LetterGen: Writing from Examples[1]

David Pautler
The Institute for the Learning Sciences
Northwestern University
Evanston, IL 60201
pautler@ils.nwu.edu

Abstract

How do people write letters? Examine the contents of any letter-writing handbook. People gain proficiency in this form of discourse through adaptation of examples (or at least there is a wide consensus that examples are an excellent way to teach good writing skills). LetterGen constructs letters in a similar manner: The programmer initially separates example letters into snippets, and provides a plan derivation for each snippet. During an interview with the user, LetterGen infers which snippets are relevant to the user's stated goals and beliefs by instantiating and adapting the stored derivations. Snippets are then ordered into a new, complete letter. Additionally, representing letters as a set of plan derivations has the consequence that translated versions of a letter do not require special treatment; the target language is treated as just one of many goals.

Introduction

LetterGen is a case-based letter-writer which adapts old letters for new uses by modifying their underlying plans. The immediate purpose of the program is to reduce the surprising amount of work usually needed to produce even the commonest, most stereotypical forms of business correspondence. For example, the letters The Institute for the Learning Sciences (ILS) sends to potential industry partners (one of LetterGen's current domains) are all pretty much the same, but finding an appropriate example letter and adapting it can take more secretarial time than it appears it should. Translating the letter for correspondence with overseas contacts can take a secretary much longer. Yet, adapting a prepared translation of the example letter should be comparably quick if the content and plan of the translation are known. Thus, LetterGen attempts to represent the plan behind a letter in a largely language-independent manner so that the content-planning of the original letter and its translations can be shared.

The choice to use case-based reasoning in LetterGen is motivated primarily by its resemblance to the human letter-writing protocol. That is, people appear to use case-based reasoning when they adapt example letters[2]. Decisions during the letter adaptation process appear to rely on factors ranging from the formation of intentions and beliefs to the production of utterances. LetterGen should have access to all points in the derivation of an original letter plan; consequently, the representation of the cases themselves must be highly detailed. The speech act planning work of Perrault et al. [Cohen & Perrault, 1979] [Perrault & Allen, 1980] [Perrault, 1990] provides a vocabulary rich enough to bridge this gap from belief to utterance. LetterGen's plan derivations are modelled on these speech act derivations to a large degree. In short, LetterGen is a case-based planner that relies on the derivational structure of a sequence of speech act plans to guide adaptation toward a complete letter.

The knowledge base

LetterGen's knowledge base has six primary parts:

1) A deterministic transition network (DTN) representation of questions and answers typical of an interview about the purpose of a letter to be produced.

2) "Deictic propositions", which are facts about the world that LetterGen learns through its interview with the user. For example, (ADDRESSEE HAS-CITY-STATE-ZIP "Chicago, IL 60602") is one such proposition. Deictic propositions have special status because their arguments are linked to the prepared interview items at branch points of the DTN.

[1] The Institute for the Learning Sciences was established in 1989 with the support of Andersen Consulting, part of The Arthur Andersen Worldwide Organization. The work was supported by DARPA through Office of Naval Research grant N00014-91-J-4092.

[2] This observation is based not just on the content of letter-writing handbooks, but also on an informal analysis of three months of ILS correspondence and interviews with those who wrote it.

3) World knowledge describing belief and goal relationships, ILS affairs, paragraph construction, and other topics.

4) Derivation structures which link the deictic propositions, rules, and rule instantiations used to generate letter plans into a MOP-like hierarchy[3] [Schank, 1982].

5) Text snippets (usually clauses) into which all letter examples and prepared translations have been decomposed [Redmond, 1990].

6) Procedures attached to SAY and IMMEDIATELY-BEFORE rule predicates which do the work of recalling and ordering text snippets into a complete letter.

Plans are not stored as wholes, nor are there any plan operators in LetterGen's representational scheme[4]. LetterGen's derivational structure serves primarily to link interview questions and answers to the text snippets which address the beliefs and goals indicated by these interview items. Since derivational paths can temporarily converge for different interview items and text snippets, the choice was made to represent this sharedness between derivations explicitly in LetterGen's derivational structure (hence, the similarity to MOPs). Thus, each letter in the library is actually a loose collection of more basic speech act plan derivations, which stretch from interview responses to text snippets. The importance of representing letters as whole plans lies in the task of ordering these speech acts into coherent and complete messages. LetterGen's communication theory is currently weak in this respect, and letters may have to be stored as wholes in the future.

Processing

LetterGen conducts an interview about the purpose of the letter to be produced by presenting all eligible query interface items from the DTN in a window. The response items the user chooses activate any attached deictic propositions [Figure 1]. Responses also advance the program down the DTN.

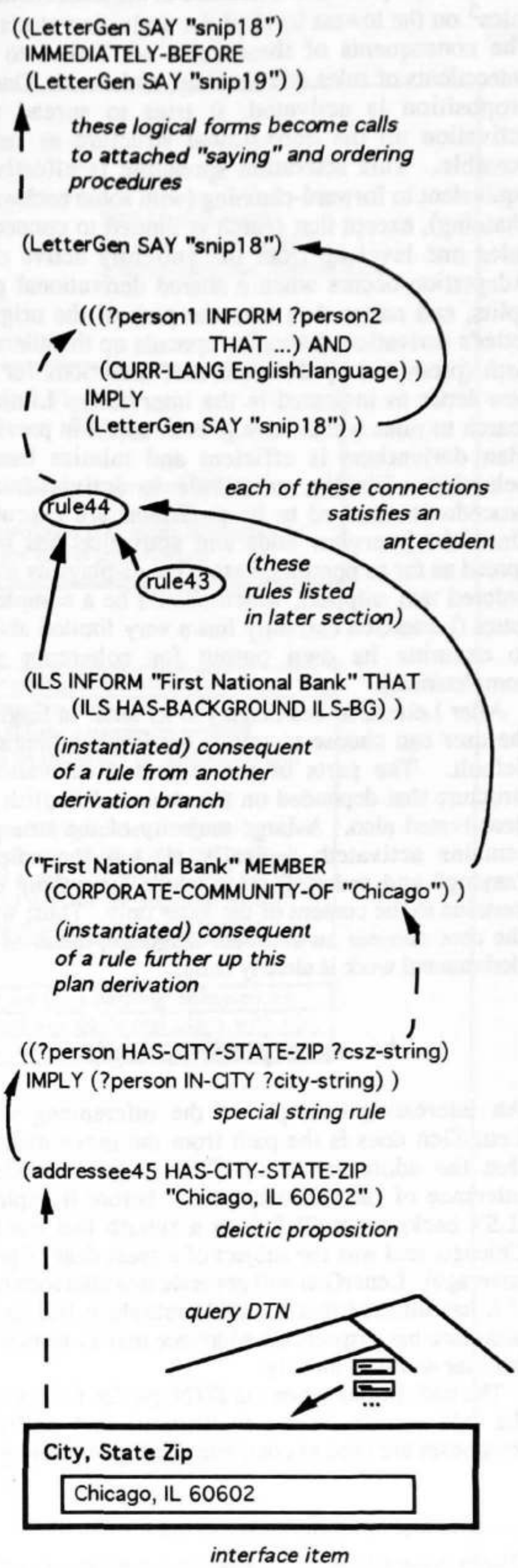

((LetterGen SAY "snip18")
IMMEDIATELY-BEFORE
(LetterGen SAY "snip19"))
these logical forms become calls to attached "saying" and ordering procedures
(LetterGen SAY "snip18")
(((?person1 INFORM ?person2
THAT ...) AND
(CURR-LANG English-language))
IMPLY
(LetterGen SAY "snip18"))
each of these connections satisfies an antecedent
rule44
rule43
(these rules listed in later section)
(ILS INFORM "First National Bank" THAT
(ILS HAS-BACKGROUND ILS-BG))
(instantiated) consequent of a rule from another derivation branch
("First National Bank" MEMBER
(CORPORATE-COMMUNITY-OF "Chicago"))
(instantiated) consequent of a rule further up this plan derivation
((?person HAS-CITY-STATE-ZIP ?csz-string)
IMPLY (?person IN-CITY ?city-string))
special string rule
(addressee45 HAS-CITY-STATE-ZIP
"Chicago, IL 60602")
deictic proposition
query DTN
City, State Zip
Chicago, IL 60602
interface item

Figure 1

[3] The links between rule consequents and antecedents resemble the links that package scenes and MOPs into larger MOPs. LetterGen also has links from rule MOPs to the instantiations of them made during processing.

[4] Perrault and Allen used operators in their STRIPS-like planner, but their operators are really just compiled derivations, which can be seen in their use of a 'body' slot to hide intermediate steps [Perrault and Allen, 1980, p. 172b]. LetterGen's derivational structure collects preconditions, intermediate steps, and effects in the same manner as operators, but the collected contents can be shared. Neither system uses delete lists or filter conditions.

Deictic propositions are linked to the antecedents of rules[5] on the lowest level of the derivation structure. The consequents of these rules are linked to the antecedents of rules one level up, and so on. Once a proposition is activated, it tries to spread that activation up the derivational structure as far as possible. This activation spreading is effectively equivalent to forward-chaining (with some backward-chaining), except that search is limited to connected rules one level up from the currently active rule. Adaptation occurs when a shared derivational path splits, and rather than take the path of the original letter's derivation, activation spreads up the alternate path (prompted by different user intentions for the new letter as indicated in the interview). Limiting search to rules which have proven useful in previous plan derivations is efficient and mimics human behavior. Finally, as a rule is activated, any procedures attached to its predicates are executed. Once the interview ends and activation has been spread as far as possible, LetterGen displays its set of ordered text snippets, which should be a completed letter (LetterGen currently has a very limited ability to examine its own output for coherence and completeness).

After LetterGen has displayed its letter in English, the user can choose to retract the English language default. The parts of the activated derivational structure that depended on the choice of English are deactivated also. A large majority of the structure remains activated: typically all but the snippet "saying" and ordering rules, since everything else pertains to the content of the letter only. Thus, when the user chooses an alternate language, much of the derivational work is already done.

An excerpted example

An interesting example of the inferencing work LetterGen does is the path from the user's mention that the addressee is in Chicago to LetterGen's utterance of "as you may know" before it explains ILS's background (ILS is in a suburb that borders Chicago, and was the subject of a great deal of press coverage). LetterGen will generate this utterance only if it has inferred that it should introduce ILS to the addressee but suspects the addressee may be somewhat familiar with ILS already.

The path begins when the DTN queries the user for the addressee's name, institution, and address. Responses are used to complete deictic propositions:

```
1:(addressee45 HAS-INSTITUTION
            "First National Bank" )
2:(addressee45 HAS-CITY-STATE-ZIP
            "Chicago, IL 60602" )
```

which a chain of rules transform into:

```
3:("First National Bank" IN-CITY "Chicago")
```

Early in the interview, the user is asked if "First National Bank" is a potential ILS industry partner (a fact which can trigger a large number of inference paths). If the answer is yes, then propositions 3 and 4 immediately imply proposition 5:

```
4:(POTENTIAL-ILS-PARTNER
  "First National Bank")
5:("First National Bank" MEMBER
  (CORPORATE-COMMUNITY-OF "Chicago") )
```

Being a member of the corporate community of Chicago triggers rule43 which then triggers rule44 in part[6]:

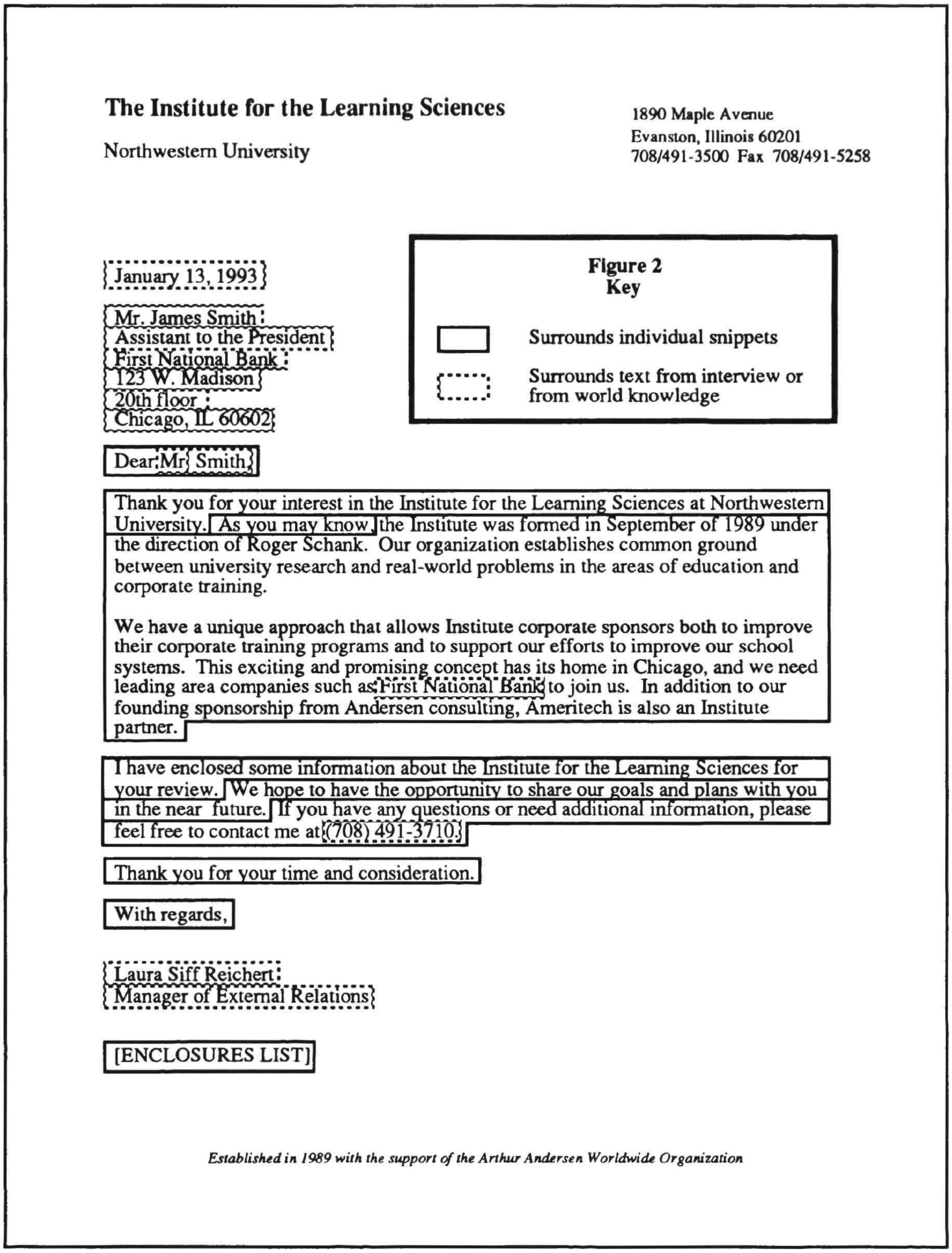
```
rule43: (ILS AWARE-OF
      ((?person MEMBER
  (CORPORATE-COMMUNITY-OF "Chicago") )
       MAKE-LIKELY
       (?person AWARE-OF
  (ILS HAS-BACKGROUND ILS-background) )))

rule44: (((?person1 INFORM ?person2
                THAT ?situation1 ) AND
       ((?person1 AWARE-OF
         (?situation2 MAKE-LIKELY
          (?person2 AWARE-OF ?situation1) ))
        AND ?situation2 ))
       IMPLY
       (?person1 DESIRE
       (?person1 INFORM ?person2  THAT
        (?person1 AWARE-OF
         (?situation2 MAKE-LIKELY
          (?person2 AWARE-OF ?situation1) )))))
```

The INFORM precondition of rule44 is satisfied by LetterGen's inference elsewhere that it should explain ILS's background (which First National Bank may already be aware of). The consequent of rule44 leads immediately to an INFORM act, since LetterGen currently has no other preconditions on INFORM acts

[5] For brevity, I have used "rule" in this paragraph where "uniquified version of a rule" would be more accurate. Uniquified versions of a rule may appear in several separate locations of a single derivation.

[6] AWARE-OF is a renaming of the more commonly used KNOW predicate, but avoids the connotation of familiarity, e.g. "John knows Bill."



The Institute for the Learning Sciences

Northwestern University

1890 Maple Avenue
Evanston, Illinois 60201
708/491-3500 Fax 708/491-5258

Figure 2
Key

□	Surrounds individual snippets
⬚	Surrounds text from interview or from world knowledge

January 13, 1993

Mr. James Smith
Assistant to the President
First National Bank
123 W. Madison
20th floor
Chicago, IL 60602

Dear Mr. Smith,

Thank you for your interest in the Institute for the Learning Sciences at Northwestern University. As you may know, the Institute was formed in September of 1989 under the direction of Roger Schank. Our organization establishes common ground between university research and real-world problems in the areas of education and corporate training.

We have a unique approach that allows Institute corporate sponsors both to improve their corporate training programs and to support our efforts to improve our school systems. This exciting and promising concept has its home in Chicago, and we need leading area companies such as First National Bank to join us. In addition to our founding sponsorship from Andersen consulting, Ameritech is also an Institute partner.

I have enclosed some information about the Institute for the Learning Sciences for your review. We hope to have the opportunity to share our goals and plans with you in the near future. If you have any questions or need additional information, please feel free to contact me at (708) 491-3710.

Thank you for your time and consideration.

With regards,

Laura Siff Reichert
Manager of External Relations

[ENCLOSURES LIST]

Established in 1989 with the support of the Arthur Andersen Worldwide Organization

Figure 2

than the goal to perform one[7]. This act and the English language default trigger proposition 6:

6:(LetterGen SAY "snippet18")

When this proposition is activated, the two arguments of SAY are passed to its attached procedure. Since the agent of the saying is LetterGen, the procedure recalls the text of snippet18, "as you may know", and places it on a queue with other snippets. Proposition 6 and other rule consequents trigger snippet ordering rules, one of which positions snippet18 before the snippet that explains ILS's background: "As you may know, the Institute was formed in September of 1989 under the direction of Roger Schank..." [Figure 2].

Once an English version of the letter has been generated, the user can press a button that causes the DTN to offer several language options. Choosing any of them deactivates the English default, which then deactivates proposition 6 and all the ordering rules which depend on it, and withdraws snippet18 from the queue. Opting for a French version triggers a new SAY proposition for the French equivalent of "as you may know", since rule44, representing the content of snippet18, is still active. A complete French translation is built on top of the still-active content rules in this way.

The English letter that this inference path helps to generate involves 12 interview questions, 180 inference steps, and 19 text snippets. The interview takes less than two minutes to complete for a first-time user. The example letters LetterGen adapts are letters that ILS has actually sent to prospective industry partners.

Ten of the twelve questions were devoted to determining the user's name, the addressee's address, and any enclosures. The majority of the letter was reconstructed from the answers to just two questions: "Who is the addressee?" (industry, academic colleague, member of the press) and "How familiar are they with ILS?" (this letter is a "cold call", they attended an Open House, they wrote to us, they're a longtime associate). Limiting queries to the user in this way serves both to simplify the user's task and to force the programmer to develop a richer motivation analysis for each example letter. Additionally, it is hoped that examining these motivations as an entire chain of rule applications (a "case"), rather than individual rules, will guide the knowledge representation task toward better statements of these rules (cf. Moore's 'Bomb in the Toilet' problem [Moore, 1980]).

Planned extensions

The choice to use a DTN to guide interviewing was meant to simplify the initial task of designing LetterGen. The program should plan queries to the user by the same means used to generate text for the addressee. A number of thorny representational issues arose during a recent implementation attempt, however[8]. One problem is representing quantified variables in nested propositions, especially those involving BELIEVE predicates. An enduring problem is the inference of mutual belief. Because a completely sound inferring of mutual belief requires an infinite regress of belief assumptions [Clark & Marshall, 1981], heuristics must be used. Perrault advocates a default logic to solve this problem [Perrault, 1990], but a weighted abduction scheme [Charniak, 1988] [Kass, 1989] [Hobbs et al., 1990] appears more suited to an inference process guided by previous derivations.

The premise underlying a marriage between weighted abduction and stored derivational structure is that human beings judge likelihood largely on the basis of previous experience. Often, the conditions we know to be good or bad evidence for a hypothesis are culled from known interactions between those conditions and the hypothesis in previously executed plans. These interactions are represented in the derivational structure of the plan. Thus, derivational structure can be used to determine the efficacy of plan goals when only some of the deictic propositions of the plan are known to be true. Typically, this derivation is constructed on the fly, but the influence of the availability heuristic on likelihood judgements, as shown by Kahneman et al. [Kahneman, Slovic, & Tversky, 1982], suggests that stored derivations would exert more influence than newly generated ones. Naturally, an improved system could refer to either source. In the case of inferring mutual belief, an accumulation of evidence (e.g., shared experience and previously inferred mutual belief) could cause the sum of rule weights to pass a persuasion or dissuasion threshold, or a preset deadline could end the search with no conclusion or only the weak confidence gained by that point. But the role of mutual belief in previous utterance plans would be the guiding force behind decisions of what evidence to look for initially.

For greater flexibility, LetterGen has to be able to look at rules and derivation sequences beyond the immediate surroundings of the current activation point. Adaptation of this kind would resemble what Carbonell has called Derivational Analogy

[7] A version in progress also requires that a "communication channel" (e.g., a letter, conversation, or phone connection) of a familiar language be "open".

[8] This isn't so much a problem with replacing the DTN as with the current weaknesses of LetterGen's communication theory. Until the DTN was to be replaced, the weaknesses could be ignored.

[Carbonell, 1986]. When writing business correspondence from example letters, people appear to perform this more expensive method of planning only after deciding that some part of the example letter cannot be adapted by simpler means. That is the current plan for LetterGen: The program will widen its search only after options immediately available in the structure prove inadequate. But deciding which options are inadequate will have to rely on a more robust theory of beliefs, goals, and their relation to utterances.

Conclusion

As a case-based system, LetterGen mimics the human letter-writing protocol. This fidelity requires that letter cases be richly represented - covering the entire span of planning from intention to utterance. To allow efficient generation of versions in other languages, the representation must also be largely language-independent. LetterGen demonstrates the viability of case-based generation in meeting these constraints.

References

Carbonell, Jaime G. 1986. Derivational analogy: A Theory of Reconstructive Problem Solving and Expertise Acquisition. In R. Michalski, J. Carbonell, and T. Mitchell (eds.), *Machine Learning: Volume II*. Los Altos, CA: Morgan Kaufmann Publishers.

Charniak, Eugene. 1988. Motivation Analysis, Abductive Unification, and Nonmonotonic Equality. *Artificial Intelligence* 34:275-295.

Clark, H. H. and Marshall, C. R. 1981. Definite reference and mutual knowledge. In A. K. Joshi, B. Webber, and I. A. Sag (eds.) *Elements of discourse understanding*. Cambridge, UK: Cambridge University Press.

Cohen, P. R. and Perrault, C. R. 1979. Elements of a Plan-based Theory of Speech Acts. *Cognitive Science* 3:177-212.

Hobbs, J., Stickel, M., Appelt, D., and Martin, P. 1990. *Interpretation as Abduction*, AI Center Technical Note 499. Menlo Park, CA: SRI International.

Kahneman, D., Slovic, P., and Tversky, A. 1982. *Judgement under uncertainty: Heuristics and biases*. Cambridge, UK: Cambridge University Press.

Kass, Alex. 1989. Adaptation-Based Explanation: Extending Script/Frame Theory To Handle Novel Input. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, 141-147. Menlo Park, CA: International Joint Conferences on Artificial Intelligence, Inc.

Moore, R. C. 1980. *Reasoning about Knowledge and Action*, AI Center Technical Note 191. Menlo Park, CA: SRI International.

Perrault, C. R. and Allen, J. F. 1980. A Plan-based Analysis of Indirect Speech Acts. *American Journal of Computational Linguistics*, 6(3-4):167-182.

Perrault, C. R. 1990. An Application of Default Logic to Speech Act Theory. In P. R. Cohen, J. Morgan, and M. E. Pollack (eds.), *Intentions in Communication*. Cambridge, MA: MIT Press.

Redmond, Michael. 1990. Distributed Cases for Case-Based Reasoning; Facilitating Use of Multiple Cases. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, 304-309. Menlo Park, CA: American Association for Artificial Intelligence.

Schank, Roger C. 1982. *Dynamic Memory*. Cambridge, UK: Cambridge University Press.