W. McCune. Otter 2.0 users guide. Technical Report ANL-90/9, Argonne National Laboratory, Maths and CS Division, Argonne, Illinois, 1990.

[92] D.V. McDermott. Planning and action. *Cognitive Science*, 2:71–109, 1967.

[93] E. Melis. Study of modes of analogical reasoning. TASSO-report 5, GMD, Birlinghoven, Germany, 1990.

[94] E. Melis. Analogies between proofs – a case study. SEKI-Report SR-93-12, University of Saarbrücken, Saarbrücken, 1993.

[95] E. Melis. Change of representation in theorem proving by analogy. SEKI-Report SR-93-07, University of Saarbrücken, Saarbrücken, 1993.

[96] E. Melis. How mathematicians prove theorems. In *Proc. of the Annual Conference of the Cognitive Science Society*, pages 624–628, Atlanta, Georgia U.S.A., 1994.

[97] E. Melis. Analogy-driven proof-plan construction. Technical Report DAI Research Paper No 735, University of Edinburgh, AI Dept, Dept. of Artificial Intelligence, Edinburgh, 1995.

[98] E. Melis. A model of analogy-driven proof-plan construction. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 182–189, Montreal, 1995.

[99] E. Melis and M. Veloso. Analogy makes proofs feasible. In D. Aha, editor, *AAAI-workshop on Case-Based Reasoning*, pages 13–17, Seattle, 1994.

[100] C. Mervis and E. Rosch. Categorization of Natural Objects. *Annual Review of Psychology*, 32:89–115, 1981.

[101] J.S. Mill. *System of Logic. In: Collected Works of John Stuart Mill, Vol.VIII, Book III.* University of Toronto Press, Toronto, Canada, 1973, first published 1873.

[102] Tom M. Mitchell, Richard M. Keller, and Smadar T. Kedar-Cabelli. Explanation-based generalization: A unifying view. *Machine Learning*, 1:47–80, 1986.

[103] J. Mostow. Design by derivational analogy: Issues in the automated replay of design plans. *Artificial Intelligence*, 40(1-3):119–184, 1989.

[104] J.C. Munyer. *Analogy as a Means of Discovery in Problem Solving and Learning.* PhD thesis, University of California, Santa Cruz, 1981.

[105] A. Newell and H.A. Simon. *Human Problem Solving.* Prentice-Hall, Englewood Cliffs, NJ, 1972.

[106] N.J. Nilsson. *Principles of Artificial Initelligence.* Tioga, Palo Alto, CA, 1980.

[107] L.R. Novick and K.J. Holyoak. Mathematical problem solving by analogy. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 14:398–415, 1991.

[108] S. O'Hara. A model of the "redescription" process in the context of geometric proportional analogy problems. In K.P. Jantke, editor, *Proceedings of the International Workshop on Analogical and Inductive Inference*, Berlin, 1992. Springer.

[109] S. Owen. *Analogy for Automated Reasoning.* Academic Press, 1990.

[110] P.L. Pirolli and J.R. Anderson. The role of learning from examples in the acquisition of recursive programming skills. *Canadian Journal of Psychology*, 39:240–272, 1985.

[111] D. Plaisted. Theorem proving with abstraction. *Artificial Intelligence*, 16:47–108, 1981.

[112] G. Polya. *Mathematics and Plausible Reasoning.* Princeton University Press, NJ, 1954.

[113] G. Polya. *How to Solve it.* 2nd ed. Doubleday, New York, 1957.

[114] A.M. Richter and S. Wess. Similarity, uncertainty, and case-based reasoning in PATDEX. In R.S. Boyer, editor, *Automated Reasoning , Essays in Honour of Woody Bledsoe.* Kluwer, 1991.

[115] L. Rips. Inductive judgements about natural categories. *Journal of Verbal Learning and Verbal Behavior*, 14:665–681, 1975.

[116] L. R.Novick. Analogical transfer, problem similarity, and expertise. *Journ. of Experim. Psych., Learning, Memory and Cognition*, 14(3):510–520, 1988.

[117] E. Rosch. Natural categories. *Cognitive Psychology*, 4:328–350, 1973.

[118] E. Rosch. Prototype classification and logical classification: The two systems. In E.K. Scholnick, editor, *New Trends in Conceptual Representation: Challenges to Piaget's Theory?*, pages 73–86. Erlbaum, Hillsdale, Nj, 1983.

[119] S.J. Russell. Analogy by similarity. In D. Helman, editor, *Analogical Reasoning*, pages 251–269. Kluwer Academic Publisher, 1988.

[120] S.J. Russell. *The Use of Knowledge in Analogy and Induction.* Pitman, London, 1989.

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

[122] R.C. Schank. *Dynamic Memory: A Theory of Learning in People and Computers.* Cambridge University Press, Cambridge England, 1982.

[123] Roger C. Schank. *Explanation Patterns.* Lawrence Erlbaum Associates, Inc., Hillsdale, NJ, 1986.

[124] D.A. Schön. Generative metaphor. In A. Ortony, editor, *A Perspective on Problem-Setting in Social Policy,Metaphor and Thought*, pages 154–283. Cambridge University Press, Cambridge, UK, 1979.

[125] Ch. Schwarz. Zur Spezifik von Erkennungsprozessen bei mathematisch Hochbegabten. *Zeitschrift für Psychol.*, 194(1):17–23, 1986.

[126] C.M. Seifert. Goals in remindings. In *Proceedings of the DARPA Workshop on Case-Based Reasoning*, pages 357–369, San Mateo, CA, 1988. Morgan Kaufmann.

[127] C.M. Seifert and K.C. Gray. Representational issues in analogical transfer. In *Proceedings of the Eleventh Conference of the Cognitive Science Society*, pages 30–37, Hillsdale, NJ, 1989. Erlbaum.

[128] B.G. Silverman. Analogy in system management. *IEEE Trans.Man and Cybernetics*, 13(6):1049–1075, 1983.

[129] E.E. Smith and D.L. Medin. *Categories and Concepts.* Harvard University Press, 1981.

[130] B. Smyth and M.T. Keane. Retrieving adaptable cases. In *Topics in Case-Based Reasoning*, pages 209–220, Berlin, 1994. Springer.

[131] B.S. Stein, K.R. Way, S.E. Benningfield, and C.A. Hedgecough. Constraints on spontaneous transfer in problem-solving tasks. *Memory and Cognition*, 14:432–441, 1986.

[132] R.J. Sternberg. Component processes in analogical reasoning. *Psych. Review*, 84(4):353–378, 1977.

[133] Peter Stone, Manuela Veloso, and Jim Blythe. The need for different domain-independent heuristics. In *Proceedings of the Second International Conference on AI Planning Systems*, pages 164–169, June 1994.

[134] P. Thagard. The greatest analogies in the hstory of science. *Canadian Artificial Intelligence*, pages 14–20, 1993.

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

[136] A. Tversky. Features of similarity. *Psych. Review*, 84(4):327–352, 1977.

[137] A.I. Ujomow. Die Hauptformen und -regeln der Analogieschlüsse. In G. Kröber, editor, *Studien zur Logik der wissenschaftlichen Erkenntnis.* Akademie Verlag, Berlin, 1967.

[138] J.D. Ullmann. *Principles of Database Systems.* Computer Science Press, Rockville, MD, 1980.

[139] S. Vadera. Proof by analogy in mural. *Formal Aspects of Computing*, 7:183–206, 1995.

[140] K. VanLehn and R.M. Jones. Better learners use analogical problem solving sparingly. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 338–345, Amherst, MA, 1993. Morgan Kaufmann.

[141] Manuela Veloso and Agnar Aamodt, editors. *Case-Based Reasoning Research and Development.* Springer Verlag, October 1996.

[142] Manuela Veloso, Jaime Carbonell, M. Alicia Pérez, Daniel Borrajo, Eugene Fink, and Jim Blythe. Integrating planning and learning: The PRODIGY architecture. *Journal of Experimental and Theoretical Artificial Intelligence*, 7(1):81–120, 1995.

[143] M.M. Veloso. *Learning by Analogical Reasoning in General Problem Solving.* PhD thesis, Carnegie Mellon University, CMU, Pittsburgh, USA, 1992. CMU-CS-92-174.

[144] M.M. Veloso. Flexible strategy learning: Analogical replay of problem solving episodes. In *Proc. of the twelfth National Conference on Artificial Intelligence 1994*, Seattle, WA, 1994.

[145] M.M. Veloso. *Planning and Learning by Analogical Reasoning.* Springer, Berlin, New York, 1994.

[146] M.M. Veloso and J.G. Carbonell. Derivational analogy in PRODIGY: Automating case acquisition, storage, and utilization. *Machine Learning*, 10:249–278, 1993.

[147] A. Villafiorita and R. Sebastiani. Proof planning by abstraction. In *Proceedings ECAI-94 workshop on interactive systems*, Amsterdam, 1994.

[148] S. Vosniadou and A. Ortony. *Similarity and Analogical Reasoning.* Cambridge University Press, 1989.

[149] R. Waldinger. Achieving several goals simultaneously. In N. J. Nilsson and B. Webber, editors, *Readings in Artificial Intelligence*, pages 250–271. Morgan Kaufman, Los Altos, CA, 1981.

[150] J.S. Weitzenfeld. Valid reasoning by analogy. *Philosophy of Science*, 51:137–149, 1984.

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

[152] L. Wos. *Automated Reasoning: 33 Basic Research Problems.* Prentice-Hall, Englewood Cliffs, 1988.

[153] J. Zhang. Selecting typical instances in instance-based learning. In D. Sleeman and P. Edwards, editors, *Proceedings of the Ninth International Workshop on Machine Learning*, pages 470–479, San Mateo, CA, 1992.