Domain Transfer via Cross-Domain Analogy

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

Abstract

Analogical learning has long been seen as a powerful way of extending the reach of one's knowledge. We present *domain transfer via analogy* (DTA) as a method for learning new domain theories via cross-domain analogy. Our model uses analogies between pairs of textbook example problems, or *worked solutions*, to create a domain mapping between a familiar and a new domain. This mapping allows us to initialize a new domain theory. After this initialization, another analogy is made between the domain theories themselves, providing additional conjectures about the new domain. We present two experiments in which our model learns rotational kinematics by an analogy with translational kinematics, and vice versa. We compare these learning rates against a version of the system that is incrementally given the correct domain theory.

Keywords: Analogy; Learning

Introduction

Cognitive scientists have long argued that cross-domain analogy is an important element in people's adaptability to new situations. It has been studied in the contexts of learning new domains quickly (Gentner & Gentner 1983; Gentner 2003), producing paradigm shifts in scientific thought (Gentner *et al.* 1997; Holyoak & Thagard 1989; Falkenhainer 1988), and future reasoning in new domains (Rand *et al.* 1989). While analogies are powerful, Gick and Holyoak (1980) found that people have difficulties spontaneously producing cross-domain analogies, but they succeed when given hints. Collins & Gentner (1987) report that successful cross-domain analogies require a known base domain and a *domain mapping*. A domain mapping consists of correspondences between the objects and relationships of the two domains. Additional evidence of the utility of cross-domain analogies comes from textbook authors, who routinely use them to explain concepts. The linear kinematics section of the textbook used for this work (Giancoli 1991) contains eight worked through examples, or *worked solutions*. These include instantiations of all four linear kinematics equations. In the rotational kinematics section, however, there are only two worked solutions, neither of which utilizes two of the equations necessary for completing problems from this chapter. The summary section of the rotational motion chapter invites the learner to use analogy to fill in the details: "The dynamics of rotation is analogous to the dynamics of linear motion" (p. 197, Giancoli 1991). Our model seeks to understand this kind of learning.

One method students use to learn physics is by completing problem sets. Afterwards, students compare their reasoning against worked solutions to refine their understanding. Our model, *domain transfer via analogy* (DTA), uses worked solutions as a starting point for learning a new domain theory through multiple cross-domain analogies. Given a known base domain with worked solutions and a worked solution from a new domain, DTA begins by learning the domain mapping. Our model uses this domain mapping to initialize the new target domain theory and constrain an analogy between the base and target domain theories. This analogy results in inferences about the new domain that are added to the agent's knowledge after verification. This paper describes DTA, which uses existing computational models of analogy and similarity based-retrieval to learn new domain theories.

We begin by discussing the structure-mapping theory of analogy and the computational models used in this work. Next, we describe the representations for the problems, worked solutions and domain theories our model uses. We then outline the DTA method of domain transfer and illustrate this process with an example. We evaluate our model via two experiments in which our model learns rotational kinematics by an analogy with translational kinematics, and vice versa. We compare these learning rates against a version of the system that is incrementally provided the correct domain theory. We close with a discussion of related work and implications for future work.

Structure-mapping and Analogy

We use Gentner's (1983) structure-mapping theory, which postulates that analogy and similarity are computed via structural alignment between two representations (the *base* and *target*) to find the maximal structurally consistent match. A structurally consistent match is based upon three constraints: *tiered-identicality*, *parallel connectivity*, and *one-to-one mapping*. The tiered-identicality constraint provides a strong preference for only allowing identical predicates to match, but allows for rare exceptions. Parallel connectivity means that if two statements are matched then their arguments must also match. The one-to-one mapping constraint requires that each element in the base corresponds to at most one element target, and vice versa. To explain why some analogies are better than others, structure-mapping uses the principle of *systematicity*: a preference for mappings that are highly interconnected and contain deep chains of higher order relations.

The Structure Mapping Engine (SME) simulates the structure-mapping 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 merge algorithm (Forbus & Oblinger 1990). If one mapping is clearly the best, it produces only that mapping, but it can produce up to three if there are close competing matches. Pragmatic constraints can be specified to guide the alignment process, e.g., one can state that particular entities and/or statements must match, or must not match. We use this capability to guide abstract domain mappings with correspondences found via the more concrete worked solution mappings.

MAC/FAC (Forbus *et al.* 1994) is a computational model of similarity-based retrieval. The inputs are a case, the *probe*, and a *case library*. The first stage (MAC) uses a computationally cheap, non-structural matcher to filter candidate cases from the case library, 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 as determined by the structural evaluation score of each match (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 a number of psychological findings (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, then SME can include them in the mapping. Figure 1 demonstrates an example of two expressions that are placed in correspondence because they have identical predicates, `stepUses`. The entities for the objects, events, and steps are already in correspondence, given the rest of the mapping. In order to include these expressions in the mapping, `primaryObjectMoving` would have to map to `objectRotating`. Minimal ascension allows this mapping because both relationships are children of `objectMoving` in the ResearchCyc[1] ontology, the taxonomic hierarchy used in this work.

```
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 1: Minimal ascension aligns
`primaryObjectMoving` to `objectRotating`

Representations and Problem Solving

Before we describe how cross-domain analogy enables the learning of new domain theories, we must describe the representation of the problems, worked solutions, and domain theories used by our model. The representations used in this work are in CycL, the predicate calculus language of the ResearchCyc knowledge base (Matuszek *et al.* 2006). The representations use the ontology of ResearchCyc, plus our own extensions. These 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. This reduces the degree of tailorability in our experiments.

Example Problem and Worked Solution

All the problems used in this work were taken from the same physics textbook (Giancoli 1991). We represent the problems and worked solutions as cases, consisting of sets 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 problem 2-6, p. 26). This problem is represented in our system as a case of nine facts, shown in Figure 2. First, there are two entities in the problem, a transportation with land vehicle event, `Acc-2-6`, and an automobile, `Car-2-6`. Next, there are 4 facts describing the motion of `Car-2-6` during `Acc-2-6`. The last 3 facts describe the case and the question of the problem.

---

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

```
(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 DistanceTravelled) Car-2-6 Acc-2-6)
        (Meter 30))
(valueOf  (MeasurementAtFn ((QPQuantityFn Acceleration) Car-2-6) Acc-2-6)
        (MetersPerSecondPerSecond 2))
(isa Gia-Query-2-6 PhysicsQuery)
(hypotheticalMicrotheoryOfTest Gia-Query-2-6 Gia-2-6)
(querySentenceOfQuery Gia-Query-2-6
        (valueOf ((QPQuantityFn Time-Quantity) Acc-2-6) Duration-2-6)))
```

Figure 2: Example problem 2-6 representation

Worked solutions are represented at the level of explained examples found in textbooks. They are neither deductive proofs nor problem-solving traces produced by our solver. Because the textbook has only a limited number of worked solutions for rotational kinematics, in instances where there are no worked solutions, we created our own from problems at the end of the chapter. The worked solutions, like the problems themselves, are represented generally, not in the internal control vocabulary of the problem-solver. The worked solution for this example problem consists 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)
4. Solve the equation for t (t = 5.8s)

```
(isa Gia-2-6-WS-Step-3 WorkedSolutionStep)
(hasSolutionSteps Gia-2-6-WorkedSolution Gia-2-6-WS-Step-3)
(priorSolutionStep Gia-2-6-WS-Step-3 Gia-2-6-WS-Step-2)
(stepOperationType Gia-2-6-WS-Step-3
        DeterminingSpecificScalarOrVectorValuesFromContext)
(stepUses Gia-2-6-WS-Step-3
        (objectStationary (StartFn Acc-2-6) Car-2-6))
(stepResult Gia-2-6-WS-Step-3
        (valueOf
                (MeasurementAtFn ((QPQuantityFn Speed) Car-2-6) (StartFn Acc-2-6))
                (MetersPerSecond 0)))
```

Figure 3: Representation for step 3, inferring that the car's velocity is 0 m/s

The entire worked solution consists of 38 facts. Figure 3 shows the predicate calculus representation for the third worked solution step. The first four facts indicate the type of solution step and its sequential position in the worked solution. The last two facts state that the step uses the fact that Car-2-6 is stationary at the beginning of Acc-2-6 to infer that its speed at that point is 0 m/s.

Domain Theories for Problem Solving

Our domain theories consist of *encapsulated histories* (Forbus 1984) representing equations. Encapsulated histories, unlike model fragments, permit constraints to be placed on the duration of events and time intervals. This is necessary for representing equations, such as the velocity/time law required to solve the problem above which involves events as well as their durations. During problem solving, our system instantiates applicable encapsulated histories to determine which equations are available.

```
(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) theEvent)
                                   (Time-Quantity theEvent)))))))
```

Figure 4: Definition for the velocity by time encapsulated history

Figure 4 shows the definition for the encapsulated history representing the equation $v_f = v_i + at$, velocity as a function of time. 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 4 is automatically translated into a set of predicate calculus facts for use in our system.

Solving physics problems requires a number of modeling decisions. For example, in a scenario of dropping a ball off a building, one must view the ball as a point mass, the falling event as a constant acceleration event, and that the acceleration of the ball as equal to Earth's gravity. Our system currently uses hand coded rules to make these decisions. While this is sufficient for the goals of this work, our future plans involve integrating more robust modeling decision methods.

The objective of domain transfer via analogy (DTA) is learning domain theories that are represented with schema-like knowledge. In this work, DTA learns the encapsulated histories of a new domain via cross-domain analogy, in terms of participants, conditions and consequences. 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 solves for quantities in three ways. First, the quantity may be given directly in the problem. Second, a modeling decision could indicate the appropriate value for the quantity (e.g. the object is in projectile motion on Earth, and air resistance is ignored; Therefore, our system assumes the acceleration on the object to be 10 m/s^2). Third, our system is able to instantiate an encapsulated history with a consequence of an equation mentioning the desired quantity. In this case, our system recursively solves for the other parameters in the equation before solving the equation for the sought after quantity using algebra routines inspired by the system described in Forbus & De Kleer (1993).

Domain Transfer via Analogy

Domain transfer via analogy (DTA) learns a new (target) domain theory using multiple cross-domain analogies. In this work, DTA transfers encapsulated histories from a known base domain to the target domain theory. Analogical learning is invoked after failure to solve a problem in the new domain. Our model is provided the worked solution for that problem. DTA uses this worked solution to create conjectures about knowledge in the new domain, via the algorithm outlined in Figure 5. The algorithm contains four main sections, learning the domain mapping, initializing the new domain theory, extending the new domain theory, and verifying the learned knowledge. We describe each in turn.

```
+---------------------------------------------------------------------------------------------------+
|  Domain Transfer via Analogy (DTA) method:                                                        |
|  1.  Learn the domain mapping                                                                      |
|         o   Retrieve analog using MAC/FAC, with the worked solution to the failed problem as the probe. |
|         o   Use SME to create mappings between the retrieved analog and the worked solution.       |
|         o   Create domain mapping by selecting correspondences in which the base element appears in the base domain |
|             theory.                                                                                |
|  2.  Initialize target domain theory using the domain mapping                                      |
|  3.  Extend target domain theory                                                                   |
|         o   Use SME to create a match between the base and the target theories constrained by the domain mapping |
|         o   Transfer domain theory contents using the candidate inferences                         |
|  4.  Verify learned domain theory by attempting the failed problem again                           |
|         o   If failure, go once more to step 1.  Otherwise, accept new target domain knowledge as correct |
+---------------------------------------------------------------------------------------------------+
```

Figure 5: The DTA method for domain learning via cross-domain analogy

Learn the Domain Mapping

DTA learns the domain mapping through a comparison between worked solutions from the two domains. Using the worked solution for the failed problem as a probe to MAC/FAC, our model retrieves an analogous worked solution from a case library containing worked solutions from the known domain. Next, SME is used to find an analogy between the retrieved worked solution and the worked solution to the failed problem. The output of this process is up to three mappings, each containing a set of correspondences which are used to form the domain mapping. Because SME can produce multiple mappings, our model starts with the best mapping, as determined by the structural evaluation score. Then, it iterates through the rest of the mappings adding the correspondences to the domain mapping that do not violate the one-to-one constraint. For example, if the best mapping had a correspondence between `PointMass` and `RigidObject` and the second mapping had a correspondence between `PointMass` and `Ball`, the domain mapping would only include the mapping between `PointMass` and `RigidObject`. The reason for combining multiple mappings is that each mapping may only cover some aspects of the worked solutions.

Initialize the Target Domain Theory

When the system attempts the first problem in a new domain, its theory for that domain is empty. DTA uses the domain mapping to initialize the new domain theory. For each encapsulated history from the base domain theory mentioned in the domain mapping, our model attempts to create an encapsulated history in the target domain. Before transferring the encapsulated history, our model verifies that all of the quantities and types mentioned in the encapsulated history appear in the domain mapping. If they do not, this encapsulated history is not transferred to the target domain theory. If they are, then using the domain mapping, our model proceeds by substituting concepts in the base encapsulated history with the appropriate concepts from the target domain with one exception. We do not substitute numbers in the base domain with numbers from the target. This is because we assume these correspondences between numbers to be spurious (e.g. 1 in the one domain should not be considered as 2 in another domain). The resulting encapsulated history is then added to the target domain theory.

Extend the Target Domain Theory

After initializing the target domain theory, DTA extends it through another cross-domain analogy. This time the comparison is between the base and target domain theories themselves. The base and target domain theories consist of the facts representing the encapsulated histories. Our model constrains this match with the domain mapping, ensuring that the overall domain theory is consistent. Recall that SME forms candidate inferences, i.e., conjectures about the target, using expressions from the base which are partially mapped. Because the base domain theory is made up encapsulated histories, these candidate inferences describe corresponding encapsulated histories in the target domain theory. Before adding these candidate inferences to the target domain theory, our model processes them in two ways. First, the candidate inferences may include *skolem* entities which are entities appearing in the base but have no correspondence in the mapping. In this case, our model creates a new entity for each skolem to be assumed in the target domain theory. Second, if a number is mentioned in the inference, as during the initialization step, we substitute the corresponding number from the base. After these processes, the candidate inferences representing new encapsulated histories are stored into the target domain theory.

Verify the Learned Knowledge

While powerful, cross-domain analogies are risky and frequently contain invalid inferences. Therefore, DTA verifies the newly proposed encapsulated histories by retrying the problem whose failure began the entire process. If this problem is

solved correctly, our system assumes that the newly acquired domain theory is correct. Otherwise, our model forgets both the new domain theory and the domain mapping, and attempts the entire process one more time, removing the analog worked solution from the case library. Currently, we only let the model try two different analogous worked solutions to learn an acceptable target domain theory. In the future, this parameter could be under control of the system. Next, we describe an example of our model's operation.

Example

To better understand how DTA uses multiple cross-domain analogies to transfer domain theories, we describe an example of how it learns rotational kinematics through an analogy to linear kinematics. The system begins with a linear kinematics domain theory and worked solutions. Because the system has no knowledge, i.e., encapsulated histories, of rotational kinematics, it fails to solve the following problem, "Assuming constant angular acceleration, through how many turns does a centrifuge rotor make when accelerating from rest to 20,000 rpm in 5 min?" Because the system failed, we invoke DTA by providing the worked solution to this problem. Using this worked solution as a probe, the model retrieves an analog from its case library of worked solutions from linear kinematics 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?"

Our model uses an analogy between these two worked solutions to produce the domain mapping necessary for cross-domain analogy. 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 correspondences to be created even when the mathematical relationships in the problems being compared are not completely isomorphic. The minimal ascension example from Figure 1 is also part of this mapping. Next, the mapping's correspondences are extracted to create the domain mapping, a subset of which appears in Table 1.

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

Table 1: Domain Mapping

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 from the base domain. In this example, `DistanceByVelocityTime-1DConstantAcceleration` is returned. This encapsulated history contains two types, `PointMass` and `ConstantLinear-AccelerationEvent` and four quantities, `Acceleration`, `Speed`, `Time-Quantity` and `DistanceTravelled`. All of these elements appear in the domain mapping, therefore our model is able to transfer this encapsulated history to the target domain. For each fact in the base domain theory mentioning `DistanceByVelocityTime-1DConstantAcceleration`, we substitute all subexpressions based upon the domain mapping. This results in a new encapsulated history, `DistanceTime-Rotational`, which represent the rotational kinematics equation, $\theta = \omega_i t + .5\alpha t^2$. Our model initializes the target domain theory by adding this new encapsulated history.

Next, our model attempts to extend this new domain theory with an analogy between the base and target domain theories themselves. To maintain consistency, this analogy is constrained with the domain mapping acting as required correspondence constraints. The 41 facts describing the linear mechanics encapsulated histories make up the base, and the 6 facts of the newly initialized rotational mechanics domain theory are the target. As expected, the sole target encapsulated history maps to the corresponding linear mechanics encapsulated history. This mapping includes the quantities, conditions and types of these encapsulated histories. These correspondences result in candidate inferences involving the facts of the other encapsulated histories from the base. For example, the candidate inference shown in Figure 6 suggests that there is an encapsulated history in the target analogous to the `VelocityByTime-1DConstantAcceleration` linear mechanics encapsulated history. This candidate inference states that the suggested encapsulated history has the operating condition of its object rotating during its event. The `AnalogySkolemFn` expression indicates that there was no corresponding entity in the target.

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

Figure 6: Candidate inference suggesting a condition of an encapsulated history type

Therefore, to extend the target domain theory, DTA creates entities for all the analogy skolems, i.e. turning `(AnalogySkolemFn VelocityByTime-1DConstantAcceleration)` into `EHType-1523`, and assumes these facts into the rotational mechanics domain theory.

Finally, we need to verify that the learned knowledge is accurate. This is done by attempting to solve the original problem 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, first, erase the domain mapping and the inferred encapsulated histories in the rotational mechanics domain theory. Then, the entire process repeats one more time with this worked solution removed from the case library. One aspect of our future work is to better diagnose faults in our domain theories and domain mappings, which we hope to incrementally improve over time.

Evaluation

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 DTA turned off. Instead of providing the spoon-fed system a worked solution after failing a problem, we provide it with the general encapsulated histories needed to solve that specific problem. We performed two experiments, one in which our model learns rotational kinematics by an analogy with linear kinematics, and the other learns linear kinematics based upon rotational kinematics. The kinematics problems for both domains are listed in Figure 7. Both systems begin with the necessary rules for problem-solving strategies and modeling decisions. The systems are then tested on a series of problems from the target domain.

Linear Kinematics	*Rotational Kinematics*
a) How long does it take a car to travel 30m if it accelerates from rest at a rate of 2 m/s^2?	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) We consider the stopping distances from a car, which are important for traffic saftey and traffic design. The problem is best deal with in two parts: (1) the time between the decision to apply the brakes and their actual application (the "reaction time"), during which we assume a=0; and (2) the actual braking period when the vehicle decelerates. Assuming the starting velocity is 28 m/s, the acceleration is -6.0 m/s^2, and a reaction time of .5 s, What is the stopping distance?	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?
c) A baseball pitcher throws a fastball with a speed of 44 m/s. It has been observed that pitchers accelerate the ball through a distance of 3.5 m. What is the average acceleration during the throwing motion?	e) A centrifuge rotor is accelerated from rest to 20,000 rpm in 5 min, what is the averaged angular acceleration?
d) Suppose a ball is dropped from a 70m tower how far will it have fallen after 3 seconds?	
e) A jetliner must reach a speed of 80 m/s for takeoff. The runway is 1500 M long, what is the constant acceleration required?	

Figure 7: Evaluation materials

Experiment 1

In this experiment, we use linear kinematics as the base domain and rotational kinematics as the learning domain. Learning curves were created by running 120 trials representing every possible ordering of the problems in the learning domain. 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.
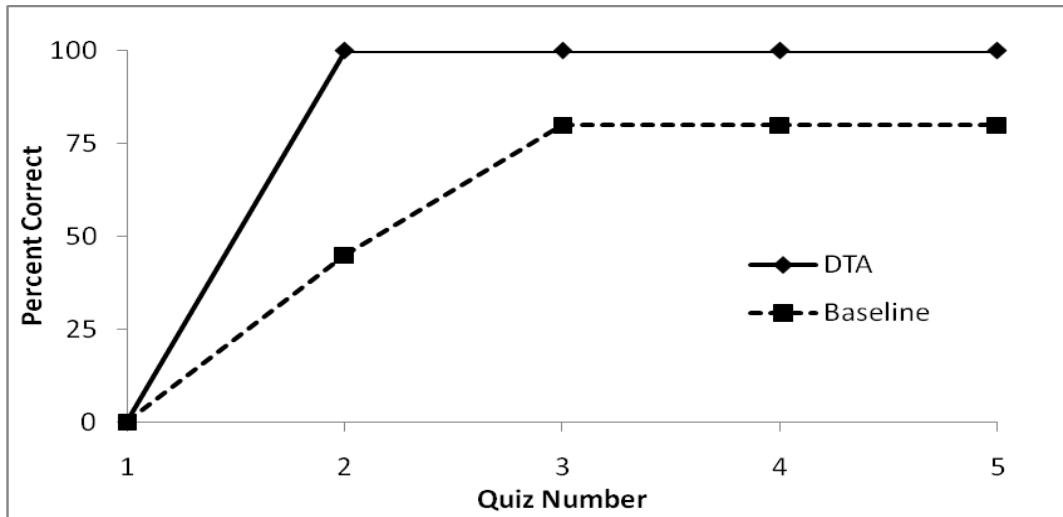

Figure 8: Rotational mechanics learning curves

Figure 8 compares the rotational kinematics learning rates for the analogy and baseline conditions. The analogy system using DTA exhibited perfect transfer. That is, after studying just one worked solution, the analogy system was able score 100% on the rest of the problems. Because the analogy system performed perfectly, there were only 120 transfer attempts, all of which succeeded. Of these, 72 attempts (60%) required an additional retrieval to generate a successful cross-domain analogy. After one problem, the baseline system was only able to solve the next problem 45 percent of the time. Also, the baseline system's ceiling was at 80 percent. This was due to the fact it was unable to solve rotational kinematics problem 'b' from Figure 7 regardless of what problems it had already seen, because none of the other problems use the same equation. DTA overcomes this through the analogy between the domain theories themselves allowing it to infer equations not mentioned explicitly in the worked solution.

Experiment 2

This experiment followed the same form as the previous experiment but with the opposite base and learning domains. Here, we use rotational kinematics as the base domain and linear kinematics as the learning domain. Once again, we are interested in comparing the learning rates between our baseline system and our model of domain transfer via cross-domain analogy.
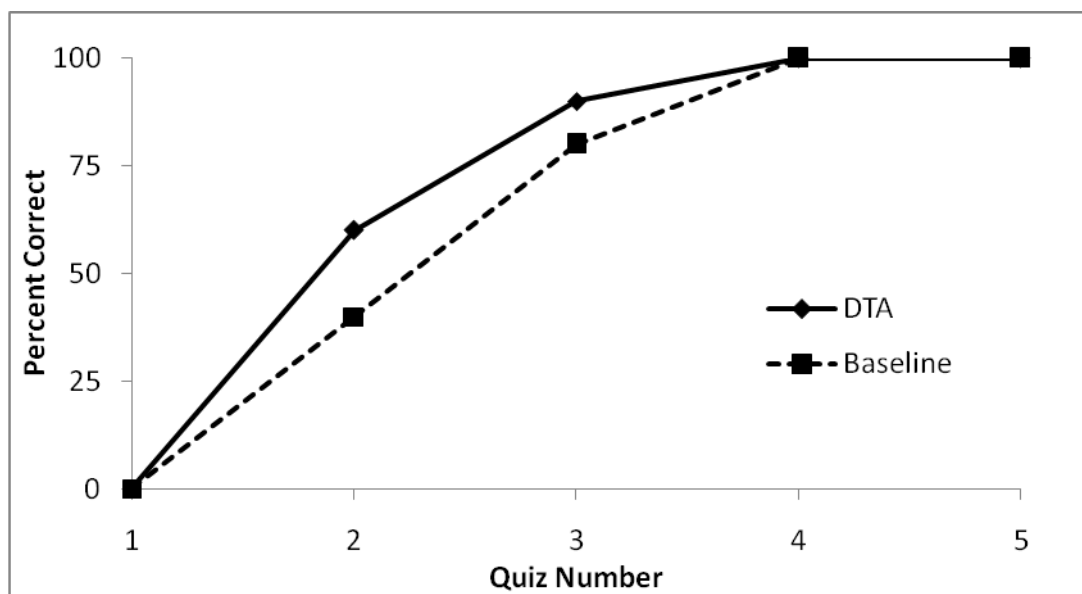
Figure 9: Linear mechanics learning curves

Once again, the DTA system outperformed the baseline system. Figure 9 graphs the learning curves of the two conditions. After the first problem, the analogy system got the next problem correct 60% of the time, compared with the 40% performance of the baseline. After seeing two worked solutions, the analogy system scored 90% on the third problem where the baseline system scored 80%. On problems 4 and 5, both systems performed at a ceiling of 100%. The baseline condition was able to achieve a ceiling of 100% as every equation required was used by at least two problems. In the analogy condition, DTA was unable to transfer the correct domain theory for two linear kinematics problems. This led to 180 transfer attempts, of which 120 (66%) were successful. Of the successful attempts, none of them required using an additional retrieval.

Discussion

In both experiments, domain transfer via analogy outperformed the spoon-fed baseline system. DTA learned faster and achieved a ceiling that was the same or higher than the baseline. An analysis of the few linear kinematics transfer failures indicates that certain sub-event structures and time intervals increase the difficulty in generating an appropriate domain mapping. Linear kinematics problems 'b' and 'd' both contained such structures. One requirement for successful transfer in these scenarios is that an `objectRotation` statement in the rotational kinematics worked solution must correspond with a `primaryObjectMoving` statement in the linear kinematics worked solution. In order for this to occur, the entities which make up these expressions must already be in alignment. Given the structure and number of the events and time intervals in these two problems, the events listed in these statements may differ from the events and time intervals referenced in the quantities and equations. Therefore, one aspect of our future work is to incorporate *rerepresentation* strategies (Yan *et al.* 2003) to bring these worked solutions into better alignment with the analogous rotational kinematics worked solutions. The added complexity of the linear kinematics problems also slowed the baseline learning system.

Another interesting result of these experiments is the difference in the utility of retrieving an additional worked solution if the first one fails. In experiment 1, this part of the algorithm was critical to the analogy system's perfect transfer. On the other hand, in experiment 2, the additional retrievals never produced an adequate domain mapping. This supports our intuition that the decision to retrieve additional analogs should be controlled by the system based upon its goals.

Related Work

There are three main branches of related work: AI systems of analogy and case based reasoning, cognitive science models of cross-domain analogy, and integrated reasoning systems working towards human-level capabilities. The majority of analogical and case based systems focus on using previous experiences to guide future problem-solving (Koloder 1991). This work has occurred in a variety of domains including transportation (Veloso & Carbonell 1993), physics problem solving (van Lehn 1998), inductive theorem proving (Melis & Whittles 1999) and thermodynamics problem-solving (Ouyang & Forbus 2006). These systems use examples to provide search control knowledge in order to solve future problems faster. DTA differs by focusing on learning schema-like domain knowledge, in this case encapsulated histories.

On the other hand, cross-domain analogy research has focused on this problem of learning abstract knowledge. The closest project to our approach is Falkenhainer's (1988) PHINEAS. PHINEAS used comparisons of (simulated) behaviors to create an initial cross-domain mapping. The mapping results in inferences that were used to create a partial theory for the new domain. Our model differs from PHINEAS 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 require very different verification work, and (3) We are using a more psychologically plausible retrieval mechanism, MAC/FAC. Holyoak and Thagard's (1989) PI model used a pragmatic theory of analogy to model solve variations of the radiation problems through schema induction. PI placed an emphasis on analogy during problem solving. On the other hand, our model makes analogies between the domain theories themselves. This allows DTA to transfer domain knowledge not explicitly referenced in the worked solution instances used in creating the domain mapping.

A number of recent projects working towards human-level AI have built architectures emphasizing the importance of integrating analogy with other forms of reasoning. Kühnberger *et al.*'s I-Cog (2008) explores the trade-offs between analogy and two other reasoning modules, all of which operate over noisy data, using the Heuristic-Driven Theory Projection (Gust *et al.* 2006) model of analogy. Schwering *et al*. (2008) identifies the importance of combining analogy with deductive and inductive techniques for achieving human level reasoning. Kokinov's AMBR (2003) explores the role of analogy in various aspects of memory, such as retrieval and distortion effects. We agree that analogy is integral to the robustness of human reasoning. Our cognitive architecture, Companions cognitive systems (Forbus & Hinrichs 2004), emphasizes that analogy is a common operation as opposed to an exotic event undertaken rarely. For example, in DTA, each attempt at transfer requires an analogy between worked solutions as well as between domain theories themselves. We plan to integrate DTA into our Companions-based learning system (2007), which utilizes within-domain analogies to formulate the models necessary to solve AP Physics problems. By focusing on human-level tasks, we believe we will learn about the constraints on the analogical processes themselves in addition to learning how to utilize them in building AI systems (Forbus 2001).

Conclusions and Future Work

We have shown that a domain theory for solving physics problems can be learned via cross-domain analogies, using our DTA method for cross-domain transfer. DTA is very general; it should work with any domain theory where reasoning involves schema-like domain theories and where analogous worked solutions are available. Furthermore, our experiments demonstrate that such analogical learning can be very efficient. In fact, when the two domains are sufficiently similar, DTA can outperform a baseline system that is incrementally given the correct domain theory. The process of constructing domain mappings by exploiting structural 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 our model with learning encapsulated histories, so we want to extend it to handle other types of domain knowledge, such as model fragments and modeling knowledge. Currently, we are developing a method for learning modeling decisions via generalization and incorporating it into our domain transfer system. Second, while DTA has an explicit verification step, this does not guarantee that all the learned knowledge is correct. 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). While linear and rotational mechanics are quite similar, the analogy between mechanics and electricity, two superficially different domains, is quite systematic. This will explore the extent that similarities in worked solution structure can learn an adequate domain mapping. Finally, an analogy between mechanics and heat flow will stress that only certain aspects of the domain should transfer, as there are important non-analogous aspects between these domains.[2]

Acknowledgments

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).

---

[2] While the spring constants in mechanics are analogous to the inductance constants in electricity, there is no analog for inductance in thermal dynamics.

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. & de Kleer, J. 1993. *Building Problem Solvers*. MIT Press.

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

Forbus, K. & Hinrichs, T. 2004. 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.

Gick, M. & Holyoak, K. 1980. Analogical problem solving. *Cognitive Psychology*. 12.

Gust, H., Kühnberger, K.-U., and Schmid, U. 2006. Metaphors and Heuristic-Driven Theory Projection (HDTP). *Theoretical Computer Science* 354(1):98-117.

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.

Kühnberger, K.-U., Geibel, P., Gust, H., Krumnack, U., Ovchinnikova, E., Schwering, A., and Wandmacher, T. 2008. Learning from Inconsistencies in an Integrated Cognitive Architecture, *1st Conference on Artificial General Intelligence (AGI08)*, Memphis, TN. IOS 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.

Kokinov, B. 2003. The Mechanisms of Episode Construction and Blending in DUAL and AMBR: Interaction Between Memory and Analogy. In: Kokinov, B., Hirst, W. (ed.) *Constructive Memory*. Sofia: NBU Press.

Matuszek, C., Cabral, J., Witbrock, M., and DeOliveria, J. 2006 An Introduction to the Syntax and Content of Cyc. In *Proceedings of the 2006 AAAI Spring Symposium on Formalizing and Compiling Background Knowledge and Its Applications to the Knowledge Representation and Question Answering*, Stanford, CA, March 2006.

Melis, E., and Whittle. J. 1999. Analogy in inductive theorem proving. *Journal of Automated Reasoning*, 1999.

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

Ouyang, 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.

Schwering, A.; Krumnack, U.; Kühnberger, K.-U., Gust, H. 2008. Analogy as Integrating Framework for Human-Level Reasoning, *1st Conference on Artificial General Intelligence (AGI08)*, Memphis, TN. IOS 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.

Yan, J., Forbus, K., and Gentner, D. 2003. A Theory of Rerepresentation in Analogical Matching. *Proceedings of the Twenty-fifth Annual Meeting of the Cognitive Science Society*.