Order Number 8916244

Learning from physical analogies: A study in analogy and the explanation process

Falkenhainer, Brian Carl, Ph.D.

University of Illinois at Urbana-Champaign, 1989

LEARNING FROM PHYSICAL ANALOGIES: A STUDY IN ANALOGY AND THE EXPLANATION PROCESS

BY

BRIAN CARL FALKENHAINER

B.S., Santa Clara University, 1982
M.S., University of Illinois, 1985

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1989

Urbana, Illinois

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

THE GRADUATE COLLEGE

DECEMBER 1988

WE HEREBY RECOMMEND THAT THE THESIS BY

BRIAN CARL FALKENHAINER

ENTITLED LEARNING FROM PHYSICAL ANALOGIES:

A STUDY IN ANALOGY AND THE EXPLANATION PROCESS

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR

THE DEGREE OF DOCTOR OF PHILOSOPHY

[illegible]
Director of Thesis Research

[illegible]
Head of Department

Committee on Final Examination†

[illegible]
Chairperson

[illegible]

Robert E. Stepp

David C. Wilkins

[illegible]

† Required for doctor's degree but not for master's.

O-517

LEARNING FROM PHYSICAL ANALOGIES: A STUDY IN ANALOGY AND THE EXPLANATION PROCESS

Brian Carl Falkenhainer, Ph.D.
Department of Computer Science
University of Illinois at Urbana-Champaign, 1989
Kenneth D. Forbus, Advisor

To make programs that understand and interact with the world as well as people do, we must duplicate the kind of flexibility people exhibit when conjecturing plausible explanations of the diverse physical phenomena they encounter. This process often involves drawing upon *physical analogies* – viewing the situation and its behavior as similar to familiar phenomena, conjecturing that they share analogous underlying causes, and using the plausible interpretation as a foothold to further understanding, analysis, and hypothesis refinement.

This thesis investigates analogical reasoning and learning applied to the task of constructing qualitative explanations for observed physical phenomena. Primary emphasis is placed on two central questions. First, how are analogies elaborated to sanction new inferences about a novel situation? This problem is addressed by *contextual structure-mapping*, a knowledge-intensive adaptation of Gentner's structure-mapping theory. It presents analogy elaboration as a *map and analyze* cycle, in which two situations are placed in correspondence, followed by problem solving and inference production focused on correspondence inadequacies. Second, how is the quality of a proposed analogy evaluated and used for some performance task? A theory of *verification-based analogical learning* is presented which addresses the tenuous nature of analogically inferred concepts and describes procedures that can be used to increase confidence in the inferred knowledge. Specifically, it relies on analogical inference to hypothesize new theories and simulation of those theories to analyze their validity. It represents a view of analogy as an iterative process of hypothesis formation, testing, and revision.

These ideas are illustrated via PHINEAS, a program which uses similarity to posit qualitative explanations for time-varying descriptions of physical behaviors. It builds upon existing work in qualitative physics to provide a rich environment in which to describe and reason with theories of the physical world.

Acknowledgements

I would like to thank Ken Forbus for being a great advisor, coworker, and friend. He has had a significant positive impact on my research content and perspective. The research environment he has established abounds in information, code, and resources, enabling one to think creatively about all the possibilities.

I would also like to thank Dedre Gentner, who has acted as my co-advisor in this work. It's a joy to work with someone so eager to debate and listen openly to alternate lines of thinking. Her work formed the seed around which this thesis has grown.

Professors Gerald DeJong, Robert Stepp, and David Wilkins served on my final examination committee, providing encouragement and suggestions for improvements.

This thesis and my sanity have benefited from the intuitions and comradery provided by the other members of the qualitative reasoning group. John Collins, Dennis DeCoste, John Hogge, Paul Nielsen, Gordon Skorstad, Janice Skorstad, and Barry Smith have stimulated many interesting ideas. Special thanks is due John Collins for his inspirational conversation and unique insight. He has both directly and indirectly affected the content and quality of this thesis and is the source for several of the more innovative analogy examples.

Dennis Decoste also deserves special credit for providing DATMI, and for adding features vital to my specific needs. It's unusual to be able to say "tell me what to type" and an hour later have a new, fully functional module in your system, with never a single error!

Over the years I have learned much from interacting with the EBL/SDBL community led by Professors Gerald DeJong and Robert Stepp. Special thanks go to Scott Bennett, Steve Chien, Lawrence Holder, Ray Mooney, Shankar Rajamoney, Bob Reinke, Bharat Rao, Jude Shavlick, and Brad Whitehall.

I would also like to thank Shankar Rajamoney for showing me new facets of the theory development problem. This had a significant and beneficial impact on this thesis.

Mark Burstein, Russell Greiner, Paul O'Rorke, Stuart Russell, Jeff Shrager, and Tom Eskridge, through both published and personal communication, have influenced this work and forced me to think more deeply about what it is I'm actually doing.

I am grateful to Professor Klosinski for introducing me to the wonders of computer science and saying "you should go to Illinois", Dr. Barker for introducing me to scientific research, and Jim Gill for introducing me to AI.

Closer to home, I am most grateful to my parents and grandparents for their support, encouragement, and faith in me throughout the years. They are always only a phone call away.

And finally, my love and gratitude go to my wife Sue, who has supported this thesis well beyond the call of duty. I hope to now make up for the long hours I have been at work over the last few years.

This research has been supported by an IBM Graduate Fellowship, two University of Illinois Cognitive Science / Artificial Intelligence summer fellowships, and by the Office of Naval Research, Contract No. N00014-85-0559. I am grateful for their support.

Table of Contents

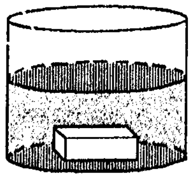
Chapter 1

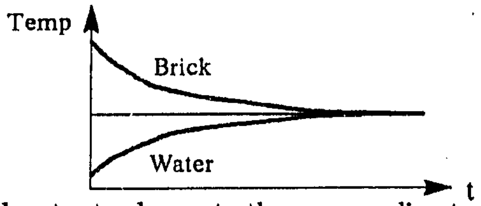
Learning from Physical Analogies

We cannot, coming into something new, deal with it except on the basis of the familiar and the old-fashioned... We cannot learn to be surprised or astonished at something unless we have a view of how it ought to be; and that view is almost certainly an analogy. We cannot learn that we have made a mistake unless we can make a mistake; and our mistake is almost always in the form of an analogy to some other piece of experience. (Oppenheimer, 1956, pg. 129-130)

To make programs that understand and interact with the world as well as people do, we must duplicate the kind of flexibility people exhibit when conjecturing plausible explanations of the diverse physical phenomena they encounter. I view this flexibility as arising from an ability to detect similarities, within and across domains, between the various phenomena. Interpreting an observation often requires the flexible integration of knowledge from multiple sources and the formation of new theories about the world. For example, suppose some unusual behavior is observed. What is causing it? In general, the process of constructing a satisfactory explanation will involve drawing upon *physical analogies* – viewing the situation and its behavior as similar to familiar phenomena, conjecturing that they share analogous underlying causes, and using the plausible interpretation as a foothold to further understanding, analysis, and hypothesis refinement.

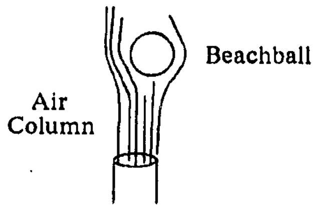
The goal of this work is to develop a system that can offer plausible answers to the types of questions shown in Figure 1.1 by relating these unfamiliar situations to more familiar concepts. The first example, a hot brick immersed in cold water, is taken from the classic fluid flow – heat flow analogy that led to the development of the caloric theory of heat. The second example, a beach ball suspended in a jet of air, demonstrates how approximate hypotheses proposed through similarity may be discriminated by envisioning their consequences. The third example demonstrates how the same analogy approach may





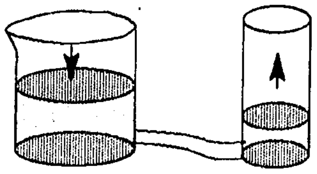
- What causes a hot brick and cold water to change to the same median temperature when the brick is immersed in the water?

This situation may be explained if it is assumed analogous to liquid flowing from one container to another. The assumption is supported by the brick's temperature and the water's temperature asymptotically approaching equality. It requires further assumptions that the brick and water contain hypothesized substance sk-water-6 and that the brick and water temperatures are proportional to the amount of sk-water-6 they contain. Then, when the brick and water are placed in contact, their difference in temperatures causes the transfer of sk-water-6 from the brick to the water. This explanation consistently predicts the observed behavior.



- A beachball may be suspended in a vertical column of flowing air, as in a vacuum cleaner set in reverse. What holds the ball in place and why is it so stable?

Two explanations may be proposed. In one, the air is pushing on either side of the ball, holding it in place. In the other, an air pressure difference is pulling on either side of the ball, holding it in place. However, the first explanation fails to accurately explain the situation, since offsetting the ball to the right results in increased force towards the right, which is not stable. The second explanation consistently predicts the observed behavior, since offsetting the ball to the right results in increased force to the left, which is stable.



- A beaker and a vial, each containing water, are connected by object3. What is causing the water in the beaker to decrease while the water in the vial is increasing?

This situation appears to be an instance of liquid flow if it is assumed that object3 is a fluid path. Then, the beaker pressure being greater than the vial pressure would cause the water to flow from the beaker to the vial through object3, until their pressures are equal. This explanation consistently predicts the observed behavior.

Figure 1.1: The types of questions this thesis is designed to answer.

be used to provide answers to conventional abductive inference situations, in this case by finding that liquid flow is the best analogue to the given liquid flow scenario.

This chapter begins by suggesting that explanation and analogy share a common core, in which the search for explanatory similarity is the driving force behind all forms of explanation. Crucial to such a unified view is a powerful model of analogy, which is summarized in Section 1.2. Section 1.3 then defines a special case of general explanatory analogies, physical analogies. Section 1.4 introduces PHINEAS, a program which uses physical analogies to provide plausible explanations of observed physical phenomena and enhance its understanding of the physical world. Finally, Section 1.5 provides a reader's guide to the subsequent chapters of this thesis.

1.1 Explanation

An important aspect of understanding and interacting with the physical world is proposing causal explanations for what we see around us. When existing knowledge is sufficient for explanation, this is commonly called the interpretation or diagnosis task: select the theory that provides the best account of the phenomenon. When knowledge is lacking, the task becomes one of theory formation: conjecture a new or revised theory that will account for the phenomenon. Typically, explanation, interpretation, and diagnosis are decoupled from theory formation in AI. Yet they are intimately related, representing different levels of certainty within a common process. Interpretation typically involves making assumptions due to incomplete knowledge about the situation. Theory formation typically involves making assumptions about both the situation and the incompleteness of current theories

Abduction is the process that generates explanations. It is a form of inference that fits the following pattern (Josephson et al., 1987):

D is a collection of data (facts, observations, givens);
H explains D (would, if true, imply D);
no other hypothesis explains D as well as H does;

therefore, H is correct.

For example, if we see that the grass is wet, and we know that rain makes the grass wet, we might hypothesize that it had rained recently. If the sky was cloudy, this would lend credence to our hypothesis. However, this is a guess. The sprinklers might have been on recently (Pearl, 1987).

Abduction is inference to the best explanation, that is, if the hypothesis were true, it would explain the phenomenon. There are two key phrases here. "If it were true" indicates

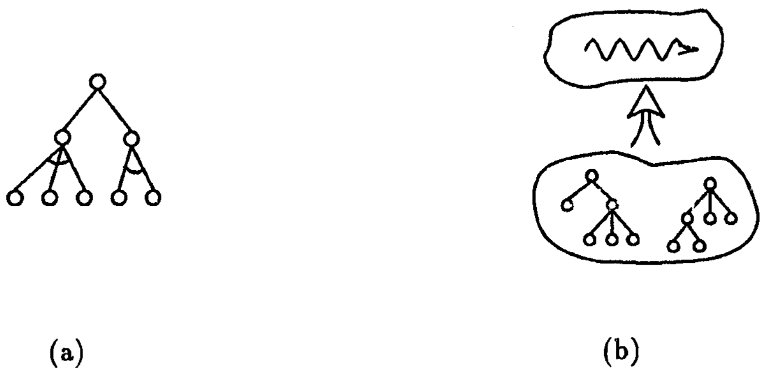


Figure 1.2: Two extremes of abduction: (a) simple backward chaining on an atomic goal and (b) best match relating explanation patterns to observation patterns.

that not all of the relevant knowledge may be known and assumptions may be required to fill in the gaps. The process of finding candidates and the assumptions that must be made along the way will be called the ***interpretation-construction task.*** "It would explain the phenomenon" indicates that the hypothesis would explain the phenomenon, not that it is the correct explanation. There may be other hypotheses that would explain it as well. The process of deciding which hypothesis is the best explanation will be called the ***interpretation-selection task.***

Analogy and abduction share both interpretation tasks. They must each propose a plausible set of candidates that fit the current situation and they must each select from those candidates the one(s) that will be accepted as the final interpretation. The model of explanation developed in this thesis is based on the view that analogy suffices as the central process model for explanation tasks. There are two arguments supporting this view.

First, consider the basic abduction process. The backchaining model of abduction works well for explaining simple atomic occurrences (Figure 1.2(a)), such as `Wet(grass)`. However, as the complexity of the phenomenon being explained increases, our ability to simply backchain to a small set of plausible causes diminishes. The entirety of the situation must be considered and all of the interrelations between aspects taken into account (Figure 1.2(b)). In a complex, multi-faceted phenomenon, meaning arises out of consideration of the whole. Hence, most multi-faceted explanation systems are based on some form of macro-matching, typically in terms of schemas or frames, that seeks minimal hypothesis sets maximally fitting the data. This is true of script-based models of story understanding (e.g., Charniak, 1972; DeJong, 1982), process models for interpreting the behavior of

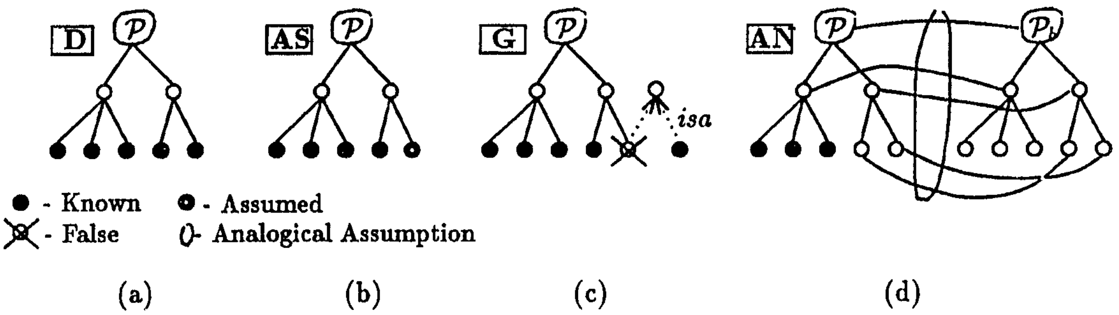


Figure 1.3: Four alternative explanation scenarios: (a) deduction scenario, (b) assumption scenario, (c) generalization scenario, (d) analogy scenario.

a physical system (e.g., Forbus, 1986a), and composite matching models of abduction and diagnosis (e.g., Reggia, 1983; Josephson et al., 1987). The desire for a minimal hypothesis, best match is further implicitly reflected in the Occam's razor heuristic found in simpler models, which backtrack on one piece of data at a time (e.g., Pople, 1973). In other words, interpretation and explanation are a form of best match process, with the goal of matching the current situation to that which could explain it. Adaptability can be achieved without loss of function by generalizing the identicality of unification-based models with the more general, constrained similarity match of analogy.

Second, consider the following explanation scenarios (Figure 1.3):

Deduction scenario: Given phenomenon $\mathcal{P}$, where $\mathcal{P}$ represents a set of observables, a complete explanation of $\mathcal{P}$ deductively follows from existing knowledge. The only open question is if it is *the* explanation, as there may be others. For example, suppose fluid flow is observed and all of the preconditions for fluid flow are known to hold (e.g., the source pressure is greater than the destination pressure, the fluid path is open, etc.). Then a fluid flow explanation directly follows. Given the observed behavior and the existing preconditions, we could say that the situation is trivially analogous, or literally similar to liquid flow.

Assumption scenario: Phenomenon $\mathcal{P}$, where $\mathcal{P}$ represents a set of observables, is given. No explanation can be grounded with current knowledge because not all of the relevant facts are known. However, a complete explanation follows from the union of existing knowledge and a consistent set of assumptions about the missing facts. For example, if liquid flow is observed but we don't know if the fluid path valve is open or closed, we may assume the valve is open if there is no evidence to the contrary.

Generalization scenario: Phenomenon $\mathcal{P}$, where $\mathcal{P}$ represents a set of observables, is given. Existing knowledge indicates that candidate explanation $\mathcal{E}$ cannot apply because condition C_1 is known to be false in the current situation. However, $\mathcal{E}$ does follow if condition C_1 is replaced by the next most general relation, since C_1's sibling is true in the current situation. This is a standard knowledge base refinement scenario (e.g., Smith et al., 1985).

Analogy scenario: Phenomenon $\mathcal{P}$, where $\mathcal{P}$ represents a set of observables, is given. No candidate explanation $\mathcal{E}$ is available directly, but explanation $\mathcal{E}_b$ is available if a series of analogical assumptions are made, that is, if the situation explained by $\mathcal{E}_b$ is assumed analogous to the current situation. For example, if heat flow is observed, but little is known about heat phenomena, then an explanation may be constructed by analogy to liquid flow.

Each scenario must perform the interpretation-construction task: retrieve from memory explanatory hypotheticals matching the current situation. Each must perform the interpretation-selection task: select from a set of candidate hypotheses that which is most probable, plausible, coherent, etc. If the best we can do is distant analogy, then that is the best we can do. Why should abductive assumptions be limited to assumptions over the boolean truth values of unknown facts? Abduction should be broadened to include assumptions about the validity of a generalization or the validity of an analogy relation. Is osmosis a generalization of standard liquid flow, or is it merely analogous? What about a siphon? At what point does something stop being a within-domain, literal similarity comparison and become an across-domain analogy comparison? Ideally, an explanation facility should be able to say "the observed phenomenon $\mathcal{P}$ is extremely like phenomenon $\mathcal{P}'$ in the relevant relations", even if some required condition is believed absent. Consider a chemical reaction R needing catalyst C. If the effects of R are seen, yet the catalyst is not present, we should still be reminded of that chemical reaction if no better hypothesis exists.

Existing explanation systems suffer from the *adaptability problem*: they are unable to offer a best guess in light of an imperfect domain theory and unable to apply knowledge of one domain to the understanding of another. We can solve this problem by (1) generalizing abduction to include analogical assumptions and (2) developing a unified architecture to support such abduction. This motivates the following conjecture:

- *Similarity conjecture:* All interpretation-construction tasks may be characterized as the search for maximal, explanatory similarity between the situation being explained and some previously explained scenario. The previous situation may be drawn from an actual experience, a prototypical experience, or an imagined scenario derivable from general knowledge.

In some sense this conjecture is conservative and plausible. Yet historically AI systems have not operated this way. Part of the reason has been the view that identifying similarity is computationally intractable (Hayes-Roth & McDermott, 1978; Winston, 1980; Kline, 1983; Greiner, 1986). This thesis counters that claim with an efficient and flexible algorithm for performing similarity matches, implemented in a program called SME. A consequence of the similarity conjecture is that the strong distinction between deductive explanation processes and analogical explanation processes should be eliminated. While phenomenologically distinguishable, this does not necessarily imply a distinction in the process model. The same basic processes may be used in each explanation scenario and the distinctions between them correspond to how well existing knowledge supports the explanation. Everyday, common deductive operations correspond to the high confidence derived from identicality matches. We can reformulate the traditional abduction processes as being special cases of analogical explanation.

A corollary to the similarity conjecture is that the same basic processes are used in both scientific theory formation and everyday interpretation and hypothesis formation. The famous, analogy-induced discoveries chronicled in the history of science literature are rare because (1) science is difficult and (2) they represent analogical reasoning in its purest and most difficult form - the sifting through irrelevant details to recognize similarities that lay hidden and thus reformulate a problem in an entirely new light. A scientific theory is distinguishable from everyday conjectures by its degree of specificity in accounting for phenomena and by how carefully it is analyzed. However, there is nothing fundamentally different in the basic process (see (Leatherdale, 1974) for a similar claim).

The benefits of this view are a single computational architecture for explanation processes. Distinctions between explanation types only influence weighing of evidence and deciding whether a new conjecture represents a revision of existing knowledge or a new separate body of knowledge. This thesis seeks to demonstrate the feasibility of this view by developing a system that uses analogical similarity to focus the search for explanations and develop novel theories when existing knowledge is insufficient. It should not be construed as a claim that the explanation problem has been solved by using analogy. Rather, it is intended as an important first step towards that goal.

1.2 Analogy

The primary research topic of this thesis is the analogy process, with explanation in physical domains the application task. How are analogies initially recognized? How are they elaborated to sanction new inferences about a novel situation? What is analogy's role as aid to a more global performance element? In particular, how may it be applied to the task of providing plausible explanations of physical phenomena?

Analogy may be described as having three functional blocks: given a current *target* situation (1) *access* retrieves another description, the *base*, from memory which is analogous or similar to the target in some respects, (2) *mapping* isolates a set of correspondences between the base and target and uses them to support the transfer of additional base knowledge to the target, and (3) *evaluation and use* estimates the quality of the analogy and uses it for some performance task.

A central point of this thesis is that analogy is often an active, iterative process of hypothesis formation, testing, and revision. Previous accounts of analogy have focused on matching two descriptions to produce new knowledge. My account goes beyond these by explicating the role of analogy in explanation and learning. This thesis explores several new questions in analogical reasoning and learning. How is an analogy used once it is established? What role does analogy play during an extended period of reasoning and problem solving? Analogy may sometimes be an atomic, "single inference" process. However, often it is not.

1.2.1 Access

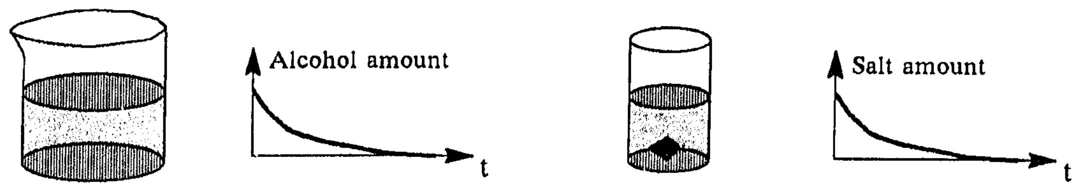
In previous models of analogy, one starts with a partially understood model of a domain (or a teacher-supplied analogical hint which serves the same purpose). This incomplete model is then used to gain access to a fully understood analogue and to constrain the mapping from the analogue to the current model. Analogical *learning* situations pose an interesting problem – when entirely new theories are developed, or knowledge of the current domain is strongly underspecified, such a model of analogy fails to function due to lack of information. This led Burstein (1986, pg. 352) to claim that matching "cannot be the basis of a general theory of learning by analogy". However, this precludes autonomous learning (his system worked in the context of a tutor). Something must match to trigger the initial recognition of similarity, and this match may then be used to sanction further inferences. I claim that three types of information are generally available in autonomous learning situations: (1) knowledge of what information is desired and what purpose it should serve, (2) *correlations* between available and desired information, and (3) *abstractions* characterizing various aspects of the situation. Important strategies in accessing analogies are keying on whatever knowledge is most readily available, is most complete, and is correlated to information sought. For example, given a complete behavioral model and a scant causal model, it would be wise to key on similar behavior rather than focusing solely on the causal model of primary interest. The resulting reminding provides a candidate analogue and an initial set correspondences that constrain the mapping process.

This thesis adopts that strategy. Behavioral similarity is used to initiate and guide the analogy process. For example, consider constructing an explanation for the heat flow situation depicted at the top of Figure 1.1 (a hot brick immersed in cold water). Little is known about thermal phenomena and what causes the two objects shown to change temperature the way they do. However, more is known about the behavior. This may be used to seek an experience (real, prototypical, or constructed) that demonstrates the same detailed type of behavior. Knowledge of the particular physical configuration, the domain, and any initial interpretations that fill in some gaps may be used to constrain the search and augment the description when the behavior is underspecified. For example, the way the temperatures are asymptotically approaching each other suggests that perhaps a single exchange is taking place between the two objects. Suppose liquid flow is suggested based on how its behavior is similar to what is happening. This similarity suggests an initial set of correspondences between the two situations: the cooling brick to the source of liquid flow, the heating water to the destination of liquid flow, and temperature to pressure. We can then recall what explains the liquid flow behavior and attempt to map it to the current *heat flow* situation.

1.2.2 Mapping

Once a candidate analogue has been identified, mapping serves to complete the initial set of correspondences (*matching*) and propose inferences sanctioned by those correspondences (*carryover*). The approach to mapping used in this thesis is a modification of Gentner's (1980, 1983, 1988) *Structure-mapping theory* of analogy. Structure-mapping provides rules for analogical mapping which are based solely on the structural properties of domain descriptions, rather than on their content. Furthermore, it introduces the *systematicity principle*, which states that the amount of common higher-order relational structure determines which of several possible matches is preferred.

Chapter 2 reviews the structure-mapping theory and describes limitations that have been found with it. These center around its assertion that relations must match identically and that only the structural properties of domain descriptions determine the mapping. No



```
Decreasing[Amount-of(alcohol1)]
Container(beaker2)
Contained-Liquid(alcohol1)
Container-of(alcohol1,beaker2)
Open(beaker2)
```

Disappearing Alcohol Scenario

```
Decreasing[Amount-of(salt1)]
Container(glass4)
Contained-Liquid(water1)
Container-of(water1,glass4)
Solid(salt1)
Soluble-in(salt1, water1)
Immersed-in(salt1,water1)
Explains(Dissolving-Process,salt-behavior-7)
Supports[Immersed-in(salt1,water1),
         Physical-Contact(salt1,water1)]
```

Dissolving Scenario

Figure 1.4: Potentially analogous situation descriptions. In one, alcohol left sitting in an open beaker is disappearing. In the other, salt is dissolving in a glass of water.

domain-specific or problem-specific information is used by the mapping process. It would have difficulty, for example, establishing a correspondence between the cylindrical shape of one liftable cup and the handle of another liftable cup.[1] It could not reason about the content of the situation descriptions to see that the two, non-identical expressions provide the same function.

This thesis adopts a knowledge-intensive view of the structure-mapping paradigm, called *contextual structure-mapping*, which is described in Chapter 2. It uses knowledge of the representations being matched and the current reasoning task to help constrain and guide the match.

Suppose we observe a situation in which alcohol left sitting in an open beaker is disappearing. Further suppose that our knowledge of physical phenomena is limited to various kinds of liquid flows, heat flows, boiling, and dissolving. We might propose that processes like liquid flow or boiling were taking place. Consider how a third possibility, salt dissolving

[1]This is a recurring example in the machine learning literature, originating in (Winston et al., 1983).

in a glass of water, may be used to propose an explanation for the alcohol's disappearance. The initial descriptions for the alcohol situation (the target) and the dissolving situation (the base) are shown in Figure 1.4. Comparing these situations, two interpretations appear to be possible. In one, `glass4` and `water1` correspond to `beaker2` and `alcohol1`, respectively. In the other, `salt1` would correspond to `alcohol1`, since they are both decreasing. Systematicity, the preference to maximize the amount of shared relational structure, provides little guidance and would appear to prefer the former interpretation. However, the purpose of the analogy is to explain the observed *behavior*, and only the second interpretation provides a comparison which includes the behavior. In contextual structure-mapping, knowledge of current reasoning goals, such as the preference to focus on features being explained, combine with systematicity to influence which matches are preferred.

1.2.3 Transfer

The role of mapping is to establish correspondence and identify base information potentially inferrable for the target. These *candidate inferences* are hypotheses which must pass a series of evaluative processes before being accepted as holding for the target domain. The first of these is *transfer*. Transfer is concerned with importing candidate inferences into the target domain and making them operational. This centers around ensuring consistent expression use and identifying unknown objects proposed by the mapping. The model of transfer developed in this thesis has two important characteristics. First, it provides an expanded account of the complexities in importing proposed analogical inferences into the target domain. Second, mapping and transfer may interact in an iterative manner by repeating the match in light of additional information found during transfer's examination of the proposed inferences.

The `salt1` to `alcohol1` correspondence provides a partial, initial explanation by supporting the inference that something analogous to the `Dissolving-Process` may explain the alcohol's behavior. However, this process describes the interactions between `salt1` and `water1`, yet there is no apparent correspondent for `water1` in the alcohol scenario. Furthermore, predicate type restrictions are violated for some of the imported base relations being applied to target objects. For example, `Immersed-in(alcohol1,?unknown)` is improper, since the first argument to `Immersed-in` must be a solid object, while `alcohol1` is a liquid. To resolve these problems, the carryover component must seek additional knowledge about the alcohol situation. To seek an alternative to `Immersed-in`, it first determines that `Immersed-in` was in the dissolving scenario to ensure physical contact between the salt and the water. Seeking objects having physical contact with the alcohol, we find that the atmosphere is `Touching` the alcohol due to the open beaker.

Detecting and resolving these problems is an important component of attempting to use the results of mapping. Analogical inferences (e.g., `Dissolving-Process` and `Immersed-in(alcohol1,?unknown)`) represent information in the base that had no correspondent in the target description. These must be examined before being posited as new target knowledge. Since base relations are being imported, they may apply predicates to objects or propositions other than their conventional referents. Substitutes for such predicates must be found or created. These inferences may also contain unknown, anticipated objects: slots occupied in the base representation by objects that had no correspondent in the match. A corresponding target object must be found, or the existence of the unknown object postulated.

Since a new object, `atmosphere`, and information about it have been added to the current working vocabulary of the target alcohol scenario, mapping should be repeated using this augmented description. Although not the case in this example, we could find that `atmosphere` is a better correspondent for `salt1` than is `alcohol1`.

Proposed inferences represent holes in the match. They need not represent new knowledge, but may simply indicate places where additional knowledge should be retrieved to help complete the analogy. Transfer conducts focused probes into memory to seek more information about places where the similarity match was incomplete. If additional knowledge is found, transfer will augment the existing base and target descriptions and iterate back to the mapping process, to see if the new information will affect the overall mapping. Mapping and transfer combine to form a *map and analyze* cycle to provide focus to the analogical mapping process. Were we to retrieve everything we possibly knew about the base and target prior to mapping, a great deal of time might be spent on unconstrained inferencing. By initiating the match on what appears to be relevant from immediately available information, transfer may focus on seeking the specific information the match indicates is lacking.

Comparing the original dissolving situation with the augmented disappearing alcohol scenario, we find that `water1` corresponds to `atmosphere`, while `salt1` corresponds to `alcohol1` as before. In the process, we must place `Immersed-in(salt1,water1)` in correspondence with `Touching(alcohol1,atmosphere)`, since both ensure physical contact. This second match is complete enough to enable consistent transfer of the dissolving explanation into the *evaporation* situation. There are no unknown objects in the proposed explanation and all predicate instances are consistent with their declared usage.

This last phase demonstrates an extension to structure-mapping theory's identicality requirement. Knowledge about the representations is used when finding correspondences by stating that items may match if they are *functionally analogous*. Items are functionally analogous if they provide the same inferential support in the context of the structures being matched. This enables `Immersed-in` to match `Touching`. In the cups example described above, it would find that the first cup's cylindrical shape corresponds to the second cup's handle by virtue of their both supporting the common constraint of being liftable.

1.2.4 Verification-Based Analogical Learning

Evaluating the quality of a proposed analogy and using it for some performance task is a crucial phase of the analogy process. Are the proposed inferences correct, likely to be correct, or consistent? Because of the approximate nature of the process, the proposed inferences must be examined to be sure they even predict the very situation they were intended to explain. Most models avoid this validity issue by either stopping once inferences are produced (Holyoak & Thagard, 1988b; Kass et al., 1986) or requiring deductive analogies in which the results could have been achieved (more slowly) without the analogy (Kling, 1971; Carbonell, 1983a; Kedar-Cabelli, 1985b). Inferences produced by analogy must be questioned. This is reflected in the following requirement:

- *Verification requirement:* When possible, the results of analogy must be tested empirically and against other knowledge.

Verification-based analogical learning (VBAL) is designed specifically to satisfy this requirement. In VBAL, analogical learning is seen as an iterative process of hypothesis formation, verification, and revision, centered around the requirement to confirm accuracy and increase the likelihood of being correct. It relies on analogical inference to propose explanations and gedanken experiments (i.e., simulation) to analyze their validity.

Consider the floating beachball behavior introduced in Figure 1.1 and repeated in Figure 1.5. Two hypotheses appear to be possible. One proposes that the air is pushing on the ball from either side, holding it in place. The other recalls that air flowing over an object can have a pulling effect (due to a pressure gradient), as in the lift provided an airplane wing. How may we evaluate these two hypotheses? Having passed all the tests so far (i.e., correlated information selected during access and local consistency checks performed during mapping), the best thing to do is try them out. Do they generate a complete and consistent explanation? If not, then hypothesis generation and revision must continue.

In this case, we find that while both proposals are locally consistent (i.e., no direct contradictions with existing knowledge), only one of them consistently predicts the observed behavior. According to the "pushing" explanation, offsetting the ball to the right results in

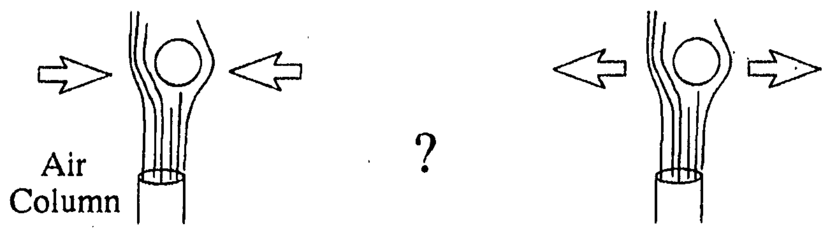


Figure 1.5: A beachball suspended in a vertical column of flowing air is extremely stable. Is the flowing air pushing on both sides, or is it "pulling" on each side?

increased force towards the right due to the additional air flow on the left. This violates the stability aspect of the observed behavior. The "pulling" explanation, however, consistently predicts the observed behavior. Offsetting the ball to the right results in increased force to the left, which draws it back into the center of the air stream.

Had both hypotheses failed, we would be faced with the revision problem: how to refine the results of an analogy when it is found to be awry. Analogy for problem solving is intimately tied with theory revision. It produces approximate, heuristic conjectures that may need adjustment. This thesis depicts the revision stage as being roughly equivalent to the original generation of explanatory hypotheses. It considers behavior analogous to that which led to the need for revision and considers how the current anomalous situation differs from prior situations that were consistently explained. In this manner, revision starts the whole process over again – what might be causing the anomaly, is it similar to other observed behaviors, and so forth.

1.3 Physical Analogies

A causal model of a physical system may be analyzed to give rise to a sequential description of what will happen in the world. For example, Figure 1.6 shows a simple model for a spring-block oscillator and the physical behavior it predicts. The predicted behavior may then be compared to actual observation (Figure 1.7), to verify that the continuous operation of the oscillator corresponds to stepping through the eight predicted states. Thus, physical analogies, such as "Heat is like water", have the unique characteristic that they relate *runnable* models (Waltz, 1981; Gentner & Stevens, 1983), that is, one may actually envision heat "flowing" from one location to another. One may reason about them, make predictions, and observe their results. Furthermore, models of physical systems may be decomposed

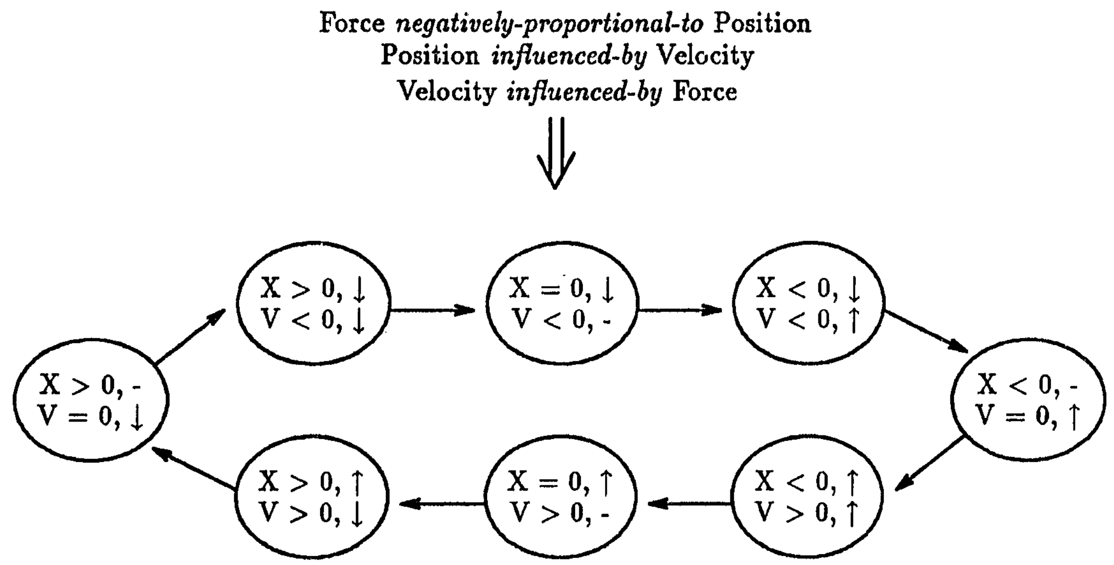


Figure 1.6: A qualitative model for a spring-block oscillator and its corresponding envisionment.

into different perspectives of the same phenomenon. For example, a behavioral model of water flow would describe the physical action of water flowing, while a causal model for water flow would relate the activation of flow to the inequality of pressures.

Physical models are defined to be models about physical phenomena which have the two following characteristics. First, the models being compared must be *generative*, that is, runnable and applicable to new, unforeseen situations. This means that systematic analysis of the model will produce measurable predictions for the physical world. Second, the models must be *decomposable* into different perspectives of the same phenomenon. Thus, the same physical system may have a behavioral model, a causal model, a structural model, and a quantitative model. *Physical analogies* are defined to be analogies which relate physical models. In this work, I focus on learning causal models. A causal model may be defined as a collection of axioms or rules for some domain which explains, by causal argument, the behavior of the domain.

1.3.1 Qualitative Physics

An interesting type of model is a naive or qualitative model of physics. People are able to use common sense knowledge to understand and predict everyday physical occurrences such as throwing a ball, boiling a pot of water, or breaking a glass bowl. They are able to reason about these occurrences in simple, qualitative terms, without need of a formal, mathematical physics. For example, upon encountering a pot of water on a hot stove, a person would be able to predict that "the water will heat up and eventually begin to boil." They are able to use purely qualitative terms, without knowledge of the precise quantities or durations involved.

A *qualitative simulator* takes a model of a particular physical configuration and produces a description of the possible behaviors for the given situation, called an *envisionment.* An envisionment describes physical states and the possible transitions between them. The behavior of the system through time may then be represented as a single path through the envisionment, with each state representing an interval of time in which behavior does not change. Such a path will be called a *history*, after (Hayes, 1979).

This thesis uses Forbus' *Qualitative Process Theory* (1981, 1984) as the primary formalism to represent and reason about change in the physical world. In QP theory, physical changes such as moving, colliding, bending, heating up, and cooling down are thought of as *processes.* A key tenet of Forbus' theory is that processes are central to human knowledge about physical domains (Forbus & Gentner, 1983). In QP theory, a situation is represented as a collection of objects, a set of relationships between them, and a set of processes which account for all changes in the world. Each object has a set of continuous parameters, such as `TEMPERATURE` and `PRESSURE`, which are represented as *quantities.* Each quantity has an amount, as in `A[TEMPERATURE(brick)]`, and a derivative, as in `D[TEMPERATURE(brick)]`. Amounts and derivatives are constrained by the inequality relations which hold among the different quantities (the *quantity space*).

Process definitions have five components: *individuals*, *preconditions*, *quantity conditions*, *relations*, and *influences.* The individuals specify what objects would be involved in the process if it were active, the preconditions and quantity conditions indicate when the process will be active, and the relations and influences specify what relations will hold while

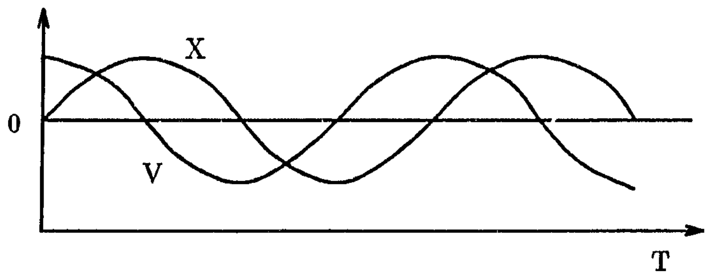


Figure 1.7: The behavior of a frictionless spring-block oscillator.

```
Process Simple-Liquid-Flow
    Individuals
        ?source,      a contained liquid
        ?destination, a contained liquid
        ?path,        a liquid path connected to ?source and ?destination
    Preconditions
        Aligned(?path)
    QuantityConditions
        Pressure(?source) > Pressure(?destination)
    Relations
        Q=(FlowRate(?lf-process-instance),
           [Pressure(?source) - Pressure(?destination)])
        FlowRate(?lf-process-instance) > Zero
    Influences
        Ctrans(Amount-Of(?source),Amount-Of(?destination),
               FlowRate(?lf-process-instance))
```

Figure 1.8: A QP Theory definition of the liquid flow process.

the process is active. Preconditions are changeable conditions that cannot be predicted in terms of change to continuous quantities. Quantity conditions correspond to changes in the relative relationship between two quantities and are predictable from the qualitative model. A typical QP theory definition for the water flow process is shown in Figure 1.8.

An additional type of QP theory description is the *individual view*. Individual views represent various configurations and states of objects that may be subject to dynamical conditions (e.g., a spring may be stretched, relaxed, or compressed). They are expressed in the same manner as processes (i.e., individuals, preconditions, quantity conditions, and relations), but differ in that they have no influences.

A central goal of this thesis is the use analogy for the development of theories about the physical world. Using the QP formalism, a "theory" consists of a set of process descriptions (implemented using `defProcess`), individual view descriptions (`defView`), entity descriptions (`defEntity`), and atomic facts.

Interpreting observations of the behavior of a physical system may be performed by finding a path through a total envisionment for the system which corresponds to the observations. Forbus (1986a) presents such a theory of *measurement interpretation* (ATMI) and describes the implementation. It takes an envisionment about the scenario and a set of time-ordered measurements. A model is able to provide an explanation for the observations if a path through the envisionment it produces can be found which corresponds to the measurements. For example, the plot of Figure 1.7 would be divided into temporal intervals of qualitatively equivalent measurements (e.g., velocity > 0 and increasing) and then compared to the eight-state description of Figure 1.6. When measurement interpretation fails, it must be because the measurement sampling was incomplete, the physical system is broken, or the qualitative domain model was incorrect. This work addresses the problem of incomplete and inconsistent domain models and does not address the sampling problem or system troubleshooting.

1.4 PHINEAS

The basic model of analogy described in Section 1.2 is instantiated in PHINEAS, a fully implemented program that offers qualitative explanations of time varying physical behaviors. It relies on analogical inference to hypothesize new theories, and uses qualitative simulation as a form of gedanken experiment to analyze their validity. It uses an iterative process consisting of four primary stages (Figure 1.9):

1. **Behavior match.** Behavioral similarity is used to initiate and guide analogy. A new observation triggers a search for previously understood experiences that exhibited analogous behavior. Abstractions of the observed situation and its behavior are used to provide indices into memory. The result of this stage is a candidate analogue and an initial set of correspondences between the two domains that serve to guide the mapping process.

2. **Theory generation.** The objective of the second stage is to produce a fully operational initial hypothesis about the current domain. This has two components. First, the models used to explain analogous aspects of the recalled experience are retrieved and analogically mapped into the current domain. This mapping is guided by the initial correspondences found in the behavioral comparison. Second, to operationalize the model, any unknown entities and properties it requires must be inferred from the domain theory or their existence must be postulated.

3. **Gedanken analysis.** The operational model is used to construct an explanation of the present observation. It is used to generate an envisionment of the scenario, which is then compared to the original observation. If the explanation is consistent and complete, then the explanation is considered successful. If the explanation is inconsistent or fails to provide complete coverage, then revision is required.

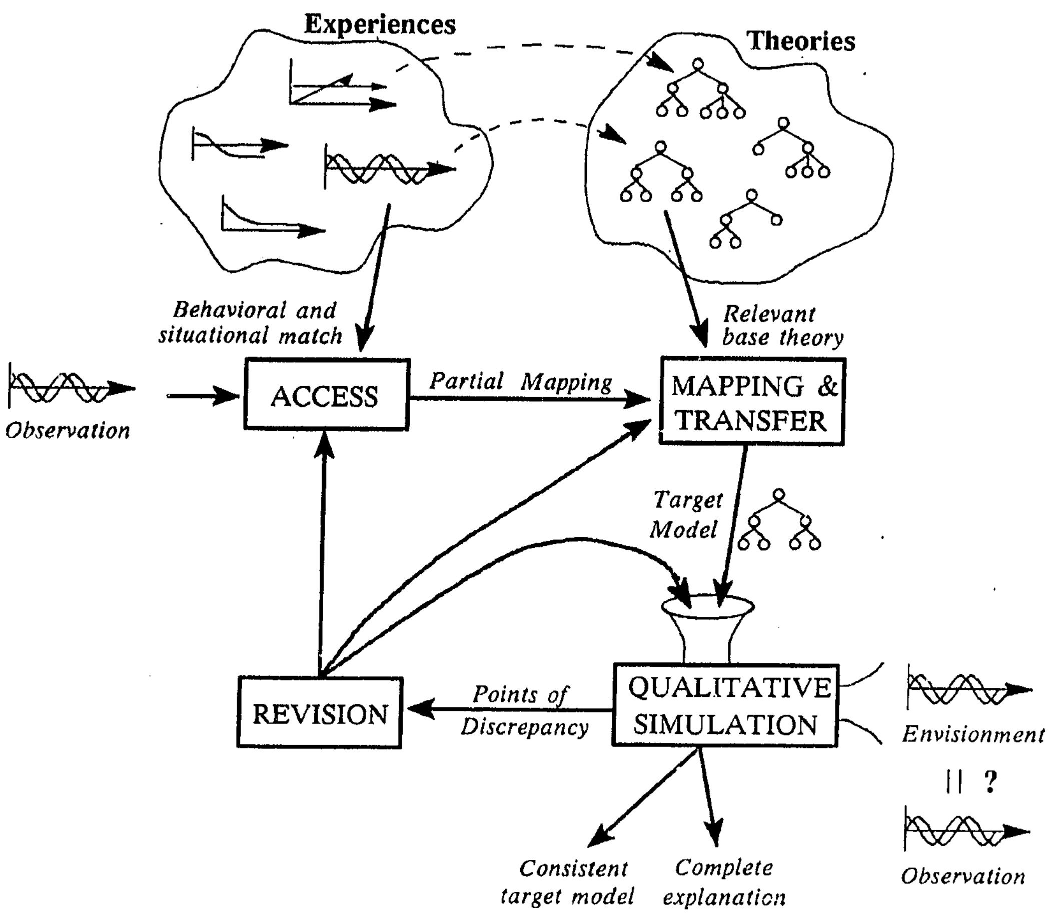


Figure 1.9: Functional decomposition of PHINEAS.

4. **Revision.** If an initial hypothesis fails, or an old hypothesis is inadequate for a new situation, an attempt is made to adapt it around points of inaccuracy. Revision relies on past experiences to guide the formation and selection of revision hypotheses. It considers behavior analogous to the current anomaly and considers how the current anomalous situation differs from prior situations that were consistently explained.

The revision module is only partially implemented. Every other aspect of PHINEAS is fully implemented and tested on over a half dozen examples.

PHINEAS demonstrates how analogical reasoning and learning can be used in scientific investigation or everyday physical interpretation. Good scientific investigation involves asking the right questions, questions which are motivated by highly plausible conjectures and potentially fruitful lines of inquiry. Analogy provides a frame of reference from which to draw expectations and motivate such questions. It can provide an initial foothold into a new domain, which may then be debugged and eventually stand on its own.

1.5 Reader's Guide

This thesis presents a detailed study of the role of analogy in explanation and learning, focusing on qualitative explanation of physical phenomena, and an implemented program, called PHINEAS which demonstrates the feasibility of the approach.

Chapter 2 describes a general framework for analogical processing, its components, and its roles in reasoning. It then reviews Gentner's structure-mapping theory and discusses the technical limitations that have been found with it. This is followed by a presentation of *contextual structure-mapping*, which addresses these limitations. The chapter includes a number of specific problems in analogy research which need addressing.

A prerequisite to general analogical processing is the ability to detect similarity by identifying correspondences. Chapter 3 describes the *structure-mapping engine* (SME), a general tool for performing various types of analogical matchings. The SME algorithm is briefly reviewed (see (Falkenhainer et al., 1987) for a complete description), followed by a detailed description of how it is configured to model the contextual structure-mapping process. The program is then analyzed from both analytical and empirical perspectives.

The next four chapters sequentially discuss how each step in the analogy process is performed. Each of these chapters is accompanied with working examples from PHINEAS demonstrating that phase of the process.

Chapter 4 describes the approach to access, which focuses on analogous behaviors to guide the formation of causal explanations.

Chapter 5 describes the mapping and transfer phase of the analogy process. It explains how analogy may be used to propose novel theories and the various technical difficulties that arise when knowledge is ported from one domain to another.

Chapter 6 discusses verification-based analogical learning and the importance of the evaluation and use phases of analogical processing. It reviews the process of measurement interpretation, which is in charge of determining if a proposed explanation accurately predicts the observed behavior. The chapter shows how VBAL serves three roles in analyzing proposed explanations: as confirmation, as discrimination, and as analysis of coverage.

Chapter 7 discusses how initial, flawed hypotheses may be revised into complete, consistent explanations. Unlike the other chapters in this thesis, which are fully implemented, this chapter describes a proposed approach to revision, which has yet to be fully implemented.

The PHINEAS implementation and how the different program modules actually combine to produce coherent behavior is described in more detail in Chapter 8. Chapter 9 demonstrates the operation of PHINEAS with a number of detailed examples. It provides a complete description of each example from start to finish, unlike the piecewise examples of the preceding chapters. Chapter 10 concludes with a summary, discussion of related work, and suggestions for future research.

Chapter 2

Similarity Comparisons and Analogical Processing

Explanation through physical analogies requires a robust model of the analogy process. This chapter establishes the requisite terminology, describes a general framework for analogical processing, what its components are, and the requirements of a comprehensive model. It begins by examining the different aspects of analogy and its various roles in reasoning. It then reviews Gentner's *Structure-mapping theory*, the primary source for the definition of similarity used in this thesis, and discusses some of its limitations. *Contextual structure-mapping*, an adaptation of Gentner's theory which addresses these limitations, is then presented. Finally, some general problems in analogy research are discussed.

2.1 The Analogy Process

Analogy is a mapping between one domain description (the *base*) and another (the *target*) that identifies a correspondence between their respective bodies of knowledge. Not all mappings represent analogies and part of the analogy research objective must be to define what mappings will be considered analogical. Each domain description consists of a set of objects and a set of related facts about those objects. Analogy then matches a base description with a target description of varying levels of completeness. When knowledge of both domains is equal, analogy indicates where the correspondences are and has no predictive power. Typically, the target will lack certain relations and *analogical inference* may occur; relations which hold in the base but are not currently known to hold in the target are mapped into the target domain if sanctioned by existing correspondences. Hence, analogy is most useful when familiar knowledge may be mapped to an unfamiliar situation, providing the necessary machinery to generate further inferences.

Analogy is a complex problem, and the appropriate decomposition is critical. Without a good decomposition, it is easy to contend with several semi-independent problems at once and become lost in the space of possible mechanisms. A number of researchers have recognized that analogy may be divided into at least the following three subprocesses (e.g., Clement, 1981; Gentner & Landers, 1985; Gentner, 1988; Kedar-Cabelli, 1985a; Hall, *to appear*; Falkenhainer, 1986):

1. **Access.** Given a current target situation, the first role of access is to retrieve from long-term memory a body of prior knowledge, the base, which is analogous or similar to the target. Second, it should attempt to garner out those features of the base that are pertinent to the analogy, by examining it in light of the current reasoning context. These two aspects may occur simultaneously. Here we will define the first aspect as *retrieval* and the second as *relevant feature selection.* Using the terminology of Clement (1981, 1982), both aspects of access are required for *spontaneous analogy* - analogies which occur without prompting. When the base and target are explicitly given, or when hints are provided (*provoked analogy*), the access stage may be reduced or eliminated entirely.

2. **Mapping.** The mapping stage consists of identifying a coherent set of analogical correspondences between the base and target (*matching* component), and possibly inferring additional base knowledge for the target (*carryover* component). Matching may involve extending previously existing correspondences. This match may also sanction a set of *candidate inferences* that identify additional knowledge possibly transferable to the target. The quality of a match may be a function of several factors: similarities, differences, and the amount and type of new knowledge or insight the analogy provides. Not all models distinguish between matching and inference, in which case mapping consists solely of carrying over a set of relevant base relations.

3. **Evaluation and Use.** The validity of the match and the inferences it sanctions must be examined through further reasoning processes, such as problem solving, consistency verification, or experimentation. Evaluation of validity may occur as a separate verification process, as a side-effect of use in a more global reasoning context, or as some combination of the two. Cooperative interplay may occur between mapping's process of creating inferences and attempts by the performance element to evaluate and use the inferences.

As a functional specification, rather than a modular decomposition, the above three components specify the general tasks that any treatment of analogy must address. Differences among alternate treatments typically correspond to where the modularity line between the tasks is drawn, how the tasks interact, and how each task is actually performed. It is generally agreed that the mapping stage is central to analogy, and most researchers have focused on this component.

Another important distinction for analogy is based on its intended use and thus what functionality a model of analogy must provide. Different uses may be distinguished by the types of inferences they draw. Take "$\approx\!\succ$" to denote the analogical inference operation, while a statement of analogical correspondence will be represented by the "$\sim$" operator. We may then describe analogical inference as being of the form

$$\begin{array}{l}\mathcal{P}_b \sim \mathcal{P}_t \wedge \\ \mathcal{P}_b \wedge \mathcal{Q}_b \wedge \\ \mathcal{P}_t \wedge \text{Unknown}[\mathcal{Q}_t] \\ \approx\!\succ \mathcal{Q}_t\end{array}$$

Given that sets of known facts, $\mathcal{P}_b$ and $\mathcal{P}_t$, are the same or similar ($\mathcal{P}_b \sim \mathcal{P}_t$), and that a set of facts, $\mathcal{Q}_b$, are known to hold while the status of a related set of facts $\mathcal{Q}_t$ is *currently* not known (but perhaps may be derived), infer that $\mathcal{Q}_t$ also holds (under whatever transformations allowed $\mathcal{P}_b$ and $\mathcal{P}_t$ to match).[1] The statement of similarity may be given (e.g., Burstein, 1983; Greiner, 1988), or autonomously derived. Other constraints are generally required, such as that $\mathcal{Q}_t$ is consistent and that $\mathcal{Q}_b$ be somehow correlated to $\mathcal{P}_b$. I will return to this issue in Section 2.4.4.

Borrowing from Indurkhya (1987), we may formally classify two types of analogical inference:

Definition 2.1 *A set of analogical inferences are* deductively sound *(d-sound) if every sentence being mapped is logically entailed by the target. The inferences are* inductively sound *(i-sound) if they are merely consistent with the target, rather than entailed by it.*[2]

Thus, some analogical inferences may be seen as knowledge which exists for the target domain, yet is not explicitly stated (i.e., search or inference would be required to retrieve it).

[1]This is a fairly standard definition, but differs from that used by Greiner (1988), whose learning model requires that the status of unknown hold over the deductive closure.

[2]Indurkhya (1987) uses *coherent* and its subset, *strongly coherent*, to refer to the entire analogical mapping (matches plus inferences). I adopt *d-sound* and *i-sound* because of the alternate meanings of "coherent" in the literature. This represents a deviation from (Falkenhainer, 1986), which used the term "coherent".

Other analogical inferences represent new knowledge – conjectured facts about the target domain. This is an important distinction for categorizing the different uses of analogy:

Similarity-Based Generalization. In analogy, one may use the similarity between two concepts found during the matching stage to form a single, generalized concept. This corresponds closely to many empirical (often called similarity-based) learning methods, some of which use pattern matching techniques to detect similarities in the structural representation of features (e.g., Hayes-Roth & McDermott, 1978; Michalski, 1980; Hoff et al., 1982; Skorstad et al., 1988).

Analogical Reasoning. Analogical reasoning is concerned with improving the effective use of existing knowledge. It involves using past experiences as heuristics to guide or accelerate current reasoning processes, as in using analogical inferences to recommend promising search paths. For example, in (Kling, 1971), analogy to a past problem suggests which clauses should be used in a new theorem proving task. Carbonell (1983a) uses traces of past reasoning steps to help guide search in planning tasks. Kedar-Cabelli (1985b) uses known examples of some concept to guide a proof that a new object is an instance of that concept. This classification also applies to most *case-based reasoning* systems, which draw on previous instances of similar situations for guidance (e.g. Kolodner et al., 1985; Bain, 1986). Here a strict definition of reasoning by analogy will be employed, in which analogy is used for guidance, and mapping and use combine to produce *d-sound* inferences.

Analogical Learning. Analogical learning is concerned with the acquisition of new knowledge. Given similarity between $\mathcal{P}_b$ and $\mathcal{P}_t$, assume the truth of $\mathcal{Q}_t$, where the status of $\mathcal{Q}_t$ cannot be determined in the current deductive closure. Roughly, if a target situation is encountered which is believed to be similar to some well understood base situation, perhaps additional knowledge available about the base may also hold for the target. Thus, analogical learning uses *i-sound* analogical inferences to posit new knowledge about the target domain (e.g., Burstein, 1983; Greiner, 1988).

These uses of analogy may be intermixed. The results of analogical reasoning processes may be stored, which is an analogy-independent process equivalent to analytical learning (EBL). In addition, one may use the results of analogical learning to form a more general concept description. For example, knowledge of the solar system could be used to learn about the Rutherford model of the atom. In turn, these could be used to form a general concept of central-force systems.

This thesis, in combining interpretation and theory formation paradigms is concerned with both analogical reasoning and analogical learning. Similarity to experience is used to suggest plausible explanations. If this represents an existing theory, then analogical reasoning has been performed. If it represents a novel (i.e., at the knowledge level (Dietterich, 1986)) theory, then analogical learning has been performed. However, most emphasis is placed on learning, since it is the more difficult of the two and must work extra hard to maximize the probability of correctness, a desirable trait in general.

From this taxonomy, we can see that the two primary functions of analogy are (1) establishing similarity and (2) sanctioning analogical inferences. While this is perhaps rather obvious, it is important to keep in mind. Few computational models of analogy explicitly provide *both* functions.

2.2 Structure Mapping Theory

The model of mapping used in this thesis is an adaptation of Gentner's (1980, 1983, 1988) *structure-mapping theory* (SMT) of analogy. This section review's Gentner's theory and introduces some of the theoretical distinctions that have been drawn from it. Problems encountered with the structure-mapping theory which helped to motivate the hybrid model used in this thesis are then discussed in the next section.

Structure-mapping describes the set of implicit constraints by which people process mappings of analogy and similarity. It is based on the intuition that analogies are about relations, rather than simple features. No matter what kind of knowledge (causal models, plans, stories, etc.), it is the structural properties (i.e., the interrelationships between the facts) that determine the content of an analogy. The target objects do not have to resemble their corresponding base objects, but are placed in correspondence due to corresponding roles in the common relational structure.

Structure-mapping theory states that predicates are mapped from the base to the target according to the following three mapping rules:

1. Discard isolated object descriptions (e.g., RED) unless they are involved in a larger relational structure.

2. Try to preserve relations between objects.

3. Use *systematicity* to determine which relations are mapped. Systematicity states a preference for systems of relations as exhibited by the existence of higher-order relations (e.g., CAUSE).

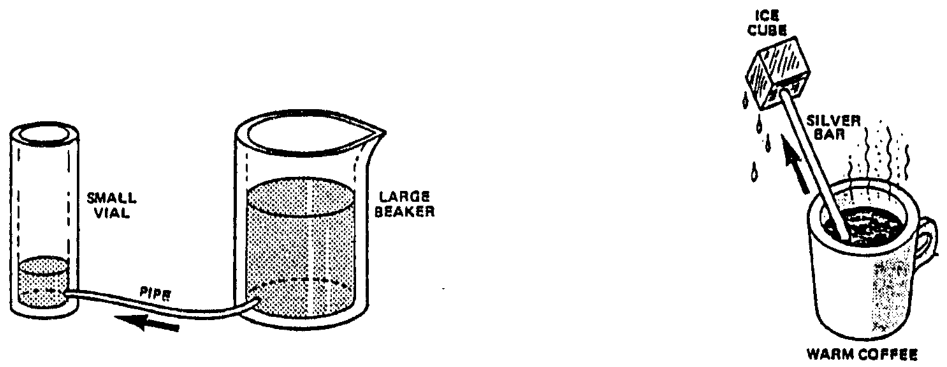


Figure 2.1: Two physical situations involving flow (adapted from Buckley, 1979).

This mapping must be

- *one-to-one:* No base item maps to two target items and no target item maps to two base items.
- *structurally consistent:* If relations B_i and T_j are placed in correspondence, then their arguments must exhaustively correspond as well.
- *identical:* Only identical relations are allowed to match.[3]

Gentner's theory is important in that it provides rules for analogical mapping that are based solely on the structural properties of domain descriptions, rather than on their content. This provides a general mapping component applicable to a variety of analogy tasks, including concept learning, problem solving, and many types of metaphor interpretation. Furthermore, the systematicity principle captures the tacit preference for deep, well-supported knowledge in analogy rather than shallow associations. Analogy implies that a relational system from one situation may be applied to another, independent of superficial similarity or dissimilarity.

For example, consider the water flow – heat-flow analogy shown in Figure 2.1 (from Forbus & Gentner, 1983). To process this analogy according to the rules of SMT, one must

- Set up the object correspondences (e.g., `beaker` goes to `coffee`).

[3]A revision to allow non-identical matching for functions appearing as arguments of identical relations appears in (Gentner, 1988).

- Discard isolated object attributes (e.g., `(Liquid water)`).
- Map the water flow relations into their corresponding heat flow relations.
- Observe systematicity by focusing on relations belonging to a systematic relational structure.

2.3 Limitations of Structure-mapping Theory

Gentner's model has many attractive features. However, several limitations have been found in attempting to apply it to a complex learning task. First, structure-mapping theory requires that all relations are mapped identically between base and target situations. This requirement is important in that it establishes semantic correspondence between the structures being matched. However, while examples of analogies satisfying the identicality restriction are easily constructed, it is too strong to achieve generality.[4] As Burstein (1983) has pointed out, it is difficult to maintain relational identicality when mapping between physically realizable situations and purely abstract ones. For example, relating how a box can "contain" things to the way a computer variable can "contain" a value. They share similar properties, but generally are distinguished, as in `INSIDE` and `INSIDE-VAR`.

A second difficulty with SMT is the assertion that systematicity is the sole selection criterion for deciding among possible interpretations during mapping. One problem is that the largest common relational system may not have anything to do with the intended goals of the analogy. If we are interested in learning about heat capacity by analogy to container volume in the liquids domain, we don't want the potentially larger relational match between heat flow and liquid flow to be the analogy. Much of the selection process may be performed prior to mapping by examining the base knowledge in the current problem solving context, as sanctioned by SMT and demonstrated by (Greiner, 1986; Falkenhainer, 1986; Kedar-Cabelli, 1988). Thus, if we were interested in learning about heat capacity, only knowledge about fluid capacity need be considered during mapping. However, there are two factors that work to defeat systematicity as a sole selection criterion: *ambiguity* and *lack of knowledge.*

Mapping may often be a useful guide when the exact relevance of particular knowledge to the analogy is *a priori* ambiguous. Alternatively, information irrelevant to the current goals yet about the goal relevant knowledge may be crucial to disambiguate the match.

[4]Gentner recognizes this limitation and has proposed iterative rerepresentation of decomposable predicates as one possible solution (Gentner, 1988).

For example, the fact that `Pressure` and `Temperature` are both intensive quantities (i.e., point measurements) may be crucial when considering whether `Temperature` should map to `Pressure` or to `Amount`. However, if potentially irrelevant relations are allowed into the base and target representations, it is possible for them to dominate, producing a "best" match that doesn't include the inferential structure wanted in the first place. In such situations, there is no way to force the desired candidate inference out of the mapping procedure if there is no way to contextually guide it. This was empirically observed during the development of PHINEAS.

The other factor, lack of knowledge, is unavoidable in learning situations. As the amount of knowledge about the target decreases, the available systematic match diminishes. At some point of ignorance, systematicity plays no decisive role. Hence, some other factor must influence the selection, such as the relevance of inferences sanctioned. Finally, it seems reasonable to assume that some relations are more salient than others. Requiring that these relations are always of higher order than less salient relations imposes strong constraints on the representation. Gentner (1987) indicates that analogy occurring in contexts other than problem solving is a reason to leave contextual factors out of the mapping component. It appears that contextual factors, when present, cannot be external to the matcher and are perhaps better viewed as optional influences on the mapping component.

Other problems are more subtle, and are not limited to learning task. These problems were found by using SME (Falkenhainer et al., 1986, 1987), a flexible analogical matching system that may be configured to obey the rules of structure-mapping theory.[5] It appears that purely structural, content independent matching rules are insufficient to prevent anomalous mappings. There are two types of anomalous mappings that arise: *spurious matches* and *structure rearrangement.*

The spurious match problem may be understood by reconsidering the water flow – heat flow analogy with the ice cube immersed in the coffee rather than attached to a metal bar. Suppose the causal description of fluid flow states necessary conditions for the participatory objects *source* (beaker), *destination* (vial), and *path* (pipe), with those describing the path (pipe) being:

$$\text{Physical-Obj(pipe)} \wedge \text{Fluid-Connection(pipe,beaker,vial)} \wedge \neg\text{Blocked(pipe)}$$

If the only instances of any of these predicates in the given target description are {`Physical-Obj(coffee)`, `Physical-Obj(ice-cube)`, `Physical-Obj(coffee-cup)`}, a purely structural match will map `beaker` to `coffee` and `vial` to `ice-cube`, as before, but also

[5]SME is described in Chapter 3. When configured for structure-mapping theory, I will refer to it as SME_{SMT}.

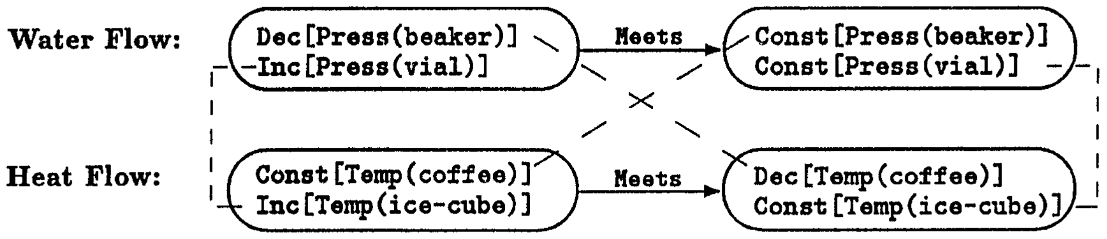


Figure 2.2: The structure rearrangement problem in behavioral descriptions of liquid flow and "heat flow".

pipe to coffee-cup. Clearly, immersion or physical contact is the path in this heat flow scenario, not the coffee cup. This requires the use of knowledge about the structures being manipulated, rather than a purely structural match. However, this is not a problem unique to SMT. It arises when one attempts to draw conclusions based purely on pattern matching, without first inspecting the results in light of a surrounding reasoning task.

The structure rearrangement problem arises from seeking the maximal structural match. In any structure matching paradigm, it is necessary to allow substructures to match, which occasionally results in some of the higher-order structure being dropped in order to perform the match. Sometimes, the best match can be achieved by ignoring higher-order constraints, moving pieces of structure around, and violating the intended semantics of the representation. For example, consider the two-state behavioral descriptions of water flow and "heat flow" shown in Figure 2.2. The heat flow description has been altered to demonstrate the anomaly: the coffee temperature is constant in the first state and decreasing in the second state. When the rules of SMT are applied, every item matches except the temporal Meets relationship. That is, Inc[Press(vial)] maps to Inc[Temp(ice-cube)] while Dec[Press(beaker)] maps to Dec[Temp(coffee)]. All four temporally-scoped water flow relations have a structural correspondent in the heat flow situation. Time has been rearranged.

Not all structure rearrangement is bad. For example, if temporal ordering is irrelevant to the causal structure of a story, then temporal ordering should not constrain matching. However, rearranging many other knowledge structures is clearly inappropriate, such as decomposed objects, temporal states, and theories. Without inspecting the content of the structures being manipulated, there is no way to make this decision. From a psychological perspective, some forms of structure rearrangement confusions are probably witnessed in humans. Some rearrangements are clearly invalid, others are arguably non-optimal. The key point is that if knowledge is available to spot these confusions, then it should be applied.

2.4 Contextual Structure-Mapping

Gentner's structure-mapping theory takes the position that only the structural properties of the representations should be examined during mapping. In applying analogy to a complex learning task, I have had to adapt this view in several important ways. Specifically, I propose that information about the structures being manipulated should be used to maintain consistency of the mapping, to reason about how items should be placed in correspondence, and to influence selection among possibly many alternative mappings. I call this approach *contextual structure-mapping*, since it uses knowledge about the representations being manipulated and the context in which they are being used as an aid in the mapping process. It is a knowledge-intensive adaptation of Gentner's theory.

Mapping is concerned with two important problems:

- *Correspondence Problem:* What objects and relations may be placed in correspondence?

- *Selection Problem:* What factors should decide how the "best" mapping is chosen?

These two problems distinguish between what is *allowed* to match and what *the* set of correspondences will be. They have many facets, as discussed below. The central difficulty of mapping is restricting the enormous space of possibilities to a small, plausible subset. Constraints on the mapping process which determine admissibility and selection generally fall into three classes (Hall, *to appear*): *structural constraints* preserve the relational structure of the descriptions, *semantic similarity* restricts pairwise matching of predicates according to their similarity, and *contextual relevance* motivates the mapping towards solutions relevant to the needs of the performance element. The constraints and influences used in contextual structure-mapping are:

1. *One-to-one:* SMT's one-to-one restriction is adopted in its exact form. The mapping must not assign the same base item to multiple target items nor any target item to multiple base items.

2. *Structurally grounded:* If base predicate B_i is mapped onto target predicate T_j, then all of B_i's arguments must also map onto the arguments of T_j. In the case of predicates

B_i and T_i forming sets of relations, called *relational groups* (e.g., AND, SET), structural grounding is weakened to require that at least one one of their arguments be paired.

3. *Domain-specific:* The mapping must not violate general representational or domain-specific constraints that apply to pairing specific base and target descriptions.

4. *Semantic similarity:* Predicates are allowed to match if they are (1) identical, (2) functionally analogous, or (3) have a common generalization. These are defined below.

5. *Loyal:* If the mapping is elaborating an existing mapping, then it must respect the correspondences of the existing mapping.

6. *Selection:* Both systematicity and contextual relevance determine which relations are mapped.

2.4.1 Structural Constraints

The first two constraints together enforce SMT's *structural consistency* requirement (see also Kling, 1971; Winston, 1980; Rumelhart & Norman, 1981; Burstein, 1983; Carbonell, 1983a; Indurkhya, 1987). However, it deviates from the standard definition in one important respect. The requirement that the mapping be structurally grounded does not cross the boundaries of a *relational group*. A relational group is distinguished as an unordered collection of relational structures that may be collectively referred to as a unit. They correspond to the abstract notion of a "set" and are associated to predicates taking any number of arguments. For example, a set of relations joined by the predicate AND defines a relational group. Other examples include the axioms of a theory, a decomposable compound object, or the relations holding over an interval of time. Intuitively, we would like to say that two groups correspond without requiring that their contents are exhaustively mapped.

If base and target propositions each contain a group as an argument, the propositions should not be prevented from matching if the groups' members cannot be exhaustively paired. For example, the set of relations

$$\begin{array}{ll} \text{B:} & \texttt{Implies[And(}P_1,P_2,P_3\texttt{), }P_4\texttt{]} \\ \text{T:} & \texttt{Implies[And(}P'_1,P'_2\texttt{), }P'_4\texttt{]} \end{array} \quad (1)$$

should match better than the set of relations

$$\begin{array}{ll} \text{B:} & \texttt{Implies[And(}P_1,P_2,P_3\texttt{), }P_4\texttt{]} \\ \text{T}': & P'_1,\ P'_2,\ P'_4 \end{array} \quad (2)$$

The original model of structural consistency would score (1) and (2) equally, since the `Implies` relations of (1) would not be allowed to match. This is a particularly important consideration when matching sequential, state-based descriptions (e.g., the behavior of a system through time). The set of relations describing a pair of states often do not exhaustively match or are of different cardinality. Yet, higher-order relations over states, such as temporal orderings, are vital and must appear in the mapping.

As discussed in the previous section, purely structural constraints are insufficient to prevent structure rearrangement. Just as a purely syntactic predicate calculus is able to recognize the mutual inconsistency of `P(x)` and `¬P(x)` but not the mutual inconsistency of `Solid(x)` and `Gas(x)`, a purely structural account of mapping will fail to recognize some inconsistencies in the mapping. Thus, additional constraints are supplied to capture important general representational or domain-specific knowledge about the structures being manipulated.

At the current time, only very general, representational constraints are defined. These prevent structure rearrangement across relational groups, such as mixing and matching elements of compound objects, theories, or temporal states. For example, the following rule preserves temporal relationships:

$$\begin{aligned}
&\mathtt{MH}(B_i,T_i) \wedge \mathtt{Temporally\text{-}Scoped}(B_i) \wedge \mathtt{Temporally\text{-}Scoped}(T_i) \wedge \\
&\mathtt{MH}(B_j,T_j) \wedge \mathtt{Temporally\text{-}Scoped}(B_j) \wedge \mathtt{Temporally\text{-}Scoped}(T_j) \wedge \\
&\{\ [\mathtt{EqualTime}(B_i,B_j) \wedge \mathtt{DisjointTime}(T_i,T_j)] \vee \\
&\quad [\mathtt{DisjointTime}(B_i,B_j) \wedge \mathtt{EqualTime}(T_i,T_j)]\ \} \\
&\qquad\quad \Rightarrow \mathtt{Conflicting}[\mathtt{MH}(B_i,T_i),\ \mathtt{MH}(B_j,T_j)]
\end{aligned}$$

where `Conflicting` indicates two mutually inconsistent pairings (see Section 3.2).

2.4.2 Semantic Similarity

Semantic similarity is a crucial component of the correspondence problem, during both mapping and transfer. During mapping, it is important to limit the possibilities and ensure that a semantic correspondence is being made. During transfer, it is important to guide search for correspondents potentially absent from the given target description and to guide creation of a new predicate if one is needed. Semantic similarity enables the adaptation of knowledge to analogous situations without requiring that the knowledge be expressed exactly the same in all situations.

Most accounts of analogy enforce semantic similarity by testing for predicate identicality (e.g., Carbonell, 1981; Gentner, 1983). As noted earlier, identicality is too restrictive to be of use for many reasoning situations. In across-domain analogies, it requires that the important relational structure has been properly decomposed, as in `Hotter-than(x,y)` being represented as Greater-than[temperature(x), temperature(y)]. Additionally, there are many near or within domain relationships that are difficult, if not impossible, to reduce to identicality. The concepts of objects Inside-of containers and solids Dissolved-in liquids are very different (Burstein, 1985). Yet, it is clear these relations are candidates for analogical mapping.

In models that allow non-identical relations to match, the preferred solution is to consult a hierarchy of predicate types and allow relations with a common ancestor to match (Winston, 1980; Burstein, 1983; Wellsch & Jones, 1986). The match score may then be inversely proportional to how distant the relations are in an ISA hierarchy (Winston, 1980; Wellsch & Jones, 1986). Take for example Burstein's CARL program. When a base relation is carried into the target domain, CARL moves up an ISA hierarchy seeking a relation with argument type restrictions loose enough that the target objects satisfy those constraints. CARL applies the same process to action predicates (e.g., Trans is a parent action of Ptrans and Mtrans).

However, in solving the identicality problem this approach raises a number of additional problems. There is the potential to go too high in the network (e.g., to Relation), and ultimately place every relation in correspondence with every other relation. This may be prevented by using forests of hierarchies, ensuring that semantically close relations are mutually reachable, while ensuring that each tree doesn't go too high so as to achieve meaningless generality. Yet, this presents a dilemma. Besides seeming rather ad hoc, it places a strong burden on the user to *a priori* know which relations should belong to common clusters and which should not. Some approaches adopt a halfway point, by forming general classes of predicates such as action, relation, plan, goal, etc. and restricting mapping within these boundaries (Burstein, 1985). The generalization tree models further suffer from being single parent hierarchies. This is easily fixed using a more sophisticated ISA lattice, but then we must decide which parental branch to follow and what the consequence of multiple intersections will be.

These approaches fail to recognize an important point: *mappability is context sensitive.* When a predicate instance is used in some context (e.g., in a chain of inference), it is used because it denotes certain characteristics about the world deemed important for that context. Thus, propositions should be semantically similar *with respect to* their functional role in the surrounding structure. Movement within the generalization hierarchy is dependent upon the characteristics the predicate was intended to possess. When moving up the hierarchy, a point may be reached where those properties are no longer present. Furthermore, this implies that mappable relations need not always share a common ancestor (except perhaps at the uppermost *relation* node).

There are numerous examples of contextually sensitive similarity. The property `Cylindrical` of one cup should map to the property `Has-Handle` of another cup if the role of these expressions is to support liftability. As another example, whether or not we use `Contained-in`, `Dissolved-in`, or `Absorbed-in` depends upon the surrounding context. Each has a privileged set of inferences. You can squeeze out liquid absorbed by a solid, but you can't squeeze salt out of the water it has dissolved in (at least not in the same way).

Therefore, if two predicates are not identical, they may still be considered semantically similar and eligible for mapping if they are *functionally analogous*:

Definition 2.2 (Functionally analogous) *Two expressions are considered* functionally analogous *and may match if they provide the same inferential support in the context of the structures being matched.*

There are several ways to determine the inferential support an expression is providing. When an expression's role is explicit in the structure, it is particularly simple. For example, expressions `P` and `R` will be placed in correspondence when matching `Implies(P,Q)` with `Implies(R,Q)`, since their respective roles are to deductively support `Q`. Being structurally explicit is the only method for determining whether two expressions are functionally analogous during mapping.

Another method addresses the problem of compiled knowledge which is absent from the explicit structure. This method may be used during transfer when seeking information about unmatched expressions. AI systems tend to use compiled knowledge, in which intermediate reasoning steps are absent to promote efficiency of use. Indeed, this is the central goal of explanation-based generalization (Mitchell et al., 1986; DeJong & Mooney, 1986). In PHINEAS, this is a common occurrence, owing to its use of QP theory syntax. A process definition consists of a set of antecedents indicating when the process will be active and a set of effects which hold when the process is active. While effective for reasoning, a great deal of information is compiled away by the model builder. No direct links between antecedents and effects are available. The actual "theory" about the domain being represented, such as why each antecedent is present in the process description, is absent. To address this problem, PHINEAS uses an augmented process description. A *cache* slot is added to link necessary prerequisites of the effects relations to the antecedents that satisfy those prerequisites. For example the effect relation

`Ctrans`(*source-stuff*, *destination-stuff*, *rate*)

indicates a continuous transfer of some "stuff" (e.g., fluid or energy) from a source to a destination. By continuity of motion, a necessary prerequisite for `Ctrans` is the existence of some physical path between the source and destination. Therefore, this prerequisite information, and how it is satisfied, should be present in the cache slot of any process that uses `Ctrans`. In the dissolving process, this appears as:

```
(Satisfies (Immersed-in ?solute ?solution)
           (Prereq (Ctrans (amount-of ?solute) (concentration ?solution) ?rate)
                   (Physical-Path ?solute ?solution ?path)
                   (Motion-Continuity)))
```

When investigating the role of `Immersed-in`, a post-mapping process may consult the dissolving process' cache slot to determine that a relation not supporting `Physical-Path` (or the analogue of `Physical-Path` if this is part of the mapping) cannot be used as an analogue for `Immersed-in`. Importantly, this information indicates that it is the physical connection *aspect* of `Immersed-in` that is important (e.g., as opposed to preventing exposure to the atmosphere). This will be described more during the discussion on transfer (Chapter 5).

The definition of *functionally analogous* is a general statement of the specific problem derivational analogy (Carbonell, 1983a) attempted to address for planning situations. Specifically, a fundamental component of analogy is knowing why the relations being considered for mapping are there. A good analogizer is able to recognize alternate ways to achieve equivalent functionality. In problem solving, adapting a prior solution to a new problem instance requires just that, adaptation of the prior solution. This task is greatly simplified if we know why decisions were made the way they were, so that we can satisfy the intent of the decision with out necessarily adhering to the same decision.

In the absence of such background knowledge, proximity within a generalization hierarchy is a good heuristic to use. A portion of the hierarchy used in PHINEAS appears in Figure 2.3 (see Appendix E for the complete set). There is a good chance that the characteristics desired of a relation will also be present in its nearest neighbors in the generalization hierarchy. The farther the relations are in the hierarchy, the less likely this will be true. Thus, the strength of this type of match should diminish in proportion to the distance within the generalization hierarchy between the predicates matched. I call this the *minimal ascension principle.*[6] Two constraints are used to limit improper generalizations. First, only structurally motivated pairings are made. Specifically, two predicates may only match via minimal ascension if their corresponding parent relations have already been paired. Second, the ISA hierarchy is assumed to be shallow and highly disconnected. Thus, the hierarchy in Figure 2.3 indicates that minimal ascension is not allowed to match a quantity to a type of physical proximity.

[6]Term suggested by Dedre Gentner. The same type of principle has long been used in structured induction, and appears as the *climbing the generalization tree* rule in (Michalski, 1983).

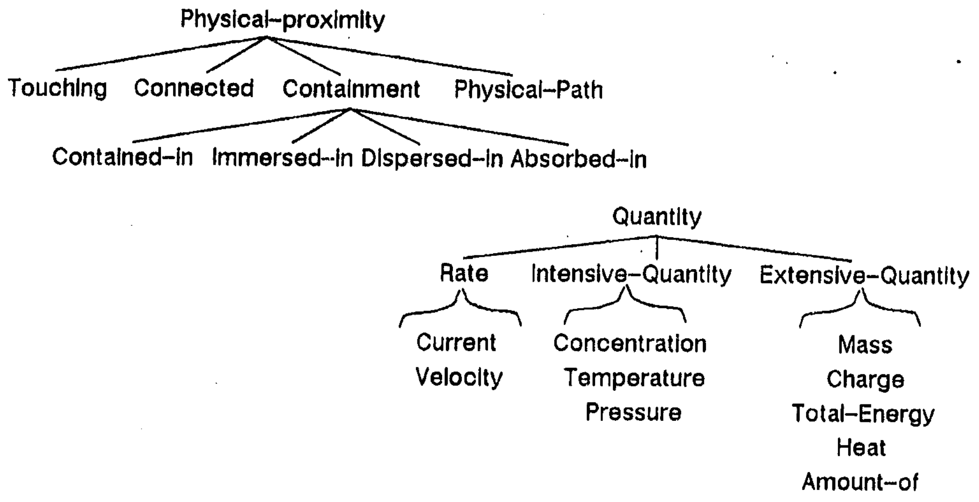


Figure 2.3: A portion of the predicate *isa* hierarchy used in Phineas.

2.4.3 Selection: Weighing systematicity and relevance

- *Selection Problem:* What factors should decide how the "best" mapping is chosen?

The importance of context and problem solving goals in analogical processing is a recurring theme (Burstein, 1983; Carbonell, 1983a; Greiner, 1988; Holyoak, 1984; Kedar-Cabelli, 1985b). It is generally agreed that in problem solving situations, the current reasoning goals have a strong influence over what base information is retrieved and how the results of analogy are ultimately evaluated and used. Debate centers around contextual effects on mapping. At one extreme, Gentner (1988) proposes that problem solving influences the processes preceding and following mapping, but play no role in the mapping process which is guided solely by systematicity. However, from the discussion of Section 2.3, it appears that this is insufficient to guide the mapping process. At the other extreme, Holyoak (1984) maintains that problem solving relevance alone drives the entire mapping process. This model is only applicable to analogical problem solving and ignores other uses of analogy. Other approaches are more agnostic on the issue, since neither systematicity nor goal related relevance appear explicitly in the mapping process. Relevant base information is selected during access, with mapping consisting of its reinstantiation and potential adaptation for the target case (Burstein, 1983; Carbonell, 1983a; Kedar-Cabelli, 1985b; Greiner, 1988).[7]

[7]Greiner (1988) does consider a host of alternative interpretations and could be said to be following a form of systematicity. During target instantiation of a base abstraction, Greiner's *fewest conjectures*

Since an explicit match is never formed and mapping is heavily constrained (i.e., through teacher-supplied hints or completely unambiguous mappings), choosing among alternative interpretations is avoided.

From pragmatic theories, as well as problems experienced with a purely structural model, we can see that relevance can be an important influence on analogical mapping. However, people can evaluate analogies in context-free settings (Gentner & Landers, 1985; Rattermann & Gentner, 1987; Clement & Gentner, 1988), supporting the view that systematicity plays a significant role in people's selection criteria. People are able to process an analogy without a goal in mind. They are able to "see" similarity without necessarily being in the middle of solving some problem. From a purely computational perspective, mapping should be able to spot similarity in the absence of goals since the various uses of analogy and similarity should be achieved by a single computational model of mapping. However, it should also be able to adapt to the needs of a surrounding problem solving context if one exists.

Thus, a hybrid approach, influenced by both systematicity and contextual relevance, is used. In the absence of problem solving goals, systematicity serves as the sole criterion for selection. When problem solving goals are present, interpretations containing the relevant base information are preferred.[8] An important feature is that contextual information is not required; its presence serves to influence the normal operation of mapping, not replace it.

2.4.4 Candidate Inferences and the Validity Problem

One of the important functions of mapping is to identify base information plausibly inferable for the target situation. This requires consideration of validity, the central problem in using analogically derived knowledge.

- *Validity Problem:* What is the basis for having confidence in the analogically proposed inferences?

The desire for validity effects each stage of analogical processing. Relations that are predictably useful are sought during access. Mapping should only propose plausible inferences that follow from the set of known correspondences. Finally, analogy is an approximate process and there is a tradeoff between generality of the process design and guaranteed correctness of what it produces. Analogical reasoning systems, by definition, only retain

heuristic (H_{FC}) prefers analogies that require adding the fewest new conjectures (i.e., candidate inferences). This is equivalent to preferring the maximal match to the relational system called the base abstraction.

[8]Recently, Holyoak and Thagard (1988a) have independently proposed a similar hybrid model of combined systematicity and pragmatic influence.

those inferences or procedures that were *a priori* derivable from the target domain theory (e.g., Carbonell, 1983a; Kedar-Cabelli, 1985b). Analogy in this context is used to improve performance. Analogical learning systems must be more cautious and must be able to revise their beliefs when inferences are shown to be invalid.

It is important to identify methods which will increase the likelihood that the analogy process produces only valid inferences. All efforts to maximize validity may be divided into four general categories:

1. *Representation.* Develop specialized knowledge structures such that only certain valid (or highly likely) analogies are allowed (e.g., the *determinations* of Davies & Russell (1987)).

2. *Access.* Constrain the access mechanism so that only highly likely analogues will be retrieved (e.g., Kedar-Cabelli's (1985) purpose-directed mechanism).

3. *Inference production.* Use mapping procedures which focus on validity preserving features (e.g., Gentner's (1983) *systematicity* principle).

4. *Post-mapping.* Analyze the established analogy to ensure maximum coherency (e.g., Clement's (1986) bridging analogies).

Representational approaches have the nice property that they make the basis for drawing analogical inferences explicit. For example, Russell and Davies (Russell, 1987; Davies & Russell 1987) have recently proposed the use of *determinations* to ensure validity in analogical inference.[9] Determinations are specifications of functional dependence. The determination $P \succ Q$ indicates that P *functionally determines* the value of Q since there is a unique value for Q given P. This declares a relationship between P and Q that is too weak to enable conclusions on its own, but enables a valid conclusion about a target case once a base instance has been encountered.

However, there are a number of problems with this approach. First, it defines away the majority of what is intuitively called analogical. For example, given that $f(x,y) = z$, and a base situation showing $f(4,9) = 7$, 7 may be recalled when a new problem situation asks for the value of $f(4,9)$. Few analogies can be described in these terms. Second, it doesn't require the presence of the base description during the analogical act. The target query is derivable from the domain theory once the base is given, eliminating interplay between base and target to influence inferencing. Analogy involves comparison, measuring *degrees* of similarity, not identicality over atomic features in a functional dependency.

[9]See (Baker et al., 1988; Clark, 1988) for other approaches to the representation problem.

Structural approaches attempt a more content-independent approach:

Arguments from models involve those analogies which can be used to predict the *occurrence* of certain properties or events, and hence the relevant relations are causal, at least in the sense of implying a tendency to co-occur. (Hesse, 1966, pg. 78)

Hesse argues, as does Gentner (1980, 1983), that a necessary (but not sufficient) condition for validity is that analogy involve the mapping of facts which are "causally" related, rather than miscellaneous features. Thus, only if the set of facts $\mathcal{P}_b$ are known to cause (or have a tendency to co-occur with) the set $\mathcal{Q}_b$ do we have any basis for believing that $\mathcal{Q}_t$ will also hold given $\mathcal{P}_t$. Gentner generalizes this criteria by using the framework of "systematic structure" rather than "causality". This creates a general definition which focuses on interrelated facts and views this interrelation as supporting predictivity. Thus, a match over some facts sanctions inferences for remaining facts in the larger relational system they appear in. For example, consider an example from (Davies & Russell, 1987): A red robin is observed and found to have long legs and a scratched beak. Since we would possess a relational system relating a robin's various body proportions to its being a robin, but no such system relating a scratched beak to its being a robin, only the former would be postulated for a second robin.

Russell and Davies correctly observe that analogy research has not directly addressed the validity issue. In the case of structural approaches, structure can easily be added to any description and not all structure supports inferencing. Since SMT does not allow one to examine the structures being manipulated, all matches and resulting inferences look the same, given equal structure topology. For example, I could add a description of how the first red robin and the scratch on its beak co-occur in time. From a purely structural vantage point, there is now no reason not to infer that a second robin should have a scratched beak. Thus, it would appear that sharing common relational structure is not sufficient to constrain inference production to producing inferences that have some grounds for validity.

More work is needed on representational and structural approaches. In this thesis, the central basis for most of the inferences drawn rests on the belief that similar behavior indicates a strong likelyhood of similar causes. Since PHINEAS is aimed at this one type of explanation, this assumption is built in rather than explicitly available to the program. This work focuses on post-mapping methods to increase confidence in proposed analogies (i.e., verification-based analogical learning).

- *Consistency Restrictiveness:* Two-valued consistency is overly restrictive as a basis for analogical processing.

One dominate point of convergence in analogy research has been the centrality of consistency in guiding and evaluating analogy production (Indurkhya, 1987; Greiner, 1988;

Kedar-Cabelli, 1985a; Hall, *to appear*). Many go beyond consistency to require that analogy produces d-sound analogical inferences (e.g., Carbonell, 1983a; Kedar-Cabelli, 1985b; Davies & Russell, 1987).

Consistency is an important component of all forms of reasoning. Yet, we must be careful not to afford it too much import. Limiting analogy to strict consistency requires that analogy be a monotonic process. However, analogy often causes the questioning of beliefs and may lead to a complete change in world view. Thus, a weaker form of consistency is needed. One which takes into account the cost of overthrowing or revising prior beliefs for the benefits of a more coherent belief state. Thagard's (1988) work on explanatory coherence may be viewed as a step in that direction.

While this thesis does not offer a general solution to this problem, it is important to keep in mind so as to avoid theories that crucially depend upon strict consistency.

2.4.5 Additional Correspondence Subproblems

There are two identifiable subproblems to the correspondence problem not yet discussed. The first has several instantiations:

- *Non-Monotonic Binding Problem:* Define a *binding* to be any pairwise correspondence between atomic units. Thus, a binding may be an analogical correspondence between two objects, or the binding of a variable during unification. The non-monotonic (N-M) binding problem occurs when an influx of new information is allowed to overturn an existing binding set, either due to internal inconsistency or in favor of a superior binding.

The following two observations are partly responsible for the N-M binding problem:

1. *Not all candidate inferences are real inferences.* In realistic memories, it is unlikely that an analogizer will be operating on every item which comprises the representation of the base or target domains. Instead, a subset of the base or target descriptions are used. Since a candidate inference is with respect to the subset of the base or target being processed, it might not be an inference at all if a different aspect of the target were fetched. Alternatively, a candidate inference may represent a place where the analogizer failed to detect semantic similarity with an available target description. If there is more than one way to say something, retrieving a different representation may make the similarity more discernible. Thus, the idea that base and target knowledge can be fully prepackaged for the mapping component breaks down in general. There may be a need to reprobe memory to seek further information.

2. *Not all matches are real matches.* Suppose the base description contains $R_1(b_i)$, $R_2(b_i)$, and $R_3(b_i)$. If the target description has no instances of R_1 or R_2, and the only instance of R_3 is $R_3(t_i)$ then, if consistent, b_i will be placed in correspondence with t_i. This isn't necessarily a good match to make. Other target objects may exist that provide a better fit, but were not mentioned in the original target description. Alternatively, conjecturing the existence of an unknown target object may be preferable. What if $R_1(t_i)$, a possible resulting inference, is known to be false? Rather than reject the analogy, it may be preferable to question if there was sufficient grounds to conjecture t_i as the correspondent of b_i.

The N-M binding problem appears as the spurious match problem mentioned in Section 2.3 and discussed above as "not all matches are real matches". It isn't limited to weak matches however. It may occur due to ambiguity over which of two good matches to choose. It may also occur when a previously unnoticed target item is detected and found to be a better correspondent for some base item than its current target correspondent. For example, consider the analogy between a spring-block oscillator and an LC circuit. In the spring-block system, position is easily measurable while force is not. In the LC circuit, voltage is easily measurable while charge is not. Thus, an analogy focusing on observables would place the block's position quantity in correspondence with the circuit's voltage quantity. However, a more thorough analysis of the two systems would show that force should map to voltage while position should map to charge.

Second, it affects variable bindings during abductive inference. Abductive inference is required when a set of unmatched base objects are carried into the target and target correspondents sought. Suppose the goal is to seek an object satisfying the conjunction

$$P(?x) \wedge Q(?x) \wedge R(?x)$$

Traditional, sequential backchaining on each conjunctive subgoal fails. Suppose there are two objects, a and b, where $P(a)$ is the only knowledge about a, and $Q(b) \wedge R(b)$ is the only knowledge about b. Sequential subgoaling will be unable to propose b as the best binding for $?x$, since the candidate binding set is $\{a\}$ after showing $P(a)$. Seeking a set of unknown base objects can compound the problem. Due to potential interdependencies between the unknown objects, what may be a best match for one unknown may not be best for matching the other unknowns.

The N-M binding problem has struck others before me. It is the implicit motivation behind Kedar-Cabelli's (1988, pg. 131-132) *near miss* assumption. She mentions that her method of selecting the first available analogue and working to adapt it has the potential to miss the best set of correspondences. This appears to be an instance of the N-M binding problem.

I claim that approaches attempting to solve a series of conjuncts, rather than seeking global similarity, will suffer from the N-M binding problem. A further implication of the problem is that one cannot always assume that an existing mapping may be consistently extended. This conflicts with the common view that the role of mapping is to consistently extend an existing mapping (e.g., as in extending a partial mapping produced during access) (Burstein, 1985; Kedar-Cabelli, 1985a; Hall, *to appear*).

The problem is compounded by the use of packaged descriptions, in which two bodies of knowledge are separated and designated "base" and "target" (e.g., Winston, 1980; Gentner, 1983; Wellsch & Jones, 1986). This is often fine for the base description, which is typically very familiar and relevant information is easily selected. However, it is often difficult to know everything that is needed for mapping until mapping is attempted, at which point mapping may spawn further probes into memory aimed at holes in the match. This entire issue is discussed further in Chapters 3 and 5.

- *Reformulation Problem:* What operations on the structures being examined are allowable?

This has two related facets. First, there are multiple ways to represent equivalent information. Matching two representations may thus require reformulating the descriptions in attempts to recognize identicality or similarity (e.g., `greater-than` and `less-than` to cite a particularly simple case). Second, being a good analogizer means being able to recognize alternative ways of doing things that still satisfy the *intent* of the analogy. For example, recognizing an alternate way to satisfy the roles in a story or function in a plan.

For what share of this problem should mapping be responsible? Clearly, reformulations arising from problem solving impasses must occur during the use phase (e.g., as in derivational analogy (Carbonell, 1983a)). However, what about minor deviations in situation or problem specification that are known prior to problem solving execution? Should the mapping component share responsibility with the use phase for adapting to these minor changes? It is important to know as soon as possible just how well and in what way two potential analogues are actually analogous. This is important to evaluating a potential analogue – an unnecessarily poor match may result in the rejection of a highly useful analogue. It is also important to getting the proper inferences and thus the proper search path. Furthermore, non problem solving settings lack a performance element to sort through the results of a proposed analogy. The results of mapping are the desired, end result. Thus, the information acquirable during mapping should be maximized without losing the ability to perform quick, inexpensive mapping.

Chapter 3

The Structure Mapping Engine

The ability to detect similarity by identifying a set of correspondences between descriptions of two situations is a prerequisite to general analogical processing. Furthermore, in PHINEAS the same matching program is used in both the access and mapping procedures. For these reasons, I will discuss the matching program used in PHINEAS first.

The Structure-Mapping Engine (SME) is a general tool for performing various types of analogical matchings. SME was developed in collaboration with Ken Forbus and Dedre Gentner to simulate Gentner's Structure-Mapping theory of analogy.[1] The intent was to develop a single program that could model all of the similarity comparisons sanctioned by Gentner's theory, such as *literal similarity*, *mere appearance*, as well as *analogy*. The only constraint built into the program is that the mapping preserve structural consistency. All other constraints and all evaluation criteria are supplied in the form of *match rules* that specify the matcher's operation. Thus, while SME was originally designed to simulate the comparisons of structure-mapping theory, it can simulate the space of theories consistent with this single criterion as well. Even so, many of the theoretical distinctions embodied in SME have their origins in Gentner's Structure-Mapping theory.

Given descriptions of a base and target, SME constructs all consistent mappings between them. Each mapping consists of pairwise matches between statements and entities in the base and target, the set of analogical inferences sanctioned by the mapping, and an evaluation score for the mapping. For example, suppose SME were given descriptions of the situations shown in Figure 2.1: water flowing from a beaker to a vial and heat flowing from hot coffee to an ice cube (described in Figure 3.1). SME might offer several alternative analogical mappings. In one, the central inference would be that water flowing between

[1]Furthermore, since portions of this chapter are taken from (Falkenhainer, Forbus, & Gentner, 1987), some of the writting credit for this chapter must go to my coauthors.

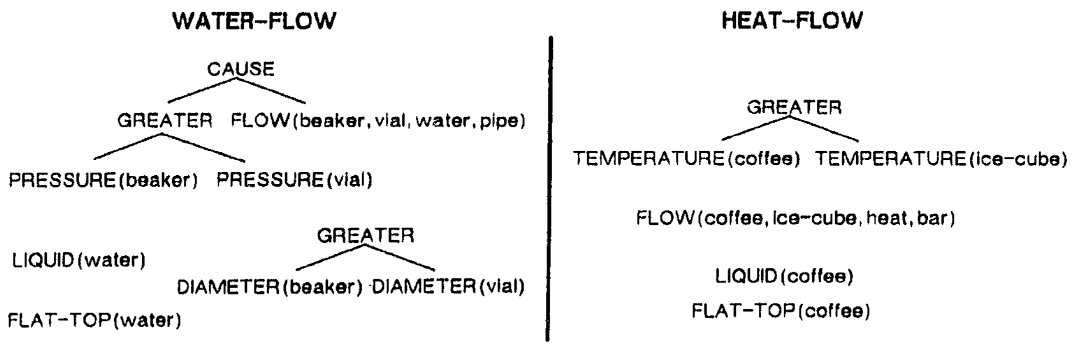


Figure 3.1: Simplified water flow and heat flow descriptions.

the containers corresponds to heat flowing from the coffee to the ice cube. Alternatively, it might map water to coffee, since they are both liquids. Which interpretation has a higher evaluation score depends on the match rules in use.

This chapter first summarizes the SME algorithm[2] and describes the contextual structure mapping configuration (i.e., a new set of match rules) along with additional features that were left out of the earlier paper. Finally, SME is analyzed from both analytical and empirical perspectives. The general program will be called SME, the program running the rules of Gentner's Structure-Mapping theory will be called SME$_{SMT}$, and the program running the rules of contextual structure mapping will be called SME$_{CSM}$. I start by reviewing SME's conventions for knowledge representation, which are essential to understanding the algorithm.

3.1 Representation Overview

A typed (higher-order, in the standard sense) predicate calculus is used to represent facts. The constructs of this language are *entities*, *predicates*, and *dgroups*:

Entities: Entities are logical individuals, i.e., the objects and constants of a domain. Typical entities include physical objects, their temperature, and the substance they are made of.

Predicates: The term "predicate" refers to any functor in a predicate calculus statement. Predicates declared to SME may be divided into primitive categories. In this work there

[2]For details see (Falkenhainer et al., 1987).

are only two: *relation* and *function* (Falkenhainer et al., 1987 also discusses an *attribute* category).

All predicates must be declared to SME prior to use (the declarations for PHINEAS are listed in Appendix E). Each declaration defines the predicate's arity, a name and type (sort) for each argument, and the next most general type (sort) the predicate maps to. For example, the declaration:

```
(defPredicate PRESSURE ((obj physob)) function
                              :expression-type intensive-quantity)
```

states that the predicate PRESSURE is a one-place function. Its argument is called obj and is of type physob. An expression using it, such as PRESSURE(water1), maps an expression of type physob, water1, to an expression of type pressure, which in turn is of type intensive-quantity, the next node up in the ISA hierarchy.

Predicates may additionally be declared *commutative*, in which the order of arguments is unimportant when matching, and/or *n-ary*, in which the predicate can take any number of arguments. Examples of commutative n-ary predicates include AND, OR, and SET.

Dgroup: For simplicity, predicate instances and compound terms are called *expressions*. A *description group*, or *dgroup*, is a collection of entities and expressions concerning them, considered as a unit. The expressions and entities in a dgroup will be referred to collectively as *items*.

Dgroups are defined with the defDescription form:

(defDescription $\langle DescriptionName \rangle$
  entities ($\langle Entity_1 \rangle$, $\langle Entity_2 \rangle$, ..., $\langle Entity_i \rangle$)
  expressions ($\langle ExpressionDeclarations \rangle$))

where $\langle ExpressionDeclarations \rangle$ take the form

$\langle expression \rangle$ or
($\langle expression \rangle$:name $\langle ExpressionName \rangle$)

The :name option is provided for convenience; $\langle expression \rangle$ will be substituted for every occurrence of $\langle ExpressionName \rangle$ in the dgroup's expressions when the dgroup is created. For example, the description of water flow depicted in Figure 3.1 was given to SME as

```
(defDescription simple-water-flow
      entities (water beaker vial pipe)
      expressions (((flow beaker vial water pipe) :name wflow)
                   ((pressure beaker) :name pressure-beaker)
                   ((pressure vial) :name pressure-vial)
                   ((greater pressure-beaker pressure-vial) :name >pressure)
                   ((greater (diameter beaker) (diameter vial)) :name >diameter)
                   ((cause >pressure wflow) :name cause-flow)
                   (flat-top water)
                   (liquid water)))
```

The description of heat flow depicted in Figure 3.1 was given to SME as

```
(defDescription simple-heat-flow
      entities (coffee ice-cube bar heat)
      expressions (((flow coffee ice-cube heat bar) :name hflow)
                   ((temperature coffee) :name temp-coffee)
                   ((temperature ice-cube) :name temp-ice-cube)
                   ((greater temp-coffee temp-ice-cube) :name >temperature)
                   (flat-top coffee)
                   (liquid coffee)))
```

3.2 SME Algorithm Overview

Given descriptions of a base and a target, represented as dgroups, SME builds all structurally consistent interpretations of the comparison between them. Each interpretation of the match is called a *global mapping*, or *gmap*. Gmaps consist of three parts:

1. *Correspondences:* A set of pairwise matches between the expressions and entities of the two dgroups.
2. *Candidate Inferences:* A set of new expressions which the comparison suggests holds in the target dgroup.
3. *Evaluation Score:* A numerical estimate of match quality. The characteristics used to determine the score depend on the rule set and may include the gmap's structural properties and its contextual relevance.

Match rules specify which local elements can match, additional restrictions on how they may be combined, and how these combinations are scored. These rules are the key to SME's flexibility. To build a new matcher one simply loads a new set of match rules. This has some important advantages. First, a theory of analogical mapping may be represented more declaratively, and the consequences of each theoretical commitment easily traced. Second, it enables a single program to emulate different analogy systems for comparison purposes.

Conceptually, the SME algorithm is divided into four stages:

1. *Local match construction:* Finds all pairs of (⟨*BaseItem*⟩, ⟨*TargetItem*⟩) that potentially can match. A *Match Hypothesis* is created for each such pair to represent the possibility that this local match is part of a global match.

2. *Gmap construction:* Combines the local matches into maximal consistent collections of correspondences.

3. *Candidate inference construction:* Derives the inferences suggested by each gmap.

4. *Match Evaluation:* Attaches evidence to each local match hypothesis and uses this evidence to compute evaluation scores for each gmap.

Each computation will now be described, using a simple example to illustrate their operation.

3.2.1 Step 1: Local match construction

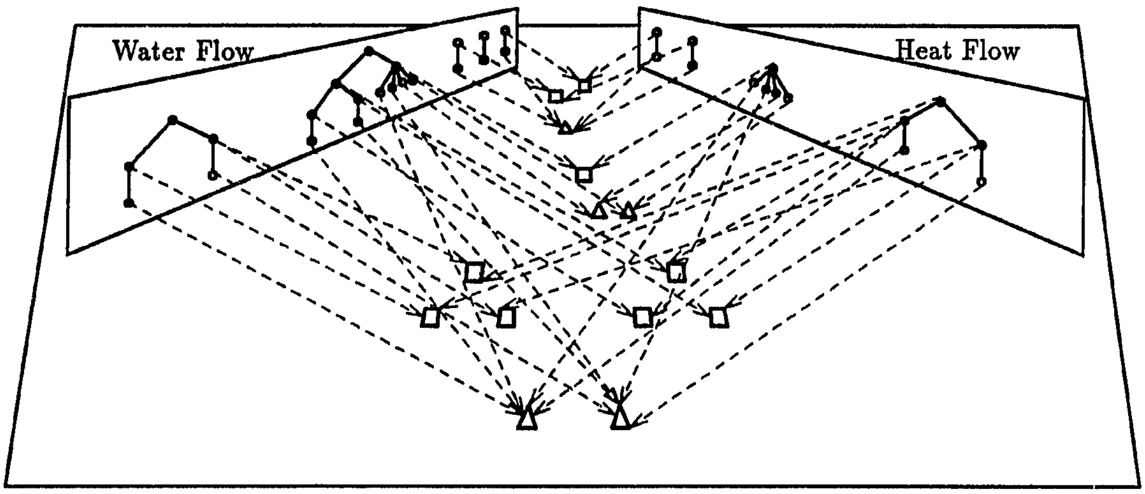
Given two dgroups, SME begins by finding potential matches between items in the base and target (see Figure 3.2). Allowable matches are specified by *match constructor* rules, which take the form:

(MHCrule (⟨*Trigger*⟩ ⟨*BaseVariable*⟩ ⟨*TargetVariable*⟩ ⟨*Condition*⟩) ⟨*Body*⟩)

There are two possible values for ⟨*Trigger*⟩. A :filter trigger indicates that the rule is applied to each pair of items from the base and target, creating a match hypothesis when the items satisfy the condition. For example, the following rule hypothesizes a match between any two expressions that have the same functor:

```
(MHCrule (:filter ?b ?t :test (equal (expression-functor ?b)
                                     (expression-functor ?t)))
      (install-MH ?b ?t))
```

An :intern trigger indicates that the rule should be run on each newly created match hypothesis. These rules create additional matches suggested by the given match hypothesis. For example, hypothesizing matches between every pair of entities would lead to combinatorial explosions. Instead, :intern rules are used to create match hypotheses between entities in corresponding argument positions of other match hypotheses.



◻ - MH between predicates
△- MH between entities (Emap)

Figure 3.2: Local Match Construction. The water flow and heat flow descriptions of Figure 3.1 have been drawn in the abstract and placed to the left and right, respectively. The objects in the middle depict match hypotheses.

The result of running the match constructor rules is a collection of match hypotheses. $MH(b_i, t_j)$ denotes the hypothesis that b_i and t_j match. Standard graph-theory terminology will be used to describe the structural properties of graphs of match hypotheses (e.g., offspring, descendants, ancestors, root).

For example, the result of running the match constructor rules on the water flow and heat flow dgroups of Figure 3.1 is shown in Figure 3.2 (see also Figure 3.3). In this example, the *literal similarity* rule set of structure-mapping theory is used (i.e., $\text{SME}_{SMT/LS}$). There are several points to notice in Figure 3.3. First, there can be more than one match hypothesis involving any particular base or target item. Second, in this rule set, predicates are paired due to identicality, while entities are matched on the basis of their roles in the predicate structure. Thus while TEMPERATURE can match either PRESSURE or DIAMETER, GREATER cannot match anything but GREATER. Third, not every possible correspondence is created. Local matches between entities are only created when justified by some other match. This significantly constrains the number of possible matches in the typical case.

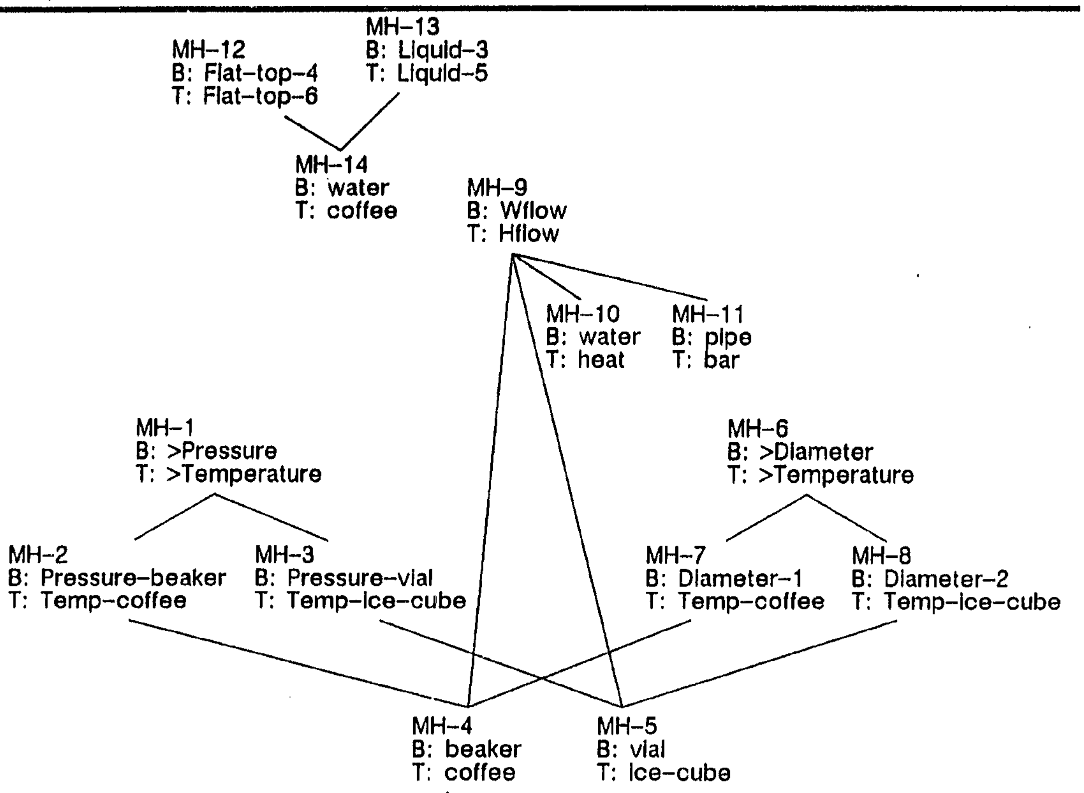


Figure 3.3: Water Flow / Heat Flow Analogy After Local Match Construction. Here we show the graph of match hypotheses depicted schematically in Figure 3.2, augmented by links indicating expression-to-arguments relationships. Match hypotheses which are not descended from others are called *roots* (e.g., the matches between the GREATER predicates, MH-1 and MH-6, and the match for the predicate FLOW, MH-9). Match hypotheses between entities are called *Emaps* (e.g., the match between beaker and coffee, MH-4).

3.2.2 Step 2: Global Match Construction

The second step in the SME algorithm combines local match hypotheses into collections of global matches (gmaps). Intuitively, each global match is the largest possible set of match hypotheses that depend on the same one to one object correspondences.

More formally, gmaps consist of *maximal, consistent* collections of match hypotheses that are *structurally grounded.* A collection of match hypotheses is structurally grounded if it satisfies the *grounding criterion*: If a match hypothesis MH is in the collection, then so are the match hypotheses which pair up all of the arguments of MH's base and target items.[3] The grounding criterion preserves connected predicate structure. Consistency is determined by the rule set and in all cases to date includes the restriction that the mapping be one-to-one. A collection is maximal if adding any additional match hypothesis would render the collection inconsistent.

The formation of global matches is composed of two primary stages: *compute consistency relationships* and *merge match hypotheses.*

3.2.2.1 Compute consistency relationships

For each match hypothesis, generate the set of entity mappings it entails and the set of match hypotheses it is inconsistent with. This information simplifies the detection of contradictory sets of match hypotheses, a critical operation in the rest of the algorithm. The result of this stage of processing appears in Figure 3.4.

The following two sets manage inconsistency and are crucial to this computation:

Definition 3.1 (Conflicting) *Given a match hypothesis $MH(b_i, t_j)$, the set Conflicting($MH(b_i, t_j)$) consists of the set of match hypotheses that are pairwise inconsistent with it.*

The set *Conflicting($MH(b_i, t_j)$)* only notes local inconsistencies (see Figure 3.4). For example, under a one-to-one restriction, *Conflicting($MH(b_i, t_j)$)* would include the set of match hypotheses that represent the alternate mappings for b_i and t_j.

Conflicting and the grounding criterion combine to produce the following set:

Definition 3.2 (NoGood) *The set NoGood(MH_i) is the set of all match hypotheses which can never appear in the same gmap as MH_i. This set is recursively defined as follows: if MH_i is an emap, then $NoGood(MH_i) = Conflicting(MH_i)$. Otherwise, $NoGood(MH_i)$ is the union of MH_i's Conflicting set with the NoGood sets for all of its descendents, i.e.,*

$$NoGood(MH_i) = Conflicting(MH_i) \bigcup \cup_{MH_j \in Args(MH_i)} NoGood(MH_j)$$

3.2.2.2 Merge match hypotheses

Compute gmaps by successively combining match hypotheses as follows:

[3]This represents a deviation from previous accounts, such as (Falkenhainer et al., 1987). In addition to the grounding criterion, a one-to-one restriction was enforced by SME. One-to-one mappings are now an option, implementable in the rules, so that variations on one-to-one may be tested. Additionally, an important exception to the grounding criterion is now allowed, as discussed in Section 3.3.

(a) *Form initial combinations:* Combine interconnected and consistent match hypotheses into an initial set of gmaps (Figure 3.5a).

(b) *Combine dependent Gmaps:* Since base and target dgroups are rarely isomorphic, some gmaps in the initial set will overlap in ways that allow them to be merged. The advantage in merging them is that the new combination may provide structural support for candidate inferences (Figure 3.5b).

(c) *Combine independent collections:* The results of the previous step are next combined to form maximal consistent collections (Figure 3.5c).

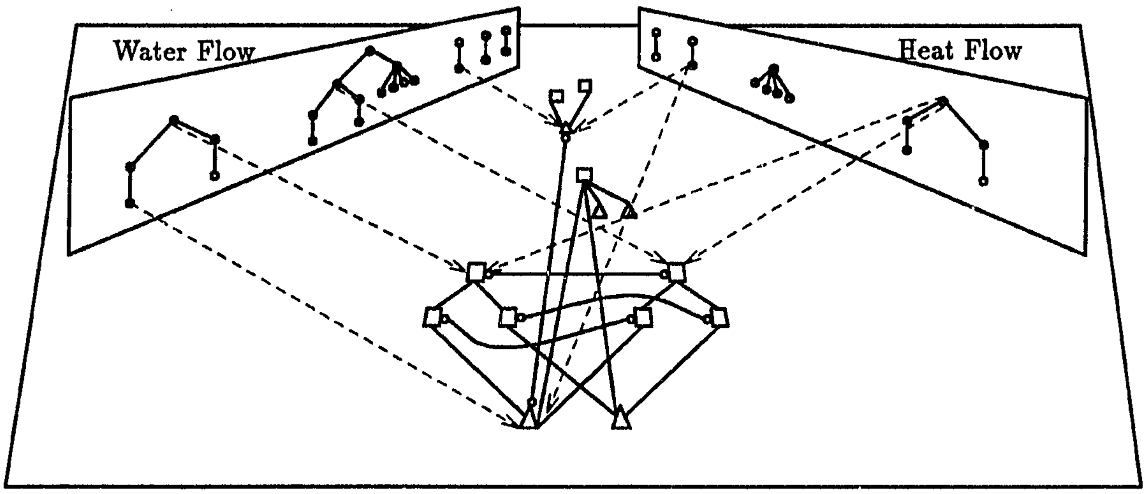
Commutative predicates are supported during step (a). When multiple, complete matches exist for the arguments of two commutative predicates, a copy of the match between them is made and assigned to each complete, consistent combination of argument matches. For example, if AND(b_1,b_2) were matched to AND(t_1,t_2), step (a) would replace the single match hypothesis between the two AND's with two alternate match hypotheses, and place them in different gmaps, corresponding to the two alternate ways to pair their arguments.

3.2.3 Step 3: Compute Candidate Inferences

Associated with each gmap is a (possibly empty) set of *candidate inferences.* Candidate inferences are base expressions that would fill in structure which is not in the gmap (and hence not already in the target). Not just any information can be carried over – it must be consistent with the substitutions imposed by the gmap, and it must be *structurally grounded* in the gmap. By structural grounding we mean that its subexpressions must at some point intersect the base information belonging to the gmap.

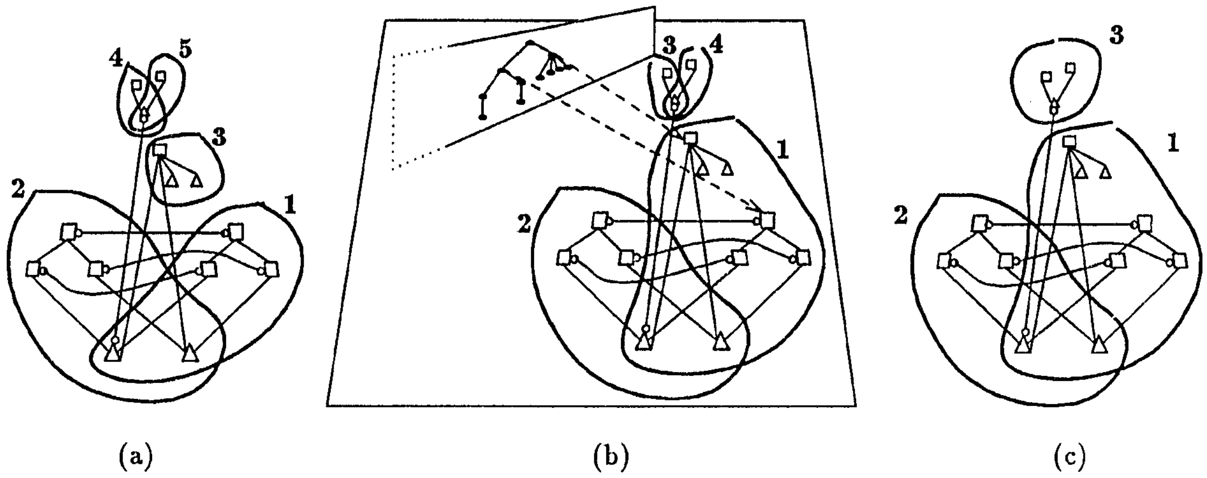
The candidate inferences often include entities. Whenever possible, SME replaces all occurrences of base entities with their corresponding target entities. If a candidate inference contains a base entity that has no corresponding target entity (i.e., the base entity is not part of any match hypothesis for that gmap), SME introduces a new, hypothetical entity into the target. Such entities are represented as a skolem function of the original base entity (i.e., (:skolem base-entity)).

In Figure 3.6, gmap #1 has the top level CAUSE predicate as its sole candidate inference. In other words, this Gmap suggests that the cause of the flow in the heat dgroup is the difference in temperatures. If the FLOW predicate was not present in the target, then the candidate inferences for a gmap corresponding to the pressure inequality would be both CAUSE and FLOW. Note that GREATER-THAN[DIAMETER(coffee), DIAMETER(ice cube)] is



□ - MH between predicates
△- MH between entities (Emap)

Figure 3.4: Water Flow - Heat Flow analogy after computation of *Conflicting* relationships. Simple lines show the tree-like graph that the grounding criteria imposes upon match hypotheses. Lines with circular endpoints indicate the *Conflicting* relationships between matches. Some of the original lines from MH construction have been left in to show the source of a few *Conflicting* relations.



□- MH between predicates
△ MH between entities (Emap)

Figure 3.5: Gmap Construction. (a) Merge step 1: Interconnected and consistent. (b) Merge step 2: Consistent members of the same base structure. (c) Merge step 3: Any further consistent combinations.

```
                SME Version 2E
    Analogical Match from SWATER-FLOW to SHEAT-FLOW.

Rule File: literal-similarity.rules
-------------------------------------------------------
# MH's | # Gmaps |      1st,2nd,Worst     | RelGroups |
   14  |    3    | 5.99 /  3.94 /  2.44 |     OFF   |
-------------------------------------------------------
Total Run Time:     0 Minutes,  0.657 Seconds
BMS Run Time:       0 Minutes,  0.441 Seconds
Best Gmaps: { 1 }

Gmap #1: (>PRESSURE >TEMP)  (PRESS-VIAL TEMP-ICE-CUBE)  (PRESS-BEAKER TEMP-COFFEE)
         (WFLOW HFLOW)  (BEAKER COFFEE)  (VIAL ICE-CUBE)  (PIPE BAR)  (WATER HEAT)
  Weight: 5.9917
  Candidate Inferences:  (CAUSE >TEMP HFLOW)

Gmap #2: (DIAM-BEAKER TEMP-COFFEE)  (DIAM-VIAL TEMP-ICE-CUBE)  (>DIAMETER >TEMP)
         (BEAKER COFFEE)  (VIAL ICE-CUBE)
  Weight: 3.9377
  Candidate Inferences:  { }

Gmap #3: (LIQUID-WATER LIQUID-COFFEE)  (FLAT-WATER FLAT-COFFEE)  (WATER COFFEE)
  Weight: 2.4446
  Candidate Inferences:  { }
```

Figure 3.6: Complete SME$_{SMT/LS}$ interpretation of Water Flow - Heat Flow Analogy.

not a valid candidate inference for the first Gmap because it does not intersect the existing Gmap structure.

3.2.4 Step 4: Compute Evaluation Scores

Typically a particular base and target pair will give rise to several gmaps, each representing a different interpretation of the match. Often it is desired to select only a single gmap, for example to represent the best interpretation of an analogy. Evaluation criteria may include structural properties, such as systematicity, as well as contextual relevance, validity, and so forth. An evaluation score for each match hypothesis and gmap is found by running *match* *evidence* rules and combining their results.[4] These scores are used to rank-order the gmaps in selecting the "best" analogy. For example, the rule

```
(rule ((:intern (MH ?b ?t) :test (and (expression? ?b) (expression? ?t)
                                      (eq (expression-functor ?b)
                                          (expression-functor ?t)))))
     (assert! (implies same-functor (MH ?b ?t) (0.5 . 0.0))))
```

states "If the two items are facts and their functors are the same, then supply 0.5 evidence in favor of the match hypothesis." The rules may also examine match hypotheses associated with the arguments of these items to provide support based on systematicity. This causes evidence for a match hypothesis to increase with the amount of higher-order structure supporting it.

Returning to Figure 3.6, note that the "strongest" interpretation (i.e., the one which has the highest evaluation score) is the one we would intuitively expect. In other words, `beaker` maps to `coffee`, `vial` maps to `ice-cube`, `water` maps to `heat`, `pipe` maps to `bar`, and PRESSURE maps to TEMPERATURE. Furthermore, it sanctions the candidate inference that the temperature difference is what causes the flow.

3.3 Modeling Contextual Structure Mapping

Contextual structure mapping is modeled within the rule set given to SME, which defines SME_{CSM}. Since the entire rule set consists of only 21 rules, I will describe the complete set here, using standard predicate calculus notation. Variables will be preceded by "?" and are assumed universally quantified. The complete set of equivalent lisp rules used by SME_{CSM} are provided in Appendix B. The description follows the program decomposition used in the previous section.

3.3.1 Step 1: Local match construction

SME begins by running *match constructor* rules, which install match hypotheses between individual base and target items that may *plausibly* match.

The first three rules match expressions by examining the predicates they use and the inferential support they provide.

[4]The management of numerical evidence is performed by a *Belief Maintenance System* (BMS) (Falkenhainer, 1988b). The BMS is much like a standard TMS, using horn clauses as justifications. However, the justifications are annotated with evidential weights, so that "degrees of belief" may be propagated. A modified version of Dempster-Shafer formalism is used for expressing and combining evidence.

Rule 1 (Same Functors) *Two expressions may match if they use the same predicate and their predicates are not part of an a priori correspondence set.*

```
Equal[functor(?b),functor(?t)] ∧
¬Paired-BItem(functor(?b)) ∧ ¬Paired-TItem(functor(?t))
        ⇒ MH(?b,?t)
```

Rule 2 (Common Generalization) *Two expressions may match if they use predicates having common ancestors in the generalization hierarchy and their predicates are not part of an a priori correspondence set.*

```
MH(?bp,?tp) ∧ Children-of?(?b,?t,?bp,?tp) ∧
Common-Ancestor?[functor(?b),functor(?t)] ∧
¬Paired-BItem(functor(?b)) ∧ ¬Paired-TItem(functor(?t))
        ⇒ MH(?b,?t)
```

The previous two rules suffer from dependence on identicality or a generalization hierarchy. They fail to take into account the context in which an expression is being used. Two very different predicates may support the same conclusion in a given context. This is addressed by the following rule.

Rule 3 (Functionally Analogous) *Two expressions are considered* functionally analogous *and may match if they provide the same inferential support in the context of the structures being matched.*

```
Implicational(?b) ∧ Implicational(?t) ∧
Equal(functor[consequent(?b)], functor[consequent(?t)])
        ⇒ [MH(antecedent(?b),antecedent(?t))
            ∧ Function-of[antecedent(?b),Support-of(consequent(?b)]
            ∧ Provides-function[antecedent(?t),Support-of(consequent(?t))]]
```

If two expressions are *implicational* and their consequents match, then this rule will match their antecedents. At the current time, the predicates IMPLIES, CAUSE, and SUPPORTS are considered implicational.

The following rule respects established mappings.

Rule 4 (Sanctioned Pairing) *Two items match if they are a priori designated as matching.*

```
Sanctioned-Pairing(?b,?t) ⇒ MH(?b,?t)
```

Other rules are used to create match hypotheses between entities in corresponding argument positions of other match hypotheses. In this manner, entities are only matched if sanctioned by their position in matching relations:

Rule 5 (Non-Commutative Corresponding Arguments, Entities) *Two entities match if they occupy the same argument position of non-commutative predicates that have already been matched and neither entity is part of an a priori correspondence set.*

```
MH(?b₁,?t₁) ∧ ¬Commutative[functor(?b₁)] ∧ ¬Commutative[functor(?t₁)] ∧
Entity(?b₂) ∧ Entity(?t₂) ∧ Children-of?(?b₂,?t₂,?b₁,?t₁) ∧
¬Paired-BItem(functor(?b)) ∧ ¬Paired-TItem(functor(?t))
        ⇒ MH(?b₂,?t₂)
```

Three more rules exist to form matches between the arguments of expressions that have already matched. The first is like Rule 5, except that the corresponding arguments must be expressions whose predicate is a function. The remaining two match constructor rules pair the arguments of commutative predicates (i.e., the "corresponding arguments" condition, `children-of`, is removed). These two rules generate all possible entity and function pairings between the arguments of two commutative predicates.

3.3.2 Step 2: Global match construction

Once an initial set of match hypotheses is formed, the pairwise consistency of match hypotheses stated by *Conflicting* is used to combine them into maximal, consistent gmaps. By the one-to-one criterion, these include match hypotheses representing alternate mappings for b_i and t_j.

Rule 9 (One-To-One (expressions & entities), Base Case) *Two match hypotheses are mutually inconsistent and may not appear in the same gmap if they pair the same base item to different target items.*

```
MH(?b,?t₁) ∧ MH(?b,?t₂) ∧ ¬Equal(?t₁,?t₂)
        ⇒ Conflicting[MH(?b,?t₁),MH(?b,?t₂)]
```

A similar rule establishes that two match hypotheses pairing the same target item to different base items are mutually inconsistent. An additional pair of rules maintains the one-to-one mapping for predicates in the same manner used for expressions and entities.

Other elements of *Conflicting*($MH(b_i,t_j)$) are defined by representation specific and domain specific rules. The following rule is used to prevent the temporal rearrangement problem described in Section 2.2.

Rule 13 (Temporal Preservation) *Two match hypotheses are mutually inconsistent if they pair base items always co-occurring in time, EqualTime($?b_1, ?b_2$), to target items that never overlap in time, DisjointTime($?b_1, ?b_2$). Likewise, two match hypotheses are mutually inconsistent if they pair target items always co-occurring in time to base items that never overlap in time.*

```
MH(?b₁,?t₁) ∧ MH(?b₂,?t₂) ∧
Temporally-Scoped(?b₁) ∧ Temporally-Scoped(?b₂) ∧
Temporally-Scoped(?t₁) ∧ Temporally-Scoped(?t₂) ∧
{ [EqualTime(?b₁,?b₂) ∧ DisjointTime(?t₁,?t₂)] ∨
  [DisjointTime(?b₁,?b₂) ∧ EqualTime(?t₁,?t₂)]}
        ⇒ Conflicting[MH(?b₁,?t₁),MH(?b₂,?t₂)]
```

The rearrangement problem exists for any undecomposable collection. The following rule preserves the compound object "contained liquid".

Rule 14 (Compound Object Preservation (contained liquids)) *Two match hypotheses are mutually inconsistent if they pair items representing a contained liquid and its container with another contained liquid and something other than its container, respectively.*

```
MH(?b₁,?t₁) ∧ MH(?b₂,?t₂) ∧
Contained-Liquid(?b₁) ∧ Contained-Liquid(?t₁) ∧
{ [Container-of(?b₁,?b₂) ∧ ¬Container-of(?t₁,?t₂)] ∨
  [¬Container-of(?b₁,?b₂) ∧ Container-of(?t₁,?t₂)]}
        ⇒ Conflicting[MH(?b₁,?t₁),MH(?b₂,?t₂)]
```

One element of contextual structure mapping required a change to the SME program itself – relaxing the structural grounding criterion to exclude relational groups. In the standard procedure for copying match hypotheses between commutative predicates (described in section 3.2.2.2), complete match sets for their arguments were required. In the case of a match between predicates forming relational groups, the requirement is weakened. Only a single match between their arguments need exist, and all maximal, consistent collections of argument pairings constitute a valid copy of the commutative pairing.[5]

3.3.3 Step 4: Compute evaluation scores

Once the gmaps have been formed (step 2) and their corresponding candidate inferences computed (step 3), each match hypothesis and gmap is assigned a match evaluation score.[6]

[5]If this sounds like the SME match problem in miniature, in many ways it is. The same gmap merge step procedure is used for this operation. Rather than "relational groups", one could view these as a dgroup within a dgroup.

[6]It should be pointed out that numerical evidence is used to provide a simple way to combine local information concerning match quality. These weights have nothing to do with any probabilistic or evidential

As described in Chapter 2, both structural and relevance criteria are used to compute these evaluation scores.

The evidence rules have a slightly different syntax from the rules described above. Rather than implying a particular match hypothesis form, they supply evidence through the form `Implies(`⟨*antecedent*⟩`,`⟨*consequent*⟩`,`⟨*weight*⟩`)` (see Falkenhainer, 1988b; Falkenhainer et al., 1987 for an explanation of the evidence mechanism). In each rule described below, ⟨*weight*⟩ will be given as *#¡parameter-name¿*. The numeric values of these parameters are then summarized at the end of the section.

The first evidence rule supports the first two match constructor rules, which examined the predicates in use. It supplies evidence in inverse proportion to the distance within the generalization hierarchy between the predicates matched. This distance is the number of nodes in the minimal path between the matched predicates in the hierarchy, reducing to one in the case of predicate identicality.

Rule 15 (Minimal Ascension) *If the expressions comprising a match hypothesis were paired due to common ancestors in the generalization hierarchy, then supply an evidence score inversely proportional to their distance (number of nodes) in the hierarchy, equal to #MA/distance.*

```
MH(?b,?t) ∧ Expression(?b) ∧ Expression(?t) ∧
¬Paired-BItem(functor(?b)) ∧ ¬Paired-TItem(functor(?t))
        ⇒ Implies[type-match, MH(?b,?t),
                    quotient(#MA,path-length(functor(?b),functor(?t)))]
```

Rule 16 (Sanctioned Pairing Evidence) *If two items are a priori designated as matching, then supply an evidence score of #SP to the match.*

```
Sanctioned-Pairing(?b,?t) ⇒ Implies[sanctioned-pairing, MH(?b,?t), #SP]
```

Rule 17 (Functionally Analogous Evidence) *If the base expression of a match hypothesis provides inferential support f in the base situation, and the target expression can provide that inferential support, then supply an evidence score of #FA to the match.*

```
MH(?b,?t) ∧ Expression(?b) ∧ Expression(?t) ∧
Function-of(?b,?f) ∧ Provides-function(?t,?f)
        ⇒ Implies[And(Function-of(?b,?f),Provides-function(?t,?f)),
                    MH(?b,?t), #FA]
```

information about the base or target per se. Additionally, the evidence scores used here are lower than those previously described for SME$_{SMT}$. It was found that the prior scores pushed the weights too far into the high end of the 0..1 spectrum, offering little difference between fair matches and very good matches.

Systematicity is supported by passing evidence from a match involving a relationship to the matches involving its arguments. The following rule accomplishes this by propagating *#SYS%* of a match hypothesis' belief to its offspring.

Rule 18 (Systematicity, Non-Commutative Case)

```
MH(?b₁,?t₁) ∧ MH(?b₂,?t₂) ∧ Children-of(?b₂,?t₂,?b₁,?t₁)
        ⇒ Implies[MH(?b1,?t1), MH(?b2,?t2), #SYS]
```

A second rule is used to propagate systematicity through commutative predicates by removing the corresponding arguments test (i.e., Children-of). The more matched structure that exists above a given match hypothesis, the more that hypothesis will be believed. Thus this "trickle down" effect provides a local encoding of Gentner's systematicity principle.

Finally, contextual relevance is used to provide additional evidence for those matches supporting the current reasoning needs of the global reasoning system. There are two factors to consider. First, which relations are more salient for the current reasoning task? For example, in PHINEAS the central focus is to explain an observed *behavior*. Matches identifying corresponding behavior are given greater import than matches for other features.

Rule 20 (Behavioral) *If a match hypothesis is between two behavioral relations (e.g., Increasing, Decreasing), then supply an evidence score of #Behavior to the match.*

```
MH(?b,?t) ∧ Expression(?b) ∧ Expression(?t) ∧
Behavioral-Relation(?b) ∧ Behavioral-Relation(?t)
        ⇒ Implies[behavioral, MH(?b,?t), #Behavior]
```

Second, which gmaps offer candidate inferences providing needed knowledge? For example, if a cause for E is sought, gmaps offering the inference Cause(C,E) would be preferred. In PHINEAS, the relation B-Explains($\mathcal{T}$,$\mathcal{B}$) is used to state that the set of theories $\mathcal{T}$ explain the behavior $\mathcal{B}$. This predicate is then sought as part of the candidate inferences for a gmap.

Rule 21 (Provides Relevant Inference) *If a gmap contains a candidate inference supporting a behavioral explanation, then supply an evidence score of #RInf to the gmap.*

```
CI(?gmap, B-Explains(?base-theory,?target-behavior)) ∧
Current-Observation(?target-behavior)
        ⇒ Implies[CI(?gmap, B-Explains(?base-theory,?target-behavior)),
                  ?gmap, #RInf]
```

The specific parameter settings used in SME$_{CSM}$ are:

Evidence Parameter	Value
#MA	0.4
#SP	0.4
#FA	0.8
#SYS	0.8
#Behavior	0.4
#RInf	0.9

Whenever numeric weights are used to influence a system's function, there is danger of (1) tailoring for particular examples and (2) sensitivity to specific values. Values for the evidence parameters in SME_{CSM} were selected based on long experience with SME_{SMT} and general intuition. Of course, a more formal sensitivity analysis is required. However, SME_{CSM}'s evidence parameters have not changed throughout the development of PHINEAS and the same rule set was used for all examples. Both Ken Forbus and myself have conducted preliminary empirical studies to determine SME_{SMT}'s sensitivity to the space of possible parameter settings. On simple examples such as the small water flow, heat flow analogy described in Section 3.2, it was found that simply having non-zero settings is sufficient. On more complex analogies, such as the short stories discussed in (Skorstad et al., 1987), performance is robust but not completely insensitive to parameter settings. Most crucial is the setting for systematicity. This must be high for SME_{SMT} to demonstrate a marked preference for higher-order systems of relations. The findings are too preliminary to draw conclusions. Analysis is tedious due to the size of the space being considered – ranging SME_{SMT}'s eight parameters through four values each yields 65,536 data points for a single base–target pair. Although these findings should apply equally well to SME_{CSM}, studies on SME_{CSM} have not yet begun.

3.4 Analysis

This section presents a critical review of SME from both analytical and empirical perspectives. First, a summary of SME's complexity is given. This is followed by an empirical discussion of alternate domain representations and their impact on performance. Finally, SME's generality is described in the context of its existing applications.

3.4.1 Complexity analysis

(Falkenhainer et al., 1987) presents a detailed analysis of each phase in the SME algorithm. This section summarizes those results. In the discussion, N_b and N_t are the number of base and target dgroup items, respectively. Their average is denoted by N.

1. *Local match construction:* In both SME$_{SMT}$ and SME$_{CSM}$, match constructor rules are simple and we may assume rule execution takes unit time. Under this assumption, both `:filter` and `:intern` rules require $\mathcal{O}(N_b * N_t)$ or roughly $\mathcal{O}(N^2)$. However, in practice the `:intern` rules have a run time of approximately $\mathcal{O}(N)$.

2. *Calculating Conflicting:* SME$_{CSM}$ assigns a *Conflicting* set to each match hypothesis, $MH(b_i, t_j)$ which represents the alternate mappings for b_i and t_j. Worst case is $\mathcal{O}(N^2)$, while the best case performance is $\mathcal{O}(max(N_b, N_t))$.

3. *Emaps and NoGood calculation:* Each match hypothesis is operated on once, which in the worst case is $\mathcal{O}(N^2)$.

4. *Gmap construction:* Global matches are formed in a sequence of three merge steps:

   (a) Assuming that most of the match hypotheses will appear in only one or two subgraphs (some roots may share substructure), the first merge step is proportional to the number of match hypotheses, or worst case $\mathcal{O}(N^2)$.

   (b) In the worst case, this step is equivalent to Step 4(c), which can display $\mathcal{O}(N!)$ performance. If the base and target dgroups give rise to a match hypothesis graph having a single, consistent root, then there is only one gmap and the second (and third) merge steps are constant-time. Typically, the second merge step is very quick and displays near best-case performance.

   (c) The complexity of this final merge step is directly related to the degree of structure in the base and target domains and how many different predicates are in use. This issue is reexamined in the next section. Worst-case performance occurs when the description language is flat (i.e., no higher-order structure) and the same predicate occurs many times in both the base and the target. In a language with a single, unary predicate, this reduces to the problem of finding all isomorphic mappings between two equal size sets, which is $\mathcal{O}(N!)$. In the other extreme, when the base and target dgroups give rise to a match hypothesis graph that has but one root, the third merge step is a constant-time operation.

5. *Finding candidate inferences:* In the worst case we have an upper bound of $\mathcal{O}(N^3)$. In the best case, there will only be one gmap and no candidate inferences, producing constant time behavior.

6. *Selection score computation:*

   The complexity of the evidence mechanism (BMS) is difficult to ascertain, and ranges from 80% of SME's processing time on small examples to less than 5% on large examples. The BMS maintains dependencies between evidential results and may be eliminated if their explanation is not required. The original SME (Falkenhainer et al., 1986) used a specialized $\mathcal{O}(N^2)$ system.

3.4.2 Implications for Representation

The proper representation becomes an issue in SME due to its significant impact on speed performance. Highly structural, nested representations provide an important source of constraint on generating potential matches. They tend to make the semantic interrelations explicit in the structure of the syntax. For example, a theory might be represented as a sequence of axiom statements

Axiom-of(T1, $axiom_i$)

or as

Theory(T1, SET($axiom_1$, ... $axiom_i$ ... $axiom_n$))

While SME is able to process domain descriptions in any predicate-based format, the latter is significantly more efficient. The reason is that the set representation for theories places the related axioms syntactically together, reducing the number of spurious local matches.

Several different representations for temporal intervals were empirically tested with SME. In each, the base described the four state cyclic behavior of a spring-block configuration, while the target described the four state cyclic behavior of an oscillating LC circuit. Static situation information was also included in all representations, such as (BLOCK block) and (CONNECTED spring block). The first representation tested was a standard situation calculus syntax:

```
(DURING (CONSTANT (POSITION spring)) S1)
(DURING (CONSTANT (POSITION block)) S1)
(DURING (DECREASING (POSITION spring)) S2)
(DURING (DECREASING (POSITION block)) S2)
```

```
(DURING (CONSTANT (POSITION spring)) S3)
(DURING (CONSTANT (POSITION block)) S3)
(DURING (INCREASING (POSITION spring)) S4)
(DURING (INCREASING (POSITION block)) S4)
(MEETS S1 S2)
(MEETS S2 S3)
(MEETS S3 S4)
(MEETS S4 S1)
```

This produced 306 possible gmaps and took SME a total of 2 minutes.

The second representation was taken from Hayes' (1979) definition of *history* as a description of a single object's behavior over time. The predicate AT is used to specify a *slice*, the intersection of an object with a period of time (either interval or instant).[7] In this representation, the oscillatory behavior is described as:

```
(CONSTANT (POSITION (AT spring S1)))
(CONSTANT (POSITION (AT block S1)))
(DECREASING (POSITION (AT spring S2)))
(DECREASING (POSITION (AT block S2)))
(CONSTANT (POSITION (AT spring S3)))
(CONSTANT (POSITION (AT block S3)))
(INCREASING (POSITION (AT spring S4)))
(INCREASING (POSITION (AT block S4)))
(MEETS S1 S2)
(MEETS S2 S3)
(MEETS S3 S4)
(MEETS S4 S1)
```

This syntax produced 756 possible gmaps and took SME a total of 29 minutes, 52 seconds. It should also be noted that this is the only description of time that is *a priori* immune to the structure rearrangement problem described in Section 2.2. This is because the time token is present at the bottom-most level of description; there is no smaller expression that contains the object token spring and does not contain the state token S1.

Finally, the nested temporal syntax currently used in PHINEAS was tested. The syntax for a temporal interval, called a *situation*, is:

SITUATION(⟨*NameToken*⟩, ⟨*Relations*⟩)

Using this syntax, the oscillator was described as:

[7]This was the representation used in an earlier version of PHINEAS (Falkenhainer, 1986).

Table 3.1: SME performance on alternate temporal representations.

Representation	# MH's	# Gmaps	Total BMS run time	Total match run time
Situation Calculus	164	306	0:27.72	1:33.18
Slices	212	756	0:49.34	29:02.32
Nested	125	46	0:19.07	0:06.63

NOTE: All times are given in minutes:seconds.fraction. Total match time is total SME run time minus BMS run time.

```
(MEETS (SITUATION S1 (SET (CONSTANT (POSITION spring))
                          (CONSTANT (POSITION block))))
       (SITUATION S2 (SET (DECREASING (POSITION spring))
                          (DECREASING (POSITION block)))))
(MEETS (SITUATION S2 (SET (DECREASING (POSITION spring))
                          (DECREASING (POSITION block))))
       (SITUATION S3 (SET (CONSTANT (POSITION spring))
                          (CONSTANT (POSITION block)))))
(MEETS (SITUATION S3 (SET (CONSTANT (POSITION spring))
                          (CONSTANT (POSITION block))))
       (SITUATION S4 (SET (INCREASING (POSITION spring))
                          (INCREASING (POSITION block)))))
(MEETS (SITUATION S4 (SET (INCREASING (POSITION spring))
                          (INCREASING (POSITION block))))
       (SITUATION S1 (SET (CONSTANT (POSITION spring))
                          (CONSTANT (POSITION block)))))
```

This syntax resulted in 46 gmaps and took a total of 26 seconds to compute. Notice, the time for computing the match alone (sans BMS) dropped from over 29 minutes for the slices notation to under 7 seconds. The apparent redundancy of the description (each situation appears twice) is virtual, not real. To SME, all syntactically identical subexpressions in a description map to the same internal expression. Furthermore, expressions may be named for use as arguments to other expressions. For example, the above description was actually given to SME as four named situation expressions. Their temporal ordering was then given in the same way it was for the other two representations, for example, (MEETS situation1 situation2). situation1 is now a pointer to a situation *description*, rather than a situation token.

These results are summarized in Table 3.1. The difference in speed is primarily due to the operation of merge step 2, which combines matches sharing a common base structure. The set notation for time enables merge step 2 to know that matches for state S1 behavior of the spring-block oscillator should be placed in the same gmap, thus reducing the number of possibilities in merge step 3.

While perhaps somewhat unorthodox, this representation has some desirable properties. First, most expressions are simpler since temporal references are implicit. At least in terms of analogical processing, indexing a situation's facts this way drastically reduces the number of match hypothesis combinations possible. Second, it makes the temporal clustering of relations explicit in the syntax.

Similar conventions have been used in PHINEAS for a number of representation problems, such as the representation of theories, with comparable savings.

The key implication for analogical processing is that *syntax should mirror semantics.* If there is a strong first or second order relationship between two expressions, this relationship should be obvious from the syntax. Such relationships are typically not mirrored in the syntax of standard first order predicate calculus. For example,

```
(Greater-than x y S1)
(Break y S2)
(Cause S1 S2)
```

does not syntactically reflect the important relationship that exists between the Greater-than expression and the Break expression. On the other hand,

```
Cause[Greater-Than(x,y), Break(y)]
```

makes their relationship structurally explicit. Similar arguments have been made in favor of semantic net representations (Winston, 1984), despite their logical equivalence with FOPC (Hayes, 1977).

3.4.3 Generality

There are a number of factors in evaluating the success and generality of a program. While it is important to be able to demonstrate more than one example, and SME has successfully run on over 40, conclusions from sheer number of examples should be limited. For example, in the water flow – heat flow example, if heat flow were replaced by an *isomorphic* description of electrical flow, SME wouldn't know the difference. Should we say these represent two examples, or merely one? How then has tailorability been reduced and coverage determined? First, only a small set of rule files has been used throughout the various studies involving SME. For example, every PHINEAS example presented in this thesis used the same rule set described earler. Second, PHINEAS places the user one step farther from SME by being in charge of generating SME's input and inspecting SME's output. Furthermore, PHINEAS requires that the representations used by SME be able to satisfy a specific performance task. A representation developed to perform useful inferences has fewer arbitrary choices than a representation developed specifically for analogical matching. Finally, SME has demonstrated a high degree of generality through its multiple uses. It has been used in cognitive simulation studies, served as a component in other systems (including PHINEAS), and been configured to emulate other analogy programs. Specifically, SME has been used in:

- *Cognitive simulation of Structure-Mapping theory:* SME has been successfully used in studies comparing the psychological predictions of structure-mapping theory (Skorstad el al., 1987). It has also been used to apply the concepts of Structure-Mapping theory to metaphor understanding (Gentner et al., 1987).

- *As a component in* SEQL*:* Janice Skorstad has used SME as a component in SEQL, a concept learning program that forms generalized structural concept descriptions from a sequence of examples (Skorstad et al., 1988; Skorstad, 1989). SEQL has further been used for studying sequence effects in concept formation.

- *Simulating* SPROUTER*:* Hayes-Roth and McDermott (1978) describe a technique for partial matching of structural descriptions called *interference matching.* Their SPROUTER program uses the matching to form a generalized conjunctive concept description of a set of target examples. SME and the generalization module it contains has successfully reproduced the first two (out of three) examples discussed in (Hayes-Roth & McDermott, 1978).[8] The third example has never been tried. (Falkenhainer, 1988a) briefly discusses how SME may be configured to emulate SPROUTER.

[8]SME contains a module that takes a gmap produced by the matching component and returns three alternate, generalized conjunctive concept descriptions covering the base and target instances (see Falkenhainer, 1988a). The three alternatives correspond to (1) only what base and target have in common identically, (2) everything base and target have in common (e.g., PRESSURE matched to TEMPERATURE turns into FUNCTION-3), and (3) everything base and target have in common, plus the candidate inferences proposed for the target.

- *Simulating* ACME*:* Holyoak and Thagard (1988a) describe a new system, ACME, which may be described as a descendant of SME, with SME's three merge steps replaced by connectionist relaxation techniques. An ACME rule set has been used to reproduce a number of examples described in their paper. This rule set is discussed further in (Falkenhainer, 1988a) and the two programs are compared in Section 10.2.1.5.

Chapter 4

Access

Access is the process of reminding and recognition. In the context of physical analogies, the first step towards explaining a newly encountered observation is attempting to relate it to understood situations. Is it an instance of an understood phenomena? Could it be if a few assumptions were made? Is it *similar* to an understood phenomena? For analogical learning, the goal is to retrieve theories most likely to provide an accurate explanation.

Analogical access has proven to be a difficult problem for AI and few analogy systems address it. As reviewed in Section 2.1, access is typically a matter of being presented with a complete base representation, some form of specific cue, as in a teacher supplied hint (Burstein, 1983; Greiner, 1988), or the specific goal concept (Kedar-Cabelli, 1985b). Most work on access falls under case-based reasoning (e.g., Kolodner, 1984; Ashley & Rissland, 1987). In these systems, access typically proceeds by indexing the features of the situation description into a memory organized as a discrimination net or decision tree. These techniques have two problems. First, they tend to be sensitive to incomplete information and the ordering of the discrimination tree. Second, they limit retrieval to looking for features that match exactly. This may be appropriate for case-based reasoning, which can be considered as a form of within-domain analogy, but is insufficient for across-domain learning.

Psychological evidence provides two suggestions about access. First, analogical accessibility tends to be governed by surface similarities, also known as *mere-appearance* matches (Ross, 1984; Gentner & Landers, 1985). This is the kind of recall we wish to avoid in expert problem solving. Surface similarities are not necessarily *predictive*. Yet how can we a priori know what will be predictive if at the same time we're trying to learn that prediction-generating knowledge?

Second, in human processing of physical analogies and many types of problem solving, *imagery* seems to play a major role in both accessibility and evaluation (Dreistadt, 1968;

Dreistadt, 1969; Kaufmann, 1979; Kosslyn, 1980; Shepard & Cooper, 1982; Miller, 1986). Some go so far as to say "analogy production related to problem solving is a visual process" (Kaufmann, 1979, page 119). These studies show imagery has implications for remindings as well as evaluation – the ability to "try out" a proposed solution before actually acting on it, an aspect discussed more in Chapter 6.

While our goal is not to build a psychological model, the second suggestion will prove useful. First, visual processing appears to have the ability to abstract and store vast amounts of information and detect patterns that are relatively easy to manipulate and recall. Second, time varying, dynamic behavior is highly predictive of underlying causal mechanisms, sometimes more so than an incomplete, static situation description. Both factors are important for computational studies of analogy. Given current technology, an autonomous account of the first will not be attempted.

This chapter describes the model and implementation of access used in PHINEAS. It is based on the claim that both the behavioral and structural similarity of two phenomena can be used to initiate and guide the mapping of an explanatory causal model.

4.1 Accessing Physical Analogies

In analogical learning, one starts with a partially understood model of a domain, or a teacher-supplied hint which serves the same purpose. This incomplete model is then used to key access and constrain the mapping that serves to complete the model. In learning from observation there is no teacher to provide a hint. If the phenomenon is new, the learner may not even have a partial causal model to drive access. Therefore, some other key into memory is required to constrain the mapping process. The only available information is the observable structural and behavioral characteristics of the situation. What must be specified is how this information can drive access. Specifically, it must

- *provide commonality between systematic pieces of knowledge.* Commonality among richly interconnected relations will maximize the probability that we have a true analogy, rather than a chance, marginal similarity.
- *highlight important aspects*, in order to reduce spurious remindings and improve the chance of finding the most relevant analogue.
- *be predictive.* Time is best spent on hypotheses that have a high chance of being correct.

One component of access is *correlational*: when seeking goal G, select the subset of available information that is known to correlate to G and find an analogue sharing the maximum amount of similar information.

The other component of access is *discriminability*: when seeking goal G, focus on the subset of available information leading to the smallest, yet plausible, analogue candidate set. In other words, seek factors that will allow us to select good analogues from a vast set of possibilities in the minimal amount of time.

Both of these components appear in some form in any decision making process. They may be in the form of probabilities leading to a particular cause for a given episode, as in diagnostic decision making (Pople, 1977), an analysis of how to partition existing data, as in ID3 (Quinlan, 1983), or of a more symbolic or heuristic nature. The important point is that if a portion of available information has a tendency to co-occur with the desired unknown, and tends to only be associated with that unknown, use it as a predictive key into memory. How can this be achieved for the physical analogy task?

The first clue is that models of physical systems are decomposable into different perspectives of the same phenomenon, such as structural, behavioral, causal, and teleological. The amount of knowledge available tends to be different for each. Thus, one element to accessing physical analogies is to recognize that one should key on whatever perspectives are most readily available. For example, suppose a complete behavioral model and a scant causal model exist for a given system. In attempting to form a full causal model, it would be wise for the access mechanism to key on similar behavior rather than trying to find all systems which overlap what little is already known about the target causal model. Because a lot is known about the behavior, it maximizes the relational information available and the descriptive space in which to maximize commonalities. This satisfies an important criterion: seek commonality among richly interconnected relations to increase the likelihood of a true analogy. The problem of drawing distinct boundaries and defining the separate categories isn't really important. The lines may often be drawn differently for different situations and by different people. What is important is that different types of knowledge, or *aspects*, about the same situation tend to have a cross correlation. It is this correlation that I wish to exploit.

A second important feature of access is the use of abstractions. Attempting to locate similar knowledge structures when given a highly detailed model can be needlessly complex. Abstracting out the key features of the detailed model simplifies the task. For example, in trying to access behavior similar to water heating on a stove, it is easier to use an abstract model of flow than a detailed model of changing pressures and moving molecules. Typically, dynamic behavior is most readily abstracted through visual processing and is the first model one has for a new domain. In more expert problem solving, structural or causal models may be abstracted, enabling the discernment of salient features for triggering the model (e.g., "central-force system"). A single phenomenon or model may be represented using multiple, overlapping abstractions. Thus, access should also key on shared abstractions.

This view should not be confused with theories that treat analogies as shared abstractions (e.g., Greiner, 1988). In these approaches, only a single model type exists, forcing the system to find a shared abstraction with the target concept. Analogy is conjectured as equivalent to instantiating a common abstraction. This overly restrictive view of analogy does not capture the breadth of the phenomenon. For example, in this thesis a complete behavioral model is used to conjecture a new causal model. In this manner, the power of using shared abstractions may be achieved without giving up the creative power of analogy.

4.1.1 What types of behaviors?

Not just any behavioral abstraction will do, and occasionally behavior alone is insufficient. For example, in searching for an analogue to a spring-block oscillator, we are more discerning than simply looking for an instance of something "going back and forth". We tend to know that the spring-mass oscillator is a passive system exhibiting a response to some initial perturbation. We also know to prefer mechanical systems such as another spring-mass example or a torsion oscillator (i.e., something oscillating due to elasticity). A person walking to and from school each day should not be seriously entertained as an analogue. Thus, an additional criterion to abstraction-based access is in having the right abstractions. The maximum amount of detail perceptually available should be captured while still having a concise representation.

Consider what information may be recorded about a hot brick immersed in cold water. One particularly useless account would relate just that information: there is some water in a bucket, and a brick in sitting in the bucket, immersed in the water. This isn't going to help too much in explaining the behavior since there is no behavior. A second approach might describe two connected situations, S_1 transitioning to S_2. In S_1, the temperature of the brick is greater than the temperature of the water and the temperature of the brick is decreasing, while the water temperature is increasing. S_2 indicates that their temperatures are equal and constant. This is a better representation of what is happening, yet it still doesn't provide many clues. Finally, we could augment this description with more of its temporal or graphical characteristics. The temperatures of the brick and water are asymptotically approaching each other. This provides a sense of "process". The temperatures approach each other and stop changing when they are equal. The two rates of change appear to be proportional in some manner. This suggests the concepts of exchange and equilibrium.

4.2 Implementation

There are two primary phases in processing a new observation. First, the raw observation must be received, translated into qualitative terms, and analyzed for patterns (e.g., sinusoidal oscillation). Second, the qualitative description is used to key into memory to see if it corresponds to a known phenomenon, or is similar to any understood phenomena. The first component is external to PHINEAS. The second component is the primary topic of this section.

The user translates the raw observation into qualitative values and derivatives (increasing, constant, decreasing), and divides it into qualitatively equivalent temporal intervals. For example, a series of values for quantity q_1 might be represented as $q_1 = 0$, $q_1 > q_2$, and Increasing(q_1) during state S_1, transitioning to $q_1 > 0$, $q_1 > q_2$, and Increasing(q_1) during state S_2. Additionally, the user provides information about global patterns in the data, such as *sinusoidal oscillation* or *asymptotic approach to zero.*

Before discussing the specific access process, a few representational conventions will be described.

4.2.1 Behavioral Segments

We need a way to represent behaviors, potentially at multiple levels of abstraction and from different ontological perspectives. Consider the behavior of alcohol disappearing when left sitting in an open container (Figure 4.1). Two classes of information are recorded (Figure 4.2): the original scenario description (e.g., (Open beaker2) and (Container-of

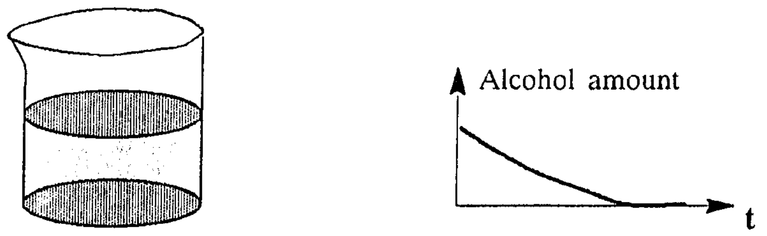


Figure 4.1: An unexplained observation of alcohol disappearing from an open container.

```
(defObservation open-alcohol :new
     Behavior open-alcohol-behavior
     Individuals (alcohol1 alcohol beaker2)
     World ((substance alcohol)
            (contained-liquid alcohol1)
            (container beaker2)
            (container-of alcohol1 beaker2)
            (substance-of alcohol1 alcohol)
            (open beaker2)
            (beaker beaker2)))

(defBSegment open-alcohol-behavior :new
  Characterizations ((matter-movement ?self)
                     (monotonic ?self)
                     (continuous-movement ?self))
  Components (alcohol-going alcohol-dry)
  Relations ((meets alcohol-going alcohol-dry)))

(defSituation alcohol-going :new
  Characterizations ((matter-movement ?self)
                     (monotonic ?self)
                     (continuous-movement ?self))
  Individuals (alcohol1)
  Dynamics  ((Decreasing (Amount-of alcohol1))
             (Constant (Change-rate (Amount-of alcohol1)))
             (Greater-than (A (amount-of alcohol1)) zero)))

(defSituation alcohol-dry :new
  Individuals (alcohol1)
  Dynamics  ((Constant (Amount-of alcohol1))
             (Equal-to (A (Amount-of alcohol1)) zero)))
```

Figure 4.2: Behavioral description of an open container of alcohol. The `defSituation` form identifies a primitive (single state) bseg. The `:new` keyword indicates a new observation, as opposed to a declaration of an old experience. Old experiences have an additional `Processes` field for each situation, providing pointers to instantiations of the theories used to explain it.

alcohol1 beaker2)) and the dynamic behavior across time (e.g., **(Decreasing (Amount-of alcohol1)))**).

Observations are recorded using the **defObservation** form, which names the observation, identifies the individuals involved, and indicates the name of its behavioral description. The scenario description (e.g., structural relationships and objects' properties) appears as the **world** component of an observation.

Behaviors are represented by collections of *behavioral segments* (bseg). A behavioral segment represents a slice through the spatial-temporal plot of an observation. A bseg may represent either a primitive situation, or an extended period of time summarizing a collection of more primitive bsegs. No important distinctions are made between the two types, since what is considered *primitive* depends upon the information available. Added detail will typically expand a situation into finer divisions.

Interrelations between bsegs may be either spatial or temporal in nature. For example, consider the multiple representations of the oscillatory behavior of a spring and block combination shown in Figure 4.3. The upper-most, single state description summarizes the behavior with an *oscillating* bseg. This decomposes into the eight state cycle below it, which is in terms of **Velocity(block1)** $>$ **0**, **Decreasing[Velocity(block1)]**, and **Position(block1)** $>$ **0**, etc. The eight state and single state descriptions are temporally related, in this case through abstraction of eight temporal states into a single temporal state. Differing spatial slices through the representation are possible as well. For example, the eight-state cycle describing velocity and position may be divided into two, four-state cycles, one describing the velocity's behavior, the other describing the position's behavior. Currently, **PHINEAS** uses only the temporal abstraction component of this representation.

Behavioral segments are recorded using the **defBSegment** form:

```
(defBSegment ⟨BsegName⟩ [:new]
   Characterizations ⟨BehavioralAbstractions⟩
   Individuals (⟨Entity1⟩, ⟨Entity2⟩,...,⟨Entityi⟩)
   Explanation ⟨ExplanationSummary⟩
   Components ⟨BSegs⟩
   Relations ⟨TemporalSpatialRelations⟩
   Dynamics (⟨BehavioralRelations⟩) )
```

The **Characterizations** field describes the behavior's abstract properties, such as asymptotic approach. A bseg's **Explanation** field lists the process and entity instances that are active during its duration. This field is only used for the declaration of past experiences, in which these theories are taken from the accepted explanation given the bseg. For example, the **Explanation** field of an explained dissolving observation appears as:

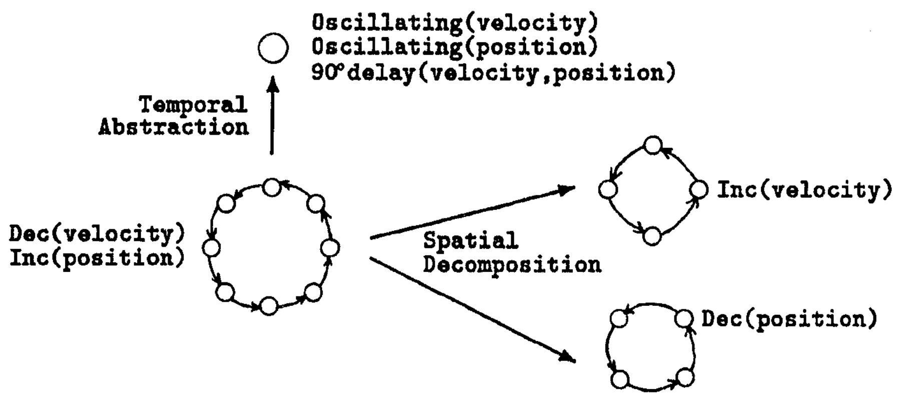


Figure 4.3: Multiple temporal and spatial views of a spring-block oscillator. Each circle represents a temporal interval during which all quantities have a constant value with respect to the level of description in use for that view.

```
Explanation ((dissolve pi1 ((?solute . salt1)(?solution . water1)))
             (solution ((?solution . water1))))
```

which indicates that the dissolving process was active, the process instance was called `pi1`, and the two individuals described by dissolving were `salt1` and `water1`. Further, the entity declaration `solution` was active. The `Components` field identifies the bsegs it temporally or spatially summaries. The `Relations` field describes the temporal or spatial interrelations between the components (e.g., `Meets` specifies temporal ordering). The `Dynamics` field records the qualitative value and derivative information of the observed behavior. The `:new` option indicates a new observation, as opposed to a declaration of an old experience. Old experiences are automatically stored in memory, while new observations are the targets for explanation.

4.2.2 Behavioral Abstractions

We also need a way to organize behaviors in memory so that they may be retrieved later. This involves indexing them using a number of different keys, such as the perceptual primitives associated with them, what their gross effects are, and the domains and situations they apply to.

Behavioral segments are indexed in memory via multiple *behavioral abstraction* hierarchies. Each tree represents an *isa* hierarchy of abstract behavioral characterizations. They

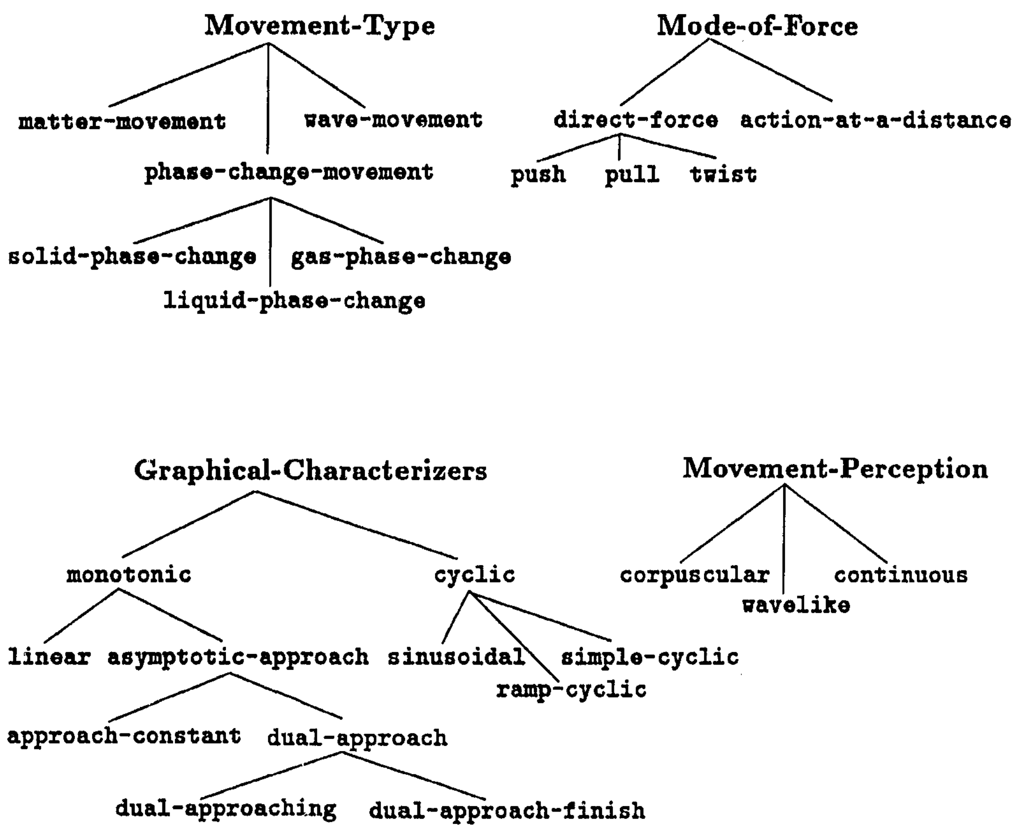


Figure 4.4: Behavioral abstraction forest.

are intended to relate those aspects of a behavior that are perceptually or analytically available, yet not easily describable using the standard value and derivative notation of qualitative quantities (e.g., positive, increasing, etc.).

The entire set of behavioral abstractions used in PHINEAS is shown in Figure 4.4. There are four primary categories:

- *Graphical characterizers.* These may be considered either visual or graphical characterizations. At the highest level are *linear*, *cyclic*, and *asymptotic approach.* The further specializations of these are shown in Figure 4.4. For example, asymptotic approach might describe approaching a constant, or two quantities asymptotically approaching each other.

- *Movement perception.* These describe perceptual abstractions of a behavior. *Cor-*

puscular movement describes a solid object in motion, whereas *continuous* movement describes continuous transfer of matter or energy, as in liquids flowing. Corpuscular movements may involve spinning, revolving, reflecting, bouncing, twisting, compressing.

- *Movement type.* This describes the general underlying mode of movement: matter simply moving its location or orientation, movement of a wave front through space, or matter moving due to a change in state.

- *Mode of force.* This describes the source of force being used, if readily obvious to an observer. These are *action-at-a-distance*, or some type of *direct* or contact force (e.g., push, pull, twist).

Each bseg is multiply indexed in memory under the set of behavioral characterizations associated with it. For example, an observation of liquid flowing between two containers may be found under three different abstract characteristics: `matter-movement`, `continuous-movement`, and `dual-approach-finish` (Figure 4.4). It is important to note that a bseg need not be indexed under every one of the four category types listed above. Typically, this is not the case. For example, the *mode of force* category is often not included in a description.

4.2.3 The Access Process

Memory consists of a library of previously observed phenomena (i.e., situation and behavior descriptions) and a collection of qualitative theories about physical processes (e.g., liquid flow), entities (e.g., fluid paths), and general physical principles (e.g., mechanical coupling).[1] Past reasoning traces are summarized by storing with each situation in an observation the instantiated collection of theories (process definitions, etc.) that were used to explain it. This enables all of the relevant information needed to explain an observation to be linked with the observation in memory, without incurring the overhead of storing full, detailed explanations. Were such an explanation needed, it could be easily regenerated from the available information. For example, the statement `Liquid-Flow(beaker1,vial8,pipe2)` would be stored with an observation of liquid flowing from `beaker1` to `vial8`, indicating an instance of the `Liquid-Flow` process.

The problem of access limited to exact feature match is avoided through a two stage process. Shared abstractions are used to focus attention on a potentially relevant subset of

[1]PHINEAS's domain knowledge is listed in Appendix C. Its set of *a priori* experiences is listed in Appendix D.

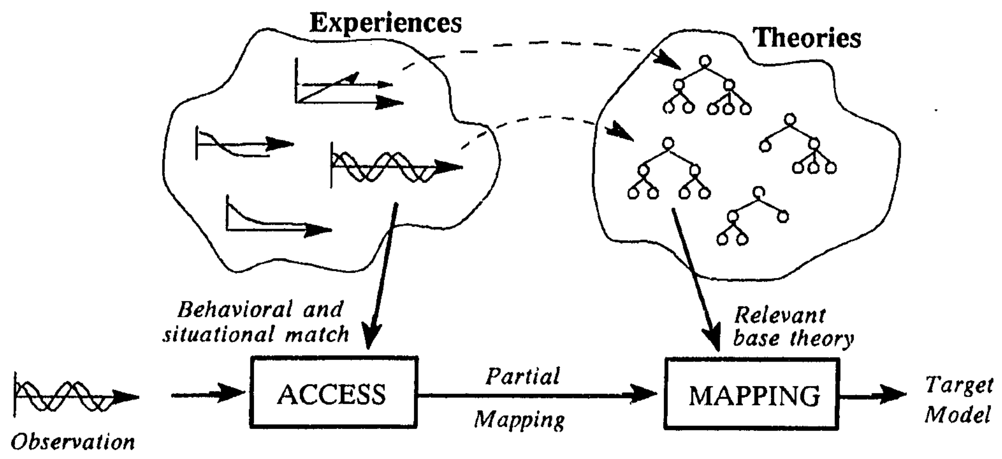


Figure 4.5: The PHINEAS access process and adjacent modules.

memory. Each experience in this subset is then compared at a detailed level to the current situation using SME.

When a new observation is encountered, each bseg is integrated into memory via its given behavioral abstractions. In doing so, past behaviors sharing those abstractions are "touched", that is, their numeric activation levels are incremented. An extremely simple activation scheme is used in PHINEAS. For each behavioral abstraction describing a newly observed bseg, the activation of every existing bseg in memory also described by that behavioral abstraction is incremented by a value of one. If there are no known instances of that specific abstraction, then instances are sought under each more general node in the hierarchy until instances are found. Increasing the level of abstraction produces greater distance between the current situation and candidate instances. Thus, the amount of activation added to a discovered bseg should decrease with distance from the observed behavioral classification. In PHINEAS, the activation weight is dropped by one-tenth for each node traversed. For example, an observation of evaporation might be classified as a type of `fluidic-phase-change-move`, while a past observation of dissolving might be classified as a type of `solid-phase-change-move`. Both are types of `phase-change-movement`. Thus, installing evaporation under `fluidic-phase-change-move` would cause the activation level of dissolving to be incremented by 0.8. An initial set of candidate analogues is then obtained by collecting the N most activated behaviors, where N is a specified beam size (currently 15).[2] These behaviors are those experiences sharing the greatest number of abstract, characterizing features. Temporal subsumption is checked during this operation, so that a general bseg is preferred over the set of bsegs it summarizes.

Each prior experience in the initial candidate set is then inspected more carefully by matching its detailed structural and behavioral description with the current situation. SME is used for this operation, producing an initial partial mapping and an evaluation score for each candidate analogue experience. This partial mapping provides an indication of what objects and quantities correspond by virtue of their behavioral similarity and will serve as an important source of constraint during the mapping process.

The match indicates where the behaviors correspond and thus what portion of the analogue behavior should be considered relevant. The problem of relevant theory selection is solved by retrieving only those domain theories that had been used to explain the matched portions of the analogue situation. Each bseg indicates what processes were active during that state. Thus, if the current observation only matches a subset of the bsegs in the analogue observation, only the relevant process models are used.

The candidates are then ordered according to SME's evaluation score and proposed one at a time as results from the access task of PHINEAS' global agenda. If more candidates are required beyond the initial N, the access task may always resume where it left off and examine those remaining. However, the ability to resume a suspended access task is not utilized at the present time.

4.3 Disappearing Alcohol Example

The access mechanism will now be reviewed in the context of a detailed example. In this example, PHINEAS is given time-ordered measurements of a situation in which the amount of alcohol sitting in an open beaker is seen to continually decrease, with the beaker eventually empty (Figure 4.1). The complete observation description appears in Figure 4.2. The system begins with knowledge of eight processes – liquid flow, liquid drain (to constantly empty an ideal sink), heat flow, boiling, heat-replenish (e.g., to constantly maintain the heat of a stove), dissolve, motion, and spring-applied force. It also has a database of physical observations fully explained by these processes.

The problem is to propose an explanation for this observation, given PHINEAS' current breadth of knowledge. First, the behavioral abstractions describing the observation are used to probe memory. In this case, there are three. First, `matter-movement` indicates

[2]Due to PHINEAS' currently impoverished set of experiential knowledge, the beam size of 15 is not a factor in any of the examples discussed in this thesis. Typically, only 3 to 4 analogues are retrieved.

that matter is moving, for the alcohol is disappearing. This directly activates two different liquid flow observations, one of liquid draining from a leaky cup and the other of flow between two containers. Second, continuous-movement indicates that the movement is happening gradually over the total mass of the alcohol, as opposed to the alcohol moving as a whole. This activates a number of experiences: a leaky cup, liquid flow between two containers, salt dissolving in water, and a pot of boiling water. Finally, the behavior is monotonic, as opposed to cyclic. This activates the leaky cup experience, dissolving, and boiling. Taking all of those that have a non-zero activation level (which is less than the beam size), we have the leaky cup situation, salt dissolving in water, a pot of boiling water, and liquid flow between two containers.

The second stage of access examines this subset of memory in more detail, using SME to establish and evaluate each match. SME produces the following observation and evaluation score pairs:

BOILING-BEHAVIOR (15.7) The alcohol disappearing is similar to what would happen if it were boiling. This match is shown in Figure 4.6.

LIQUID-DRAINING-BEHAVIOR (14.9) In this behavior, a leaky cup is found to correspond to the beaker and the water leaking out corresponds to the alcohol that is disappearing.

DISSOLVE-BEHAVIOR (12.6) Salt dissolving in water behaves similarly to the disappearing alcohol.

2-CONTAINER-LF (11.1) Water flowing out of one container and into another until pressure equilibrium is reached has a similar behavior, although it ends with liquid still present in the source container.

At this point, the access task returns the boiling analogue as the best behavioral and structural match for the current situation and passes it to a mapping task. The remaining three analogues are each assigned a mapping task as well, with their match scores used to establish priority.

4.4 Perspective

Access is a difficult problem. Part of the difficulty is methodological, in that it is hard to build programs with the same volume and variety of experience we receive.

Boiling History **Disappearing Alcohol History**

```
(Situation boiling-bseg                         (Situation alcohol-going
  (Set (Constant (Temperature stove9))            (Set (Decreasing (Amount-of alcohol1))
       (Constant (Temperature bwater1)                 (Constant
       (Equal-to (A (Temperature stove9))                (Change-Rate (Amount-of alcohol1)))
                 (A (Temperature bwater1)))            (Greater-than (A (Amount-of alcohol1))
       (Greater-than (A (Amount-of bwater1))                         zero)))
                     zero)
       (Constant (Change-Rate (Amount-of bwater1)))
       (Constant (Change-Rate (Amount-of bsteam1)))
       (Decreasing (Amount-of bwater1))
       (Increasing (Amount-of bsteam1))))

(Situation boiling-dry                          (Situation alcohol-dry
   (Set (Constant (Amount-of bwater1))            (Set (Constant (Amount-of alcohol1))
        (Equal-to (A (Amount-of bwater1))              (Equal-to (A (Amount-of alcohol1))
                  zero)))                                        zero)))
        (Greater-than (A (Amount-of bsteam1))
                      zero)))
(Meets situation1 situation2)                   (Meets situation1 situation2)
```

Entity & Quantity Correspondences

bwater1	↔	alcohol1
pan7	↔	beaker2
water	↔	alcohol
Amount-of	↔	Amount-of
Change-Rate	↔	Change-Rate
boiling-bseg	↔	alcohol-going
boiling-dry	↔	alcohol-dry

Figure 4.6: SME$_{CSM}$ match between the disappearing alcohol and a boiling pan of water.

Several factors have been left out of PHINEAS that seem important. First, the more novel a situation and less it is readily visible, the harder recognition. Data must be collected and analyzed, patterns sought, and overall familiarity increased. At some point, enough details may be in place to trigger recognition, reorganization, and insight (Dreistadt, 1968). In PHINEAS, the access task is greatly simplified, with enough information provided by the user to find relevant candidate analogues. For example, the fundamental clue leading to the caloric theory of heat, namely that the temperatures reach equality (Roller, 1961; Wiser & Carey, 1983), is part of the behavioral description given to PHINEAS. Currently, there is no mechanism in PHINEAS to model this gradual buildup of experiences. See

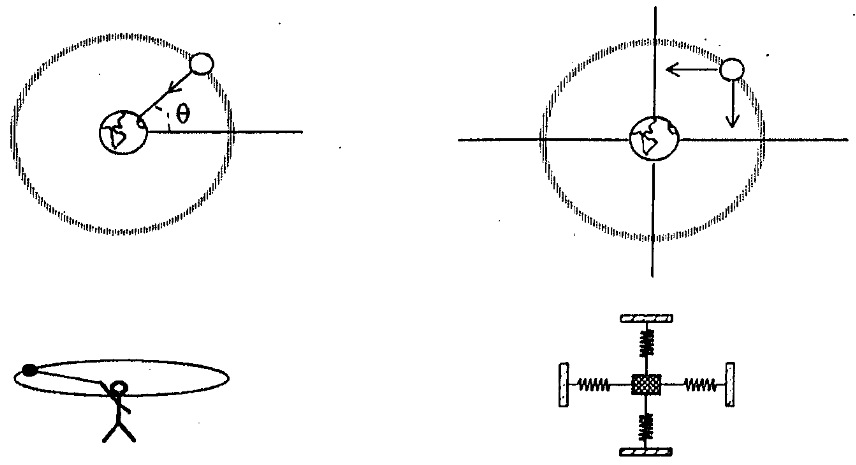


Figure 4.7: Two views of the moon orbiting the earth.

(Dreistadt, 1968, 1969) for a discussion of this phenomenon in human reasoning, and (Langley & Jones, 1988) for an initial computational framework.

A second problem is the translation of raw input data to qualitative patterns (e.g., sinusoidal oscillation). PHINEAS has the ability (using Decoste's (1989) *measurement interpretation* system) to translate a continuous, real valued measurement sequence into a succession of equivalent states capturing each quantity's qualitative value and derivative. However, it does not have the ability to automatically analyze the data and identify the appropriate behavioral abstractions (e.g., sinusoidal oscillation). For this reason, actual real-valued measurement sequences are not currently used. At some point this should be automated.

Just as important as behavioral abstraction recognition is behavioral abstraction development. The abstractions used in PHINEAS are theory laden and greatly simplify memory indexing. What is needed is a richer vocabulary for describing behaviors and situations.

Finally, the ability to change perspective on a set of data appears crucial to successful scientific explanation. For example, consider attempting to explain the moon's motion around the earth. Different perspectives can lead to different answers. In Figure 4.7(a), the moon is viewed from a polar coordinate system, which supports an analogy with a ball spun about on a string. However, a cartesian view of the situation, with x and y coordinates oscillating (Figure 4.7(b)), supports a (rather odd) analogy with a pair of spring systems.[3]

[3]I owe this double-spring possibility to John Collins.

PHINEAS currently does not reason about multiple perspectives of a situation, a difficult modeling problem in general (see (Falkenhainer & Forbus, 1988) for initial work in this area).

Chapter 5

Mapping and Transfer

Mapping and transfer form the center of the analogy process. Mapping represents the primary analogical act: establish a complete set of correspondences and propose any inferences these correspondences sanction. Transfer represents the first evaluation phase for the results of mapping: attempt to transfer the inferences mapping proposes into the target domain. This chapter discusses these two intimately related processes. It begins by framing their respective roles in the larger analogy process and how they may interact. It then discusses each, along with a detailed example demonstrating their operation.

5.1 Overview

The term "mapping" used in reference to the correspondence phase of analogy has its origins in formal mathematics and may be defined as follows:

Definition 5.1 (Analogical Mapping) *Take $\mathcal{B}$ to represent the set of individual items from the base domain description (i.e., predicates, expressions, and entities), and $\mathcal{T}$ to represent the set of individual items from the target domain description. An* analogical mapping, *$M: \mathcal{B} \rightarrow \mathcal{T}$, is a partial function that associates elements of $\mathcal{B}$ to elements of $\mathcal{T}$.*[1]

Thus, an analogical mapping identifies a correspondence between some elements of the base domain description and elements of the target domain description. This aspect of mapping is sometimes called its *match* component. The mapping may also sanction a set of *candidate inferences*: base information that may plausibly be applied to the target domain. This aspect of mapping is sometimes called its *carryover* component (Gentner, 1988). The

[1]This is equivalent to definitions appearing in (Gentner, 1983; Indurkhya, 1987).

use of the term "carryover" is perhaps unfortunate, for it seems to imply that it results in actual, usable target inferences. However, rather than replace the term, I will afford it a more precise meaning:

Definition 5.2 *Given $\mathcal{I}_B$, the set of base information proposed as inferable for the target, the carryover operation produces the set of candidate inferences $\mathcal{CI}$ sanctioned by mapping by applying the existing mapping function to $\mathcal{I}_B$:*

$$\mathcal{CI} = M(\mathcal{I}_B)$$

Candidate inferences are hypotheses which must pass a series of evaluative processes before being accepted as holding for the target domain. A distinction may be made between *usable* inferences and *useful* or *adopted* inferences. Usable inferences are operational, that is, they apply predicates in a manner consistent with their intended semantics, they produce no immediate contradictions, and there are no syntactic holes (i.e., unknown objects are either found or conjectured). Useful or adopted inferences have been tested through further evaluation or problem solving and found to be "satisfactory" under some criterion.

Importantly, the set of candidate inferences proposed by mapping need not be usable. Thus, the first stage in the evaluative processes is *transfer*, which attempts to form a working hypothesis from a potentially fragmented candidate inference. Significant adaptation may be required before a candidate inference represents usable target domain information. First, since base relations are being imported, allowing cross-domain analogies means they may apply predicates to objects or propositions other than their conventional referents. Additionally, the inferences may suggest facts that contradict what is explicitly known about the target. Substitutes for such expressions must be found or created. Second, these inferences may also contain unknown, anticipated objects: slots occupied in the base representation by objects that had no correspondent in the match. A corresponding target object must be found, or the existence of the unknown object postulated.

An overview of the mapping and transfer phases is shown in Figure 5.1.

In addition to base and target descriptions, the mapping box allows contextual knowledge, if supplied, to influence its operation. Contextual knowledge may include the current plans and goals of the performance element or any associations already established. These associations may arise either from teacher supplied hints or prior processing, such as access.

There may be strong interplay between mapping and attempts during transfer to form a coherent conclusion from its statement of correspondence. This is in part due to the non-monotonic binding problem discussed in Chapter 2. In realistic memories, mapping will be operating on a subset of all that is inferrably known about the base and target. Since

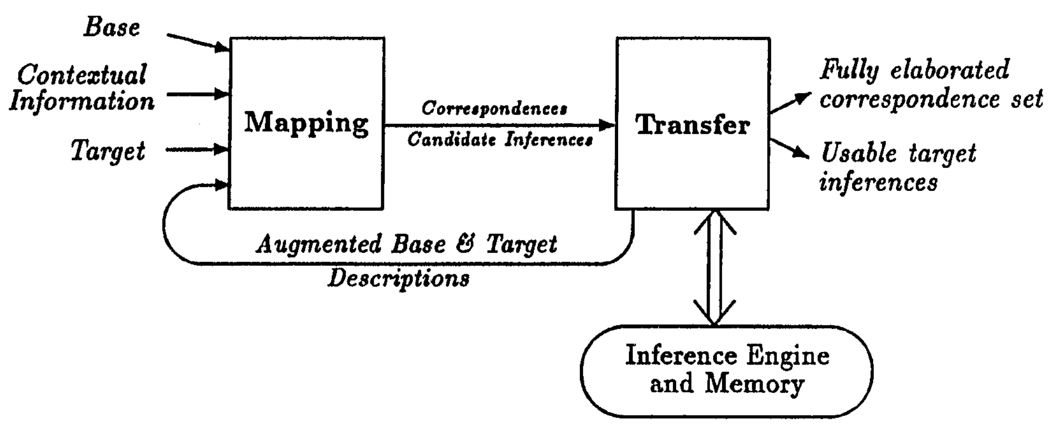


Figure 5.1: Overview of the mapping and transfer phases of the analogy process.

a candidate inference is with respect to the subset of the base or target being processed, it need not represent new knowledge. Rather, it may indicate places where additional knowledge should be retrieved to help complete the mapping.

Transfer conducts focused probes into memory to seek more information about places where the similarity match was incomplete. If additional knowledge is found, transfer will augment the existing base and target descriptions and iterate back to mapping, to see if the new information will affect the overall mapping. Mapping and transfer combine to form a *map and analyze* cycle to provide focus to the analogy process. Were everything possibly known about the base and target retrieved prior to mapping, a great deal of time might be spent on unconstrained inferencing. Conversely, were no match performed and relevant base information simply carried into the target, little focus would be available for transforming the candidate inference into useful target information. By initiating the match on what appears to be relevant from immediately available information, transfer may focus on seeking the specific information the mapping indicates is lacking.

Typically, mapping and transfer will proceed in a simple sequential manner. In within-domain analogies, transfer may provide little service beyond blessing the proposed inferences as usable. In cross-domain analogies, particularly when very little is known about the target domain, transfer may require many map and analyze cycles.

5.2 Contextual Structure-Mapping

Due to the importance of mapping in elaborating the various aspects of the analogy process, contextual structure-mapping was discussed in Chapter 2. In this chapter, it is placed within PHINEAS' overall process by way of example.

5.2.1 Example: Disappearing Alcohol

The second stage in PHINEAS is *theory generation*: produce a fully operational initial hypothesis about the current observation from an analogue retrieved during access. This has two components, mapping and transfer, of which the first will be demonstrated here.

The mapping operation in PHINEAS consists of three steps:

1. Determine base and target representations.
2. Declare sanctioned pairings.
3. Invoke SME.

Before mapping may commence, we must determine the base and target inputs to mapping. Recall that past reasoning traces are summarized by storing with each bseg in an observation the instantiated collection of theories (process definitions, etc.) that were used to explain it. The problem of knowing what to map is solved by retrieving the relevant domain theory which led to prior understanding of the base observation. If the current observation only matches a subset of the bsegs in the old observation, only the relevant theories for that subset are used as the base description.[2] The current observation description (the target) consists solely of its original scenario description (e.g., structural relationships and objects' properties).

For example, the previous chapter described how an observation of alcohol disappearing from an open container triggered a reminding of four similar experiences in the following order: liquid-draining-behavior, boiling-behavior, dissolve-behavior, and 2-container-lf. Resuming that example with the dissolving analogue, we find that the two behaviors fully matched, so both dissolving bsegs are relevant.[3] The first dissolving bseg, `dissolving`,

[2]If PHINEAS supported spatial decomposition of behaviors, selection could be further specialized. Rather than all information about a temporal interval, information about specific behavioral aspects of that temporal interval could be selected.

[3]"fully matched" means that for each bseg in the base behavior, there is a unique bseg in the target behavior that SME found to possess similar aspects. Recall that due to relational groups, two bsegs need not exhaustively match on all relations.

described the disappearance of `salt1` and was explained by a solution entity and an active dissolving process. The second bseg, `dissolve-stopped`, described a state with no salt left in the glass and dissolving inactive. This explanation summary is represented using the `B-Explains(`*theories*`,`*bseg*`)` form. Both base and target representations for this example are shown in Figure 5.2.[4] Two types of forms appear as elements of *theories*:

```
(Process-Definition process-name instance-name (Implies conditions effects))
(Packet-Definition header body)
```

The `Packet-Definition` is used to represent a packet of information (e.g., predicate schema, QP theory defEntity). Objects for which the packet header holds have the properties listed in the body ascribed to them.

The second step to theory mapping explicitly declares the partial mapping determined during access. Access established that certain properties from the two situations behave in the same way, and correspondences between entities or between their quantities (e.g., `Pressure` and `Temperature`) were noted. Prior to invoking SME, each of these correspondences is declared as a *sanctioned pairing*, requiring SME's match rules to respect these established pairings. In the disappearing alcohol example, there are two: `salt1` maps to `alcohol1` and `amount-of` maps to `amount-of`.

The final step to theory mapping is the actual mapping. SME is given the base and target descriptions and the set of sanctioned pairings. Additionally, the SME_{CSM} rule file contains the *provides relevant inference* rule, which favors mappings supporting the inference `B-Explains(`*theories*`,`*target-behavior*`)`. SME's output is shown in Figure 5.3. The initial explanation of the disappearing alcohol observation appears in the candidate inferences field. This model states that there is some process analogous to `dissolve` which is removing the alcohol at a rate proportional to its surface area. Note that the model is not operational at this stage. First, it contains the unknown `(:skolem water1)`, which indicates there is no object in the alcohol scenario corresponding to `water1` in the dissolving scenario.[5] Additionally, it proposes the condition `(Solid alcohol1)` on the alcohol, which clearly is false. Evaluation and adaptation is required before the model will be usable.

This example demonstrates several points. First, the mapping is composed almost entirely of candidate inferences, since the system had no prior model of evaporation. Hence,

[4]The base representation contains the expression (Qprop *quantity*$_1$ *quantity*$_2$). This is QP theory syntax indicating that *quantity*$_1$ is *qualitatively proportional to quantity*$_2$. All else being equal, *quantity*$_1$ increases when *quantity*$_2$ increases and decreases when *quantity*$_2$ decreases. Qprop– indicates *inversely proportional to.*

[5]The other skolem objects appearing in the candidate inference are expected by PHINEAS and handled by another mechanism. For example, when a new process is being conjectured, there will not be a correspondent for the process name or process instance name.

Base: *Dissolving explanation*

```
(B-EXPLAINS
    (SET (PACKET-DEFINITION (SOLUTION WATER1)
             (SET (QUANTITY (CONCENTRATION WATER1))
                  (QUANTITY (SATURATION-POINT WATER1))
                  (NOT (LESS-THAN (A (CONCENTRATION WATER1)) ZERO))
                  (NOT (LESS-THAN (A (SATURATION-POINT WATER1)) ZERO))))
         (PROCESS-DEFINITION DISSOLVE PI1
             (IMPLIES
                (AND (INDIVIDUAL SALT1 (CONDITIONS (SOLID SALT1) (SOLUBLE SALT1)))
                     (INDIVIDUAL WATER1
                                 (CONDITIONS (SOLUTION WATER1)
                                             (IMMERSED-IN SALT1 WATER1)))
                     (SOLUBLE-IN SALT1 WATER1)
                     (GREATER-THAN (A (AMOUNT-OF SALT1)) ZERO))
                (AND (QUANTITY (DISSOLVE-RATE PI1))
                     (QPROP (DISSOLVE-RATE PI1) (SURFACE-AREA SALT1))
                     (GREATER-THAN (A (DISSOLVE-RATE PI1)) ZERO)
                     (CTRANS (AMOUNT-OF SALT1) (CONCENTRATION WATER1)
                             (A (DISSOLVE-RATE PI1))))))
   DISSOLVE-BEHAVIOR)
```

Target: *Disappearing alcohol scenario*

```
(SUBSTANCE ALCOHOL)
(CONTAINED-LIQUID ALCOHOL1)
(CONTAINER BEAKER2)
(CONTAINER-OF ALCOHOL1 BEAKER2)
(SUBSTANCE-OF ALCOHOL1 ALCOHOL)
(OPEN BEAKER2)
(BEAKER BEAKER2)
```

Figure 5.2: Base and target representations in initial theory mapping of the dissolving explanation for the disappearing alcohol observation.

```
               SME Version 2E
     Analogical Match from DISSOLVE-BEHAVIOR-THEORY to OPEN--ALCOHOL-BEHAVIOR-THEORY.

Rule File: ssm.rules
------------------------------------------------------
# MH's | # Gmaps |      1st,2nd,Worst     | RelGroups |
    2  |     1   |  0.99 /  0.99 /  0.99 |  ACTIVE   |
------------------------------------------------------
Total Run Time:     0 Minutes,  0.509 Seconds
BMS Run Time:       0 Minutes,  0.032 Seconds
Best Gmaps: { 1 }

Gmap #1:   (SALT1 ALCOHOL1)  (AMOUNT-OF-299 AMOUNT-OF-320)
  Weight: 0.9920
  Candidate Inferences:
      (B-EXPLAINS
          (SET (PACKET-DEFINITION (SOLUTION (:SKOLEM WATER1))
                  (SET (QUANTITY (CONCENTRATION (:SKOLEM WATER1)))
                                 (QUANTITY (SATURATION-POINT (:SKOLEM WATER1)))
                                 (NOT (LESS-THAN (A (CONCENTRATION (:SKOLEM WATER1))) ZERO))
                                 (NOT (LESS-THAN (A (SATURATION-POINT (:SKOLEM WATER1)))
                                                ZERO))))
               (PROCESS-DEFINITION (:SKOLEM DISSOLVE) (:SKOLEM PI1)
                  (IMPLIES
                       (AND (INDIVIDUAL ALCOHOL1
                                (CONDITIONS (SOLID ALCOHOL1) (SOLUBLE ALCOHOL1)))
                            (INDIVIDUAL (:SKOLEM WATER1)
                                (CONDITIONS (SOLUTION (:SKOLEM WATER1))
                                            (IMMERSED-IN ALCOHOL1 (:SKOLEM WATER1))))
                            (SOLUBLE-IN ALCOHOL1 (:SKOLEM WATER1))
                            (GREATER-THAN (A AMOUNT-OF-320) ZERO))
                       (AND (QUANTITY (DISSOLVE-RATE (:SKOLEM PI1)))
                            (QPROP (DISSOLVE-RATE (:SKOLEM PI1)) (SURFACE-AREA ALCOHOL1))
                            (GREATER-THAN (A (DISSOLVE-RATE (:SKOLEM PI1))) ZERO)
                            (CTRANS AMOUNT-OF-320 (CONCENTRATION (:SKOLEM WATER1))
                                    (A (DISSOLVE-RATE (:SKOLEM PI1))))))))
          OPEN-ALCOHOL-BEHAVIOR)
```

Figure 5.3: SME$_{CSM}$ output mapping dissolving theories into the disappearing alcohol situation. An analogically inferred candidate model of the alcohol "dissolving" appears in the candidate inference slot of the gmap.

the model was *constructed* by analogy rather than augmented by analogy. This shows the power of SME's candidate inference mechanism. Second, the entity and function correspondences provided by the behavioral similarity provide significant constraint for carrying over the explanation. SME's rule-based architecture is critical to this operation: the SME_{CSM} rules only allow hypotheses consistent with the specific entity and quantity correspondences previously established. Entities and quantities left without a match after the accessing stage are still allowed to match other unmatched entities and quantities.

5.3 Transfer

The evaluative processes have several phases. The first of these is transfer. Transfer is concerned with *importing* candidate inferences into the target domain and making them *operational.* This centers around two issues: (1) ensuring consistent expression use and (2) resolving skolem objects. Since creating new predicates and conjecturing unknown objects is undesirable, effort is aimed at searching through memory to find possible resolutions of these points of conflict. Objects satisfying an unknown's stated conditions yet absent from the original target description may be found. Alternate beliefs or predicates may be found as substitutes for inconsistent expressions, which may in turn lead to previously unconsidered objects satisfying these alternate conditions. If new information is found, mapping is repeated to see how it affects the overall mapping. It could lead to a new mapping as opposed to an extension (c.f. the N-M binding problem).

This section begins by discussing methods that address specific aspects of the expression consistency and skolem object issues. It then uses these as primitive operations in describing the transfer process as a whole and how it may interact with mapping. It concludes with a continuation of the disappearing alcohol example to demonstrate the transfer procedure.

5.3.1 Inference engines and abductive retrieval

PHINEAS' domain knowledge, beliefs, and inferences rules are maintained using an *assumption-based truth maintenance system* (ATMS) (de Kleer, 1986a) and an associated problem solver (ATRE), using the problem-solver protocol described by deKleer (1986b, 1986a).[6] The ATMS is crucial to PHINEAS' operation, as it allows simultaneous consideration of multiple, mutually inconsistent theories about the world. In an ATMS, a *context* is a set of assumptions combined with the set of all beliefs derivable from those assumptions. When

[6]The ATMS and associated rule engine, ATRE, used in PHINEAS are enhanced versions of programs written by Ken Forbus for his *Building Problem Solvers* course.

PHINEAS is first initialized, its *a priori* domain knowledge is taken to be universally true. When a new theory is proposed, all of its consequents are made dependent upon the assumption that the theory holds. Reasoning about the consequences of a particular theory $\mathcal{T}_i$ is performed by reasoning in the context of the assumption `Holds(`$\mathcal{T}_i$`)`.

In support of the various transfer operations, the forward chaining ATRE rule system is paired with an *abductive retriever.* This is a backward chaining, breadth-first prover akin to the *deductive retriever* described in (Charniak et al., 1980), with one important difference. Abductive assumptions are allowed when deductive reasoning over existing beliefs is insufficient to derive a specified request pattern. Stated formally, the abductive retriever's task given wff ρ is to find a substitution θ associating all variables in ρ to known entities such that there exists an ATMS context in which `(Consistent` $\rho\theta$`)` is believed.

When a new datum is entered in the database, ATRE exhaustively runs all rules made executable by the datum's presence in a forward manner. Thus, if a proposition is not believed true or false, its status is not derivable from existing beliefs. When the abductive retriever is invoked on a request pattern, it alternates between checking the belief status of the current goal node and backward chaining if the status of that goal node is unknown. Two operations remove the apparent redundancy with ATRE. First, if all subgoal attempts to show a current goal fail, that goal is assumed true if it is consistent to do so. A goal may only be assumed if it is a ground wff (i.e., no variables). Logical consistency is constrained by inference rules, the typed logic, and by *closed world assumptions* (Reiter, 1978). For example, all spatial relationships, such as `Touching`, are assumed to be either known or false. Second, if only a subset of a conjunctive goal's elements may be shown, the remaining conjuncts are assumed, contingent on their joint consistency (i.e., `(Consistent` $C_1 \wedge \ldots \wedge C_N$`)` is true, where $\{C_1 \wedge \ldots \wedge C_N\}$ denotes the elements of a conjunctive goal of size N). Assumptions, explanation dependencies, and determination of joint consistency are handled through normal ATMS operations. For example, given the goal to show

$$P(?x) \;\wedge\; Q(?x) \;\wedge\; R(?x)$$

with $P(a)$, $Q(b)$, and $R(b)$ believed true, the abductive retriever would return two possibilities:

	Binding	Assumptions
1:	$(\,?X\,.\,a)$	$Q(a) \wedge R(a)$
2:	$(\,?X\,.\,b)$	$P(b)$

Such "assume if consistent" is clearly inadequate as a general theory of abduction, but it has proved adequate for PHINEAS (see also (Charniak, 1988) for a similar approach).[7] A depth bound (default 4) is used to limit rule chaining depth. *Monitors* are used to return sets of answers one at a time[8] and control communication between goals and subgoals (they are derived from Charniak et al's (1980) *generators*). Coordination of conjunctive subgoals to prevent the N-M binding problem is a somewhat complicated process astray of the primary topic of this thesis. The details are described in Appendix A.

5.3.2 Expression consistency

Candidate inferences represent base knowledge applied to target referents. However, vocabulary that was appropriate for the base may not be appropriate for the target case, either due to semantic changes in crossing domains or alternate characteristics of the situations. Therefore, each expression proposed as a candidate inference about the target is checked for consistency in the target environment and adapted if necessary. Three operations support this: *show consistency*, *retrieve alternate expression*, and *create predicate*.

5.3.2.1 Show consistent

Semantic applicability of a predicate may be evaluated by testing if the proposed target instantiation is consistent with the type constraints declared for its arguments (Burstein, 1983). However, sorted logic alone is insufficient to maintain consistency. The new predicate instance may violate what is known about things described by that predicate. There isn't much that can be said about the single argument to `Fluid-Path` except that it must be a physical object. Yet, we know a solid metal bar should not be considered a `Fluid-Path`, an inference proposed if heat flowing through the bar is explained via analogy to liquid flow. To properly tell if the proposed instantiation has the requisite properties, it is necessary to test the expression itself for consistency with existing knowledge. Thus, the `show-consistent` operation uses three tests to determine consistency: (1) each argument is consistent with the declared type of the position it appears in, (2) the proposition is not assumably false by closed-world assumption, and (3) the proposition's negation is not provably true. The `show-consistent` operation is often denoted by the goal to show

`(Consistent` *expression*`)`

[7]As an aside, I would highly recommend temporary installation of such a mechanism into any large reasoning system. Holes in one's "complete" domain theory become amusingly clear when wild assumptions start being made based on what the domain theory deems consistent.

[8]If a direct fetch on the database returns multiple possible instantiations, these possibilities are returned as a set rather than one at a time.

When `show-consistent` is given an expression containing variables, only argument type consistency for ground arguments may be determined (PHINEAS doesn't support pure quantified reasoning). The other two consistency tests are not attempted. For example, (`Immersed-in alcohol1` ?*x*) is inconsistent for any instantiation of ?*x*, since `alcohol1` occupies a slot limited to solids.

A proposed expression will be called *singularly consistent* if it is consistent with existing knowledge when considered in isolation.[9] This is distinguished from *global consistency*, which examines the interrelations of the entire inference set and its ability to provide useful and coherent problem solving performance. Global consistency is the province of post-transfer evaluation, and is discussed in the next chapter.

5.3.2.2 Retrieve alternate expression

Singular inconsistency offers two options. A new predicate may be created or an alternate target correspondent sought. Of course, the latter is preferable to the former since it prevents unconstrained spawning of new vocabulary and provides access to knowledge attached to a known relation. Thus, singular inconsistency establishes the goal of seeking an alternate predicate that may fulfill the intent of the original base expression and form an expression that is either true or consistently assumable. This operation is called `retrieve-alternate-expression` and has two components, corresponding to the two methods of mapping non-identical predicates: paired predicates are functionally analogous or they represent a minimal ascension of the predicate hierarchy.

A functionally analogous expression is found by determining why the base expression was present in the base description and searching for a target expression that can provide the same service. The cache slot of the base's process description supplies the necessary information by stating what consequent prerequisites each antecedent satisfies:

(`Satisfies` *antecedent* (`Prereq` *consequent prerequisite*))

The general prerequisites the expression's base correspondent satisfied are retrieved and mapped to the target by application of the existing mapping function. Depth-bound, exhaustive backchaining is then used to locate all known target propositions deductively supporting any of these prerequisites. For each one found, the expression

(`Supports` *target-proposition prerequisite*)

[9]Greiner (1988) also uses general consistency to test aptness of an inference, but as a substitute for mapping to select target correspondents. Mapping consists of attempting to reinstantiate a base abstraction in the target environment, without further reference to the base instance. I prefer to attempt to use base vocabulary, as this eliminates a lot of needless search if singularly consistent (typical of within-domain analogies) and provides a strong clue when searching for an alternative.

is returned. Additionally, a supports proposition for the original base version of this expression is returned.

If a functionally analogous expression cannot be found, then the predicate hierarchy is searched for a nearest neighbor. This is the minimal ascension operation. The ISA hierarchy is climbed until a predicate is found having argument type constraints consistent with their proposed instantiation.[10] All predicates below this point are then examined. If consistent instantiations exist among these descendants, the following heuristic ordering is used to select a single candidate proposition:

1. The proposition is true.
2. Each argument *satisfies* the type declared for its position and the proposition is consistent.
3. Each argument *is consistent with* the type declared for its position and the proposition is consistent.

Within each category, the predicate closest to the original inconsistent predicate (in terms of number of nodes between them) is chosen.

For example, if `(Contained-in salt1 water)` is present in a candidate inference about dissolving, the mapping component would be able to recognize that `(Dissolved-in salt1 water)` is more appropriate. Why would the mapper use the base relation `Contained-in` if the relation `Dissolved-in` is known? Even if `(Dissolved-in salt1 water)` is true in the target instance, it may have been absent from the target description given to the mapper and thus prevented from matching. Alternatively, the mapping component could not have mapped `Contained-in` to `Dissolved-in` if the instantiated expression had never been formed before.

5.3.2.3 Create predicate

If a predicate cannot be mapped as is, and no suitable correspondent can be found, then a new predicate is built using the `create-predicate` operation. The ISA hierarchy is climbed until a predicate is found having argument type constraints consistent with their proposed instantiation. A new child is then created below this point (Burstein, 1985). The new predicate inherits all of the original's properties (e.g., commutative, n-ary, relational group), with each of the new predicate's argument types either remaining the same if consistent or changing to a known attribute of the argument occupying that position.

[10]The *climb until consistent* operation is taken from Burstein's (1985) method for creating new predicates.

In the case of `Fluid-Path` being applied to a heat conducting metal bar, a new child of the more general `Path` predicate would be created (e.g., `Path-18`).

5.3.3 Unknown objects

When a candidate inference contains slots occupied by unknown objects (*skolem objects*), suitable target objects must be found or their existence conjectured. There are four options:

1. *General physical knowledge.* Search for a known item that may actually be the item in question. This is a component of the general abduction problem and is dealt with by the abductive retriever.

2. *Analogous conditions.* Search for a known item that satisfies constraints considered functionally analogous to those conjectured for the skolem object.

3. *Directed experimentation.* Experiments may be devised to empirically determine what the missing entity is (e.g., Rajamoney et al., 1985).

4. *Hypothesize existence.* The object's existence may simply be assumed and represented by a skolem constant. If it is a known type of object, then a standard assumption mechanism as used in abduction will suffice. Otherwise, the existence of some new, hypothetical entity may be assumed, as was done when *ether* was conjectured as a medium for light waves.

The first and last options are described in this section. The second option was addressed above, under the heading of alternate expression retrieval. The third option is briefly reviewed later in this chapter and demonstrated in (Falkenhainer & Rajamoney, 1988).

5.3.3.1 Abductive retrieve

The `abductive-retrieve` operation seeks known objects that may consistently fill the role of a given skolem object. It begins by collecting all of the conditions the skolem must satisfy. For transfer of proposed QP theory processes, these conditioning relations are defined to be all propositions containing the skolem object and appearing as an antecedent in one of the proposed process definitions. These conditions are then passed to the abductive retriever as a single conjunctive goal. Since skolem objects are expressed as existentially quantified variables during this operation, returned instantiations represent entities assumably equal to the unknown skolem object.

A branch bound *bb* (default 5) is used to limit the number of acceptable possibilities. If more than *bb* candidates are retrieved, it is assumed that not enough information exists to make a decision and the skolem object remains unknown.

`abductive-retrieve` is not applied to skolem tokens representing *compound objects.* These are objects fully defined by their constituents and thus are known when all of their constituents are known. For example, a specific contained liquid (e.g., `(Contained-Liquid cl1)`) is uniquely identified by a substance (e.g., `(Substance-of cl1 water)`), in a liquid state (e.g., `(Liquid cl1)`), in a particular container (e.g., `(Container-of cl1 beaker1)`). If the `cl1` token were not known for this contained liquid, a new token would simply be created.

5.3.3.2 Create entity

If a skolem object cannot be resolved a new entity token may be created to fill the role. New entities are created using `create-entity`, which makes a new entity token and then analyzes the consistency of its proposed existence.

Consistency for created entities deviates from the standard consistency operation. First, the new token is substituted into the skolem object's N conditions and the status of

$$(\texttt{Consistent}\ C_1 \wedge \ldots \wedge C_N)$$

is determined as described above.[11] However, if the conjunction $C_1 \wedge \ldots \wedge C_N$ is inconsistent using the new entity token, then

$$(\forall x)\neg[C_1(\ldots x \ldots) \wedge \ldots \wedge C_N(\ldots x \ldots)]$$

is true, where all instances of the new entity are replaced by x. In other words, no such object could possibly exist under current beliefs. In this case, we are faced with the problem of conjecturing a new kind of entity as opposed to simply creating a new instance of a plausible entity type. The specific object conditions participating in the contradiction are extracted by examining the data dependency network.[12] One of these propositions is then modified using `create-predicate` to resolve the contradiction. A heuristic preferential ordering is used to determine which proposition is selected for replacement: (1) a unary proposition having the entity as its only argument or (2) a proposition having the entity

[11]Note that the consistency of the C_i must be determined together rather than individually, since each atomic sentence may be consistent when considered individually, but may not be consistent when considered in conjunction with the other $N-1$ assumptions. (Charniak, 1988) makes this point as well.

[12]In ATMS specifics, the minimal set of contradictory assumptions taken from the union of assuming each entity condition C_i.

as its only skolem object. The third "choose one at random" option is currently grounds to fail the analogy, primarily because its repercussions have not been fully considered.

Conjectured entities have a long history in science and are often at the center of controversy in a developing theory. For example, an all-pervading *ether* was long postulated to provide a medium for the flow of light waves because other kinds of waves require a medium. Consider explaining the temperature changes occurring when a hot brick is immersed in cold water, by analogy to liquid flow. If no knowledge of heat exists, an analogue to the flowing liquid is proposed and given the following conditions:

(Substance *?liquid*) ∧ (Liquid *?liquid*) ∧ (Contained-in *?liquid* brick1)

Note that no object could ever satisfy this conjunction, given that brick1 is a non-porous object (i.e., (not (Porous brick1))). create-entity detects the contradiction

(Liquid *?liquid*) ∧ (Contained-in *?liquid* brick1) ∧ (not (Porous brick1)) $\rightarrow \perp$

and conjectures a new entity having the conditions

```
(Substance sk-water-6) ∧ (Sk-phase-1 sk-water-6)
∧ (Contained-in sk-water-6 brick1)
```

Sk-phase-1 is a new child predicate to Phase, which was the immediate parent to Liquid in the predicate hierarchy.

5.3.4 The map and analyze cycle

The transfer task is to create a set of usable target hypotheses from candidate inferences proposed by mapping, or reject the analogy if this cannot be accomplished. A decomposition of this process appears in Figure 5.4. Transfer consists of two phases. The *probe phase* attempts to repair inadequacies of the candidate inferences by looking for alternatives within the set of beliefs derivable or assumable from existing knowledge. If everything is adequate (all expressions are consistent and there are no skolem objects) or can be repaired with current knowledge, then the set of candidate inferences are returned as operational target hypotheses. The *resolve phase* repairs disclosed inadequacies when the probe phase fails, by going beyond existing knowledge. This may involve hypothesizing the existence of an unknown object or creating a new predicate.

Throughout the discussion, I will refer to *the* mapping and *the* inferences. However, the transfer process has the potential to branch (e.g., multiple alternative fillers for a skolem object), producing a set of mapping, inferences pairs. The discussion has been simplified by considering the operation of a single branch.

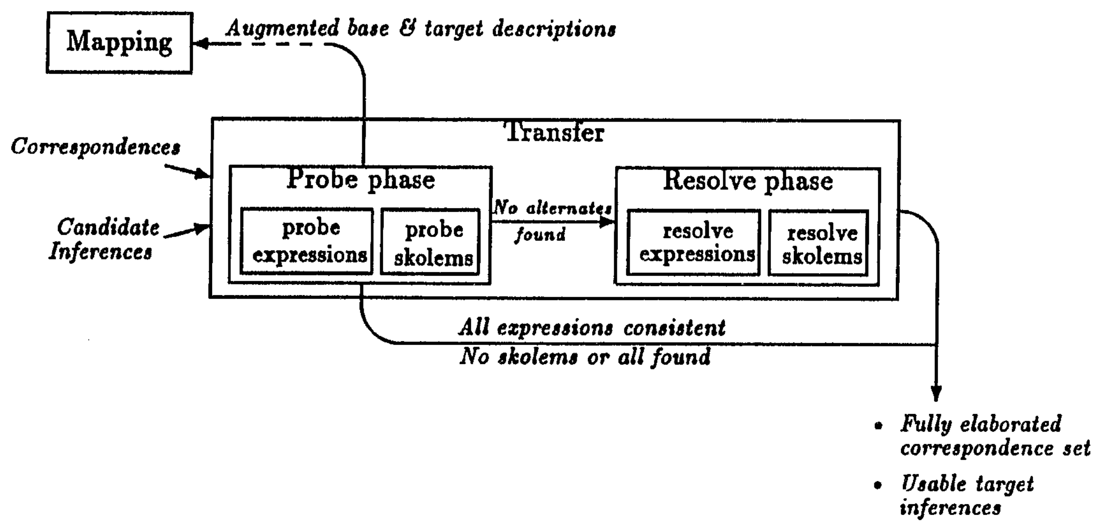


Figure 5.4: The transfer process

Transfer uses the set of operations described in the preceding sections and summarized in Table 5.1. The algorithm is shown in Table 5.2. The process starts when a new mapping is received and the probe phase is entered.

5.3.4.1 Probe phase

The probe phase begins by examining each proposed fact in the set of candidate inferences. Because of the hierarchical representations being used in conjunction with SME, I should

Table 5.1: Summary of transfer operations.

- `show-consistent(`*expression*`)` $\Rightarrow$ {*success* | *failure*}
- `retrieve-alternate-expression(`*(P...)*`)` $\Rightarrow$ *(P'...)*
- `create-predicate(`*(P...)*`)` $\Rightarrow$ *(P'...)* and P' installed
- `abductive-retrieve(`*expression*`)` $\Rightarrow$ {(*expression*θ_1,$\mathcal{A}_1$)...(*expression*θ_N,$\mathcal{A}_N$)} | *failure*
- `create-entity(`*skolem-object*, *conditions*`)` $\Rightarrow$ *entity-token [conditions']*

Table 5.2: Transfer algorithm.

```
1. Probe phase

   Repeat Until no new information can be retrieved or there are no inconsistent expressions
   and no skolem objects:

   (a) Probe expressions

           For every expression $\mathcal{E} \in \mathcal{CI}$
               (a) show-consistent($\mathcal{E}$)
               (b) If not consistent, retrieve-alternate-expression($\mathcal{E}$)
                   (accept provably true alternates only)

   (b) Probe skolems

           For every skolem object $\mathcal{S} \in$ skolem-objects($\mathcal{CI}$)
             Let $\mathcal{C_S} = \{\mathcal{E_S} \mid \mathcal{E_S} \in \mathcal{CI}$ and $\mathcal{S} \in \mathcal{E_S}\}$
               (a) abductive-retrieve($\bigwedge(\mathcal{C_S})$)
               (b) If retrieval fails, for every $\mathcal{E_S} \in \mathcal{C_S}$
                   retrieve-alternate-expression($\mathcal{E_S}$)

   (c) If (there exists an expression $\mathcal{E} \in \mathcal{CI}$ that is not consistent
       or a skolem object that cannot be identified)
       and new information has been found by retrieve-alternate-expression
       then invoke mapping with base and target descriptions augmented with the new in-
       formation.

2. Resolve phase

   If inconsistent expressions or skolem objects still exist

   (a) Resolve expressions

           For every expression $\mathcal{E} \in \mathcal{CI}$ that is not consistent
               (a) retrieve-alternate-expression($\mathcal{E}$)
                   (accept any alternate in preferential order (section 5.3.2.2))
               (b) If retrieval fails, create-predicate($\mathcal{E}$)

   (b) Resolve skolems

           For every skolem object $\mathcal{S} \in$ skolem-objects($\mathcal{CI}$)
             Let $\mathcal{C_S} = \{\mathcal{E_S} \mid \mathcal{E_S} \in \mathcal{CI}$ and $\mathcal{S} \in \mathcal{E_S}\}$
               create-entity($\mathcal{S}$,$\mathcal{C_S}$)

3. Return set of operational target inferences $\{\mathcal{I}_{T1} \ldots \mathcal{I}_{TN}\}$,
   where each $\mathcal{I}_{Ti}$ represents a different permutation of the set of proposed modifications to
   $\mathcal{CI}$.
```

clarify that what is meant by "proposed fact" depends upon the representation specifics, but may be identified as the smallest boolean expressions present in the representation. In terms of the predicate calculus, these correspond to all *atomic sentences* in the candidate inferences. Thus, in each QP process description proposed

```
(Process-definition name instance-name
        (Implies (And (Individual name (Conditions conditioning-facts))
                      ...
                      preconditions and quantity conditions
                      ...)
                 (And effects)))
```

the expressions to be analyzed are comprised of the conditions placed on each individual, the process' preconditions and quantity conditions, and the process' consequent effects.

During the probe phase, each atomic expression is checked for consistency using `show-consistent`. If an expression is inconsistent, an alternative expression is sought, using `retrieve-alternate-expression`. It first seeks functionally analogous expressions that are currently believed. If none are found, alternate expressions are sought via minimal ascension, and only those currently believed are accepted (recall that minimal ascension may return alternates that are merely consistent with what is known). Any alternative, believed expressions found are collected and later used to augment the current base and target descriptions.

The probe phase next examines each skolem object present in the candidate inferences. `abductive-retrieve` is used to seek known objects that may be consistently substituted for the unknown object. If retrieval fails, then `retrieve-alternate-expression` is applied to each of the skolem's proposed conditions. This addresses an important component of the correspondence problem: how can a target correspondent for a base object be found given only the base object's stated conditions?[13] Unless these conditions have been fully mapped, there may be no target correspondent to satisfy them. This is especially true of cross-domain analogies, in which an analogous pair of objects may have no identical characteristics in common.

If all expressions are consistent and there are no unknown skolem objects remaining, transfer is successfully completed. The potentially modified candidate inferences are now considered operational for the target domain (recall there may be branching at this point,

[13]Previous analogy systems have failed to address this problem. They either create a new token and assume the proposed conditions for it (Winston, 1982), work on constrained within-domain analogies that eliminate the problem (Carbonell, 1983a; Kedar-Cabelli, 1988), or don't examine the possibility of skolem objects produced by mapping (Burstein, 1983; Greiner, 1988).

producing a set of alternate target hypotheses). When this is not accomplished, there are two options. If new information was retrieved, transfer returns to the mapping stage, using base and target descriptions augmented with the additional information.[14] This may lead to an augmented or alternate mapping and possibly new points of discrepancy on which to focus the transfer process. Otherwise, transfer proceeds to the resolve phase.

5.3.4.2 Resolve phase

Intuitively, the resolve phase corresponds to abandoning attempts at using existing knowledge to resolve candidate inference inadequacies. It begins by reconsidering any remaining inconsistent expressions. `retrieve-alternate-expression` is used to seek a consistent alternate predicate. Only the minimal ascension component applies at this point, since functionally analogous alternatives would have been found during probing. Unlike the probe phase, the resolve phase accepts any alternative predicate suggested, using the three preferential orderings described in section 5.3.2.2. If there are no consistent alternate predicates found, `create-predicate` is used to create a new expression.

The resolve phase completes with all remaining skolem objects replaced by new object tokens, using `create-entity`. Transfer ends with the resulting set of usable target inferences.

5.3.5 Making necessary assumptions

A "candidate inference" may be any number of things. Importantly, it need not be a simple answer. It will often represent an entire line of reasoning, either in the form of a compiled schema or model, an entire proof trace, or a complete planning or design sequence. For these to be useful to the performance element, the assumptions upon which they rest (e.g., preconditions or premises) must explicitly be assumed as true and the analogical relationship stated as the basis for the assumption. This is a phase of operation not seen in other analogy systems.[15] Yet, it is a vital component of any complex reasoning task.

In PHINEAS, each candidate theory $\mathcal{T}$ about the current observation is associated with two nodes, `(Believe` $\mathcal{T}$`)` and `(Holds` $\mathcal{T}$`)` (see Figure 5.5). This results in dependencies emanating from $\mathcal{T}$ in opposite directions. `(Believe` $\mathcal{T}$`)` is supported by those assumptions upon which $\mathcal{T}$ depends, which are all the assumptions made in the course of deriving $\mathcal{T}$ (e.g.,

[14]In the current implementation, only one repetition of mapping is supported. This is due to time constraints on program development, rather than a serious technical problem. It has been sufficient for all examples examined to date.

[15](Burstein, 1985) may be considered an exception, as a record of past mappings is maintained.

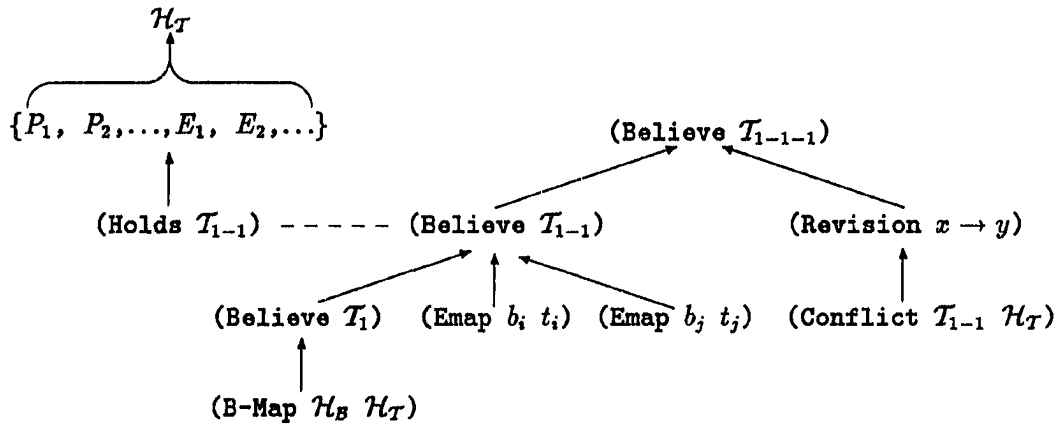

$\mathcal{H}_T$
$\{P_1, P_2, \ldots, E_1, E_2, \ldots\}$
(Believe $\mathcal{T}_{1-1-1}$)
(Holds $\mathcal{T}_{1-1}$)
(Believe $\mathcal{T}_{1-1}$)
(Revision $x \rightarrow y$)
(Believe $\mathcal{T}_1$)
(Emap b_i t_i)
(Emap b_j t_j)
(Conflict $\mathcal{T}_{1-1}$ $\mathcal{H}_T$)
(B-Map $\mathcal{H}_B$ $\mathcal{H}_T$)

Figure 5.5: Prototypical justification lattice used in PHINEAS to support analogical reasoning. $\mathcal{T}_i$ represents a proposed theory about an observed behavior history $\mathcal{H}_j$. A theory may be composed of process definitions P_i and entity definitions E_i.

the behavioral analogy, additional modifications made during transfer, or prior theories of which $\mathcal{T}$ is a revision). `(Holds` $\mathcal{T}$`)`, on the other hand, is an assumption upon which all components and consequences of $\mathcal{T}$ depend.

5.3.6 Example: Disappearing Alcohol

Figure 5.6 repeats the candidate inference proposed by mapping dissolving into the disappearing alcohol situation. As previously noted, there are a number of problems with the inference as it stands. Entering the probe phase of transfer, the first task is to examine the consistency of each proposed expression. The following are found to be inconsistent:

`(Solid alcohol1)` Alcohol is known to be a liquid, which contradicts the taxonomy that something can only be one of {solid, liquid, gas} at a time.

`(Soluble alcohol1)` The argument to `Soluble` must be a solid.

`(Immersed-in alcohol1 (:skolem water1))` The first argument to `Immersed-in` must be a solid.

`(Soluble-in alcohol1 (:skolem water1))` The first argument to `Soluble-in` must be a solid.

Applying `retrieve-alternate-expression` on each of these produces the following augments to the existing target description:

```
(B-EXPLAINS
   (SET (PACKET-DEFINITION (SOLUTION (:SKOLEM WATER1))
           (SET (QUANTITY (CONCENTRATION (:SKOLEM WATER1)))
                 (QUANTITY (SATURATION-POINT (:SKOLEM WATER1)))
                 (NOT (LESS-THAN (A (CONCENTRATION (:SKOLEM WATER1))) ZERO))
                 (NOT (LESS-THAN (A (SATURATION-POINT (:SKOLEM WATER1))) ZERO))))
        (PROCESS-DEFINITION (:SKOLEM DISSOLVE) (:SKOLEM PI1)
           (IMPLIES
                 (AND (INDIVIDUAL ALCOHOL1
                           (CONDITIONS (SOLID ALCOHOL1) (SOLUBLE ALCOHOL1)))
                      (INDIVIDUAL (:SKOLEM WATER1)
                           (CONDITIONS (SOLUTION (:SKOLEM WATER1))
                                       (IMMERSED-IN ALCOHOL1 (:SKOLEM WATER1))))
                      (SOLUBLE-IN ALCOHOL1 (:SKOLEM WATER1))
                      (GREATER-THAN (A (AMOUNT-OF ALCOHOL1)) ZERO))
                 (AND (QUANTITY (DISSOLVE-RATE (:SKOLEM PI1)))
                      (QPROP (DISSOLVE-RATE (:SKOLEM PI1)) (SURFACE-AREA ALCOHOL1))
                      (GREATER-THAN (A (DISSOLVE-RATE (:SKOLEM PI1))) ZERO)
                      (CTRANS AMOUNT-OF-3268 (CONCENTRATION (:SKOLEM WATER1))
                           (A (DISSOLVE-RATE (:SKOLEM PI1))))))))
  OPEN-ALCOHOL-BEHAVIOR)
```

Figure 5.6: Candidate inference proposed by mapping dissolving into the disappearing alcohol situation.

```
(Liquid alcohol1)
(Supports (Touching alcohol1 atmosphere)
          (Physical-Path alcohol1 atmosphere (common-face alcohol1 atmosphere)))
(Supports (Open beaker2)
          (Physical-Path alcohol1 atmosphere (common-face alcohol1 atmosphere)))
```

The first expression is suggested as an alternative for `(Solid alcohol1)`. The other two are alternatives for `(Immersed-in alcohol1 (:skolem water1))`. All three of these facts were absent from the original target description given SME, but are derivable from it. For example, `(Touching alcohol1 atmosphere)` follows from a contained liquid in a container that is open. `(Open beaker2)` is the only expression appearing in the original target description, but its relevant function was not present.

The latter two target augments lead to the following base augment

```
(Supports (Immersed-In salt1 water1)
          (Physical-Path salt1 water1 (common-face salt1 water1)))
```

The second task of the probe phase is to seek known objects that may be substituted for the skolem objects present in the candidate inference. In this case there is only one, (:skolem water1), which indicates there was no target correspondent for water1, the liquid into which salt1 was dissolving. abductive-retrieve is invoked on the conjunction[16]

```
(Soluble-In alcohol1 ?water1) ∧ (Solution ?water1)
 ∧ (Immersed-In alcohol1 ?water1)
```

which fails to find any instantiations for which the conjunction may be assumed.

At this point, the probe phase is completed. It failed to operationalize the candidate inference, but successfully retrieved additional relevant information about the target. The target and base are augmented with the expressions shown above and mapping is repeated, which produces

```
(B-Explains
  (Set (Packet-Definition (Solution atmosphere)
        (Set (Quantity (Concentration atmosphere))
             (Quantity (Saturation-Point atmosphere))
             (Not (Less-than (A (Concentration atmosphere)) zero))
             (Not (Less-than (A (Saturation-Point atmosphere)) zero))))
       (Process-Definition (:skolem dissolve) (:skolem pi1)
        (Implies
         (And (Individual alcohol1
                  (Conditions (Liquid alcohol1) (Soluble alcohol1)))
              (Individual atmosphere
                  (Conditions (Solution atmosphere)
                              (Touching alcohol1 atmosphere)))
              (Soluble-in alcohol1 atmosphere)
              (Greater-than (A (amount-of alcohol1)) zero))
         (And (Quantity (dissolve-rate (:skolem pi1)))
              (Qprop (dissolve-rate (:skolem pi1)) (surface-area alcohol1))
              (Greater-Than (A (dissolve-rate (:skolem pi1))) zero)
              (I- (amount-of alcohol1) (A (dissolve-rate (:skolem pi1))))
              (I+ (concentration atmosphere) (A (dissolve-rate (:skolem pi1))))))))
 open-alcohol-behavior)
```

This second mapping is much more complete. Importantly, the object atmosphere has been introduced as a consideration in the target vocabulary, leading to the complete absence of skolem objects.[17]

[16]A simple efficiency improvement on the current implementation should become obvious at this point. PHINEAS doesn't know that it already determined that (Immersed-in alcohol1 ?water1) is inconsistent in isolation.

[17]Recall that the skolems (:skolem dissolve) and (:skolem pi1) are treated specially by PHINEAS.

Repeating the probe phase of transfer, the following expressions are now found to be inconsistent:

`(Solution atmosphere)` The argument to `Solution` must be a liquid, while `atmosphere` is a gas.

`(Soluble alcohol1)` The argument to `Soluble` must be a solid. This was discovered during the previous probe session as well.

`(Soluble-in alcohol1 atmosphere)` The first argument to `Soluble-in` must be a solid. This was discovered during the previous probe session as well.

When `retrieve-alternate-expression` is applied to each of these expressions, no alternatives are found. In the case of `(Solution atmosphere)`, this indicates `PHINEAS'` lack of domain theory about general mixtures.

At this point, the probe phase is completed. However, unlike the first probe phase, it has failed to operationalize the candidate inference and failed to retrieve additional information about the target. As a result, the resolve phase begins. There are no skolem objects to resolve, so the only task is to create new expressions for the three contradictory ones shown above. The following translations are made:

```
(Solution-8 atmosphere)
(SK-Soluble-4-1 alcohol1)
(SK-Soluble-in-4-1 alcohol1 atmosphere)
```

These postulate that the `atmosphere` is analogous to a liquid solution and that `alcohol1` is "soluble in" the atmosphere similar to the way salt is soluble in water. The final transferred hypothesis, translated to pure QP theory syntax is shown in Figure 5.7. While usable, there is no guarantee it will fully and consistently explain the observed behavior. Verifying that is the topic of the next chapter.

5.3.7 Interactive, empirical transfer

In complicated analogies, the transfer process will motivate a number of questions that are very difficult to answer from a passive, logical analysis. Is condition C_i in the theory necessary in this case to bring about the observation? Is this object not mentioned in the original observation description present? (Falkenhainer & Rajamoney, 1988) presents a protocol for just this type of empirical question answering. In that work, `PHINEAS` was merged with Rajamoney's (1988a) experimentation system (`ADEPT`). This provided `PHINEAS` with a means to obtain empirical answers to the various questions that arise during the

```
(DEFPROCESS  (PROCESS-4 ?V-10 ?V-11)
   INDIVIDUALS  ((?V-10 :CONDITIONS (LIQUID ?V-10) (SK-SOLUBLE-4-1 ?V-10))
                 (?V-11 :CONDITIONS (SOLUTION-8 ?V-11) (TOUCHING ?V-10 ?V-11)))
   PRECONDITIONS  ((SK-SOLUBLE-IN-4-1 ?V-10 ?V-11))
   QUANTITYCONDITIONS  ((GREATER-THAN (A (AMOUNT-OF ?V-10)) ZERO))
   RELATIONS  ((QUANTITY (DISSOLVE-RATE ?SELF))
               (QPROP (DISSOLVE-RATE ?SELF) (SURFACE-AREA ?V-10))
               (GREATER-THAN (A (DISSOLVE-RATE ?SELF)) ZERO))
   INFLUENCES ((CTRANS (AMOUNT-OF ?V-10) (CONCENTRATION ?V-11)
                                         (A (DISSOLVE-RATE ?SELF)))))

(DEFENTITY  (SOLUTION-8 ?V-12)
   (QUANTITY (CONCENTRATION ?V-12))
   (QUANTITY (SATURATION-POINT ?V-12))
   (NOT (LESS-THAN (A (CONCENTRATION ?V-12)) ZERO))
   (NOT (LESS-THAN (A (SATURATION-POINT ?V-12)) ZERO)))

(ASSUME (SOLUTION-8 ATMOSPHERE))
(ASSUME (SK-SOLUBLE-4-1 ALCOHOL1))
(ASSUME (SK-SOLUBLE-IN-4-1 ALCOHOL1 ATMOSPHERE))
```

Figure 5.7: Transferred explanation for the disappearing alcohol observation by analogy to dissolving.

normal course of analogical theory development. It provided ADEPT with a means to obtaining a restricted number of ordered hypotheses and focused, theoretically motivated queries about the world.

For example, consider how the disappearing alcohol observation may be explained by analogy to boiling. When PHINEAS is operating in isolation, it must assume the presence of an ideal heat source (e.g., stove) and alcohol gas. When coupled with ADEPT, PHINEAS is able to ask

```
(Present?  ?steam1  ((Contained-Gas ?steam1)
                     (Container-of ?steam1 beaker2)
                     (Substance-of ?steam1 alcohol1)))
```

which ADEPT answers positively by asking that litmus paper be placed in contact with the air in beaker2 and noting its change in color. Additionally, PHINEAS is now able to ask if the heat flow process is necessary for the observation to occur. When ADEPT repeats the observation using a thermally isolated container (through instructions to a human assistant), the alcohol continues to disappear. This leads to a new evaporation process by analogy to boiling which does not require heat flow, something PHINEAS cannot propose when operating in isolation.

Chapter 6

Verification-Based Analogical Learning

The preceding chapters described methods for generating explanatory hypotheses by analogy. However, a good analogizer doesn't simply stop with the statement of an analogy. How these hypotheses are evaluated and ultimately used is just as important as how they are generated. Are the proposed inferences correct, likely to be correct, or consistent? Do they provide a solution for the task at hand? The analogy must be evaluated for consistency and coverage. Interaction with other beliefs must be checked. Indeed, the hypothesis must be tested to be sure it predicts the very observation it was intended to explain.

Nevertheless, the problem of evaluating an analogy's validity and using it in a complex reasoning task has received little attention. Most models avoid the issue by either stopping once inferences are produced (Holyoak & Thagard, 1988b; Kass et al., 1986; Indurkhya, 1987) or requiring that analogy produce d-sound inferences in which the results could have been achieved (more slowly) without the analogy (Kling, 1971; Carbonell, 1983a; Kedar-Cabelli, 1985b). The validity problem is central to robust analogical processing. In particular, if i-sound inferences are to be allowed, they must undergo a series of evaluative processes. This is reflected in the following requirement:

- *Verification requirement:* When possible, the results of analogy must be tested empirically and against other knowledge.

In analogical reasoning, the verification requirement is automatically satisfied as a natural side-effect of achieving the goals of the problem solver (i.e., backtracking may occur at points of inaccuracies). The verification requirement establishes a similar set of goals for an analogical learning system - the goal of improving the believed accuracy of its knowledge.

Verification-based analogical learning (VBAL) is designed specifically to address this requirement. It depicts analogical learning as an iterative process of hypothesis formation, verification, and revision, centered around the requirement to confirm accuracy and increase the likelyhood of being correct. VBAL relies on analogical inference to propose explanations and gedanken experiments in the form of qualitative simulation to analyze their validity. Specifically, the predictions of a new model are compared against observed behavior, enabling the system to test the validity of the analogy and sanction refinements where the analogy is incorrect.

VBAL may be illustrated with the following scenario. When two bodies, one hot and one cold, are placed in contact with each other, they will eventually reach the same temperature. If the notion of water flow suggests itself, we may construct a model for the situation in which a heat fluid is seen "flowing" from a higher temperature to a lower temperature. Using this new model shows that it accurately explains the phenomenon. This is called *verifying the initial adequacy* of the model. The new theory now predicts that certain other events must be possible for the given physical configuration, such as the bidirectionality of heat flow. We attempt to recollect a prior experience demonstrating this predicted behavior or we conduct simple experiments to explore the space of hypothesized behaviors for the given objects. This is called *verifying the local predictions* of the model. If we were to explore the consequences of the analogy beyond the current situation, a number of additional predictions may be made based upon the intrinsic properties of liquids and physical objects. For example, *conservation of matter* would lead to predictions based on *conservation of heat.* Exploring the consequences of these additional predictions is called *verifying the extended predictions* of the analogy. This cycle of hypothesis formation, confirmation, refutation, and subsequent refinement is the essence of verification-based analogical learning. Its focal point is a view of analogical inference as a means to propose an initial model of a domain that may need adjustment.

The consistency of a proposed model may be verified in two ways. First, the model may be formally analyzed and proven consistent both internally and with respect to existing knowledge. Second, the model can be used in some performance task and any resulting inconsistencies sought. The first is preferable for obvious contradictions arising from simple lookups of atomic sentences, as is done during transfer. However, for all but the most trivial theories, determining absolute consistency is undecidable (Boolos & Jeffrey, 1974). Hence, an approximate form of consistency verification is necessary.[1] VBAL is a proposal for *performance* as a basis for analyzing the global consistency of a hypothesized model.

[1]Indurkhya (1987) and Greiner (1988) use finite complexity bounds to form an approximate definition of consistency.

Unless obvious contradictions are detected, the system should test the model against a sample of situations, including the current one.

In terms of physical models, such simulation-oriented testing corresponds to the idea of a *gedanken*[2] or "thought" experiment. A complete gedanken analysis consists of taking a general theory, or set of premises, and imagining artificial scenarios to see what the theory has to say about each scenario. When it conflicts with prior conceptions or when we reach mutually incompatible conclusions, we have a setting for learning. Simulation is needed to tease apart implicit inconsistencies of belief that are difficult, if not impossible, to detect from an analysis of a model in isolation. Only until we try to use a model, with all of its potential interaction with prior beliefs, do we expose inconsistencies.

This chapter describes the use of one particular type of gedanken experiment – qualitative simulation. It begins by reviewing the processes of qualitative reasoning and measurement interpretation, which generate the predictions of given model and compare those predictions to observation. The acceptance criteria used in model evaluation are then described, followed by examples demonstrating the different roles of qualitative simulation as an evaluative method. Finally, methods for examining the local and extended predictions of an analogy through both simulation and empirical experimentation are proposed.

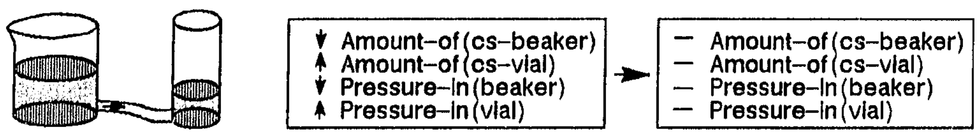
6.1 Qualitative Reasoning

One of the goals of qualitative reasoning research is to formalize the tacit rules people use to mentally simulate the behavior of a system through time (Bobrow, 1985). When a qualitative model has been constructed, an analysis of the model by the reasoner produces an *envisionment* – a description of the possible behaviors for the current physical configuration. The behavior of the system through time may then be represented as a single path through the envisionment. The system is able to provide an explanation for the observations and verify the model's consistency if a path through the envisionment formed from the model can be found that corresponds to the measurements.

In terms of envisioning, a given model may produce two classes of predictions:

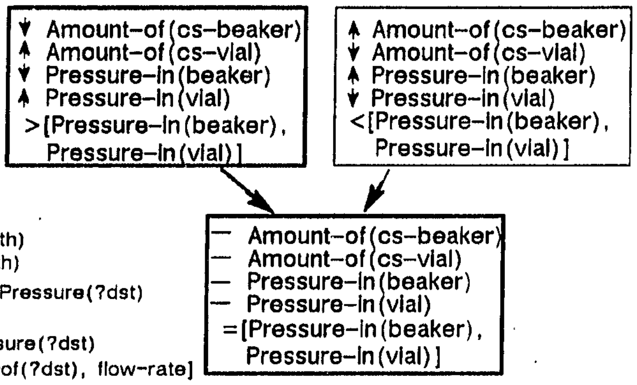
- *Scenario Predictions:* Those behaviors predicted from the model applied to the current physical configuration (i.e., the observed set of objects and the structural relationships between them). A total envisionment for the current situation will generate all possible behaviors. An attainable envisionment will generate the subset reachable

[2]German word for *thought*. As far as I can determine, the phrase "gedanken experiment" was popularized due to Einstein's masterful use of the technique, although the technique itself dates back to at least Galileo.



(a)

```
PROCESS Liquid-Flow
Individuals
    ?subst,  liquid
    ?src,  Can-Contain(?src, ?subst)
    ?src-cs,  Contained-Liquid
    ?dst,  Can-Contain(?dst, ?subst)
    ?dst-cs,  Contained-Liquid
    ?path,  Fluid-Path(?src, ?dst, ?path)
Preconditions         Fluid-Aligned(?path)
QuantityConditions    Pressure(?src) > Pressure(?dst)
Relations
    flow-rate = Pressure(?src) - Pressure(?dst)
    Ctrans[Amount-of(?src), Amount-of(?dst), flow-rate]
```



(b)

Figure 6.1: Qualitative simulation.

from some specified starting state, which in this case would be the initial state of the observation.

- *Extended Predictions:* Those behaviors describable from the same model applied to new configurations of objects.

In these terms, adequacy verification succeeds when a path through the model's scenario predictions can be found which corresponds to the observed behavior. Local prediction verification consists of confirming the possibility of the alternate scenario predictions, either exhaustively or heuristically (Section 6.4). Verifying the extended predictions of the analogy involves exploration of the derived model's impact on the system's total beliefs through a select set of extended predictions. Of course, extended verification is potentially unbounded and implementing it would require a specification of what predictions to explore.

PHINEAS uses Forbus' (1986b) *qualitative process engine* (QPE) to produce an envisionment for an analogically proposed model applied to the given physical configuration. For example, Figure 6.1(a) shows a beaker connected to a vial and the observed liquid flow behavior for this configuration. Figure 6.1(b) shows a liquid flow process description and the scenario predictions it produces for the beaker-vial configuration. The darkened two-state path indicates where the liquid flow model consistently explains the observed behavior. Initially, the beaker pressure is greater than the vial pressure, and liquid is flowing from the beaker to the vial. Eventually, a new state is reached in which their pressures are equal and flow has stopped. However, the model makes additional scenario predictions. If the vial had started out with a higher pressure than the beaker, liquid should flow the other way. Consider how a unidirectional valve in the pipe would effect the validity of this prediction. The scenario predictions of Figure 6.1(b) only envision behaviors for two containers attached by a liquid path. Extended predictions might include the expected behavior of three containers connected in series. The liquid flow model might be extended even further to consider its consequences for situations like siphon or faucet flow.

6.1.1 Measurement Interpretation

Central to analyzing the consistency of a model through qualitative simulation is the process of comparing its predictions to observation and identifying points of discrepancy. This is the *measurement interpretation* problem. Measurement interpretation is the process of finding the best, temporal preserving mapping between an observed behavior and a corresponding set of envisioned states (Forbus, 1986a; DeCoste, 1989). This correspondence represents an interpretation of the behavior by associating it to specific predictions of a model. It may be distinguished from the general interpretation or explanation problem in two ways. First, measurement interpretation will assume all potentially relevant theories were used in producing the scenario envisionment. Second, measurement interpretation addresses the problems of data analysis and real time processing. Mapping a continuous process to an internal discretization can be a dynamic process, dependent upon where gaps, noise, faulty sensors, etc. are believed to exist at any given point in the measurement sequence. For PHINEAS, perfect data is assumed (i.e., no noise or faulty sensors). The task itself, and resolution of the corresponding control issues, is performed by Decoste's *dynamic across-time measurement interpretation* system, DATMI (DeCoste, 1989). This includes selecting a best match between model and observation, and deciding when to doubt the model and allow a sequence to go uninterpreted.

Before further discussing the measurement interpretation task, we require the appropriate vocabulary. This draws primarily from Forbus' (1987) *logic of occurrence.*

A single, time-varying description of the observed behavior will be called a *history* $\mathcal{H}$ (Hayes, 1979). Since there is no unique behavioral description of an observation (Chapter 4), $\mathcal{H}$ represents the behavioral description in current use. A history is composed of *behavioral segments* (bsegs) that are temporally extended and spatially bounded. Bsegs divide a measurement sequence into maximal, contiguous intervals over which all measured quantities are qualitatively constant (i.e., maintain the same qualitative value). Bsegs were defined in Chapter 4. The function *Bsegs* maps from a history to the set of bsegs that comprise it.[3]

An envisionment $\mathcal{E}$ represents all possible qualitative states a particular system may take on and all legal transitions between them. The function *States* maps from an envisionment to the set of states that comprise it. It will be assumed that $\mathcal{E}$ and $\mathcal{H}$ agree in perspective and granularity.

Definition 6.1 (ConsistentWith) *ConsistentWith(b,s) is true whenever bseg b describes behavior of the system that is not inconsistent with state s.*

The set of possible interpretations of an observed bseg correspond to all envisionment states that consistently explain it. This is all states s such that *ConsistentWith(b,s)* is true. A single interpretation must then be selected based on global consistency derived by considering temporally adjacent bsegs and states.

Definition 6.2 (OccursAt) *Given* $s \in States(\mathcal{E})$, $b \in Bsegs(\mathcal{H})$, *OccursAt(s,b) is true exactly when the state s represents what is happening during b.*

Since "what is happening" can never be absolutely determined and there may be multiple interpretations possible, *OccursAt* will be taken to mean s is the *assumed* interpretation for bseg b. Since bsegs may partially specify the properties that distinguish individual states, a path of qualitative states may occur within a given bseg. The converse of *OccursAt* may be defined as well:

Definition 6.3 (NoInterpret) *Given* $b \in Bsegs(\mathcal{H})$, *NoInterpret(b) is true iff for all* $s \in States(\mathcal{E})$, *OccursAt(s,b) is false.*

A *registration* is a mapping which relates an observation to an envisionment by pairing bsegs in the observation with corresponding states in the envisionment. In other words, it is the chosen interpretation for an observation and consists of all true *OccursAt* statements for bsegs in the observation.

DATMI takes a history $\mathcal{H}$ and an envisionment $\mathcal{E}$. It returns a registration which is a temporally ordered (over $\mathcal{H}$) sequence of $OccursAt(s_i,b_i)$ and $NoInterpret(b_j)$ statements.

[3]This deviates from the definition used in (Forbus, 1987), which refers to *the* behavior and uses the term "episode" to refer to behavioral segments. Here I assume a single fixed description of the behavior is in use at a particular time and will use "bseg" to remain consistent with Chapter 4.

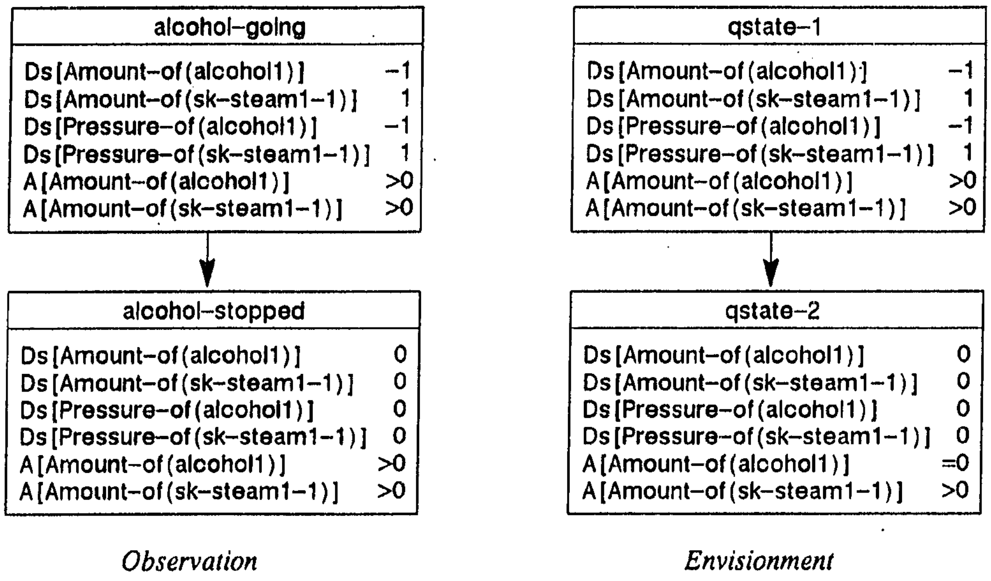

alcohol-going
Ds[Amount-of(alcohol1)] -1
Ds[Amount-of(sk-steam1-1)] 1
Ds[Pressure-of(alcohol1)] -1
Ds[Pressure-of(sk-steam1-1)] 1
A[Amount-of(alcohol1)] >0
A[Amount-of(sk-steam1-1)] >0
alcohol-stopped
Ds[Amount-of(alcohol1)] 0
Ds[Amount-of(sk-steam1-1)] 0
Ds[Pressure-of(alcohol1)] 0
Ds[Pressure-of(sk-steam1-1)] 0
A[Amount-of(alcohol1)] >0
A[Amount-of(sk-steam1-1)] >0
qstate-1
Ds[Amount-of(alcohol1)] -1
Ds[Amount-of(sk-steam1-1)] 1
Ds[Pressure-of(alcohol1)] -1
Ds[Pressure-of(sk-steam1-1)] 1
A[Amount-of(alcohol1)] >0
A[Amount-of(sk-steam1-1)] >0
qstate-2
Ds[Amount-of(alcohol1)] 0
Ds[Amount-of(sk-steam1-1)] 0
Ds[Pressure-of(alcohol1)] 0
Ds[Pressure-of(sk-steam1-1)] 0
A[Amount-of(alcohol1)] =0
A[Amount-of(sk-steam1-1)] >0
Observation
Envisionment

Figure 6.2: Forming a registration between observed bsegs and model derived states.

For example, Figure 6.2 shows the behavior of a closed container filled with liquid and the envisionment from a flawed evaporation model that indicates all contained liquids evaporate. The registration returned by `DATMI` for this observation – model pair is:[4]

```
((OccursAt qstate-1 evaping) (NoInterpret evap-stopped))
```

This indicates that the model consistently explains the `evaping` bseg, which corresponds to `qstate-1` in the envisionment. However, the model is inconsistent with the behavior in `evap-stopped`. The behavior shows no loss of liquid in a non-empty container, while the model believes evaporation should continue until the container is empty.

6.2 Determining the adequacy of a model

The first step in evaluating a proposed model is to ensure that the model is at least consistent with the observation it is intended to explain. This is called *verifying the initial adequacy* of the model. Recall that mapping and transfer combine to produce an operational model. The consequences of this model must then be explored. Does it make sense? Does

[4]This is modified for readability. `DATMI` actually returns the following list of internal structures: `(<Pinterp-1 q-1,seg-1> (:NO-INTERPRET 1.0 2.0))`.

it conflict with current beliefs? Does it provide a complete and consistent explanation for the original observation?

Once an operational model is produced by the transfer component, it is used to generate scenario predictions in order to:

1. *Verify initial adequacy.* In the ideal case, the proposed hypothesis provides a complete and consistent explanation of the observed behavior.

2. *Detect inconsistencies.* The model may be inconsistent, either internally or with respect to prior beliefs.

3. *Analyze coverage.* The base and target behaviors may have only shared a few important properties, leading to extra predictions or unexplained behavior:

   (a) *Identify explanatory inadequacies.* The model may only explain a subset of the phenomenon. What is and isn't explained must be identified before attempting to augment the existing hypothesis through additional explanation processing.

   (b) *Reveal novel predictions.* The model may predict additional behavior that was not reported in the original observation description.

The registration is the basis for verification and revision. Specifically, it states the conditions for acceptance under initial adequacy:

Definition 6.4 (Adequacy) *A proposed model is considered* adequate *iff the registration between the predictions $\mathcal{E}$ of the model and the observed behavior $\mathcal{H}$ is* consistent *and* complete. *A registration is* consistent *if it assigns every bseg in $\mathcal{H}$ to at least one state in $\mathcal{E}$. It is* complete *if there are no OccursAt(s,b) in which bseg b describes a property not described in state s.*

Revision occurs where the registration fails, either where *NoInterpret(b_j)* is true or where additional behavior was observed for which the model makes no prediction.

The method used in PHINEAS for determining initial adequacy is quite simple. This is due primarily to the presence of QPE and DATMI, which are treated as primitive operations by PHINEAS. Adequacy determination begins when an operational model is presented for testing. The model and a description of the scenario under investigation are given to QPE, which produces a total envisionment for the scenario. The envisionment and the observed behavior description are then given to DATMI, which returns a registration between the envisionment and the observation. If the registration is complete, the model is accepted as initially adequate by PHINEAS. If there is more than one initially adequate hypothesis, a preferential ordering is applied. Preferential orderings are handled by PHINEAS' task agenda, which is described in Chapter 8. Finally, if the registration is not complete, revision will be attempted on the model if no hypotheses are found to be adequate.

6.3 Examples

A simulation experiment may function in one of three general ways: As confirmation that the model is complete and consistent, as a discrimination between alternate hypotheses, or as an analysis of a model's coverage. This section presents three examples demonstrating each of these roles.

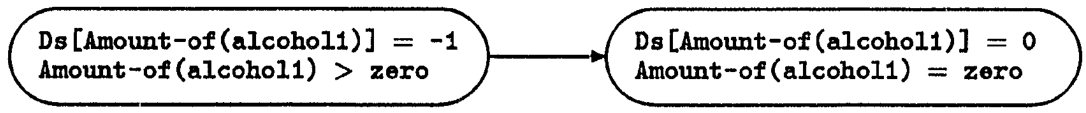
6.3.1 As confirmation

In the simplest case, a hypothesized model will fully predict the observed behavior. In such cases, PHINEAS is able to verify the model's adequacy by finding a path through the envisionment generated with the model that corresponds to the observation.

Consider the model explaining the disappearing alcohol observation that was developed in the previous two chapters. Initially, PHINEAS was told that the amount of alcohol sitting in an open beaker was decreasing, leaving the beaker eventually dry. From this observation, it conjectured an alcohol "dissolving into the atmosphere" model through an analogy with dissolving (shown in Figure 5.7). Thus, it may now attempt to interpret the original situation. The proposed process model (Process-4) and associated assumptions are given to QPE, which produces the two-state envisionment show in Figure 6.3. PHINEAS finds that the darkened path is consistent with the original observation and hence the new theory accurately models the heat flow situation (at least in its overall qualitative behavior). It has thus verified that the theory is consistent with this and functionally similar instances of heat flow, enabling the new model to be added to the set of known domain theories. Experiments may be used to further verify the theory and the new model, as discussed in Section 6.4.

6.3.2 As discrimination

Another role of simulation experiments is to discriminate between two or more mutually incompatible, competing hypotheses. Only a subset of the hypotheses may be consistent with the full constraints of the situation. When confronted with a set of candidate hypotheses, the ideal is to be able to find only a single explanation that makes sense when fully analyzed. In that case, the refuted hypotheses may be rejected outright (i.e., no revision


Ds[Amount-of(alcohol1)] = -1
Amount-of(alcohol1) > zero
Ds[Amount-of(alcohol1)] = 0
Amount-of(alcohol1) = zero

```
Analysis of OPEN-ALCOHOL according to theory OPEN-ALCOHOL-BEHAVIOR-THEORY-8

  In behavioral segment 1
      (AMOUNT-OF ALCOHOL1) is Decreasing
      (CHANGE-RATE (AMOUNT-OF ALCOHOL1)) is Constant
      (A (AMOUNT-OF ALCOHOL1)) is Greater Than ZERO

      Due to the following processes being active:
          PROCESS-4(ALCOHOL1 ATMOSPHERE)
              which is analogous to DISSOLVE.

  In behavioral segment 2
      (AMOUNT-OF ALCOHOL1) is Constant
      (A (AMOUNT-OF ALCOHOL1)) is Equal To ZERO

      There are no processes active.
```

Figure 6.3: Demonstrating the adequacy of the proposed disappearing alcohol model constructed by analogy to dissolving. The two-state envisionment at the top was produced by QPE. The analysis at the bottom was produced by DATMI as a summary of its successful interpretation.

attempted) and the winning hypothesis retained as the one "true" explanation (assuming only one, rather than all possible, is sought).

For example, consider the behavior of a beach ball suspended in an upward column of flowing air, as in Figure 6.4. Stores will sometimes reverse the air flow of a vacuum cleaner and suspend a beach ball in the exhaust jet as part of their display. The ball is very stable and will remain within the jet even when slapped around. What holds the ball in place and why is it so stable (Walker, 1975, problem 4.20)? PHINEAS is able to conjecture two plausible explanations for what is happening. One is derived from the notion of pushing and proposes that the air jet is pushing the ball upward from underneath and towards the center on both sides (Figure 6.4(a)). The core to this model are the relations:

$$\texttt{Force(ball,position)} > \texttt{zero}$$
$$\texttt{Force(ball,position)} \propto_{+} \texttt{Amount-of(air,position)}$$

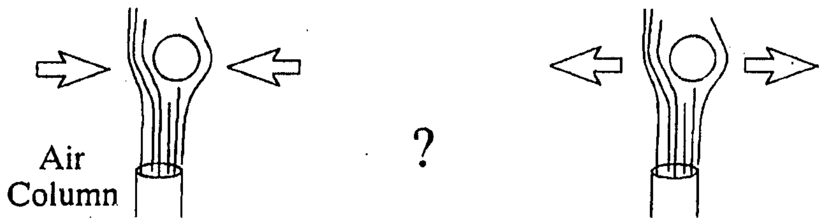


Figure 6.4: Discriminating between two proposed explanations.

The other explanation is derived from PHINEAS' approximate knowledge of Bernoulli's effect on airplane wings - that air flow can cause upward pull from above (this is a simplification of the actual pressure drop leading to the pull effect). The core component of this model is the relation:

```
Force(ball,position) < zero
Force(ball,position) ∝+ Amount-of(air,position)
```

When these two hypotheses are "run", we find that only the second produces stability, as evidenced by the oscillatory envisionment it produces. If the air were pushing on the sides of the ball in proportion to how much air is on a given side, offsetting the ball to the right would lead to further pushing by the air to the right – positive feedback and instability. On the other hand, if the air where "pulling" on the sides due to pressure gradients, offsetting the ball to the right would lead to increased force toward the left – negative feedback and stability. Hence, the hypothesis drawn from abstract knowledge of Bernoulli's effect consistently leads to the desired behavior and is proposed as the explanation. The pushing hypothesis is rejected due its inconsistency with the desired behavior.

6.3.3 As analysis of coverage

If a set of explanations are produced and a non-zero subset are found consistent, then it is probably reasonable to reject the failed hypotheses on the grounds that better ones are known. However, if there is only one hypothesis, or every available hypothesis initially fails, then outright rejection must be replaced by analysis and revision.

A proposed explanation may correctly predict portions of the observation, yet leave other portions unexplained. Alternatively, it may correctly explain all of the observed behavior, yet predict additional, unrecorded behavior. The primary role of simulating a flawed model is to see exactly what behavior it does predict, so that points of discrepancy may be found between model and observation.

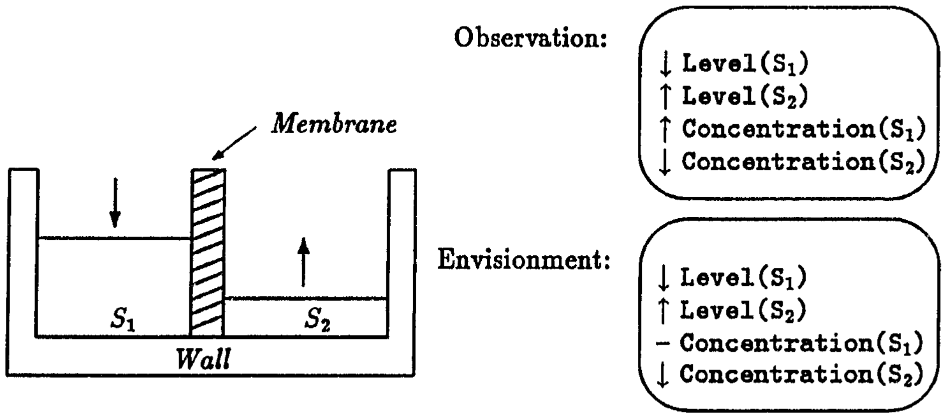


Figure 6.5: Determining the coverage of a proposed explanation for osmosis.

Consider the situation in Figure 6.5, which depicts two containers, sharing a common bottom and separated by a wall called "membrane". Each chamber contains a solution. The level of solution S_1 is observed to be decreasing while its concentration is increasing. At the same time, the level of solution S_2 is observed to be increasing while its concentration is decreasing. In other words, *osmosis* is taking place. When PHINEAS is assigned to explain the situation, it knows nothing about osmosis and focuses first on liquid flow due to the overall behavioral similarity. An initial "liquid flow" model is proposed and used to envision the possible behavior for the situation. This model corresponds to solution flowing through the membrane from S_1 to S_2. It correctly predicts the change in fluid levels, as well as the drop in S_2's concentration (due to flow of S_1's lower concentration). However, the model also states that S_1's concentration should remain constant, in conflict with its observed rise (see Figure 6.5). The model is therefore inadequate and must undergo revision, hopefully to realize that solvent is flowing rather than the entire solution.

6.4 Further Verification

Given that the new model is consistent with what has been observed, we may now seek further empirical confirmation of its validity through various stages of experimentation. A time-based planner has been used to explore the possibility of constructing simple experiments to perform *prediction verification.* More complex experiments may have to be performed to verify *extensions of the analogy.*

6.4.1 Verifying local predictions of the model

Given that the consistency of the model has been confirmed by finding a path in the envisionment that explains the current situation, what may be said about the other paths in this new envisionment? The new model states that the system should be able to exhibit all the behaviors described by the envisionment.

A temporal planner, TPLAN (Hogge, 1987c), and its associated *domain compiler* (Hogge, 1987b, 1987a) have been used to explore constructing simple experiments for local prediction verification.[5] This planner is able to develop plans leading to situations in the envisionment, allowing it to manipulate the scenario through every path and confirm or disconfirm the validity of the model's predictions. Only one example was successfully completed until problems in planning technology (and other commitments) delayed further investigation, such as the need to plan for prevention (Hogge, 1988). However, this work demonstrates several nice properties. First, the domain compiler enables dynamic generation of new planning knowledge in response to learning new theories about the physical world. Learning a model of heat flow enables learning new planning knowledge to raise the temperature of objects. Second, because the envisionment explicitly describes all states relevant to local prediction verification, creative experiment design of novel situations is unnecessary.

Consider the scenario predictions generated for a novel heat flow situation in which a hot brick is cooling off in hot water. Its model may further predict the bidirectionality of heat flow, that is, the opposite behavior should occur if a cool brick is placed in hot water. First, operators for heating and cooling are created from the system's new knowledge about heat flow. Second, the goal is posted to achieve the "cool brick in hot water" situation identified as one of the unconfirmed starting states of the scenario predictions. Starting from the current equilibrium state, TPLAN generates a plan to (1) remove the brick from the water (to prevent its heating up), (2) heat the water by placing it in contact with a hot stove, and (3) return the brick to the now hotter water. When the plan is executed, behavior leading from the initial achieved state may be observed and found to match prediction. Notice that had the system been testing a new electrical flow through a diode model, rather than heat flow, the experiment would have failed, uncovering a flaw in the model.

[5]This work was done in collaboration with John Hogge.

6.4.2 Verifying extended predictions of the analogy

If an analogy proves useful in understanding a given phenomenon, it would be wise to extend the analogy further and explore the limits of the analogy's validity. For example, the water flow – heat flow analogy may be extended by hypothesizing that heat is itself a type of fluid and possesses the properties known to hold for fluids (the *caloric* theory of heat). By extending the analogy in this manner, we are forced to conjecture a law of conservation of heat which states that heat can never be lost nor created. In the early nineteenth century, the caloric theory was widely believed and evidence for or against conservation of heat was sought. It was Count Rumford's experiments with friction which helped lead to the eventual downfall of the caloric theory of heat and supported the energy interpretation. While the original flow model may remain essentially intact, its theoretical underpinnings originating from extending the analogy to conjecture a heat fluid must be replaced by a notion of *enery* flowing.

6.5 Psychological Relevance

The qualitative simulation phase of testing used in PHINEAS is drawn from people's tendencies to mentally "try out" ideas before acting on them, often called imagery or a *gedanken experiment.*

Gedanken experiments are frequently used in science, both as an explanatory device or as a mechanism for creative insight (e.g., Newton, 1729; Einstein & Infeld, 1938; Dreistadt, 1968; Leatherdale, 1974; Miller, 1986). For example, Bohr and Heisenberg explained the Heisenburg uncertainty principle by imagining an attempt to determine the current state of an electron, that is, its position and velocity. First, obtain a powerful microscope with illuminating light of wave-length smaller than the dimensions of the electron, say γ-rays ($10^{-??}$cm) (never mind for now if such a microscope is feasible). Now, imagining the actual observation we suddenly get an unanticipated effect – at this wave length, the illuminating light bombards the electron, knocking it out of view. Attempting to view the electron has changed it's initial velocity by an unpredictable amount (Hanson, 1958, pg. 137).

A revolutionary gedanken experiment was conceived by Einstein in 1895 which laid the foundation for his development of the special theory of relativity. Einstein imagined an experimenter traveling at the speed of light. Physics at that time dictated that the experimenter should be able to follow a light wave, and hence view it as behaving like a standing wave. However, according to Einstein's intuition, the laws of optics should be independent of the observer's motion. His thought experiment represented an important paradox.

However, gedanken experiments are not limited to scientific investigation. People have been known to invoke mental simulations for interpretation, prediction, or analysis of conjectures in everyday reasoning about the physical world (e.g., Waltz, 1981; Gentner & Stevens, 1983). How often have we observed a person make a quick guess to explain something, think for a short time, and then declare "No, that doesn't make sense after all."? For example, Williams *et al* (1983) describe a subject explaining the operation of a heat exchanger. The subject would mentally simulate what transitions should take place. Conflicting predictions, or predictions that could not be justified, triggered a new round of model development. Collins & Gentner (1987) found similar results in response to the question "How does evaporation affect water temperature?". Their subject initially guessed that it doesn't. Then, the subject imagined that the water molecules able to leave the surface of the water must be the most energetic, thus lowering the average temperature. Finally, the subject imagined the surface of a lake being warmed by the sun, with the deeper layers at a lower temperature. As successive layers evaporated, the level would drop, new layers would be warmed, and the average temperature would increase.

Chapter 7

Theory Revision

When a proposed hypothesis is found to be inadequate during verification, that hypothesis must be revised before it may be accepted. Since analogy can produce flawed or incomplete inferences, any general model of analogical learning must include a complementary model of hypothesis repair.

Most accounts of theory formation and revision have failed to depict the experiential, integrated nature of theory development. They focus solely on first-principles analysis of failed theories (e.g., Dietterich & Buchanan, 1983; Rose & Langley, 1986; Rajamoney, 1988a). But why should the generation of revision hypotheses be fundamentally different from the generation of explanatory hypotheses in the first place? Past experiences should still be examined to see if any known phenomena might explain the anomaly. Furthermore, the additional information available during revision may enable use of hypothesis generation techniques that were inapplicable before.

PHINEAS cannot interact with the world. Instead, it uses a knowledge-intensive, heuristic approach to theory revision. This chapter discusses three techniques to analyze anomalies and propose plausible revisions. A *first principles analysis* provides a strong foundation and exhaustive source of hypotheses. *Precedent-guided revision* uses knowledge of analogous phenomena to provide a focused, ordered set of hypotheses. Finally, *difference-based reasoning* examines prior successes to see what is different about the current situation that may provide empirical justification for proposed hypotheses.

This chapter begins by overviewing the revision process, showing how the three revision techniques combine to provide a focused source of hypotheses. It then describes each revision technique in turn. *They are only partially implemented*, so the reader should assume everything described in this chapter is unimplemented unless explicitly stated otherwise. The chapter closes with an example and a perspective look at how this work relates to the general theory revision problem.

7.1 Overview

A newly proposed hypothesis may be flawed in several ways. It may interact with prior beliefs, leading to various kinds of inconsistency. It may only provide a partial explanation, thus requiring further interpretation. The hypothesis may be overzealous in that it predicts behaviors that did not occur. Finally, it may suffer from a generalization flaw, in which the hypothesis is primarily correct but the conditions on when it should and should not be applied are inaccurate.

Repairing a faulty hypothesis is a two stage process:

1. *Credit assignment.* Identify the portion(s) of the hypothesis responsible for the incorrect conclusions, either explicitly or by identifying general classes of problems.

2. *Repair.* Modify the hypothesis so that offending conclusions will be retracted and correct conclusions will be maintained.

A number of approaches to the credit assignment problem have been proposed. Typically, they involve analyzing a trace of program execution or dependency structure to isolate "points" of failure. For example, an operator leading down the wrong search path may be identified (Mitchell et al., 1983) or a wrong conclusion traced to an inappropriately applied rule or assumption (Smith et al., 1985; Rose & Langley, 1986).

There are three aspects of PHINEAS' task that lead to difficulties for these approaches. First, there is no single answer or goal state, but rather a multi-state description that is supposed to contain a subsequence of states matching an observed continuous behavior. Hence, the same model or assumption might correctly predict one state while failing to match another, all for the same scenario. Second, continuous quantities may have multiple influences. If a quantity is predicted to be increasing but found to be constant, it's not necessarily the case that the predicted positive influence is incorrect. There might be an additional, unknown negative influence cancelling the positive influence. Finally, a model will typically drive predictions about multiple components of a scenario (e.g., amount, pressure, and volume). Hence, a single faulty model may lead to non-local failures. These problems are all analogous to the problems encountered in multiple fault diagnosis (de Kleer & Williams, 1987).

An example is useful at this point to help clarify the revision task. Suppose in response to the disappearing alcohol observation PHINEAS had learned the "evaporation" model shown in Figure 7.1.[1] This model indicates that any liquid in a container will evaporate

[1]This model is taken from (Falkenhainer & Rajamoney, 1988). It was learned by PHINEAS with the aid

```
(defProcess (PROCESS-3318 ?v-3309 ?v-3310 ?v-3311 ?v-3312)
        Individuals  ((?V-3309 :conditions (Substance ?V-3309))
                      (?V-3310 :conditions (Can-Contain ?V-3310 ?V-3309))
                      (?V-3311 :conditions (Contained-Liquid ?V-3311)
                                           (Container-of ?V-3311 ?V-3310)
                                           (Substance-of ?V-3311 ?V-3309))
                      (?V-3312 :conditions (Contained-Gas ?V-3312)
                                           (Container-of ?V-3312 ?V-3310)
                                           (Substance-of ?V-3312 ?V-3309)))
        QuantityConditions  ((Greater-Than (A (Amount-of ?V-3311)) zero))
        Relations  ((Quantity (Vaporization-Rate ?self))
                    (Greater-Than (A (Vaporization-Rate ?self)) zero))
        Influences ((I- (Heat ?V-3311) (A (Vaporization-Rate ?self)))
                    (Ctrans (Amount-of ?V-3311) (Amount-of ?V-3312)
                           (A (Vaporization-Rate ?self)))))
```

Figure 7.1: A flawed model of evaporation. This model indicates that any liquid in a container will evaporate until the container is empty.

until the container is empty. Consider what happens when that same model is applied to a new situation in which alcohol is placed in a closed container. The observed behavior and the model's incorrect prediction are shown in Figure 7.2. What went wrong? Examination of the justification structure shown in Figure 7.3, which states why the model predicts a decrease in the alcohol, provides some clues. The derivative of the amount of alcohol is less than zero because the alcohol has the quantity `amount-of`, process `PROCESS-3318` is active as the process instance `PI0`, and its vaporization rate is greater than zero. Further examination reveals that `PI0` is active because it "exists" (i.e., instantiates on known individuals) and the quantity condition "amount of alcohol greater than zero" is true. Note that there are many places where a justification might be changed to defeat the flawed belief. The first simplification made is to reason at the level of processes, rather than about each individual belief. Thus, the reasons for believing `Active(PI0)` are most relevant. What changes will remove the erroneous belief? One possibility is to add a new quantity condition, for example that the amount of alcohol must be greater than some lower limit point `l(alcohol1)`. Alternatively, the process' stated effect on the amount of alcohol could be

of Rajamoney's directed experimentation system (ADEPT). PHINEAS is unable to develop this exact model (it was derived from boiling) when operating on its own, but the model has useful characteristics for this discussion.

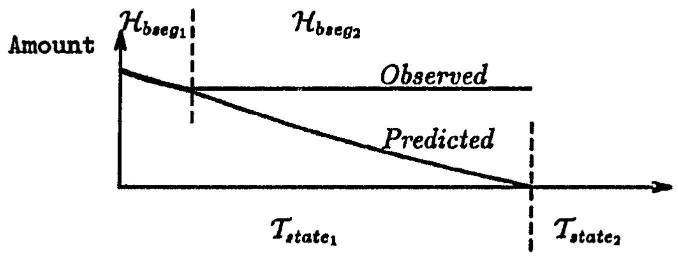


Figure 7.2: Observed behavior of alcohol in a closed container and the behavior predicted by the flawed evaporation model.

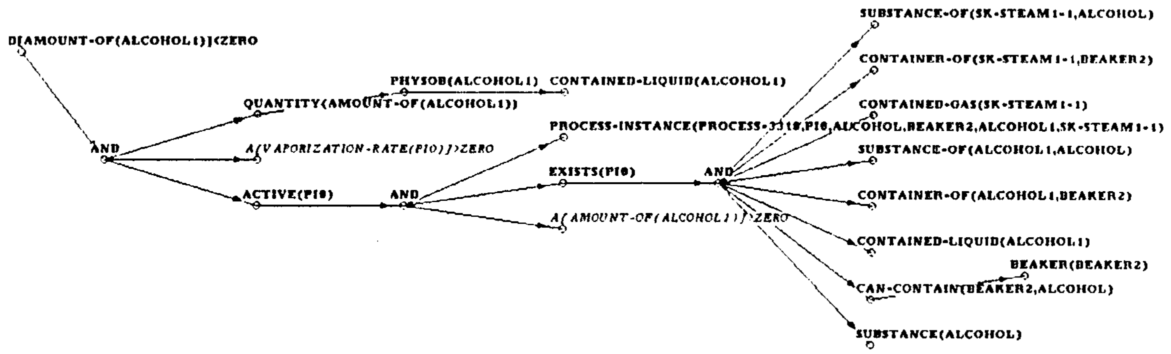

D[AMOUNT-OF(ALCOHOL1)]<ZERO
AND
QUANTITY(AMOUNT-OF(ALCOHOL1))
PHYSOB(ALCOHOL1)
CONTAINED-LIQUID(ALCOHOL1)
A[VAPORIZATION-RATE(PI0)]>ZERO
ACTIVE(PI0)
AND
PROCESS-INSTANCE(PROCESS-3318,PI0,ALCOHOL,BEAKER2,ALCOHOL1,SK-STEAM1-1)
EXISTS(PI0)
A[AMOUNT-OF(ALCOHOL1)]>ZERO
AND
SUBSTANCE-OF(SK-STEAM1-1,ALCOHOL)
CONTAINER-OF(SK-STEAM1-1,BEAKER2)
CONTAINED-GAS(SK-STEAM1-1)
SUBSTANCE-OF(ALCOHOL1,ALCOHOL)
CONTAINER-OF(ALCOHOL1,BEAKER2)
CONTAINED-LIQUID(ALCOHOL1)
BEAKER(BEAKER2)
CAN-CONTAIN(BEAKER2,ALCOHOL)
SUBSTANCE(ALCOHOL)

Figure 7.3: Justification structure produced by PROCESS-3318 indicating why the derivative of the alcohol's amount should be negative. Nodes in italic indicate assumptions made by QPE during envisioning.

removed (the Ctrans expression). Caution is needed, for repairing an erroneous prediction in one envisionment state may reverse a correct prediction in another envisionment state.

Due to these complexities, rather than examine the underlying beliefs behind each incorrect parameter prediction one at a time, global anomalies are classified into failure categories. This categorization is then used by the revision processes to determine possible revisions. In describing the alternate categories, the following simplification is made:

- *Process simplification:* Only revision of flawed process models are considered. Individual rules and entity definitions are not subject to change.

A process is syntactically equivalent to a schema definition. It specifies constraints on the objects it may apply to, a set of conditioning relations that state when it may apply (i.e., a process' preconditions and quantity conditions), and a set of consequent effects on the world. A process is considered *active* when it is applied in a given situation and *inactive* when it does not apply. For example, a process' preconditions and quantity conditions together indicate when it is active and inactive. Note that the same process may be both active and inactive in the same envisionment, but never in the same state of that envisionment.

Two assumptions are made in the revision process:

- *Single influence assumption:* There are no influences on an observed quantity that is constant.

By this assumption, the case of a quantity being constant due to equal and opposite influences on it is not considered.

- *Unit influence-chain assumption:* If the hypothesis that Q_1 is influencing Q_2 is being proposed, but is currently not believed, the chain of influences from Q_1 to Q_2 is of length one.

By this assumption, the case of *hidden influencers* is not considered. This prevents proposing an influence from Q_1 to Q_2 of the form $Q_2 \propto \ldots Q_i \ldots \propto Q_1$, where each Q_i is an unknown, hypothesized quantity.

Due to the limited resolution of qualitative models, qualitative simulators are designed to branch in times of ambiguity. A given model may produce a large number of scenario predictions, corresponding to different run-time assumptions about quantity relationships, object existence, and influence ambiguities. A revision problem that will not have to be addressed is revising these types of assumptions, since the qualitative simulator will automatically branch on these ambiguities.

7.1.1 Combining Analysis with Experience Provides Focus

In summary, revision proceeds as follows:

1. *Failure detection and isolation.* Failures are detected when the verification process classifies a hypothesis as inadequate. The registration indicates which states of the behavior are anomalous.

2. *Failure classification.* The type of failure is classified (e.g., *premature stop*, *should cause*, etc.).

3. *Propose revisions.* Potential revisions of the theory are hypothesized to enable it to correctly explain the previously anomalous observation.

4. *Select revision(s).* A subset of the proposed revisions is chosen and revised theories formed. These theories must be reexamined for adequacy.

The task of proposing and selecting revision hypotheses is severely underconstrained. Using a purely formal analysis of a prediction's causal structure, the number of possible revisions is infinite (e.g., creating an endless transitive chain of new quantities and proportionalities to hypothesize an influence on q_1 by q_2). Even with the stated assumptions, analysis of the causal structure alone will produce a large number of possible revisions for a model of any complexity, making it extremely difficult to identify the "one true fix". Additionally, there is no clear best selection criteria. Rose and Langley (1986) have proposed a *cost* measure for ordering possible revisions, in which revisions to premises supporting the fewest beliefs are preferred. However, they correctly observed that some hypotheses about an anomaly seem more plausible than others, for which the cost measure does not account.[2] I propose that this sense of plausibility arises from experience, both with behavior analogous to a given anomaly and with prior applications of the theory under investigation. These two forms of analogy may be used in tandem with a first-principles analysis to provide a preferential ordering and empirical support for proposed revision hypotheses.

The revision process is depicted in Figure 7.4. During the verification stage, anomalies are detected through the registration; bsegs having no interpretation correspond to physical states where the theory fails. The failure is classified according to a set of failure categories. *First principles analysis* is then used to examine the behavior history, the failure classification, and the flawed theoretical predictions to propose a list of possible changes to the theory. This list is typically long and always unordered. *Precedent-guided revision* also proposes revisions, but from a different perspective. It seeks analogues to the current observation that display the same behavioral aspect causing the current problem. It then determines how that aspect was explained in the analogue case and suggests revisions that would produce a similar explanation. Revisions drawn from analogous situation explanations have experiential corroboration and are preferred over revisions drawn solely from an analysis of the situation in isolation. *Difference-based reasoning* seeks empirical explanation for the change in behavior between the current anomalous case and a prior observation successfully explained by the model under investigation. It identifies how the two scenario descriptions differ ($\Delta(\mathcal{H}_{old}, \mathcal{H}_{new})$) and attempts to determine which quantities might be affected by that change. Revisions concerning those quantities have empirical corroboration, while revisions about other quantities are deemed less desirable.

[2]In diagnostic reasoning, this observation is addressed by use of probability measures on possible faults (e.g., Buchanan & Shortliffe, 1984; de Kleer & Williams, 1987). It is not clear how Bayesian probability measures for flaws in a model would be obtained.

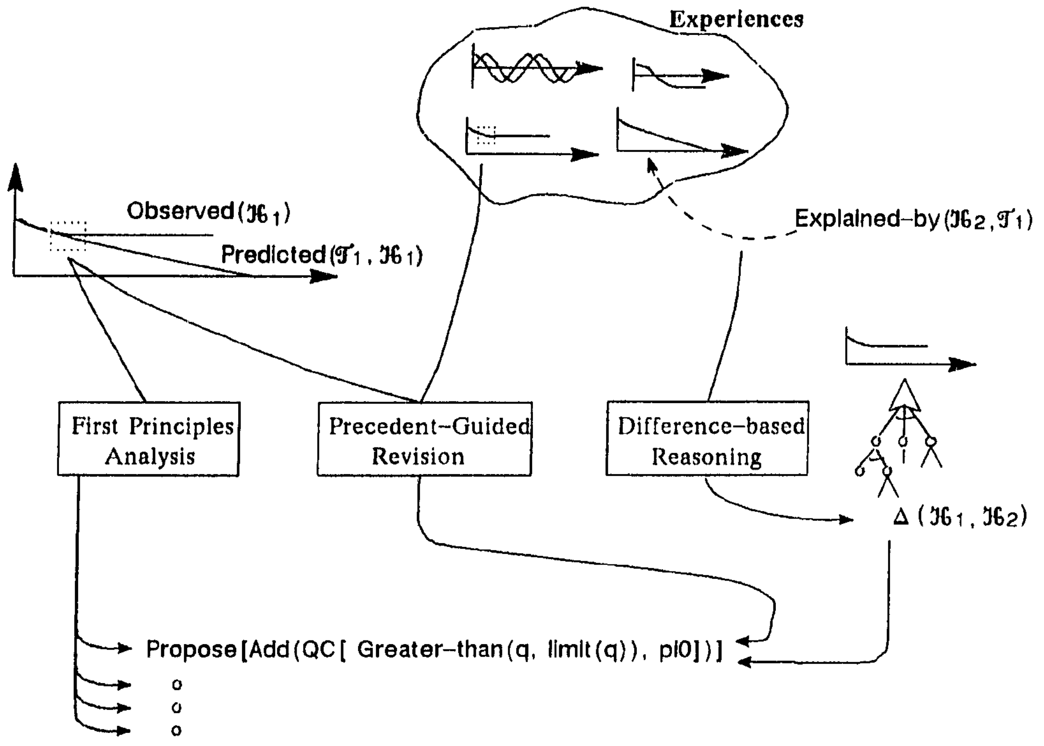

Experiences
Observed($\mathcal{M}_1$)
Predicted($\mathcal{T}_1$, $\mathcal{M}_1$)
Explained-by($\mathcal{M}_2$,$\mathcal{T}_1$)
First Principles Analysis
Precedent-Guided Revision
Difference-based Reasoning
Δ ($\mathcal{M}_1$, $\mathcal{M}_2$)
Propose[Add(QC[Greater-than(q, limit(q)), pi0])]
o
o
o

Figure 7.4: Overview of the proposed PHINEAS revision process.

7.1.2 Categorizing the Flaw

The *NoInterpret(b_j)* statements of a registration between model and observation indicate specific behavioral segments that were considered anomalous. Failures are detected and isolated by examining the individual bseg disagreements, or considering the behavior of a sequence of uninterpretted bsegs. Points of disagreement are then classified according to the categories listed in Table 7.1. These categories are similar in intent to the taxonomy discussed in (Smith et al., 1985). However, they primarily assumed correctness of the underlying domain theory and examined failures due to violated assumptions.

Anomalies may be of three general types: *conditioning relation*, *effects relation*, or *behavioral gap*. Each of these distinctions has an associated set of revision distinctions.

7.1.2.1 Conditioning relation

The conditioning relations of a process (i.e., preconditions and quantity conditions) indicate when the process is active and inactive. Problems related to incorrect conditioning rela-

- *Conditioning relation.* If knowledge is applied when it shouldn't, or not applied when it should, then the conditioning relations describing applicability must be revised. This corresponds to the following pair of possibilities:

  1. `Prevent`: A theory was applied when it should not have been. This may be further classified into:
     (a) `Stopped`: The theory says it should be active, the behavioral state indicates inactive, and the theory was correctly active in some non-empty set of states. Having time in the representation enables two further distinctions:
        i. `Premature-stop`: Stopped prior to, but on the way to, stopping normally.
        ii. `Chance-stop`: An unanticipated ending.
     (b) `Blocked`: The behavior indicates always inactive, while the theory sometimes or always indicates active.
  2. `Cause`: A theory was not applied when it should have been. This may be further classified into:
     (a) `Kept-going`: The behavior kept going in all the same directions, yet the theory driving it went inactive.
     (b) `Is-going`: The theory is always inactive, while the behavior indicates that it should be active at times.

- *Effects relation.* When a packet of knowledge is properly applied, it may still be inadequate if its believed effects are in error. There are two possibilities:

  1. `Should-cause`: Additional behavior was observed that wasn't accounted for.
  2. `Shouldnt-cause`: Additional behavior was predicted that didn't occur.

- *Behavioral gap.* If a sub-sequence of behavioral states is unexplained, perhaps with surrounding states properly explained, there might be more wrong than simply a precondition or effect relation of a theory. Additional theories may be required to incrementally develop a complete picture through multiple models. Alternatively, the existing theory might be faulty beyond repair and need to be replaced.

Table 7.1: Categories of potential flaws.

tions appear in two forms: a process is not prevented from being active when it shouldn't (Prevent), or excessive constraint is causing a process to be inactive when it shouldn't (Cause).

The Prevent condition holds for an active process instance if every quantity influenced by that process (1) conflicts in the given bseg and (2) was observed to be constant during that bseg. The Prevent condition may be further specialized. If the process instance is correctly active in at least one bseg, then the problem is the more specific Stopped condition. Otherwise it is a Blocked condition. A Stopped condition is classified as a Premature-stop if the process stopped prior to, but on the way to, stopping normally. It is classified as a Chance-stop if no future stop was anticipated, that is, it was not heading for a limit point. The evaporation anomaly described above was of type Premature-stop.

The Cause condition holds for an inactive process instance if every quantity influenced by that process (1) conflicts in the given bseg and (2) was observed to be changing during that bseg. The Cause condition is further specialized into two subcategories. It is a Kept-Going condition if the process was active during the previous bseg, the previous bseg was correctly interpreted, and the derivatives of all quantities influenced by the process instance are the same in both the current and previous bsegs. It is a Is-Going condition if the process is never active in any bseg of the observation, while the quantities it influences are observed to be changing in some bsegs for which *NoInterpret* is true.

7.1.2.2 Effects relation

The effects relations of a process indicate how it influences continuous quantities when active. There are two types of influence forms. Qprop(q_1,q_2) indicates that q_1 *is qualitatively proportional to* q_2. In QP theory this is called an *indirect influence.* All else being equal, q_1 will increase if q_2 increases and decrease if q_2 decreases. I+(q_1,q_2) indicates that the derivative of q_1, $\dot{q}_1$, is equal to the sum $q_i \ldots + q_2 + \ldots q_j$. In QP theory this is called a *direct influence.*

Problems related to incorrect effects relations appear in two forms: If processes are believed active for an uninterpreted bseg and all of the quantities they influence are changing in a manner consistent with those influences, while other uninfluenced quantities are observed to be changing, then the condition Should-Cause holds for that bseg. If there are quantities changing in an uninterpreted bseg in accordance with influences from processes believed to be active in that bseg, while other quantities influenced by those processes are constant in the given bseg, then the condition Shouldnt-Cause holds for that bseg.

For example, if a proposed model of osmosis (solvent flowing through a membrane) incorrectly states that solution is flowing, rather than solvent alone, it will correctly predict the change in solution amount but incorrectly predict that the solution's concentration will remain constant. This is a `Should-cause` anomaly, since the osmosis model *should cause* a change in the concentration quantity.

Note that these conditions are sensitive to the consistency of adjacent bsegs. If a quantity is consistently predicted to be influenced by an active process in one bseg, then `Shouldnt-Cause` will *not* be true of the next bseg if that quantity is constant but predicted to be changing during that next bseg.

7.1.2.3 Behavioral gap

If an anomaly cannot be classified as a specific problem with conditioning or effects relations, it is classified as a *behavioral gap.* A behavioral gap indicates there is a severe problem with the model. For example, if there is a sequence of behavioral segments for which *NoInterpret* is true, it might be the case that the model is incomplete and additional explanation hypotheses are needed. This case is currently not addressed.

7.2 First Principles Analysis

A *first principles* approach to revision is one which depends strongly on the underlying domain theory, analyzing the causal structure supporting a flawed set of beliefs and using only weak heuristics to guide search. It is a powerful technique in that the reasons for a flawed belief may be identified explicitly. Previous sections have described how anomalous bsegs are detected and categorized. This section describes how those classifications may be used to propose revisions to a flawed process description.

There are five types of revision hypotheses, which either `Add`, `Remove`, or `Change` elements of a flawed theory:

- *Quantity condition modification.* Add or remove a quantity condition for a specified process (e.g., $\texttt{Greater-than}(q_1,q_2)$).

  $$\texttt{Propose(Add[QC(}\textit{inequality}\texttt{[}\textit{quantity}_1\texttt{,}\textit{quantity}_2\texttt{]}, \textit{process}\texttt{)}, \textit{theory}\texttt{])}$$

- *Precondition modification.* Add or remove a precondition for a specified process (e.g., `Valves-Open(`*fluid-path*`)`).

  $$\texttt{Propose(Add[PC(}\textit{atomic-sentence}, \textit{process}\texttt{)}, \textit{theory}\texttt{])}$$

- *Effect modification.* Add or remove an effect relation of a specified process (e.g., $\texttt{Qprop}(q_1,q_2)$).

Propose(Add[Effect(*influence*, *process*), *theory*])

- *Participants modification.* Change the individuals taking part in a specified process.

  Propose(Change[Individual(obj_{old}, *process*),Individual(obj_{new}, *process*), *theory*])

- *Merge models.* Recursively invoke the normal interpretation process to augment the existing theory and incrementally develop a complete global interpretation. For example, fully explaining a single observation possessing both thermal and chemical aspects may require multiple analogies.

At this time, only modification of conditioning relations, effects relations, and a process' participatory objects will be considered. Full recursive invocation of PHINEAS, leading to integration of multiple models of the observation is an important problem, but beyond the scope of this thesis.

The ability to examine the behavior history is central to this process. If a model is inconsistent with some state, what the behavior and model were doing in the previous state, as well as in the next state, will have a lot to say about the problem. Static analysis of a single anomalous state lacks important contextual information that may rule out many revision hypotheses. For example, suppose a quantity is decreasing, which is consistent with the model, then suddenly stops while the model states it should continue to decrease. The quantity may have reached an important limit point. From the prior history, it is clear that if such a limit point was reached, it was approached from above, not below. Thus, QC[q > l(q)] is a potential new quantity condition, while QC[q < l(q)] is not.

Revision hypotheses are proposed by running a set of rules that examine the failure category, the observed behavior, and the theory's predictions. The rules described below are fully implemented, but the current set is incomplete. Therefore, a representative sample will be described.[3] In describing these rules, a number of quantified variables are used. th is the theory under investigation, pi is a process instance that is part of the theory, s, s1, s2 ... are behavioral segments of the observation, and q denotes a quantity.

7.2.1 Conditioning relations

Revision of a process' conditioning relations is sanctioned when an anomaly is associated with one of the Prevent or Cause categories. The revision proposal rules examine the predicted activity of the process and compare that to its *apparent* activity as indicated by the observation.

[3]Some of the rules described were originally developed by Shankar Rajamoney. Rajamoney (1988b) is examining the same revision problem from a different perspective.

7.2.1.1 Prevent category

A **prevent** condition occurs when a process is believed active when the behavior indicates it is inactive. If the behavior indicates the process has transitioned from active to inactive, or is inactive and will become active at some latter time, then the process is in a temporary **stopped** condition.

The following rule addresses the case of a process failing to transition from active to inactive.

Rule 1 (Approaching from above) *If a quantity is decreasing and then prematurely stops, it may have reached an important limit point,* `limit(q)`*, that the quantity must be greater than.*

```
Behavior-Indicates[th, During(Premature-Stop(pi), s2)] ∧
Process-Influences(pi, q) ∧
Observed[During(Decreasing(q), s1)] ∧ Meets(s1, s2)
        ⇒ Propose(Add[QC(Greater-than[q, limit(q)], pi), th])
```

The rule applies to a situation where pi, which influences quantity q, prematurely stopped and q was decreasing in the previous bseg. It proposes adding a quantity condition that requires q to be greater than some limit value for that quantity (yet to be determined). A reciprocal rule adds a **Less-than** quantity condition if the quantity was increasing in the previous state.

A specialization of this rule accounts for two quantities of the same type approaching each other toward equality:

Rule 2 (Dual approach) *If a quantity decreasing for one object and increasing for another object prematurely stop at the same instant they reach equality, it may be that the decreasing quantity must be greater than the increasing quantity for the process to be active.*

```
Behavior-Indicates[th, During(Premature-Stop(pi), s2)] ∧
Process-Influences(pi, q(obj1)) ∧ Process-Influences(pi, q(obj2)) ∧
Observed[During(Decreasing(q(obj1)), s1)] ∧
Observed[During(Increasing(q(obj2)), s1)] ∧ Meets(s1, s2)
        ⇒ Propose(Add[QC(Greater-than[q(obj1), q(obj2)], pi), th])
```

A process' activity may also be conditioned on an equality relationship between two quantities:

Rule 3 (Transition from equality) *If an equality transitions to inequality at the same moment a process prematurely stops, that equality may be a prerequisite for the process' activity.*

```
Behavior-Indicates[th, During(Premature-Stop(pi), s2)] ∧
Observed[During(Constant(q1), s1)] ∧ Observed[During(Equal-to(q1,q2), s1)] ∧
Observed[During(¬Equal-to(q1, q2), s2)] ∧ Meets(s1, s2)
        ⇒ Propose(Add[QC(Equal-to[q1, q2], pi), th])
```

When the behavior indicates a process should never be active, while the process is believed to be active during periods of the observation (i.e., Blocked), a precondition may be absent. Preconditions state important physical conditions whose change cannot be predicted in terms of changes to continuous quantities (e.g., a switch being on or off).

Rule 4 (Missing precondition) *If the behavior indicates a process is* blocked, *an unknown condition,* c, *for the process' activity may not hold in the scenario.*

```
Behavior-Indicates[th, Blocked(pi)]
        ⇒ Propose(Add[PC(c(pi), pi), th])
```

7.2.1.2 Cause category

A cause failure occurs when a process is believed inactive when the behavior indicates it is active. If the process incorrectly transitions from active to inactive, then the anomaly is a kept-going condition.

Rule 5 (Unnecessary greater-than condition) *If a process transitions from active to inactive due to a* greater-than *quantity condition, while the behavior indicates a* kept-going *condition, that quantity condition may be unnecessary.*

```
Behavior-Indicates[th, During(Kept-Going(pi), s2)] ∧
QC(Greater-than[q1, q2], ?pi) ∧
Observed[During(Greater-than(q1,q2), s1)] ∧
Observed[During(Equal-to(q1,q2), s2)] ∧ Meets(s1, s2)
        ⇒ Propose(Remove[QC(Greater-than[q, limit(q)], pi), th])
```

A reciprocal rule removes a Less-than quantity condition if it was the reason for a process stopping in a kept-going state. Two other rules remove equality quantity conditions whose change to inequality was the reason for a process stopping in a kept-going state.

If a process description contains an unnecessary precondition, failure of that precondition will lead to occasions when the process is incorrectly classified as inactive throughout a given scenario (the is-going category):

Rule 6 (Spurious precondition) *If the behavior indicates a process is active, while the process is believed to be inactive throughout the scenario (i.e.,* is-going*), one of the process' preconditions may be unnecessary.*

```
Behavior-Indicates[th, Is-Going(pi)] ∧ PC(c, pi)
        ⇒ Propose(Remove[PC(c, pi), th])
```

7.2.2 Effects relations

A **should-cause** condition occurs when a process correctly predicts the behavior of all quantities it influences, yet other quantities are observed to be changing that the process does not influence.

Rule 7 (Missing proportionality) *If an uninfluenced quantity is observed to be changing, it may be proportional to an influenced quantity that is changing.*

```
Behavior-Indicates[th, Should-cause(pi)] ∧
Process-Influences(pi, q1) ∧ ¬Process-Influences(pi, q2) ∧
During(Active(pi), s) ∧
Observed[During(Increasing(q1), s)] ∧ Observed[During(Increasing(q2), s)]
        ⇒ Propose(Add[Effect(Qprop(q2,q1) pi), th])
```

Three other rules treat the various permutations possible (e.g., q1 increasing and q2 decreasing leads to a `Qprop-(q2,q1)` (inversely proportional) being proposed).

A **shouldnt-cause** condition occurs when a process correctly predicts the behavior of some of the quantities it influences, while the other quantities it influences are observed to be constant.

Rule 8 (Spurious proportionality) *If an influenced quantity is observed to be constant while the quantity it is proportional to is changing, the belief in the proportionality may be incorrect.*

```
Behavior-Indicates[th, Shouldnt-cause(pi)] ∧
Effect(Qprop(q1 q2), pi) ∧ During(Active(pi), s) ∧
Observed[During(Constant(q1), s)] ∧ Observed[During(¬Constant(q2), s)]
        ⇒ Propose(Remove[Effect(Qprop(q1,q2) pi), th])
```

7.2.3 Participants

Some should-cause or shouldnt-cause conditions are due to the process describing the behavior of the wrong set of individuals. For example, when two objects or quantities are closely related (e.g., a solution and its solvent), a misaligned analogical mapping may be formed which places the wrong target item in correspondence with a given base item. The problem might be detected when the target item fails to support the required predictions.

Only the *decomposition rule* has been used:

Rule 9 (Decomposition) *If the amount of a mixture is correctly believed to be changing, but the belief that the relative proportions of the mixture's constituents are constant is incorrect, it may be that the amount of a constituent alone is changing, rather than the mixture as a whole.*

```
Behavior-Indicates[th, Should-cause(pi)] ∧
Individual(m, pi) ∧ Mixture(m) ∧ Component-of(m, c) ∧
Process-Influences(pi, Amount-of(m)) ∧
Observed[During(Decreasing(Amount-of(m)), s)] ∧
Tprediction[th, During(Decreasing(Amount-of(m)), s)] ∧
Observed[During(Decreasing(Percentage-of(c)), s)] ∧
Tprediction[th, During(Constant(Percentage-of(m)), s)]
        ⇒ Propose(Change[Individual(m, pi), Individual(c, pi), th])
```

7.2.4 Closed Container Example

Let us return to the situation described at the beginning of this chapter, in which alcohol in a closed container violated PHINEAS' model of evaporation. The first step in revising the model is to classify the apparent anomaly. In this example, the problem is a **premature-stop**: the model predicted that the amount of alcohol would decrease until the container was empty; the observation showed the alcohol's decrease stopped well before its amount reached zero.

The second step is to run the revision rules. For a **premature-stop**, this requires consideration of the various quantities' behavior immediately prior to the stop. The complete set of proposed revisions, in conjunction with the behavior that suggested each revision, is shown in the following table:

Derivative	*Value*	*Proposed Additions*
Ds[Amount-of(alcohol1)]	-1	QC(Greater-than[Amount-of(alcohol1),limit(alcohol1)])
Ds[Amount-of(sk-steam1-1)]	1	QC(Less-than[Amount-of(sk-steam1-1),limit(sk-steam1-1)])
		QC(Greater-than[Amount-of(alcohol1),Amount-of(sk-steam1-1)])
Ds[Pressure(alcohol1)]	-1	QC(Greater-than[Pressure(alcohol1),limit(alcohol1)])
Ds[Pressure(sk-steam1-1)]	1	QC(Less-than[Pressure(sk-steam1-1),limit(sk-steam1-1)])
Ds[Temperature(alcohol1)]	-1	QC(Greater-than[Temperature(alcohol1),limit(alcohol1)])
Ds[Heat(alcohol1)]	-1	QC(Greater-than[Heat(alcohol1),limit(alcohol1)])

Each of the seven proposed revisions will enable PROCESS-3318 to consistently explain the observed behavior. The question remaining is how to choose which revision to make. Notice how some revisions seem more implausible than others (e.g., QC(Greater-than[Amount-of(alcohol1),limit(alcohol1)])). This is the topic of the next two sections.

7.3 Precedent-Guided Revision

While first-principles analysis is able to identify the set of possible revisions for a flawed theory, it has two important limitations. First, it offers no preferential ordering on a set of revision hypotheses. Second, many of the revisions it proposes contain unknown quantities, expressed as skolem functions over an existing quantity. For example, the *approaching from above* rule proposes the quantity condition `Greater-than[q, limit(q)]`.

Precedent-guided revision addresses both limitations by recognizing that experience is an important factor in selecting among alternative revisions. First, it ranks revision hypotheses according to their experiential plausibility by seeking understood behavior that is analogous to the current anomaly. Second, adapting explanations of analogous behavior often attaches known concepts to the unknown quantities proposed by the first-principles analysis. For example, a quantity condition mapped from a prior explanation will have a known quantity in place of `limit(q)` in `Greater-than[q, limit(q)]`.

The procedure is an adaptation of PHINEAS' normal explanation process:

1. Identify the aspect of the current behavior that is anomalous.

2. Access potential analogues, requiring that the behavioral mapping contain the anomalous aspect of the current behavior. This uses PHINEAS' standard access mechanism, with a candidate analogue rejected if the behavioral match fails to contain a correspondence for the anomalous aspect of the current situation.

3. Identify what explained the relevant aspect of the analogue behavior. Once the relevant aspect of the analogue behavior is detected during access, the underlying explanation for that component of its behavior may be retrieved. This uses the same explanation retrieval mechanism used during PHINEAS' normal mapping process.

4. Map those explanation components to the current situation.

5. Propose the mapped elements as plausible revisions to the flawed theory under investigation.

The only component of this process that has not been described in previous chapters is the first – identify the anomalous aspect. This information is provided by the observation, registration and failure categorization. The process is best demonstrated by example.

7.3.1 Closed Container Example

Recall that the closed container anomaly for `PROCESS-3318` was a `premature-stop` failure. In terms of the specific situation, this indicates that the four liquid `alcohol1` quantities (`Amount-of`, `Pressure`, `Temperature`, and `Heat`) were decreasing, the two gas `sk-steam1-1` quantities (`Amount-of` and `Pressure`) were increasing, and the behavior stopped prior to the amount of liquid reaching zero. Thus, an understood behavior is sought which must provide a "decreasing and transitioning to stopped prior to reaching zero" analogue for the amount of alcohol1. Recall from Chapter 4 that PHINEAS knows of four analogues to the disappearing alcohol scenario: water flowing from a leaky cup, water flow between two containers, boiling, and dissolving. Not all of these analogues display the requisite behavioral aspect. The leaky cup scenario stops when the amount of water reaches zero. This is precisely the kind of behavior that is not relevant to explaining the current failure. The same reasoning rejects boiling as a relevant analogue. Mapping the liquid flow and dissolving domain theories suggests two possible revisions:

Liquid Flow: `QC(Greater-than[Pressure(alcohol1),Pressure(sk-steam1-1)])`
Dissolving: `QC(Less-than[Amount-of(sk-steam1-1),Saturation-Point(sk-steam1-1)])`

The pressure inequality condition mapped from the liquid flow situation is inconsistent with the observation. Since the pressure at the bottom of the liquid alcohol is always greater than the pressure of the gas alcohol, a transition to equality never occurred. The saturation point condition suggested by dissolving replaces a less precise version suggested earlier:

```
QC(Less-than[Amount-of(sk-steam1-1),limit(sk-steam1-1)])
```

7.4 Difference-Based Reasoning

The preceding two sections described formal and experiential grounds for generating and ranking revision hypotheses. One other important clue is often possible: can any of the novel features about the situation itself be used to explain the anomaly? Specifically, if the theory under revision has been used successfully in the past, what is novel about the current situation that could cause the observed change in behavior? By identifying and explaining the effects of differences between situations in which a theory is applied, *difference-based reasoning* (DBR) provides empirical evidence for what revision hypotheses to consider.

DBR is a general technique designed to facilitate the resolution of expectation failure. (Falkenhainer, 1988c) describes its use in theory formation, diagnosis, and planning failure explanation. It is relevant to situations in which an expectation was violated and an instance of the desired performance is available. In the context of PHINEAS, the expectation failure corresponds to violation of a theory and the instance of desired performance corresponds to a prior, successful application of the theory.

Here we examine a very special interpretation of DBR: if the theory has been used successfully in the past and a difference may be identified between the current and previous situations, determine what quantities that difference has the potential to affect. Any revision hypotheses that do not mention these quantities should be discounted.

In PHINEAS, DBR consists of four stages:

1. Retrieve a situation description successfully explained by the theory under investigation. Each process description lists the situations it has explained, just as each situation stores pointers to the domain theory that explained it.

2. Compute Δ, the set of differences between the current and retrieved situations. This computation is performed by SME, where Δ is defined to be those aspects that failed to be placed in correspondence, that is

$$\Delta = [\mathcal{H}_{old} - M] \cup [\mathcal{H}_{new} - M]$$

   where M represents the analogical mapping produced by SME for the base and target descriptions $\mathcal{H}_{old}$ and $\mathcal{H}_{new}$.

3. Use the domain theory to predict the behavioral changes Δ could cause.

4. Favor revision hypotheses concerning quantities affected by Δ over revisions concerning unaffected quantities.

The only component of this process not described previously is the third – predict the behavioral changes Δ could cause. In general, this is a very hard problem. The most advanced work on this topic is Weld's (1988) *comparative analysis*, a technique for determining the effects of qualitative changes to a system's continuous parameters. However, it is not applicable to structural modifications, additions, or deletions. In PHINEAS, a simple mechanism for achieving the desired affect is available: if the domain theory has the ability to predict behavioral changes caused by Δ, these changes will appear as differences in the theory's predictions for the two situations.[4] Hence, rather than explicitly examining all possible ramifications of Δ, the quantities empirically relevant to revision are those for which different behavior was predicted. This requires reexplaining the prior situation (using QPE) and comparing that explanation to the current anomalous prediction.

Note that DBR is neutral with respect to revision hypotheses involving quantities for which the domain theory makes no prediction. Such hypotheses may be produced by the precedent-guided analysis, which is capable of introducing new quantities.

[4]This only notes changes such as *constant* to *increasing*. It does not indicate changes in degree, such as an increased rate. Weld's comparative analysis would be needed to detect such differences.

7.4.1 Closed Container Example

PROCESS-3318, which fails to predict the behavior of alcohol in a closed container, was originally developed to explain an observation of alcohol in an open container. Consider the predicted behavior for the first state in each of the two scenarios:

Derivative	*Container Open*	*Container Closed*
Ds[Amount-of(alcohol1)]	-1	-1
Ds[Amount-of(sk-steam1-1)]	1	1
Ds[Pressure(alcohol1)]	-1	-1
Ds[Pressure(sk-steam1-1)]	0	1
Ds[Temperature(alcohol1)]	-1	-1
Ds[Heat(alcohol1)]	-1	-1

There is only one change in the model's predictions – in an open container, the gas pressure remains constant due to the infinite capacity of the atmosphere. Since closing the container caused the steam's pressure to rise where it had not before, revisions based on the steam's pressure have empirical justification. There is only one:

```
QC(Less-than[Pressure(sk-steam1-1),limit(sk-steam1-1)])
```

Had the model possessed knowledge of alcohol vapor concentration (as it learns from the dissolving analogy), the following revision would be proposed as well:

```
QC(Less-than[Amount-of(sk-steam1-1),Saturation-Point(sk-steam1-1)])
```

7.5 Closed Container Example: Denouement

When the three revision techniques are combined, the following additions to PROCESS-3318 are proposed:

Proposed Addition	*F-P*	*P-G*	*DBR*
QC(Greater-than[Amount-of(alcohol1),limit(alcohol1)])	√		
QC(Less-than[Amount-of(sk-steam1-1),Saturation-Point(sk-steam1-1)])	√	√	
QC(Greater-than[Amount-of(alcohol1),Amount-of(sk-steam1-1)])	√		
QC(Greater-than[Pressure(alcohol1),limit(alcohol1)])	√		
QC(Less-than[Pressure(sk-steam1-1),limit(sk-steam1-1)])	√		√
QC(Greater-than[Temperature(alcohol1),limit(alcohol1)])	√		
QC(Greater-than[Heat(alcohol1),limit(alcohol1)])	√		

Thus, according to the evidence, one of the following proposed additions to PROCESS-3318 should be selected:

```
QC(Less-than[Pressure(sk-steam1-1),limit(sk-steam1-1)])
QC(Less-than[Amount-of(sk-steam1-1),Saturation-Point(sk-steam1-1)])
```

Since the former requires postulating an unknown quantity, the second should be chosen as the best revision to make. Notice that because *saturation* was not a concept present in the model under revision, DBR was unable to predict how it would be affected by a closed container. Thus, DBR neither supports nor discounts the second hypothesis.

7.6 Perspective

This chapter has proposed that relative likelihood measures for ordering potential revision hypotheses arise from experience, both with behavior analogous to a given anomaly and with prior applications of the theory under investigation. At this time, it remains simply a proposal. The first-principles rules have been fully implemented. However, the installation of precedent-guided revision and DBR has only recently begun.

The analogy approach described above is designed to provide focus to an otherwise unguided first-principles technique. An alternative approach to this problem is *experimentation-based theory revision* (Rajamoney et al., 1985; Rajamoney, 1988a), which prunes inconsistent revision hypotheses through directed experimentation.[5] By splitting the set of possible revisions through discrimination experiments, Rajamoney's ADEPT system is able to isolate the appropriate change to a flawed theory. However, it has the potential to sanction many experiments, often on hypotheses that look silly to a human observer, because it lacks experiential knowledge indicating what is likely. This was the motivation behind combining experimentation-based theory revision with analogical hypothesis generation (Falkenhainer & Rajamoney, 1988). These two approaches to theory development are complementary. ADEPT was provided PHINEAS' analogical mechanism to focus the revision process. At the same time, PHINEAS was given the ability to interact and ask questions of the world through ADEPT.

For example, the union is sometimes crucial for analyzing the coverage of a model. An analogy will often predict additional, unobserved behavior. Did this additional behavior actually happen, or does the prediction represent a flaw in the model? Some of these predictions may be refuted logically, by relation to what is already known. Others, however, must be empirically tested by repeating the scenario if possible and specifically looking for the predicted properties.

In one of the implemented examples, the loss of alcohol sitting in an open container is explained as being analogous to the vaporization portion of boiling. When the new evaporation model is used to anticipated what should happen in the given situation, a new

[5]An indepth study of the revision problem may be found in (Rajamoney, 1988b). There he presents two techniques not used in PHINEAS, active experimentation and exemplar-based theory rejection.

secondary prediction is produced – the alcohol's temperature must have dropped, due to the loss of latent heat during vaporization. Since the alcohol's temperature behavior was not originally reported, ADEPT called for the physical scenario to be repeated and changes in alcohol temperature noted. The test confirmed the evaporation theory's hypothesis.

Chapter 8

The PHINEAS System

PHINEAS is an explanation system which uses analogy as the primary source of hypothesis generation, rather than one of the more conventional abductive, unification-based methods (e.g., Charniak, 1972; DeJong, 1982; Forbus, 1986a; Josephson et al., 1987; Mooney, 1987; Pople, 1973; Reggia, 1983; Simmons, 1988). In preceding chapters, the individual stages embodied in PHINEAS were presented sequentially, showing how a complete and consistent explanation of a given observation is developed. This chapter discusses the process from a global perspective. It begins by reviewing PHINEAS in terms of the programs comprising it and describes how they interact. The chapter then discusses the criteria used to prefer one hypothesis over another. This is a primary determinant of how PHINEAS' flow of control moves from one stage to another and from one working hypothesis to another. Finally, the disappearing alcohol example is presented in its entirety to show how the dissolving analogue that has been described in preceding chapters is but one of several considered.

8.1 Program components

The VBAL approach to analogical learning and reasoning is knowledge-intensive. It requires the ability to make analogical comparisons, perform deductive, abductive, and qualitative reasoning, and analyze the completeness of a theory with respect to the observations it should explain. In support of these tasks, PHINEAS uses three auxiliary modules:

SME: The *structure-mapping engine* functions as the system's mapping module, identifying similarity and proposing candidate inferences to posit explanations.

QPE: Forbus' (1986b) *qualitative process engine* is used to envision the scenario predictions of a hypothesized model.[1]

DATMI: Decoste's (1989) *dynamic across-time measurement interpretation* program serves to relate observations of physical behavior to the predictions of a qualitative theory.

One of the goals in the construction of PHINEAS was to apply the model of analogy developed in this thesis to a nontrivial reasoning task. To that end, PHINEAS embodies representation and analysis techniques from the state of the art in qualitative physics in order to maximize scale and generality. It is a large program, consisting of over 6,000 lines of CommonLisp code and 435 functions.[2] When combined with the other program modules making up the complete system (i.e., SME, QPE, and DATMI), this rises to 39,260 lines of lisp code and 3,000 functions.

A block diagram of PHINEAS showing the five primary stages of operation and their interaction with the various program modules appears in Figure 8.1. These five stages correspond to the preceding four chapters of this thesis:

1. **Behavior match.** A new observation triggers a search for previously understood experiences that exhibited analogous behavior. Abstractions of the observed situation and its behavior are used to focus attention on a potentially relevant subset of memory. Each experience in this subset is then compared at a detailed level to the current situation, using SME as the comparative mechanism.

2. **Theory generation.** The central objective of the second stage is to produce a fully operational initial hypothesis about the current domain. This has two components.

   (a) *Mapping.* First, the models used to explain analogous aspects of the recalled experience are retrieved and SME is used to analogically map these into the current domain. This mapping is guided by the initial correspondences found in the behavioral comparison.

   (b) *Transfer.* Second, to operationalize the model, the consistency of its expressions must be ensured and any unknown skolem objects it requires must be inferred from the domain theory or their existence postulated. A *map and analyze* cycle may ensue.

[1]QPE, like PHINEAS, is actually a system of programs consisting of the QPE code itself, deKleer's ATMS, and Forbus & deKleer's rule-based problem solver for the ATMS, called ATMoSphere.

[2]The number of lines represents just lines of code. Blank lines and comment lines are not included.

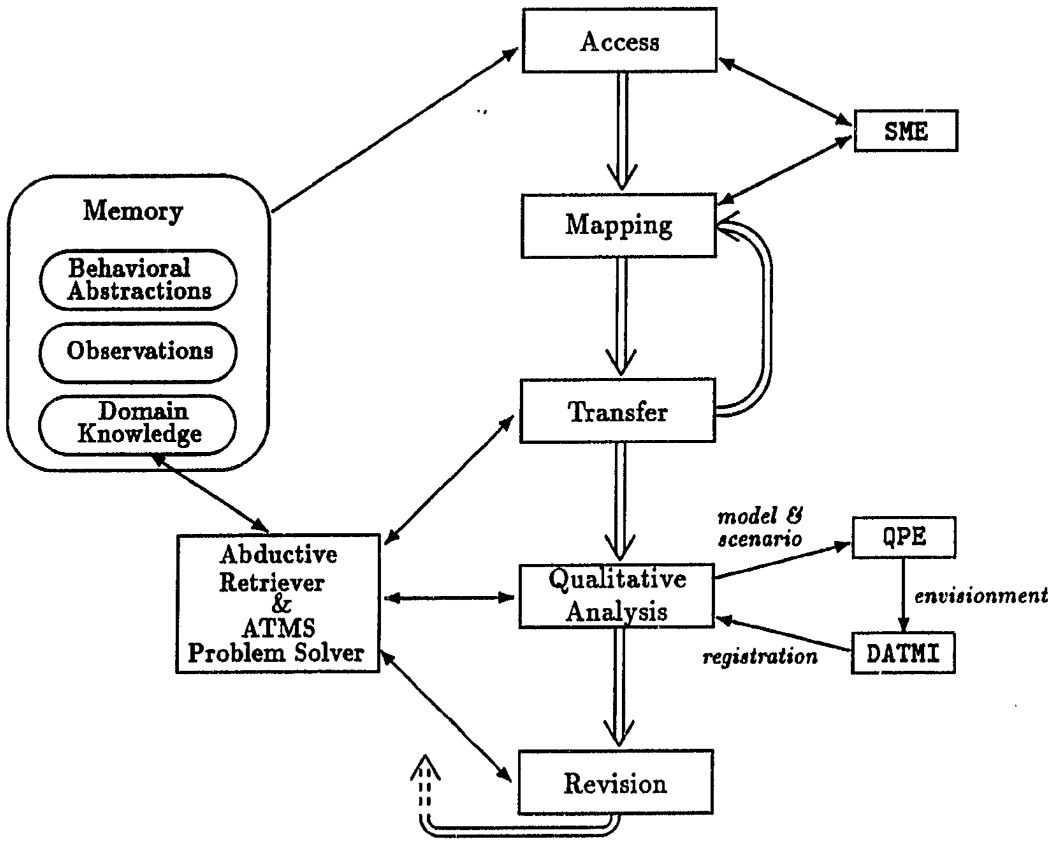


Figure 8.1: Block diagram of the Phineas system modules.

3. **Gedanken analysis.** The operational model is used to construct an explanation of the present observation. The model is given to QPE, which generates an envisionment for the model applied to the observed physical configuration. DATMI then compares these predictions to the observation and either determines that the model is adequate or identifies points of discrepancy.

4. **Revision.** If an initial hypothesis fails, or an old hypothesis is inadequate for a new situation, an attempt should be made to adapt it around points of inaccuracy. A model of revision is advocated which relies on past experiences to guide the formation and selection of revision hypotheses. It considers behavior analogous to the current anomaly and considers how the current anomalous situation differs from prior

situations that were consistently explained. This is the only component that is not fully implemented.

While described as a sequential process concerned with the development of a single explanatory hypothesis, the program's focus may change from one hypothesis to another. At some point, the cost of additional work on a poor hypothesis is outweighed by the potential of other hypotheses needing further development. A prerequisite to altering focus is the ability to evaluate a working hypothesis and prefer one candidate over another. These issues are the topics of the next two sections.

8.2 Preference Criteria

PHINEAS is primarily concerned with the *interpretation-construction task* of explanation: find candidate explanations and the assumptions on which they rest. However, a system that exhaustively generated an unordered set of possible hypotheses would not be of much use. It should focus on the most promising explanations first and provide a preferential ordering on fully developed hypotheses.

Two general types of preference criteria are used in PHINEAS. During the early stages of hypothesis development, the only preferential guidance available is the degree of similarity between a candidate analogue and the current situation. This is represented by SME's evaluation score for the behavioral and structural match between the two situations computed during access. When determining which of two candidate analogues to consider next, the one with the higher similarity score is chosen. This metric supports the *similarity conjecture* stated in Chapter 1, which proposes that interpretation-construction tasks may be characterized as the search for maximal, explanatory similarity between the situation being explained and some explainable scenario.

Once an actual hypothesis has been formed (i.e., the result of transfer), the preference criterion must change to consider the characteristics of the hypothesis itself. A complete account of theory selection requires consideration of many complex factors, such as a theory's plausibility, coherence, effect on prior beliefs, simplicity, and specificity in accounting for the phenomenon. Unfortunately, these are significant open research problems in their own right, and certainly beyond the scope of this thesis. However, a number of important, more specific preference criteria are readily available and have been found useful in PHINEAS for establishing preference between competing hypotheses. These are:

C_{CE} *Conjectured entities.* Does the hypothesis conjecture the existence of a novel kind of entity, and if so, how many?

C_{VE} *Vocabulary extensions.* Does the hypothesis require the creation of new predicates, and if so, how many?

C_{CA} *Composite assumptions.* Does the hypothesis conjecture the existence of new physical processes or new knowledge structures (e.g., schemas, etc.), and if so, how many?

C_{AE} *Assumed entities.* Does the hypothesis assume the presence of a known type of entity not mentioned in the original scenario description, and if so, how many?

C_{AA} *Atomic assumptions.* Does the hypothesis make additional assumptions about the properties and interrelationships of objects in the scenario, and if so, how many?

The single preference criterion used to evaluate a hypothesis or compare two competing hypotheses is a function of these five. The method for combining them is adapted from Michalski (1983), who describes the use of a *lexicographic evaluation functional* (LEF) for evaluating alternate inductive concept descriptions. A LEF is a list of elementary criterion–tolerance pairs, in which each elementary criterion is applied sequentially to prune the space of hypotheses. In PHINEAS, the elementary preference criteria are ordered according to an approximate measure of decreasing "cost":

$$\text{LEF} = (C_{CE},\ C_{VE},\ C_{CA},\ C_{AE},\ C_{AA})$$

Thus, an explanation which postulates the existence of a novel kind of entity (C_{CE}) is at all times deemed inferior to one which does not. Each criterion returns a number ($N \geq 0$) as described above, where a value of zero indicates success and a value greater than zero indicates failure. The LEF is used to select the most preferable explanation(s) from a given set as follows: First, each proposed explanation is evaluated by criterion C_{CE} and those that pass C_{CE} are retained. The process is repeated with the next criterion on the set of retained hypotheses until only a single hypothesis remains or the list of criteria is exhausted. If at any point all hypotheses evaluated by a particular criterion fail, the process stops and the current set is returned in increasing order according to their score N for that criterion.

This evaluative function produces an interesting property when viewed from the perspective of the four explanation scenarios described in Chapter 1:

1. *Deductive scenario.* Given phenomenon $\mathcal{P}$, where $\mathcal{P}$ represents a set of observables, a complete explanation of $\mathcal{P}$ deductively follows from existing knowledge. This corresponds to explanations passing every criterion.

2. *Assumption scenario.* No explanation can be grounded with current knowledge because not all of the relevant facts are known. However, a complete explanation follows from the union of existing knowledge and a consistent set of assumptions about the missing facts. This corresponds to explanations passing every criterion but one of the last two, C_{AE} and C_{AA}.

3. *Generalization scenario.* Existing knowledge indicates that candidate explanation $\mathcal{E}$ cannot apply because condition C_1 is known to be false in the current situation. However, $\mathcal{E}$ does follow if condition C_1 is replaced by the next most general relation, since C_1's sibling is true in the current situation. This corresponds to explanations passing the first two criteria, C_{CE} and C_{VE}, but failing C_{CA}, in which a knowledge structure is viewed as "new" if it represents a modification of an existing knowledge structure.[3]

4. *Analogy scenario.* No candidate explanation $\mathcal{E}$ is available directly, but explanation $\mathcal{E}_b$ is available if a series of analogical assumptions are made, that is, if the situation explained by $\mathcal{E}_b$ is assumed analogous to the current situation. This corresponds to explanations failing one of the first three criteria, C_{CE}, C_{VE}, or C_{CA}.

The evaluative function causes PHINEAS to propose standard, deductive explanations if found. In their absence, conventional abductive explanations will be preferred. If existing theories are insufficient to provide an explanation, explanations adapting knowledge of potentially analogous phenomena will be offered. By using analogy as the single source for explanation generation, PHINEAS is able to offer a "best guess" in the presence of an imperfect or incomplete domain theory.

8.3 Flow of Control

In addition to theory selection, preference criteria are important for guiding PHINEAS toward developing the most promising hypotheses first. PHINEAS' global operation is controlled by a task agenda, which maintains an ordered sequence of task–hypothesis pairs. Multiple hypotheses in various stages of development may exist at any one time. The task agenda ordering determines which hypothesis to expend effort on next, enabling the program's focus to change from one hypothesis to another. Repeatedly, the task–hypothesis pair at

[3]The issue of whether to actually create a new knowledge structure or modify the existing one is an important but orthogonal issue. Here we are concerned with hypothesis evaluation rather than storage of an accepted hypothesis.

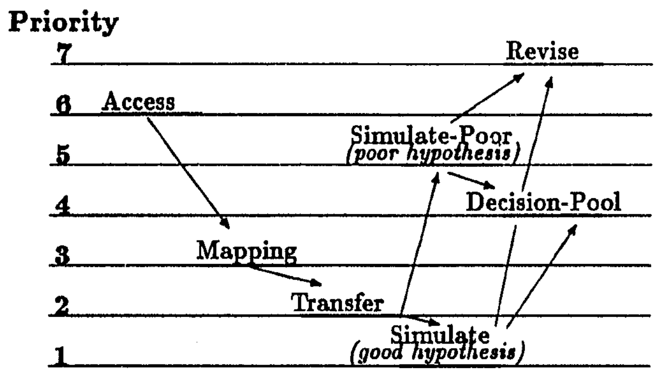


Figure 8.2: PHINEAS task scheduling.

the front of the agenda is selected and executed, resulting in further development of its corresponding hypothesis. This task may in turn spawn other tasks, modify tasks waiting for execution, or signal the acceptance of a hypothesis, which halts the cycle.

Each task is given a priority level giving rise to the priority lattice shown in Figure 8.2. Lower priority numbers indicate increasing precedence. In addition to their normal priority levels, the mapping and transfer tasks have an auxiliary score for sorting tasks within the same priority level. This auxiliary score is SME's evaluation score for the behavioral and structural match between the current observation and the task's associated analogue.

There are eight task types currently used in PHINEAS: access, mapping, form-unique-mappings, transfer, simulate, simulate-poor, revise, and decision-pool. The function of most of these should be evident from their name. For example, the transfer task may be paraphrased as:

```
Task TRANSFER:   Given candidate inference $\mathcal{CI}$,
                     For each theory $\mathcal{T} \in$ Transfer($\mathcal{CI}$)
                         If Poor-hypothesis?($\mathcal{T}$)
                             then Schedule(Simulate-Poor($\mathcal{T}$), 5)
                             else Schedule(Simulate($\mathcal{T}$), 1)
```

The transfer task produces a set of operational theories from a given candidate inference and then schedules each of those theories for simulation. If transfer produces a *poor hypothesis*, the subsequent simulate-poor task is given a priority of 5. Otherwise, the simulate task is given a priority of 1. A poor hypothesis is defined as one which contains conjectured entities (C_{CE}) or requires the creation of new predicates (C_{VE}). The two simulate tasks are identical except that simulate-poor additionally reschedules all other simulate-poor tasks waiting on the agenda to be normal simulate tasks. Due to the priority arrangement, the execution of a simulate-poor task indicates that there are no "good" hypotheses available. In such a situation, all "poor" hypotheses are reclassified as "good".

form-unique-mappings is invoked when a behavioral comparison is ambiguous, producing multiple correspondence sets. At times, multiple behavioral matches support the same candidate inference. Hence, mapping is applied to each and their resulting candidate inferences compared, with duplications removed.

decision-pool is a decision-making task applied to hypotheses found to be adequate during the simulation phase. Its priority level is set such that when a decision-pool task reaches the front of the agenda, all hypotheses to be considered have completed at least the transfer phase of development. The decision-pool task collects all adequate explanations existing on the task agenda and applies the LEF described above to this set. The best explanation(s) according to the LEF are returned as PHINEAS' proposed explanation(s) of the observed phenomenon and the cycle halts.

This priority setting for decision-pool is chosen primarily to enable observation of PHINEAS' operation across the entire space of possibilities. It ensures that all of the initial analogues proposed by access complete at least the transfer phase. This results in the extra work of developing all these hypotheses, but enables application of the LEF to a well-developed candidate set. If a more best-first approach is desired, then the priority of decision-pool may be set to 0, causing immediate acceptance of the first adequate hypothesis developed.

8.4 Disappearing Alcohol Observation: Reprise

The disappearing alcohol example described at various points in preceding chapters will now be presented in its entirety, showing the different hypotheses PHINEAS considers, how it moves from one hypothesis to another, and how it makes its final selection(s).

The explanation process begins with a statement of the observation, which is given in its entirety in Figure 8.3. At this point in the thesis, the reader has probably assumed that the observation corresponds to evaporation. There are two important items to consider about the observation and PHINEAS' approach to it. First, evaporation is only one of the possible explanations consistent with the given information. Second, PHINEAS does not possess knowledge about evaporation, so it must examine the situation from that perspective. At this point, PHINEAS' task queue contains:[4]

[4]Here I am showing the task forms exactly as they appear when printed by PHINEAS. The first number is

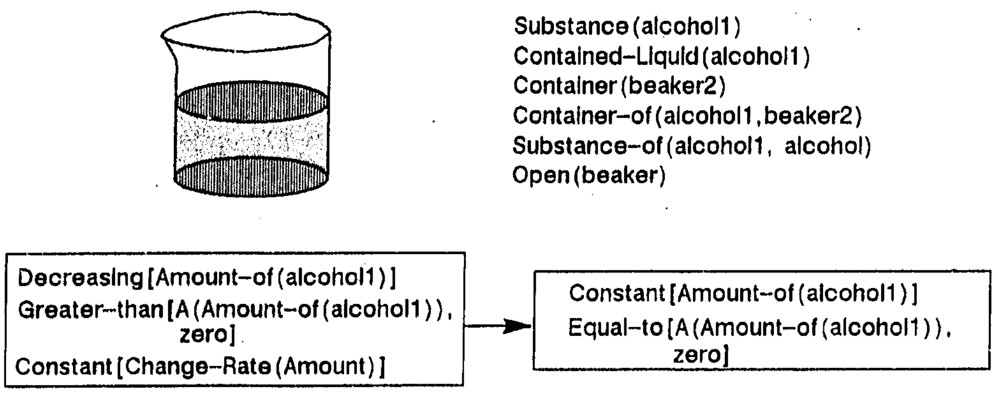

Substance(alcohol1)
Contained-Liquid(alcohol1)
Container(beaker2)
Container-of(alcohol1,beaker2)
Substance-of(alcohol1, alcohol)
Open(beaker)
Decreasing[Amount-of(alcohol1)]
Greater-than[A(Amount-of(alcohol1)), zero]
Constant[Change-Rate(Amount)]
Constant[Amount-of(alcohol1)]
Equal-to[A(Amount-of(alcohol1)), zero]

Figure 8.3: The "disappearing alcohol" scenario description and its corresponding observed behavior. The amount of alcohol in an open beaker is decreasing at a constant rate. This activity stops when the amount of alcohol equals zero.

```
(7  0.0 (ACCESS OBS#OPEN-ALCOHOL))
```

The access task operates in two steps (Chapter 4). First, the behavioral abstractions of the observation are used to activate prior experiences. Second, SME is used to provide a closer examination of the N (15) most activated experiences, resulting in their rank ordering and an initial set of correspondences between the current observation and each retrieved scenario. The four possibilities discovered and their SME scores are shown in Figure 8.4. At this point, PHINEAS' task queue contains:

```
(4 15.73 (MAPPING MAP(BSeg#BOILING-BEHAVIOR,BSeg#OPEN-ALCOHOL-BEHAVIOR)))
(4 14.90 (MAPPING MAP(BSeg#LIQUID-DRAINING-BEHAVIOR,BSeg#OPEN-ALCOHOL-BEHAVIOR)))
(4 12.62 (MAPPING MAP(BSeg#DISSOLVE-BEHAVIOR,BSeg#OPEN-ALCOHOL-BEHAVIOR)))
(4 11.06 (MAPPING MAP(BSeg#2-CONTAINER-LF,BSeg#OPEN-ALCOHOL-BEHAVIOR)))
```

In executing the first mapping task, PHINEAS is given the behavioral mapping comparing the current open-alcohol-behavior observation to the recalled boiling-behavior experience. This experience describes a pan of water that is boiling on a stove. The first step in mapping it to the current situation is retrieving the domain theory used to explain

the priority level (one represents the highest possible priority), the second number is the auxiliary auxiliary score for sorting tasks within the same priority level. OBS#OPEN-ALCOHOL indicates the presence of a LISP structure holding the data for the observation "open-alcohol".

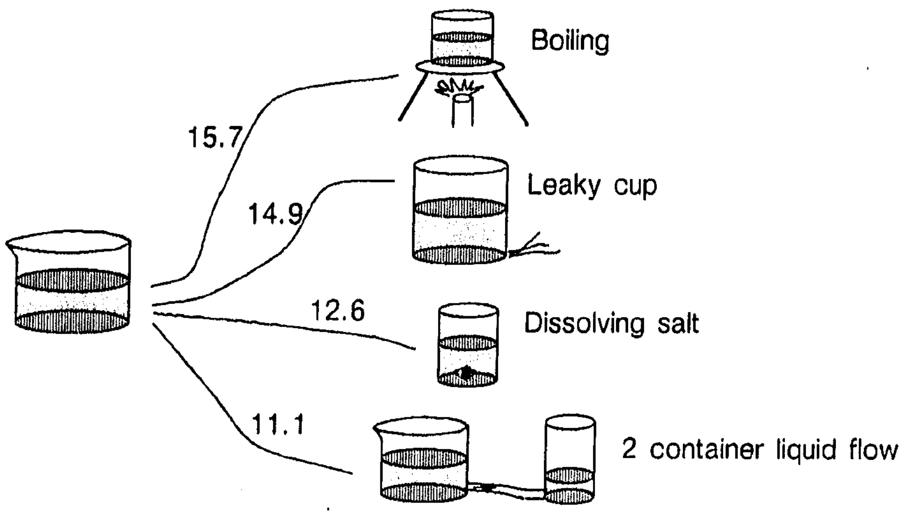


Figure 8.4: Candidate analogues for the "disappearing alcohol" scenario returned by the access task.

analogous aspects of its behavior. In this case, there were three processes used in the prior boiling explanation: *boiling*, *heat flow*, and *heat replenish*. The boiling process specifies the vaporization of a liquid when at or above its boiling temperature at a rate qualitatively equal to the rate of heat flow into the liquid. The heat replenish process is a common technique for modelling the behavior of an ideal heat source in QP theory by resupplying it with heat as fast as it is depleted. Where this added heat is coming from is not considered.

The second step in mapping these processes to the current situation is to declare the correspondences sanctioned by access. These are (pan7 ↔ beaker2), (bwater1 ↔ alcohol1), (water ↔ alcohol), (amount-of ↔ amount-of), and (change-rate ↔ change-rate).

The third mapping step invokes SME, which returns the gmap shown in Figure 8.5. The initial explanation for the disappearing alcohol, via analogy to boiling water, appears in the candidate inferences field. It proposes that something like boiling is removing alcohol1 from beaker2. However, this mapping and its candidate inference must be analyzed. At this point, PHINEAS' task queue contains:

```
(3 15.73 (TRANSFER MAP(BSeg#BOILING-BEHAVIOR,BSeg#OPEN-ALCOHOL-BEHAVIOR)))
(4 14.90 (MAPPING MAP(BSeg#LIQUID-DRAINING-BEHAVIOR,BSeg#OPEN-ALCOHOL-BEHAVIOR)))
(4 12.62 (MAPPING MAP(BSeg#DISSOLVE-BEHAVIOR,BSeg#OPEN-ALCOHOL-BEHAVIOR)))
(4 11.06 (MAPPING MAP(BSeg#2-CONTAINER-LF,BSeg#OPEN-ALCOHOL-BEHAVIOR)))
```

```
             SME Version 2E
    Analogical Match from BOILING-BEHAVIOR-THEORY to OPEN-ALCOHOL-BEHAVIOR-THEORY.

Rule File: sam.rules
-----------------------------------------------------------
# MH's | # Gmaps |       1st,2nd,Worst     | RelGroups |
   12  |     1   |  4.41 /  4.41 /  4.41 |  ACTIVE   |
-----------------------------------------------------------
Total Run Time:     0 Minutes,  2.264 Seconds
BMS Run Time:       0 Minutes,  1.237 Seconds
Best Gmaps: { 1 }

Gmap #1:  (AMOUNT-OF-251 AMOUNT-OF-293)  (SUBSTANCE-OF-234 SUBSTANCE-OF-299)  (PAN7 BEAKER2)
          (CONTAINED-LIQUID-232 CONTAINED-LIQUID-296)  (CONTAINER-OF-233 CONTAINER-OF-298)
          (WATER ALCOHOL)  (SUBSTANCE-226 SUBSTANCE-295)  (BWATER1 ALCOHOL1)
  Weight: 4.4056
  Candidate Inferences:
      (B-EXPLAINS
          (SET (PROCESS-DEFINITION (:SKOLEM HEAT-FLOW) (:SKOLEM PI1)
                  (IMPLIES
                       (AND (INDIVIDUAL (:SKOLEM STOVE9) (CONDITIONS (THERMAL-OBJECT (:SKOLEM STOVE9))))
                            o
                            o
               (PROCESS-DEFINITION (:SKOLEM BOILING) (:SKOLEM PI2)
                  (IMPLIES
                       (AND (INDIVIDUAL ALCOHOL (CONDITIONS SUBSTANCE-295))
                            (INDIVIDUAL BEAKER2 (CONDITIONS (CAN-CONTAIN BEAKER2 ALCOHOL)))
                            (INDIVIDUAL ALCOHOL1
                               (CONDITIONS CONTAINED-LIQUID-296 CONTAINER-OF-298 SUBSTANCE-OF-299))
                            (INDIVIDUAL (:SKOLEM BSTEAM1)
                               (CONDITIONS (CONTAINED-GAS (:SKOLEM BSTEAM1))
                                           (CONTAINER-OF (:SKOLEM BSTEAM1) BEAKER2)
                                           (SUBSTANCE-OF (:SKOLEM BSTEAM1) ALCOHOL)))
                            (INDIVIDUAL (:SKOLEM PI1)
                               (CONDITIONS (PROCESS-INSTANCE-OF (:SKOLEM HEAT-FLOW) (:SKOLEM PI1))
                                           (PI1 DESTINATION ALCOHOL1)))
                            (ACTIVE (:SKOLEM PI1))
                            (NOT (GREATER-THAN (A (TBOIL ALCOHOL1)) (A (TEMPERATURE ALCOHOL1))))
                            (GREATER-THAN (A AMOUNT-OF-293) ZERO))
                       (AND (QUANTITY (VAPORIZATION-RATE (:SKOLEM PI2)))
                            (Q= (VAPORIZATION-RATE (:SKOLEM PI2)) (HEAT-FLOW-RATE (:SKOLEM PI1)))
                            (GREATER-THAN (A (VAPORIZATION-RATE (:SKOLEM PI2))) ZERO)
                            (I- (HEAT ALCOHOL1) (A (VAPORIZATION-RATE (:SKOLEM PI2))))
                            (CTRANS AMOUNT-OF-293 (AMOUNT-OF (:SKOLEM BSTEAM1))
                                  (A (VAPORIZATION-RATE (:SKOLEM PI2)))))))
               (PROCESS-DEFINITION (:SKOLEM HEAT-REPLENISH) (:SKOLEM PI3)
                  (IMPLIES
                       (AND (INDIVIDUAL (:SKOLEM STOVE9) (CONDITIONS (HEAT-SOURCE (:SKOLEM STOVE9))))
                            (INDIVIDUAL (:SKOLEM PI1)
                               (CONDITIONS (PROCESS-INSTANCE-OF (:SKOLEM HEAT-FLOW) (:SKOLEM PI1))
                                           (PI1 SOURCE (:SKOLEM STOVE9))))
                            (ACTIVE (:SKOLEM PI1)))
                       (AND (EQUAL-TO (D (HEAT (:SKOLEM STOVE9))) ZERO)
                            (I+ (HEAT (:SKOLEM STOVE9)) (A (HEAT-FLOW-RATE (:SKOLEM PI1)))))))
          OPEN-ALCOHOL-BEHAVIOR)
```

Figure 8.5: SME$_{CSM}$ mapping from the boiling processes (boiling, heat flow and heat replenish) to the disappearing alcohol scenario.

When transfer is attempted on the boiling hypothesis, all of the expressions are consistent, but two skolem objects are found: `(:skolem stove9)`, the ideal heat source, and `(:skolem bsteam1)`, the vapor produced from boiling. When the abductive retriever is used to seek possible analogues for these objects in the current situation, no candidates are found. Therefore, PHINEAS assumes a new entity token for each unknown object and confirms that the candidate inference remains consistent upon their installation. The transfer operation is completed by the explicit assumption of the now operational hypothesis. PHINEAS first determines that the three proposed processes are identical to the original boiling, heat flow and heat replenish processes (they were never changed through mapping and transfer) and removes them from the hypothesis. It then makes the following set of atomic assumptions:

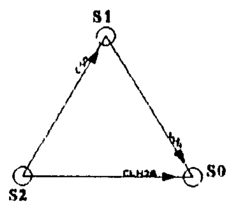
```
(THERMAL-OBJECT SK-STOVE9-1)
(HEAT-SOURCE SK-STOVE9-1)
(CONTAINED-GAS SK-BSTEAM1-1)
(CONTAINER-OF SK-BSTEAM1-1 BEAKER2)
(SUBSTANCE-OF SK-BSTEAM1-1 ALCOHOL)
(HEAT-CONNECTION BEAKER2 SK-STOVE9-1 ALCOHOL1)
```

The transfer task then schedules a simulate task for the hypothesis. Since the hypothesis does not contain conjectured entities (C_{CE}) or create new predicates (C_{VE}), `simulate` is scheduled rather than `simulate-poor`. At this point, PHINEAS' task queue contains:[5]

```
(2 15.73 (SIMULATE TH#OPEN-ALCOHOL-BEHAVIOR-THEORY-1-1))
(4 14.90 (MAPPING MAP(BSeg#LIQUID-DRAINING-BEHAVIOR,BSeg#OPEN-ALCOHOL-BEHAVIOR)))
(4 12.62 (MAPPING MAP(BSeg#DISSOLVE-BEHAVIOR,BSeg#OPEN-ALCOHOL-BEHAVIOR)))
(4 11.06 (MAPPING MAP(BSeg#2-CONTAINER-LF,BSeg#OPEN-ALCOHOL-BEHAVIOR)))
```

The simulate task invokes QPE on the given hypothesis to envision its predictions for the current scenario. This is shown in Figure 8.6. State S2 describes the alcohol below the boiling point and heating up. State S1 describes the alcohol boiling, with its amount decreasing at a constant rate and the amount of hypothesized alcohol steam increasing. Both S2 and S1 are able to transition to state S0. QPE individuates states in an envisionment by the derivatives of quantities and the active processes. For this scenario, there are two distinct situations having no active processes and derivatives equal to zero. In situation S0-1, the alcohol has completely boiled away, leaving nothing more to boil. This is the conclusion observed in the disappearing alcohol scenario. Alternatively, S0-2 describes the alcohol and stove temperatures equalizing, again leading to a halt in active behavior. When

[5]The current working theory, TH#OPEN-ALCOHOL-BEHAVIOR-THEORY-1-1, is the result of transferring TH#OPEN-ALCOHOL-BEHAVIOR-THEORY-1, the original candidate inference derived from mapping.



Quantity	S2	S1	S0-1	S0-2
Ds[HEAT-FLOW-RATE(PI0)]	-1	–	0	0
Ds[AMOUNT-OF(ALCOHOL1)]	0	-1	0	0
Ds[AMOUNT-OF(SK-BSTEAM1-1)]	0	1	0	0
Ds[CHANGE-RATE(AMOUNT-OF(ALCOHOL1))]	0	0	0	0
Ds[TEMPERATURE(ALCOHOL1)]	1	0	0	0
A[TEMPERATURE(ALCOHOL1)] A[TEMPERATURE(SK-STOVE9-1)]	<	<	?	=
A[AMOUNT-OF(ALCOHOL1)]	>0	>0	=0	>0
A[CHANGE-RATE(AMOUNT-OF(ALCOHOL1))]	=0	<0	=0	=0
ACTIVE(PI0)	T	T	F	F
ACTIVE(PI1)	T	T	F	F
ACTIVE(PI2)	F	T	F	F

Processes:

```
PI0: HEAT-FLOW(SK-STOVE9-1,ALCOHOL1,BEAKER2)
PI1: HEAT-REPLENISH(SK-STOVE9-1,PI0)
PI2: BOILING(ALCOHOL,BEAKER2,ALCOHOL1,SK-BSTEAM1-1,PI0)
```

Figure 8.6: Envisionment of the boiling hypothesis for the disappearing alcohol observation. Since QPE distinguishes states by the derivatives of quantities and active processes, state S0 is actually two difference situations (S0-1 and S0-2), but both have the same derivative and process characteristics. Ds[...] denotes the sign of the derivative. "–" indicates a quantity is undefined during that state (e.g., its process doesn't exist) and "?" indicates that any value is possible.

DATMI is used to determine adequacy, it finds that the envisionment is both complete and consistent with respect to the observed behavior. At this point, PHINEAS has constructed an adequate explanation of the situation. However, due to the task settings in use, it continues to examine other candidates. Its task queue now contains:

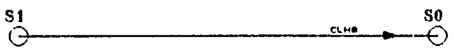
```
(4 14.90 (MAPPING MAP(BSeg#LIQUID-DRAINING-BEHAVIOR,BSeg#OPEN-ALCOHOL-BEHAVIOR)))
(4 12.62 (MAPPING MAP(BSeg#DISSOLVE-BEHAVIOR,BSeg#OPEN-ALCOHOL-BEHAVIOR)))
(4 11.06 (MAPPING MAP(BSeg#2-CONTAINER-LF,BSeg#OPEN-ALCOHOL-BEHAVIOR)))
(5  0.00 (DECISION-POOL TH#OPEN-ALCOHOL-BEHAVIOR-THEORY-1-1))
```

The second candidate analogue proposed during access is a leaky cup: perhaps the alcohol is leaking through a hole in the beaker. When the leaky cup's explanation is mapped, two processes are proposed, corresponding to liquid flow and liquid drain (an ideal sink – the opposite of heat replenish described above). The transfer task applied to this hypothesis finds that the individual expressions are consistent, but the candidate inference contains three unknown objects: (:skolem sink5), the destination of flow, (:skolem cs-sink5), the destination liquid, and (:skolem hole7), the fluid path leading out of the beaker. In each case, no analogue can be found and a new entity token is assumed. Once again the proposed processes are found to be identical to their original base analogues and the final operational hypothesis consists of the following set of assumptions:

```
(LIQUID-SINK SK-SINK5-1)
(CAN-CONTAIN SK-SINK5-1 ALCOHOL)
(CONTAINED-LIQUID SK-CS-SINK5-1)
(CONTAINED-FLUID SK-CS-SINK5-1 ALCOHOL SK-SINK5-1)
(CONTAINER-OF SK-CS-SINK5-1 SK-SINK5-1)
(PHYSICAL-PATH BEAKER2 SK-SINK5-1 SK-HOLE7-1)
(FLUID-PATH SK-HOLE7-1)
(FLUID-ALIGNED SK-HOLE7-1)
```

PHINEAS now executes the simulate task on the leak hypothesis. Its envisionment is shown in Figure 8.7. State S1 indicates that alcohol is flowing from the beaker to the ideal sink, sk-sink5-1. The alcohol's amount is decreasing, the pressure in the beaker is decreasing (taken at the beaker's bottom), and the flow rate is decreasing. State S0 indicates that the amount-of alcohol is equal to zero, all quantities are constant, and there are no processes active.

When the hypothesis is compared to the original observation, an anomaly is found. The alcohol's rate of change was constant in the observation, yet the proposed model predicts that the change rate increases (becomes less negative) as the amount of alcohol decreases. Consequently, PHINEAS schedules the leak hypothesis for revision. The task queue now contains:



Quantity	S1	S0
Ds[FLOW-RATE(PI0)]	-1	–
Ds[AMOUNT-OF(ALCOHOL1)]	-1	0
Ds[AMOUNT-OF(SK-CS-SINK5-1)]	0	0
Ds[CHANGE-RATE(AMOUNT-OF(ALCOHOL1))]	1	0
Ds[PRESSURE-IN(BEAKER2)]	-1	0
Ds[PRESSURE-IN(SK-SINK5-1)]	0	0
Ds[TEMPERATURE(ALCOHOL1)]	0	0
Ds[TEMPERATURE(SK-CS-SINK5-1)]	0	0
A[AMOUNT-OF(ALCOHOL1)]	>0	=0
A[CHANGE-RATE(AMOUNT-OF(ALCOHOL1))]	<0	=0
A[PRESSURE-IN(BEAKER2)] A[PRESSURE-IN(SK-SINK5-1)]	>	?
ACTIVE(PI0)	T	F
ACTIVE(PI1)	T	F

Processes:

```
PI0: LIQUID-FLOW(ALCOHOL,BEAKER2,ALCOHOL1,SK-SINK5-1,SK-CS-SINK5-1,SK-HOLE7-1)
   PI1: LIQUID-DRAIN(SK-SINK5-1,SK-CS-SINK5-1,PI0)
```

Figure 8.7: QPE envisionment of the "leaky container" hypothesis for the disappearing alcohol observation.

```
(4 12.62 (MAPPING MAP(BSeg#DISSOLVE-BEHAVIOR,BSeg#OPEN-ALCOHOL-BEHAVIOR)))
(4 11.06 (MAPPING MAP(BSeg#2-CONTAINER-LF,BSeg#OPEN-ALCOHOL-BEHAVIOR)))
(5  0.00 (DECISION-POOL TH#OPEN-ALCOHOL-BEHAVIOR-THEORY-1-1))
(8 14.90 (REVISE TH#OPEN-ALCOHOL-BEHAVIOR-THEORY-2-1
          ((NO-INTERPRET ALCOHOL-GOING) (OCCURSAT QSTATE-1 ALCOHOL-DRY))
          (ALCOHOL-GOING ALCOHOL-DRY)))
```

The next hypothesis considered is the dissolving analogue (salt dissolving in water), which has been discussed throughout the preceding chapters. The mapping task produces a candidate inference that is flawed by inconsistent predicate use (e.g., (Immersed-in alcohol1 *?water1*)) and the presence of an unknown correspondent for the water. During the **transfer** task, the atmosphere is found as a potential analogue for the water and mapping is repeated. In the second transfer pass, all object correspondences are known. All that remains is to resolve three inconsistent expressions:

```
(Solution atmosphere)
(Soluble alcohol1)
(Soluble-in alcohol1 atmosphere)
```

which become

```
(Solution-8 atmosphere)
(SK-Soluble-4-1 alcohol1)
(SK-Soluble-in-4-1 alcohol1 atmosphere)
```

Having produced an operational hypothesis from the dissolving candidate inference, the **transfer** task schedules it for simulation. Due to the new predicates required (C_{VE}), it is scheduled foı the simulate-poor task. The task queue now contains:

```
(4 11.06 (MAPPING MAP(BSeg#2-CONTAINER-LF,BSeg#OPEN-ALCOHOL-BEHAVIOR)))
(5  0.00 (DECISION-POOL TH#OPEN-ALCOHOL-BEHAVIOR-THEORY-1-1))
(6 12.62 (SIMULATE-POOR TH#OPEN-ALCOHOL-BEHAVIOR-THEORY-4))
(8 14.90 (REVISE TH#OPEN-ALCOHOL-BEHAVIOR-THEORY-2-1
          ((NO-INTERPRET ALCOHOL-GOING) (OCCURSAT QSTATE-1 ALCOHOL-DRY))
          (ALCOHOL-GOING ALCOHOL-DRY)))
```

The final analogue considered is a standard liquid flow behavior between two containers connected by a pipe. It results in theory OPEN-ALCOHOL-BEHAVIOR-THEORY-5-1, which conjectures that the alcohol is flowing to another container through a fluid path connecting them. A description of PHINEAS invoking mapping, **transfer**, and **simulate** on this analogue would be redundant with the prior description of the leaky cup hypothesis (including its rejection due to a non-constant change rate). Hence, it will not be repeated here.

At this point, each candidate analogue has completed at least the transfer phase of operation. The task queue now contains:

```
(5  0.00 (DECISION-POOL TH#OPEN-ALCOHOL-BEHAVIOR-THEORY-1-1))
(6 12.62 (SIMULATE-POOR TH#OPEN-ALCOHOL-BEHAVIOR-THEORY-4))
(8 14.90 (REVISE TH#OPEN-ALCOHOL-BEHAVIOR-THEORY-2-1
          ((NO-INTERPRET ALCOHOL-GOING) (OCCURSAT QSTATE-1 ALCOHOL-DRY))
          (ALCOHOL-GOING ALCOHOL-DRY)))
(8 11.06 (REVISE TH#OPEN-ALCOHOL-BEHAVIOR-THEORY-5-1
          ((NO-INTERPRET ALCOHOL-GOING) (OCCURSAT QSTATE-1 ALCOHOL-DRY))
          (ALCOHOL-GOING ALCOHOL-DRY)))
```

The priority of the decision-pool task guarantees that once a decision-pool task reaches the front of the agenda, PHINEAS has reached the hypothesis selection point. The first decision-pool task (boiling) collects all other hypotheses from the task agenda waiting for decision-pool execution. In this case there are no others. It then applies the LEF to this set. Because there is only one candidate explanation at this point, boiling is selected as the final explanation.

As shown in Chapter 6, had the dissolving hypothesis been simulated, it too would have produced an adequate explanation. However, it was deemed inferior to the boiling explanation because of its creation of new predicates (C_{VE}). It was given a prominent position in preceding chapters due to its demonstration of a large percentage of the main ideas in this thesis.[6]

8.4.1 Discussion

Variations on this example have been used to explore PHINEAS' range of behavior with differing amounts of information:

- *With observation duration.* A phenomenon's duration is often an important key to theory development.[7] In this case, it took 15 hours for the alcohol to completely disappear. For the sake of this example, a "prototypical" amount of time a beaker of alcohol takes to boil away was chosen to be the range of 30 minutes to 4 hours. When this information is included, the boiling hypothesis is accepted with the warning

  ```
  Hypothesis OPEN-ALCOHOL-BEHAVIOR-THEORY-1-1 is adequate, but fails duration bounds
  ```

  At selection time, adequate candidates consistent with the observation's duration are preferred over those having an inconsistent duration range. In this particular example,

[6]As a side note, the dissolving hypothesis came about as a complete surprise. When the "evaporation" example was first attempted, I expected PHINEAS to propose boiling or liquid flow. When dissolving appeared as a candidate analogue as well, I initially considered this a flaw since it seemed clear that dissolving was not analogous. Further consideration reveals numerous analogous aspects between the two situations.

[7]DATMI is able to use real-valued duration information to influence measurement interpretation.

boiling is the only hypothesis and is therefore accepted despite the unusually long time it took the alcohol to vanish. At the present time, this is the only example which takes advantage of DATMI's ability to reason about the durations of events. The approach represents an important aspect of the theory development problem. More work is needed to better understand its applications.

- *Without change rate information.* When the derivative of the alcohol's change rate is unknown (i.e., only the alcohol's decrease is observed), the leaky container and liquid flow hypotheses become adequate explanations. When PHINEAS reaches the hypothesis selection point, there are three candidate explanations to choose from. When the LEF is applied to this set, all hypotheses pass the first three criteria (C_{CE}, C_{VE}, C_{CA}). They all fail the fourth criteria, *assumed entities* (C_{AE}). The leaky cup and two container liquid flow explanations assume three objects, while the boiling hypothesis assumes two. Thus, boiling is selected as the best explanation.

- *With temperature information.* When the alcohol's temperature is included as an observed quantity (`Temperature(alcohol1) < Tboil(alcohol1)`), the boiling hypothesis is invalidated. This is reflected by transfer proposing a new process analogous to boiling which is active for temperatures below the boiling temperature. The new process is otherwise identical to boiling.

- *With temperature, without change rate.* When the change rate information is not included, the leaky container and liquid flow hypotheses become adequate. When the alcohol's temperature is included, the boiling analogy produces a new kind of boiling process for temperatures below the boiling temperature. Thus, it fails the *composite assumptions* (C_{CA}) criterion. As a result, PHINEAS rejects the "boiling" explanation and selects the leak and liquid flow hypotheses. Each is identical under the LEF, as they both assume the existence of three entities.

Chapter 9

Examples

PHINEAS has been used to hypothesize explanations of observations from a variety of physical domains and situations. Previous chapters have focused on its investigations of evaporation, via examples of alcohol's behavior in open and closed containers. This chapter describes PHINEAS' operation for several additional examples. It also analyzes the success and limitations of each example.

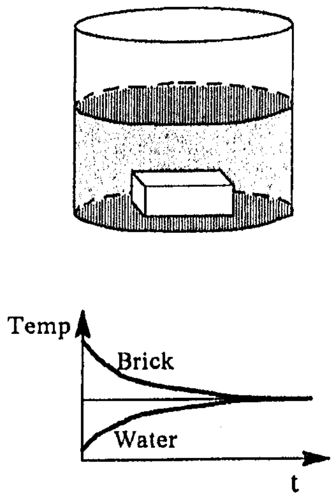
9.1 Caloric Theory of Heat Flow

The view of heat as a material substance originates from the Greeks and dominated thermal science during the eighteenth and first half of the ninteenth centuries (Roller, 1961). During the eighteenth century, it developed into what is generally called the caloric theory of heat, which postulates a heat substance called *caloric*. The temperature of an object was believed proportional to the amount of caloric present. Furthermore, caloric tended toward equilibrium, causing it to flow between bodies placed in contact until an equilibrium of their temperatures was achieved. This section describes how PHINEAS achieves a naive level of the caloric view when shown thermal behavior for the first time.

The explanation task is illustrated in Figure 9.1. When a hot brick is immersed in cold water, their temperatures will asymptotically approach each other until reaching equality.

PHINEAS begins by searching memory for analogous behavior.[1] First, the behavioral abstractions describing the observation are used to probe memory. In this case, dual-approach-finish applies, which characterizes two quantities asymptotically approaching each other and reaching equality. Only one candidate analogue demonstrates

[1]PHINEAS' default domain theory includes three thermal processes: heat flow, boiling, and heat-replenish. They are removed for this example.



```
(Physob brick)
(Solid brick)
(Volume-solid brick)
(Liquid water1)
(Contained-liquid water1)
(Container-of water1 bucket)
(Substance-of water1 water)
(Immersed-in brick water1)
(Contained-in water1 bucket)
(Dual-approach-finish 2-obj-hf)
(Meets (Situation 2-obj-hf-sit0
           (Set (Decreasing (Temperature-in brick))
                (Increasing (Temperature-in water1))
                (Greater-Than (A (Temperature-in brick))
                              (A (Temperature-in water1)))))
         (Situation 2-obj-hf-sit1
           (Set (Constant (Temperature-in brick))
                (Constant (Temperature-in water1))
                (Equal-to (A (Temperature-in brick))
                          (A (Temperature-in water1))))))
```

Figure 9.1: An unexplained thermal situation. When a hot brick is immersed in cold water, the brick's temperature decreases and the water's temperature increases. This transitions to a state in which the temperatures are constant and equal.

this abstract behavior – 2-container-liquid-flow. This scenario describes liquid flowing from one container (`beaker3`) to another (`vial2`), through a pipe (`pipe1`) connecting them. Using SME to compare the current and recalled situations, PHINEAS determines that the roles of the beaker and vial in the liquid flow description correspond to the roles of the brick and water in the thermal situation, respectively. Additionally, it finds that `pressure` in the liquid flow situation corresponds to `temperature` in the thermal situation.

Upon completion of access, PHINEAS attempts to map the relevant liquid flow domain theory into the current thermal situation. First, the domain theory used to explain the 2-container-liquid-flow experience is retrieved. This consists of the liquid flow process and two instantiations of

(`Contained-Fluid` *contained-fluid substance container*)

one for the beaker water and one for the vial water. The partial mapping established during access is then declared and SME invoked. SME's results are shown in Figure 9.2. Its candidate inferences propose a new contained-fluid relationship, in which the temperature of the container (`brick` and `water1`) is proportional to the amount of substance it contains. This

```
            SME Version 2E
    Analogical Match from 2-CONTAINER-LF-THEORY to 2-OBJ-HF-THEORY.

Rule File: csm.rules
-------------------------------------------------------
# MH's | # Gmaps |      1st,2nd,Worst     | RelGroups |
  17   |    1    |  2.07 /  2.07 /  2.07 |  ACTIVE   |
-------------------------------------------------------
Total Run Time:    0 Minutes,  1.398 Seconds
BMS Run Time:      0 Minutes,  0.299 Seconds
Best Gmaps: { 1 }

Gmap #1:   (PRESSURE-IN-82 TEMPERATURE-IN-117)  (PRESSURE-IN-80 TEMPERATURE-IN-118)
           (BEAKER3 BRICK)  (VIAL2 WATER1)
  Weight: 2.0710
  Candidate Inferences:
      (B-EXPLAINS
         (SET (PACKET-DEFINITION
                 (CONTAINED-FLUID (:SKOLEM CS-WATER-BEAKER) (:SKOLEM WATER) BRICK)
                     (SET (PHYSOB (:SKOLEM CS-WATER-BEAKER))
                          (CONTAINER-OF (:SKOLEM CS-WATER-BEAKER) BRICK)
                          (SUBSTANCE-OF (:SKOLEM CS-WATER-BEAKER) (:SKOLEM WATER))
                          (QUANTITY (AMOUNT-OF (:SKOLEM CS-WATER-BEAKER)))
                          (QUANTITY TEMPERATURE-IN-118)
                          (QPROP TEMPERATURE-IN-118 (AMOUNT-OF (:SKOLEM CS-WATER-BEAKER)))))
              (PACKET-DEFINITION
                 (CONTAINED-FLUID (:SKOLEM CS-WATER-VIAL) (:SKOLEM WATER) WATER1)
                     (SET (PHYSOB (:SKOLEM CS-WATER-VIAL))
                          (CONTAINER-OF (:SKOLEM CS-WATER-VIAL) WATER1)
                          (SUBSTANCE-OF (:SKOLEM CS-WATER-VIAL) (:SKOLEM WATER))
                          (QUANTITY (AMOUNT-OF (:SKOLEM CS-WATER-VIAL)))
                          (QUANTITY TEMPERATURE-IN-117)
                          (QPROP TEMPERATURE-IN-117 (AMOUNT-OF (:SKOLEM CS-WATER-VIAL)))))
              (PROCESS-DEFINITION (:SKOLEM LIQUID-FLOW) (:SKOLEM PI1)
                 (IMPLIES
                    (AND (INDIVIDUAL (:SKOLEM WATER)
                            (CONDITIONS (SUBSTANCE (:SKOLEM WATER)) (LIQUID (:SKOLEM WATER))))
                         (INDIVIDUAL BRICK (CONDITIONS (CAN-CONTAIN BRICK (:SKOLEM WATER))))
                         (INDIVIDUAL (:SKOLEM CS-WATER-BEAKER)
                            (CONDITIONS (CONTAINED-FLUID (:SKOLEM CS-WATER-BEAKER) (:SKOLEM WATER) BRICK)))
                         (INDIVIDUAL WATER1 (CONDITIONS (CAN-CONTAIN WATER1 (:SKOLEM WATER))))
                         (INDIVIDUAL (:SKOLEM CS-WATER-VIAL)
                            (CONDITIONS (CONTAINED-FLUID (:SKOLEM CS-WATER-VIAL) (:SKOLEM WATER) WATER1)))
                         (INDIVIDUAL (:SKOLEM PIPE1)
                            (CONDITIONS (FLUID-PATH (:SKOLEM PIPE1))
                                        (PHYSICAL-PATH BRICK WATER1 (:SKOLEM PIPE1))))
                         (FLUID-ALIGNED (:SKOLEM PIPE1))
                         (GREATER-THAN (A TEMPERATURE-IN-118) (A TEMPERATURE-IN-117))
                         (GREATER-THAN (A (AMOUNT-OF (:SKOLEM CS-WATER-BEAKER))) ZERO))
                    (AND (QUANTITY (FLOW-RATE (:SKOLEM PI1)))
                         (Q= (FLOW-RATE (:SKOLEM PI1)) (- TEMPERATURE-IN-118 TEMPERATURE-IN-117))
                         (GREATER-THAN (A (FLOW-RATE (:SKOLEM PI1))) ZERO)
                         (CTRANS (AMOUNT-OF (:SKOLEM CS-WATER-BEAKER)) (AMOUNT-OF (:SKOLEM CS-WATER-VIAL))
                              (A (FLOW-RATE (:SKOLEM PI1))))))))
         2-OBJ-HF)
```

Figure 9.2: SME$_{CSM}$ mapping from the liquid flow process instance to the hot brick in cold water scenario.

substance is currently unknown, but is analogous to the water in the liquid flow situation. Additionally, it proposes a new process: when two objects of differing temperature are connected by a physical path, the unknown substance continuously flows from the object of higher temperature to the one of lower temperature, at a rate equal to their difference in temperatures.

The candidate inferences are then passed to the transfer task. First, it determines that none of the proposed expressions are inconsistent in their current state. Next, these inferences are inspected for the presence of skolem objects. Four are found: **(:skolem cs-water-beaker)**, **(:skolem cs-water-vial)**, **(:skolem water)**, and **(:skolem pipe1)**. The first two are compound objects (i.e., objects defined solely by their constituents) and are therefore ignored. The unknown **(:skolem pipe1)** indicates that no correspondent for the pipe connecting the beaker and vial was found. However, when the abductive retriever is given the task of solving the conjunction

(Physical-Path brick water1 *?pipe***)** ∧ **(Fluid-Aligned** *?pipe***)**
  ∧ **(Fluid-path** *?pipe***)**

it finds **(Physical-Path brick water1 (common-face brick water1))** and **(Fluid-Aligned (common-face brick water1))** are true in the current scenario and the third conjunct is assumable. PHINEAS therefore establishes **(common-face brick water1)** as the analogue for **pipe**.

The remaining unknown, **(:skolem water)**, indicates that no correspondent for the water flowing from beaker to vial was found. Additionally, no correspondent is found when the abductive retriever is invoked on the conjunction

(Substance *?pipe***)** ∧ **(Liquid** *?pipe***)** ∧
(Can-Contain brick *?pipe***)** ∧ **(Can-Contain water1** *?pipe***)**

When **create-entity** is used to make a new entity token for the missing water correspondent, a contradiction is found:

(Liquid sk-water-1) ∧ **(Volume-Solid brick)** → ¬**(Can-Contain brick sk-water-1)**

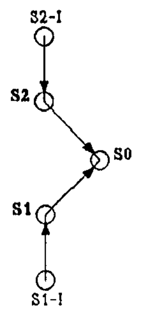
As a result, **(Liquid sk-water-1)** is changed to **(SK-Phase-1 sk-water-1)**, with **SK-Phase-1** added as a new child predicate to **Phase**.

At this point, the transfer task is completed, resulting in the model shown in Figure 9.3. This model postulates that the brick and water each contain **sk-water1-1**, and their temperatures are proportional to how much of it they contain. Additionally, it proposes the new **Process-1**, which might be called a heat flow process. It indicates that **sk-water1-1**

```
     Hypotheses for theory 2-OBJ-HF-THEORY-1-1 derived from 2-CONTAINER-LF:

(DEFPROCESS  (PROCESS-1 ?SUBST ?SOURCE ?SRC-CS ?DESTINATION ?DST-CS ?PATH)
  INDIVIDUALS  ((?SUBST :CONDITIONS (SUBSTANCE ?SUBST) (SK-PHASE-1 ?SUBST))
               (?SOURCE :CONDITIONS (CAN-CONTAIN ?SOURCE ?SUBST))
               (?SRC-CS :CONDITIONS (CONTAINED-FLUID-1 ?SRC-CS ?SUBST ?SOURCE))
               (?DESTINATION :CONDITIONS (CAN-CONTAIN ?DESTINATION ?SUBST))
               (?DST-CS :CONDITIONS (CONTAINED-FLUID-1 ?DST-CS ?SUBST ?DESTINATION))
               (?PATH :CONDITIONS (FLUID-PATH ?PATH)
                                  (PHYSICAL-PATH ?SOURCE ?DESTINATION ?PATH)))
  PRECONDITIONS  ((FLUID-ALIGNED ?PATH))
  QUANTITYCONDITIONS  ((GREATER-THAN (A (TEMPERATURE-IN ?SOURCE))
                                     (A (TEMPERATURE-IN ?DESTINATION)))
                      (GREATER-THAN (A (AMOUNT-OF ?SRC-CS)) ZERO))
  RELATIONS  ((QUANTITY (FLOW-RATE ?SELF))
             (Q= (FLOW-RATE ?SELF)
                 (- (TEMPERATURE-IN ?SOURCE) (TEMPERATURE-IN ?DESTINATION)))
             (GREATER-THAN (A (FLOW-RATE ?SELF)) ZERO))
  INFLUENCES ((CTRANS (AMOUNT-OF ?SRC-CS) (AMOUNT-OF ?DST-CS) (A (FLOW-RATE ?SELF)))))

(DEFENTITY  (CONTAINED-FLUID-1 ?V-1 ?V-2 ?V-3)
    (CONTAINER-OF ?V-1 ?V-3)
    (SUBSTANCE-OF ?V-1 ?V-2)
    (QUANTITY (AMOUNT-OF ?V-1))
    (QUANTITY (TEMPERATURE-IN ?V-3))
    (QPROP (TEMPERATURE-IN ?V-3) (AMOUNT-OF ?V-1)))

(ASSUME (SUBSTANCE SK-WATER-1))
(ASSUME (SK-PHASE-1 SK-WATER-1))
(ASSUME (CAN-CONTAIN BRICK SK-WATER-1))
(ASSUME (CONTAINED-FLUID-1 SK-CS-WATER-BEAKER-1 SK-WATER-1 BRICK))
(ASSUME (CAN-CONTAIN WATER1 SK-WATER-1))
(ASSUME (CONTAINED-FLUID-1 SK-CS-WATER-VIAL-1 SK-WATER-1 WATER1))
(ASSUME (FLUID-PATH (COMMON-FACE BRICK WATER1)))
```

Figure 9.3: Final PHINEAS hypothesis explaining the hot brick in cold water behavior.

Quantity	S2-I	S2	S0	S1-I	S1
Ds[FLOW-RATE(PI0)]	–	–	–	-1	-1
Ds[FLOW-RATE(PI1)]	-1	-1	–	–	–
Ds[AMOUNT-OF(SK-CS-WATER-BEAKER-1)]	-1	-1	0	1	1
Ds[AMOUNT-OF(SK-CS-WATER-VIAL-1)]	1	1	0	-1	-1
Ds[PRESSURE(SK-CS-WATER-BEAKER-1)]	-1	-1	0	1	1
Ds[PRESSURE(SK-CS-WATER-VIAL-1)]	1	1	0	-1	-1
Ds[TEMPERATURE-IN(BRICK)]	-1	-1	0	1	1
Ds[TEMPERATURE-IN(WATER1)]	1	1	0	-1	-1
A[AMOUNT-OF(SK-CS-WATER-BEAKER-1)]	>0	>0	>0	>0	=0
A[AMOUNT-OF(SK-CS-WATER-VIAL-1)]	=0	>0	>0	>0	>0
A[TEMPERATURE-IN(BRICK)] A[TEMPERATURE-IN(WATER1)]	>	>	=	<	<
ACTIVE(PI0)	F	F	F	T	T
ACTIVE(PI1)	T	T	F	F	F



Processes:

PI0: PROCESS-1(SK-WATER-1 WATER1 SK-CS-WATER-VIAL-1 BRICK SK-CS-WATER-BEAKER-1 (COMMON-FACE BRICK WATER1))
PI1: PROCESS-1(SK-WATER-1 BRICK SK-CS-WATER-BEAKER-1 WATER1 SK-CS-WATER-VIAL-1 (COMMON-FACE BRICK WATER1))

Figure 9.4: Envisionment produced by the hypothesized caloric model when applied to the brick immersed in water scenario. Some states are split by QPE (e.g., S2 and S2-I). States are distinguished only by derivative and process values. They are split when this distinction produces a state lasting an interval of time (S2) and also lasting for an instant (S2-I).

will flow from the object of higher temperature to the object of lower temperature. PHINEAS does not perform generalization beyond replacing constants with variables.[2] Hence, only the brick and water1 are believed to contain sk-water1-1.

Only one test remains – verify the adequacy of the model in explaining the original observation. As shown in Figure 9.4, the model produces a five-state envisionment, with state S2 transitioning to state S0 demonstrating that the model is able to predict the observed temperature changes. In state S2, Process-1 is active, the substance sk-water-1 is flowing from the brick to the water, and the temperature of the brick is decreasing while the temperature of the water is increasing, each at a rate equal to the difference in their temperature. In state S0, the brick and water temperatures are equal and all quantities are constant.

[2]The explanation-based learning community has shown that this is not sufficient to ensure proper generalization (DeJong & Mooney, 1986; Mitchell et al., 1986). Explanation-based generalization should be performed at this step, but it has not been a problem so far.

9.1.1 Discussion

This example demonstrates several points. First, identifying the path of thermal flow shows the utility of transfer in filling out an incomplete analogical mapping. An analogy will often evoke additional remindings or perspectives made relevant by its consideration. Second, the example illustrates PHINEAS' ability to create new object tokens (i.e., `sk-water-1`) when a skolem object produced by mapping cannot be resolved. Further, it is able to distinguish between assuming the presence of an unobserved object and conjecturing a theoretically novel entity. This is important information for theory evaluation and selection processes.

The example also points to an interesting aspect of PHINEAS' behavior. As in any knowledge intensive approach to learning, PHINEAS' results are dependent on the knowledge it has and how that knowledge is expressed. In the model shown in Figure 9.3, two questionable relations appear:

`(Fluid-Path ?path)` and `(Fluid-aligned ?path)`

These are odd predicates for a theory of heat. Ideally, we would like to see both replaced by new predicates representing the concepts of *heat path* and *heat aligned.* However, they are consistent with the current example, since `?path` is the common surface between the brick and water in this instance.

There are several ways in which the prefered result could have been achieved. First, if PHINEAS had been asked to explain a situation where a solid metal bar was acting as the path, its inability to act as a fluid path would have been detected and new path predicates created. Second, the current consistency of the two predicates may be viewed as a flaw in the domain theory. Using `Fluid-Path` as a one place predicate and stating it as a precondition to `Process-1` has detached its meaning from the particular fluid it is to transport. Finally, important experiential information is lacking, since this is the first and only time PHINEAS has encountered thermal flow. The plausibility of PHINEAS' conjectures must be evaluated with respect to the information it possesses. If `sk-water-1` is thought to be a fluid, there is no evidence indicating that the `sk-water-1` fluid requires something different than standard fluid paths (e.g., heat flow through a solid metal bar). Often, several examples of a phenomenon are necessary to isolate the conditions in which it occurs.

9.2 Oscillation

Oscillation is a common phenomenon in physical systems. PHINEAS' initial knowledge contains theories about a prototypical spring-mass system, in which a spring is anchored

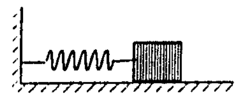
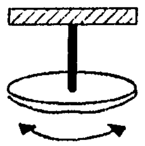
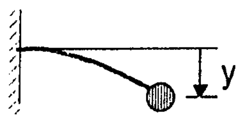
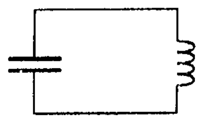
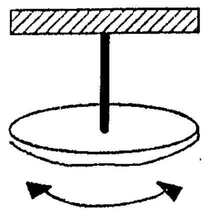
Oscillator	Period of Oscillation	Compliant Element	Inertial Element
 Spring–Mass Oscillator	$\tau = 2\pi\sqrt{\frac{m}{k}}$	Spring (k)	Block's mass (m)
 Torsional Pendulum	$\tau = 2\pi\sqrt{\frac{J}{K}}$	Torsional Stiffness (K)	Disk's moment of Inertia (J)
 Cantilevel Pendulum	$\tau = 2\pi\sqrt{\frac{ml^3}{3EI}}$	Flexural Stiffness (EI)	Ball's mass (m)
 L–C Electric Circuit	$\tau = 2\pi\sqrt{\frac{L}{1/C}}$	Compliance (1/C)	Inductor (L)

Figure 9.5: Some simple harmonic oscillators. (Adapted from Shive & Weber, 1982; Thomson, 1948)

to a wall on one end and attached to a mobile mass on the other (Figure ??). If the block is pulled and then released, it will oscillate back and forth forever.[3] Drawing from this knowledge, PHINEAS has been able to explain several examples of simple harmonic motion, all depicted in Figure 9.5.

Here we consider the behavior of a torsion oscillator. PHINEAS is initially given a description of a ball rotating while suspended by a string, and the ball's sinusoidal behavior, represented as a cycle of eight qualitatively described temporal intervals. Each interval

[3]Modeling friction and resistance in oscillators is a difficult modeling problem in QP theory. Ideal oscillators are discussed throughout this section.



```
(Connected string1 ball9)
(String string1)
(Ball ball9)
(Rotating-Object ball9)
(Twisting-Object string1)
(Sinusoidal ball-oscillating)

                              o
                              o
                              o

(Situation ball-string-s3
   (Set (Decreasing (Angular-displacement string1))
        (Decreasing (Angle ball9))
        (Increasing (Angular-Velocity ball9))
        (Less-than (A (Angular-Velocity ball9)) ZERO)
        (Less-Than (A (Angle ball9)) zero)
        (Less-Than (A (Angular-displacement string1)) zero)))

                              o
                              o
                              o
```

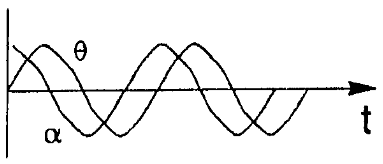


Figure 9.6: A torsional oscillator.

contains facts describing the derivatives and amounts of angle and angular velocity. In addition, it is told that the ball is a rotating object and the string is a twisting object.

When PHINEAS probes memory for prior experience with sinusoidal oscillation, it finds the spring-mass system. A detailed comparison of this behavior and that of the rotating ball reveals a correspondence between the eight behavioral states of each system. This correspondence indicates that the compressing spring corresponds to the twisting string and the translating block corresponds to the rotating ball. Additionally, position is mapped to angle and velocity is mapped to angular velocity, due to their similar behavior.

With a behavioral correspondence established, PHINEAS fetches the domain theory used to explain the spring-mass system. This consists of a Force process which applies the spring's force to the attached block, a spring-mass-system object definition describing the system's total energy and the relationship between the block's position and the spring's displacement, and a spring object definition describing its restorative force as a function of displacement. When the spring-mass theory is mapped into the oscillating ball situation, transfer first examines each relation and finds no inconsistencies. The transfer phase next checks for skolem objects in the candidate inference and finds (:skolem sm-sys). sm-sys

```
Hypotheses for theory BALL-OSCILLATING-THEORY-3-1 derived from SPRING-MASS-OSCILLATING.

(DEFENTITY  (SPRING-MASS-SYSTEM-22 ?V-78 ?V-79 ?V-80)
    (CONNECTED ?V-79 ?V-80)
    (QUANTITY (TOTAL-ENERGY ?V-78))
    (NOT (LESS-THAN (A (TOTAL-ENERGY ?V-78)) ZERO))
    (EQUAL-TO (D (TOTAL-ENERGY ?V-78)) ZERO)
    (Q= (TOTAL-ENERGY ?V-78) (+ (KINETIC-ENERGY ?V-80) (POTENTIAL-ENERGY ?V-79)))
    (Q= (ANGULAR-DISPLACEMENT ?V-79) (ANGLE ?V-80))
    (FORCE-APPLICATION (RESTORATIVE-FORCE ?V-79) (ANGULAR-VELOCITY ?V-80)))

(DEFENTITY  (SPRING-8 ?V-76)
    (QUANTITY (ANGULAR-DISPLACEMENT ?V-76))
    (QUANTITY (RESTORATIVE-FORCE ?V-76))
    (QPROP- (RESTORATIVE-FORCE ?V-76) (ANGULAR-DISPLACEMENT ?V-76))
    (CORRESPONDENCE ((A (RESTORATIVE-FORCE ?V-76)) ZERO)
                    ((A (ANGULAR-DISPLACEMENT ?V-76)) ZERO))
    (QUANTITY (POTENTIAL-ENERGY ?V-76))
    (NOT (LESS-THAN (A (POTENTIAL-ENERGY ?V-76)) ZERO))
    (Q= (POTENTIAL-ENERGY ?V-76)
        (* (ANGULAR-DISPLACEMENT ?V-76) (ANGULAR-DISPLACEMENT ?V-76))))

(ASSUME (SPRING-MASS-SYSTEM-22 SK-SM-SYS-23 STRING1 BALL9))
(ASSUME (SPRING-8 STRING1))
(ASSUME (FORCE-APPLICATION (RESTORATIVE-FORCE STRING1) (ANGULAR-VELOCITY BALL9)))
(ASSUME (QUANTITY (RESTORATIVE-FORCE STRING1)))
```

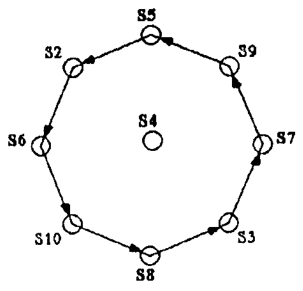
Figure 9.7: Final PHINEAS hypothesis explaining the behavior of the torsion oscillator.

is a token representing the spring-mass system taken as a whole. This compound object token is replaced by sk-sm-sys-23, which represents the newly defined string-ball system:

```
(spring-mass-system-22 sk-sm-sys-23 string1 ball9)
```

The proposed model of the rotating ball scenario, shown in Figure 9.7, is now usable. When the model is applied to the ball–string pair, it produces an envisionment containing an eight-state cycle, as shown in Figure 9.8. When PHINEAS examines the envisionment, it finds a perfect match between the observed and predicted behavior. Thus, the model is adequate and the explanation process is completed.

```
                    o
                    o
                    o

In behavioral segment 3
            (Angular-Displacement string1) is Decreasing
            (Angle ball9) is Decreasing
            (Angular-Velocity ball9) is Increasing
            (A (angular-velocity ball9)) is Less Than zero
            (A (angle ball9)) is Less Than zero
            (A (angular-displacement string1)) is Less Than zero

     Due to the following processes being active:
         FORCE-PROCESS(STRING1 RESTORATIVE-FORCE
                       BALL9 ANGULAR-VELOCITY)

                    o
                    o
                    o
```



Quantity	S5	S2	S6	S10	S8	S3	S7	S9	S4
Ds[ANGLE(BALL9)]	-1	-1	-1	0	1	1	1	0	0
Ds[ANGULAR-DISPLACEMENT(STRING1)]	-1	-1	-1	0	1	1	1	0	0
Ds[ANGULAR-VELOCITY(BALL9)]	-1	0	1	1	1	0	-1	-1	0
Ds[KINETIC-ENERGY(BALL9)]	1	0	-1	0	1	0	-1	0	0
Ds[POTENTIAL-ENERGY(STRING1)]	-1	0	1	0	-1	0	1	0	0
Ds[RESTORATIVE-FORCE(STRING1)]	1	1	1	0	-1	-1	-1	0	0
ACTIVE(PI0)	T	T	T	T	T	T	T	T	T
ACTIVE(PI1)	T	T	T	T	T	T	T	T	T
A[RESTORATIVE-FORCE(STRING1)]	<0	=0	>0	>0	>0	=0	<0	<0	=0
A[ANGULAR-VELOCITY(BALL9)]	<0	<0	<0	=0	>0	>0	>0	=0	=0

Processes:

```
PI0: FORCE-PROCESS(STRING1,BALL9)
PI1: DERIVATIVE-PROCESS(ANGLE(BALL9),ANGULAR-VELOCITY(BALL9))
```

Figure 9.8: Complete envisionment produced by the hypothesized torsional oscillator model.

9.2.1 Discussion

The behavioral descriptions used in this example are significantly larger and more ambiguous than any other SME has been applied to (55 expressions each, producing 1,972 gmaps). It took approximately 45 minutes for SME to compare the spring and torsion behavioral descriptions. Although this time is large, early attempts at the example were competely unsuccessful. After several hours, SME would exhaust the memory capacity of the machine.[4] Two modifications have been crucial to the example's current success. First, the nested representation of time described in Chapter 3 provides significant constraint on the mapping process. Second, relaxing structural consistency to allow relational groups reduces fragmentation, further constraining the process.

9.2.1.1 LC Circuit

The LC circuit is one of the most difficult scenarios to which PHINEAS has been applied. It is initially given a description of the circuit's sinusoidal behavior in terms of the derivatives and amounts of charge and current. In addition, it is told that L and C are an `inductor` and a `capacitor`, respectively, although PHINEAS has no knowledge about inductors or capacitors. Finally, PHINEAS is given the hint that the capacitor possesses an electrical restorative force. Without the hint, two answers are equally plausible rather than one (i.e., L might possess the electrical restorative force). The hint merely serves to simplify the discussion.

PHINEAS successfully maps `position` to `charge`, `velocity` to `current`, and the spring and mass to C and L respectively. When the charge and current model is applied to the LC circuit, an envisionment containing a consistent eight-state cycle is produced.

This example required a small degree of tailoring before it would work as well as the other oscillator analogies. The reasons for this provide insight to a hard problem. The original attempt at modeling the spring-mass system described the spring's length in terms of its rest length and the block's current position. When mapped to the LC circuit scenario, the quantities `length` and `rest-length` were applied to the capacitor, with no apparant contradiction. This misses the point of the analogy entirely. It appears that a general solution to the problem requires in-depth study of the role of `length` and `rest-length` in the spring-mass system. For example, the spring's rest length should probably correspond to the capacitor's point of zero voltage. The problem was resolved for the above example by reformulating the spring-mass model. `length` and `rest-length` were removed, with the

[4]All examples in this thesis are taken from PHINEAS running on a Symbolics 3640 with 12 Mb Ram and 100 Mb disk swap space.

```
(Substance solution-stuff)
(Liquid solution-stuff)
(Contained-liquid solution1)
(Container-of solution1 container1)
(Substance-of solution1 solution-stuff)
(Solution solution1)
(Contained-liquid solution2)
(Container-of solution2 container2)
(Substance-of solution2 solution-stuff)
(Solution solution2)
(Can-contain container1 solution-stuff)
(Can-contain container2 solution-stuff)
(Touching container1 common-wall)
(Touching container2 common-wall)
(Touching container1 common-floor)
(Touching container2 common-floor)
(Volume-solid common-floor)
(Greater-than (A (amount-of solution1)) zero)
(Greater-than (A (amount-of solution2)) zero)

(Meets (Situation osmosis-sit1
            (Set (Decreasing (Amount-of solution1))
                 (Increasing (Concentration solution1))
                 (Increasing (Amount-of solution2))
                 (Decreasing (Concentration solution2))
                 (Greater-than (A (Amount-of solution1))
                               (A (Amount-of solution2)))
                 (Greater-than (A (Concentration solution2))
                               (A (Concentration solution1)))))
       (Situation osmosis-sit2
            (Set (Constant (Amount-of solution1))
                 (Constant (Concentration solution1))
                 (Constant (Amount-of solution2))
                 (Constant (Concentration solution2))
                 (Equal-to (A (Concentration solution2))
                           (A (Concentration solution1))))))
```

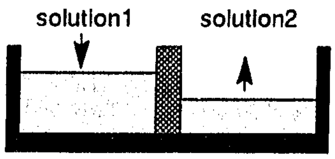


Figure 9.9: Two containers sharing a common wall of unknown substance. Each container holds a solution.

spring force determined by the position of the block. This was sufficient to produce the results described above.

9.3 Osmosis

Figure 9.9 depicts two containers sharing a common wall composed of an unknown substance. Each container holds different amounts of a solution. We observe that the amount of solution is decreasing in the left container (container1) and increasing in the right container (container2). At the same time, the concentration of solution1 is increasing while the concentration of solution2 is decreasing. What is happening?

PHINEAS begins by exploring memory for potential analogues. It finds four possibilities, but at the same time encounters significant ambiguity, as evidenced by the number of interpretations for each analogue:

Analogue	# interpretations	similarity score
2-container-liquid-flow	(2)	17.7
leaking-container	(1)	16.8
dissolving	(2)	14.8
boiling	(6)	16.4

The multiple interpretations arise for two reasons. First, there is ambiguity over whether to map `amount-of` to the solution's `amount-of` quantity or the solution's `concentration` quantity. Second, the observation describes the change of two contained liquids. Thus, in the dissolving analogue, one interpretation views the salt mapping to `solution1` while the other views the salt mapping to `solution2`. A similar situation occurs for the boiling analogue. However, the reason is of significance. The one-to-one criterion prevents `Contained-liquid` from mapping to both `Contained-liquid` and `Contained-gas`. Additionally, both gas and liquid are in the same container for the boiling scenario. This again leads to ambiguity in mapping to `container1` or `container2`.

PHINEAS examines each of these possible analogues and finds that none of them completely accounts for the observed behavior. The greatest coverage comes from the two-container-liquid-flow analogue. Its entire hypothesis consists of:

```
(Fluid-Path common-wall)
```

The envisionment for this hypothesis was shown in Figure 6.5. It consistently describes the loss of `solution1` and the rise of `solution2`. However, it incorrectly predicts that the concentration of each will remain constant.

Because the theory revision module is still incomplete, and all of the hypotheses fail to adequately explain the observation, PHINEAS concludes with a task queue filled with pending revision tasks. For this example, I had PHINEAS return the best, inadequate explanation available. When the LEF is applied, the two container liquid flow hypothesis is selected, due to its resting on a single assumption about the nature of the wall separating the two containers. It is important to note that PHINEAS' work was made needlessly complex due to the experimental settings of the task priorities. The primary point of this "osmosis" example is to show how PHINEAS uses global similarity to focus on the most plausible explanations first. Because of the observation's strong similarity to liquid flow, PHINEAS developed the liquid flow hypothesis first and only afterwards did it entertain the other possibilities.

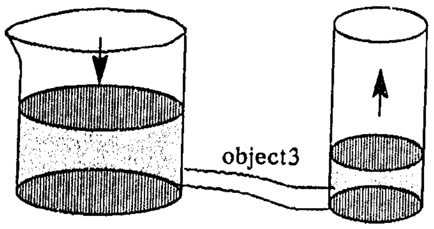


Figure 9.10: A beaker and a vial, each containing water, are connected by object3. What is causing the water in the beaker to decrease while the water in the vial is increasing?

9.4 Ordinary Liquid Flow

One of the goals of this work is to show the feasibility of similarity as a single source for both analogical and more traditional, deductive or abductive explanations. Consider the scenario illustrated in Figure ??:

- A beaker and a vial, each containing water, are connected by object3. What is causing the water in the beaker to decrease while the water in the vial is increasing?

PHINEAS begins by probing memory for the best set of candidate analogues. It finds four initial possibilities:

Similarity Score	*Analogue*
28.07	two container liquid flow
15.77	leaky container
14.24	dissolving
14.14	boiling

Examining each of these possibilities, PHINEAS finds that not only is the two container liquid flow scenario most similar to the current situation, it is the only candidate that produces a consistent set of predictions. Thus, PHINEAS concludes with the single assumption:

```
(Fluid-Path object3)
```

which is sufficient to completely explain the observed behavior (see Figure 9.11). Under this assumption, the situation is viewed as a normal instance of liquid flow, with object3 serving as the fluid path.

```
Hypotheses for theory OBJECT3-FLOW-THEORY-5 derived from 2-CONTAINER-LF:

    (ASSUME (FLUID-PATH OBJECT3))

Analysis of OBJECT3-LF according to theory OBJECT3-FLOW-THEORY-5

    In behavioral segment 1
        (PRESSURE-IN VIAL6) is Increasing
        (AMOUNT-OF CS-WATER-VIAL1) is Increasing
        (PRESSURE-IN BEAKER6) is Decreasing
        (AMOUNT-OF CS-WATER-BEAKER1) is Decreasing
        (A (PRESSURE-IN BEAKER6)) is Greater Than (A (PRESSURE-IN VIAL6))

        Due to the following processes being active:
           LIQUID-FLOW(WATER BEAKER6 CS-WATER-BEAKER1 VIAL6 CS-WATER-VIAL1 OBJECT3)

    In behavioral segment 2
        (PRESSURE-IN VIAL6) is Constant
        (AMOUNT-OF CS-WATER-VIAL1) is Constant
        (PRESSURE-IN BEAKER6) is Constant
        (AMOUNT-OF CS-WATER-BEAKER1) is Constant
        (A (PRESSURE-IN BEAKER6)) is Equal To (A (PRESSURE-IN VIAL6))

        There are no processes active.
```

Figure 9.11: To explain why the amount of water in the beaker is decreasing and the amount of water in the vial is increasing, PHINEAS requires only a single assumption: `object3` is a fluid path.

Chapter 10

Discussion

There were two primary motivating goals for this thesis. First and foremost, develop a general, working model of analogical processing that is powerful enough to perform a complex reasoning task. This I feel has been accomplished. While PHINEAS is certainly not task-independent, I believe the model it embodies has broad utility. Second, apply that model to the task of providing qualitative explanations for observed physical phenomena, using representation and analysis techniques from the state-of-the-art in qualitative physics to demonstrate scale and generality. This too I feel has been accomplished to a large extent. PHINEAS is able to offer analogically derived explanations for a variety of observations novel to its initial domain theories. It is able to envision the predictions of a developing explanation, from which the explanation's adequacy may be tested and revisions sanctioned.

Embedding the model in a non-trivial performance task is an important methodological constraint for research in machine learning. Learning should provide a service, such as improving system performance or enabling solutions to problems otherwise unsolvable. Techniques developed in support of complex inferences have fewer arbitrary choices than techniques developed specifically for learning research. Additionally, placing learning research in the context of a serious performance task often leads to discovery of fundamental research problems that would have otherwise gone unnoticed. This was certainly true in the development of PHINEAS. To my dismay, I was constantly surprised by problems that, for whatever reason, had not been described in the literature. These include the N-M binding problem, the structure rearrangement problem, and the many ways in which matching predicates based on an ISA hierarchy fails. Conversely, the learning research may help to motivate research in the application task domain. For example, my motivation for the work on large-scale, multiple-perspective qualitative modeling described in (Falkenhainer & Forbus, 1988) arose from frustration in attempting to model analogical access for commonsense physical domains.

Despite the progress made, far more remains undone. This chapter begins by reviewing the results of this work. It then compares the work to related research from different aspects of AI. Finally, a number of important open research problems are discussed.

10.1 Summary of Results

In summary, the primary contributions of this thesis are:

- It presents *verification-based analogical learning*, a view of analogy as an iterative process of hypothesis formation, testing, and revision. This results from the *verification requirement*, which demands that an analogy system analyze the conjectures it makes and attempt to repair them when insufficient.

- It presents *contextual structure-mapping*, an adaptation of Gentner's structure-mapping theory which stresses the need to use context and knowledge about the representations being manipulated as an aid in the mapping process. This has three primary components:

  1. *Pairwise mappability of predicates is context sensitive.* An implemented definition of *functionally analogous* was presented which solves a specialization of this more general problem. Specifically, two expressions are functionally analogous if they provide the same inferential support in the context of the structures being mapped.
  2. The *structure rearrangement problem* for purely structural models (e.g., SMT) was identified, and domain-specific mapping constraints were defined to prevent the problem. Additionally, the structural consistency constraint of SMT was weakened to allow for the mapping of *relational groups.*
  3. The selection criterion for mapping is generalized to include contextual relevance in addition to systematicity. This overcomes the tendency of the pure systematicity criterion to focus on coherent but irrelevant interpretations.

- It describes how contextual structure-mapping may be modeled as a set of match rules given to SME, a general, domain-independent computational model of analogical mapping that has been tested on many examples from numerous domains and in several research projects.

- The *transfer* operation is formalized in the context of explanation. Transfer is concerned with importing candidate inferences proposed by mapping into the target domain and making them operational. This centers around ensuring expression consistency and resolving skolem objects.

- Transfer and mapping have been paired to form a *map and analyze* cycle. Mapping provides focus by globally identifying analogical commonalities between base and target descriptions. This enables transfer to focus analysis on the candidate inferences, which represent either base information inferable for the target or points of discrepancy in the analogical comparison.

- It describes an approach to the problem of analogical access across different domains, which requires retrieval over non-identical features. Shared behavioral abstractions are used to focus attention on a potentially relevant subset of memory. Each experience in this subset is then compared at a detailed level to the current target situation to determine if they share a common relational structure, despite their surface-level differences.

- It introduces a performance-based measure for evaluating the adequacy of an analogically proposed model. In PHINEAS, this centered around the use of qualitative simulation as a form of gedanken experiment to test the predictions of a model against observation.

- It describes an approach to theory revision based on the view that generation of revision hypotheses should not be fundamentally different from generation of the original explanatory hypotheses. It proposes the use of three sources of knowledge. *First principles analysis* provides completeness in suggesting possible revisions. *Precedent-guided revision* uses knowledge of analogous phenomena to provide a focused, ordered set of hypotheses. Finally, *difference-based reasoning* examines prior successes to see what is different about the current situation that may provide empirical explanations for hypothesis plausibility.

- It describes PHINEAS, a fully implemented system based on the proposed analogical explanation architecture. PHINEAS posits qualitative explanations for time-varying descriptions of physical behaviors.

Based on these ideas and my experience with PHINEAS, I have also proposed that interpretation-construction tasks may be viewed as the search for maximal, explanatory similarity between the situation being explained and some explainable scenario. This led to the conjecture that analogy suffices as the central process model for explanation tasks, with distinctions between deductive, abductive, and analogical explanations arising out of the evaluation process. While PHINEAS demonstrates the feasibility of this view, much further research will be required to prove this conjecture.

10.2 Related Research

This thesis covers a range of AI concerns, including analogy, scientific theory formation, and qualitative reasoning. This section identifies important similarities and differences with related research from these various disciplines.

10.2.1 Analogy

Analogy has many facets, making it difficult to thoroughly review all of the work from philosophy, psychology, and AI which might overlap some portion of this research. In the following subsections, a select subset of work in computational analogy most relevant to this thesis is discussed. One should also refer to (Falkenhainer et al., 1987) for a thorough survey of work comparable to SME and analogical mapping. For example, Winston's (1980, 1982) highly influential work and its relation to SME is discussed. See also (Hall, *to appear*) and (Kedar-Cabelli, 1985a) for general surveys of analogy.

10.2.1.1 Burstein's CARL

Burstein (1983, 1985) presents a detailed investigation of how a student may be taught about assignment statements in BASIC through a series of analogies across multiple-domains. CARL interacts with a tutor, who identifies important similarities between variables and boxes, algebraic equality, and human memory. One of CARL's important features is its ability to use multiple analogies in constructing a model of BASIC assignment statements. An analogy with boxes is used to provide a sense of "placing a number *inside* a variable". Algebraic equality is used to repair the conception that "X = Y" places "Y" inside X, rather than the value for Y. Finally, a human metaphor is used to express the feel of a computer as a communicative agent.

The *minimal ascension* mapping rule (Chapters 3 and 5) is adapted from Burstein's method for creating a new predicate when a base relation is inconsistent for the target domain. The minimal ascension technique adds two functions. First, it may be used during mapping to allow an explicit match between non-identical predicates. This match is inversely proportional to the distance of the predicates paired and enables direct comparison of alternate pairings. Second, it is used during transfer to seek existing predicates, not only to create new predicates.

Some of Burstein's work may be viewed as important next steps in the development of PHINEAS. Specifically, Burstein has focused on multiple analogies, in which a working hypothesis about a situation arises through integration of alternate sources of information. Consider the disappearing alcohol example. From a dissolving analogue, PHINEAS was able to hypothesize that the rate of disappearance is proportional to the alcohol's surface area in contact with the atmosphere. From a boiling analogue, PHINEAS was able to tap its knowledge about the vaporization process. However, the program was not able to merge these hypotheses about different aspects of the situation to form a more complete model of the situation. Flexible integration of multiple sources of information has always been part of the desiderata for PHINEAS, but required the development of the current PHINEAS system as a prerequisite.

10.2.1.2 Derivational Analogy

Analogical problem solving seeks to solve a current problem situation by recourse to a previous problem solving episode. Two distinct approaches have been presented by Carbonell: *transformational analogy* (1983b) and *derivational analogy* (1983a). Each recognize that applying old solutions to new problems may involve modification to the original solution. Transformational analogy applies transformation operators to a base solution until it solves the current problem. There are number of problems with this approach, as Carbonell has noted, and derivational analogy was proposed in response. Specifically, mapping only final plans or operator sequences hides important planning information used to derive the sequence. For example, translating a quicksort routine from PASCAL to LISP by direct translation would be far more difficult than reusing the abstract design process used in developing the original PASCAL version (Carbonell, 1983a).

Derivational analogy replays the derivational history of a previously solved problem. The derivation may contain a record of all important decisions made in the course of solving the base problem, including the alternate plans and subgoals considered, justifications for successful steps, reasons behind failed steps, and any other constraints influencing the development of the final solution. The target problem is solved by examining each step in the derivational history. If the justifications for that step hold in the current situation, the step is applied. If the justifications are missing, then alternate support for the step is sought, one of the alternate choices considered in the derivation is applied, or some form of subgoaling is invoked.

An important contribution of Carbonell's proposal is the observation that the mapping process as commonly described in matching models of analogy (e.g., Winston, 1980; Gentner, 1983) is overly simplistic for complex analogical problem solving. That is, mapping may often involve manipulation of the original base description in response to the current target environment. Replay is one approach that enables a base situation to be adapted to the current target instance. In Chapter 5, I proposed that this is an implicit specialization of a more general problem in analogy – the general need for knowledge about the structures being manipulated in the mapping process. The definition of *functionally analogous* is an attempt to address this problem, but is still a specialization of a much larger problem.

As a proposal, derivational analogy leaves many questions unanswered. First, what information should be stored in the derivational history? Clearly, remembering everything about every problem solving experience is prohibitive. If the derivation is too detailed, analogical reasoning could be more time consuming than solving the problem without analogy. Second, the model lacks an explicit account of similarity, particularly important for mapping solutions across domains. As described, derivational analogy is most appropriate when a nearly equivalent problem has been solved in the past. Finally, derivational analogy is based purely on the problem solving paradigm and does not attempt to represent a model for analogy in general. For instance, it has little to say about analogical learning.

10.2.1.3 Kedar-Cabelli's PER

Kedar-Cabelli (1988) describes a method for *purposive explanation replay*, PER. The problem PER addresses is, given an existing plan schema and an old individual that filled one of the roles in the schema, find an individual in the current state that may be substituted for that role and still achieve the plan. Like derivational analogy, PER is a replay method for mapping, which attempts to adapt an old plan for a new situation by rejustifying and replanning around portions of the old plan that are not supported by the current situation. PER is like the functionally analogous portion of PHINEAS' transfer process in that both attempt to find alternate inferential support for unsupported portions of a proposed analogical inference. PHINEAS is an extension of PER in several ways. First, PER's implementation, REPeat, guesses the functional role of an unsupported item by retrieving a horn clause in which the item appears as an antecedent, followed by reproving the consequent. Since a given item may support many different types of consequents, this is a rough heuristic. In PHINEAS, the *cache* slot serves the same purpose, but explicitly states the actual reason why the unsupported item was needed in the base description. Second, since PER picks a candidate base analogue at random (each goal is associated with past planning instances that have solved it in the past), it may spend a great deal of time reproving an analogue that violates PER's *near miss* assumption. PHINEAS uses global similarity throughout its reasoning process, enabling it to focus first on the optimal candidate analogue (optimal in the sense of maximal match). The *map and analyze* cycle further focuses the process by maintaining a global perspective of what has and has not matched between the two situations. Locating one missing correspondent may fortuitously uncover other correspondents due to their interrelatedness. Analyzing each missing correspondent one at a time may fail to detect such discoveries. Finally, PER is designed to generate d-sound inferences only. Of course, this is important for planning, in which the system should attempt to ensure that the proposed plan will succeed. However, it requires that all relevant knowledge be available and is limited to non-learning situations. PHINEAS' mechanisms for seeking assumable, alternate relations may be seen as an attempt to alleviate this problem.

10.2.1.4 Greiner's NLAG

Greiner's (1986, 1988) work, like this thesis, addresses the problem of analogy in the context of i-sound inferences. Given an analogical hint "A is like B" (A∼B), a target problem PT, and a theory *Th*, his NLAG program uses the ***abstraction-based useful analogical inference*** operator $\mid\!\sim$ to propose a solution for PT in the target domain A. $\mid\!\sim$ is defined as:

$$\begin{array}{lll} Th,\ \mathtt{A}\sim\mathtt{B} & \mid\!\sim_{\mathrm{PT}}\ \varphi(\mathtt{A}) & \\ \quad \text{where } \textbf{Unknown:} & Th \not\models\ \varphi(\mathtt{A}) & \\ \qquad \textbf{Consistent:} & Th \not\models \neg\varphi(\mathtt{A}) & \\ \qquad \textbf{Common:} & Th \models\ \varphi(\mathtt{B}) & \\ \qquad \textbf{Useful:} & Th \cup \{\varphi(\mathtt{A})\} \models PT & \end{array}$$

NLAG begins with a problem statement (e.g., "find the flowrate through the stem of water pipes forming a Y-junction"), an impoverished theory unable to solve the problem, and an analogical hint (e.g., "flowrate is like current"). NLAG then searches through its known set of abstractions which refer to **current** and are able to solve a base rendition of the target problem. The analogy process is primarily concerned with reinstantiating the base abstraction for the target problem, such that it is both consistent and useful.

From a strictly practical level, NLAG is one of the few domain-independent analogical learning systems demonstrated on a series of non-trivial tasks. From a more theoretical level, Greiner's work provides important first-steps in formalizing analogy. He states two important constraints on the analogy process. First, it establishes consistency as a central constraint. Second, it proposes utility as an additional constraint on analogical inference. This is a predecessor to the performance measure of inference adequacy described in Chapter 6.

However, Greiner's work places excessive constraints on the analogy process. First, the **consistent** requirement forces analogical inference to be monotonic. Second, the theory views analogy as a form of instantiation over extensionally defined schemas or abstractions. Thus, much of the answer is given before the process starts. Finally, abstractions must apply without change. This does not take into account the potential adaptation required when moving from one situation or domain to another.

10.2.1.5 Holyoak and Thagard's ACME

Since the first publication of SME (Falkenhainer et al., 1986), Holyoak and Thagard (1988a) have developed ACME, a program which places much of the SME design into a connectionist relaxation framework. Specifically, rather than the three merge steps used in SME, ACME feeds its local match hypotheses into a lateral-inhibition, spreading activation network (figuratively equal to Figure 3.4). This eliminates the potentially expensive merge steps by replacing them with a connectionist relaxation technique that is able to produce a "best" interpretation without explicitly generating all interpretations. By using a relaxation network, their constraints on mapping are not absolute, but rather provide "pressures" that guide the emergence of a global mapping. This is very elegant. On the other hand, they cite SME's development of absolute alternate interpretations as a limitation that ACME's pressures are able to avoid. This conclusion needs further consideration. Unlike SME, ACME's pressures are not governed by global context to ensure that purely local considerations do not produce global incoherency. For example, this "feature" would cause ACME to generate a single interpretation in mapping one necker cube to another, since many–to–many mappings are allowed if there is equal evidence supporting each interpretation.

10.2.2 Explanation and Theory Formation

In addition to analogy, this work is relevant to research in explanation and scientific discovery. The following subsections examine this relationship.

10.2.2.1 Interpretation and qualitative reasoning

PHINEAS is an interpretation system, ascribing plausible causal explanations to an observed behavior. Prior models of causal analysis reason from a fixed set of axioms to provide an explanation of a physical system's behavior. They assume knowledge of all relevant physical processes and the complete structural description required to apply these processes to the current situation.

One such method is ATMI (Forbus, 1986a; DeCoste, 1989).[1] The ATMI method approaches the interpretation problem in a purely forward-going manner: determine everything that could possibly happen given the current physical configuration and then see if the observation corresponds to one of those predictions. This has two problems. First, predicting everything that could possibly happen for a specified physical setting is prohibitive for large problems. *Incremental envisioning*, in which predictions are generated incrementally in response to observation, appears to be a potential solution for this problem (DeCoste, 1989). Second, if the physical configuration is not fully specified, the set of predictions with which to compare will be empty unless assumptions can be made about the scenario. For a forward-going mechanism, this requires *a priori* specification of what may be assumed, which seems untenable if generality is desired. Specifically, the ATMI approach has no mechanism for dictating model selection once assumptions are allowed. Placing PHINEAS on top of an ATMI approach has the appealing characteristic of providing the focus common to backward-going abductive systems, while retaining ATMI's concern for the problems of numeric data analysis and real time processing. PHINEAS is able to focus model selection, identify needed assumptions, and conjecture novel models if the domain theory is incomplete.

10.2.2.2 Scientific discovery

When viewed as research in machine discovery, this work represents a significant deviation from traditional research (Langley, 1981; Langley et al., 1987; Falkenhainer, 1985; Falkenhainer & Michalski, 1986; Rose & Langley, 1986; Zytkow & Simon, 1986), which has focused on the formation of empirical laws characterizing a set of observations. These equations, whether numeric or symbolic, characterize the behavior of a situation rather than explain it. For example, BACON (Langley, 1981) and ABACUS (Falkenhainer, 1985) could "discover" the ideal gas law (PV=nRT) but could not explain it. Likewise, STAHL (Zytkow & Simon, 1986; Rose & Langley, 1986) could not explain how coal is composed of "matter of fire" and ash. One of the drawbacks with these systems is their complete lack of knowledge about the domain, the data variables being manipulated, and prior reasoning experiences. They fail to capture the use of models in scientific investigation, which serve to drive prediction and suggest relevant questions, leading to further investigation.

While ABACUS was an investigation of the knowledge-weak end of the spectrum, PHINEAS is an investigation of the opposite, knowledge-intensive end. In being theory-driven, it is

[1]Here, I do not distinguish between Forbus' ATMI program and Decoste's DATMI program, which each share the features relevant to this discussion.

more in the spirit of recent work by Rajamoney (1988a, 1988b). He examines the problem of revising existing theories in light of new, anomalous observations, using explanation-based and directed-experimentation techniques to prune the space of revision hypotheses. These methods are powerful for identifying consistent revisions, but prune candidates rather than reduce the search space. The analogical approach to theory revision proposed in Chapter 7 and the combined use of analogy and experimentation presented in (Falkenhainer & Rajamoney, 1988) are intended to alleviate this problem.

10.2.2.3 Thagard and Holyoak's PI

The type of theory formation through analogy described in this thesis is an instance of what Thagard (1987) calls *analogical abduction.* While his PI system (Thagard & Holyoak, 1985; Thagard, 1987) addresses the same type of task, our methods are very different. First, PI reasons using rule and schema representations that are substantially less sophisticated than those used for PHINEAS. For example, a rule that waves "reflect" is used to propose that sound might "reflect" because it is a wave (i.e., `Wave(x)` $\Rightarrow$ `Reflect(x)`). Second, the need for potential refinement is not recognized. Furthermore, these schema-based models are not runnable and thus cannot generate predictions at the level needed for empirical confirmation.

10.2.2.4 The Yale SWALE project

SWALE (Kass, 1986; Leake & Owens, 1986; Kass et al., 1986) is a system under development at Yale which uses stored *explanation patterns* to construct explanations of novel events, such as the death of a famous race horse. It is able to adapt prior explanations to novel situations by *tweaking* the schema to fit the specifics of a new case. In this manner, it has many similarities to Kedar-Cabelli's PER model, since both attempt to deductively rederive portions of a prior explanation that do not apply in the new situation. SWALE's replay process is able to achieve a fair degree of sophistication by drawing from a library of tweaking strategies which suggest ways to repair inapplicable portions of a recalled explanation. SWALE does not attempt to address the complete correspondence problem, since object level mappings are unambiguous in the small stories analyzed. Additionally, SWALE is designed as more of a deductive rather than analogical system. For example, it does not address cross-domain analogy issues, such as semantic similarity and forming new relations or objects.

10.2.2.5 Langley and Jones

Around the same time that PHINEAS was first published (Falkenhainer, 1986), Langley & Jones (1986) proposed an architecture sharing its "analogy through behavioral similarity" component. Their proposal was directed at capturing the process of *scientific insight* (e.g., Dreistadt, 1968), in which recognition of a potential analogue is a temporally-extended process consisting of a gradually increasing familiarity with the target concept culminating in sudden recognition. Modeling this aspect of analogy is important and their work represents the only AI research in this area that I know of.

10.3 The Future

This section describes some of the fundamental open research problems encountered during the development of PHINEAS. In addition, a few more realistic short-term extensions and applications of the current work are described.

10.3.1 Analogy

Research on computational analogy is still in its infancy. Below are a number of open problems that seem particularly important for future progress in the field.

10.3.1.1 The access problem

The problem of retrieving a plausibly useful analogue from memory still stands as the least-understood, most important unsolved problem in analogy. To date, most research has been able to avoid the problem, either through tutorial settings in which valuable hints are provided or by restricting work to specialized forms of within-domain analogy. The two-stage mechanism described in this thesis (i.e., use abstractions to focus on candidate set, use structural comparison to prune and order this set) still side-steps important issues. How are these abstractions formed for the stored situations? How are they recognized in the target situation? How are they organized so that an excessive number of analogues are not retrieved?

10.3.1.2 The consistency – coherency problem

In Chapter 2 I claimed that two-valued consistency is overly restrictive as a basis for analogical processing (the *consistency fallacy*). However, no real solution is offered. PHINEAS' task is to do almost whatever it takes to construct an explanation for an observed situation. It has no "I have no idea" mechanism, unless no candidate analogue can be found to initiate the explanation process or no hypothesis can be formed that is consistent with the observation. This is part of the intended design. The best explanation available that is consistent with the observation will be offered, no matter what the "cost". Determining if that "cost" is too great for the explanation to be accepted is considered the province of final acceptance and storage, which are postprocesses external to PHINEAS. An evaluative measure which takes into account the cost of overthrowing prior beliefs for the benefits of a more coherent belief state is needed. Additional factors such as plausibility and specificity in accounting for the phenomenon are required as well.[2]

10.3.1.3 The semantic similarity problem

The definition of *functionally analogous* given in Chapter 2 solved a simplified version of what appears to be a fundamental problem in analogy research – that the semantic similarity of two items depends on their respective roles in the representations being manipulated. Indeed, the entire meaning of these representations may be context-sensitive. Recently, work on *situated meaning* (Agre & Chapman, 1987; Suchman, 1987) has begun to explore these ideas.

10.3.1.4 The compiled knowledge problem

As discussed in Chapter 2, AI systems tend to use compiled knowledge, in which intermediate reasoning steps are absent to promote efficiency of use. This runs counter to the need in analogy to understand the reasons behind why the base domain description is organized the way it is, so that it may be adapted for use in a new target situation. The amount of knowledge required to be a proficient analogizer is greater than the amount of knowledge required to simply perform useful inferences in the original base domain. In PHINEAS, this was addressed by adding a *cache* slot to theories, indicating past reasoning not explicit in the theory's description. Specifically, the cache slot links necessary prerequisites of a process' effects relations to the antecedents that satisfy those prerequisites. However, this is a small fix to a much larger problem. Specifically, a fundamental component of analogy is knowing why the relations being considered for mapping are there. Knowing why prior decisions were made the way they were is important to being able to satisfy the intent of the decision without necessarily adhering to the same decision.[3] Work is needed to specify

[2]Steps in this direction include HYPO (Ashley & Rissland, 1987) and ECHO (Thagard, 1988).
[3]A similar point has been made by Carbonell (1983a) in the context of planning by analogy.

what knowledge is needed, what knowledge should be cached, how that knowledge is to be retrieved, and how that knowledge is to be used in the analogy process.

10.3.1.5 The one-to-one constraint

In the specification of contextual structure-mapping, I adopted Gentner's one-to-one mapping constraint. It is clearly important, in that its complete removal results in nonsense mappings.[4] However, there are cases where relaxing the one-to-one constraint would be very useful. For example, consider an example adapted from (Kedar-Cabelli, 1988), in which a conical styrofoam cup (`scup`) is compared to a metal cup (`mcup`) with a handle, both for the purpose of containing a hot liquid. In the case of the metal cup, a single structural characteristic (i.e., `Has-Handle`) satisfies two functional roles (i.e., `Insulated` and `Grasping-Area`):

Function:	*Styrofoam cup*	*Metal cup*
Insulated:	`(Styrofoam scup)`	`(Has-Handle mcup)`
Grasping-Area:	`(Conical scup)`	`(Has-Handle mcup)`

Thus, a many-to-one mapping is necessary to process this rather obvious analogy. This is a prevalent phenomenon in mechanical design, in which a common goal is to minimize the number of parts by maximizing the multiple functions each part provides. Research is needed to identify when the important constraint provided by the one-to-one requirement should be violated.

10.3.2 Abduction and default reasoning

What is assumable? Work on abduction tends to side-step this issue in several ways. First, probabilistic models (e.g., Szolovits, 1982; Buchanan & Shortliffe, 1984) have the luxury of *a priori* probabilities assigned to antecedent information, allowing the use of probabilistic decision analysis. When addressing open-ended, common-sense problems about the world, having such probabilities seems unrealistic. Second, most work in abduction has been applied to diagnostic problems in which the set of possible diseases or malfunctions is explicitly enumerated (e.g., Buchanan & Shortliffe, 1984; Josephson et al., 1987). An enumerated set of all assumables for the domain of general commonsense physics could conceivably consist of the entire body of knowledge about the domain. What to assume seems dependent on

[4]An experiment was tried using SME with no one-to-one constraints installed. At all times, this results in a single interpretation which is effectively the union of all possible interpretations SME would have generated were one-to-one being enforced.

the particular situation under consideration. Finally, work in interpretation and story understanding (e.g., Charniak, 1988; Mooney, 1987) tends to use schema-based models and identify the assumables as the unknown elements of a relevant schema. The approach seems to work in practice, but the reasons for this have yet to be studied. Most likely, schema representations implicitly package important interrelations between the constituent facts, producing intuitively appealing performance.

In PHINEAS, Charniak's (1988) approach of *assume if consistent* was adopted. While all nonsense assumptions (according to my own personal evaluation) PHINEAS ever made were attributable to gaps in the domain theory, I'm still hesitant to consider it an actual solution. Some form of likelihood estimates seem necessary, though actual probabilities provide their own set of problems.

For example, two alternate answers were possible in the osmosis example described in the previous chapter. These alternatives center around the belief that the membrane separating the two containers is a `Volume-Solid` (i.e., solid all the way through, non-porous, no holes, etc.) in which case the membrane could not be a fluid path.

```
(Volume-Solid ?obj) ⇒ (not (Fluid-Path ?obj))
```

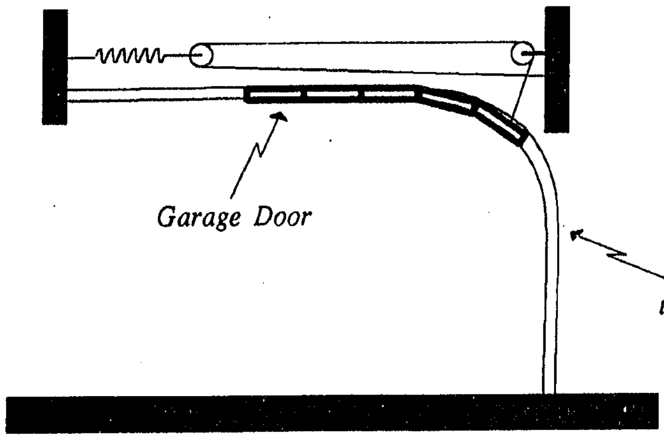
In the previous chapter, PHINEAS was shown to propose that the membrane was a `Fluid-path` by defeating its default assumption

```
(Solid ?obj) ⇒ Assume(Volume-Solid ?obj)
```

However, if the membrane was firmly believed to be a `Volume-Solid`, then PHINEAS would have proposed the existence of an unknown fluid path, (FLUID-PATH SK-PIPE-3), rather than identify the membrane as the suspected path. Whether or not a third alternative, that the membrane is a new kind of `Fluid-Path`, should be one of PHINEAS' options is uncertain. It might lead to a combinatoric explosion in assumptions.

10.3.3 Integrating and refining multiple models

Figure 10.1 shows a sliding garage door in its open, equilibrium position. Told that the door is on a track, the task is to explain its operation and why it oscillates when opened (importantly, the observer does not know about the spring shown in the figure). Explaining the entirety of its behavior was one of the intended, but unrealized goals for PHINEAS. It is a highly desirable example for an analogical explanation system like PHINEAS. First, it requires more sophisticated reasoning than previously described examples. Second, it requires the integration of multiple models about the door's different aspects (e.g., the action



Why does my sliding garage door oscillate up and down when I open it quickly?

Figure 10.1: Sliding garage door in open, equilibrium position.

of both gravity and the hypothesized spring). Finally, it requires substantial adaptation of the relevant models.

Consider the intricacies of the door's behavior. Simply applying a spring-block oscillator model to this system would not produce accurate enough predictions. First, as the door nears its closed position, a point is reached where the closing force of gravity dominates the opening force of the spring. This is because the door's overhang is a function of position. Second, consider what happens as the door oscillates. Moving further up the track moves the door's anchoring line to the left, thus stretching the spring. Moving further down the track pulls the door's anchoring line down (or to the right), thus stretching the spring as well. This configuration produces standard spring-block oscillation, yet does so only by stretching the spring, never compressing it. Using analogy to understand how this garage door operates requires some minimal knowledge of geometry and the ability to refine an initial stretch-compress model into a stretch-stretch model. Even so, it is clear that there is a strong analogy here, showing the utility of analogy for providing an initial model from which to work. Developing a system that can explain the garage-door scenario will require further research in analogy using multiple models (e.g., extending work begun by Burstein (1983)), complex qualitative reasoning and modeling (e.g., Falkenhainer & Forbus, 1988), and spatial reasoning (e.g., Forbus et al., 1987; Joskowicz, 1987; Nielsen, 1988)

10.3.4 Generalizing the model

At the start of this chapter, I expressed the belief that the model PHINEAS embodies has broad utility. This remains to be shown. Adapting the ideas to applications in automated design and planning seems a natural first-step in testing their breadth and utility. Interpretation, design, and planning tasks are of a similar nature: identify a sequence of events that transform some initial state into a given final state (Simmons & Davis, 1987).

There are two prerequisites to constructing the next generation PHINEAS. First, knowledge which is procedurally encoded in the program must be made declarative before task independence can be achieved. This includes the access belief of

(Correlated-to *cause behavior*)

as well as the same correlations assumed during the generation of candidate inferences. Some have already begun to address this problem (Baker et al., 1988; Clark, 1988; Davies & Russell 1987; Russell, 1987). Second, the *adequacy* definition and its corresponding evaluative method must be generalized to include any problem solving activity. For example, a proposed design must actually produce the specified behavior. Note the similarity between this requirement and the adequacy requirement currently used in PHINEAS.

10.3.5 Scientific Discovery

Undoubtedly, a lot of information has been implicitly and unintentionally built into PHINEAS. How else could we explain the relative ease with which it conjectured a caloric, flowlike model of the thermal phenomena given it. For example, PHINEAS lacks the experiential and theoretical breadth and depth that makes science a tedious, complex, and often ambiguous process. Furthermore, the data given PHINEAS is predominately relevant and distilled. Perhaps the initial proposal of the caloric theory was actually this simple once all the relevant observational information was available (e.g., the fact that the temperatures of a hot and cold object placed in contact actually become *equal* was not discovered until the invention of the thermometer (Roller, 1961; Wiser & Carey, 1983)). Unfortunately, we probably will never know. Accurately recreating the knowledge (or lack of) existing immediately prior to the development of the caloric theory would be extremely difficult if not impossible.

This problem is common to all existing work in computational scientific discovery. Famous theories, taking months, years, or centuries to develop are somehow "rediscovered" in a matter of minutes or hours. Despite this obvious problem, much has been and will continue to be learned from such endeavors. However, soon we must attempt "the real thing" – apply these systems to open research problems, or at the very least, problems that may be considered unknown to the program developer. While the field is still too much in its infancy to be making such attempts, the time to tackle real problems is near.

10.3.6 Integrated experiential learning

PHINEAS and all other analogy systems built to date,[5] use analogy as their sole learning method. However, analogy, like any other single learning mechanism, is best viewed as a single component in a synergistic cooperation of learning methods. For example, in scientific investigation, an analogically derived hypothesis may suddenly "come to mind". However, this *flash of insight* may have been preceded by a tedious, incremental process in which data was collected and analyzed, patterns sought, and overall familiarity increased (Dreistadt, 1968; Langley & Jones, 1988). In order to build a general investigative system, we must integrate analogy with directed experimentation, empirical learning, and analytic learning. Some work on developing a general protocol enabling such interaction has already begun (Falkenhainer & Rajamoney, 1988). However, the protocol does not answer the question of how an analogy system may take advantage of the work prior problem solving and trend detection provides. Additionally, it says nothing about how the results of analogy should be integrated into memory. At what point does the generalization process often associated with analogy take place?

10.3.7 Lifespan

I will close this thesis by considering a somewhat loftier, long-term research problem. Eventually, in buiding learning systems that are able to create new concepts and overthrow existing belief structures (i.e., learn at the knowledge-level (Dietterich, 1986)), we will have to address the problem of how knowledge evolves. True knowledge-level learning over an extended period of time is non-monotonic. This leads to the following problem:

- *The insecurity problem:* A learning system operating in a real-world environment and capable of augmenting, restructuring, and doubting its knowledge will be faced with the dilemma of deciding between doubting the validity of what it encounters, doubting the validity of its hypothesis about the encounter, or doubting the correctness of what it believes.

If a new experience leads to an explanatory hypothesis that contradicts existing beliefs, should the system question the observation, its hypothesis, or its prior knowledge? The insecurity problem also emcompasses the *experiential consistency problem* (Rajamoney, 1988b):

[5]An exception is the EBL system developed by Winston *et al* (1983), which used Winston's general ANALOGY program to suggest explanations for explanation-based generalization.

a new hypothesis must be consistent with the entirety of a system's past experiences. This is an untenable requirement in its absolute form. Yet, once exceptions to beliefs are allowed, we are once again faced with the problem of knowing where and when to doubt.

This is a particularly acute problem in an analogical learning system like PHINEAS, which is capable of proposing hypotheses about an observation that are consistent with the observation, but conflict with existing beliefs. PHINEAS has been restricted to emulate an essentially a monotonic learning system. The only beliefs PHINEAS is allowed to overthrow are closed world assumptions about the breadth of its knowledge. If an existing belief is contradicted, a new concept is generated rather than attempt to overthrow that existing belief. That restriction effectively avoids the problem. For example, when considering a slight modification of an existing concept, should the old concept be generalized, modified by addition of disjuncts, or a new concept created? In PHINEAS, a new concept is proposed along with detailed information about how the concept was created. The storage issue concerned with what to do with the proposed concept is viewed as the province of a module external to PHINEAS.

The comfort of maintaining a constant deductive closure is very seductive. As soon as you step beyond the deductive closure, you effectively enter the twilight zone. The reasoning agent can actually question the validity of its own beliefs. Insecurity sets in. This is part of the current gap between machine learning systems and what might be called an intelligent, learning agent. However, this gap is what I believe makes the research interesting.

Appendix A

Note on Abductive Backchaining

As described in Chapter 5, the abductive retrieval task is to find consistent instantiations for a given well-formed formula ρ. If

$$\rho \equiv C_1 \wedge C_2 \wedge \ldots \wedge C_N$$

and only a subset of its elements may be shown, the remaining conjuncts are assumed, contingent on their joint consistency. If ρ contains free variables, this task requires consideration of the N-M binding problem. For example, given the goal to show

$$P(?x) \ \wedge \ Q(?x) \ \wedge \ R(?x)$$

with $P(a)$, $Q(b)$, and $R(b)$ believed true, the abductive retriever should return two possibilities:

	Binding	Assumptions
1:	$(\,?x\ .\ a)$	$Q(a) \wedge R(a)$
2:	$(\,?x\ .\ b)$	$P(b)$

Solving each subgoal sequentially will prevent the second possibility, because the values for $?x$ after solving for $P(?x)$ are limited to $\{a\}$.

This appendix begins by briefly describing the general structure of the abductive retriever used in PHINEAS. It then presents the method by which conjunctive goals are solved.

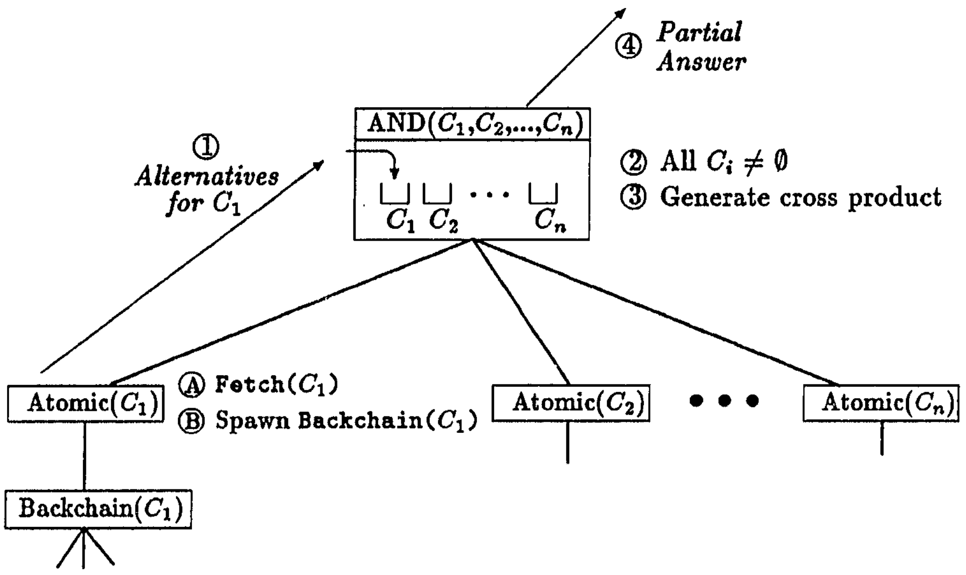


Figure A.1: Monitors manage communication between subgoals and parent goal. (1) Receive a response from a subgoal. (2) Have all the subgoals responded? (3) If so, generate the current set of possible answers. (4) Return those answers to the parent goal.

A.1 Monitors

Each goal is managed by a *monitor*, which coordinates efforts to solve the goal and sends responses to its parent goal each time a new solution is found (Figure A.1). Monitors are of three types, corresponding to the goal managed: *atomic*, *conjunctive*, and *backchain*. Monitor execution is controlled by an agenda, which is an ordered list of monitors corresponding to the leaves of the search tree.[1]

Responses from a subgoal to its parent goal consist of ⟨*answers*, *response-type*⟩ pairs. There are three response types. A `complete` reponse indicates that no further answers are possible beyond the set of answers currently being returned. A `partial` response returns an existing set of answers and indicates that additional responses may be forthcoming. For example, a goal to show an atomic formula will both perform a fetch for existing instances, which are returned through a `partial` response, and spawn a backward chaining subgoal to seek additional possibilities. Finally, a `fail` response indicates that no answers have

[1]Monitors are placed in order of increasing depth, producing breadth-first search. A more thorough treatment would take into consideration the plausibility of assumptions, complexity and specificity of the explanation, and the current success of alternate explanations.

been found and will never be found. For example, a backchaining monitor will immediately send a `fail` response if there are no applicable rules. A monitor responding with `complete` or `fail` is considered *closed.* For example, an atomic goal will close with a response of `complete` if its initial fetch for existing instances is successful and its backchaining subgoal fails.

A.2 Conjunctive Goals

The retrieval task may be reformulated in terms of the free variables $\{v_1, v_2, \ldots, v_m\}$ contained in ρ: find a substitution θ which associates each v_i to a known entity such that $\rho\theta$ is believed consistent in some ATMS context. In this view, each retrieved instantiation C_{i_j} of conjunct C_i supplies new possible bindings for the free variables it contains. Each possible value for a free variable is implicitly represented as an ATMS node corresponding to the assignment $?v_i = value_j$ and is supported by the instantiated nodes using that value for $?v_i$.[2] For example, if the task is to retrieve

$P(?x, ?y) \wedge Q(?y, ?z)$

then retrieval of $P(3,4)$, $P(3,5)$, and $Q(4,7)$ produces the following justifications

$P(3,4) \rightarrow\ ?x=3$
$P(3,4) \rightarrow\ ?y=4$
$P(3,5) \rightarrow\ ?x=3$
$P(3,5) \rightarrow\ ?y=5$
$Q(4,7) \rightarrow\ ?y=4$
$Q(4,7) \rightarrow\ ?z=7$

With this representation, the retrieval task is to find all consistent bindings containing exactly one selection from each variable's choice set:

```
?v1: (( ?v1=value11)
      ( ?v1=value12)
      ...)
```

[2]It is an *implicit* ATMS node in that the node is never actually created. Rather, its *label* (all minimal sets of assumptions under which the node is believed) is computed and stored separately by the abductive retriever. This is a tremendous efficiency savings and eliminates the total contradiction resulting from absolute support for two alternate values of $?v_i$ combined with justifications that would be required to enforce a unique value for each $?v_i$.

$?v_2$: (($?v_2{=}value_{2_1}$)
($?v_2{=}value_{2_2}$)
...)

...

$?v_m$: (($?v_m{=}value_{m_1}$)
($?v_m{=}value_{m_2}$)
...)

There are two senses of consistency relevant to this operation. First, unification consistency of the bindings must be maintained. This is assured by the unification procedure used to retrieve the instantiated expressions and the criterion that only one selection from each variable's choice set is allowed. Second, each instantiation for the conjunction ρ must be consistent with existing knowledge.

The algorithm for solving a conjunctive goal is shown in Table A.1. The task begins by activating a subgoal for each conjunct C_i, followed by a period of waiting for an initial response from each subgoal. As subgoals respond, the stored set of possible instantiations for ρ's free variables is updated. Once each subgoal has responded, the conjunctive goal's monitor computes all possible substitutions θ_i from the existing choice sets. Only those substitutions supported by an existing ATMS context are retained. This is determined by computing the conjunction of the individual binding nodes ($?v_i{=}value_j$) contained in a given θ_i and verifying that this conjunction is believed in some context.[3] As stated, this has the possibility to form an instantiation for ρ in which none of the conjuncts are believed. For example, given the goal

$$P(?x,?y) \ \wedge \ Q(?x,?y)$$

Finding $P(a,b)$ supports assumption of $Q(a,b)$ and finding $Q(c,d)$ supports assumption of $P(c,d)$. Together, these candidates suggest the instantiation $P(a,d) \wedge Q(a,d)$. Such unsupported candidate instantiations are rejected.

Note that lack of an ATMS context in which `(Consistent` $\rho\theta$`)` is believed does not imply $\rho\theta$ is inherently inconsistent. It simply means that existing assumptions supporting the proposed instantiation of ρ are mutually inconsistent. Alternate, consistent assumptions supporting the same instantiation are possible at some future time. However, assuming

[3]The ATMS *label* for a node ($?v_i{=}value_j$) indicates all minimal sets of assumptions under which the node is believed. The existence of a context supporting a given binding set is verified by computing the label for the conjunction of all nodes ($?v_i{=}value_j$) supporting the binding. If the label is non-empty, there is at least one context in which $\rho\theta$ is believed.

Table A.1: Abductive retrieval for conjunctive goals.

```
1. Spawn an atomic subgoal on each conjunct $\{C_1, \ldots, C_n\}$
2. Wait for initial responses
   Repeat Until all conjuncts have responded:
     (a) Receive response from $C_i$
     (b) If response type = fail and
         $C_i$ is closable via closed world assumption or $C_i$ is the only conjunct containing variable
         $v_j$,
         return fail.
     (c) Store suggested bindings from $C_i$
3. Generate bindings cross product
   If there are binding sets and every $C_i$ task is closed
   Respond(binding-sets, complete)
   If there are binding sets and there is a $C_i$ task that is not closed
   Respond(binding-sets, partial)
   If there are no binding sets and every $C_i$ task is closed
   Respond(NIL, fail)
4. Extend existing possibilities
   Repeat Until every $C_i$ task is closed
     (a) Receive response from $C_i$
     (b) Store suggested bindings from $C_i$
     (c) Extend existing binding cross product
         If there are extensions and every $C_i$ task is closed
         Respond(binding-sets, complete)
         If there are extensions and there is a $C_i$ task that is not closed
         Respond(binding-sets, partial)
         If there are no extensions and every $C_i$ task is closed
         Respond(NIL, complete)
```

$\rho\theta$ in the hope of future consistency is deemed too weak as a basis for making abductive assumptions.

Appendix B

Contextual Structure Mapping Rule Set

Contextual structure mapping is modeled by the *CSM* rule set given to SME, which defines SME_{CSM}. This single rule set was used without change in every PHINEAS example described in this thesis. Chapter 3 described these rules and gave the motivations behind each. Here, the complete set of lisp rules used in SME_{CSM} are listed. With each rule is an indication of the predicate calculus rule(s) from Chapter 3 it implements. The name and numbers for each rule refer to the rule's description in Chapter 3.

B.1 Step 1: Local Match Construction

Given two dgroups, SME begins by finding potential matches between items in the base and target. Allowable matches are specified by *match constructor* rules, which take the form:

(MHCrule (⟨*Trigger*⟩ ⟨*BaseVariable*⟩ ⟨*TargetVariable*⟩ [:test ⟨*TestForm*⟩])
    ⟨*Body*⟩)

In all match constructor rules, ⟨*Body*⟩ is executed in an environment in which ⟨*BaseVariable*⟩ and ⟨*TargetVariable*⟩ are bound to items from the base and target dgroups, respectively. If ⟨*TestForm*⟩ is present, the bindings must satisfy the test (i.e., the form when evaluated must return non-NIL). There are two possible values for ⟨*TestForm*⟩. A :filter trigger indicates that the rule is applied to each pair of items from the base and target. These rules create an initial set of match hypotheses between individual base and target expressions. An :intern trigger indicates that the rule should be applied to each newly created match hypothesis, binding the variables to its base and target items. These rules create additional matches suggested by the given match hypothesis.

;;; Rule 1 (Same Functors)

```
(MHC-rule (:filter ?b ?t :test (and (eq (expression-functor ?b) (expression-functor ?t))
                                    (not (paired-item? :base-item (expression-functor ?b)))
                                    (not (paired-item? :target-item (expression-functor ?t)))))
    (install-MH ?b ?t))
```

;;; Rule 2 (Common Generalization)

```
(MHC-rule (:filter ?b ?t :test (and (not (paired-item? :base-item (expression-functor ?b)))
                                    (not (paired-item? :target-item
                                                        (expression-functor ?t)))
                                    (not (function? (expression-functor ?b)))
                                    (not (function? (expression-functor ?t)))
                                    (predicate-type-intersection? (expression-functor ?b)
                                                                  (expression-functor ?t))))
    (install-MH ?b ?t))
```

;;; Rule 3 (Functionally Analogous)

```
(MHC-rule (:intern ?b ?t :test (and (implicational? ?b) (implicational? ?t)))
    (when (and (eq (expression-functor (implicational-consequent ?b))
                   (expression-functor (implicational-consequent ?t)))
               ;; Implying a logical connective doesn't count
               (not (logical? (expression-functor (implicational-consequent ?b))))
               ;; A logical connective implying something doesn't count
               (not (logical? (expression-functor (implicational-antecedent ?b)))))
          ;; If atomic consequents match, then match atomic antecedents
          (install-MH (implicational-antecedent ?b) (implicational-antecedent ?t))
          (sme:assert! '(function-of ,(implicational-antecedent ?b)
                                     (supports ,(implicational-consequent ?b))))
          (sme:assert! '(provides-function ,(implicational-antecedent ?t)
                                           (supports ,(implicational-consequent ?t))))))
```

;;; Rule 4 (Sanctioned Pairing)

```
(MHC-rule (:filter ?b ?t :test (sanctioned-pairing? (expression-functor ?b)
                                                    (expression-functor ?t)))
    (install-MH ?b ?t))
```

;;; Rules 5-6 (Non-Commutative Corresponding Arguments)

```
(MHC-rule (:intern ?b ?t :test (and (expression? ?b) (expression? ?t)
                                    (not (commutative? (expression-functor ?b)))
                                    (not (commutative? (expression-functor ?t)))))
    (do ((bchildren (expression-arguments ?b) (cdr bchildren))
         (tchildren (expression-arguments ?t) (cdr tchildren)))
        ((or (null bchildren) (null tchildren)))
      (cond ((and (entity? (first bchildren)) (entity? (first tchildren))
                 (or (sanctioned-pairing? (entity-name (first bchildren))
                                          (entity-name (first tchildren)))
                     (and (not (paired-item? :base-item
                                             (entity-name (first bchildren))))
                          (not (paired-item? :target-item
                                             (entity-name (first tchildren)))))))
             (install-MH (first bchildren) (first tchildren)))
            ((and (function? (expression-functor (first bchildren)))
                 (function? (expression-functor (first tchildren)))
                 (not (paired-item? :base-item (expression-functor (first bchildren))))
                 (not (paired-item? :target-item (expression-functor (first tchildren)))))
             (install-MH (first bchildren) (first tchildren))))))
```

;;; Rules 7-8 (Commutative Corresponding Arguments)

```
(MHC-rule (:intern ?b ?t :test (and (expression? ?b) (expression? ?t)
                                    (commutative? (expression-functor ?b))
                                    (commutative? (expression-functor ?t))))
    (dolist (bchild (expression-arguments ?b))
      (dolist (tchild (expression-arguments ?t))
        (cond ((and (entity? bchild) (entity? tchild)
                    (or (sanctioned-pairing? (entity-name bchild)
                                             (entity-name tchild))
                        (not (paired-item? :base-item (entity-name bchild)))
                        (not (paired-item? :target-item (entity-name tchild)))))
               (install-MH bchild tchild))
              ((and (function? (expression-functor bchild))
                    (function? (expression-functor tchild))
                    (not (paired-item? :base-item (expression-functor bchild)))
                    (not (paired-item? :target-item (expression-functor tchild))))
               (install-MH bchild tchild))))))
```

B.2 Step 2: Global Match Construction

Once an initial set of match hypotheses is formed, the pairwise consistency of match hypotheses stated by *Conflicting*($MH(b_i, t_j)$) is used to combine them into maximal, consistent gmaps. Rules are used to define the contents of *Conflicting* for each match hypothesis and are executed by the pattern-directed rule engine attached to the BMS. Their syntax is slighly different than those described above. Rather than being assigned specific base and target items, each trigger contains a pattern which must unify with a known BMS datum.

;;; Rules 9-12 (One-to-one constraints)

```
;;; one-to-one (expressions & entities) base case
(rule ((:intern (MH ?b ?t1))
       (:intern (MH ?b ?t2) :test (not (eq ?t1 ?t2))))
   (Conflicting (fetch-mh ?b ?t1) (fetch-mh ?b ?t2)))
```

```
;;; one-to-one (expressions & entities) target case
(rule ((:intern (MH ?b1 ?t))
       (:intern (MH ?b2 ?t) :test (not (eq ?b1 ?b2))))
   (Conflicting (fetch-mh ?b1 ?t) (fetch-mh ?b2 ?t)))

;;; one-to-one (predicates) base case
(rule ((:intern (MH ?b1 ?t1) :test (and (expression? ?b1) (expression? ?t1)))
       (:intern (MH ?b2 ?t2) :test (and (expression? ?b2) (expression? ?t2)
                                        (eq (expression-functor ?b1)
                                            (expression-functor ?b2))
                                        (not (eq (expression-functor ?t1)
                                                 (expression-functor ?t2))))))
   (Conflicting (fetch-mh ?b1 ?t1) (fetch-mh ?b2 ?t2)))

;;; one-to-one (predicates) target case
(rule ((:intern (MH ?b1 ?t1) :test (and (expression? ?b1) (expression? ?t1)))
       (:intern (MH ?b2 ?t2) :test (and (expression? ?b2) (expression? ?t2)
                                        (not (eq (expression-functor ?b1)
                                                 (expression-functor ?b2)))
                                        (eq (expression-functor ?t1)
                                            (expression-functor ?t2)))))
   (Conflicting (fetch-mh ?b1 ?t1) (fetch-mh ?b2 ?t2)))
```

;;; Rule 13 (Temporal preservation)

```
(rule ((:intern (MH ?b1 ?t1) :test (and (temporally-scoped? ?b1) (temporally-scoped? ?t1)))
       (:intern (MH ?b2 ?t2) :test (and (temporally-scoped? ?b2) (temporally-scoped? ?t2))))
   (unless (or (and (same-time? ?b1 ?b2) (same-time? ?t1 ?t2))              ;both same
               (and (different-time? ?b1 ?b2) (different-time? ?t1 ?t2)))   ;both different
     (Conflicting (fetch-mh ?b1 ?t1) (fetch-mh ?b2 ?t2))))
```

;;; Rule 14 (Compound object rearrangement: contained-liquid case)

```
(rule ((:intern (MH ?b1 ?t1) :test (and (contained-liquid? ?b1 :base-item)
                                        (contained-liquid? ?t1 :target-item)))
       (:intern (MH ?b2 ?t2) :test (or (and (container-of? ?b1 ?b2 :base-item)
                                            (not (container-of? ?t1 ?t2 :target-item)))
                                       (and (not (container-of? ?b1 ?b2 :target-item))
                                            (container-of? ?t1 ?t2 :base-item)))))
     (Conflicting (fetch-mh ?b1 ?t1) (fetch-mh ?b2 ?t2)))
```

B.3 Step 4: Compute evaluation scores

Once the gmaps have been formed and their candidate inferences determined, SME concludes by assigning each gmap a match evaluation score. Managed by the BMS, these rules supply evidence through the form Implies(⟨*antecedent*⟩,⟨*consequent*⟩,⟨*weight*⟩).

;;; Rule 15 (Minimal Ascension)

```
(initial-assertion (sme:assert! 'type-match))

(rule ((:intern (MH ?b ?t) :test (and (expression? ?b) (expression? ?t)
                                      (not (paired-item? :base-item (expression-functor ?b)))
                                      (not (paired-item? :target-item
                                                         (expression-functor ?t))))))
  (let ((pl (map-path-length (expression-functor ?b) (expression-functor ?t))))
    (when (and (numberp pl) (> pl 0))
      (sme:assert! '(implies type-match (MH ,?b ,?t) (,(/ 0.4 pl) . 0.0))))))
```

;;; Rule 16 (Sanctioned Pairing Evidence)

```
(initial-assertion (sme:assert! 'sanctioned-pairing))

(rule ((:intern (MH ?b ?t) :test (and (expression? ?b) (expression? ?t)
                                      (sanctioned-pairing? (expression-functor ?b)
                                                           (expression-functor ?t)))))
      (sme:rassert! (implies sanctioned-pairing (MH ?b ?t) (0.4 . 0.0))))

(rule ((:intern (MH ?b ?t) :test (and (entity? ?b) (entity? ?t)
                                      (sanctioned-pairing? (entity-name ?b)
                                                           (entity-name ?t)))))
      (sme:rassert! (implies sanctioned-pairing (MH ?b ?t) (0.4 . 0.0))))
```

;;; Rule 17 (Functionally Analogous Evidence)

```
(initial-assertion (sme:assert! 'same-role))

(rule ((:intern (MH ?b ?t) :test (and (expression? ?b) (expression? ?t)))
       (:intern (function-of ?b ?f) :var ?f-of)
       (:intern (provides-function ?t ?f) :var ?p-f))
   (sme:rassert! (implies (and ?f-of ?p-f) (MH ?b ?t) (0.8 . 0.0))))
```

;;; Rules 18-19 (Systematicity)

```
;;; Non-commutative case
(rule ((:intern (MH ?b1 ?t1) :test (and (expression? ?b1) (expression? ?t1)
                                        (not (commutative? (expression-functor ?b1)))))
       (:intern (MH ?b2 ?t2) :test (children-of? ?b2 ?t2 ?b1 ?t1)))
    (sme:rassert! (implies (MH ?b1 ?t1) (MH ?b2 ?t2) (0.8 . 0.0))))

;;; Commutative case
(rule ((:intern (MH ?b1 ?t1) :test (and (expression? ?b1) (expression? ?t1)
                                        (commutative? (expression-functor ?b1))))
       (:intern (MH ?b2 ?t2) :test (and (member ?b2 (expression-arguments ?b1) :test #'eq)
                                        (member ?t2 (expression-arguments ?t1) :test #'eq))))
    (sme:rassert! (implies (MH ?b1 ?t1) (MH ?b2 ?t2) (0.8 . 0.0))))
```

;;; Rule 20 (Behavioral)

```
(initial-assertion (sme:assert! 'behavioral))

(rule ((:intern (MH ?b ?t) :test (and (expression? ?b) (expression? ?t)
                                      (behavioral-relation? ?b) (behavioral-relation? ?t))))
    (sme:rassert! (implies behavioral (MH ?b ?t) (0.4 . 0.0))))
```

;;; Rule 21 (Provides Relevant Inference)

```
(rule ((:intern (CI ?gmap (B-Explains ?theory ?behavior))
                :test (Current-observation ?behavior)))
    (setf (getf (gm-plist ?gmap) :relevant?) T))   ;handle evidence in gmap rule
```

One final rule is needed to sum all the evidence for each gmap. For efficiency, the sum is computed directly rather than going through the BMS justification mechanism.

```
(rule ((:intern (GMAP ?gm)))
    (setf (node-belief+ (gm-bms-node ?gm)) 0)
    (dolist (mh (gm-elements ?gm))
        (incf  (node-belief+ (gm-bms-node ?gm))  (node-belief+ (mh-bms-node mh))))
    (if (getf (gm-plist ?gm) :relevant?)
        (incf (node-belief+ (gm-bms-node ?gm)) 0.9)))
```

Appendix C

Phineas Initial Domain Knowledge

PHINEAS begins each explanation task with a collection of qualitative theories about physical processes, entities, and general physical rules. Except where indicated in Chapter 9, the initial set of domain knowledge was the same for all examples described in this thesis and represents the contents of this section. A few special cases required additions or deletions to the basic set. For example, all knowledge concerning heat phenomena was removed for the caloric heat flow task.

All domain knowledge is managed by PHINEAS' ATMS problem solver. This consists of a pattern-directed rule engine and the ATMS itself. Because QPE requires access to the same domain knowledge that PHINEAS possesses, all rule and process declarations are switchable, in that they may expand into either a PHINEAS declaration or a QPE declaration. This is indicated by the asterik in the form *name** and is controlled by a global flag, `*qp-toggle-switch*`.

C.1 Rules

PHINEAS rules appear in two forms. The `rule*` construct indicates a forward-chaining rule which always runs whenever possible. These are used strictly for declaring inconsistencies (e.g., `(gas ?x)` is inconsistent with `(liquid ?x)`). The (`<==*` *consequent antecedents*) form expands into two rules. One is an exhaustive, forward chaining rule (i.e., `rule*`). The other is stored for backward chaining purposes by PHINEAS' abductive retriever.

;;; Closed World Assumptions

```
;;; Assume all spatial relationships are subject to closed world assumption
(mem:defClosable Touching)
(mem:defClosable Physical-Path)
(mem:defClosable Immersed-in)
```

```
(mem:defClosable Spring)
(mem:defClosable Block)
(mem:defClosable Beaker)
```

;;; Phase

```
;;; Must be in only one state at a time
(rule* :intern ((solid ?obj))  (mem:at-most-one '((solid ,?obj) (liquid ,?obj) (gas ,?obj))))
(rule* :intern ((liquid ?obj)) (mem:at-most-one '((solid ,?obj) (liquid ,?obj) (gas ,?obj))))
(rule* :intern ((gas ?obj))    (mem:at-most-one '((solid ,?obj) (liquid ,?obj) (gas ,?obj))))

(<==* (not (Fluid ?fl))  ((Solid ?fl)))
(<==* (not (Solid ?fl))  ((Fluid ?fl)))
(<==* (Fluid ?fl)  ((Gas ?fl)))
(<==* (Fluid ?fl)  ((Liquid ?fl)))
```

;;; A QPE specific rule

```
(adb:rule :intern ((((process-instance ?name) ?inst) . :TRUE))
    (adb:rassert! ((process-instance-of ?name ?inst) . :TRUE)))
```

;;; Miscellaneous nogoods...

```
(rule* :intern ((container-of ?obj ?obj))            ;something can't contain itself
   (mem:rassert! ((container-of ?obj ?obj) . :FALSE)))

;;; The inequality taxomony
(mem:rule :intern (((greater-than ?n1 ?n2) . ?cond))
     (mem:taxonomy (list (list 'greater-than ?n1 ?n2)
                         (list 'equal-to ?n1 ?n2)
                         (list 'less-than ?n1 ?n2))))
(mem:rule :intern (((equal-to ?n1 ?n2) . ?cond))
     (mem:referent '((greater-than ,?n1 ,?n2) . ?cond)))
(mem:rule :intern (((less-than ?n1 ?n2) . ?cond))
     (mem:referent '((greater-than ,?n1 ,?n2) . ?cond)))

(mem:rule :intern (((greater-than ?n1 ?n2) . ?cond))
     (mem:rjustify ((less-than ?n2 ?n1) . ?cond)
                   (((greater-than ?n1 ?n2) . ?cond))))

(mem:rule :intern (((greater-than ?n1 ?n2) . :TRUE))
     (mem:rjustify ((greater-than ?n2 ?n1) . :FALSE)
                   (((greater-than ?n1 ?n2) . :TRUE))))
```

;;; Containment...

```
(<==* (not (Can-absorb ?solid ?fl))
      ((Fluid ?fl) (Solid ?solid) (not (Porous ?solid))))

(<==* (not (Can-contain ?container ?fl))
      ((Fluid ?fl) (Volume-Solid ?container)))

(rule* :intern (((can-contain ?n1 ?n1) . ?cond))   ;something can't contain itself
   (mem:rassert! ((can-contain ?n1 ?n1) . :FALSE)))
```

```
;;; Spatial relationships...

(<==* (Touching ?obj1 ?obj2)
      ((Immersed-in ?obj1 ?obj2)))

(<==* (Touching ?cl atmosphere)
      ((Contained-Liquid ?cl)
       (Container-of ?cl ?container)
       (Open ?container)))

(<==* (Greater-than (A (contact-surface ?obj1 ?obj2)) zero)
      ((Touching ?obj1 ?obj2)))

(<==* (Quantity (contact-surface ?obj1 ?obj2))
      ((Touching ?obj1 ?obj2)))

;;; We need to do the next one twice because QPE needs to go through its macro expander
(mem:<== (Qprop (contact-surface ?obj1 ?obj2) (amount-of ?obj1))
         ((Immersed-in ?obj1 ?obj2)))

(adb:rule :intern ((immersed-in ?obj1 ?obj2))
    (install-runtime-qprop '(contact-surface ,?obj1 ,?obj2)
                           '(Qprop (contact-surface ,?obj1 ,?obj2) (amount-of ,?obj1))
                           '(((immersed-in ,?obj1 ,?obj2) . :TRUE))))

;;; Paths...

(rule* :intern (((Physical-Path ?obj1 ?obj2 (Common-Face ?objA ?objB)) . :TRUE))
   (unless (and (or (equal ?obj1 ?objA) (equal ?obj1 ?objB))
                (or (equal ?obj2 ?objA) (equal ?obj2 ?objB))
                (not (equal ?objA ?objB)))
     (mem:rassert! ((Physical-Path ?obj1 ?obj2 (Common-Face ?objA ?objB)) . :FALSE))))

(rule* :intern (((Physical-Path ?obj ?obj ?path) . :TRUE))
     (mem:rassert! ((Physical-Path ?obj ?obj ?path) . :FALSE)))

(rule* :intern (((Container-of ?obj ?can1) . :TRUE)
                ((Container-of ?obj ?can2) . :TRUE) :test (not (equal ?can1 ?can2)))
   (mem:at-most-one '((Container-of ,?obj ,?can1)(Container-of ,?obj ,?can2))))

(<==* (Physical-Path ?obj1 ?obj2 (Common-Face ?obj1 ?obj2))
      ((Touching ?obj1 ?obj2)))

(<==* (Touching ?obj2 ?obj1)  ((Touching ?obj1 ?obj2)))

(rule* :intern (((Touching ?obj1 ?path) . :TRUE)
                ((Touching ?path ?obj2) . :TRUE) :test (not (equal ?obj1 ?obj2)))
    (mem:rjustify ((Physical-Path ?obj1 ?obj2 ?path) . :TRUE)
                  (((Touching ?obj1 ?path) . :TRUE)
                   ((Touching ?path ?obj2) . :TRUE))))

(rule* :intern (((Solid ?obj) . :TRUE))
   (mem:rassume! ((Volume-Solid ?obj) . :TRUE) :DEFAULT-ON-SOLID))
```

```
(<==* (not (Fluid-Path ?obj))  ((Volume-Solid ?obj)))
(<==* (not (Fluid-Path ?obj))  ((Gas ?obj)))
(<==* (not (Fluid-Path ?obj))  ((Liquid ?obj)))

(<==* (Fluid-Path (Common-Face ?obj2 ?obj1)) ((Fluid-Path (Common-Face ?obj1 ?obj2))))

(<==* (Aligned (Common-Face ?obj1 ?obj2))  ((Touching ?obj1 ?obj2)))

(<==* (Supports-Flow (Common-Face ?obj1 ?obj2))  ((Touching ?obj1 ?obj2)))

;;; Miscellaneous object properties...

(<==* (heat-path ?obj)  ((beaker ?obj)))
(<==* (can-contain ?obj alcohol)  ((beaker ?obj)))

;;; Linear and angular spatial quantities...

(rule* :intern ((linear-q ?q))
   (mem:at-most-one '((linear-q ,?q) (angular-q ,?q))))
(rule* :intern ((angular-q ?q))
   (mem:at-most-one '((linear-q ,?q) (angular-q ,?q))))

(<==* (Linear ?dq)  ((Derivative-of (?q ?obj) (?dq ?obj))
                     (Linear ?q)))
(<==* (Linear ?q)  ((Derivative-of (?q ?obj) (?dq ?obj))
                    (Linear ?dq)))
(<==* (Angular-q ?dq)  ((Derivative-of (?q ?obj) (?dq ?obj))
                        (Angular-q ?q)))
(<==* (Angular-q ?q)  ((Derivative-of (?q ?obj) (?dq ?obj))
                       (Angular-q ?dq)))

(<==* (Linear ?acc)  ((Force-Application (?force ?src) (?acc ?dst))
                      (Linear ?force)))
(<==* (Angular ?acc) ((Force-Application (?force ?src) (?acc ?dst))
                      (Angular ?force)))

;;; Defining quantity types for QPE...

(defQuantity-Type Amount-of Individual)
(defQuantity-Type Pressure Individual)
(defQuantity-Type Pressure-in Individual)
(defQuantity-Type Flow-Rate Individual)
(defQuantity-Type Heat-Flow-Rate Individual)

(defQuantity-Type Heat Individual)
(defQuantity-Type Temperature Individual)

(defQuantity-Type Change-Rate Individual)
(defQuantity-Type Vaporization-Rate Individual)
(defQuantity-Type Dissolve-Rate Individual)
(defQuantity-Type Surface-Area Individual)
(defQuantity-Type Contact-Surface Individual Individual)
```

```
(defQuantity-Type Concentration Individual)
(defQuantity-Type Saturation-Point Individual)

(defQuantity-Type Temperature-in Individual)
(defQuantity-Type Tboil Individual)
(defQuantity-Type Tfreeze Individual)

(defQuantity-Type Force Individual Individual)
(defQuantity-Type Internal-Force Individual)
(defQuantity-Type Torque Individual)
(defQuantity-Type Restorative-Force Individual)
(defQuantity-Type Length Individual)
(defQuantity-Type Rest-Length Individual)
(defQuantity-Type Displacement Individual)
(defQuantity-Type Angular-Displacement Individual)

(defQuantity-Type Position Individual)
(defQuantity-Type Angle Individual)
(defQuantity-Type Velocity Individual)
(defQuantity-Type Angular-Velocity Individual)
(defQuantity-Type Acceleration Individual)
(defQuantity-Type Angular-Acceleration Individual)

(defQuantity-Type Total-Energy Individual)
(defQuantity-Type Potential-Energy Individual)
(defQuantity-Type Kinetic-Energy Individual)

(defQuantity-Type Charge Individual)
(defQuantity-Type Current Individual)
(defQuantity-Type Voltage Individual)

(defInfluence I+)
(defInfluence I-)
(defInfluence Ctrans)
```

C.2 Entities

In QP theory, entities are defined using the form `(defEntity` *header body*`)`. This defines a predicate schema in which belief in the header form implies belief in the facts contained in the body.

```
(assert* '((contained-gas atmosphere) . :TRUE))

(assert* '((liquid alcohol) . :TRUE))

(assert* '((substance water) . :TRUE))
(assert* '((substance alcohol) . :TRUE))
(assert* '((substance milk) . :TRUE))
(assert* '((substance fuel) . :TRUE))
```

```
(assert* '((substance air) . :TRUE))

(defEntity* (Rotating-Object ?obj)
   (Quantity (Angle ?obj))
   (Quantity (Angular-velocity ?obj))
   (Quantity (Angular-acceleration ?obj))
   (Derivative-of (Angle ?obj) (Angular-Velocity ?obj))
   (Derivative-of (Angular-Velocity ?obj) (Angular-Acceleration ?obj))
   (Quantity (Kinetic-Energy ?obj))
   (not (less-than (A (Kinetic-Energy ?obj)) zero))
   (Q= (Kinetic-Energy ?obj) (* (Angular-Velocity ?obj) (Angular-Velocity ?obj))))

(defEntity* (Translating-Object ?obj)
   (Quantity (Position ?obj))
   (Quantity (Velocity ?obj))
   (Quantity (Acceleration ?obj))
   (Derivative-of (Position ?obj) (Velocity ?obj))
   (Derivative-of (Velocity ?obj) (Acceleration ?obj))
   (Quantity (Kinetic-Energy ?obj))
   (not (less-than (A (Kinetic-Energy ?obj)) zero))
   (Q= (Kinetic-Energy ?obj) (* (Velocity ?obj) (Velocity ?obj))))

(defEntity* Thermal-Object
   (Quantity (Temperature ?self))
   (Quantity (Heat ?self))
   (Quantity (Tboil ?self))
   (Quantity (Tfreeze ?self))
   (Greater-Than (A (Tboil ?self)) (A (Tfreeze ?self)))
   (Greater-Than (A (Tfreeze ?self)) zero)
   (Greater-Than (A (Heat ?self)) zero)
   (Greater-Than (A (Temperature ?self)) zero)        ;Kelvin
   (Qprop (Temperature ?self) (Heat ?self)))

(defEntity* Physob
   (Quantity (Amount-of ?self))
   (Quantity (Surface-area ?self))
   (not (less-than (A (Amount-of ?self)) zero))
   (not (less-than (A (Surface-area ?self)) zero))
   (Qprop (surface-area ?self) (amount-of ?self))
   (Thermal-Object ?self))

(defEntity* Contained-Liquid
   (Physob ?self)
   (Liquid ?self)
   (State-of ?self liquid))

(<==* (Contained-Fluid ?cl ?sub ?container)
      ((Contained-Liquid ?cl)
       (Container-of ?cl ?container)
       (Substance-of ?cl ?sub)))

(defEntity* Contained-Gas
     (Physob ?self)
     (Gas ?self)
     (State-of ?self gas))

(<==* (Contained-Fluid ?cg ?sub ?container)
      ((Contained-Gas ?cg)
       (Container-of ?cg ?container)
       (Substance-of ?cg ?sub)))

(defEntity* (Contained-Fluid ?cs ?sub ?container)
     (Physob ?cs)
     (Container-of ?cs ?container)
     (Substance-of ?cs ?sub)
     (Quantity (Amount-of ?cs))
     (Quantity (Pressure-in ?container)))

(adb:rule :intern ((contained-fluid ?cs ?sub ?container)
                   (liquid ?cs))
    (install-runtime-qprop '(Pressure-in ,?container)
                           '(Qprop (Pressure-in ,?container) (amount-of ,?cs))
                           '(((contained-fluid ,?cs ,?sub ,?container) . :TRUE)
                             ((liquid ,?cs) . :TRUE))))

(defEntity* (Solution ?solution)
     (Quantity (Concentration ?solution))
     (Quantity (Saturation-Point ?solution))
     (not (less-than (A (Concentration ?solution)) zero))
     (not (less-than (A (Saturation-Point ?solution)) zero)))

(defentity* (Spring ?spring)
  (Quantity (Displacement ?spring))
  (Quantity (Restorative-Force ?spring))
  (Qprop- (Restorative-Force ?spring) (Displacement ?spring))
  (Correspondence (pair (A (Restorative-force ?spring)) ZERO)
                  (pair (A (Displacement ?spring)) zero))
  (Quantity (Potential-energy ?spring))
  (not (Less-than (A (Potential-Energy ?spring)) zero))
  (Q= (Potential-Energy ?spring) (* (Displacement ?spring) (Displacement ?spring))))

(defEntity* (Spring-Mass-System ?system ?spring ?mass)
   (Connected ?mass ?spring)
   (Quantity (Total-Energy ?system))
   (not (less-than (A (Total-Energy ?system)) zero))
   (Equal-to (D (Total-Energy ?system)) zero)
   (Q= (Total-Energy ?system) (+ (Kinetic-Energy ?mass) (Potential-Energy ?spring)))
   (Q= (Displacement ?spring) (Position ?mass))
   (Force-Application (Restorative-Force ?spring) (Acceleration ?mass)))

(defView* (Applied-Force ?src ?dst)
    Individuals ((?src :conditions (Quantity (?force ?src)))
                 (?dst :conditions (Quantity (?acc ?dst))
                                   (Force-Application (?force ?src) (?acc ?dst))))
```

```
Relations ((Q= (?acc ?dst) (?force ?src)))
```

C.3 Processes

Processes are defined using the form

```
(defProcess (⟨Name⟩ ⟨Individuals⟩)
            Individuals ((⟨Individual⟩ :conditions ⟨Conditions⟩)
                         ...)
            Preconditions ⟨Facts⟩
            QuantityConditions ⟨Inequalities⟩
            Relations ⟨Facts⟩
            Influences ⟨Influences⟩)
```

The individuals specify what objects would be involved in the process if it were active, the preconditions and quantity conditions indicate when the process will be active, and the relations and influences specify what relations will hold while the process is active.

```
(defProcess* (Liquid-Flow ?subst ?source ?src-cs ?destination ?dst-cs ?path)
    Individuals ((?subst :conditions (substance ?subst)       ;some stuff
                                     (Liquid ?subst))         ;state
                 (?source :conditions (Can-Contain ?source ?subst))
                 (?src-cs :conditions (Contained-Fluid ?src-cs ?subst ?source))
                 (?destination :conditions (Can-Contain ?destination ?subst))
                 (?dst-cs :conditions (Contained-Fluid ?dst-cs ?subst ?destination))
                 (?path :conditions (Fluid-Path ?path)
                                    (Physical-Path ?source ?destination ?path)))
    Preconditions ((fluid-aligned ?path))
    QuantityConditions ((greater-than (A (Pressure-in ?source))
                                      (A (Pressure-in ?destination)))
                        (greater-than (A (Amount-of ?src-cs)) zero))
    Relations ((Quantity (flow-rate ?self))
               (Q= (flow-rate ?self) (- (pressure-in ?source) (pressure-in ?destination)))
               (Greater-than (A (flow-rate ?self)) zero))
    Influences ((Ctrans (amount-of ?src-cs) (amount-of ?dst-cs) (A (flow-rate ?self)))))


(defProcess* (Liquid-Drain ?sink ?sink-cs ?lf)              ;an ideal sink
   Individuals ((?sink :conditions (Liquid-Sink ?sink))
                (?sink-cs :conditions (contained-liquid ?sink-cs)
                                      (container-of ?sink-cs ?sink))
                (?lf :conditions (Process-Instance-of Liquid-Flow ?lf)
                                 (?lf destination ?sink)))
   QuantityConditions ((Active ?lf))
   Influences ((I- (amount-of ?sink-cs) (A (flow-rate ?lf)))))
```

```
(defProcess* (Heat-Flow ?source ?destination ?path)
   Individuals ((?source :conditions (Thermal-Object ?source))
                (?destination :conditions (Thermal-Object ?destination))
                (?path :conditions (Heat-Path ?path)
                                   (Heat-Connection ?path ?source ?destination)))
   Preconditions ((heat-aligned ?path))
   QuantityConditions ((greater-than (A (temperature ?source)) (A (temperature ?destination)))
                       (greater-than (A (amount-of ?source)) zero)
                       (greater-than (A (amount-of ?destination)) zero))
   Relations ((Quantity (heat-flow-rate ?self))
              (Q= (heat-flow-rate ?self) (- (temperature ?source)
                                            (temperature ?destination))))
   Influences ((CTrans (heat ?source) (heat ?destination) (A (heat-flow-rate ?self)))))


(defProcess* (Boiling ?subst ?container ?cl ?cg ?hf)
   Individuals ((?subst :conditions (substance ?subst))
                (?container :conditions (can-contain ?container ?subst))
                (?cl :conditions (contained-liquid ?cl)
                                 (container-of ?cl ?container)
                                 (substance-of ?cl ?subst))
                (?cg :conditions (contained-gas ?cg)
                                 (container-of ?cg ?container)
                                 (substance-of ?cg ?subst))
                (?hf :conditions (Process-Instance-of Heat-Flow ?hf)
                                 (?hf destination ?cl)))
   QuantityConditions ((Active ?hf)
                       (not (greater-than (A (tboil ?cl)) (A (temperature ?cl))))
                       (greater-than (A (Amount-of ?cl)) zero))
   Relations ((Quantity (Vaporization-Rate ?self))
              (Q= (Vaporization-Rate ?self) (heat-flow-rate ?hf))
              (greater-than (A (vaporization-rate ?self)) zero))
   Influences ((I- (heat ?cl) (A (vaporization-rate ?self)))
               (CTrans (amount-of ?cl) (amount-of ?cg) (A (vaporization-rate ?self)))))


(defProcess* (Heat-Replenish ?source ?hf)                    ;an ideal source
   Individuals ((?source :conditions (Heat-Source ?source))
                (?hf :conditions (Process-Instance-of Heat-Flow ?hf)
                                 (?hf source ?source)))
   QuantityConditions ((Active ?hf))
   Relations ((Equal-to (D (Heat ?source)) ZERO))
   Influences ((I+ (heat ?source) (A (heat-flow-rate ?hf)))))


(defProcess* (Dissolve ?solute ?solution)
   Individuals ((?solute :conditions (Solid ?solute)
                                     (Soluble ?solute))
                (?solution :conditions (Solution ?solution)
                                       (immersed-in ?solute ?solution)))
   Preconditions ((Soluble-in ?solute ?solution))
   QuantityConditions ((Greater-than (A (Amount-of ?solute)) zero))
```

```
    Relations ((Quantity (Dissolve-rate ?self))
               (qprop (dissolve-rate ?self) (contact-surface ?solute ?solution))
               (greater-than (A (dissolve-rate ?self)) zero))
    Influences ((Ctrans (amount-of ?solute) (concentration ?solution)
                        (A (dissolve-rate ?self))))
    Cache ((Supports (immersed-in ?solute ?solution)
                     (prereq (Ctrans (amount-of ?solute) (concentration ?solution)
                                     (A (dissolve-rate ?self)))
                          (Physical-Path ?solute ?solution ?path)
                          (Continuity-Law)))))


;;; Generic process for handling the statement (derivative-of ?q ?qd)
(defProcess* (Derivative-Process ?amount ?derivative)
  Individuals ((?amount :conditions (Quantity ?amount))
               (?derivative :conditions (Quantity ?derivative)
                                         (Derivative-of ?amount ?derivative)))
  Influences ((I+ ?amount (A ?derivative))))
```

Appendix D

Phineas Experiences

PHINEAS begins each explanation task with a set of previously explained experiences which are indexed according to their behavioral abstractions. It's initial task is to retrieve potentially relevant analogues from this set of stored experiences. This section lists the complete set of experiences with which PHINEAS begins.

D.1 Behavioral Abstractions

```
(defBehavioral-Abstraction PHYSICAL-MOVEMENT
     SubTypes (matter-movement wave-movement phase-change-movement))

(defBehavioral-Abstraction matter-movement)
(defBehavioral-Abstraction wave-movement)

(defBehavioral-Abstraction phase-change-movement
     SubTypes (solid-phase-change-move fluidic-phase-change-move gas-phase-change-move))
(defBehavioral-Abstraction solid-phase-change-move)
(defBehavioral-Abstraction fluidic-phase-change-move)
(defBehavioral-Abstraction gas-phase-change-move)

;;; Observable, characterizing type physical movement

(defBehavioral-Abstraction PHYSICAL-MOVEMENT-CHARACTERISTICS
     SubTypes (corpuscular-movement continuous-movement wavelike))

(defBehavioral-Abstraction corpuscular-movement)
(defBehavioral-Abstraction wavelike)

(defBehavioral-Abstraction continuous-movement)

;;; Interaction types

(defBehavioral-Abstraction Contact-scenes
     SubTypes (fluid-flow solid-contact))
```

```
(defBehavioral-Abstraction fluid-flow)
(defBehavioral-Abstraction solid-contact)

;;; Force-Action types

(defBehavioral-Abstraction MODE-OF-FORCE
     SubTypes (action-at-a-distance direct-force))

(defBehavioral-Abstraction action-at-a-distance)

(defBehavioral-Abstraction direct-force
     SubTypes (push-force pull-force twisting-force))

(defBehavioral-Abstraction pull-force)
(defBehavioral-Abstraction twisting-force)
(defBehavioral-Abstraction push-force)

;;; Behavior types - graphical

(defBehavioral-Abstraction GRAPHICAL-CHARACTERIZERS
     SubTypes (monotonic cyclic))    ;non-descript)

(defBehavioral-Abstraction monotonic
     SubTypes (linear asymptotic-approach))

(defBehavioral-Abstraction linear)

(defBehavioral-Abstraction cyclic
     SubTypes (sinusoidal ramp-cyclic simple-cyclic))

(defBehavioral-Abstraction sinusoidal)
(defBehavioral-Abstraction ramp-cyclic)
(defBehavioral-Abstraction simple-cyclic)

(defBehavioral-Abstraction asymptotic-approach
     SubTypes (approach-constant dual-approach))

(defBehavioral-Abstraction approach-constant)

(defBehavioral-Abstraction dual-approach
     SubTypes (dual-approaching dual-approach-finish))

(defBehavioral-Abstraction dual-approaching)
(defBehavioral-Abstraction dual-approach-finish)

;;; Compile it

(compile-abstraction-hierarchy)
```

D.2 Past Observations

;;; Two container liquid flow

```
(sme:defEntity water)
(sme:defEntity liquid :constant? t)
(sme:defEntity beaker3)
(sme:defEntity vial2)
(sme:defEntity cs-water-beaker)
(sme:defEntity cs-water-vial)
(sme:defEntity pipe1)

(defObservation 2-container-lf
     Behavior 2-container-lf
     Individuals (beaker3 vial2 pipe1)
     World ((contained-liquid cs-water-beaker)
            (container-of cs-water-beaker beaker3)
            (substance-of cs-water-beaker water)
            (contained-liquid cs-water-vial)
            (container-of cs-water-vial vial2)
            (substance-of cs-water-vial water)
            (can-contain beaker3 water)
            (can-contain vial2 water)
            (substance water)
            (Liquid water)
            (Fluid-Path pipe1)
            (Physical-Path beaker3 vial2 pipe1)))

(defBSegment 2-container-lf
  Characterizations ((matter-movement ?self)
                     (continuous-movement ?self)
                     (dual-approach-finish ?self))
  Components (2-container-sit0 2-container-sit1)
  Relations ((meets 2-container-sit0 2-container-sit1)))

(defSituation 2-container-sit0
  Characterizations ((matter-movement ?self)
                     (continuous-movement ?self)
                     (dual-approaching ?self))
  Processes ((liquid-flow pi1 ((?subst . water)(?source . beaker3)(?src-cs . cs-water-beaker)
                               (?destination . vial2)(?dst-cs . cs-water-vial) (?path . pipe1)))
             (contained-fluid ((?cs . cs-water-beaker)(?sub . water)
                               (?container . beaker3)))
             (contained-fluid ((?cs . cs-water-vial)(?sub . water)
                               (?container . vial2))))
  Individuals (water cs-water-beaker cs-water-vial beaker3 vial2 pipe1 liquid)
  Dynamics  ((Decreasing (Amount-of cs-water-beaker))
             (Decreasing (Pressure-in beaker3))
             (Increasing (Amount-of cs-water-vial))
             (Increasing (Pressure-in vial2))
             (Exponential-Approach-Dual (Pressure-in beaker3) (Pressure-in vial2))
             (Greater-Than (A (Pressure-in beaker3)) (A (Pressure-in vial2)))))
```

```
(defSituation 2-container-sit1
  Individuals (cs-water-beaker cs-water-vial)
  Dynamics ((Constant (Amount-of cs-water-beaker))
            (Constant (Pressure-in beaker3))
            (Constant (Amount-of cs-water-vial))
            (Constant (Pressure-in vial2))
            (Equal-to (A (Pressure-in beaker3)) (A (Pressure-in vial2)))))
```

;;; Liquid draining from a leaky cup

```
(sme:defEntity water)
(sme:defEntity leaky-cup)
(sme:defEntity cs-leaky)
(sme:defEntity sink5)
(sme:defEntity cs-sink5)
(sme:defEntity hole7)

(defObservation liquid-draining
     Behavior liquid-draining-behavior
     Individuals (leaky-cup cs-leaky water sink5 cs-sink5 hole7)
     World ((Liquid-Sink sink5)
            (container leaky-cup)
            (can-contain leaky-cup water)
            (contained-liquid cs-leaky)
            (container-of cs-leaky leaky-cup)
            (substance-of cs-leaky water)
            (can-contain sink5 water)
            (contained-liquid cs-sink5)
            (container-of cs-sink5 sink5)
            (substance-of cs-sink5 water)
            (substance water)
            (Liquid water)
            (Hole-of leaky-cup hole7)
            (Fluid-Path hole7)
            (Physical-Path leaky-cup sink5 hole7)))

(defBSegment liquid-draining-behavior
  Characterizations ((matter-movement ?self)
                     (continuous-movement ?self)
                     (monotonic ?self))
  Components (liquid-draining-bseg liquid-drained-bseg)
  Relations ((meets liquid-draining-bseg liquid-drained-bseg)))

(defSituation liquid-draining-bseg
  Characterizations ((matter-movement ?self)
                     (continuous-movement ?self)
                     (monotonic ?self))
  Processes ((liquid-flow pi1 ((?subst . water)(?source . leaky-cup)(?src-cs . cs-leaky)
                               (?destination . sink5)(?dst-cs . cs-sink5)(?path . hole7)))
             (liquid-drain pi2 ((?sink . sink5)(?sink-cs . cs-sink5)(?lf . pi1))))
  Individuals (leaky-cup cs-leaky water sink5 cs-sink5 hole7)
  Dynamics  ((Decreasing (Amount-of cs-leaky))
            (Greater-than (A (Amount-of cs-leaky)) zero)))

(defSituation liquid-drained-bseg
  Individuals (cs-leaky)
  Dynamics ((Constant (Amount-of cs-leaky))
            (Equal-to (A (Amount-of cs-leaky)) zero)))
```

;;; Boiling

```
(sme:defentity water)
(sme:defentity magnalite)
(sme:defentity illini-water)
(sme:defentity illini-steam)
(sme:defentity westinghouse)

(defObservation boiling-obs
     Behavior boiling-behavior
     Individuals (westinghouse magnalite illini-water illini-steam)
     World ((stove westinghouse)
            (pan magnalite)
            (contained-liquid illini-water)
            (substance-of illini-water water)
            (container-of illini-water magnalite)
            (state-of illini-water liquid)
            (contained-gas illini-steam)
            (substance-of illini-steam water)
            (container-of illini-steam magnalite)
            (state-of illini-steam gas)
            (can-contain magnalite water)
            (thermal-object westinghouse)
            (thermal-object illini-water)
            (heat-source westinghouse)
            (physical-path westinghouse illini-water magnalite)
            (heat-path magnalite)))

(defBSegment boiling-behavior
  Characterizations ((fluidic-phase-change-move ?self)
                     (continuous-movement ?self)
                     (monotonic ?self))
  Components (boiling-bseg boiling-dry)
  Relations ((meets boiling-bseg boiling-dry)))

(defSituation boiling-bseg
  Characterizations ((fluidic-phase-change-move ?self)
                     (continuous-movement ?self)
                     (monotonic ?self))
  Processes ((heat-flow pi1 ((?source . westinghouse)(?destination . illini-water)
                             (?path . magnalite)))
             (boiling pi2 ((?container . magnalite)(?subst . water)(?cl . illini-water)
                           (?cg . illini-steam)(?hf . pi1)))
             (heat-replenish pi3 ((?source . westinghouse)(?hf . pi1))))
  Individuals (water westinghouse magnalite illini-water illini-steam)
  Dynamics ((Constant (Temperature westinghouse))
                (Constant (Temperature illini-water))
                (Equal-to (A (Temperature westinghouse)) (A (Temperature illini-water)))
                (Greater-than (A (Amount-of illini-water)) zero)
                (Decreasing (Amount-of illini-water))
                (Increasing (Amount-of illini-steam))))

(defSituation boiling-dry
  Individuals (illini-water)
  Dynamics ((Constant (Amount-of illini-water))
                (Equal-to (A (Amount-of illini-water)) zero)
                (Greater-than (A (Amount-of illini-steam)) zero)))
```

;;; Dissolving

```
(sme:defentity water)
(sme:defentity water1)
(sme:defentity glass4)
(sme:defentity salt1)

(defObservation dissolving
     Behavior dissolve-behavior
     Individuals (water1 water salt1 glass4)
     World ((contained-liquid water1)
             (substance-of water1 water)
             (container-of water1 glass4)
             (state-of water1 liquid)
             (solid salt1)
             (soluble salt1)
             (soluble-in salt1 water1)
             (solution water1)
             (immersed-in salt1 water1)))

(defBSegment dissolve-behavior
  Characterizations ((solid-phase-change-move ?self)
                     (continuous-movement ?self)
                     (monotonic ?self))
  Components (dissolving dissolve-stopped)
  Relations ((meets dissolving dissolve-stopped)))

(defSituation dissolving
  Characterizations ((solid-phase-change-move ?self)
                     (continuous-movement ?self)
                     (monotonic ?self))
  Processes ((dissolve pi1 ((?solute . salt1)(?solution . water1)))
              (solution ((?solution . water1))))
  Dynamics ((Greater-than (A (Amount-of salt1)) zero)
             (Decreasing (Amount-of salt1))
             (Increasing (Concentration water1))))

(defSituation dissolve-stopped
  Individuals (water glass4 water1 salt1)
  Processes ((solution ((?solution . water1))))
  Dynamics  ((Constant (Amount-of salt1))
              (Constant (Concentration water1))
              (Equal-to (A (Amount-of salt1)) zero)))
```

;;; Spring oscillating

```
(sme:defEntity sm-sys)
(sme:defEntity spring5)
(sme:defEntity block5)
(sme:defEntity mechanical :constant? T)

(defObservation spring-mass-oscillator
     Behavior spring-mass-oscillating
     Individuals (sm-sys spring5 block5)
     World ((Connected spring5 block5)
            (Spring-Mass-System sm-sys spring5 block5)
            (Spring spring5)
            (Block  block5)
            (Compressing-Object string1)
            (Translating-object block5)))

(defBSegment spring-mass-oscillating
  Characterizations (;(matter-movement ?self)
                     ;(corpuscular-movement ?self)
                     (sinusoidal ?self))
  Components (spring-mass-s1 spring-mass-s2 spring-mass-s3 spring-mass-s4
              spring-mass-s5 spring-mass-s6 spring-mass-s7 spring-mass-s8)
  Relations ((meets spring-mass-s1 spring-mass-s2)
             (meets spring-mass-s2 spring-mass-s3)
             (meets spring-mass-s3 spring-mass-s4)
             (meets spring-mass-s4 spring-mass-s5)
             (meets spring-mass-s5 spring-mass-s6)
             (meets spring-mass-s6 spring-mass-s7)
             (meets spring-mass-s7 spring-mass-s8)
             (meets spring-mass-s8 spring-mass-s1)))

;;;   ----0--[<]--
(defSituation spring-mass-s1
  Processes ((Spring ((?spring . spring5)))
             (Translating-Object ((?obj . block5)))
             (Applied-Force vi0 ((?src . spring5)(?dst . block5)
                                 (?force . restorative-force)(?acc . acceleration)))
             (Spring-Mass-System ((?system . sm-sys)(?spring . spring5)(?mass . block5))))
  Individuals (spring5 block5)
  Dynamics ((Decreasing (Displacement spring5))
            (Decreasing (Position block5))
            (Decreasing (Velocity block5))
            (Less-than (A (Velocity block5)) ZERO)
            (Greater-Than (A (Position block5)) zero)
    (Greater-Than (A (Displacement spring5)) zero)))

;;;   ----[<0]----
(defSituation spring-mass-s2
  Processes ((Spring ((?spring . spring5)))
           (Translating-Object ((?obj . block5)))
           (Spring-Mass-System ((?system . sm-sys)(?spring . spring5)(?mass . block5))))
  Individuals (spring5 block5)
  Dynamics ((Decreasing (Displacement spring5))
           (Decreasing (Position block5))
           (Constant (Velocity block5))
           (Less-than (A (Velocity block5)) ZERO)
           (Equal-To (A (Position block5)) ZERO)
           (Equal-To (A (Displacement spring5)) ZERO)))

;;;   ----[<]--0---
(defSituation spring-mass-s3
  Processes ((Spring ((?spring . spring5)))
           (Translating-Object ((?obj . block5)))
           (Applied-Force vi0 ((?src . spring5)(?dst . block5)
                               (?force . restorative-force)(?acc . acceleration)))
           (Spring-Mass-System ((?system . sm-sys)(?spring . spring5)(?mass . block5))))
  Individuals (spring5 block5)
  Dynamics ((Decreasing (Displacement spring5))
           (Decreasing (Position block5))
           (Increasing (Velocity block5))
           (Less-than (A (Velocity block5)) ZERO)
           (Less-Than (A (Position block5)) zero)
           (Less-Than (A (Displacement spring5)) zero)))

;;;   []----0---
(defSituation spring-mass-s4
  Processes ((Spring ((?spring . spring5)))
           (Applied-Force vi0 ((?src . spring5)(?dst . block5)
                               (?force . restorative-force)(?acc . acceleration)))
           (Spring-Mass-System ((?system . sm-sys)(?spring . spring5)(?mass . block5))))
  Individuals (spring5 block5)
  Dynamics ((Constant (Displacement spring5))
           (Constant (Position block5))
           (Increasing (Velocity block5))
           (Equal-To (A (Velocity block5)) ZERO)
           (Less-Than (A (Position block5)) zero)
           (Less-Than (A (Displacement spring5)) zero)))


;;;   ---[>]--0---
(defSituation spring-mass-s5
  Processes ((Spring ((?spring . spring5)))
           (Translating-Object ((?obj . block5)))
           (Applied-Force vi0 ((?src . spring5)(?dst . block5)
                               (?force . restorative-force)(?acc . acceleration)))
           (Spring-Mass-System ((?system . sm-sys)(?spring . spring5)(?mass . block5))))
  Individuals (spring5 block5)
  Dynamics ((Increasing (Displacement spring5))
           (Increasing (Position block5))
           (Increasing (Velocity block5))
           (Greater-than (A (Velocity block5)) ZERO)
```

```
            (Less-Than (A (Position block5)) zero)
            (Less-Than (A (Displacement spring5)) zero)))

;;;   ----[0>]-----
(defSituation spring-mass-s6
  Processes ((Spring ((?spring . spring5)))
             (Translating-Object ((?obj . block5)))
             (Spring-Mass-System ((?system . sm-sys)(?spring . spring5)(?mass . block5))))
  Individuals (spring5 block5)
  Dynamics ((Increasing (Displacement spring5))
            (Increasing (Position block5))
            (Constant (Velocity block5))
            (Greater-than (A (Velocity block5)) ZERO)
            (Equal-To (A (Position block5)) Zero)
            (Equal-To (A (Displacement spring5)) Zero)))

;;;   --0--[>]----
(defSituation spring-mass-s7
  Processes ((Spring ((?spring . spring5)))
             (Translating-Object ((?obj . block5)))
             (Applied-Force vi0 ((?src . spring5)(?dst . block5)
                                 (?force . restorative-force)(?acc . acceleration)))
             (Spring-Mass-System ((?system . sm-sys)(?spring . spring5)(?mass . block5))))
  Individuals (spring5 block5)
  Dynamics ((Increasing (Displacement spring5))
            (Increasing (Position block5))
            (Decreasing (Velocity block5))
            (Greater-than (A (Velocity block5)) ZERO)
            (Greater-Than (A (Position block5)) zero)
            (Greater-Than (A (Displacement spring5)) zero)))

;;;   --0----[]
(defSituation spring-mass-s8
  Processes ((Spring ((?spring . spring5)))
             (Applied-Force vi0 ((?src . spring5)(?dst . block5)
                                 (?force . restorative-force)(?acc . acceleration)))
             (Spring-Mass-System ((?system . sm-sys)(?spring . spring5)(?mass . block5))))
  Individuals (spring5 block5)
  Dynamics ((Constant (Displacement spring5))
            (Constant (Position block5))
            (Decreasing (Velocity block5))
            (Equal-To (A (Velocity block5)) ZERO)
            (Greater-Than (A (Position block5)) zero)
            (Greater-Than (A (Displacement spring5)) zero)))
```

;;; Air plane wing

```
(sme:defEntity wing9 :type physob)
(sme:defEntity air-stream9 :type physob)

(defObservation air-plane-wing
     Behavior plane-wing-lift
     Individuals (wing9 air-stream9)
```

```
    World ((Touching air-stream9 (Tside wing9))
           (Touching air-stream9 (Bside wing9))
           (Flowing-Air air-stream9)
           (Physob wing9)
           (Enveloped wing9 air-stream9)
           (object-part-of (Tside wing9) wing9)
           (object-part-of (Bside wing9) wing9)
           (Opposite-sides (Tside wing9) (Bside wing9))
           (derivative-of (position wing9)(velocity wing9))
           (Equal-to (A (Curvature (Bside wing9))) (A (Curvature (Tside wing9))))))

(defSituation plane-wing-lift
  Characterizations ((fluid-flow ?self))
  Processes ((air-plane-wing ((?wing . wing9)))
             (air-contact vi0 ((?air . air-stream9)(?obj . wing9)
                               (?pos-part . (Tside wing9))(?neg-part . (Bside wing9))))
             (air-plane-lift vi1 ((?air . air-stream9)(?obj . wing9)(?part . (Tside wing9))))
             (air-plane-lift vi2 ((?air . air-stream9)(?obj . wing9)(?part . (Bside wing9))))
             (applied-force-pos pi0 ((?obj . wing9)(?part . (Bside wing9))
                                     (?src . air-stream9)))
             (applied-force-neg pi1 ((?obj . wing9)(?part . (Tside wing9))
                                     (?src . air-stream9))))
  Individuals (air-stream9 wing9)
  Dynamics ((Constant (Position wing9))
            (Constant (Velocity wing9))))
```

Appendix E

Language Declarations

All predicates must be declared to SME prior to use. Each declaration defines the predicate's arity, a name and type for each argument, and the next most general type to which the predicate maps. Predicates may additionally be declared *commutative*, in which the order of arguments is unimportant when matching, and/or *n-ary*, in which the predicate can take any number of arguments. This section presents a complete listing of the language declarations used for PHINEAS.

;;;; Standard entities

```
(sme:defentity zero :type number :constant? t)
(sme:defentity atmosphere :type physob :constant? t)
```

;;;; Predicates

;;; Time

```
(sme:defPredicate At ((object physob)(time time-interval)) function :expression-type slice)
(sme:defPredicate Situation ((time-token time-interval)(relations set)) relation
                  :expression-type :temporal)
(sme:defPredicate Bseg ((time-token time-interval)) Attribute)
(sme:defPredicate Meets ((before time)(after time)) Relation
  :documentation "(before after): interval before ends at point where interval after begins.")
(sme:defPredicate Summary-of ((parent-time time)(set set)) relation)
```

;;; Derivatives

```
(sme:defPredicate Decreasing ((arg linear)) Relation)
(sme:defPredicate Increasing ((arg linear)) Relation)
(sme:defPredicate Constant   ((arg linear)) Relation)
```

;;; Change descriptors

```
(sme:defPredicate Exponential-Decay ((Q quantity)) Relation)
(sme:defPredicate Exponential-Growth ((Q quantity)) Relation)
(sme:defPredicate Exponential-Approach ((Q quantity)) Relation)
(sme:defPredicate Exponential-Approach-Dual ((Q-decreasing quantity)(Q-increasing quantity))
                  Relation)

;;; Inequality information

(sme:defPredicate Equal-To ((arg1 linear) (arg2 linear))  relation :commutative? t)
(sme:defPredicate Greater-Than ((arg1 linear) (arg2 linear)) relation
                :equivalent ((less-than arg2 arg1)))
(sme:defPredicate Less-Than ((arg1 linear) (arg2 linear)) relation
                :equivalent ((greater-than arg2 arg1)))
(sme:defPredicate Unrelated ((arg1 linear) (arg2 linear)) relation :commutative? t)

;;; Quantities
;;; The defQtype form is a PHINEAS routine which invokes defPredicate
;;; as well as identifying it as a quantity for PHINEAS internal use.

(defQtype Change-Rate ((obj physob)) function :expression-type rate)
(defQtype Amount-of ((obj physob)) function :expression-type extensive-quantity)
(defQtype Amount-of-in ((sub physob)(state state)(container physob)) function
                      :expression-type extensive-quantity)
(defQtype Pressure ((obj physob)) function :expression-type intensive-quantity)
(defQtype Volume ((obj physob)) function :expression-type extensive-quantity)
(defQtype Surface-Area ((obj physob)) function :expression-type extensive-quantity)
(defQtype Contact-Surface ((obj1 physob)(obj2 physob)) function
                         :expression-type extensive-quantity)

(defQtype Pressure-in ((object physob)) function :expression-type intensive-quantity)
(defQtype Temperature-in ((object physob)) function :expression-type intensive-quantity)

(defQtype Heat (physob) function :expression-type extensive-quantity)
(defQtype Temperature ((obj physob)) function :expression-type intensive-quantity)
(defQtype Tboil (physob) function :expression-type temperature)

(defQtype Mass ((obj physob)) function :expression-type extensive-quantity)
(defQtype Moment-of-Inertia ((obj physob)) function :expression-type mass)

(defQtype Position ((obj physob)) function :expression-type distance :spatial :linear)
(defQtype Angle ((obj physob)) function :expression-type distance :spatial :angular)
(defQtype Displacement ((obj physob)) function :expression-type distance :spatial :linear)
(defQtype Angular-Displacement ((obj physob)) function :expression-type distance
                                                       :spatial :angular)

(defQtype Velocity ((obj physob))     function :expression-type rate :spatial :linear)
(defQtype Angular-Velocity ((obj physob)) function :expression-type velocity
                                                   :spatial :angular)

(defQtype Acceleration ((obj physob)) function :expression-type rate :spatial :linear)
(defQtype Angular-Acceleration ((obj physob)) function :expression-type acceleration
                                                       :spatial :angular)
```

```
(defQtype Compliance ((obj physob))   function :expression-type extensive-quantity)
(defQtype Concentration ((obj liquid)) function :expression-type intensive-quantity)
(defQtype Saturation-Point ((obj liquid)) function :expression-type concentration)

(defQtype Force-Restoring ((obj physob)) function :expression-type force)
(defQtype Internal-force ((obj physob)) function :expression-type force)
(defQtype Restorative-force ((obj physob)) function :expression-type force)
(defQtype Torque ((obj physob)) function :expression-type force)

(defQtype Energy ((obj physob)) function :expression-type extensive-quantity)
(defQtype Kinetic-Energy ((obj physob)) function :expression-type energy)
(defQtype Total-Energy ((obj physob))   function :expression-type energy)
(defQtype Potential-Energy ((obj physob)) function :expression-type energy)

(defQtype Length ((obj physob)) function :expression-type distance :spatial :linear)
(defQtype Rest-Length ((obj physob)) function :expression-type length :spatial :linear)

(defQtype Charge  ((obj physob)) function :expression-type extensive-quantity)
(defQtype Current ((obj physob)) function :expression-type rate)
(defQtype Voltage ((obj physob)) function :expression-type extensive-quantity)

(sme:defPredicate Quantity ((obj physob)) function)
(sme:defPredicate Rate ((Q quantity)) function :expression-type quantity) ;***
(sme:defPredicate Intensive-Quantity ((quantity quantity)) function :expression-type quantity)
(sme:defPredicate Extensive-Quantity ((quantity quantity)) function :expression-type quantity)
(sme:defPredicate Linear ((num linear)) function)
(sme:defPredicate Number ((Q quantity)) function :expression-type linear)
(sme:defPredicate Ordinal ((Q ordinal)) function :expression-type linear)
```

;;; Miscellaneous movements

```
(sme:defPredicate Moving-Object ((obj physob)) relation)
(sme:defPredicate Translating-Object ((obj physob)) relation :expression-type moving-object)
(sme:defPredicate Rotating-Object ((obj physob)) relation :expression-type moving-object)

(sme:defPredicate Deformed-Object ((obj physob)) relation)
(sme:defPredicate Twisting-Object ((obj physob)) relation :expression-type deformed-object)
(sme:defPredicate Compressing-Object ((obj physob)) relation :expression-type deformed-object)
```

;;; QP syntax

```
(sme:defpredicate A ((Q quantity)) function :expression-type number)
(sme:defpredicate D ((Q quantity)) function :expression-type number)
(sme:defpredicate Q= ((quantity quantity) (exp math-expression)) relation)
(sme:defPredicate Qprop  ((Q1 quantity) (Q2 quantity)) relation)
(sme:defPredicate Qprop- ((Q1 quantity) (Q2 quantity)) relation)
(sme:defPredicate I- ((Influenced quantity) (Influencer number)) relation)
(sme:defPredicate I+ ((Influenced quantity) (Influencer number)) relation)
(sme:defPredicate CTrans ((source quantity)(dest quantity)(rate quantity)) relation)
(sme:defPredicate Exists ((obj physob)) attribute)
(sme:defPredicate - ((Q quantity)(form expression)) relation :n-ary? T)
(sme:defPredicate + ((Q quantity)(form expression)) relation)
(sme:defPredicate * ((Q1 quantity)(Q2 quantity)) relation)
```

```
(sme:defPredicate Process-Definition ((name process)(PI process-instance)
                                        (consequences sentence)) relation)
(sme:defPredicate Packet-Definition ((P-name packet)(consequences sentence)) relation)
(sme:defPredicate View-Definition (sentence) relation)
(sme:defPredicate Process (entity) attribute)
(sme:defPredicate Individual ((individual item)(conditions expression)) relation)
(sme:defPredicate Conditions ((condition expression)) relation :commutative? t :n-ary? t)
(sme:defPredicate Has-Quantity (quantity entity) attribute)
(sme:defpredicate Process-instance-of ((process-type process)(process-token pid)) relation)
(sme:defPredicate Active ((process process-instance)) attribute :expression-type sentence)
(sme:defPredicate Correspondence ((pair1 pair)(pair2 pair)) relation)
```

;;; Logic

```
(sme:defPredicate not (sentence) relation)
(sme:defPredicate Implies ((antecedent predicate) (consequent predicate)) relation
                  :expression-type :implicational)
(sme:defPredicate Cause ((antecedent event) (consequent event)) relation
                  :expression-type :implicational)
(sme:defPredicate Supports ((antecedent predicate) (consequent predicate)) relation
                  :expression-type :implicational)
(sme:defpredicate or ((disjunct sentence) (disjunct sentence)) logical
                  :n-ary? T :commutative? T :relgroup? T)
(sme:defPredicate and ((conjunct sentence) (conjunct sentence)) logical
                  :n-ary? T :commutative? T :relgroup? T)
```

;;; Process quantities

```
(defQtype Vaporization-Rate ((obj physob)) function :expression-type rate)
(defQtype Heat-Flow-Rate ((obj physob)) function :expression-type rate)
(defQtype Flow-Rate ((obj physob)) function :expression-type rate)
(defQtype Restorative ((obj physob)) function :expression-type number)
(defQtype Replenish-Rate ((obj physob)) function :expression-type rate)
(defQtype Absorption ((obj physob)) function :expression-type number)
(defQtype Dissolve-Rate ((obj physob)) function :expression-type number)
```

;;; Objects

```
(sme:defPredicate Physob ((obj physob)) attribute)
(sme:defPredicate Spring ((obj physob)) attribute :expression-type physob)
(sme:defPredicate Block ((obj physob)) attribute :expression-type physob)
(sme:defPredicate Inductor ((obj physob)) attribute :expression-type physob)
(sme:defPredicate Capacitor ((obj physob)) attribute :expression-type physob)
(sme:defPredicate Container ((obj physob)) attribute :expression-type physob)
(sme:defPredicate Pan ((obj physob)) attribute :expression-type container)
(sme:defPredicate Pipe ((obj physob)) attribute :expression-type physob)
(sme:defPredicate Stove ((obj physob)) attribute :expression-type physob)
(sme:defPredicate Beaker ((obj physob)) attribute :expression-type physob)
(sme:defPredicate Ball ((obj physob)) attribute :expression-type physob)
(sme:defPredicate String ((obj physob)) attribute :expression-type physob)
```

;;; Set notation

```
(sme:defPredicate Set ((set-item physob)) relation :commutative? T :n-ary? T :relgroup? T)
(sme:defPredicate Pair ((obj1 physob)(obj2 physob)) relation :commutative? T)
(sme:defPredicate Ordered-Pair ((obj1 physob)(obj2 physob)) relation)
```

;;; Phase states

```
(sme:defPredicate Phase ((substance physob)) relation
  :documentation "(substance): top-level node representing substance is in some phase state")
(sme:defPredicate Solid ((substance physob)) relation :expression-type phase)
(sme:defPredicate Liquid ((substance physob)) relation :expression-type phase)
(sme:defPredicate Gas ((substance physob)) relation :expression-type phase)
```

;;; Paths

```
(sme:defPredicate conduit ((path physob)) relation)
(sme:defPredicate fluid-path ((path physob)) relation :expression-type conduit)
(sme:defPredicate heat-path ((path physob)) relation :expression-type conduit)
(sme:defPredicate closed-path ((object physob) (end-points pair)) relation)
```

;;; Path alignment

```
(sme:defPredicate Aligned ((generic-path path)) attribute
  :documentation "(generic-path): will currently allow its substance to pass through")
(sme:defPredicate Heat-Aligned ((heat-path heat-path)) attribute :expression-type aligned
  :documentation "(heat-path): the specified object is currently a FUNCTIONAL heat path")
(sme:defPredicate Fluid-Aligned ((fluid-path fluid-path)) attribute :expression-type aligned
  :documentation "(fluid-path): the specified object is currently a FUNCTIONAL fluid path")
(sme:defPredicate Gas-Aligned ((gas-path gas-path)) attribute :expression-type aligned
  :documentation "(gas-path): the specified object is currently a FUNCTIONAL gas path")
```

;;; Spatial relationships

```
(sme:defPredicate Physical-Proximity ((object physob)(object physob)) relation :commutative? T)
(sme:defPredicate Physical-Path ((obj1 physob)(obj2 physob)(path physob)) relation
                  :expression-type physical-proximity)

(sme:defPredicate Touching ((obj physob) (obj physob)) relation
                  :expression-type physical-proximity   :commutative? T)

(sme:defPredicate Connected ((obj1 physob)(obj2 physob)) relation
                  :expression-type physical-proximity   :commutative? T)
(sme:defPredicate Connection (physob physob physob) relation)

(sme:defPredicate Common-Face ((obj1 solid)(obj2 solid)) function :expression-type solid)
```

;;; Containment

```
(sme:defpredicate Containment ((containee physob)(container physob)) relation
                  :expression-type physical-proximity)
(sme:defpredicate Contained-in ((containee fluid)(container physob)) relation
                  :expression-type containment)
(sme:defpredicate Absorbed-in ((absorbee liquid)(absorber solid)) relation
                  :expression-type containment)
```

```
(sme:defpredicate Dispersed-in ((solute fluid)(solvent fluid)) relation
                  :expression-type containment)
(sme:defpredicate Immersed-in ((containee solid)(container liquid)) relation
                  :expression-type physical-proximity)
(sme:defpredicate Floating-in ((containee solid)(container liquid)) relation
                  :expression-type physical-proximity)

(sme:defpredicate Contained-Fluid ((cs entity)(sub substance)(can container)) relation)
(defCompound-Object Contained-Fluid)

(sme:defpredicate Contained-Stuff ((cs entity)) relation)
(defCompound-Object Contained-Stuff)
(sme:defpredicate Contained-Liquid ((cl liquid)) relation :expression-type contained-stuff)
(defCompound-Object Contained-Liquid)
(sme:defpredicate Contained-Gas ((cg gas)) relation :expression-type contained-stuff)
(defCompound-Object Contained-Gas)

(sme:defPredicate Atmosphere ((air gas)) relation :expression-type contained-gas)

(sme:defpredicate can-containment ((container entity)(sub entity)) relation)
(sme:defpredicate can-contain ((container entity)(sub entity)) relation
                              :expression-type can-containment)
(sme:defpredicate can-absorb ((absorber solid)(absorbee fluid)) relation
                  :expression-type can-containment)

(sme:defPredicate Solution ((solution liquid)) relation)
(sme:defPredicate Insoluble ((obj solid)) relation)
(sme:defPredicate Soluble ((obj solid)) relation)
(sme:defPredicate Soluble-in ((solute solid)(solvent liquid)) relation)

;;; Miscellaneous

(sme:defPredicate Active-Source (physob) attribute :expression-type boolean)
(sme:defPredicate Function-Of ((q1 linear)(q2 linear)) Relation)
(sme:defPredicate state ((st inanimate)) attribute)
(sme:defPredicate substance ((sub inanimate)) attribute)

(sme:defPredicate water ((substance physob)) attribute :expression-type physob)

(sme:defpredicate holds-liquid ((obj physob)) relation)
(sme:defpredicate container-of ((cs entity)(container entity)) relation)
(sme:defpredicate substance-of ((cs entity)(substance substance)) relation)
(sme:defpredicate state-of ((cs entity)(state state)) relation)

(sme:defpredicate Volume-Solid ((obj physob)) relation)
(sme:defpredicate Thermal-object ((obj physob)) relation)

(sme:defpredicate open ((can container)) relation)
(sme:defpredicate closed-container ((can container)) relation)
(sme:defpredicate hole-of ((can container)(hole physob)) relation)

(sme:defpredicate b-explains ((theories (set theories)) (situation time-interval)) relation
  :documentation "(theories situation): theories explain, via b-analogy, the observation.")
```

```
(sme:defpredicate flow-process (pid) relation)
(sme:defPredicate liquid-sink ((obj physob)) attribute)
(sme:defPredicate heat-source (entity) attribute)
(sme:defPredicate source (pid entity) relation)
(sme:defPredicate destination ((pid process-instance)(entity physob)) relation)

(sme:defPredicate isa (entity entity) relation)
(sme:defPredicate force ((from physob)(to physob)) relation)
(sme:defPredicate force-application ((from quantity)(to quantity)) relation)
(sme:defPredicate force-type (force-name) relation)
(sme:defPredicate dt ((Q quantity)(Derivative quantity)) relation)
(sme:defPredicate derivative-of ((Q quantity)(Derivative quantity)) relation)
(sme:defPredicate Spring-Mass-System ((system entity)(spring spring)(block physob)) relation
                  :expression-type object-system)
(defCompound-Object Spring-Mass-System)
(sme:defPredicate object-system (entity entity entity) relation)

(sme:defPredicate connected-to-spring ((mass physob)(spring spring)) relation)
(sme:defPredicate sinusoidal ((Q quantity)) relation)
(sme:defPredicate 90-degree-delay ((referent quantity)(delayed quantity)) relation)

(sme:defPredicate mobile ((obj physob)) relation)

;;; Obscure QPE needs
(sme:defpredicate PI0 ((arg argument)(value argument)) relation)
(sme:defpredicate PI1 ((arg argument)(value argument)) relation)
(sme:defpredicate PI2 ((arg argument)(value argument)) relation)
(sme:defEntity source :constant? T :type :process-arg)
(sme:defEntity destination :constant? T :type :process-arg)
```

References

Agre, P. E and Chapman, D. Pengi: An implementation of a theory of activity. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 268–272, July 1987.

Ashley, K. D and Rissland, E. L. Compare and contrast, A test of expertise. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 273–278, July 1987.

Bain, W. M. A case-based reasoning system for subjective assessment. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 523–527, August 1986.

Baker, M, Burstein, M. H, and Collins, A. Implementing a model of human plausible reasoning. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 185–188, August 1988.

Bobrow, D (Ed.). *Qualitative Reasoning about Physical Systems.* MIT Press, Cambridge, MA, 1985.

Boolos, G. S and Jeffrey, R. C. *Computability and Logic.* Cambridge University Press, Cambridge, 1974.

Buchanan, B. G and Shortliffe, E. H. *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project.* Addison-Wesley, Reading, MA, 1984.

Buckley, S. *Sun up to sun down.* McGraw-Hill Company, New York, 1979.

Burstein, M. Concept formation by incremental analogical reasoning and debugging. In *Proceedings of the Second International Workshop on Machine Learning*, Monticello, IL, June 1983. (Revised version appears in R.S. Michalski, J. Carbonell, T. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach, Volume II*, Morgan Kaufmann, 1986).

Burstein, M. *Learning by Reasoning from Multiple Analogies.* PhD thesis, Yale University, May 1985.

Carbonell, J. G. Invariance hierarchies in metaphor interpretation. In *Proceedings of the Third Meeting of the Cognitive Science Society*, pages 292–295, August 1981.

Carbonell, J. G. Derivational analogy in problem solving and knowledge acquisition. In *Proceedings of the Second International Workshop on Machine Learning*, Monticello, IL, June 1983. A revised version appears in *Machine Learning: An Artificial Approach Vol. II*, R.S. Michalski, J.G. Carbonell, and T.M. Mitchell (Eds.), Morgan Kaufman, 1986.

Carbonell, J. G. Learning by analogy: Formulating and generalizing plans from past experience. In R. S Michalski, J. G Carbonell, and T. M Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach*, Morgan Kaufman, 1983.

Charniak, E. *Towards a Model of Children's Story Comprehension.* PhD thesis, Massachusetts Institute of Technology, 1972. Technical Report AI TR-266.

Charniak, E. Motivation analysis, abductive unification, and nonmonotonic equality. *Artificial Intelligence*, **34**(3):275–295, 1988.

Charniak, E, Riesbeck, C. K, and McDermott, D. V. *Artificial Intelligence Programming.* Lawrence Erlbaum Associates, Hillsdale, NJ, 1980.

Clark, P. Representing arguments as background knowledge for the justification of case-based inferences. In *Proceedings of the AAAI-88 Workshop on Case-Based Reasoning*, pages 24–29, August 1988.

Clement, C and Gentner, D. Systematicity as a selection constraint in analogical mapping. In *Proceedings of the Tenth Meeting of the Cognitive Science Society*, Montreal, August 1988.

Clement, J. Analogy generation in scientific problem solving. In *Proceedings of the Third Meeting of the Cognitive Science Society*, August 1981.

Clement, J. Spontaneous analogies in problem solving: The progressive construction of mental models. In *Paper presented at AERA*, New York, 1982.

Collins, A and Gentner, D. How people construct mental models. In D Holland and N Quinn (Eds.), *Cultural Models in Language and Thought*, Cambridge University Press, New York, 1987.

Davies, T. R and Russell, S. J. A logical approach to reasoning by analogy. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 264–270, August 1987.

DeCoste, D. *Dynamic Across-Time Measurement Interpretation: Maintaining Qualitative Understandings of Physical System Behavior.* Master's thesis, University of Illinois at Urbana-Champaign, 1989. (in preparation).

DeJong, G and Mooney, R. Explanation-based learning: An alternative view. *Machine Learning*, **1**(2), 1986.

DeJong, G. An overview of the FRUMP system. In W Lehnert and M Ringle (Eds.), *Strategies for Natural Language Processing*, Lawrence Erlbaum and Associates, Hillsdale, NJ, 1982.

de Kleer, J and Williams, B. Diagnosing multiple faults. *Artificial Intelligence*, **32**, 1987.

de Kleer, J. An assumption-based TMS. *Artificial Intelligence*, 28(2), March 1986.

de Kleer, J. Problem solving with the ATMS. *Artificial Intelligence*, **28**(2), March 1986.

Dietterich, T and Buchanan, B. G. The role of experiments in theory formation. In *Proceedings of the Second International Workshop on Machine Learning*, Monticello, IL, June 1983.

Dietterich, T. Learning at the knowledge level. *Machine Learning*, 1(3):287–316, 1986.

Dreistadt, R. An analysis of the use of analogies and metaphors in science. *The Journal of Psychology*, 68:97–116, 1968.

Dreistadt, R. The use of analogies and incubation in obtaining insights in creative problem solving. *The Journal of Psychology*, 71:159–175, 1969.

Einstein, A and Infeld, L. *The Evolution of Physics: From Early Concepts to Relativity and Quanta.* Simon and Schuster, 1938.

Falkenhainer, B and Forbus, K. D. Setting up large-scale qualitative models. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 301–306, St. Paul, MN, August 1988.

Falkenhainer, B and Michalski, R. Integrating quantitative and qualitative discovery: The ABACUS system. *Machine Learning*, 1(4):367–401, 1986.

Falkenhainer, B and Rajamoney, S. The interdependencies of theory formation, revision, and experimentation. In *Proceedings of the Fifth International Conference on Machine Learning*, pages 353–366, Ann Arbor, MI, June 1988.

Falkenhainer, B. Proportionality graphs, units analysis, and domain constraints: Improving the power and efficiency of the scientific discovery process. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 552–554, Los Angeles, CA, August 1985.

Falkenhainer, B. *An Examination of the Third Stage in the Analogy Process: Verification-Based Analogical Learning.* Technical Report UIUCDCS-R-86-1302, Department of Computer Science, University of Illinois, October 1986. A summary appears in *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, 1987.

Falkenhainer, B. *The SME User's Guide.* Technical Report UIUCDCS-R-88-1421, University of Illinois at Urbana-Champaign, September 1988.

Falkenhainer, B. Towards a general-purpose belief maintenance system. In J. F Lemmer and L. N Kanal (Eds.), *Uncertainty in Artificial Intelligence 2*, pages 125–131, Elsevier Science Publishers B.V. (North-Holland), 1988. (Also appears as University of Illinois Technical Report UIUCDCS-R-87-1329, April 1987).

Falkenhainer, B. The utility of difference-based reasoning. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 530–535, St. Paul, MN, August 1988.

Falkenhainer, B, Forbus, K. D, and Gentner, D. The structure-mapping engine. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 272–277, August 1986.

Falkenhainer, B, Forbus, K. D, and Gentner, D. *The Structure-Mapping Engine: Algorithm and Examples.* Technical Report UIUCDCS-R-87-1361, University of Illinois at Urbana-Champaign, July 1987. To appear in *Artificial Intelligence*, spring, 1989.

Forbus, K. D and Gentner, D. Learning physical domains: Towards a theoretical framework. In *Proceedings of the Second International Workshop on Machine Learning*, Monticello, IL, June 1983. A revised version appears in *Machine Learning: An Artificial Approach Vol. II*, R.S. Michalski, J.G. Carbonell, and T.M. Mitchell (Eds.), Morgan Kaufmann, 1986.

Forbus, K. D. Qualitative reasoning about physical processes. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, August 1981.

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

Forbus, K. D. Interpreting measurements of physical systems. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 113–117, August 1986. (Technical Report UIUCDCS-R-86-1248, University of Illinois, 1986).

Forbus, K. D. *The Qualitative Process Engine.* Technical Report UIUCDCS-R-86-1288, University of Illinois at Urbana-Champaign, December 1986.

Forbus, K. D. The logic of occurrence. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, August 1987. (Technical Report UIUCDCS-R-86-1300, University of Illinois, December 1986).

Forbus, K, Nielsen, P, and Faltings, B. The inferential structure of qualitative kinematics. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Milan, Italy, August 1987.

Gentner, D and Landers, R. Analogical reminding: A good match is hard to find. In *Proceedings of the International Conference on Systems, Man and Cybernetics*, Tucson, AZ, 1985.

Gentner, D and Stevens, A. L (Eds.). *Mental Models.* Lawrence Erlbaum Associates, Hillsdale, NJ, 1983.

Gentner, D. *The Structure of Analogical Models in Science.* Technical Report BBN Technical Report No. 4451, Bolt Beranek and Newman Inc., Cambridge, MA., 1980.

Gentner, D. Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, 7(2):155–170, April-June 1983.

Gentner, D. *Analogical Inference and Analogical Access.* Technical Report UIUCDCS-R-87-1365, University of Illinois, August 1987. (Appears in A. Preiditis (Ed.), *Analogica: Proceedings of the First Workshop on Analogical Reasoning*).

Gentner, D. Mechanisms of analogical learning. In S Vosniadou and A Ortony (Eds.), *Similarity and Analogical Reasoning*, Cambridge University Press, London, 1988.

Gentner, D, Falkenhainer, B, and Skorstad, J. Metaphor: The good, the bad and the ugly. In *Proceedings of the Third Conference on Theoretical Issues in Natural Language Processing*, Las Cruces, NM, January 1987.

Greiner, R. *Learning by Understanding Analogies.* PhD thesis, Stanford University, 1986.

Greiner, R. Learning by understanding analogies. *Artificial Intelligence*, **35**(1):81–125, 1988.

Hall, R. P. Computational approaches to analogical reasoning: A comparative analysis. *Artificial Intelligence, to appear.* (Also Technical Report 86-11, University of California at Irvine, Irvine Computational Intelligence Project, May, 1986).

Hanson, N. R. *Patterns of Discovery.* Cambridge University Press, London, 1958.

Hayes, P. J. In defence of logic. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pages 559–565, 1977.

Hayes, P. J. The naive physics manifesto. In D Michie (Ed.), *Expert Systems in the Micro-Electronic Age*, Edinburgh University Press, 1979.

Hayes-Roth, F and McDermott, J. An interference matching technique for inducing abstractions. *Communications of ACM*, **21**(5):401–411, 1978.

Hesse, M. B. *Models and Analogies in Science.* Notre Dame Press, Notre Dame, IN, 1966.

Hoff, B, Michalski, R. S, and Stepp, R. E. *Induce 2 - A Program for Learning Structural Descriptions from Examples.* Technical Report 82-5, University of Illinois at Urbana-Champaign, Intelligent Systems Group, October 1982.

Hogge, J. *The Compilation of Planning Operators from Qualitative Process Theory Models.* Master's thesis, University of Illinois at Urbana-Champaign, September 1987. (Technical Report UIUCDCS-R-87-1368).

Hogge, J. Compiling plan operators from domains expressed in qualitative process theory. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, July 1987.

Hogge, J. *TPLAN: A Temporal Interval-based Planner with Novel Extensions.* Technical Report UIUCDCS-R-87-1367, University of Illinois at Urbana-Champaign, September 1987.

Hogge, J. Prevention techniques for a temporal planner. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 43–48, August 1988.

Holyoak, K. J and Thagard, P. Analogical mapping by constraint satisfaction. *Submitted for Publication*, June 1988.

Holyoak, K. J and Thagard, P. A computational model of analogical problem solving. In S Vosniadou and A Ortony (Eds.), *Similarity and Analogical Reasoning*, Cambridge University Press, London, 1988.

Holyoak, K. J. The pragmatics of analogical transfer. In G. H Bower (Ed.), *The Psychology of Learning and Motivation*, Academic Press, New York, 1984.

Indurkhya, B. Approximate semantic transference: A computational theory of metaphors and analogies. *Cognitive Science*, 11:445–480, 1987.

Josephson, J. R, Chandrasekaran, B, Smith, J. W, and Tanner, M. C. A mechanism for forming composite explanatory hypotheses. *IEEE Transactions on Systems, Man, and Cybernetics*, 17(3):445–454, May/June 1987.

Joskowicz, L. Shape and function in mechanical devices. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, July 1987.

Kass, A. Modifying explanations to understand stories. In *Proceedings of the Eighth Meeting of the Cognitive Science Society*, pages 691–696, August 1986.

Kass, A, Leake, D, and Owens, C. SWALE, A program that explains. In R Schank (Ed.), *Explanation Patterns: Understanding Mechanically and Creatively*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1986.

Kaufmann, G. *Visual Imagery and Its Relation to Problem Solving.* Columbia University Press, New York, 1979. (PhD Dissertation, Universitetsforlaget, Oslo, Norway).

Kedar-Cabelli, S. T. *Analogy: From a Unified Perspective.* Technical Report ML-TR-3, Rutgers University, December 1985.

Kedar-Cabelli, S. T. Purpose-directed analogy. In *Proceedings of the Seventh Meeting of the Cognitive Science Society*, August 1985.

Kedar-Cabelli, S. T. *Formulating Concepts and Analogies According to Purpose.* PhD thesis, Rutgers University, May 1988. (Technical Report ML-TR-26).

Kline, P. J. *Computing the Similarity of Structured Objects by Means of a Heuristic Search for Correspondences.* PhD thesis, Department of Psychology, University of Michigan, 1983.

Kling, R. E. A paradigm for reasoning by analogy. *Artificial Intelligence*, 2:147–178, 1971.

Kolodner, J. *Retrieval and Organizational Strategies in Conceptual Memory: A Computer Model.* Lawrence Erlbaum Associates, Hillsdale, NJ, 1984.

Kolodner, J, Simpson, R. L, and Sycara-Cyranski, K. A process model of cased-based reasoning in problem solving. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, August 1985.

Kosslyn, S. M. *Image and Mind.* Harvard University Press, Cambridge, 1980.

Langley, P and Jones, R. *A Computational Model of Scientific Insight.* Technical Report 87-01, Department of Computer Science, University of California, Irvine, December 1986.

Langley, P and Jones, R. A theory of scientific problem solving. In *Proceedings of the Tenth Meeting of the Cognitive Science Society*, Montreal, August 1988.

Langley, P. Data-driven discovery of physical laws. *Cognitive Science*, **5**:31–54, 1981.

Langley, P, Simon, H. A, Bradshaw, G. B, and Zytkow, J. M. *Scientific Discovery: Computational Explorations of the Creative Processes.* MIT Press, Cabridge, MA, 1987.

Leake, D. B and Owens, C. C. Organizing memory for explanation. In *Proceedings of the Eighth Meeting of the Cognitive Science Society*, pages 710–715, August 1986.

Leatherdale, W. H. *The Role of Analogy, Model and Metaphor in Science.* North-Holland, 1974.

Michalski, R. S. Pattern recognition as rule-guided inductive inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-2**(4):349–361, 1980.

Michalski, R. S. A theory and methodology of inductive learning. In R Michalski, J Carbonell, and T Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach*, Morgan Kaufmann, 1983.

Miller, A. I. *Imagery in Scientific Thought.* MIT Press, Cambridge, MA, 1986.

Mitchell, T. M, Utgoff, P. E, and Banerji, R. Learning by experimentation: Acquiring and refining problem-solving heuristics. In *Machine Learning: An Artificial Intelligence Approach*, pages 163–190, Morgan Kaufmann Publishers, Inc., 1983.

Mitchell, T, Keller, R, and Kedar-Cabelli, S. Explanation-based generalization: A unifying view. *Machine Learning*, **1**(1), 1986.

Mooney, R. *A General Explanation-Based Learning Mechanism and its Application to Narrative Understanding.* PhD thesis, University of Illinois at Urbana-Champaign, December 1987.

Newton, I. *Mathematical Principles of Natural Philosophy and His System of the World, Volume Two: The System of the World.* University of California Press, Berkeley, CA, 1729. (Translated into English by Andrew Motte in 1729. Translation revised by Florian Cajori, 1934.).

Nielsen, P. *A Qualitative Approach to Rigid Body Mechanics.* PhD thesis, University of Illinois at Urbana-Champaign, September 1988.

Oppenheimer, R. Analogy in science. *American Psychologist*, **11**:127–135, 1956.

Pearl, J. Embracing causality in formal reasoning. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, July 1987.

Pople, H. On the mechanization of abductive logic. In *Proceedings of the Third International Joint Conference on Artificial Intelligence*, pages 147–152, 1973.

Pople, H. E. The formation of composite hypotheses in diagnostic problem solving: An exercise in synthetic reasoning. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, August 1977.

Quinlan, R. Learning efficient classification procedures and their application to chess end games. In R Michalski, J Carbonell, and T Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach*, Morgan Kaufmann, 1983.

Rajamoney, S. Experimentation-based theory revision. In *Proceedings of the 1988 AAAI Spring Symposium Series: EBL*, Stanford, CA, March 1988.

Rajamoney, S. *Explanation-Based Theory Revision: An Approach to the Problems of Incomplete and Incorrect Theories.* PhD thesis, University of Illinois at Urbana-Champaign, December 1988.

Rajamoney, S, DeJong, G, and Faltings, B. Towards a model of conceptual knowledge acquisition through directed experimentation. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, August 1985.

Rattermann, M. J and Gentner, D. Analogy and similarity: Determinants of accessibility and inferential soundness. In *Proceedings of the Ninth Meeting of the Cognitive Science Society*, Seattle, WA, July 1987.

Reggia, J. A. Diagnostic expert systems based on a set covering model. *International Journal of Man-Machine Studies*, **19**:437–460, November 1983.

Reiter, R. On closed world data bases. In H Gallaire and J Minker (Eds.), *Logic and Data Bases*, Plenum Press, New York, 1978.

Roller, D. The early development of the concepts of temperature and heat. In *Harvard Case Histories in Experimental Science, Volume 3*, Harvard University Press, Cambridge, MA, 1961.

Rose, D and Langley, P. Chemical discovery as belief revision. *Machine Learning*, **1**(4):423–451, 1986.

Ross, B. H. Remindings and their effects in learning a cognitive skill. *Cognitive Psychology*, **16**:371–416, 1984.

Rumelhart, D. E and Norman, D. A. Analogical processes in learning. In J. R Anderson (Ed.), *Cognitive Skills and Their Acquisition*, Lawrence Erlbaum & Associates, Hillsdale, NJ, 1981.

Russell, S. J. Analogy and single-instance generalization. In *Proceedings of the Fourth International Workshop on Machine Learning*, June 1987.

Shepard, R. N and Cooper, L. A. *Mental Images and their Transformations.* MIT Press, Cambridge, 1982.

Shive, J. N and Weber, R. L. *Similarities in Physics.* Wiley-Interscience, New York, 1982.

Simmons, R and Davis, R. Generate, test and debug: combining associational rules and causal models. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 1071–1078, August 1987.

Simmons, R. G. A theory of debugging plans and interpretations. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 94–99, St. Paul, MN, August 1988.

Skorstad, J. *A Structural Approach to Abstraction Processes During Concept Learning.* Master's thesis, University of Illinois at Urbana-Champaign, (*in preparation*), 1989.

Skorstad, J, Falkenhainer, B, and Gentner, D. Analogical processing: A simulation and empirical corroboration. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, Seattle, July 1987.

Skorstad, J, Gentner, D, and Medin, D. Abstraction processes during concept learning: A structural view. In *Proceedings of the Tenth Meeting of the Cognitive Science Society*, Montreal, August 1988.

Smith, R, Winston, P, Mitchell, T, and Buchanan, B. Representation and use of explicit justifications for knowledge base refinement. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, August 1985.

Suchman, L. A. *Plans and Situated Actions. The Problem of Human Machine Communication.* Cambridge University Press, Cambridge, 1987.

Szolovits, P. *Artificial Intelligence in Medicine.* Westview Press, Boulder, CO, 1982.

Thagard, P and Holyoak, K. Discovering the wave theory of sound: Inductive inference in the context of problem solving. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, August 1985.

Thagard, P. *Computational Philosophy of Science.* MIT Press, Cambridge, MA, 1987.

Thagard, P. *Explanatory Coherence.* Technical Report CSL 16, Princeton University, March 1988.

Thomson, W. T. *Mechanical Vibrations.* Prentice-Hall, Inc, New York, 1948.

Walker, J. *The Flying Circus of Physics with Answers.* John Wiley & Sons, New York, 1975.

Waltz, D. Toward a detailed model of processing for language describing the physical world. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vancouver, Canada, August 1981.

Weld, D. Comparative analysis. *Artificial Intelligence*, **36**:333–373, 1988.

Wellsch, K and Jones, M. Computational analogy. In *Proceedings of the Seventh European Conference on Artificial Intelligence (ECAI)*, pages 153–162, July 1986.

Williams, M. D, Hollan, J. D, and Stevens, A. L. Human reasoning about a simple physical system. In D Gentner and A. L Stevens (Eds.), *Mental Models*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1983.

Winston, P. H. Learning and reasoning by analogy. *Communications of ACM*, **23**(12), December 1980.

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

Winston, P. H. *Artificial Intelligence, Second Edition.* Addison Wesley, Reading, MA, 1984.

Winston, P. H, Binford, T. O, Katz, B, and Lowry, M. Learning physical descriptions from functional definitions, examples, and precedents. In *Proceedings of the Third National Conference on Artificial Intelligence*, pages 433–439, 1983.

Wiser, M and Carey, S. When heat and temperature were one. In D Gentner and A. L Stevens (Eds.), *Mental Models*, Lawrence Erlbaum Associates, Hillsdale, NJ, 1983.

Zytkow, J. M and Simon, H. A. A theory of historical discovery: The construction of componential models. *Machine Learning*, **1**(1):107–136, 1986.

Vita

Brian Carl Falkenhainer was born on July 23, 1960 at Clark USAFB in the Philippines. He attended Santa Clara University, where he received a B.S. degree in Engineering Physics in 1982. In the fall of 1983 he entered the graduate program in Computer Science at the University of Illinois and spent the next two years as a research assistant for Professor Ryszard Michalski. He received the M.S. degree in Computer Science for the development of ABACUS, a program which discovered quantitative empirical laws characterizing a set of observed data.

In 1985 he became a research assistant for Professor Ken Forbus in the Qualitative Reasoning Group and began investigating analogical learning and qualitative physics. He received the Ph.D. in Computer Science from the University of Illinois at Urbana-Champaign in January 1989. His dissertation explored analogical learning and its use in hypothesizing causal explanations for observed physical phenomena.

He has co-authored an article on empirical discovery for the *Machine Learning* journal and an article on analogical mapping for the *Artificial Intelligence* journal. He has also presented papers in the areas of scientific discovery, analogical reasoning, and probabilistic truth maintenance for several scientific conferences on artificial intelligence.

He is currently a member of the research staff at the Xerox Palo Alto Research Center.