Analogical Interpretation in Context

Brian Falkenhainer
Xerox Palo Alto Research Center
3333 Coyote Hill Road, Palo Alto CA 94304

Abstract: This paper examines the principles underlying analogical similarity and describes three important limitations with traditional views. It describes *contextual structure-mapping*, a more knowledge intensive approach that addresses these limitations. The principle insight is that each element of an analogue description has an identifiable *role*, corresponding to the dependencies it satisfies or its relevant properties in the given context. Analyzing role information provides a powerful framework for characterizing analogical similarity, relaxing the one-to-one mapping restriction prevalent in computational treatments of analogy, and understanding how such similarities may be used to assist problem solving. Second, it provides a unifying view of some of the central intuitions behind a number of converging efforts in analogy research.

1 Introduction

The core of analogy is mapping: the identification of correspondences between two analogues and using them to adapt a prior experience to a new situation. To identify similarities, most approaches find matching patterns in the two analogue's representation form, seeking both isomorphic structures and features described by the same predicate.

One such system is SME [7], an analogical mapping program originally developed to study Gentner's (1983) Structure-Mapping theory. However, in using SME for complex problem solving tasks [4, 5], three fundamental problems were found with traditional views of analogy. First, the restriction that matching expressions must be represented by the same predicate is too strong, and existing solutions based on conceptual closeness (e.g., ISA hierarchies) are too weak. Second, the one-to-one mapping restriction is too strong whenever multiple functions are supported by a single element in one of the analogues (e.g., function sharing). Third, the correspondences cannot always be determined from the initial descriptions of each analogue since their descriptions may not contain all of the relevant objects or inferences. Thus, additional information may have to be inferred or retrieved during the mapping computation.

This paper describes *contextual structure-mapping* (CSM), an approach that relaxes these restrictions while ensuring meaningful similarity judgments and tractable computation. It was essential to the success of PHINEAS [4, 5], a program that constructs new causal models of observed phenomena based on their similarity to understood phenomena. The key idea is explicit consideration of the contextual factors affecting analogical interpretation. Specifically, each element of an analogue description has an identifiable *role*, corresponding to the dependencies it satisfies or its relevant properties in the given context. The notion of role makes two important contributions. First, it provides a powerful framework for characterizing predicate "similarity", relaxing the one-to-one mapping restriction, and focusing problem solving and memory retrieval aimed at elaborating analogue correspondences. Second, it provides a unifying view of some of the intuitions behind derivational replay [3, 11], tweaking [10], knowledge-based pattern matching [1].

We begin by briefly describing role information. We then describe how this information is used to compute similarity correspondences within SME and adapt a prior problem solving experience to a new case. Finally, we discuss how the concepts presented here unify and explain several aspects of related work.

2 The role of role

We assume that each analogue is described by a set of *expressions*, where an expression may be a predicate-calculus formula, a feature in a feature vector representation, or a node or link in a semantic network. Take $\mathcal{B}$ and $\mathcal{T}$ to denote the sets of expressions representing the base and target analogues, respectively. Take $\mathcal{K}$ to be one's complete body of knowledge, such that $\mathcal{B}, \mathcal{T} \subset \mathcal{K}$. In general, $\mathcal{K}$ will include or entail information about each analogue not explicit in $\mathcal{B}$ or $\mathcal{T}$. The general mapping goal is to find a set of correspondences between the elements of $\mathcal{B}$ and $\mathcal{T}$. Some correspondences may be directly identifiable from the base and target descriptions; others arise as a side effect of adapting base information to apply to the target case.

Most accounts of analogy and case-based reasoning establish similarity correspondences by testing for predicate or feature identicality, but this is too restrictive. The typical solution for allowing non-identical relations to match is to evaluate similarity by measuring conceptual closeness, using ISA hierarchies [13, 2] or a-priori similarity scores [9]. However, these approaches can produce unmotivated and incorrect similarity correspondences. They fail to recognize an important point: similarity is context sensitive. Having some aspects in common is not an explanation for why two predicates should be viewed as corresponding; only some of their properties are relevant for determining similarity in a given context.

For example, consider classifying a cylindrical tin cup with a handle as a `Hot-Cup` (adapted from [13, 11]). In general, a `Hot-Cup` is something that a person can lift and drink from while it holds a hot liquid. From a previous styrofoam cup (`scup`) example, the following sufficient conditions were found:

$$\texttt{Styrofoam(scup)} \wedge \texttt{Open-Conical(scup)} \wedge \cdots \rightarrow \texttt{Hot-Cup(scup)}$$

In classifying the tin cup via analogy to the styrofoam case, conceptual closeness measures might pair `Open-Conical` with the tin cup's `Open-Cylinder` (both open concavities) and `Styrofoam` to `Tin` (both materials). Yet, if the original dependencies satisfied by the `Styrofoam` condition are retrieved (or reconstructed), it becomes clear that this misses the point of the analogy: the aspect of styrofoam important in this context is its insulating characteristics, just as the tin cup's handle is important because it provides another form of insulation. In this context, the property styrofoam should map to the property of having a handle.

Most matchers require one-to-one mappings because allowing many-to-many mappings can dramatically increase the number of possibilities and lead to incoherence. However, in the styrofoam cup, styrofoam and conical shape provide insulation and a grasping area, respectively, while both functions are provided by the tin cup's handle. Here, an isomorphic mapping fails to fully capture the correspondence; a many-to-one mapping from { `Styrofoam`, `Open-Conical`} to { `Has-Handle`} is needed. Similarly, consider the dual murderer / victim roles of someone committing suicide. Due to function sharing, many-to-many mappings occur in many physical systems.

The key to relaxing these constraints and determining an expression's relevant aspects is an understanding of its *role* in the analogue description. In CSM, an expression's roles are identified by the dependencies it satisfies. Within each analogue, if $\mathcal{C}$ is some predication whose truth is dependent on predication $\mathcal{A}$, then the *role* of $\mathcal{A}$ is to satisfy the dependency relationship with $\mathcal{C}$. $\mathcal{A}$ may fill other roles as well, in as much as $\mathcal{A}$ satisfies other dependencies.

Correspondences between two analogues' expressions are determined by analyzing their roles in the context of their respective analogues. Thus, if the dependencies supported by an expression $\mathcal{E}_b$ may be satisfied in an alternate manner, say by expression $\mathcal{E}_t$, then $\mathcal{E}_b$ and $\mathcal{E}_t$ are considered *functionally analogous* and may be placed in correspondence. We may generalize this (and form a recursive definition) by saying that given two corresponding roles (i.e., not necessarily identical), their role fillers may be considered functionally analogous and eligible for being placed in correspondence, independent of predicate (or feature) identicality. Importantly, a fundamental component

of analogy is knowing what aspects of a given relation are relevant and hence be able to recognize alternate ways to achieve similar functionality. For example, the property **RAINFALL** should map to the property **IRRIGATION** if the role of these conditions is to ensure that a given crop receives sufficient water. Note they are not analogous in other roles, such as washing a plant's leaves. Further, to the extent that knowledge of $\mathcal{E}_b$'s role is incomplete and $\mathcal{E}_t$ does not provide identical functionality, the functionally analogous relation is merely plausible (e.g., the humidity associated with rainfall may be relevant).

Roles take many forms. In design, the role of particular design decisions and artifact components is the satisfaction of particular design specifications and rationale. The role of an agent's actions (as in planning or a story) may be in support of certain outcomes. In physical systems, the role of a given component is typically the behavior it contributes to. In deductive proof, the role of an antecedent is to provide logical support for its consequent. There are several ways to determine and exploit the role an expression is servicing. We discuss two of these next.

2.1 Explicit dependencies

When an expression's role is explicit in a given analogue representation, role determination consists simply of consulting this information. This appears in two forms. First, the roles of b_i and t_j may be explicit in both base and target descriptions. For example, expressions `P` and `R` will be placed in correspondence when matching `IMPLIES(P,Q)` with `IMPLIES(R,Q)`, since their respective roles are to deductively support `Q`.

Second, the role of b_i may be explicit in $\mathcal{B}$ but there is not enough information in $\mathcal{T}$ to identify its correspondent. In this case, the task is to take the unmapped b_i's role and search for a corresponding role and role filler applicable to the target setting. Since $\mathcal{T}$ is generally a subset of all available knowledge about the target, this case will require memory retrieval and inferencing to find the desired information. For example, mapping

$\mathcal{B}$: `High-Rainfall(region`$_1$`)` $\rightarrow$ `Well-Watered(region`$_1$`)`
to
$\mathcal{T}$: `Irrigated(region`$_2$`)`, `Arid(region`$_2$`)`, `Northern-Hemisphere(region`$_2$`)`

requires retrieval of

`Irrigated(region`$_2$`)` $\rightarrow$ `Well-Watered(region`$_2$`)`

in order to determine which expression in $\mathcal{T}$ corresponds to `High-Rainfall` in $\mathcal{B}$.

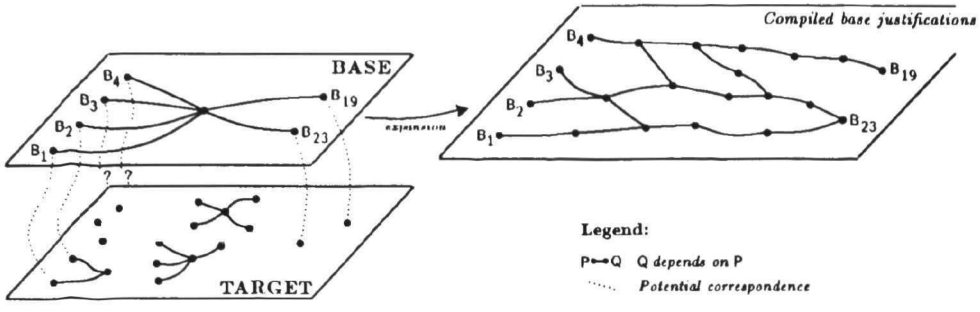
2.2 Compiled and abstracted knowledge

Representations that are adequate for traditional approaches to problem solving may not be suitable for performing analogical reasoning. AI systems tend to represent a minimalist approach to encoding knowledge, in which detailed descriptions or intermediate reasoning steps are avoided to promote efficiency of use. For example, a physical process may be modeled at some level of abstraction and a design plan may not contain the rationale behind the decisions it embodies. Indeed, knowledge compilation is a central goal of explanation-based generalization. While effective for accelerating reasoning, a great deal of information is intentionally removed. This poses a significant problem for adaptation of a given base analogue: these intermediate or second-order justifications are often needed to ascertain the requisite role information (Figure 1).

For example, consider the following schema used to explain why crops grow in `region`$_1$:

```
Cro p-Growing(region1)
   PRECONDITIONS   High-Rainfall(region1) ∧  Fertile-Soil(region1) ∧
                   Sunny(region1) ∧ ...
   EFFECTS         Growing-Crops(region1)
```

Figure 1: Compiled knowledge problem. Intermediate reasoning steps removed to increase problem solving efficiency are often required when considering how to elaborate analogical correspondences and adapt a solution to unanticipated cases. The **BASE** representation depicts a macro compiled from the inferences to the right. A question mark indicates a relevant base expression having no known target correspondent without more information about its role in the base context.



Why is the `High-Rainfall` precondition there? Suppose a deeper explanation is retrievable and reveals that this condition is important in this context because it ensures that the region is well watered. Without this deeper explanation, it would be impossible to justify why it is reasonable to place `High-Rainfall(region`$_1$`)` in correspondence with `Irrigated(region`$_2$`)`.

To address the compiled knowledge problem, we assume the availability of needed background knowledge, either cached or reconstructable. This background knowledge serves to decompose and elaborate the reasons underlying a particular dependency relationship. This information is consulted as needed during the mapping computation.[1] In the implementation, a `CACHE` field accompanies all compiled rules. For example, the crop growing schema should have an added `CACHE` field to store compiled reasoning steps like `High-Rainfall(region`$_1$`)` $\rightarrow$ `Well-Watered(region`$_1$`)`.

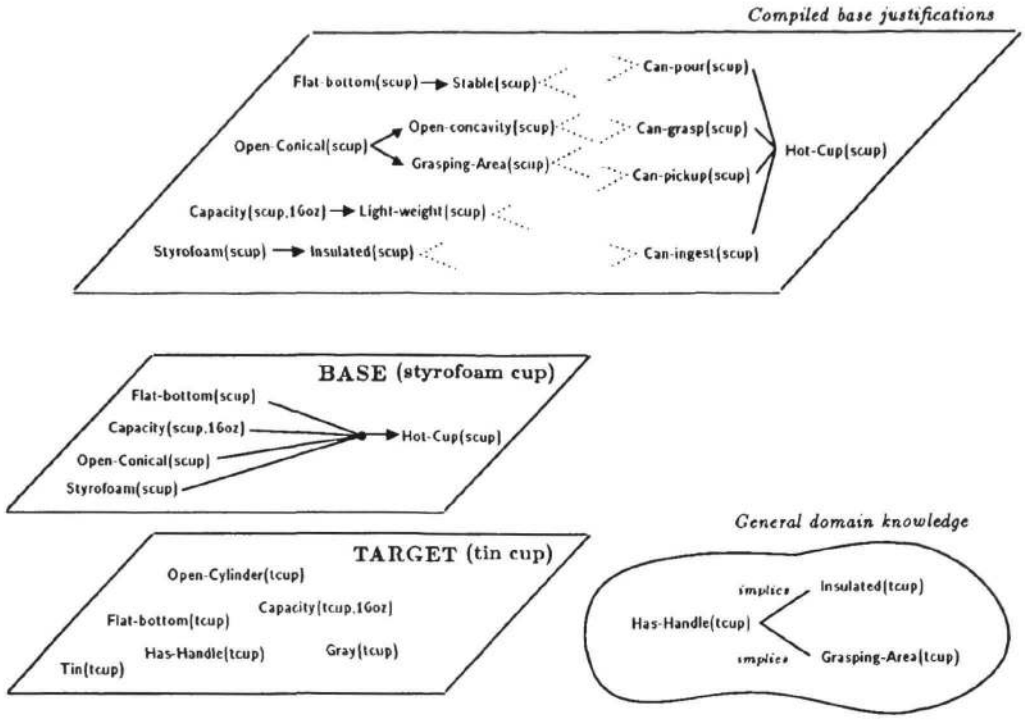
3 The Map and Analyze Process

The mapping process is traditionally depicted as a form of pattern matching between base and target descriptions. However, in realistic memories, mapping will be operating on a subset of all that is inferrably known about the base and target. Thus, the process of elaborating correspondences and adapting elements of the base to fit the target situation often requires inferring additional information during the mapping computation in response to mapping impasses. Rather than endow the mapping mechanism with unlimited inferencing power, we decompose the process to form a *map and analyze* cycle: use simple matching criteria to determine the best, initial mapping between the analogues, analyze the results and seek additional relevant information about unmatched areas, reexamine the mapping to determine the information's impact on the mapping (i.e., extensions or complete shifts), analyze the new results, etc.

The mapping phase is computed by SME [7] a general mapping tool which formulates mappings based on user-supplied *match rules*. Match rules specify which local, pairwise correspondences (called *match hypotheses*) between expressions and entities are possible, restrictions on how they may be combined, and preference criteria for scoring these combinations. Using the rules of contextual structure-mapping forms SME$_{CSM}$. Each time a mapping is computed, one out of the set of possible mappings is selected based on systematicity [8, 7] and relevance to the current problem solving goals [5, 6].

[1] Including all background knowledge in the original base and target descriptions would be too expensive. Additionally, which added details are needed cannot be identified until impasses arise during the mapping computation.

Figure 2: Categorizing a tin cup by analogy to a styrofoam cup.



In the remainder of this section, we illustrate how role information is used to provide controlled relaxation of the same predicate and one-to-one mapping restrictions.

3.1 Relaxing identicality

Given base and target expressions, what criteria are used to propose a match between them? We start with the standard set taken from Gentner's structure-mapping theory [7]. The main rule pairs expressions that use the same predicate. Additional rules are used to support objects and commutative predicates. However, these rules suffer from a dependence on identicality to initiate matches. To motivate matches between expressions using different predicates, we add a rule to support the functionally analogous criterion:

Rule 1 (Functionally Analogous) *Two expressions are considered* functionally analogous *and may match if they fill corresponding roles in the context of the structures being matched.*

When role information is explicit in the base and target representations, SME_{CSM} uses this rule to match functionally analogous expressions. When a relevant base expression has no discernible correspondent in the target case, its role is analyzed in greater depth.

To illustrate these match criteria, we use a simple example taken from [13, 11] and adapted to illustrate the key ideas. The task is to classify a given tin cup as an instance of `Hot-Cup`. The base exemplar is a previously classified styrofoam cup. Their descriptions are shown in Figure 2.

The process begins by comparing the initial descriptions of the two cases. `SME` finds that the relevant `Flat-bottom` and `Capacity` properties are shared by both cups. It also fails to find correspondents for the styrofoam cup's other important properties: `Open-Conical` and `Styrofoam`. During the analysis phase, more detail is retrieved about the roles of these two properties in

the styrofoam case's classification as a `Hot-Cup`. For example, its `Styrofoam` property provides insulation. The system then seeks aspects of the tin cup that fill these roles. For example, the tin cup's `Has-handle` property also provides insulation. When the mapping is reexamined, this added information now enables a complete match and successful classification of the tin cup as an instance of `Hot-Cup`.

3.2 Relaxing one-to-one

The notion of role provides exactly the right constraint for introducing many to many mappings while maintaining coherence and tractability. In CSM, the one-to-one restriction may be violated by a single base or target item filling multiple roles that are filled by multiple items in the other domain. In the cups example, Figure 2 shows the dependencies that motivate a many-to-one mapping from `Styrofoam(scup)` and `Open-Conical(scup)` to `Has-Handle(tcup)`.

The following three rules define *sanctioned* many-to-one mappings and enforce one-to-one mappings as the normal default by examining all cases of two base items mapping to a single target item.

Rule 2 (Direct role-filler) *Multiple base items b_1 and b_2, filling roles $\mathcal{R}_{b1}$ and $\mathcal{R}_{b2}$ respectively, may map to a single target item, t_i, filling roles $\mathcal{R}_{t1}$ and $\mathcal{R}_{t2}$, if role $\mathcal{R}_{b1}$ corresponds to role $\mathcal{R}_{t1}$ and role $\mathcal{R}_{b2}$ corresponds to role $\mathcal{R}_{t2}$.*

This rule sanctions match hypotheses that may violate the one-to-one restriction. In the cups example, it pairs the `Has-Handle` predicate in the tin cup description to both `Styrofoam` and `Open-Conical`. However, unless this sanctioning is propagated to their respective subexpressions (e.g., the predicates' arguments), a one-to-one restriction will still be in effect.

Rule 3 (Role-filler sub-expressions) *Multiple base items may map to a single target item if the base and target items are subexpressions of a sanctioned many-to-one mapping.*

In the cups example, because `Styrofoam` and `Open-Conical` apply to the same object, `scup`, only a one-to-one mapping from `scup` to `tcup` is needed and this rule does not apply.

With sanctioned violations of one-to-one identified, we are now ready to define when two match hypotheses are conflicting. Due to its non-monotonic nature, implicit in the following rule is the assumption that all sanctioned pairings are known at the time the rule is invoked.

Rule 4 (One-To-One) *Unless explicitly sanctioned, two match hypotheses are* conflicting *if they pair multiple base items to the same target item.*

Thus, the two match hypotheses (`Styrofoam`, `Has-Handle`) and (`Open-Conical`, `Has-Handle`) are compatible in the cups example. A symmetric set of three rules exist for multiple target items mapping to a single base item. Together, the six rules identify sanctioned many-to-many mappings and enforce one-to-one for all other, unsanctioned cases.

4 A Unifying View

The contextual structure-mapping framework provides a unifying view of the basic intuitions behind several recently developed methods for computing similarity. *Derivational replay* mechanisms make the observation that it is typically easier to reuse the problem solving process of a prior episode than the episode's final solution [3]. Taking a stored problem solving plan, a new problem is solved by replaying the plan top-down, resolving subgoals that no longer apply in the current situation. CSM's definition of *functionally analogous* explains the underlying intuition behind replay's appeal

over mapping only a final solution: the root problem in reusing a prior solution is the need to understand the various roles or functions the solution fulfills. In problem solving, adapting a prior solution to a new problem instance is greatly simplified if decisions' rationale are known, so that their intent can be satisfied without necessarily adhering to the same decisions. Top-down replay achieves this by essentially reseeking each role filler in the new situation. An alternate method would start at the solution, and work backwards, analyzing role information at solution transfer impasses. The implicit assumption in replay mechanisms is that it is more efficient to work forward, replaying the entire decision-making process, than to work backward, reconsidering the decisions (alternate ways to achieve functionality) where needed. This tradeoff is influenced in part by the solution's modularity.

The Yale SWALE project [10], Kedar-Cabelli's PER [11], and PHINEAS (which uses CSM) [4, 5] are efforts to use analogy to reduce the cost and increase the creativity of explanation building. These systems demonstrate operations very closely related to the notions of role and derivational replay. For example, SWALE applies stored *explanation patterns* (schemas) to new situations, *tweaking* them as needed to adapt to the situation's novel aspects. Like Kedar-Cabelli's PER model, it attempts to rederive portions of a prior explanation that do not apply in the new situation. For the tweaking operation, SWALE uses a set of revision rules, such as *substitute alternate theme* or *substitute related action*. These suggest ways to repair inapplicable portions of a recalled schema. CSM offers a more general explanation of these tweaking operations: determine the anomalous item's role and seek elements of the current situation that could satisfy that role's dependencies. For example, if the current actor lacks a requisite theme, *substitute alternate theme* tries to find an alternate theme for the actor. In the CSM framework, this corresponds to an unsatisfied dependency (i.e., role) which either must be satisfied, assumed, or conjectured with new concepts.

PROTOS [1] performs diagnosis by relating a new case to a store of previously classified exemplars, performing classification by finding the exemplar that best matches the new instance. *Knowledge-based pattern matching* computes a match between an exemplar and a new instance by using the domain knowledge stored with each exemplar to explain the equivalence of non-identical features. For example, in comparing two chairs, LEGS and PEDESTAL are equivalent because they both provide SEAT-SUPPORT. In CSM terms, this process corresponds to showing that two features are functionally analogous. However, PROTOS limits the scope of a match to the explicitly given features of the two cases. It does not include the possibility that additional features may be derived or retrieved from memory in response to the needs of the match elaboration process. Furthermore, it uses a feature-vector representation (i.e., a set of attribute-value pairs) which is inadequate for representing a complex set of interrelationships between an analogue's parts.

5 Discussion

Analogical mapping for simple representations and tasks, particularly for within-domain comparisons, is a straightforward process. On the other hand, rich representations and complex problem solving tasks (both within and across domains) require a more sophisticated mechanism for computing similarities that is able to compare syntactically different analogues and select the best mapping when ambiguities arise. The utility of contextual structure-mapping for such settings has been demonstrated in PHINEAS [4, 5], a program that proposes qualitative causal explanations of observed, time-varying phenomena based on their similarity to understood phenomena. PHINEAS has been extensively tested on over a dozen examples, such as explaining evaporation by analogy to dissolving, heat flow and osmosis by analogy to liquid flow, and a variety of mechanical and electrical harmonic oscillators.

This paper has claimed that analogy requires role information to at least plausibly suggest the relevance and interrelatedness of each analogue's features. The ability to learn this relevance information is of fundamental importance. One approach is to use a developing analogy to motivate

specific questions about the world and use directed experimentation to answer them and ascertain the requisite relevance information [5]. A second approach is to again use a developing analogy to motivate specific questions, but place the system in a learning apprentice setting and obtain requisite relevance information from the user [1]. We are currently investigating a third approach: use inductive methods to suggest which factors are relevant to the concept under study.

Acknowledgements

Ken Forbus, Dedre Gentner, Danny Bobrow, and Mark Shirley provided insightfull discussions and helpful comments on prior drafts of this paper. The foundation for this work is taken from the author's dissertation, which was supported by an IBM Graduate Fellowship and by the Office of Naval Research, Contract No. N00014-85-0559.

References

[1] Bareiss, R., Porter, B., & Wier, C. (1987). Protos: An exemplar-based learning apprentice. *Proceedings of the Fourth International Workshop on Machine Learning.*

[2] Burstein, M. (1986). Concept formation by incremental analogical reasoning and debugging. In R.S. Michalski, J. Carbonell, T. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach, Volume II.*

[3] Carbonell, J. (1986). Derivational analogy: A theory of reconstructive problem solving and expertise acquisition. In R.S. Michalski, J. Carbonell, T. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach, Volume II.*

[4] Falkenhainer, B. (1986). *An examination of the third stage in the analogy process: Verification-based analogical learning* (Technical Report UIUCDCS-R-86-1302). Department of Computer Science, University of Illinois. A summary appears in *IJCAI-87.*

[5] Falkenhainer, B. (1988). *Learning from physical analogies: A study in analogy and the explanation process.* PhD thesis, University of Illinois, 1988.

[6] Falkenhainer, B. (1990). Contextual structure-mapping. (*submitted for publication*).

[7] Falkenhainer, B., Forbus, K.D., & Gentner, D. (1989). The structure-mapping engine: Algorithm and examples. *Artificial Intelligence, 41,* 1-63.

[8] Gentner, D. (1983). Structure-mapping: A theoretical framework for analogy. *Cognitive Science, 7.*

[9] Holyoak, K. & Thagard, P (1989). Analogical mapping by constraint satisfaction. *Cognitive Science, 13*(3).

[10] Kass, A., Leake, D., & Owens, C. (1986). SWALE, A program that explains. In R Schank (Ed.), *Explanation patterns: Understanding mechanically and creatively.* Lawrence Erlbaum Associates.

[11] Kedar-Cabelli, S. (1988). Formulating concepts and analogies according to purpose. PhD Thesis, Rutgers University.

[12] Kolodner, J., Simpson, R. & Sycara-Cyranski, K. (1985). A process model of cased-based reasoning in problem solving. *Proceedings of IJCAI-85.*

[13] Winston, P., Binford, T, Katz, B. & Lowry, M. (1980). Learning physical descriptions from functional definitions, examples, and precedents. *Proceedings AAAI-83.*