Running head: DOMAIN TRANSFER VIA CROSS-DOMAIN ANALOGY

Domain Transfer via Cross-Domain Analogy

Matthew Klenk and Ken Forbus

Qualitative Reasoning Group, Northwestern University

2133 Sheridan Rd, Evanston, IL 60201 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 the *domain transfer via analogy* (DTA) 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. These learning rates outperform those from a version of the system that is incrementally given the correct domain theory.

Domain Transfer via Cross-Domain Analogy

1 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 affecting future learning 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 physics 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 capture this kind of learning.

One method students use to learn physics is by studying worked solutions. While these step-by-step explanations are not deductive proofs, they do provide students with valuable feedback. An insight of this work is that the structure of these explanations provides aspects of

underlying structure of the domain. Our hypothesis is that we can use this structure to learn domain mappings between physics domains. Our model, *domain transfer via analogy* (DTA), uses worked solutions and a known base domain as a starting point for learning a novel domain theory through multiple cross-domain analogies. DTA learns a domain mapping through an analogy between worked solutions from the known and novel domains. Our model uses this domain mapping to initialize the new domain theory and constrain an analogy between the base and novel 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 and illustrate it 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 with the correct domain theory. We close with a discussion of related work and implications for future work.

2 Structure-mapping and Analogy

Before discussing the DTA method, it is necessary to describe the structure-mapping processes of analogy and similarity-based retrieval which are used by our model. 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 (e.g., the minimal ascension method described below). 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.* 1995) is a computational model of similarity-based retrieval. The inputs are a set of predicate calculus facts, the *probe* case, 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. In this work, we use *minimal ascension* (Falkenhainer 1988) to match 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. This allows a looser form of semantic similarity, in addition to identicality, the strongest form, to guide the matching process.

3 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.

3.1 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. The problem representations are intended to be direct translations into predicate calculus from natural language problem statements, without any abstraction or reasoning. 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. The first two facts define the entities in the problem, a transportation with land vehicle event, `Acc-2-6`, and an automobile, `Car-2-6`. The next 4 facts describe the motion of `Car-2-6` during `Acc-2-6`. The last 3 facts describe the question of the problem and how these facts are grouped together in a case.

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. Below is an English rendering of the worked solution for this example problem:

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 ($t = 5.8s$)

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 0m/s. Later, we show how DTA uses analogies between worked solutions from different domains to learn a domain mapping.

3.2 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 in Figure 2 which involves events as well as their durations. During problem solving, our system instantiates applicable encapsulated histories to determine which equations are available.

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. This knowledge is necessary for our physics problem-solver to successfully answer questions.

In addition to understanding physics phenomena and equations, 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. We evaluate the domain theories learned through DTA on new 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).

4 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. DTA is invoked after failure to solve a problem in the new domain. Our model is provided the worked solution for that problem. Therefore, the inputs to the DTA algorithm are a known base domain and a worked solution from the new domain. The known base domain consists of a set of worked solutions and a domain theory, which is represented in this work as a set of encapsulated histories. DTA uses this worked solution to create conjectures about knowledge in the new domain, via the algorithm outlined in Figure 5. Recall that Cognitive Science research has identified the importance of domain mappings, i.e., the mapping of types and relations between two domains, for successful cross-domain analogies. Therefore the first step in our algorithm is to learn the domain mapping. Next DTA initializes the target domain theory, extends this new domain theory using this mapping, and verifies the learned knowledge. We describe each of these steps in detail below.

4.1 Step 1: Learn the Domain Mapping

Because different domains are represented with different predicates and conceptual types, a domain mapping is essential to successful cross-domain analogies. DTA learns the domain mapping through an analogical mapping between worked solutions from the two domains. Using the worked solution for the failed problem as a probe, MAC/FAC selects an analogous worked solution from a case library containing worked solutions from the known domain. Next, SME creates 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 sorts the mappings by their structural evaluation scores. Beginning with the best mapping, we add all of the correspondences in which the base element is a non-numeric entity (e.g., types, quantities, and relations) mentioned in the base domain theory. Then, our model iterates through the rest of the mappings adding 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.

4.2 Step 2: Initialize the Target Domain Theory

Recall that when the system attempts the first problem in a new domain, its theory for that domain is empty. That is, it does not have any encapsulated histories to understand the phenomena of the new domain. 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. The resulting encapsulated history is then added to the target domain theory.

4.3 Step 3: Extend the Target Domain Theory

After initializing the target domain theory, DTA extends it through a second cross-domain analogy. This time the analogy is between the base and target domain theories themselves. The domain theories consist of the facts representing the encapsulated histories. Our model constrains this match with the domain mapping to ensure the consistency of the target domain theory. One result of SME is a set of 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 encapsulated histories to the target domain theory, it is necessary to resolve the *skolem entities* mentioned in the candidate inferences. SME creates skolem entities for entities appearing in the base which have no correspondence in the mapping. If the skolem represents a number from the base, then that number is used in the target. Otherwise, our model creates a new entity for each skolem to be assumed in the target domain theory. After this process, the candidate inferences representing new encapsulated histories are stored into the target domain theory.

4.4 Step 4: 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 analogue worked solution from the case library so that the next-best worked solution is retrieved.

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 employing DTA. Next, we describe an example of our model's operation.

4.5 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 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.

In order to create a domain mapping between linear and rotational kinematics, DTA creates an analogy between a linear kinematics worked solution and the provided rotational kinematics worked solution. Using the provided worked solution as a probe, the model uses MAC/FAC to retrieve an analogue from its case library of linear kinematics worked solutions. 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 $2m/s^2$?" 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 placing

`primaryObjectMoving` in correspondence with `objectRotating` 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.

After learning the domain mapping, the next step is 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 found. In order to transfer this encapsulated history to the rotational kinematics domain theory, all of its participant types and quantities must participate in the domain mapping. This encapsulated history contains two participant types, `PointMass` and `ConstantLinear-AccelerationEvent` and four quantities, `Acceleration`, `Speed`, `Time-Quantity` and `DistanceTravelled`. All of these appear in the domain mapping allowing DTA 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, DTA extends the new domain theory with a cross-domain 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. DTA uses these candidate inferences to infer rotational kinematics encapsulated histories. 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. 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. During this step, DTA assumes three new encapsulated histories into the rotational kinematics 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 assume that the learned knowledge is correct. If the system gets the problem wrong, it takes two steps. First, it erases the domain mapping and the inferred encapsulated histories in the rotational mechanics domain theory. Second, it repeats the entire process one more time, with the next-best retrieval from the case library. Abandoning, rather than debugging, a mapping may be unrealistic. We plan to explore diagnosis and repair of faults in our domain theories and domain mappings in future work, as explained in Section 7. In this case, the system used the learned encapsulated histories to get the problem correct.

Recall at the beginning the system had zero encapsulated histories in its rotational kinematics domain theory. While the rotational kinematics worked solution contained an example of one encapsulated history, after employing DTA, the system learned four rotational kinematics encapsulated histories. Now, these are available for solving future problem-solving episodes. In the next section, we evaluate DTA by giving our system a sequence of problems from a new domain and measuring its ability to answer questions correctly.

5 Evaluation

To examine how well DTA works, we need a baseline. Our baseline *spoon-fed* system consists of the exact same problem-solver. 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 to evaluate learning of rotational kinematics by an analogy with linear kinematics, and the other to evaluate learning of linear kinematics based upon rotational kinematics. 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.

The problems for both domains are listed in Figure 7. For these problems, we created representations and worked solutions in the manner described in section 3. For linear kinematics, the representations for the problems and worked solutions averaged 15.4 and 52 facts respectively. These representations included 38 different types and 52 unique relations. For rotational kinematics, the problem and worked solution representations averaged 9.8 and 46.2 facts respectively. These representations included 21 types and 33 relations. 9 types and 26 relations appear in problems and worked solutions from both domains.

This evaluation seeks to answer the following questions. First, can DTA transfer the encapsulated histories to solve problems in new domains? Second, can DTA learn better than a baseline system which is incrementally given the correct domain theory? Third, how important is the verification of the learned domain theory and the ability retrieval additional analogues?

5.1 Experiment 1

In this experiment, we use linear kinematics as the base domain and rotational kinematics as the target domain. To address our research questions, we generated learning curves by aggregating performance across the 120 trials representing every possible ordering of the five rotational kinematics problems. Because rotational kinematics is the target domain, both systems begin the trial without any rotational kinematics encapsulated histories. Therefore, we expect neither system to solve the first problem on any of the trials. During each trial, when the DTA system fails to solve a problem, the DTA method is invoked with the worked solution to that problem. After each problem in the baseline condition, the system is given the necessary encapsulated histories to solve the problem. These encapsulated histories, either learned by cross-domain analogy in the DTA system or provided to the baseline system, allow the system to answer future rotational kinematics questions. At the end of each, the system's knowledge was reset.

Figure 8 compares the rotational kinematics learning rates for the DTA and baseline conditions. The DTA system 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 DTA system performed perfectly, there were only 120 transfer attempts, all of which succeeded. Of these, 72 attempts (60%) required retrieving a second worked solution to generate a successful domain mapping. This highlights the importance of verifying the knowledge learned from the

cross-domain analogy. The performance in the baseline condition was markedly worse than DTA. 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. This allows DTA to infer equations not mentioned explicitly in the worked solution from the target domain.

5.2 Experiment 2

This experiment followed the same form as the previous experiment but with the opposite base and target domains. Here, we use rotational kinematics as the base domain and linear kinematics as the target 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.

Once again, the DTA system outperformed the baseline system. Figure 9 graphs the learning curves of the two conditions. After the first problem, the DTA 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, 120 after the first problem of the trial, 48 after failures on the second problem and 12 after the third problem. Out of the 180 attempts, 120 (66%) were successful. Of the successful attempts, none of them required using an additional retrieval.

5.3 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 increasing the complexity of sub-event structures and time intervals increases 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 `objectRotating` 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 of linear mechanics problems 'b' and 'd', the events listed in the step uses 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 phenomena illustrated by 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 analogues should be controlled by the system based upon its goals.

6 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 (Kolodner 1993). This work has occurred in a variety of domains including transportation (Veloso & Carbonell 1993), 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. The most striking difference between DTA and these systems is that we focus on learning new domain concepts as opposed to search control knowledge. That is, without the analogical transfer, our system would not be able to answer any questions in the target domain, and each analogical transfer allows our system to solve problems it otherwise would not have been able to solve.

Focusing on learning from examples, the Cascade system models learner behavior in the Newtonian physics problem-solving (VanLehn 1999). Cascade uses analogy in two ways. First, as in the above systems, analogy is used to learn search control knowledge. Second, while overly general rules are the primary method for acquiring new physics knowledge, Cascade is able to use within-domain analogy during example studying to form new rules as a last resort. We differ by focusing on learning by analogy and using multiple cross-domain analogies to construct a new domain theory. Cascade has been used to model two psychological effects during physics learning: the self-explanation effect (VanLehn *et al.* 1992) and learning from faded examples (Jones & Fleischman 2001). Instead of focusing on matching individual human data, we seek to understand how knowledge of related domains can accelerate learning in new

ones. Building upon existing cognitive models of analogical processing, DTA represents an important step in this direction.

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 solving a variation of the radiation problem through schema induction. PI only used analogy during problem-solving, and its retrieval model was never extensively tested. On the other hand, our model makes analogies between both example problems as well as analogies between domains themselves. The domain analogies allow DTA to transfer domain knowledge not explicitly referenced in the particular worked solutions used in creating the domain mapping. Moreover, DTA tests its learned knowledge, and uses it to solve new problems from the target domain, whereas PI did neither.

A number of recent projects working towards human-level AI have built architectures emphasizing the importance of integrating analogy with other forms of reasoning. ICARUS employs a representational mapping mechanism to transfer plan goal decompositions between different 2d grid games (Shapiro *et al.* 2008). 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 *et al.* 2008), 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 (Klenk & Forbus 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).

7 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 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 (Klenk *et al.* 2008). 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. Integrating both generalization techniques for within-domain learning and diagnostic techniques for cross-domain learning requires additional commitments as to when and how these processes are invoked by a larger system. As indicated above, we plan to use the Companions cognitive architecture to extend DTA, putting the learning process under control of plans in the Executive. This will place decisions like whether to debug or abandon a problematic cross-domain mapping under system control, and thereby extend the range of phenomena the model can tackle.

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]

8 References

Collins, A. & Gentner, D. (1987). How people construct mental models. In D. Holland & N. Quinn (Eds.), *Cultural models in language and thought*. (pp. 243-265). 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), 173-229.

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

Falkenhainer, B., Forbus, K., and Gentner, D. (1989). The Structure-Mapping Engine. *Artificial Intelligence*, *41*, 1-63.

Forbus, K. (1984). Qualitative process theory. *Artificial Intelligence*, *24*, 85-168.

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

Forbus, K. & de Kleer, J. (1993). *Building Problem Solvers*. Cambridge, MA: MIT Press.

Forbus, K., Gentner, D., & Law, K. (1995). MAC/FAC: A model of similarity-based retrieval. *Cognitive Science*, *19*, 141-205.

Forbus, K. & Hinrichs, T. (2004). Companion cognitive systems: a step toward Human-Level AI. In *AAAI Fall Symposium on Achieving Human-level Intelligence through Integrated Systems and Research*, Washington, DC.

Forbus, K. & Oblinger, D. (1990). Making SME greedy and pragmatic. In *Proceedings of the 12th annual Conference of the Cognitive Science Society*, Boston, MA, 61-68.

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

7(2), 155-170.

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). Cambridge, MA: 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*, (pp. 99-129), Hillsdale, NJ: Lawrence Erlbaum Associates.

Giancoli, D. (1991). *Physics: Principles with Applications*. (3rd ed.). Englewood Cliffs, NJ: Prentice Hall.

Gick, M. & Holyoak, K. (1980). Analogical problem solving. *Cognitive Psychology, 12*, 306-356.

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*. (243-267). New York: Cambridge University Press.

Jones, R. M. & Fleischman, E. S. (2001). Cascade explains and informs the utility of fading examples to problems. In *Proceedings of the Twenty-Third Annual Conference of the Cognitive Science Society,* Hillsdale, NJ, 459-464.

Klenk, M., Friedman, S., and Forbus, K. (2008). Learning Modeling Abstractions via

Generalization. In *The Proceedings of the 22nd International Workshop on Qualitative Reasoning*. Boulder, CO.

Klenk, M. & Forbus, K. (2007). Measuring the level of transfer learning by an AP Physics problem-solver. In *The Proceedings of 22nd Conference of Artificial Intelligence (AAAI-07)*. Vancouver, CA, 446-452.

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.

Kolodner, J.L., (1993). *Case-Based Reasoning*, San Mateo, CA: Morgan Kaufmann Publishers.

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. In *The Proceedings of the 1st Conference on Artificial General Intelligence (AGI-08)*, Memphis, TN, 212-223.

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.

Melis, E., and Whittle. J. (1999). Analogy in inductive theorem proving. *Journal of Automated Reasoning, 22*(2), 117-147.

Olson, H. (1966). *Solutions of Engineering Problems by Dynamical Analogies*, New York, NY: D. Van Nostrand.

Ouyang, T. & Forbus, K. (2006). Strategy variations in analogical problem solving. In *The Proceedings of 21st National Conference on Artificial Intelligence (AAAI-06)*. Boston,

MA, 446-452.

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*. (pp. 498-531) New York: Cambridge University Press.

Schwering, A.; Krumnack, U.; Kühnberger, K.-U., Gust, H. (2008). Analogy as Integrating Framework for Human-Level Reasoning, In *The Proceedings of the 1st Conference on Artificial General Intelligence (AGI-08)*, Memphis, TN. IOS Press. 419-423.

Shapiro, D., Konik, T., and O'Rorke, P. (2008). Achieving far transfer in an integrated cognitive architecture. In *The Proceedings of the 23rd National Conference on Artificial Intelligence (AAAI-08),* Chicago, IL. 1325-1331

VanLehn, K. (1999). Rule-learning events in the acquisition of a complex skill: An evaluation of Cascade. *Journal of the Learning Sciences*, 8(1), 71-125.

VanLehn, K., Jones, R. M., & Chi, M. T. H. (1992). A model of the self-explanation effect. *Journal of the Learning Sciences, 2*, 1–59.

Veloso, M. & Carbonell, J. (1993). Derivational analogy in PRODIGY: Automating case acquisition, storage, and utilization. *Machine Learning*, *10*(3). 249-278.

Yan, J., Forbus, K., and Gentner, D. (2003). A Theory of Rerepresentation in Analogical Matching. In *The Proceedings of the Twenty-fifth Annual Meeting of the Cognitive Science Society*. Boston, MA, 1265-1270.

Author Note

Matthew Klenk and Ken Forbus, Department of Electrical Engineering and Computer Science, Northwestern University.

This research was supported by the Cognitive Science Program of the Office of Naval Research. The authors would like to thank the participants of the AnICA 2007 workshop for the helpful feedback on the ideas presented here. Also, we thank Thomas Hinrichs, Kate Lockwood, and Scott Friedman for their comments on this work and help revising this document.

Correspondence concerning this article should be addressed to Matthew Klenk, 2133 Sheridan Rd., Evanston, IL 60201. Email: m-klenk@northwestern.edu

Footnotes

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

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

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

Figure Captions

Figure 1. Minimal ascension aligns `primaryObjectMoving` to `objectRotating`

Figure 2. Example problem 2-6 representation

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

Figure 4. Definition for the velocity by time encapsulated history

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

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

Figure 7. Evaluation materials

Figure 8. Rotational mechanics learning curves

Figure 9. Linear mechanics learning curves

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

```
(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

```
(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

```
(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

Domain Transfer via Analogy (DTA) method:

Given: Case library of base domain worked solutions, worked solution from target domain, base domain encapsulated histories

1. Learn the domain mapping
   - Retrieve analogue worked solution from case library using MAC/FAC, with target worked solution as the probe
   - Use SME to create an analogical mappings between the analogue worked solution and the target worked solution
   - Create domain mapping by selecting correspondences from the mappings in which the base element appears in the base encapsulated histories
2. Initialize target domain theory using the domain mapping
   - For each base encapsulated history mentioned in the domain mapping, attempt to create a corresponding encapsulated history in the target domain
3. Extend target domain theory
   - Use SME to create a match between the base and the target encapsulated histories constrained by the domain mapping
   - Transfer domain theory contents using the candidate inferences
4. Verify learned domain theory by attempting the failed problem again
   - If failure, go once more to step 1. Otherwise, accept new target domain knowledge as correct

Figure 5

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

Figure 6

Linear Kinematics

a) How long does it take a car to travel 30m if it accelerates from rest at a rate of $2m/s^2$?

b) We consider the stopping distances from a car, which are important for traffic safety 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 28m/s, the acceleration is $-6.0m/s^2$, and a reaction time of .5s. What is the stopping distance?

c) A baseball pitcher throws a fastball with a speed of 44m/s. It has been observed that pitchers accelerate the ball through a distance of 3.5m. What is the average acceleration during the throwing motion?

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 80m/s for takeoff. The runway is 1500m long. What is the constant acceleration required?

Rotational Kinematics

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 7

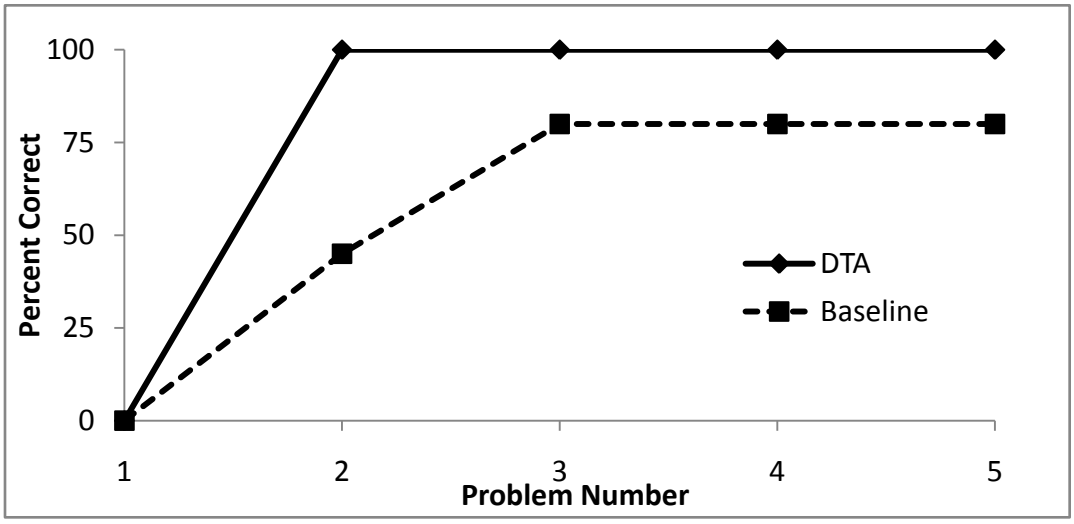
100
75
50
25
0
Percent Correct
1
2
3
4
5
Problem Number
DTA
Baseline

Figure 8

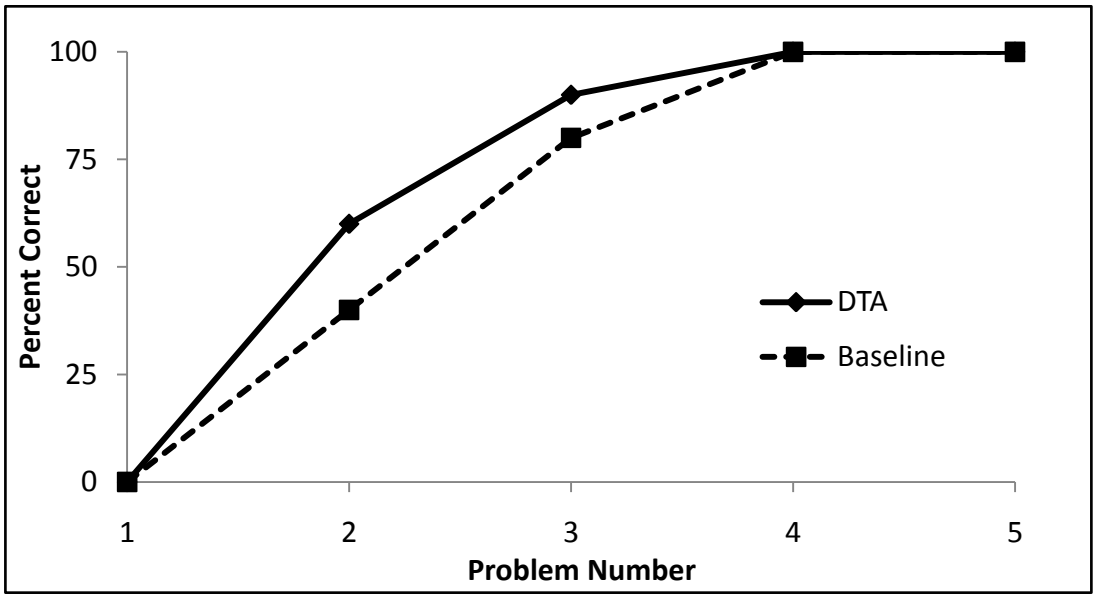
100
75
50
25
0
Percent Correct
1
2
3
4
5
Problem Number
DTA
Baseline

Figure 9