Communicating Abstract Advice: the Role of Stories*

Eric K. Jones
Victoria University of Wellington
P.O. Box 600, Wellington, New Zealand
eric.jones@comp.vuw.ac.nz

Abstract

People often give advice by telling stories. Stories both recommend a course of action and exemplify general conditions in which that recommendation is appropriate. A computational model of advice taking using stories must address two related problems: determining the story's recommendations and appropriateness conditions, and showing that these obtain in the new situation. In this paper, we present an efficient solution to the second problem based on caching the results of the first. Our proposal has been implemented in BRAINSTORMER, a planner that takes abstract advice.

Introduction

People often use stories to give advice. Advice in the form of a story frequently seems more convincing than an unadorned list of instructions [Schank, 1991]. Why should this be so? While there are probably a number of factors at work, it is important not to overlook the obvious: stories are often more convincing in large part because they are more informative. A story not only recommends a course of action, it also exemplifies conditions in which this recommendation is appropriate.

Unfortunately for an advice taker, a story's recommendations and appropriateness conditions may not be explicitly mentioned; even if they are, they may not always be identified as such. It follows that stories present a harder problem of inference than other kinds of advice such as lists of instructions, in which these features are explicit. In addition to the usual problems of operationalization [Mostow, 1983], a story-based advice taker must face the problem of working out which features of the story are relevant to the problem at hand.

To obtain a better understanding of this problem, we have studied a particularly simple class of naturally-occurring stories: familiar, advice-giving proverbs. People often give abstract advice using proverbs; many proverbs are worded in terms of a little story or vignette of a domain-specific situation in which the point of the proverb applies. Examples include *make hay while the sun shines, a bird in the hand is worth two in the bush, a stitch in time saves nine*, and *far from eye, far from heart.*

Proverbs are conventionally associated with an abstract point, which it is the proverb's function to communicate. The abstract point of *make hay while the sun shines*, for example, can be paraphrased as *take advantage of opportunities while they exist.* Knowing a proverb at a minimum requires grasping its abstract point. Proverbs' stories are easier to interpret as advice than than many other kinds of stories, because their abstract points have been computed in advance and stored in memory as part of the representation of the proverb.

Proverbs with stories, however, often contain more information than just their abstract point: their story may also provide a comprehensible exemplar of a specific situation in which the abstract point of the proverb applies. Most people who understand a proverb expressed in terms of a story also understand at least in part how the story illustrates the proverb's abstract point. For example, most people familiar with *make hay while the sun shines* seem to understand that hay making is some kind of goal-directed activity for which sunshine is a precondition, and that the opportunity the proverb refers to is exemplified by sunshine. This understanding is also cached as part of their representation of the proverb.

Even a partial understanding of a proverb's story is usually sufficient to suggest reasons why the abstract point of the proverb might plausibly apply in new situations. Consider, for example, the proverb *far from eye, far from heart*, whose abstract point is that long-distance relationships tend to weaken or dissolve. The proverb's metonymic reference to visual perception (*eye*) suggests the following causal chain in support of its abstract point: if you don't *see* someone regularly, then you aren't reminded of them, so your attachment to them weakens.

In summary, proverbs with stories come with their abstract points precomputed, together with some appropriateness conditions for their recommendations. It follows that the stories of proverbs are somewhat easier to make sense of than other kinds of stories: their interpretation is more tightly con-

*The research described here 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. The Institute for the Learning Sciences was established in 1989 with the support of Andersen Consulting, part of The Arthur Andersen Worldwide Organization.

```
positive-evaluation
   object  scheduling-decision
             object plan =p
             time   time-interval =t
   context (opportunity
              plan   =p
              object =obj
                       time-interval =t)
```

Figure 1: Abstract point of *make hay ...*

```
positive-evaluation
   object  scheduling-decision
             object hay-making-plan =p
             time   time-interval =t
   context (precondition-opportunity
              plan   =p
              object sunshine
                       time-interval  =t)
```

Figure 2: Story of *make hay ...*

strained. An account of how proverbs with stories can be represented and used for advice taking is a first step towards developing a system that can extract advice from a wider range of stories. This paper presents such an account.

Our approach is implemented in BRAINSTORMER, a planner linked to an advice taker [Jones, 1991a; Jones, 1992]. The advice taker elicits advice from a user in the form of a representation of a proverb, and transforms it into an operational planner data structure that help the planner resolve a current problem-solving difficulty. BRAINSTORMER operates in the domain of political and military policy as it relates to terrorism.

Representing Proverbs with Stories

The key problem of representing proverbs with stories is encoding them in ways that highlight their functionally significant features, by which we mean their recommendations and any appropriateness conditions that constrain their application. It is useful to divide these functionally significant features into two categories: essential features and inessential features. Essential features of a proverb's story are aspects of the story that must hold in every situation in which the proverb applies; the union of these features constitutes the proverb's abstract point. Inessential features, when true in a new situation, lend weight to a proverb's recommendation, but need not obtain for the proverb to apply.

We offer a two-part solution to the problem of representing proverbs with stories. First is a notation for highlighting essential features of stories. Second is a way to encode non-standard but functionally significant abstractions of components of stories' representations.

Relating Stories and Abstract Points

Our approach to highlighting essential features can be summarized as follows. We view proverbs as consisting of a story and an abstract point that are structurally related to each other. The essential features of a proverb's story are those shared by its abstract point; therefore, by encoding this structural relationship, we highlight the story's essential features.

Consider, for example, the proverb *make hay while the sun shines*. As shown in figure 1, the proverb's abstract point is the recommendation *take advantage of opportunities when they exist.*[1] The story of *make hay while the sun shines*, on the other hand, concerns a particular kind of opportunity: an opportunity afforded by the satisfaction of a precondition of a hay-making plan. The representation of this story is shown in figure 2.

While the proverb's abstract point must obtain in a new situation for the proverb to apply, the same is not true of its story. Obviously, the new situation need not be about sunshine; perhaps less obviously, it need not involve fortuitous satisfaction of a precondition of a plan. *Make hay while the sun shines* could, for example, be used to advise selling a stock after its price increases; the higher price signifies a good opportunity not because high prices are a precondition for selling, but because selling when prices are high yields a greater profit.

To capture the structural relationship between a proverb's story and its abstract point, we represent the proverb in terms of its story, but with its abstract point superimposed. That is, we begin with representations of its story and its abstract point, and then embed the abstract point in the story. Viewing representations as graph structures [Sowa, 1984], we make the assumption that the abstract point can be represented as a *subgraph* of the story, where possibly some of the nodes in the subgraph are more general than the corresponding nodes in the story. Under this assumption, each frame in the abstract point corresponds to a frame in the story.

To build up a representation of the proverb, we start with a representation of its story, and annotate each frame in the representation as follows:

1. If there is no corresponding frame in the abstract point, we postfix the frame with a ⇑. Thus, for example, a `resource` frame would become `resource`⇑.
2. If there is a corresponding frame in the abstract point, but its type label differs from the story (in

[1] Brainstormer uses a frame-based representation system with a slot-filler notation:

```
<frame>
    <slot1> <filler1>
    <slot2> <filler2>
    ...
```

Equality relationships between frames are encoded using the notation =<symbol>.

```
positive-evaluation
 object  scheduling-decision
           object  hay-making-plan↑plan =p
           time    time-interval =t
 context (precondition-opportunity↑opportunity
           plan    =p
           object sunshine↑entity
                    time-interval  =t
```

Figure 3: Embedding an abstract point in a story.

which case, it must be more general), then we annotate the frame in the story with this more general type label: `sunshine↑entity`, for example.

3. Otherwise, the frame in the story corresponds to a frame in the abstract point, and the type labels of the two frames are the same. In that case, we do nothing.

Figure 3 illustrates the result of applying this process to the proverb *make hay while the sun shines*.

We represent proverbs in this fashion because we want to glean as much information as possible from their stories, yet be free to generalize away inessential parts that do not hold when the proverb is used in a new situation. The abstract point of a proverb is a representation of its essential core claim that must be true in every situation in which it applies. Our chosen representation has the advantage that it can be easily interpreted as instructions to that effect:

- Those parts of the story that overlap with the abstract point are annotated with "↑"s, which can be thought of as instructions about how far they can be generalized. If there is no annotation at all, the story coincides with the abstract point, so no generalization is permitted. If the annotation is of the form $x\uparrow y$, then story component x corresponds to more general component y in the abstract point, so it can be generalized as far as y but no further.
- Those parts of the story that do not overlap with the abstract point at all (annotated with a ⇑) can be freely generalized or even deleted.

Encoding Non-Standard Abstractions

So far, we have presented a notation for distinguishing essential features of proverb's stories from inessential ones. By itself, however, this notation is not enough. Many proverbs have stories containing features that although inessential to the proverb's application, are nevertheless functionally significant.

The story of *make hay while the sun shines*, for example, involves features of his kind. As we have seen, the abstract point of this proverb can be paraphrased as *take advantage of opportunities when they exist*, while its story talks about hay making and sunshine. There is an important intermediate level of representation, however, which is missed if only the proverb's abstract point and story are represented. In particular, the choice of *sunshine* for the opportunity is significant. Sunshine is a resource that is only intermittently and unpredictably available, but that is essentially free when available; moreover, the availability of sunshine is not under the planner's control, so it cannot be planned for. These features of sunshine are all very relevant for planning: it is particularly important to take immediate advantage of opportunities afforded by the availability of resources that have some or all of these properties.

This description of sunshine as a resource is a generalization of BRAINSTORMER's usual representation for sunshine. Sunshine can be classified as a resource of this kind, but so can any number of other things, including rain and the generosity of kings. Moreover, these features of sunshine do not have to obtain in a new situation for the proverb to apply, although their presence constitutes good evidence that the proverb's recommendation is appropriate.

Encoding this abstraction presents a potential difficulty, however. Concepts in BRAINSTORMER are arranged in an abstraction hierarchy, which supports "property inheritance" of necessary properties of concepts. Unfortunately, it impossible to use the abstraction hierarchy to represent resources as an abstraction of sunshine. Almost any entity can serve as resource in some situations, so if `resource` were to be encoded as an abstraction of `sunshine`, it would also have to be an abstraction of most other concepts in memory. In that case, every represented object would inherit various properties of resources. This would be wasteful, and in many instances, incorrect: an object can be a resource for some purposes and in some situations, but not in others. It follows that `resource` cannot be a generalization of `sunshine` in the abstraction hierarchy.

While it is not *always* useful or correct to think of sunshine as a resource, it is certainly desirable in *some* circumstances. For example, in representing *make hay while the sunshine*, we want to encode that sunshine as an unpredictably available resource whose availability is beyond the planner's control.

Fortunately, there is a way out of this dilemma. BRAINSTORMER is able to reason with multiple descriptions of given objects, using a process called *redescription inference* [Jones, 1991a; Jones, 1991b]. Redescription inference is used to transform instances of one concept into co-referential instances of another. Redescription inference leaves traces in the form of *views* or co-reference links with attached justifications. We use views to explicitly encode important but non-standard abstractions of components of stories that are not represented in the abstraction hierarchy. Thus, we represent the proverb as it would look had `sunshine` been *redescribed* as the appropriate kind of resource. Figure 4 illustrates the resulting representation of the proverb (omitting details of the internal structure of views). Sunshine is described in terms of both the concepts `sunshine` and `resource`. `Resource` frames index information

```
positive-evaluation
 object scheduling-decision
          object hay-making-plan↑plan =p
          time   time-interval =t
 context (precondition-opportunity↑opportunity
           plan =p
           object sunshine↑entity =s
                  time-interval =t)

VIEWS:
 (=s 1. sunshine↑entity
     2. resource↑↑
          availability (intermittent↑↑
                        unpredictable↑↑
                        not-plan-for↑↑)
          unit-cost zero↑↑)
```

Figure 4: Full representation of *make hay* ...

relevant to planning regarding the availability and cost of objects.

In summary, we have described two notations that together allow us to explicitly represent functionally significant features of proverbs' stories. In the next section, we describe how these representations assist BRAINSTORMER's advice taker.

The Advice-Taking Process

BRAINSTORMER consists of a planner with an associated advice taker. The planner is handed goals having to do with countering terrorism and attempts to come up with plans of action. If the planner gets into trouble, it issues a query for information sufficient to resolve its difficulty. A user then presents advice in the form of a representation of proverb, which the advice taker attempts to transform into an answer to the query. Answers must take the form of operational planner data structures that match (abductively unify) with the query [Charniak, 1988].

Suppose, for example, that the planner is currently considering whether or not to carry out a preemptive raid against a terrorist organization. It therefore issues a query for information that would allow it to make a principled decision. Let us additionally suppose that the planner knows that public opinion is currently running high against this terrorist organization; the planner, however, has not considered the implications of this fact for its decision.

In that case, the proverb *make hay while the sun shines* could be aptly used with the intent of recommending *carry out the raid now, in the window of opportunity afforded by temporarily favorable public opinion.* The task of advice taking is to generate this specific recommendation from the initial, generic representation of the proverb.

The advice-taking process has three phases. The first involves inferring an operational planner data structure from the proverb that can match the planner's query. As we explain in [Jones, 1992], this transformation requires reasoning with an explicit model of the planning process. In this paper, however, we focus our attention on the latter phases of advice taking, because this is where the impact of highlighting functionally significant features of proverbs' stories is most acutely felt.

In the second phase of advice taking, the advice taker generates a common abstraction of the proverb's story and the planner's current problem that matches the planner's query. Here the common abstraction recommends running a plan on the basis of an unspecified opportunity afforded by an unpredictably available resource not under the planner's control. This common abstraction is then matched to the query, yielding a hypothetical answer to the query: *carry out the raid now, because there currently exists an opportunity afforded by a resource like sunshine that is unpredictably available and not under the planner's control.*

The system has now been provided with a recommendation, but thus far it has no reason to believe it particularly plausible, except that it originated with a user who presumably intends to be helpful. In the third and final phase of advice taking,[2] the system attempts to flesh out this provisional answer into a complete and well-justified recommendation. In the current example, this involves positing that the object of the opportunity is in fact the favorable state of public opinion, and then attempting to verify that public opinion can indeed be described as an unpredictably available resource not under the planner's control.

The central task of the two later phases of advice-taking is computing a common abstraction of the proverb's story and the planner's situation that can be fleshed out into a well-justified answer to the planner's query. We want this abstraction to be as useful to the planner as possible, so in fact we desire a *maximally specific* abstraction of the story in terms of features that are *functionally significant* for planning. To reiterate, a feature is functionally significant if it is part of a proverb's recommendation or if it helps specify an appropriateness condition for that recommendation.

BRAINSTORMER computes this abstraction using a knowledge-intensive matching process that compares the proverb's story with the planner's query. Abstractions are computed by incrementally generalizing components of the proverb's story in response to local inconsistencies that the matcher detects when attempting to fit these components to the planner's query. Hay making, for example, cannot be consistently described as a preemptive attack on terrorists, so this aspect of the proverb's story must be generalized. Generalization is only attempted when an inconsistency is detected; it follows that the resulting abstraction will be maximally specific.

[2] In fact, the second and third phases of advice taking are to some extent interleaved.

```
(QUERY-MATCH story query):
 IF the type labels of story and query match
 THEN For each slot of query,
        Recursively QUERY-MATCH corresponding slot
          fillers of story and query
 ELSE IF (REDESCRIBE story and query)
      THEN return success
 ELSE IF the type label of story can be generalized
      THEN LET gen be the generalization of story
                obtained by replacing its type label
                with the next more general type in
                the abstraction hierarchy.
           Generalize or delete subparts of story
           justified only by property inheritance
           from the old type label;
           (QUERY-MATCH gen query)
 ELSE IF the type label of story is entity and
         story can be deleted
      THEN delete it and return success
```

Figure 5: Finding maximally-specific abstractions

Two Problems

The matcher faces two problems in generating this abstraction, both of which are substantially mitigated by having precomputed and cached all of the functionally significant features of the proverb's story. The first problem is to ensure that the common abstractions constructed by the matcher are useful to the planner. There are a great many possible generalizations of the proverb's story, most of which do not focus on functionally significant commonalities. When the matcher faces a local inconsistency, which generalization should it prefer?

Our representation of proverbs' stories is very helpful in this regard. All of the features of the story that are conceivably useful to the planner are either implicit in the abstraction hierarchy, or are precomputed and cached with the proverb. Moreover, those features that can be generalized or deleted without invalidating the proverb's recommendation have been explicitly tagged as such. It follows that generalizing a component of a proverb's story can be accomplished by simply replacing it with an instance of a more general concept in the abstraction hierarchy (or perhaps deleting it altogether); see figure 5. BRAINSTORMER's abstraction hierarchy admits no upward branching, so this process is quite efficient.

A second problem the matcher faces is determining which components of the planner's problematic situation should be matched against candidate answers to a query. Some of these components are mentioned in the query, in which case a correspondence is established automatically by the query-matching process. Further components, however, must be determined during the third phase of advice taking, to justify this candidate answer.

For example, we have seen that a candidate answer to a query can be constructed from *make hay while the sun shines* that mentions sunshine. To justify this answer, the system has to find something in the planner's problematic situation that has something in common with sunshine. If the proverb did not encode functionally significant abstractions of sunshine, it would be necessary to search for an instance of an arbitrary abstraction of sunshine. However, sunshine is tagged as an unpredictably available resource not under the planner's control, so the advice taker can instead engage in a more focused search for a hitherto unnoticed *resource* of the appropriate kind.

Of course, this search may not always succeed in identifying a suitable resource. If the search succeeds, however, it will succeed more quickly than a search for an instance of an arbitrary generalization of sunshine. Moreover, if a resource is found that is in fact unpredictably available or not under the planner's control, then the system can have greater confidence that the proverb's recommendation actually applies.

Discussion and Related Work

Taking advice in the form of a story can be viewed as a process of analogical reasoning, or more specifically, exploiting an "analogical hint" [Greiner, 1988]. Proverbs' stories help both to construct recommendations and to determine aspects of the planner's problematic situation that justify these recommendations. We have seen that both can be accomplished by computing a common abstraction of the story and the current problem in terms of functionally significant features of the proverb's story. In our chosen representation, all such features are cached in advance with the story. These cached features guide generalization towards a suitable abstraction, and simultaneously constrain search for aspects of the planner's problem that justify the proverb's recommendation.

A knowledge-intensive approach to analogical inference is central to BRAINSTORMER's success. The system infers commonalities between a proverb's story and a planning problem on the basis of explicit representations of the story's functionally significant features. Stuart Russell likewise advocates a knowledge-intensive approach [Russell, 1989]. His DBAR system relies on knowledge about functional significance encoded as explicit *determinations*. A determination $P \succ Q$ (pronounced "P determines Q") licenses analogical inference that Q holds for a target, provided that Q also holds for a source that shares properties P with the target. For example, logos on running shoes determine their manufacturer.

Russell notes that determinations guide inference of commonalities between a source and a target. $P \succ Q$ explicitly encodes that P are the features that source and target must share to license analogical inference of Q. Trying to demonstrate that P holds of a source and target is a considerably less open-ended task than an arbitrary search for common features of a source and target. Similarly, BRAINSTORMER ex-

plicitly encodes the features of proverbs' stories that if true of a target support the proverb's recommendation, and uses these features to guide inference.

BRAINSTORMER's advice taker relates proverbs to queries using a knowledge-intensive matching process. Knowledge-intensive matching is motivated by the following realization about the nature of analogy: once it is accepted that objects, properties, and relations can be described in different ways using different vocabularies, no purely syntactic criterion for assessing similarity of a source and target could conceivably be adequate. This is not to say, of course, that no syntactic criterion intervenes at *any* stage of analogical reasoning. Rather, it is to emphasize that the major part of a theory of analogical reasoning must consist in specifying (1) which features of the source and target can be meaningfully compared, and (2) the knowledge needed to compute these features efficiently if they are not already explicit. Once this is done, *identity* of functionally relevant features may suffice as a criterion for similarity.

BRAINSTORMER accomplishes (1) by explicit encodings of functionally significant features, and (2) in terms of viewing schemas for redescription inference. As explained in [Jones, 1991a], redescription inference derives ultimately from MERLIN [Moore and Newell, 1973], and is related to the idea of "views" in Jacob's ACE system [Jacobs, 1987].

In contrast, most existing work in analogy ignores one or both of these aspects of analogical reasoning. (Russell's DBAR is a notable exception.) Greiner sidesteps (2) altogether [Greiner, 1988]. Gentner's theory of structure mapping [Falkenhainer *et al.*, 1986] advocates a purely syntactic approach to assessing similarity of source and target, and thus does not address either (1) or (2). However, Falkenhainer's PHINEAS system [Falkenhainer, 1989], while ostensibly an implementation of structure mapping, in fact adopts a more knowledge-intensive approach.

What are the limitations of BRAINSTORMER's approach to advice taking? As currently implemented, there is no way to specify that a feature of a proverb's story is functionally *insignificant*, so currently the system attempts to transfer *all* features of proverbs' stories to the planning problem at hand, even seemingly irrelevant features such as *hay*. While this slows down the inference process a little, it has not proven problematic in practice. (Tagging features of stories as functionally insignificant of course immediately raises the question of why they should be kept around at all. One reason is to allow for the possibility of future learning. A feature of a proverb that currently appears to be irrelevant to its abstract point may later be discovered to be important.)

A more fundamental limitation of the approach is the assumption that all functionally significant features of stories are computed in advance. While this assumption is reasonable for many proverbs, it is less reasonable for arbitrary stories. A story about a given episode can be used to make a variety of very different points on different occasions [Schank, 1991]. It is possible, however, that the following extension of BRAINSTORMER's approach might be adequate. Stories are usually laden with linguistic cues designed to highlight the aspects of the story that the speaker intends the hearer to focus on. Perhaps these cues can be used by an advice taker to help infer candidate abstractions of the story ready for matching to a problem. This is a direction for future research.

References

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

[Falkenhainer *et al.*, 1986] Falkenhainer, B., Forbus, D.K., and Gentner, D. 1986. The structure-mapping engine. In *Proceedings AAAI-86*, Philadelphia, PA. AAAI. 272–277.

[Falkenhainer, 1989] Falkenhainer, B. 1989. *Learning from Physical Analogies: A Study of Analogy and the Explanation Process.* Ph.D. Dissertation, University of Illinois at Urbana-Champaign.

[Greiner, 1988] Greiner, R.D. 1988. Leaning by understanding analogies. *Artificial Intelligence* 35:81–126.

[Jacobs, 1987] Jacobs, P.S. 1987. Knowledge-intensive natural language generation. *Artificial Intelligence* 33:325–378.

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

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

[Jones, 1992] Jones, E.K. 1992. Model-based case adaptation. In *Proceedings AAAI-92 Tenth National Conference on Artificial Intelligence*, San Jose, CA. Morgan Kaufmann.

[Moore and Newell, 1973] Moore, J. and Newell, A. 1973. How can MERLIN understand? In Gregg, L. W., editor 1973, *Knowledge and Cognition*. Lawrence Erlbaum Associates, New Jersey.

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

[Russell, 1989] Russell, S.J. 1989. *The Use of Knowledge in Analogy and Induction.* Morgan Kaufmann, San Mateo, CA.

[Schank, 1991] Schank, R.C. 1991. *Tell Me a Story: A New Look at Real and Artificial Intelligence.* Simon and Schuster, New York.

[Sowa, 1984] Sowa, J.F. 1984. *Conceptual Structures: Information Processing in Mind and Machines.* Addison-Wesley, Reading, MA.