Cross Domain Analogies for Learning Domain Theories

Matthew Klenk and Ken Forbus
Qualitative Reasoning Group, Northwestern University
2133 Sheridan Rd, Evanston, IL 60657 USA
{m-klenk, forbus}@northwestern.edu

Abstract

Analogical reasoning has long been seen as a powerful way of extending the reach of ones knowledge. One product of analogical reasoning is analogical learning in which the result of the comparison increases our understanding of some domain. This work describes a method for learning new domain theories by analogy. We use analogies between pairs of problems and worked solutions to create a domain mapping between a familiar and a new domain. This mapping allows us to initialize the new domain. After this initialization, another analogy is made between the domain theories themselves providing additional conjectures about the new domain. An experiment is described where the system learns to solve rotational kinematics problems by analogy with translational kinematics problems, outperforming a version of the system that is incrementally given the correct domain theory.

Keywords: Analogy; Learning

Introduction

Psychologists and cognitive scientists have long hypothesized that cross-domain analogy is a core aspect of human cognition. It has been studied in the context of the ability to adapt to new domains quickly (Collins & Gentner 1987, Gentner 2003), its usefulness in producing paradigm shifts in scientific thought (Gentner *et al.* 1997; Holyoak & Thagard 1989, Falkenhainer 1988), and also its lasting effects on future reasoning in the new domain (Rand *et al.* 1989). In order to exploit cross domain analogies, there must be a known base domain theory and a mapping between the objects and relationships in the two domains. This work assumes we have been told the base domain and provides a method for learning the domain mapping through analogies between explanations. This domain mapping drives an analogy between the known and unknown domain theories providing inferences in the new domain theory. This paper presents a system which uses computational models of analogy and similarity based retrieval to learn new domain theories.

We begin by discussing the representations, the domains, and rule based problem solver which we use to evaluate our theory. Then we describe the structure mapping theory of analogy and similarity and our computational models used in this work. Next, we explain how these computational models can be used to learn new domain theories via analogy. After walking through an example, we describe an experiment in which the system learns to solve rotational kinematics problems by analogy with translational kinematics problems, outperforming a version of the system that is incrementally given the correct domain theory. We close with a discussion of related work and our plans for the future.

Representations and Problem Solving

Representing physics problems requires a broad background of everyday knowledge, including the object and event types found in such problems. We use the ResearchCyc[1] knowledge base contents, augmented with our own extensions. Our extensions concern QP theory (Forbus 1984) and problem-solving strategies, and are small compared to the 30,000+ concepts and 8,000+ predicates already defined in the KB. Thus, objects, relations, and events that appear in physics problems such as "rotor", "car", and "driving" are already defined in the ontology for us, rather than being created specifically for this project.

Example Problem and Worked Solution

All problems and worked solutions used in this work were taken from the same physics textbook (Giancoli 1991). We represent the problems and worked solutions from the textbook as cases of predicate calculus facts. Consider the problem of "How long does it take a car to travel 30m if it accelerates from rest at a rate of 2 m/s^2?" (Example 2-6, p. 26). This problem is represented in our system as a case of 10 facts, a subset of which appears in Figure 1.

```
(isa Car-2-6 Automobile)
(isa Acc-2-6 TransportWithMotorizedLandVehicle)
(objectStationary (StartFn Acc-2-6) Car-2-6)
(primaryObjectMoving Acc-2-6 Car-2-6)
(valueOf
  ((QPQuantityFn Distance) Car-2-6 Acc-2-6)
  (Meter 30))
```

Figure 1: Problem 2-6 Representation (partial)

Worked solutions are represented at the level of explained examples found in textbooks, which is more abstract than a proof or problem-solving trace. The worked solutions, like the problems themselves, are represented in a general way not in the internal language of the problem solver. For example, the worked solution for problem 2-6 consisted of four steps:

1. Categorize the problem as a constant acceleration linear mechanics problem
2. Instantiate the distance by velocity time equation ($d = v_i t + .5at^2$)
3. Because the car is stationary at the start of the event infer that its velocity is zero ($v_i = 0$ m/s)

[1] http://research.cyc.com/

4. Solve the equation for t (t = 5.8s)

Figure 2 shows how step three is represented.

```
(isa Gia-2-7-Step-3 WorkedSolutionStep)
(hasSteps Gia-2-7-WS Gia-2-7-Step-3)
(priorStep Gia-2-7-Step-3 Gia-2-7-Step-2)
(stepType Gia-2-7-Step-3 AssumingValue)
(stepUses Gia-2-6-WS-Step-3
 (objectStationary (StartFn Acc-2-6) Car-2-6))
(stepResult Gia-2-6-WS-Step-3
 (valueOf
  (AtFn ((QPQuantityFn Speed) Car-2-6)
        (StartFn Acc-2-6))
  (MetersPerSecond 0)))
```

Figure 2: Problem 2-6 worked solution step 3

Domain Theories for Problem Solving

Our domain theories consist of *encapsulated histories* (Forbus 1984) representing equations. We use encapsulated histories because they permit constraints to be placed on time itself, which is necessary for representing equations. Equations like the velocity/time law above involve events and their durations (e.g., translational motion under constant acceleration).

Figure 3 illustrates the encapsulated history representing the equation of velocity as a function of time (v_f=v_i+at). There are two participants, `theObject` and `theEvent`, which must satisfy their type constraints, the abstractions `PointMass` and `Constant1DAccelerationEvent` respectively. Furthermore, the conditions of the encapsulated history must be satisfied in order to instantiate it and conclude its consequences. In this case, it is necessary that `theObject` be the object moving in `theEvent`. The compound form shown in Figure 3 is automatically translated into a set of predicate calculus facts.

```
(def-encapsulated-history
       VelocityByTime-1DConstantAcceleration
 :participants
 ((theObject :type PointMass)
  (theEvent :type Constant1DAccelerationEvent))
 :conditions
  ((primaryObjectMoving theEvent theObject))
 :consequences
  ((equationFor VelocityByTime
    (mathEquals
       (AtFn (Speed theObject)(EndFn theEvent))
       (PlusFn (AtFn (Speed theObject)
                     (StartFn theEvent))
        (TimesFn (AtFn (Acceleration theObject)
```

Figure 3: Example Encapsulated History

Learning the encapsulated histories of a new domain via analogy in terms of participants, conditions and consequences is the goal for this work. To evaluate the accuracy of learned abstract knowledge, it must be able to solve problems. The physics problems in this work all ask for the values of specific quantities. Our system uses rule-based reasoning for modeling decisions and instantiates encapsulated histories, representing equations, from its domain theory to solve for the quantity asked for in the problem.

Structure-mapping and Analogy

We use Gentner's (1983) structure-mapping theory, which postulates that analogy and similarity are based on structural alignment between two representations (the *base* and *target*) to find the maximal structurally consistent match between them. The Structure Matching Engine (SME) simulates the process of analogical matching between a base and target (Falkenhainer *et al.* 1989). The output of this process is one or more *mappings*. A mapping is a set of *correspondences* representing a construal of what items (*entities* and *expressions*) in the base go with what items in the target. Mappings include a *structural evaluation score* indicating the strength of the match, and *candidate inferences* which are conjectures about the target using expressions from the base which, while unmapped in their entirety, have subcomponents that participate in the mapping's correspondences. SME operates in polynomial time, using a greedy algorithm (Forbus & Oblinger 1990).

MAC/FAC (Forbus *et. al.* 1994) models similarity-based retrieval. The inputs are a case, the *probe*, and a library of cases. The first stage (MAC) uses a computationally cheap, non-structural matcher to filter candidates from a pool of memory items, returning up to three if they are very close. The second stage (FAC) uses SME to compare the cases returned by MAC to the probe and returns the best candidate (or candidates, if they are very similar). Both SME and MAC/FAC have been used as performance systems in a variety of domains and as cognitive models to account for numerous psychological results (Forbus 2001).

Different domains are often represented using different predicates, especially when they are first being learned and underlying commonalities with previous knowledge have yet to be found. *Minimal ascension* (Falkenhainer 1988) is one method for matching non-identical predicates. If two predicates are part of a larger aligned structure and share a close common ancestor in the taxonomic hierarchy, as defined by the ResearchCyc ontology, then SME can include them in the mapping. Figure 4 demonstrates an example in which the solution steps. The objects, events, and steps of the worked solutions are already in correspondence allowing `primaryObjectMoving` and `objectRotating` to map to each other based upon minimal ascension due to both of them being children of `objectMoving` in the Cyc ontology.

Base Expression:
```
(stepUses Gia-2-6-WS-Step-2
  (primaryObjectMoving Acc-2-6 Car-2-6))
```
Target Expression:
```
(stepUses Gia-8-5-WS-Step-2
  (objectRotating Acc-8-5 Rotor-8-5))
```

Figure 4: Minimal ascension maps `primaryObjectMoving` to `objectRotating`

Analogical Learning of Domain Theories

Our system learns a domain theory by using multiple analogies. Learning is invoked after failing to solve a problem. The system is given the worked solution for that

1. Retrieve analog using the target worked solution as a probe in MAC/FAC
2. Use SME to crate a match between the analog and worked solution
3. Retrieve correspondences from resulting mappings
4. Create domain mapping by selecting correspondences in the base element appears in the base theory
5. Initialize target domain theory using these correspondences
6. Use SME to create an analogy between the base and target domain theories constrained by the domain mapping
7. Transfer domain theory using candidate inferences
8. Verify learned domain theory by retrying the failed problem
9. If failure, go once more to step 1, else accept new domain knowledge as correct.

Figure 5: Analogical domain learning

problem, as a student would find in a textbook. It uses this worked solution to create conjectures about knowledge in the new domain, using the algorithm outlined in Figure 5. The case library contains a set of worked solutions from the known domain. First, the worked solution for the failed problem is used as a probe to MAC/FAC, to retrieve an analogous worked solution from memory. A comparison is made using SME, with the retrieved worked solution constituting the base and the worked solution for the failed problem as the target. The mappings SME produces are then combined to create a *domain mapping*. The reason for combining multiple mappings is that each mapping may only cover some aspects of the worked solutions. The best mapping is used as a starting point, with correspondences drawn from the others included only if they do not violate the one-to-one constraint.

When the system gets the first problem in a new domain, its theory for that domain is empty. The domain mapping is used to initialize the new domain theory. For each encapsulated history from the base domain theory mentioned in the domain mapping, the system attempts to create an encapsulated history in the target domain. We currently require that every concept in the base encapsulated history is mentioned in the domain mapping, i.e., there are no analogy skolems where we must postulate a new predicate or category of entity.

The system also extends a partially learned, or just initialized, domain theory with another analogy. This time between the base and target domain theories themselves with the domain mapping acting as required correspondence constraints, ensuring that the overall domain theory is consistent. The resulting candidate inferences include conjectures about encapsulated histories in the new domain theory.

While powerful, analogies are not guaranteed to be sound. Consequently, we verify the newly proposed domain knowledge by trying again to solve the problem whose failure motivated the learning. If this problem is solved correctly, our system assumes that the new domain theory constructs are correct. Otherwise, it deletes both the new

a) Through how many turns does a centrifuge rotor make when accelerating from rest to 20,000 rpm in 5 min? Assume constant angular acceleration
b) A phonograph turntable reaches its rated speed of 33 rpm after making 2.5 revolutions, what is its angular acceleration?
c) Through how many turns does a centrifuge rotor make when accelerating from rest to 10,000 rpm in 270 Seconds? Assume constant angular acceleration
d) An automobile engine slows down from 3600 rpm to 1000 rpm in 5 seconds, how many radians does the engine turn in this time?
e) A centrifuge rotor is accelerated from rest to 20,000 rpm in 5 min, what is the averaged angular acceleration?

Figure 6: Evaluation Questions

domain theory constructs and the domain mapping, and tries one more time, considering the next best worked solution retrieved from memory. Next, we walk through an example of how encapsulated histories can be inferred in the domain of rotational mechanics.

Example

To better understand this algorithm, we describe how the system learns after failing to solve problem 'a' from Figure 6. Using the worked solution for this problem (a collection of 65 facts of the form in Figure 2), the system retrieves an analog from its memory using MAC/FAC, in this case the analogous worked solution retrieved is for the problem discussed previously "How long does it take a car to travel 30m if it accelerates from rest at a rate of 2 m/s^2?". Now the system generates the analogy between the retrieved worked solution and the worked solution to problem 'a'. In this case, the mathematical relationships are isomorphic, $d = v_i t + .5at^2$ and $\theta = \omega_i t + .5\alpha t^2$, which places the quantities between the domains into correspondence. It should be noted that SME handles partial matches allowing for the correspondences to be created even when the problems being compared are not completely analogous. The minimal ascension example from Figure 4 is also part of this mapping. Next, the correspondences of this mapping are extracted to create the domain-mapping, a subset of which appears in Table 1.

Once the domain mapping has been made, the system attempts to initialize the target domain theory. This is done by searching the domain mapping for encapsulated histories, in this example `DistanceByVelocityTime-1DConstantAcceleration` is an encapsulated history in the base domain. For each fact in the base domain theory mentioning `DistanceByVelocityTime-1DConstantAcceleration` we substitute subexpressions based upon the domain mapping. Now we have a target domain theory based directly the worked solution comparison.

Next, we attempt to extend this domain theory with a comparison between the base and target domain theories with the domain mapping acting as required correspondence constraints. We use the 41 facts of the linear mechanics encapsulated histories as the base, the 6 facts of the just

Table 1: Domain Mapping

Base Item	Target Item
`PointMass`	`RigidObject`
`ConstantLinear-AccelerationEvent`	`ConstantRotational-AccelerationEvent`
`primaryObjectMoving`	`objectRotating`
`Acceleration`	`AngularAcceleration`
`Speed`	`RateOfRotation`
`DistanceTravelled`	`AngularDistTravelled`
`Time-Quantity`	`Time-Quantity`
`DistanceByVelocityTime-1DConstantAcceleration`	`DistanceTime-Rotational`

initialized rotational mechanics domain theory as the target and we constrain the match by setting up required correspondences between the elements of the domain mapping. Now the shared structure between the quantities and conditions will produce candidate inferences involving the facts of the other encapsulated histories in the linear mechanics, base, domain theory. For example, the candidate inference shown in Figure 7 postulates that there is an encapsulated history based upon `VelocityByTime-1DConstantAcceleration`, an encapsulated history from the base domain that did not have a corresponding entity in the mapping, that has the operating condition of its object rotating during its event. The system then creates entities for all the analogy skolems, i.e. turning `(SkolemFn VelocityByTime-1DConstantAcceleration)` into `EHType-1523`, and assumes these facts into the rotational mechanics domain theory.

```
(qpConditionOfType
 (AnalogySkolemFn
       VelocityByTime-1DConstantAcceleration)
 (objectRotating :theEvent :theObject))
```

Figure 7: Analogical Domain Learning

Finally, we need to verify that the learned knowledge is accurate. This is done by attempting to solve problem 'a' again. Since the worked solution contains the answer, we can compare our computed answer against it. If they match, then we infer that the learned knowledge is correct. If the system gets the problem wrong, then we clear out the inferred encapsulated histories in the rotational mechanics domain theory and the entire process repeats one more time. One aspect of our future work is to better diagnose faults in our domain theories and domain mappings which we hope to incrementally improve overtime.

Experimental Results

To examine how well this analogical learning method works, we need a baseline. Our baseline *spoon-fed* system consists of the exact same problem-solver, but with analogical learning turned off. Instead, when the spoon-fed system receives a problem it cannot solve, it is given not just a worked solution, but whatever general encapsulated histories are needed to solve that specific target domain problem.

Method

Both systems begin with a linear kinematics domain theory, two worked solutions of linear kinematics problems, and hard-coded rules for problem-solving strategies and making modeling decisions. The systems are then tested on a serie of rotational mechanics problems. The testing materials are 5 problems, listed in Figure 7, and worked solutions. Learning curves were created by running 120 trials representing every possible ordering of the test materials. In each trial, after each problem, the system was given either the worked solution or encapsulated histories for that problem, depending on the condition. After each trial, the system's knowledge was reset.

Results

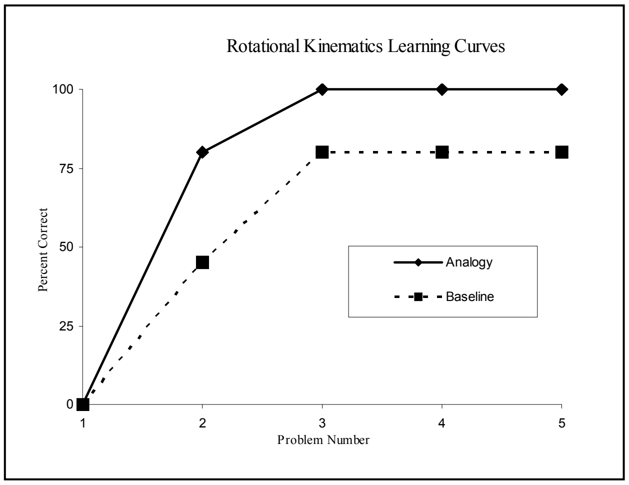

Figure 8: Experiment Results

Figure 8 compares the learning curves for the analogy and baseline conditions. After studying just one worked solution, the analogy system was able to solve next problem correctly 80 percent of time. Furthermore, the analogy system has perfect performance after working on any two test problems. The baseline system's ceiling was at 80 percent, and after one problem it was only able to get the next problem correct 45 percent of the time.

Further analysis of these results details the strength of the analogy approach. The baseline system failed to score above 80 percent of any of the conditions. The baseline system was unable to solve problem 'b' from Figure 6 regardless of what problems it has already seen, because none of the other problems use the same equation. The analogy system performed quite well, only in one situation did the analogical domain transfer fail to learn the whole rotational kinematics domain after seeing just one worked solution. This occurred when problem 'b' was the first problem. Problem 'b' makes no mention of a time quantity preventing a correspondence to be created for it. While a time quantity exists in both of these domains, it does not necessarily mean they should be aligned. The strength of the analogical approach is that transfer is guided by structural similarity. This is critical for broader application of this theory. For example, in linear and rotational dynamics, both domain theories have a mass quantity, but transfer is only possible when a domain mapping is made between mass, in linear dynamics, and moment of inertia, in rotational dynamics. (e.g. F=ma and *T*=*Iα*)

Related Work

Falkenhainer's (1988) PHINEAS is the closest system to our own in addressing these problems. PHINEAS which used comparisons of (simulated) behavior to create an initial cross-domain mapping that was subsequently used to create a partial theory for the new domain. It differs, however, in several significant ways: (1) We use analogies between problem explanations to drive the process, (2) We are learning quantitative, rather than qualitative, domain theories, which requires very different verification work, and (3) We are using a more psychologically plausible retrieval mechanism. Holyoak and Thagard's (1989) PI model uses a pragmatic theory of analogy to model solve variations of the radiation problems through schema induction. Our model of cross-domain analogy requires analogies between domain theories that are isolated from problem solving episodes, which PI does not allow. Other analogical problem solving systems have focused on using previous experiences to guide future problem solving. Examples include Veloso and Carbonell's (1993) Derivational Analogy, van Lehn's (1993) Cascade system and Ouyang and Forbus's (2006) APSS system. Our work differs by focusing on learning abstract domain knowledge, as opposed to search control knowledge.

Discussion

We have shown that a domain theory for solving physics problems can be learned via cross-domain analogies. Our experiment shows furthermore that such analogical learning can be very efficient, when the two domains are sufficiently similar. The process of constructing domain mappings by exploiting similarities in worked solutions, and using that mapping to import theories from one domain to another, is, we believe, a general and powerful process.

There are several directions we intend to pursue next. First, we have only tested this method with encapsulated histories, so we want to extend it to handle other types of domain knowledge such as model fragments and modeling knowledge. Second, we plan to integrate this algorithm into our Companion-based learning system (2007), which utilized within-domain analogical problem solving so that we can combine both methods of analogical learning. A central goal to the Companions project (Forbus & Hinrichs 2006) is to build a reasoning and learning agent that can operate for weeks and months at a time. Therefore, we plan to implement model-based diagnostic strategies to debug our analogically-derived domain theories, similar to the strategies used by de Koning *et al.* (2000) to diagnose misconceptions in student models. Finally, we plan to explore a broader range of domain pairs, including domains which are quite distant such as those found in system dynamics (Olsen 1966).

Acknowledgments

This research was supported by the Cognitive Science Program of the Office of Naval Research.

References

Collins, A. & Gentner, D. 1987. How people construct mental models. In D. Holland & N. Quinn (Eds.), *Cultural models in language and thought*. England: Cambridge University Press.

de Koning, K., Bredeweg, B., Breuker, J., and Wielinga, B. 2000. Model-Based Reasoning about Learner Behavior. *Artificial Intelligence*, 117(2).

Falkenhainer, B. 1988. Learning from Physical Analogies. Technical Report No. UIUCDCS-R-88-1479, University of Illinios at Urbana-Champaign. (Ph.D. Thesis)

Falkenhainer, B., Forbus, K. and Gentner, D. 1989. The Structure-Mapping Engine. *Artificial Intelligence.* 41.

Forbus, K. 1984. Qualitative process theory. *Artificial Intelligence* 24.

Forbus, K. 2001. Exploring analogy in the large. In Gentner, D., Holyoak, K., & Kokinov, B. (Eds.) *Analogy: Perspectives from Cognitive Science.* MIT Press.

Forbus, K., Gentner, D., & Law, K. 1994. MAC/FAC: A model of similarity-based retrieval. *Cognitive Science*, 19.

Forbus, K. & Hinrichs, T. 2006. Companion cognitive systems: a step toward Human-Level AI.

Forbus, K. & Oblinger, D. 1990. Making SME greedy and pragmatic. In *Proceedings of CogSci-1990*.

Gentner, D. 1983. Structure-mapping: A theoretical framework for analogy, *Cognitive Science* 7(2).

Gentner, D. 2003. Why we're so smart. In Gentner, D. and Goldin-Meadow, S. (Eds.), *Language in mind: Advances in the study of language and thought.* pp. 195-235. MIT Press.

Gentner, D., Brem, S., Ferguson, R.W., Markman, A.B., Levidow, B.B., Wolff, P., and Forbus, K. 1997. Analogical reasoning and conceptual change: A case study of Johannes Kepler. *The Journal of the Learning Sciences*, 6(1), 3-40.

Gentner, D. & Gentner, D. R. 1983. Flowing waters or teeming crowds: Mental models of electricity. In D. Gentner & A. Stevens (Eds.), *Mental Models*. Lawrence Erlbaum Associates.

Giancoli, D. 1991. Physics: Principles with Applications. 3rd Edition. Prentice Hall.

Holyoak, K. J. & Thagard, P. 1989. A computational model of analogical problem solving. In S. Vosniadou & A. Ortony (Eds.), *Similarity and analogical reasoning*. New York: Cambridge University Press.

Klenk, M. & Forbus, K. 2007. Measuring the level of transfer learning by an AP Physics problem-solver. In *Proceedings of AAAI-07*. Vancouver, CA.

Olson, H. 1966. *Solutions of Engineering Problems by Dynamical Analogies*, D. Van Nostrand.

Ouyan, T. & Forbus, K. 2006. Strategy variations in analogical problem solving. *Proceedings of AAAI-06*. Boston, MA.

Rand, S., Feltovich, P., Coulson, R, and Anderson, D. 1989. Multiple analogies for complex concepts: antidotes for analogy-induced misconception in advanced knowledge acquisition. In S. Vosniadou & A. Ortony (Eds.), *Similarity and analogical reasoning*. New York: Cambridge University Press.

VanLehn, K. 1998. Analogy Events: How examples are used during problem solving. *Cognitive Science* 22(19).

Veloso, M. & Carbonell, J. 1993. Derivational analogy in PRODIGY: Automating case acquisition, storage, and utilization. *Machine Learning*, 10.