Model-Based Case Adaptation*

Eric K. Jones
Victoria University of Wellington
Wellington, New Zealand
eric.jones@comp.vuw.ac.nz

Abstract

In this paper, we demonstrate an important role for model-based reasoning in case adaptation. Model-based reasoning can allow a case-based reasoner to apply cases to a wider range of problems than would otherwise be possible.

We focus on case adaptation in BRAINSTORMER, a planner that uses abstract advice to help it plan in the domain of political and military policy as it relates to terrorism. We show that by equipping a case adapter with an explicit causal model of the planning process, cases presented as advice can be flexibly applied to difficulties that arise at a variety of different stages of planning.

Introduction

Most knowledge-based systems cannot use their prior knowledge flexibly: they cannot use what they already know unless it exactly matches the needs of the current problem. Case-based reasoning has been proposed as a framework for addressing this limitation [Kolodner *et al.*, 1985]. Traditional systems employ a knowledge-poor process of pattern matching or unification to relate prior knowledge to new problems. Case-based reasoning, in contrast, countenances a knowledge-intensive process of bringing prior knowledge to bear, thereby aiming to increase the range of problems that a given knowledge base can address [Kolodner *et al.*, 1985]. A case-based reasoner proceeds by retrieving prior knowledge in the form of a case that may only partly fit the needs of a current problem, then adapting it to resolve any discrepancies with the problem. Because a case adapter can cope with a range of discrepancies, a given case can be applied to a wider range of problems than a conventional system employing only knowledge-poor methods such as unification. In this paper, we identify an important role for model-based reasoning in case adaptation.

We focus on case adaptation in BRAINSTORMER [Jones, 1991b]. BRAINSTORMER is a planner that uses abstract advice to help it plan in the domain of political and military policy as it relates to terrorism. The planner first tries to solve problems it is given on its own; if it gets into trouble, it elicits advice from a user. The user responds with abstract planning advice in the form of a case, which BRAINSTORMER proceeds to adapt to fit the problem, by transforming it into specific, contextualized information that resolves the planner's difficulty. Adaptation in BRAINSTORMER is thus primarily a task of *operationalization* in the sense of [Mostow, 1983]: converting generic knowledge in an abstract vocabulary into specific useful knowledge in an operational vocabulary.

BRAINSTORMER's adapter works to resolve several kinds of discrepancies between advice and planning problems. In this paper we focus on just one of these, which we term *mismatch in stage of the planning process*. See [Jones, 1991b] for a discussion of several others. In the next two sections, we describe the inputs to adaptation and explain what we mean by "mismatch in stage of the planning process." We then outline BRAINSTORMER's approach to resolving mismatches of this kind, which involves reasoning with a causal model of the planning process.

Culturally-Shared Models of Planning

A major goal of our research is to develop instructable systems that can be advised in a high-level, human-like vocabulary [Jones, 1991c]. We start from the belief that people communicate advice about planning in terms of high-level culturally-shared models of the planning process. Models describe planning actions, states they produce, plans, goals, and causal relations between the actions and the states. We have attempted to identify culturally-shared models and to construct a vocabulary sufficient to represent advice expressed in terms of these models.

To this end, many of the examples of advice that BRAINSTORMER handles, including the ones in this paper, are representations of proverbs, encoded in BRAINSTORMER's high-level vocabulary of planning concepts. Proverbs are *culturally-shared cases*: they identify generic strategies that everyone uses to deal with commonly-occurring problems in planning and social interaction. As such, proverbs provide a rich source of data on culturally-shared models of planning [Schank, 1986; White, 1987]. Different proverbs

*This research was conducted at the Institute for the Learning Sciences at Northwestern University, and was supported in part by the Air Force Office of Scientific Research (AFOSR). The Institute for the Learning Sciences was established in 1989 with the support of Andersen Consulting, part of The Arthur Andersen Worldwide Organization.

implicitly presuppose different culturally-shared models. Representing a large number of proverbs has proved an effective strategy for developing and testing our representational vocabulary. It is, however, important to emphasize that we have no special commitment to proverbs other than as a source of data on culturally-shared models of planning.

As we will see, culturally-shared models of planning play two distinct but related roles in BRAINSTORMER: first, they provide a substrate for representing high-level advice; second, schemas encoded in this vocabulary form the declarative component of a model-based reasoning process for transforming high-level advice into operational planner data structures.

The Problem

BRAINSTORMER's cases embody abstract planning advice. It follows that adaptation in BRAINSTORMER presents a challenge that many other systems do not have to face: most cases can be made operational in a number of different ways, each of which impacts a different stage of the planning process. As an example, suppose BRAINSTORMER is attempting to come up with plans for the goal of preventing terrorism and, asking for advice, is presented with the proverb *an old poacher makes the best keeper.* This proverb can be paraphrased as follows: *in a stereotyped attack-defense situation, a former attacker is a good choice for the actor of a plan of defense.* There are a number of different ways this advice might be made operational. Which one is appropriate depends upon what stage of the planning process the planner has reached at the time it requests advice.

Suppose, for example, that BRAINSTORMER is considering a plan of defense, and is searching for candidates for the actor. At this point, the proverb should be interpreted as suggesting *try an ex-terrorist.* Alternatively, suppose the planner is considering a plan of defense against terrorism, and is trying to decide between two plausible candidates for the actor of this plan. The proverb should then be interpreted as suggesting *pick the candidate with the most experience or expertise.* As a third scenario, suppose that the planner is stuck at the first stages of planning and has no concept of how to proceed towards its goal of preventing terrorism. In that event, the proverb could be interpreted as *try a plan of defense with an ex-terrorist actor.* BRAINSTORMER represents each of these interpretations as different operational planner data structures.

In short, the adapter faces a problem of *mismatch in stage of planning process.* Cases the adapter is handed are often initially represented in a form that can be used fairly directly by one stage of the planning process but that must be substantially transformed in order to assist other stages of planning. If BRAINSTORMER is to use cases it is handed as flexibly as possible, it has to be able to carry out the relevant inferences. This turns out to centrally involve reasoning with a causal model of the planning process, as we now describe.

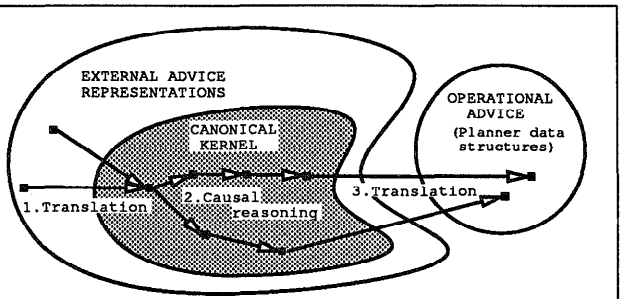

Figure 1: Information flow in the adapter

Overview of the Approach

We begin by distinguishing three separate vocabularies for advice. First is the *external vocabulary* in which the user presents advice. This vocabulary puts as few constraints as possible on the form of the advice: cases can be expressed in terms of a variety of culturally-shared models of planning. Second, we distinguish a privileged subset or kernel of the external vocabulary called the *canonical vocabulary.* This is a vocabulary of planning actions consistent with a particular culturally-shared model of the planning process, the *canonical model.* The canonical model is so called because any action that can be represented in the external vocabulary can also be redescribed in terms of a planning action in the canonical model. The canonical model describes planning in terms of *plan design* and *plan execution,* building on Schank's idea of the goal-plan-action-state chain [Schank, 1986]. Third, there is the operational vocabulary of the planner's data structures, in which the outputs of the adapter are encoded.

BRAINSTORMER is equipped with three kinds of knowledge for transforming advice from one vocabulary to another:

1. Knowledge for translating actions expressed in the external vocabulary into the canonical vocabulary.
2. Knowledge for causal reasoning within the canonical model of the planning process.
3. Rules for translating expressions in the canonical vocabulary into the planner's operational vocabulary.

This knowledge gives rise to the information flow depicted in figure 1. The system first converts external advice into the vocabulary of the canonical model, then variously reexpresses it by causal reasoning within this model; finally, it translates canonical representations into the operational vocabulary of the planner.

This three-stage organization of knowledge is well suited to the task of resolving mismatches between stage of planning process. Causal reasoning within the canonical model (stage 2) allows the system to relate advice to different stages of the planning process. Translating external advice into the canonical vocabulary (stage 1) serves to minimize the size of the knowledge base needed for this causal reasoning. As the figure illustrates, different external expressions of advice can give rise to the same canonical representations, which can then be operated on by identical

```
(def-schema top-down-design =self
   object          =obj
   partial-designs (design-part
                     input  part-spec
                              object =obj
                     output part-spec =pspec
                              object =obj)
   output          spec
                     object     =obj
                     part-specs (=pspec)
   &indices        (partial-designs output)
```

Figure 2: Definition of a **top-down-design** schema.

causal knowledge. Rules at stage 3 further simplify causal reasoning by encapsulating implementation details of BRAINSTORMER's planner.

We now describe how model-based case adaptation is implemented as a process of schema-based reasoning, then we describe the model-based case adaptation process in greater detail.

Modeling the Planning Process

Culturally-shared models of the planning process are represented as collections of *schemas*, which are structured descriptions of planning actions and information that those actions manipulate. Complex actions are represented as partially-ordered collections of subactions linked by key enabling conditions and results.

Schemas are encoded in a frame-based representation language using a slot-filler notation. A typical schema definition is shown in figure 2. This schema is part of the canonical model of the planning process. It describes a process of top-down design, in which a specification of an artifact to be designed is built up by a sequence **design-part** actions, each of which specifies the design of a component of the artifact. This schema can be applied to any top-down design task that does not involve interactions between the design of sub-parts. In particular, simple hierarchical planning in the absence of goal interactions can be described using this schema, as we illustrate below. The **&indices** slot is treated specially; it specifies how the schema is to be indexed in memory.

Logically speaking, schemas are universally quantified implications that relate a schema type to a conjunction of slot-filler assertions. The fillers of slots are existentially quantified, except for "list" slots such as **partial-designs** in figure 2, which can take an indefinite number of fillers and are universally quantified. **Top-down-design**, for example, can be translated as the following first-order formula:

$$\begin{aligned}
&\forall x\ Isa(x, top\text{-}down\text{-}design) \supset \\
&\quad \exists obj, sp[Object(x, obj) \wedge Output(x, sp) \\
&\qquad \wedge Isa(sp, spec) \wedge Object(sp, obj) \wedge \\
&\qquad \forall dp[Partial\text{-}designs(x, dp) \supset \\
&\qquad\quad [Isa(dp, design\text{-}part) \wedge \\
&\qquad\quad \exists p_1, p_2[Input(dp, p_1) \wedge Isa(p_1, part\text{-}spec) \wedge \\
&\qquad\qquad Object(p_1, obj) \wedge Isa(p_2, part\text{-}spec) \wedge \\
&\qquad\qquad Object(p_2, obj) \wedge Part\text{-}specs(sp, p_2)]]]]
\end{aligned}$$

Implementing Model-Based Adaptation

The input to adaptation is a query from the planner and a culturally-shared case that a user has presented as advice. The task of adaptation is to transform the initial representation of the case into an operational planner data structure that answers the query.

Model-based case adaptation is implemented as a process of forward reasoning from advice to queries, an employs a schema-based inference engine that operates over the three kinds of knowledge outlined above. We now describe relevant aspects of the schema-based inference engine, then sketch how it is used to implement the three stages of model-based reasoning.

Schema-Based Reasoning in Brainstormer

Schema-based reasoning involves activating schemas to "explain" a user's advice in terms of planning actions that the advice can assist. The resulting explanations are abstract descriptions of planning actions that the planner could carry out using the advice; the advice fills a slot of the schema. This explanation process is similar in essence to *motivational analysis* as described in [Charniak, 1988], with several extensions.

Four basic inference mechanisms are required for schema-based reasoning: schema activation, slot filling, redescription inference, and "if-added" rules. A schema is *activated* by retrieving it from memory and instantiating it. Schemas are stored in memory in terms of the types of other schemas that they can be plausibly retrieved to explain, as specified by the **&indices** slot of the schema. The **top-down-design** schema shown in figure 2 above, for example, is indexed in terms of the schemas **design-part** and **spec**, corresponding to the **partial-designs** and **output** slots of the schema. A schema is instantiated by creating a new constant, abductively asserting that the schema's type holds of that constant, and then creating skolem terms as prototypical fillers of each non-"list" slot. Similarly, functions for generating prototypical slot fillers on demand are also associated with each "list" slot.

Schemas are activated with the aim of explaining advice from the user. To form an explanation, however, the advice has to be filled into the slot of the instantiated schema that corresponds to the index used to retrieve it. For example, if an instance of a **design-part** is present in the advice, a **top-down-design** schema will be activated to explain it. The explanation is formed by filling the **design-part** into the **partial-designs** slot of the instantiated schema. Slot filling is accomplished by a process of *abductive unification*, in which the representation to be explained is hypothesized to be equal to a prototypical filler of the appropriate slot of the schema that explains it, if this equality is consistent with the system's knowledge of them both [Charniak, 1988].

As an additional complication, a schema can be retrieved to explain advice that does not abductively unify with any of its slot fillers, if the advice can be *redescribed* in a different way that does abductively unify. BRAINSTORMER uses a specialized inference mechanism called *redescription inference* for this purpose;

see [Jones, 1991a] for details. A third inference mechanism, *if-added* inference, allows one to write forward-chaining rules that trigger upon filling a slot.

Using Schema-Based Reasoning

We now have sketched the inferential machinery that BRAINSTORMER uses to implement model-based case adaptation. Adaptation entails a search through a space of hypotheses established by schema instantiation, abductive unification, and redescription inference. Adaptation is successful if a small set of schema instances can be found that link an initial representation of advice to an operational planner data structure that abductively unifies with a query from the planner.

As we discussed above, adaptation proceeds in three stages: conversion to canonical form, causal reasoning, and conversion to operational form (see figure 1). These stages are not distinguished in procedural terms so much as in terms the content of the schemas and inference rules that they manipulate.

At the start of stage 1, if the advice is not already in the form of an recommended action—it might, for example, supply information relevant to performing a planner action—then schemas representing planner actions are activated to explain the advice. Next, if the resulting schema instances are not part of the canonical model, redescription inference is invoked to transform them into canonical representations.

During stage 2 (causal reasoning), schemas in the canonical model of the planning process are activated to provide further explanatory context for the representations produced at stage 1. These schemas describe larger chunks of the planning process as complexes of smaller planning actions linked by key enabling conditions and results. Variable bindings represented using the notation `=<symbol>` provide constraints between the actions and their enabling conditions and results. For example, a variable binding `=pspec` in the `top-down-design` schema of figure 2 constrains the `output` slot to reflect the results of any sub-actions filled into the `partial-designs` slot.

When a slot of a schema instance is filled by advice (or by an schema instance that explains some advice), variable bindings associated with the slot are used to propagate causal implications of the advice to representations of earlier and later phases of the planning process. Suppose, for example, that a `design-part` instance is filled into the `partial-designs` slot of a `top-down-design`. The partial design, or `part-spec`, produced by the `design-part` will be propagated to a representation of the complete design, or `spec`, stored in the `output` slot of the `top-down-design`. Propagation occurs automatically, as a side effect of slot filling.

Stage 3 of adaptation (conversion to operational form) is implemented using "if-added" rules that trigger upon filling slots of schemas in the canonical model.

An Example

In this section, we present an example of case adaptation, in which BRAINSTORMER uses model-based reasoning to resolve a mismatch in stage of the planning process. We elaborate on the third of the scenarios described above, in which the planner gets stuck early in planning, while attempting to retrieve a plan for the goal of preventing terrorism. At that point, the planner issues a query for information to resolve its difficulty; the adapter's task is to transform the advice it is handed into an answer to this query.

```
?plan-for
   plan ?plan
   goal prevent-goal
          state terrorist-attack
          actor Islamic-fundamentalist
```

Figure 3: A query for a plan

```
optimal-evaluation
   object  part-spec
             object     defend-plan
             parameter  actor
             value      attacker-stereotype
                           isa =type
   context goal-conflict
             actor1     attacker-stereotype
                           isa =type
```

Figure 4: Initial representation of the case.

The planner's query asks for a plan for the goal of preventing terrorism, as shown in figure 3. Note that if instead the planner were stuck at a different stage of the planning process, its information requirements would be different, so it would issue a different query.

Once the planner issues the query, a user provides BRAINSTORMER with advice in the form of a case represented in the flexible external vocabulary. In the current example, the user supplies a representation of the proverb *an old poacher makes the best keeper*. We represent this proverb as an *evaluation that can inform a choice*, since the proverb's central point is that old poachers are *best*. See figure 4. This representation can be paraphrased as follows: "in the context of a goal conflict involving a stereotyped attacker, the optimal actor for a plan of defense is someone who satisfies the attacker's stereotype."

The adapter's job is to transform this case representation into an answer to the query of figure 3. The resulting representation is shown in figure 5. It is completely different from the original representation, primarily because it informs an entirely different stage of the planning process. The initial representation of the case provides information relevant to *choosing the actor of a plan*, an action that takes place during plan refinement. The transformed representation, in contrast, provides information that BRAINSTORMER uses during *plan retrieval*, an earlier stage in the planning

```
plan-for
  plan defend-plan
         actor attacker-stereotype
  goal defend-goal
         state stereotyped-attack
```

Figure 5: Output of the adapter

```
choose =ch
 options      (=pspec)
 evaluations (optimal-evaluation
               object part-spec =pspec
                        object    defend-plan
                        parameter actor
                        value attacker-stereotype)
```

Figure 6: An instance of a **choose** schema.

process in which the planner uses features of goals to retrieve appropriate plan schemas. In the remainder of this section, we sketch the reasoning required to effect this transformation.

In the first stage of adaptation, the adapter converts the external advice—the initial representation of the proverb as an **optimal-evaluation**—into planner actions in the canonical vocabulary. The advice is transformed in two steps. First, the adapter activates schemas that represent planning actions, to explain the advice. One schema that is retrieved is **choose**, which represents the idea of choosing between options on the basis of **evaluations** of alternatives. The schema is indexed under **evaluation** (among other indices), as the filler of its **evaluations** slot is a schema of this type. The **choose** schema is retrieved using the **optimal-evaluation** in the advice as a retrieval cue, as this is a subtype of **evaluation**. The resulting explanation is shown in figure 6.

The second step is to redescribe the **choose** action as an action in BRAINSTORMER's canonical vocabulary, in which plan formulation is viewed as a design task. The result is an instance of a **design-part** schema, which feeds into the next stage of adaptation.

During stage 2 of adaptation (causal reasoning within the canonical model), the adapter first activates an overarching **top-down-design** schema to explain the **design-part** schema, then activates a **goal-plan-action-state** schema to explain the **top-down-design**. Each explanation is a schema instance one of whose slots is filled with another schema instance it explains. As a side effect of slot filling, features of the explained schema instance are propagated within the new explanation, establishing key enabling conditions and results. The result of this process is an explanation of the advice in terms of a plausible history of plan construction that starts with the adoption of a hypothetical **defend-goal** and ends when the planner uses the knowledge in the advice to specify the actor of a **defend-plan** for this goal. (See figure 7.)

The final step of adaptation produces an operational planner data structure that answers the planner's query and allows it to continue planning. Recall that the planner originally requested a **plan-for** frame nominating a particular plan for an existing goal to prevent terrorism. A rule indexed on the **plan** and **goal** slots of the **goal-plan-action-state** frame of figure 7 generates the desired **plan-for** frame (figure 5, above), which describes the connection between the fillers of these slots using the operational vocabulary of the planner. This **plan-for** frame answers the planner's original query, and adaptation is complete.

```
goal-plan-action-state
  goal        defend-goal
  plan        defend-plan =obj
  select-plan --
    top-down-design
      object defend-plan =obj
      partial-designs --
        design-part
          input  part-spec
                   object    defend-plan =obj
                   parameter actor
          output part-spec =pspec
                   object    defend-plan =obj
                   parameter actor
                   value     attacker-stereotype
      output spec
               object     =obj
               part-specs (=pspec)
```

Figure 7: An instance of **goal-plan-action-state**.

Discussion and Related Work

One of the key aims of case-based reasoning is to improve the flexibility with which prior knowledge can be used as compared to traditional (*e.g.* schema-based) approaches. Flexibility is measured in terms of the number of different kinds of problems that given prior knowledge can solve within fixed time constraints. If knowledge can be used more flexibly, a smaller knowledge base will provide equivalent problem-solving power.

It follows that the amount of knowledge that adaptation itself requires must be less than the decrease in knowledge base size entailed by adopting a case-based approach. Thus, a theory of adaptation must identify generally-applicable kinds of knowledge for resolving discrepancies between problems and cases retrieved to solve them, and must in particular show that the amount of knowledge required for adaptation increases less than linearly with the size of the case base.

Model-based reasoning in BRAINSTORMER meets this condition. When building BRAINSTORMER, we factored out knowledge for reasoning about the planning process from knowledge the planner needs to solve planning problems. As we have seen, the former is represented as an explicit abstract causal model of the planning process; the latter remains in the case base. The two kinds of knowledge are combined dynamically during adaptation.

The knowledge about the planning process, if not factored out, would have to have been represented over and over again throughout the case base. A traditional schema-based system, for example, would require multiple representations of each of BRAINSTORMER's cases: one (or sometimes several) representations would be required for each stage of the planning process that could conceivably benefit from the case's content. This would substantially increase the size of the overall knowledge base.

Of course, representing the content of cases requires making implicit or explicit reference to some aspect of the model of the planning process: for example, *an old*

poacher makes the best keeper is represented in terms of information relevant to the planning action of choosing a component of a plan. However, the case representer is free to represent cases in terms of any aspect of planning that facilitates exposition. BRAINSTORMER allows this kind of flexibility, because model-based reasoning can resolve discrepancies between the stage of planning referenced by the advice and the needs of any particular planning problem.

The utility of explicit self models has long been recognized in the field of knowledge acquisition. Starting with Teiresias [Davis, 1982], a number of knowledge acquisition systems (*e.g.* ASK [Gruber, 1989] and MOLE [Eshelman *et al.*, 1986]) have employed explicit models of the problem solver under construction to simplify knowledge entry. Knowledge is typically added in response to the system's failing to handle some new example. Unfortunately, new knowledge must be entered in a form that is very close to fully operational. Moreover, failure diagnosis typically requires the user to have a detailed knowledge of reasoning process of the underlying system. (See [Birnbaum *et al.*, 1990] and [Smith *et al.*, 1985], however, for attempts to automate diagnosis completely.) Existing systems employ model-based reasoning to help users manage the complexity of entering data at the operational level, as opposed to facilitating an informal, high-level dialog that sidesteps this complexity.

BRAINSTORMER, in contrast, uses model-based reasoning to relieve the user of the burden of spelling out details of how the advice is to be used, permitting higher-level, more flexible interactions between the user and the planner. [Jones, 1991c] describes BRAINSTORMER's relation to other knowledge acquisition and advice-taking systems in greater detail.

Conclusion

BRAINSTORMER is an exploratory prototype and should be evaluated as such. A complete, practical system that can flexibly employ high-level advice remains a distant prospect; nevertheless, it is important to design experimental systems that explore the hard problems that lie en route.

In this paper, we have discussed one such problem: the need to flexibly relate abstract advice to a wide range of problem-solving situations. A central contribution of our research is to demonstrate an important role for model-based reasoning. Reasoning with an explicit model of the planning process allows given abstract advice to be brought to bear on a variety of different stages of planning. It follows that a case-based reasoner equipped with a model-based reasoning component can adapt abstract knowledge more flexibly. More generally, any system that needs to operationalize advice to assist ongoing problem solving can benefit from reasoning with an explicit causal model of its own problem-solving process.

What are the limitations of our approach? Any conclusions are necessarily tentative. The current system only works on a small number of examples, so the amount of additional knowledge the adapter would need to handle a much wider range of advice is uncertain. Nevertheless, in light of our discussion in the previous section, we suspect that the knowledge required will increase sublinearly in the size of the case base. Might model-based reasoning become intractable as we add more vocabulary? The current system uses a very simple forward-chaining control regimen, which may have to be replaced with some form of means-ends analysis to scale acceptably. These and other issues await further research.

References

Birnbaum, Lawrence; Collins, G.; Freed, M.; and Krulwich, B. 1990. Model-based diagnosis of planning failures. In *Proceedings AAAI-88 Eighth National Conference on Artificial Intelligence*, Boston. AAAI. 318–323.

Charniak, E. 1988. Motivation analysis, abductive unification, and nonmonotonic equality. *Artificial Intelligence* 34:275–295.

Davis, Randall 1982. Teiresias: Applications of meta-level knowlege. In Davis, Randall and Lenat, D.B., editors 1982, *Knowledge-Based Systems in Artificial Intelligence*. McGraw-Hill, New York. 229–484.

Eshelman, Larry and McDermott, J. 1986. MOLE: A knowledge acquisition tool that uses its head. In *Proceedings AAAI-86 Fifth National Conference on Artificial Intelligence*, Philadelphia. AAAI. 950–955.

Gruber, Thomas R. 1989. *Exemplar-Based Knowledge Acquistion*. Academic Press, San Diego.

Jones, Eric K. 1991a. Adapting abstract knowledge. In *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*, Chicago, IL. Lawrence Erlbaum Associates.

Jones, Eric K. 1991b. *The Flexible Use of Abstract Knowledge in Planning*. Ph.D. Dissertation, Yale University.

Jones, Eric K. 1991c. Knowledge refinement using a high-level, non-technical vocabulary. In *Machine Learning: Proceedings of the Eighth International Workshop*, Chicago, IL. Morgan Kaufmann.

Kolodner, Janet L.; Simpson, R.L.; and Sycara-Cyranski, K. 1985. A process model of case-based reasoning in problem-solving. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA. IJCAI, Inc.

Mostow, David J. 1983. Machine transformation of advice into a heuristic search procedure. In Michalski, Ryszard S.; Carbonell, J.G.; and Mitchell, T.M., editors 1983, *Machine Learning: An Artificial Intelligence Approach*. Tioga Publishing Company, Cambridge, MA. 367–404.

Schank, Roger C. 1986. *Explanation Patterns: Understanding Mechanically and Creatively*. Lawrence Erlbaum Associates, Hillsdale, NJ.

Smith, Reid; Winston, H.; Mitchell, T.; and Buchanan, B. 1985. Representation and use of explicit justifications for knowledge-base refinement. In *Proceedings Ninth International Joint Conference on Artificial Intelligence*, Los Angeles. IJCAI, Inc. 675–680.

White, Geoffrey M. 1987. Proverbs and cultural models. In Holland, Dorothy and Quinn, N., editors 1987, *Cultural Models in Language and Thought*. Cambridge University Press, New York. 151–172.