The Role of Mapping in Analogical Transfer[1]

Hong S. Shinn
School of Information and Computer Science
Georgia Institute of Technology
Atlanta, GA 30332, U.S.A.

Abstract

This paper aims to provide a view of the role of analogical mapping in the entire process of analogical problem solving. In many models, analogical mapping is responsible for identifying the analogy between two problems by considering structural and semantic similarities. However, given a non-trivial analogy problem, success of mapping does not always guarantee successful transfer of analogy. In fact, there exist many analogy problems, which succeed on analogical mapping but which fail on analogical transfer. While a potential mapping between problems can be generated, that mapping might not be justifiable until transfer from one problem to another is attempted.

We present our analogical mapping method and show how it works for inter-domain and intra-domain analogies. We demonstrate several analogy problems in which a mapping can be generated that cannot be transferred. We also compare our method to two general mapping mechanisms, SME and ACME, and show that it performs at least as well, and sometimes better than either of those methods.

Key Words: analogical problem solving, analogical mapping.

1 Introduction

This paper aims to provide a view of the role of analogical mapping in the entire process of analogical problem solving. Analogical problem solving contains at least the following components [Shi88,CM85,HT88]: retrieval of a plausibly analogous case, analogical mapping, and analogical transfer. The step of analogical transfer may involve modification of a previous solution and justification of the result obtained before the result is transferred [Shi88].

In many models, analogical mapping is responsible for identifying the analogy between two problems by considering structural and semantic similarities. However, given a non-trivial analogy problem, analogical mapping by itself does not always guarantee that an analogy will be successful. While it can produce a potential mapping between problems, a mapping might not be justifiable until transfer from one problem to another is attempted.

In this paper, we illustrate the role of mapping in analogical problem solving and we present a hierarchical method of analogical mapping that is based primarily on similarity of structures. The method uses relatively little semantic information. Instead, it relies on the analogical transfer step to determine the merit of a potential analogical mapping.

We show how our method works for inter-domain and intra-domain analogies. We demonstrate several analogy problems in which a mapping can be generated that cannot be transferred. We also compare our method to two general mapping mechanisms, Structure-Mapping Engine (SME) [FFG86] and Analogical Constraint Mapping Engine (ACME) [HT88], and show that it performs at least as well, and sometimes better than either of those methods.

[1]This research has been supported in part by the Army Research Institute under Contract No. MDA-903-86-C-173, is currently supported in part by NSF under Grant No. IST-8608362, and in part by Lockheed AI Center under Grant No. DTD 09-25-87.

2 Analogical Mapping Algorithm

Before introducing our algorithm, we need to clarify the problem of analogical mapping. Polya [Pol54] views "analogy" as a systematic correspondence between two systems preserving certain relations. For his basic type of analogy, Polya defines analogy as "similarity of relations", where relations are similar if they are governed by the same laws. He illustrates this with an example: the multiplication of numbers *multiply(x,y)* is analogous to the addition of numbers *add(x,y)* in the sense that both multiplication and addition are commutative. In other words, two relations *multiply(a,b)* and *add(a,b)* are similar because they are governed by the same commutative law: *equals[OP(a,b),OP(b,a)]*. Interpreting Polya's definition of similarity in analogical problem solving, "similarity" in problems is what leads to similar effects on their solutions. The problem here is that, without knowing beforehand what the similarity's effect will be on the solution to the target problem, we must find that similarity which can be used in deriving the solution. Thus, what analogical mapping does is to find the most probable similarity candidates before transfer of knowledge from source to target is attempted.

Our analogical mapping algorithm follows Gentner's systematicity principle ([Gen83], p. 163) in that it transfers "a system of connected knowledge, not a mere assortment of independent facts". In other words, during mapping between structures, even the highest order predicates may not be mapped separately from their lower level entities.

In dealing with similarity, however, we do not accept Gentner's entire theory of structure mapping. In our mapping scheme, two relations which are structurally similar (i.e., the current partonomic roles in both structures are the same) will not be thrown out. For example, according to Gentner, two relations *equals[multiply(a,b),multiply(b,a)]* and *equals[add(a,b),add(b,a)]* are not mappable to each other because the highest level predicates are identical (i.e., *equals*), but not the lower level predicates (i.e., *add* and *multiply*). On the other hand, in our scheme, these are mappable because their high order predicates are the same while the low level predicates "add" and "multiply" are structurally similar due to their similar roles in the whole relations. Burstein [Bur86] demonstrates with his system CARL the necessity of mapping between nonidentical relations, criticizing Gentner's structure mapping which fails on this kind of similarity.

Another characteristic of our mapping scheme is hierarchical mapping. This is frequently used when problems are represented in hierarchical structure. In fact, analogy between problems usually exists at an abstract level. Thus, mapping starts at the highest level first and proceeds to the next lower level and so on until analogy breaks down.

In our scheme, the entire mapping process is a recursive application of a two-step hierarchical mapping: first map the two problem structures systematically under structural similarity and then decompose them into the next lower level structures (see Figure 1). Structural similarity is found not only in physical structures but also in functional structures. Functional structures are described by functional objects and relations such as functions, purposes, goals, constraints, conditions, and states. For example, *an air conditioner is like an electric fan* because their top level functions are the same (i.e., excite-air). Another example of analogy is found between *society* and *organism* because they are similar in their functional organizations.

As a result of analogical mapping, an analogy map is generated for two cases showing correspondences between both relations and their objects. An analogy map represents a common structure between source and target structures with a binding list between source and target elements. The common structure represents a common problem schema which is used as a medium of transfer in analogical problem solving [Shi88,CM85]. For example, analogical mapping between *multiply(a,b)* and *add(a,b)* generates the analogy map as a common structure *OP(a,b)* with the binding list [(OP multiply add)] meaning that there is one binding *OP* and it binds to *multiply* in the source and to

Input: A source case and a target problem
Output: An analogy map (AMAP)
Algorithm:
Recursive application of two-step hierarchical mapping:
given two problem structures,
1. Map them systematically under structural similarity:
   identify components whose partonomic roles in both structures are the same;
   map components as specifically as possible under the current AMAP;
   if mappable
   then add correspondences between components to AMAP
   else return AMAP
2. Hierarchical refinement:
   decompose the current level into the next lower level structures;
   pair them in the same partonomic roles

Figure 1: Analogical Mapping Algorithm

add in the target.

3 Related Work

Gentner's structure mapping theory with its implementation, SME [FFG86], demonstrates the importance of systematicity in interpreting an analogy. But, it is often criticized because of its syntactic approach.

Many recent models consider semantic and pragmatic characteristics of analogy as well as syntactic information to guide analogical mapping. For instance, Burstein [Bur86] introduces some top-down constraints on relations and primarily relates objects in terms of their functional roles in analogical mapping. Winston's mapping is driven by importance-dominated matching [Win80,Win82]; importance is mainly determined by causal relations in the situations.

Holyoak and Thagard's mapping theory [HT88] attempts to take into account all three dimensions of analogy: syntax, semantics and pragmatics. Their program called ACME computes an analogical map by means of constraint-satisfaction based on five heuristic constraints[2]: logical compatibility, uniqueness, relational consistency, semantic similarity, and role identity. Semantic and pragmatic information help to constrain the search for the most plausible mapping. But, the problem with this approach is that there exist many analogy problems on which such heuristics do not work (an example will be shown in Section 5.1).

Our mapping algorithm is similar to SME in that both enforce systematicity (as shown in the previous section), but different in that ours maps predicates under similarity by functional roles while SME maps under identity. Ours is also similar to Burstein's and ACME in that it maps components by considering part-whole relationships. However, unlike ACME and Winston's, much of the semantic information is not explored during mapping. Rather, it will be checked when the knowledge to be transferred is justified in the transfer step.

[2]Several of these constraints were renamed in a later version of ACME [personal communication with Keith J. Holyoak, May 1988].

4 Applications of Analogical Mapping

Analogical mapping is a step of predicting a plausible analogy, which will be tried for transfer. During the actual transfer attempt, mapping results are filtered considering semantic similarity. The following applications show how these processes are performed.

The first two applications of our mapping algorithm rely on similarity in physical structures: Section 5.1 shows analogy examples between different domains, while Section 5.2 compares analogies within the same domain. These examples are also used to compare our algorithm to two general analogical mapping mechanisms, SME and ACME. In Section 5.3, an application from the JULIA project shows an example in functional structures.

4.1 Inter-Domain Examples

Applying our analogical mapping algorithm, let's solve the problem $\frac{d}{dx}[\sin x - \ln x]$ using the following case:

Problem: $\int [e^x + 1]\, dx$
Solution: $e^x + x + C$
Reasoning steps: $\int [e^x + 1]\, dx \Longrightarrow \int e^x\, dx + \int 1\, dx \Longrightarrow e^x + x + C$

When the mapping algorithm, in the first cycle, is applied to the top level structures (i.e., $\frac{d}{dx}[\sin x - \ln x]$ and $\int [e^x + 1]\, dx$), it successfully produces the analogy map

$$\mathcal{F}[f(x)\ OP\ g(x)]$$

with bindings $[(\mathcal{F}\ \frac{d}{dx}\ \int)\ (\mathrm{OP}\ -\ +)\ (f(x)\ \sin x\ e^x)\ (g(x)\ \ln x\ 1)]$. Since the first reasoning step of the source case predicts the following analogy (in an abstract form):

$$\mathcal{F}[f(x)\ OP\ g(x)] = \mathcal{F}[f(x)]\ OP\ \mathcal{F}[g(x)]$$

the target problem reduces as follows:

$$\frac{d}{dx}[\sin x - \ln x] = \frac{d}{dx}\sin x - \frac{d}{dx}\ln x$$

In the next cycle, the mapping between the next lower level structures $\int e^x\, dx$ and $\frac{d}{dx}\sin x$ succeeds, but analogical transfer between these two fails. This is the level where the analogy breaks down and the mapping process halts. Thus, the analogy between the above two cases resides only at the top level. This example shows the utility of hierarchical mapping in identifying analogy, since the analogy at higher levels of abstraction can be used even though there does not exist a complete analogy.

Consider another problem

$$e^{2x+3}$$

using the same source case. It is similar to the first example in that mapping predicts

$$e^{2x+3} = e^{2x} + e^3$$

However, this hypothesis is not correct; the correct transformation is $e^{2x+3} = e^{2x} * e^3$. This example shows that successful analogical mapping may not guarantee the existence of analogy when semantic similarity is missing. The semantic similarity is checked using reasonings similar to those of the source case. This is done during the process of analogical transfer to justify the hypothesized analogy. (See [Shi88] for more discussion of the justification problem.)

Let us apply SME and ACME to the first analogy problem:

Target: $\frac{d}{dx}[\sin x - \ln x]$
Source: $\int [e^x + 1]\, dx$

SME would fail to recognize this analogy because the two high level predicates $\frac{d}{dx}$ and $\int$ are not identical. This example shows that structural mapping under predicate identity is too strong. In ACME, the logical compatibility requires the second arguments $\ln x$ and 1 to be the same logical kind (e.g., n-place predicates to n-place predicates, constants to constants). Because this constraint precludes a mapping between the one-place predicate $\ln x$ and the constant 1, ACME would also fail on this analogy. This case suggests that semantic and pragmatic information should be used cautiously because of their heuristic nature.

4.2 Intra-Domain Examples

Given a problem

$$\int \frac{1}{\sqrt{3 - y^2}}\, dy$$

consider analogical mapping problems with each of the following three cases.

Case 1:

Problem: $\int \frac{1}{\sqrt{3+y^2}}\, dy$
Solution: $\sinh^{-1} \frac{y}{\sqrt{3}} + C$

Case 2:

Problem: $\int \frac{1}{\sqrt{1-x^2}}\, dx$
Solution: $\sin^{-1} x + C$

Case 3:

Problem: $\int \frac{1}{\sqrt{5-z^2}}\, dz$
Solution: $\sin^{-1} \frac{z}{\sqrt{5}} + C$

All three mappings succeed with our mapping algorithm because the three cases are all structurally similar to the target problem.

In the first case, analogy transfer from case 1 to the target problem is not possible (because the previous reasoning of case 1 is not applicable to the target problem). In the second case, transfer from the source case is not possible until some modification is performed. That is, in order to apply the solution of case 2 to the target problem, the form $\sqrt{a - z^2}$ embedded in the target problem needs be transformed to the form $\sqrt{1 - x^2}$ in case 2. In the third case, the source solution can be transferred to the target domain so the target solution will be $\sin^{-1} \frac{y}{\sqrt{3}} + C$.

The success of analogical mapping leads directly to analogy transfer in the third example. The first example shows, however, that the success of mapping may not guarantee successful analogy transfer. (It only predicts a possibility of transfer which should subsequently be verified.) Furthermore, the second example shows that even when analogical mapping eventually leads to analogy transfer, successful analogical mapping may not directly dictate what is to be transferred from the source case to the target problem. (It may only hint at what is to be transformed in order to reach a transferable state.) So, the role of analogical mapping is to identify a plausible analogy based on known similarity before transfer of analogical knowledge from the source case to the new problem is attempted [Shi88].

Note that SME and ACME are similar to our mapping algorithm in that they will come up with successful mappings with all three cases. This shows that, even when ACME considers semantic and pragmatic accounts, it is not able to distinguish analogies which lead to analogical transfer (i.e., case 3) from analogies which do not (i.e., cases 1 and 2). In other words, ACME is not more powerful than SME and ours in dealing with these three.

4.3 An Application in JULIA

Our analogical mapping mechanism is part of the case-based reasoner [Shi88] in JULIA, an intelligent caterer's advisory system [CK86,Kol87]. Each problem case in JULIA has problem and solution parts. The problem part describes its problem functionally in terms of goals and constraints while the solution part contains a solution plan and the reasoning history.

Since problem cases are represented in a hierarchical structure, JULIA maps the top level problem structures first. It tries to identify similarity between two functional structures. In a frame-based representation, it is straightforward to identify the same functional components (e.g., goals and constraints). JULIA starts mapping with goals between problems: if the goals fully match, the mapping proceeds to constraints; in case of a partial match, which means some goals match but others do not, only the matched goals will be considered for possible transfer; otherwise, the mapping fails. Mapping then proceeds to constraints on only the matched goals to establish correspondences between them. For example, JULIA would consider two cost constraints LOW-COST and INEXPENSIVE[3] similar, because they are functionally the same in that they both constrain the cost. Then, should LOW-COST and EXPENSIVE be considered similar, too? JULIA views that they, too, are functionally similar due to the same reason. However, these do not have as much in common semantically as LOW-COST and INEXPENSIVE.

This problem will be resolved during actual transfer and there the degree of semantic similarity determines the degree of learning involved. Suppose the source case made the following inference during its problem solving:

> **If** c-cost(LOW-COST)
> **then** c-ingredient-cost(LOW-COST) and c-cooking(LOW-COST)
> **because** cost of dish is cost of ingredients plus cooking cost

Then, during analogical transfer, JULIA will try to transfer the previous inference rule with the similar concept INEXPENSIVE using its justification ("because") clause. In other words, JULIA hypothesizes a rule substituting LOW-COST in the rule for INEXPENSIVE, seeing if the justification previously used is similarly applicable. Since the justification also holds for the target case, the new rule will be transferred:

> **If** c-cost(INEXPENSIVE)
> **then** c-ingredient-cost(INEXPENSIVE) and c-cooking(INEXPENSIVE)
> **because** cost of dish is cost of ingredients plus cooking cost

However, if it were EXPENSIVE, the similar inference may not be true because not every ingredient needs to be expensive to make a dish expensive.

[3] INEXPENSIVE ranges from low cost to moderate cost so that its meaning is slightly broader than that of LOW-COST.

5 Summary and Conclusions

We have shown that successful mapping may not guarantee successful transfer of analogy. Analogical mapping only predicts a possibility of transfer which should subsequently be verified. Even when analogical mapping eventually leads to analogy transfer, successful analogical mapping may not directly dictate what is to be transferred from the source case to the target problem.

An analogical mapping algorithm has been introduced as a recursive application of two-step hierarchical mapping: first map the two problem structures systematically under structural similarity and then decompose them into the next lower level structures. Structural similarity is identified during this mapping process, while semantic similarity is checked during analogical transfer. These two processes together guarantee the correctness of analogy transfer.

Acknowledgments

This work could not have been done without the support and guidance of Janet L. Kolodner. I would like to thank my fellow-students Patsy L. Holmes, David Wood, Mark A. Graves, Joel Martin, and Mike Redmond for useful comments and discussion on earlier versions of this paper. I would like to especially thank Keith J. Holyoak for his valuable comments.

References

[Bur86] M.H. Burstein. Concept formation by incremental analogical reasoning and debugging. In R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, Kaufmann, Los Altos, CA, 1986.

[CK86] R.E. Cullingford and J.L. Kolodner. Interactive advice giving. In *Proceedings of the 1986 IEEE International Conference on Systems, Man, and Cybernetics*, 1986.

[CM85] J.G. Carbonell and S. Minton. Metaphor and commonsense reasoning. In J.R. Hobbs and R.C. Moore, editors, *Formal Theories of the Commonsense World*, Ablex, Norwood, NJ, 1985.

[FFG86] B. Falkenhainer, K.D. Forbus, and D. Gentner. The structure-mapping engine. In *Proc. AAAI-86*, 1986.

[Gen83] D. Gentner. Structure-mapping: a theoretical framework for analogy. *Cognitive Science*, 7:155–170, 1983.

[HT88] K.J. Holyoak and P. Thagard. Analogical mapping by constraint satisfaction. 1988. Unpublished manuscript, Department of Psychology, UCLA.

[Kol87] J.L. Kolodner. Capitalizing on failure through cased-based inference. In *Proceedings of the Ninth Annual Conference of the Cognitive Science Society*, Seattle, Washington, July 1987.

[Pol54] G. Polya. *Mathematics and Plausible Reasoning: Induction and Analogy in Mathematics.* Volume 1, Princeton University Press, Princeton, NJ, 1954.

[Shi88] H.S. Shinn. Abstractional analogy: a model of analogical reasoning. In *Proceedings of the DARPA Workshop on Case-Based Reasoning*, May 1988.

[Win80] P.H. Winston. Learning and reasoning by analogy. *Comm. ACM*, 23(12):689–703, 1980.

[Win82] P.H. Winston. Learning new principles from precedents and exercises. *Artificial Intelligence*, 19:321–350, 1982.