Learning Modeling Abstractions via Generalization

Matthew Klenk, Scott E. Friedman, Kenneth D. Forbus

Qualitative Reasoning Group, Northwestern University
21331 Sheridan Rd, Evanston, IL 60208 USA
{m-klenk, friedman, forbus}@northwestern.edu

Abstract

In domains with everyday scenarios, an important aspect of model formulation concerns moving from broad descriptions to the technical abstractions necessary for effective problem-solving. We present a method for learning how to make abstraction decisions from experience via analogical generalization. Specifically, we generalize abstraction decisions from worked examples, abstracting away irrelevant information. When faced with a new situation, our method compares the entities in the situation with the generalizations, and makes its decision by using the best match. We argue that the similarity score from the comparison is an effective heuristic for judging the quality of the modeling decision. Using textbook physics problems, we show that our method can make accurate abstraction decisions, and that these decisions improve as the system gains experience.

Introduction

One of the important contributions of qualitative reasoning has been formalizing the process of model formulation (cf. Falkenhainer & Forbus 1991; Nayak 1994; Rickel & Porter 1994). Most model formulation work has focused on ascertaining what levels of detail and which perspectives should be used in a model, given a particular task. In general, model formulation research has ignored the problem of computing structural descriptions, *i.e.* moving from the broad set of concepts used in everyday life to a concise, technical vocabulary of abstractions that can be used effectively for problem-solving. We use the term *participant abstraction* to refer to the type of a participant in a domain theory, and the term *scenario entity* to refer to an entity within the everyday domain scenario. This work addresses how decisions about participant abstractions can be learned. Specifically, we use analogical techniques from structure-mapping theory (Gentner 1983) to decide how to represent everyday entities in a scenario model.

The typical method for making these participant abstraction decisions is as follows. Given a scenario and a domain theory, one can use the type of each scenario entity in an ontology to determine its appropriate participant abstraction in the domain theory. For example, in the ResearchCyc[1] ontology, the collection `Coin` is a specialization of the collection `PartiallyTangible`. Consequently, we could write a rule stating that a `PartiallyTangible` should be considered a `PointMass` in a model. This rule-based approach is problematic for several reasons. First, these rules would contain false positives (e.g. a `Lake`, which is a `PartiallyTangible`, should not be considered a point mass in most situations). Second, participant abstraction decisions are very contextual. While a coin falling off a building could be considered a `PointMass`, the same coin spinning on a table, in a rotational mechanics problem, should be a viewed as an object with extent. Accounting for this necessary contextual information greatly increases the complexity of such rules. As noted by Falkenhainer and Forbus (1991), modeling rules are very domain specific; that is, for each new domain a knowledge engineer will have to construct a new set of rules.

We propose an alternative method that learns from examples the necessary connections between everyday scenario entities and participant abstractions to construct scenario models. Our method uses psychological simulations of analogical processing to learn these participant abstraction decisions.

This paper uses physics problem solving to demonstrate our method, but we believe it is applicable across a wide range of domains. We begin by summarizing the analogical processing components we use and our representations of the physics domain. Next, we describe how participant abstraction decisions can be learned from examples, using generalization. We present experimental results demonstrating the effectiveness of our method. Finally, we close with a discussion of related work and future work.

Background

Our approach to learning participant abstraction decisions utilizes the SEQL generalization model (Kuehne *et al.* 2000). SEQL constructs generalizations incrementally via analogical comparison using SME, the Structure-Mapping Engine (Falkenhainer *et al.* 1989). For this work, the

[1] http://research.cyc.com/ - a large scale effort to formalize commonsense knowledge

generalizations are formed over example participant abstraction decisions from physics problems. We begin with an overview of the analogical processes utilized in our method, and then we describe the participant abstractions of our domain theory and physics representations.

Analogical Processes: SME and SEQL

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. For maximality, structure-mapping uses the principle of *systematicity*: mappings that are highly interconnected and contain deep chains of higher order relations are preferred.

SME simulates analogical matching. It takes as input two structured representations (base and target). It produces one or more *mappings* that describe how the two representations can be aligned. A mapping includes *correspondences* that link items (entities and relations) in one representation with items of the other, a *structural evaluation score* which reflects the quality of the entire mapping, and a set of *candidate inferences* that are conjectures about the target created by projecting partially mapped base expressions.

SEQL simulates analogical generalization. It maintains a list of generalizations and exemplars, which are structured representations, called a *generalization context*. It takes as input a sequence of new examples. Given a new example, SME is used to compare it to the existing generalizations. When the structural evaluation score is above the *assimilation threshold*, the example is assimilated into that generalization by keeping the common overlapping structure. If the example is not assimilated into an existing generalization, it is compared against the exemplars. Again, if it is sufficiently similar to another exemplar, a new generalization is created from the common structure. Otherwise, the new example is added to the list of exemplars.

Physics problem-solving and QP theory

de Kleer's (1977) pioneering work emphasized the importance of modeling in solving physics problems. Given a domain theory and a physics problem, a problem-solver must make a number of modeling decisions to arrive at the correct solution. Consider a problem where a ball is dropped off the top of a building. The ball should be considered a point mass, and the falling event should be considered a constant linear acceleration event. These are examples of participant abstraction decisions and are the focus of this paper. Solving this problem requires additional modeling decisions, including assuming that the event occurs on Earth. As noted in related work, analogy may be useful for learning how to make these types of decisions as well. In this paper, these other modeling decisions are made via hand-coded rules, so we can focus exclusively on participant abstraction decisions here. Determining which abstraction to apply, given problem scenarios whose entities can range over tens of thousands of possible categories, is quite challenging.

Our physics domain theories consist of *encapsulated histories* (Forbus 1984) that represent physics equations. Encapsulated histories, unlike model fragments, permit constraints to be placed on the duration of events and time intervals. The encapsulated histories for our physics domain theory include participant abstractions such as `PointMass` and `ConstantTranslationAccelerationEvent`. These abstractions are not used within the scenario descriptions of physics problems; rather, the scenario entities are encoded as real-world objects such as `Automobile` and `Driving`. Moving from the real-world entities to the technical language for problem-solving is one kind of *simplifying assumption* (Falkenhainer & Forbus 1991).

```
(def-encapsulated-history
  VelocityByTime-1DConstantAcceleration
  :participants
 ((theObject :type PointMass)
  (theEvent  :type
             ConstantTranslationAccelerationEvent))
 :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 1: Encapsulated history definition

Figure 1 shows the definition for the encapsulated history representing the equation $v_f = v_i + at$, velocity as a function of time. The two participants, `theObject` and `theEvent`, must satisfy their type constraints, `PointMass` and `ConstantTranslationAccelerationEvent`, respectively. Furthermore, the conditions of the encapsulated history must be satisfied in order to instantiate it and conclude its consequences. In this example, `theObject` must be the object moving in `theEvent` for the encapsulated history to be instantiated.

The method we describe in this paper learns how everyday entities in problems should be modeled in terms of the abstractions used in the domain theory.

Representing Physics Problems and Examples

The representations used in this work are in CycL, the predicate calculus language of the ResearchCyc knowledge base (Matuszek *et al.* 2006). We use a subset of the ResearchCyc KB, consisting of 33,000+ concepts, and

13,000+ relations, plus our own extensions for QP theory (Forbus 1984) and problem-solving strategies. Consequently, objects, relations, and events that appear in physics problems such as "rotor," "car," and "driving" are predefined in the ontology. This reduces the degree of tailorability in our experiments.

All the problems used in this work were taken from a common physics textbook (Giancoli 1991). We represent the problems and examples as cases, consisting of predicate calculus facts. Consider the following physics problem:

> Suppose a ball is dropped from a 70m tower. How far will it have fallen after 3 seconds?
> *Example problem 2-9, p. 30.*

This problem is represented in our system as a case of 19 facts, a subset of which is shown in Figure 2. There are five entities in the problem: the top of the tower, the tower, the ball, the dropping event, and the 3-second interval. The facts in Figure 2 pertain to the ball's motion during the dropping event, the description of the time interval, and the query of the problem.

```
...
(objectStationary (StartFn Drop-2-10) Ball-2-10)
(primaryObjectMoving Drop-2-10 Ball-2-10)
(directionOfTranslation Fall-2-10 Down-Directly)
(objectTopSide Tower-2-10 Top-2-10)
(fromLocation Drop-2-10 Top-2-10)
(temporallyCooriginating Drop-2-10 Interval-2-10)
(valueOf (Time-Quantity Interval-2-10)
         (SecondsDuration 3))
(querySentence Gia-Query-2-10
  (valueOf
     (DistanceTravelled Ball-2-10 Interval-2-10)
     Distance-2-10))
```

Figure 2: Part of example problem 2-9 representation

One common learning method physics students use is to solve problem sets and compare their answers to worked solutions. This technique motivates the feedback we provide our system. Worked solutions are neither deductive proofs nor problem-solving traces produced by our solver. The worked solution for this example problem consists of five steps:

1. Categorize the problem as a constant acceleration linear mechanics problem
2. Assume that the acceleration of the ball ($a = 10\ m/s^2$)
3. Instantiate the distance by velocity time equation ($d = v_it + .5at^2$)
4. Because the ball is stationary at the start of the drop infer that its velocity is zero ($v_i = 0\ m/s$)
5. Solve the equation for d ($d = 45\ m$)

The entire worked solution for this problem consists of 38 facts. The third step is most relevant to the goals of this paper – the instantiation of the distance by the velocity time equation. This step depends upon two abstraction decisions, one for the ball and one for the dropping event, as illustrated in Figure 3. Next we describe how we build generalizations from these example decisions and apply the learned knowledge to new problems.

```
(StepUses Gia-2-10-WS-Step-3
   (abstractionForObject Ball-2-10 PointMass))
(StepUses Gia-2-10-WS-Step-3
   (abstractionForObject Drop-2-10
      ConstantTranslationAccelerationEvent))
```

Figure 3: Worked solutions indicate appropriate participant abstractions for problem entities

Learning Participant Abstraction Decisions

The primary contribution of this work is our method for learning how to make decisions about participant abstractions for problem entities described in everyday terms. Our method is best understood in two stages: generalization and execution. First, we generalize examples of participant abstraction decisions. Then, when faced with a problem, our method uses analogies between the entities in the problem and the generalizations to make participant abstraction decisions. These decisions allow our solver to instantiate the necessary encapsulated histories to solve the problem.

Generalization of Participant Abstractions

We create generalizations at the granularity of the participant abstraction. As such, we contextualize the generalizations such that all examples of a given participant abstraction are considered together. We achieve this with generalization contexts. Each context has an *entry pattern* that exemplars must satisfy to be generalized within. The entry patterns used here reflect the various participant abstractions. Figure 4 depicts the four generalization contexts after generalizing decisions from eight worked solutions.

Our system populates the generalization contexts with exemplars generated from worked solutions. Exemplars are created for each participant abstraction within the worked solutions and generalized within the appropriate contexts. For example, in the worked solution from Figure 3, there are two statements indicating participant abstractions. The statement `(abstractionForObject Ball-2-10 PointMass)` signals that an exemplar case should be constructed, including all statements that mention the entity `Ball-2-10` in the problem plus the `abstractionForObject` statement. Since the worked solution contains a second participant abstraction, the system generates a separate exemplar in the same manner for the entity `Drop-2-10`. Next, these exemplars are added to their appropriate generalization contexts as indicated by Figure 4 and generalized via SEQL as described above.

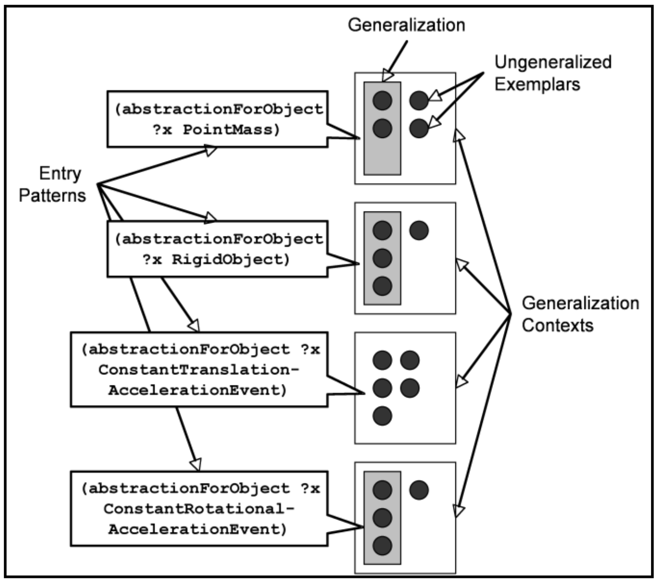
Generalization
Ungeneralized Exemplars
(abstractionForObject ?x PointMass)
Entry Patterns
(abstractionForObject ?x RigidObject)
Generalization Contexts
(abstractionForObject ?x ConstantTranslation-AccelerationEvent)
(abstractionForObject ?x ConstantRotational-AccelerationEvent)

Figure 4: Example generalization contextualization

This allows the system to maintain several contexts simultaneously, each representing a participant abstraction with its own lists of generalizations and ungeneralized exemplars.

As new worked solutions are made available, our method builds participant abstraction examples and adds them to the appropriate generalization contexts. Therefore, our method learns incrementally by refining and extending its generalizations.

Making Participant Abstraction Decisions

Given a problem, a domain theory, and contextualized generalizations of participant abstractions, our method uses analogy to determine if and how entities in the problem should be included in the model. In addition to making the modeling decision, our method returns a *confidence* value (0-1) as a heuristic for confidence in the decision.

The algorithm listed in Figure 5 is performed on every entity in the problem. The process consists of three steps: building a case around the entity, comparing it against the best match from each generalization context, and deciding which, if any, abstraction is appropriate for the entity.

Our method begins by building an *entity case* from the entity. As in building the worked solution exemplars, we include all facts in the problem that mention the entity. The system then compares this case to each generalization context.

From each generalization context, our method identifies the generalization or exemplar with the highest structural evaluation score via SME comparison with the entity case. The systematicity principle implemented in SME means that matches with deeper relational structures have higher structural evaluation scores; therefore, the best mapping for

```
1) Given entity, e, from problem, P
   a) Build entity case, ec, with each fact in P
      mentioning e
2) For each Generalization Context gc_i
   a) Compare ec with each exemplar and generalization
      within gc_i
   b) Use the best matching exemplar or generalization
      as the base of an analogy with ec
   c) If a candidate inference of this match includes a
      fact of the form: (abstractionForObject e gc_i)
      i)  Return the normalized structural evaluation
          score for this match as the confidence for this
          generalization context
      ii) Otherwise, return 0
3) Select as the participant abstraction for e from the
   generalization with the highest confidence
   a) If all generalization context score 0, do not make a
      participant abstraction for e
```

Figure 5: Participant abstraction decision algorithm

a generalization context is not necessarily the largest, but the one with the most relational structure.

The confidence value of the match is computed by analyzing several aspects of the match. First, our method analyzes the candidate inferences of the match between the best match and the entity case. Because every case in the generalization context has an `abstractionForObject` fact, we search the mapping for a corresponding candidate inference in the target (entity case) under consideration. If there is no such candidate inference, the confidence for this abstraction is zero. Otherwise, the confidence value is the SME structural evaluation score normalized against a self-match of the exemplar or generalization. Normalization is necessary for comparing confidence values across generalization contexts. Normalizing against the best match means the maximum confidence score approaches one as the entire exemplar or generalization, aside from the `abstractionForObject` statement, participates in the mapping.

The confidence values are compared, and the system identifies the generalization context that generated the highest confidence value. The participant abstraction represented by this generalization context is selected as the abstraction for the entity. If the highest confidence value is zero, the entity is not considered a participant in the model.

Evaluation

Our evaluation focuses on exploring the following questions. First, is our method able to make accurate participant abstraction decisions? Second, does our method's performance improve as examples are added to the system? Finally, does the confidence value provide a useful heuristic in determining the accuracy of a participant abstraction for a particular problem entity?

Method

Our materials include five linear kinematics problems and five rotational kinematics problems. In these problems, there are 34 entities, of which 21 should be modeled as one of four different participant abstractions: `PointMass`, `LinearConstantAccelerationEvent`, `RigidObject`, `RotationalConstant-AccelerationEvent`. To evaluate the effect of learning, we created four conditions based upon the size of the training set (2, 4, 6, and 8). To ensure that each generalization context has at least one exemplar, each training set consists of an equal number of problems from linear and rotational kinematics. Using the worked solution for each training set problem, we added participant abstraction exemplars to the appropriate generalization contexts. The remaining problems were used for testing. That is, for each entity in each problem our method selected a participant abstraction based upon the generalizations created by the training set. We evaluated every possible combination of problems for the training sets in each trial (size 2=25 trials, size 4=100 trials, size 6=100 trials, and size 8=25 trials).

For each decision, we compare the result of our method to the desired result, as indicated by the worked solutions. There are five possible results:

1. Correct: The entity was a model participant and identified correctly.
2. Correctly Ignored: The entity was not a model participant and was not identified as one.
3. Extraneous: The entity was not a model participant, and our method selected an abstraction.
4. Wrong: The entity was a model participant, but was identified as the wrong abstraction.
5. Failed: The entity was a model participant, but was not identified as any abstraction by our method.

Correct and correctly ignored answers are considered successful modeling decisions. Extraneous answers result in more participants to consider when formulating the model, but should not cause errors when solving the problem. In the worst case, additional encapsulated histories will be instantiated resulting in valid but irrelevant equations for the problem-solver to consider. Wrong and failed answers are errors, as they provide the rest of the model formulation process with incorrect information.

Results

As the number of trials varies by the size of the training set and the number of entities per trial depends on the problems in the test set, each condition has a different number of total participant abstraction decisions. Therefore, we report the frequency of each decision type as a percentage of the total decisions made in Table 1.

These results support our hypothesis that our method is able to learn to make participant abstraction decisions.

Training Set Size (# of decisions)	2 (680)	4 (2040)	6 (1360)	8 (170)
Correct	72%	74%	75%	76%
Correctly Ignored	17%	16%	15%	14%
Extraneous	6%	7%	8%	8%
Wrong	4%	2%	1%	.5%
Failed	0%	0%	0%	0%

Table 1: Participant abstraction decision results

With two worked solutions, the system made successful inferences (correct + correctly ignored) 89% of the time, extraneous inferences 6% and incorrect inferences (wrong + failed) 4% of the time. Furthermore, these results support the learning hypothesis because the number of incorrect decisions decreases down to 0.5% as the number of worked solutions in the training set increases to eight.

Figure 6 contains a graph of our method's mean confidence values for each of the inference categories. Correctly ignored and failed decisions always have a confidence value of 0; consequently, they are not shown. The confidence values are a useful discriminator. The correct answer values are significantly different from the irrelevant and wrong answers ($p < .001$). Additionally, our method's confidence values for correct classifications is significantly higher ($p < .001$) with eight worked solutions than with two, supporting our learning hypothesis.

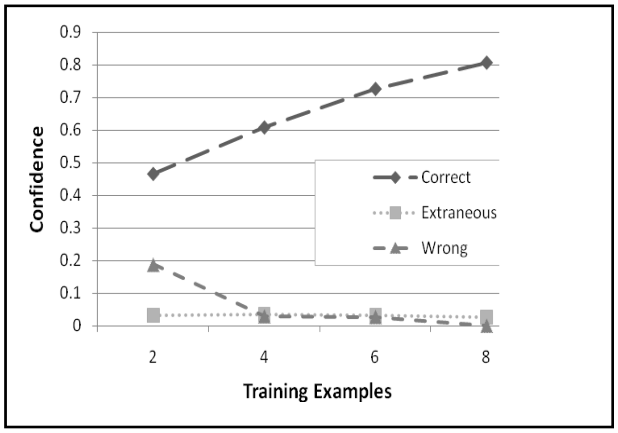

Figure 6: Confidence by answer type and number of examples

Discussion

These results indicate that our method is effective for making participant abstraction modeling decisions. Our method not only makes these decisions, but also returns a confidence score, permitting additional reflection during the model formulation process. Furthermore, our method has a learning component, such that its decisions and confidence estimates improve with experience. We can explain these results by noting that SEQL generalizations abstract away the aspects of the exemplars that are not shared across scenarios. As such, this focuses the participant abstraction decision on the appropriate

```
Generalization:
  (Automobile :genent0)
  (primaryObjectMoving :genent1 :genent0)
  (valueOf
   (AtFn (Acceleration :genent0) :genent1)
   (MetersPerSecondPerSecond :genent2))
  (abstractionForObject :genent0 PointMass)
Exemplar:
  (PassengerAirplane Jet-2-19-P)
  (TurbojetPropelledAircraft Jet-2-19-P)
  (primaryObjectMoving TakeOff-2-19-P Jet-2-19-P)
  (objectStationary
    (StartFn TakeOff-2-19-P) Jet-2-19-P)
  (valueOf  (AtFn (Speed Jet-2-19-P)
                  (EndFn TakeOff-2-19-P))
            (MetersPerSecond 80))
  (querySentenceOfQuery Gia-Query-2-19-P
   (valueOf (AtFn (Acceleration Jet-2-19-P)
                  TakeOff-2-19-P)
         Acceleration-2-19-P))
  (abstractionForObject Jet-2-19-P PointMass)
```

Figure 7: Generalization abstracts away unnecessary facts, highlighting important relations

relational structure in the problem representation. The few failure cases within our results are due to extraneous relational structure within the generalization contexts because the system has not seen enough examples.

This behavior is more evident when we compare a generalization and an exemplar from the experiment, both of which are illustrated in Figure 7. This generalization contains four facts and three generalized entities. `:genents` are entities that have been abstracted by SEQL. The generalization contains an entity that is an automobile, which is the primary object moving in some event with a known acceleration. On the other hand, the exemplar contains seven concrete facts about a jet plane taking off, some of which may complicate the modeling decision. For example, the `objectStationary` fact provides distracting relational structure that could align erroneously with facts in the entity case and result in incorrect modeling decisions. As generalizations are formed from additional examples, however, our method is better able to extract and preserve the relational structure important for making modeling decisions.

Related Work

As noted previously, the majority of model formulation work has focused on ascertaining the levels of detail and perspectives that should be used in a model, given a particular task (cf. Falkenhainer & Forbus 1991; Nayak 1994 Rickel & Porter 1994). A notable exception is Flores and Cerda's (2000) work in analog electronics, which formalized a number of equivalent circuit configurations as rewrite rules to simplify circuit schematics in a human-like way. While these systems perform well in the domains in which they were designed, the goal of this work is to learn how to make modeling decisions in new domains based upon examples. By focusing on learning, we believe our approach will be applicable in a wide variety of domains.

An alternative to these rule-based approaches is analogical model formulation (Klenk *et al.* 2005; Klenk & Forbus 2007). Motivated by the observation that engineers frequently use analogies with their experiences in formulating new models (Falkenhainer 1992), analogical model formulation allows an agent to make a number of modeling decisions about a situation, described in everyday terms, based upon explanations of similar situations. In this work, our method learns how to make participant abstraction decisions via generalization. These generalizations allow the learned knowledge to be applied more generally than in analogical model formulation.

Conclusion & Future Work

This paper presents a method for learning participant abstraction decisions from examples via generalization. We present results from an evaluation in which participant abstraction decisions were learned and applied in the physics domain.

This represents a significant step towards building systems that learn how to model situations from examples. While our results demonstrate the utility of generalizing at the granularity of the model participant decision, we plan on extending this method to other modeling decisions. For example, we plan to explore how generalization could be used to learn situation-appropriate simplifying or operating assumptions (e.g. ignoring friction, laminar flow, or elastic collisions). We also plan to investigate generalization at the level of physical processes or encapsulated histories, perhaps accelerating the model formulation process with experience.

The ability to leverage previously understood domains when faced with new domains is an important frontier for AI research. We plan to incorporate this method for learning domain specific modeling decisions into our Domain Transfer via Analogy (DTA) framework (Klenk & Forbus 2007), which uses multiple cross domain analogies to transfer domain theories between areas of physics. Transferring the modeling knowledge encoded in these generalizations is an important direction for transfer learning research.

References

de Kleer, J. 1977. Multiple representations of knowledge in a mechanics problem solver, pp.299–304. *Proc. IJCAI-77.*

Falkenhainer, B. 1992. Modeling without amnesia: Making experience-sanctioned approximations. *Proceedings of QR02.*

Falkenhainer, B. and Forbus, K. 1991. Compositional modeling: finding the right model for the job. *Artificial Intelligence* 51:95–143.

Falkenhainer, B., Forbus, K. and Gentner, D. 1989. The Structure-Mapping Engine: Algorithm and examples. *Artficial Intelligence.* 41.

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

Flores, J. and Cerda, J. 2000. Efficient modeling of linear circuits to perform qualitative reasoning tasks. *AI Communiations*, 13(2) 125-134.

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

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

Klenk, M. & Forbus, K. 2007. Learning domain theories via analogical transfer. *Proceedings of Qualitative Reasoning Workshop.* Aberystwyth, UK.

Klenk, M. & Forbus, K. 2007. Measuring the level of transfer learning by an AP physics problem-solver. *Proceedings of Association for the Advancement of Artificial Intelligence (AAAI-07).* Vancouver, Canada.

Klenk, M., K. Forbus, E. Tomai, H. Kim, and B. Kyckelhahn. 2005. Solving everyday physical reasoning problems by analogy using sketches. *Proceedings of the American Association for Artificial Intellegence (AAAI-05).* Pittsburgh, PA.

Kuehne, S.E., Forbus, K.D., Gentner, D., & Quinn, B. (2000). SEQL: Category learning as progressive abstraction using structure mapping. *Proceedings of CogSci 2000,* August.

Matuszek, C., J. Cabral, M. Witbrock, and J. DeOliveria. An Introduction to the Syntax and Content of Cyc. 2006. *Proceedings of AAAI-06 Spring Symposium on Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering.* Standford, CA.

Nayak, P. 1994. Causal approximations. *Artificial Intelligence* 70:277–334.

Rickel, J. and Porter, B. 1994. Automated modeling for answering prediction questions: selecting the time scale and system boundary, pp. 1191–1198. *Proc. AAAI-94*.