Automated Rationalization

Kevin Knight
knight@cs.cmu.edu

Yolanda Gil
yg@cs.cmu.edu

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Connectionist and symbolic AI techniques have different strengths and weaknesses. Connectionist representations are easily learned, but are often impenetrable. Symbolic systems, on the other hand, are able to explain their behavior to humans by reporting chains of reasoning. There have been several proposals recently for combining these approaches. This paper presents a radical proposal. For a given task, we build two independent modules, a reasoner and a rationalizer. The reasoner uses neural or statistical learning techniques to achieve the highest possible performance on the task. The rationalizer's job is to produce symbolic explanations of whatever decisions the reasoner might make, without recourse to the actual decision procedure. We describe how to combine these modules, and we present some psychological and engineering motivations for this approach. Because the reasoner and the rationalizer perform fundamentally different tasks, they require different learning strategies. We characterize the types of domains for which this approach is appropriate and present two sample rationalizers.

Key words: Explanation, rationalization, neural networks, hybrid systems.

1 Introduction

Connectionist techniques for representation and learning have been steadily gaining adherents, and now outperform other techniques on many real-world tasks. One advantage of connectionist representations is that they seem to be more learnable than their symbolic counterparts. Algorithms like back-propagation (Rumelhart *et al.*, 1986) learn input-output mappings from examples and yield good generalization performance. The downside is that learning is usually very slow, and more importantly, the result is an impenetrable mass of connection weights. It is often impossible to state concisely how a particular neural network arrives at its conclusions. Similar strengths and weaknesses appear in other kinds of statistical modeling.

On the other hand, symbolic reasoning techniques are very good at producing explanations of how input data is mapped onto output. One of the first examples of an explanation program was TEIRESIAS (Davis, 1977), which answered questions about the behavior of the MYCIN expert system (Shortliffe, 1976). It could explain its final and intermediate conclusions by backchaining on the MYCIN rules it employed, and it could explain its requests for data by displaying the rules it was trying to match. More recent work on explanation includes SALT (Marcus and McDermott, 1989).

There have been numerous proposals for hybrid architectures. Fu (1991) gives an algorithm for extracting symbolic rules from a connectionist network after learning. This approach is worth pursuing for small networks, but we believe it will prove unwieldy for large networks whose representations are highly distributed, i.e., those in which every processing element participates in every decision. Another hybrid approach is to initialize a system with roughly correct symbolic information, then fine-tune the system with connectionist learning. This has been done in both robotic manipulation (Handelman *et al.*, 1989) and classification (Shavlik and Towell, 1989) tasks. Such an approach addresses the speed of learning problem, providing improvements over raw backpropagation. However, the result is

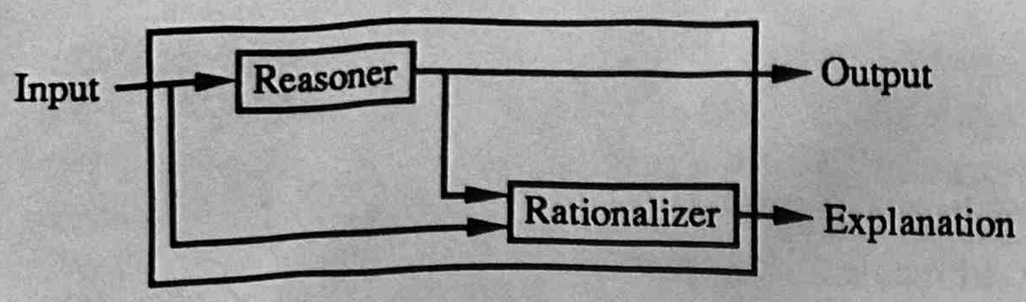

Figure 1: *Automated Rationalization. The reasoner is built for performance, while the rationalizer is built for coverage and believability.*

still an uninterpretable mass of connection weights.

Despite these drawbacks, there are often no alternatives that yield equally high task performance. As Hinton (1990) points out:

> If . . . the large set of weights performs consistently better than an alternative system that can explain its reasoning, it might be better to settle for the system that works best.

This paper addresses the problem of combining the advantages of subsymbolic and symbolic reasoning methods, in order to yield high-performance learning systems that can provide explanations for their decisions. The rest of this paper is organized as follows. The next section presents a system for automated rationalization and motivations for it. Section 3 gives an example of a reasoner and rationalizer for a simple domain. Section 4 discusses the merits of our approach in different types of domains. Section 5 turns to a more complex, user-modeling task. The final section concludes with some discussion.

2 Automated Rationalization

We propose to create, for a given problem, two independent modules: a *reasoner* and a *rationalizer*. The reasoner takes some input and maps it to an output using whatever techniques it can to achieve highest performance. For our applications, these will be connectionist, statistical, or otherwise highly distributed learning techniques. The rationalizer takes both the input to the reasoner and its output—the rationalizer's job is to produce a plausible explanation of why the reasoner mapped the input onto that particular output. However, the rationalizer is not allowed to look inside the reasoner. The organization is shown in Figure 1. The two modules are completely independent—they can be built by separate teams of programmers, for example.

The sole criterion for the reasoner is performance on the task. There are two criteria for the rationalizer: coverage (how many input/output pairs it can explain or "explain away") and believability (how plausible or palatable its explanations are to human users of the system).

As for coverage, we distinguish between two types of rationalizers. A *general-purpose* rationalizer is able to explain *any* input/output pair. A general-purpose rationalizer is completely portable with respect to reasoners, since it can explain any behavior. A *special-purpose* rationalizer only needs to explain the subset of input/output pairs actually producible from the specific reasoner it is hooked up to. In one sense, special-purpose rationalizers are easier to build, since they do not have to explain outlandish behaviors. Yet they are more problematic in that it may be hard to exactly characterize the subset of possible behaviors produced by a given connectionist network.

The advantage of this organization is that it can produce both high-performance reasoning and explanations. We have also produced two learning problems, one for the reasoner and one for the rationalizer. While the reasoner will typically use some neural or statistical learning method, improving the performance of the rationalizer is somewhat more difficult—we take up that topic in Section 6.

Our model draws its inspiration from both psychological and engineering concerns, which we turn to next.

Engineering Motivations

From an engineering point of view, we have two design requirements: have good performance and produce good explanations of the results. Our proposal is modular, in that it breaks the task down along these lines. By considering the problems separately, we can concentrate on what makes a good explanation independently of how to find the right answer.

In many domains, it is theoretically possible to construct sound, complete, and consistent symbolic theories, but it is practically very difficult. Automatic learning systems help to alleviate the knowledge acquisition problem, but often do not represent knowledge in an understandable fashion. Rather than throwing away our unsound, incomplete, and inconsistent symbolic knowledge, we propose to make full use of it in the rationalizer.

An important advantage of our organization is portability. A general-purpose rationalizer is portable to

any reasoner that solves the task. If a new statistical technique comes along, we can completely redo our reasoner without having to reconsider our explanation facility. This type of modularity is not found in other hybrid connectionist-symbolic architectures, e.g., those that draw symbolic rules out of a backpropagation network.

Psychological Motivations

We have also been inspired to a degree by psychological phenomena. We humans often find ourselves constructing symbolic, logical explanations for decisions after we have made those decisions. Much of what we do is without explicit reasoning, or symbolic reasoning, and yet we insist on having explanations for all of our behaviors.

Social psychology supplies a wealth of information on human rationalization. For example, the theory of *cognitive dissonance* (Festinger, 1957), in which an individual's beliefs are modified in order to line up with his actions, is well supported by thirty years of experimental evidence.

Even more interesting phenomena come from neuropsychology, particularly split-brain research. A subject whose *corpus collosum* has been severed, so that the left hemisphere of his brain cannot communicate with the right, can be induced to display amazing powers of rationalization. In one experiment (Gazzaniga, 1985), a subject is briefly shown a divided picture: the left side of the picture contains a snowy scene, and the right side contains a chicken's claw. The snowy scene is available only to the right brain, while the chicken claw is available only to the left. The subject's task is then to choose matching pictures from any array of cards in front of him. Subjects typically solve the task correctly. For example, the right brain guides one hand to rest on the picture of a snow shovel, while the left brain picks out a chicken's head. However, when a subject is asked to explain his answers, only the left brain is able to issue verbal comments. Gazzaniga (1985) reports:

> ...I asked [the subject], "Paul, why did you do that?" Paul looked up and without a moment's hesitation said from his left hemisphere, "Oh, that's easy. The chicken claw goes with the chicken and you need a shovel to clean out the chicken shed."
>
> Here was the left-half brain having to explain why the left half was pointing to a shovel when the only picture it saw was a claw ... The left-brain's cognitive system needed a theory and instantly supplied one that made sense given the information it had on the particular task. It is hard to describe the spell-binding power of seeing such things ... [p. 72]
>
> In short, our species has a special brain component I will call the "interpreter." ...This special interpreter accommodates and instantly constructs a theory to explain why the behavior has occurred. [p. 5]

	Cups					Non-cups				
Features	c_1	c_2	c_3	c_4	c_5	n_1	n_2	n_3	n_4	n_5
Has handle	✓			✓	✓		✓			
Handle on top							✓			
Handle on side	✓			✓	✓					
Bottom is flat	✓	✓	✓	✓	✓	✓	✓		✓	✓
Has concavity	✓	✓	✓	✓	✓		✓	✓	✓	✓
Concavity points up	✓	✓	✓	✓	✓		✓	✓	✓	
Light	✓	✓	✓	✓	✓	✓	✓		✓	
Made of ceramic	✓							✓		
Made of styrofoam		✓	✓			✓				✓
Made of paper				✓	✓		✓		✓	
Expensive	✓		✓		✓		✓		✓	
Fragile	✓	✓			✓	✓		✓		✓

Figure 2: *The Extended Cups Domain (Shavlik and Towell, 1989)*

While we are not making or supporting any psychological claims here it is interesting to note the similarity between Gazzaniga's "interpreter," and our rationalizer.

3 A Simple Example

This section shows how we can apply our ideas to a simple domain. The problem is the *extended cups domain*, a classification task introduced by Shavlik and Towell (1989). The task is to identify an object as CUP or NOT-CUP, based on twelve basic features. Figure 2 shows ten example objects.

First, we build a reasoner that learns to perform this task from examples. For example, we may use a three-layer backpropagation network. Given an example, the network will provide the correct classification as a CUP or a NOT-CUP. Figure 3 shows one such network, due to Shavlik and Towell (1989).

The rationalizer is a completely independent module that provides the explanations of why an example is a CUP or a NOT-CUP. Figure 4 shows a rationalizer's rule base. The rules are written in Prolog style, and are meant to be used in the given order, with chronological backtracking.

The intuition behind the "prove cup" rules is to be relaxed about what counts as a cup. Rules 1–4

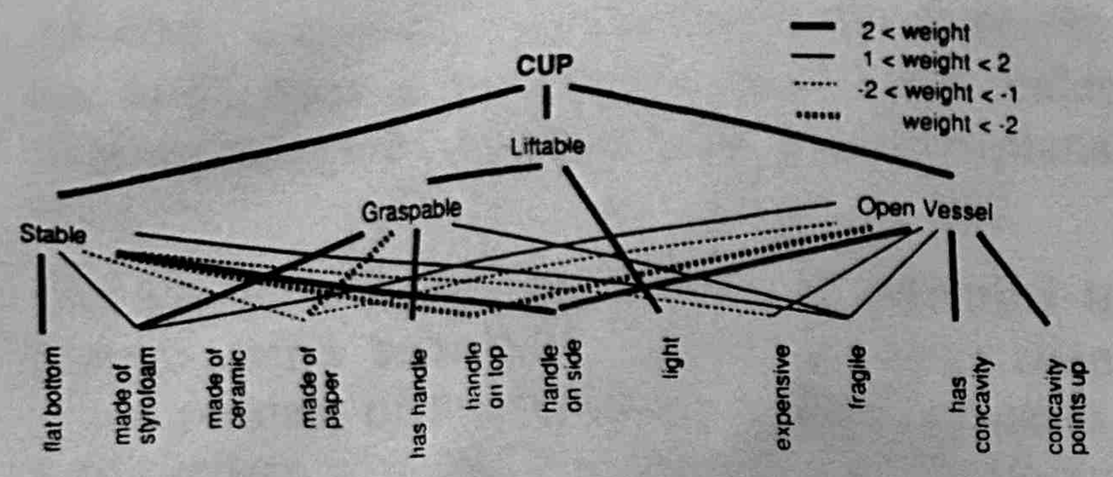

Figure 3: *A Connectionist Network That Solves the Cup Classification Problem (Shavlik and Towell, 1989)*

encode a very general cup definition, one that will probably cover any object classified as a cup by the reasoner. If these rules do not hold, e.g., if the object is not stable, then more relaxed rules are tried. Rule 6 produces an explanation on the order of "it's open and liftable, so it's a cup." These rules are further relaxed until finally Rule 12 insists "it's a cup because it's a *cup*."[1]

Notice that the most useful rationalizations are presented to the user first.

The "prove not-cup" rules are based on the idea of a prototype. Rules 13–16 encode deviations from a specific prototypical cup. The rules are ordered such that most blatant deviations from the prototype are checked first. For example, if an object has a handle across the top, the rationalizer seizes on this fact as persuasive. Otherwise, if the object does not have a concavity, then it is not open, so the rationalizer reports (roughly) "it's not a cup because it's not open," and if the user presses, "it's not a cup because it has no concavity." As before, we must fall back on other rules if these should fail. If we are faced with a very cup-like object which the reasoner has declared NOT-CUP, then we might insist "it's not a cup because cups are made of ceramic" (Rule 23) or "it's not a real cup because it's just not" (Rule 24).

Note that Figure 4 is not a "theory" as normally construed, since it is possible to prove both CUP and NOT-CUP for any object. Such a contradiction would allow a first-order theorem prover to start drawing any conclusions at all, since $p \ \& \ \neg p$ implies everything. Such contradictions are inherent in any general-purpose rationalizer.

While this section has presented a concrete example of automated rationalization, the example is flawed

[1]Here is where a good rationalizer might make use of a general stylistic or diversionary tactics, of the flavor of those employed by the ELIZA program (Weizenbaum, 1966) when it had nothing to go on.

```
 /* prove cup rules */

 (1) cup :- stable, liftable, open.
 (2) stable :- flat.
 (3) liftable :- light.
 (4) open :- has-concavity,
             concavity-points-upward.
 (5) open :- has-concavity.
 (6) cup :- open, liftable.
 (7) cup :- open, stable.
 (8) cup :- liftable, stable.
 (9) cup :- open.
(10) cup :- stable.
(11) cup :- liftable.
(12) cup.

 /* prove not-cup rules */

(13) not-cup :- has-handle-on-top.
(14) not-cup :- not-open.
(15) not-cup :- not-stable.
(16) not-cup :- not(liftable).
(17) not-open :- not(has-concavity).
(18) not-open :-
       not(concavity-points-upward).
(19) not-stable :- not(flat).
(20) not-liftable :- not(light).
(21) not-liftable :- not(graspable).
(22) not-graspable :-
              not(has-handle-on-side).
(23) not-cup :- not(made-of-ceramic).
(24) not-cup.
```

Figure 4: *Rationalizer for the Extended Cups Domain*

in several respects. First, the cups domain does not require much in the way of explanation or interaction with a human user. (Rarely must one answer the question "You say this is a cup ... but *why*?") Second, the domain is small enough that one can look into the connectionist network and report on what it is actually doing. Therefore, our ideas are not particularly suited to the cups domain. Characteristics of appropriate and inappropriate domains are covered in the next section.

4 Good and Bad Domains

This section discusses characteristics of domains for which automated rationalization is appropriate. Attractive characteristics include:

The task involves interaction with human users. We have not yet reached the stage where our programs need to consume rationalizations of their own behaviors. So far, the human monopoly on this ability seems safe.

Users require explanations. There are many such domains: diagnosis, prediction, etc. Basic perceptual tasks, such as recognizing objects and understanding sentences, do not usually require explanations.

The system's explanations are not intended to change user's behaviors. Since the rationalizer is independent of the reasoner, it does not necessarily reflect what the reasoner is actually doing. Therefore its explanations should be aimed only at placating the user, persuading the user, or perhaps allaying the user's fears. An example of a bad domain would be processing credit applications. If a neural network rejects an application, we should not offer a bogus rationalization (e.g., "you are in a high-risk profession"), since the applicant may take it as prescriptive (and demand the credit after changing jobs).[2]

High performance on the task is paramount. We are not willing to trade performance for perspicuity.

The best model of the task is connectionist, statistical, or otherwise highly distributed. In the previous section, we saw how a backpropagation network was used to classify examples. We might have used a small decision tree instead (Very large decision trees, like backpropagation networks, are hard to understand. If a reasoner uses a very large decision tree, a rationalizer might be very useful.) Note that most connectionist systems are successful in low-level, perceptual tasks, which typically do not require explanations. It is not clear whether they are better suited to low-level tasks, or whether their lack of explanatory capabilities keeps them from being applied to higher-level tasks. We are interested in high-level tasks which are best modeled by non-symbolic techniques.

There are usually large amounts of conflicting evidence for any proposition. Having evidence for both p and $\neg p$ gives a rationalizer more room to construct coherent, self-consistent explanations.

There are no recognized experts. If the domain has recognized experts, those experts will (as users) shoot holes through bogus rationalizations, and (as developers) be likely to prefer rule-based reasoners that rival highly distributed models. Some examples of "expert-free" domains are aesthetics, politics, and emotion.

[2]In such cases, we can screen our rationalizations by feeding hypothetical test cases back into the reasoner and observing the results.

Reasoning can be carried out without regard for the palatability of explanations. Consider political decision making. A politician may not be able to "do the right thing" for fear that his constituency will not buy his explanation. In such a case, it is difficult to separate reasoning from explanation, since reasoning must take the form and substance of explanations into account.

Different audiences may require different explanations. If different audiences require (may be persuaded by) different types of explanations, then we can build several rationalization modules, or we can parameterize the rationalization module by characteristics of the user. Unlike the situation in the previous paragraph, here a separation of reasoning and explanation is very useful.

Of course, a domain need not have all of these characteristics for automated rationalization to be of service, but they all help. One type of domain that has many of these characteristics is user-modeling, which we explore next.

5 A User-Modeling Domain

Consider the task of suggesting travel locations for a client, given a large database of client preferences. The database might look like that of Figure 5. Here, clients $c_1, c_2, \ldots c_n$ are listed across the top, and travel locations $t_1, t_2, \ldots t_m$ are listed down the left side. Examples of clients might be *John* and *Mary*; examples of travel locations might be *Waikiki* and *Taipei*. After each trip, clients express their satisfaction with a number from 0 to 100. These numbers (user preferences) appear in the figure. Blank squares indicate locations that a client has never visited.

Given what we know about a client's past preferences, we want to predict how the client will like other locations. In other words, we want to fill in the blank squares as accurately as possible. While this domain is hypothetical, we expect many such databases to be available in the future, and the following discussion is general with respect to all kinds of similar user-modeling tasks, such as recommending novels to read or television shows to watch.

First, we want to build a powerful reasoner that will fill in the blank squares of Figure 5 with high accuracy. There are many ways to approach this problem. One way is to cluster clients into *stereotypes*, which are then used as predictors of preferences. (This was the basic approach of the GRUNDY sys-

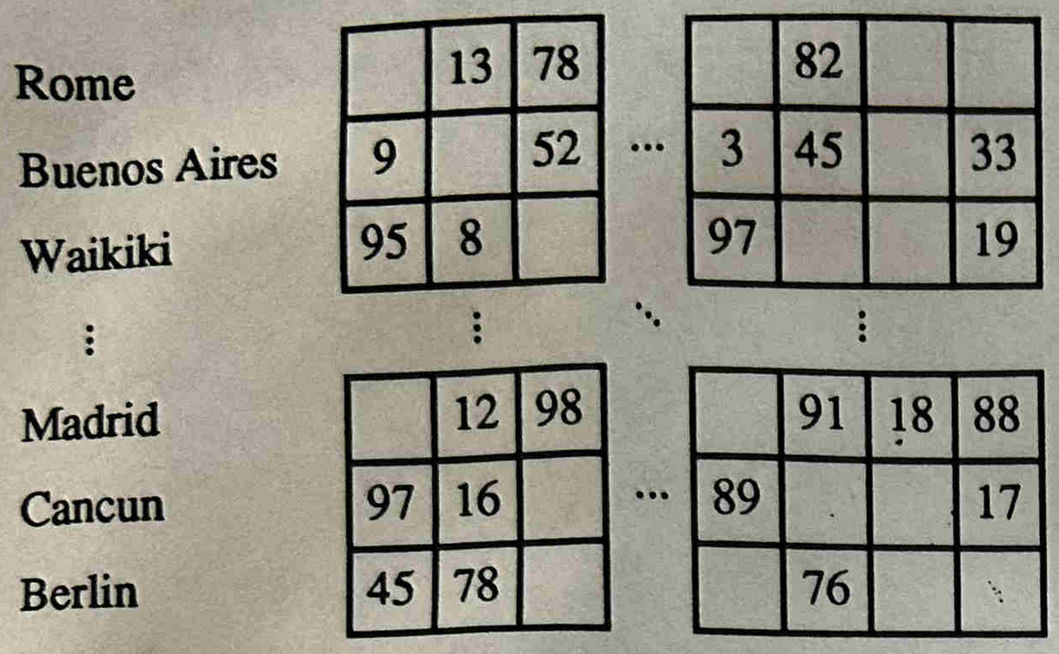

Figure 5: *Client Preference Database for Travel Destinations*

tem (Rich, 1979)). Here, we might use some clustering technique such as AUTOCLASS (Cheeseman *et al.*, 1988), a Bayesian reasoner.

A second way might be to train an auto-associative neural network (e.g., (Peterson, 1991)) on the client vectors $c_1, c_2, \ldots c_n$. The network would act as a content-addressable memory, or pattern completion facility, so that given some stated user preferences, it would fill in the rest.

Yet a third method would be to use statistical measures such as linear correlation. For example, if vectors t_4 and t_7 are highly correlated, then one can be used as a predictor of the other. That is, given sufficient instances of clients who have been to both t_4 and t_7, we can come up with a formula like $t_4 = 0.82 \cdot t_7 + 13$, with a coefficient of correlation $r = 0.79$. *Rome* and *Madrid* might be correlated very highly, *Oslo* and *Athens* correlated negatively, and *Boston* and *Denver* correlated not at all. We can do the same with clients—if we look at the places both c_3 and c_{11} have visited, we might find a high correlation of preferences, such that c_3 is an accurate predictor of c_{11}, and vice-versa. To fill in an empty square, we make use of all filled-in squares in the same row and same column, plugging numbers into linear equations, weighting the results accordingly to the absolute values of the various coefficients of correlation (r).

A critical feature of all of these methods is their distributed nature: potentially, the value of every square in the database is relevant to every other square. This makes compact explanations unlikely. If we forced our reasoner to produce faithful explanations, they might look like this:

1. You and 1536 other clients have been clustered into Traveler Stereotype #761. Members of this stereotype typically enjoy Aruba (estimated value = 91).
2. Our neural network feels that you will enjoy Anchorage, Alaska (estimated value = 86). Hidden unit 132 is especially excited; its activation level is 0.92 due to its positively weighted connections to input units 4 (Buenos Aires), 12 (Newfoundland), . . .
3. Your stated preferences are highly correlated with those of Mary Schexnayder of Butte, Montana. This indicates that you will enjoy Duluth (she did). However, Duluth is correlated positively ($y = 0.6x+20, r = 0.9$) with Buenos Aires, which you did not like (value = 25). Furthermore . . .

When we recommend a travel location, we need to provide persuasive arguments to the user. These arguments are not persuasive.

Yet there is a wealth of symbolic information on the attractive and unattractive features of various travel locations. This information can be gleaned from travel brochures, from client interviews, etc. We suppose that travelers do not consciously pinpoint the actual reasons for liking or disliking certain locations, but do use rationalizations and are likely to believe them. Our neural reasoning system will pick up regularities, trends, and exceptions in the data that people have not got any inkling about. Given that our reasoner gives the highest performance available, we decide to use it, and to develop an independent rationalization model.

The basic task of the rationalizer is to amass evidence both pro- and con- the location in question. If the reasoner predicts a value of 91, say, the rationalizer will present overwhelming positive evidence; if the value is 12, the rationalizer will present overwhelming negative evidence; and if the value is in between, it will present a mix. Consider the following factors, and how they can be used as evidence:

- Climate. Temperate weather a plus. Rain a minus. Heat can be either ("warm, tropical" versus "hot, humid").
- Sightseeing. A plus; list particulars.
- Isolation. Can be a plus or a minus.
- Inoculations. A minus.
- Accomodations. Can be a plus or minus.

- Food. Good food a plus. Bad food a minus.
- Shopping, airfare, etc.

Once we are done, here is how we expect the reasoner and the rationalizer to interact with a human user:

```
USER:  Where should I go on vacation?

COMPUTER:  I recommend Cancun, Mexico.
   ; This answer is produced by the
   ; reasoner, and is based on the
   ; user's past stated preferences.

USER:  Why?

COMPUTER: The beaches are among the
  finest and whitest anywhere.  Airfare
  is cheap.  Mayan ruins, one hour away,
  are a must see.  Tremendous seafood.
  Laid-back atmosphere.
    ; This answer is produced by the
    ; rationalizer, which chooses
    ; positive features of the location.

USER:  What about Mexico City?

COMPUTER:  You would hate Mexico City.
   ; The reasoner produces this
   ; answer based on user's stated
    ; preferences.
It is extremely hot and crowded.
While there are many sightseeing
opportunities, you will not feel
relaxed.  Mexico City lies in a bowl
that is usually filled with smog.
 ; The rationalizer is using
 ; negative features of the location.
```

To the user, there is no visible evidence that a reasoner and a rationalizer are operating independently. The interface appears to be helpful and seamless. Of course, when the next user steps up and inquires about Mexico City, the reasoner may respond more favorably, in which case the rationalizer will tell a completely different story.

Other domains that are attractive form the viewpoint of automated rationalization include:

1. Stock trading. A computerized stock trading advisor will need a rationalization component if it is win over customers.
2. Sports betting. Similarly, a service which sells predictions of sporting event outcomes will need a rationalization component, unless "mysterious computer-based system" is used as a gimmick.

6 Discussion

We have described automated rationalization, a technique for producing high-performance learning systems that can explain their behaviors. In this section, we conclude and expand upon our basic ideas.

Reasoning

It should be clear that what we call the reasoner need not be a connectionist network. It can also be a statistical model, a very large decision tree, or even a case-based reasoner. The important criterion is the difficulty of extracting symbolic explanations from the system. That these systems often outperform easily-understood rule-based systems is a fact of life strangely at odds with Occam's razor, which tells us to seek and prefer simple explanations. Where apparent violations of Occam's razor occur (e.g., (Dietterich and Bakiri, 1991)), our separation of reasoning and explanation looks more attractive.

Rationalizing

A general-purpose rationalizer is a powerful tool that can be ported from one reasoning system to another. It can also be used to explain the behaviors of other agents in the world. One might imagine that given enough knowledge, our rationalizer could eventually serve as a full-fledged reasoner. But this is impossible, since the rationalizer can construct plausible explanations for any decision. Since all decisions look plausible, the rationalizer has no basis on which to choose.

Integration

We have described a simple, loosely-coupled model of integrating reasoning and rationalization. We should also consider more tightly-coupled models. Recall that a special-purpose rationalizer need only explain the input/output pairs generated by a particular reasoner. A special-purpose rationalizer might also look into (say) the weights on connections in the reasoner to get inspiration for an explanation. For example, it might consider only very heavily weighted connections.

Learning

So far, we have said little about where the rationalizer's knowledge comes from. We have only suggested some guidelines for manually constructing

rationalizers (Sections 3 and 5). In this final section, we will speculate on more automatic methods for acquiring knowledge for rationalization.

There are two main resources to draw on: the reasoner and the human user. The reasoner is a powerful source of knowledge about the task. If we allow the rationalizer to present hypothetical inputs to the reasoner, and examine the corresponding outputs, the rationalizer could construct some roughly correct theories about the reasoner. Such experimentation could also be used to identify the relevant features of a particular input. For example, if varying one feature caused the reasoner's output to change dramatically, this feature might be important in an explanation.

To implement a learning method for rationalization, we need some performance yardstick, some way to measure improvement. Unfortunately, this is hard to provide, since it is ultimately the user who (subjectively) decides how plausible a given explanation is. Moreover, different users will respond to different types of explanations. For example, some users will prefer detailed explanations over broad ones, or vice-versa. Some users will respond to certain rhetorical devices or word choices. Some users may be more susceptible to logical argument, others to emotional appeal. In general, the more in tune the rationalizer is with the belief system of the user, the more plausible its explanations will seem.

The learning task, then, becomes one of tailoring explanations to suit a particular user. The problem of learning in human interfaces is a difficult one, but one which has been successfully tackled in the past (Lehman, 1989). We first need some mechanism by which the user can indicate which parts of an explanation he finds persuasive and/or bogus. We can then apply inductive learning and EBL techniques to make hypotheses about why certain explanations fail and others succeed. How to apply these techniques to improve the performance of a rationalizer is a topic for future research.

7 Acknowledgements

This research was sponsored in part by the NSF under contract IRI-8858085, and by the Avionics Laboratory, Air Force Wright Aeronautical Laboratories, Aeronautical Systems Division (AFSC), Wright-Patterson AFB, OH 45433-6543 under Contract F33615-90-C-1465, ARPA Order No. 7597.

References

(Cheeseman *et al.*, 1988) P. Cheeseman, M. Self, J. Kelly, W. Taylor, D. Freeman, and J. Stutz. Bayesian classification. In *Proceedings AAAI-88*, 1988.

(Davis, 1977) R. Davis. Interactive transfer of expertise: Acquisition of new inference rules. In *Proceedings IJCAI-77*, 1977.

(Dietterich and Bakiri, 1991) T. Dietterich and G. Bakiri. Error-correcting output codes: A general method for improving multiclass inductive learning programs. In *Proceedings of AAAI-91*, 1991.

(Festinger, 1957) L. Festinger. *A Theory of Cognitive Dissonance*. Row, Peterson, Evanston, IL, 1957.

(Fu, 1991) L. Fu. Rule learning by searching on adapted nets. In *Proceedings of AAAI-91*, 1991.

(Gazzaniga, 1985) M. S. Gazzaniga. *The Social Brain.* Basic Books, New York, NY, 1985.

(Handelman *et al.*, 1989) D. A. Handelman, S. H. Lane, and J. J. Gelfand. Integrating knowledge-based system and neural network techniques for robotic skill acquisition. In *Proceedings IJCAI-89*, 1989.

(Hinton, 1990) G. Hinton. Preface to the special issue on connectionist symbol processing. *Artificial Intelligence*, 46(1-2), 1990.

(Lehman, 1989) J. F. Lehman. *Adaptive Parsing: Self-extending Natural Language Interfaces.* PhD thesis, Carnegie Mellon University, 1989.

(Marcus and McDermott, 1989) S. Marcus and J. McDermott. SALT: A knowledge acquisition language for propose-and-revise systems. *Artificial Intelligence*, 39(1), 1989.

(Peterson, 1991) C. Peterson. Mean field theory of approximations for neural networks. *Connection Science*, 1991.

(Rich, 1979) E. A. Rich. *Building and Exploiting User Models*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, 1979.

(Rumelhart *et al.*, 1986) D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, editors, *Parallel Distributed Processing*, pages 318–362. MIT Press, Cambridge, MA, 1986.

(Shavlik and Towell, 1989) J. Shavlik and G. Towell. An approach to combining explanation-based and neural learning algorithms. *Connection Science*, 1(3), 1989.

(Shortliffe, 1976) E. H. Shortliffe. *Computer-Based Medical Consultations: MYCIN*. Elsevier, New York, 1976.

(Weizenbaum, 1966) J. Weizenbaum. ELIZA—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–44, Jan. 1966.