A Model-Based Approach for Supporting Dialogue Inferencing in a Conversational Case-Based Reasoner

David W. Aha and Tucker Maney
Navy Center for Applied Research in Artificial Intelligence
Naval Research Laboratory
Washington, DC 20375
lastname@aic.nrl.navy.mil

Abstract

Conversational case-based reasoning (CCBR) is a form of interactive case-based reasoning where users input a partial problem description (in text). The CCBR system responds with a ranked *solution* display, which lists the solutions of stored cases whose problem descriptions best match the user's, and a ranked *question* display, which lists the unanswered questions in these cases. Users interact with these displays, either refining their problem description by answering selected questions, or selecting a solution to apply. CCBR systems should support *dialogue inferencing*; they should infer answers to questions that are implied by the problem description. Otherwise, questions will be listed that the user believes they have already answered. The standard approach to dialogue inferencing allows case library designers to insert rules that define implications between the problem description and unanswered questions. However, this approach imposes substantial knowledge engineering requirements. We introduce an alternative approach whereby an intelligent assistant guides the designer in defining a model of their case library, from which implication rules are derived. We detail this approach, its benefits, and explain how it can be supported through an integration with Parka-DB, a fast relational database system. We will evaluate our approach in the context of our CCBR system, named *NaCoDAE*.

Conversational Case-Based Reasoning

We introduce an integrated reasoning approach in which a model-based reasoning component performs an important inferencing role in a *conversational case-based reasoning* (CCBR) system named NaCoDAE (Breslow & Aha, 1997) (Figure 1). CCBR is a form of case-based reasoning where users enter text queries describing a problem and the system assists in eliciting refinements of it (Aha & Breslow, 1997). Cases have three components:

1. *Summary*: A brief, partial textual description of the case.
2. *State*: A set of ⟨Question,Answer⟩ pairs.
3. *Solution*: A sequence of actions for responding to this state.

Given a user's problem description, NaCoDAE computes its similarity with case summaries in the library and displays the solutions of a pre-determined number of the most similar cases. It also displays a predetermined number of the unanswered questions from those cases. Both displays are ranked by estimated quality. In response, the user can select and answer a question from the question display, thereby refining their problem description. This allows the similarity function, which consults the stored cases' summaries *and* states, to generate a more accurate re-ranking for the two displays. Alternatively, the user can select and apply a displayed solution.

CCBR has achieved tremendous success, especially for solving interactive diagnosis tasks, in commercial, industrial, and government applications (Watson, 1997). Five companies market CCBR shells; the market leader is Inference Corporation.

This paper focuses on a specific concern with the CCBR approach: supporting inferencing between user-entered problem descriptions and the case library to ensure that the system's conversations appear to be intelligent. In the following sections, we detail this *dialogue inferencing problem*, explain why the standard approach to solving it is problematic, and introduce our alternative solution. We finish with a description of our research agenda for evaluating our approach.

The Dialogue Inferencing Problem

CCBR systems should not display questions that the user considers to have already (implicitly) answered in their initial problem description and/or previously answered questions. Figure 2 demonstrates an example of this dialogue inferencing problem for a library whose

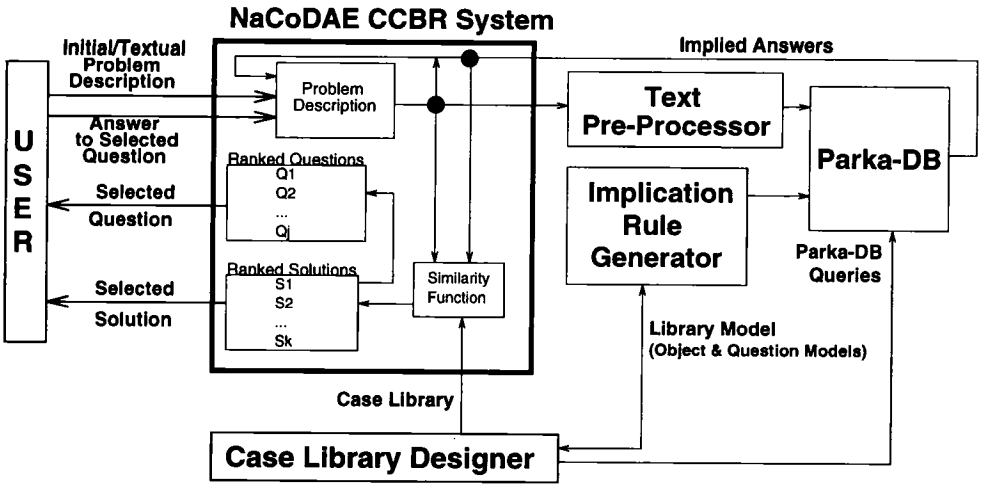

Figure 1: Model-Based Support for Dialogue Inferencing in CCBR Conversations

User's Initial Problem Description: I'm getting black streaks on my paper
Display of Top-Ranked Questions in Response to this Description:

1. Q21: What does the print quality look like?
2. Q24: Are you having print quality problems?
3. Q18: Are you printing on the correct side of the paper?
4. Q25: What is the display message?

Figure 2: Example of the Dialogue Inferencing Problem During a CCBR Conversation

cases define common troubleshooting scenarios for a computer printer.

In this conversation, two of the displayed questions were implicitly answered in the user's text. If these questions are not understood to have been answered, then this negatively impacts case retrieval performance by reducing *precision* and *efficiency*. That is, if the system could infer the implied answers, then it could define the problem's state more precisely for similarity computations and, thus, reduce the number of questions the user must answer during conversations before retrieving a satisfactory solution.

Two types of implication rules exist: *text* rules, which relate user-entered text in a problem description to ⟨question,answer⟩ pairs in the case library, and *chaining* rules, which relate these pairs to each other. For example, a text rule would be responsible for answering question Q21 with "Black Streaks" in Figure 2. However, a chaining rule would be used to infer an answer of "Yes" to question Q24, given that question Q21 is "Black Streaks," which is indicative of a print quality problem. In our approach, the implication rule generator (Figure 1) is responsible for deriving both types of rules.

Standard Approach for Supporting Dialogue Inferencing

Some commercial CCBR systems attempt to solve this problem by allowing library designers to manually enter implication rules. For example, text implication rules are consulted whenever the user modifies their text description. If "black streaks" and "paper" are in the user's description and the user has entered the following rule:

> IF text includes "black streaks" and "paper"
> THEN assign "Yes" as the answer to Q24,

then the second question in Figure 2 will automatically be answered "Yes," and the user will have the opportunity to verify this conclusion.

Although this approach can solve the dialogue inferencing problem, it presents significant knowledge engineering challenges:

1. *Input Size*: Rule sets are often large (e.g., the well-known printer troubleshooting library, which contains 25 cases and 28 questions, easily yields over 100 rules). Inserting them all is tedious, and can be prone to errors.
2. *Comprehensibility*: Large sets of unrelated rules can be difficult to examine.

3. *Maintenance*: Maintaining large rule sets can be difficult, if not impossible, for case libraries that require updating.

Some CCBR case library designers avoid using rules, either because an incomplete set of rules will decrease case retrieval performance for their applications, or because maintenance issues complicate the rule updating process.

Model-Based Dialogue Inferencing

Instead of inserting implication rules, we propose that the library designer interactively enter a *library model*, composed of an *object* model and a *question* model. A rule-extraction routine can then dynamically extract rules that are complete with respect to this model. For many tasks, this model will be more compact than its corresponding rule set, and thus be more comprehensible and easier to maintain.

Figure 1 summarizes our approach, which integrates NaCoDAE with Parka-DB, a high-performance knowledge representation system for processing relational queries (Hendler et al., 1996). The object model relates the objects in the domain of the case library. A question model relates the library's questions to the object model. The implication rule generator inputs these models, along with the current problem description, and dynamically creates a knowledge base (KB). Given this KB and appropriate relational queries, Parka-DB can then derive answers implied by the current conversation, which are added to the user's problem description.

Interactive Model Creation

Library models are created by the user when using NaCoDAE's editors to create cases, questions, and actions. Parsing techniques will be used to assist in identifying potential objects and their relationships. For example, when the user enters the question *Can your printer print a self test?*, "printer" and "self test" will be identified as (possibly previously identified) objects connected by the relationship "print." Users will be queried to confirm all tentative identifications. As explained below, a semantic network will be interactively created to denote objects, questions, and their relations. User-supplied text at the start of conversations will be pre-processed to automatically identify known synonymous phrases.

Representing Library Models

A small part of the object model for the printer troubleshooting case library is shown in Figure 3. This model relates a printer with its printout and possible print qualities, where ellipses denote individuals, rounded boxes denote categories, and variable names begin with a question mark (e.g., "?PQ" denotes a variable for print quality).

Figure 4 displays a partial question model that relates two questions to this object model. Each question is associated with an *interpretation* that determines how its answer can be derived. Interpretations for boolean questions (e.g., Q24) test for subgraph existence, perhaps involving existentially quantified variables. Interpretations for list questions (e.q., Q21) instead focus on locating bindings during subgraph matching.

Deriving Implication Rules

Implication rules, if satisfied, insert answers to previously unanswered questions. The rule generator will derive both text and chaining rules.

Deriving text rules requires recognizing existing objects and relations in the object model, but this time from user text rather than from the case library designer. For example, in response the user's input in Figure 2, we want the system to answer question Q21 with "black streaks." This requires binding variable "?PQ" to "black streaks" in the object model and then locating questions (e.g., Q21) whose answers depend on having a value for this variable.

Deriving chaining rules instead requires relating the interpretations of two questions in the question model. For example, having an answer to Q21 ("What does the print quality look like?") allows us to derive an answer for Q24 ("Is there a print quality problem?"). Relating these interpretations requires matching subgraphs, and recognizing that the unknown variables of the target subgraph are bound in the source graph. The two chaining implication rules that relate the interpretations shown in Figure 4 are shown in Figure 6. The first of these two rules states that if Question 21 is answered *Black Streaks*, then Question 24 should be answered *Yes*. The second chaining rule is similar, but for *Faded* print quality.

The rule generator will derive all of the implication rules off line (i.e., prior to user conversations). However, because an enormous number of text rules could be written, the rule derivation process must be controlled so as to yield text rules whose conditions are abstract. Towards this goal, our approach stores triggering phrases with each ⟨question,answer⟩ pair, uses a thesaurus capability (i.e., WordNet (Miller, 1995)) and user interaction to identify synonymous phrases, and thereby incorporates text rules into a single "global" rule. Likewise, interactive assistance will is also involved in identifying and verifying chaining rules derived from the library model.

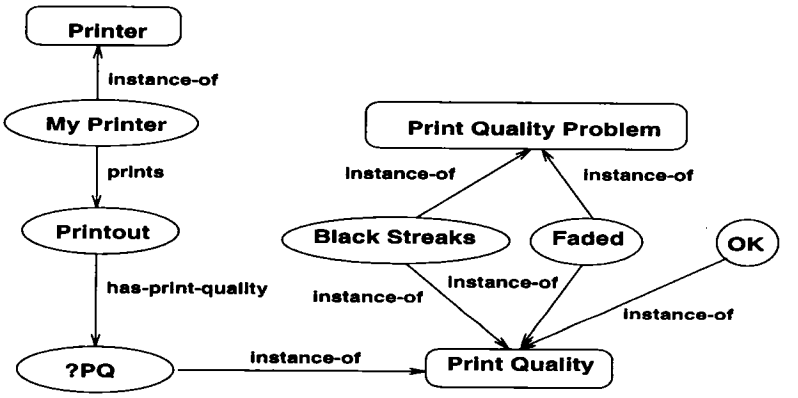

Figure 3: Partial Object Model for the Printer Troubleshooting Library

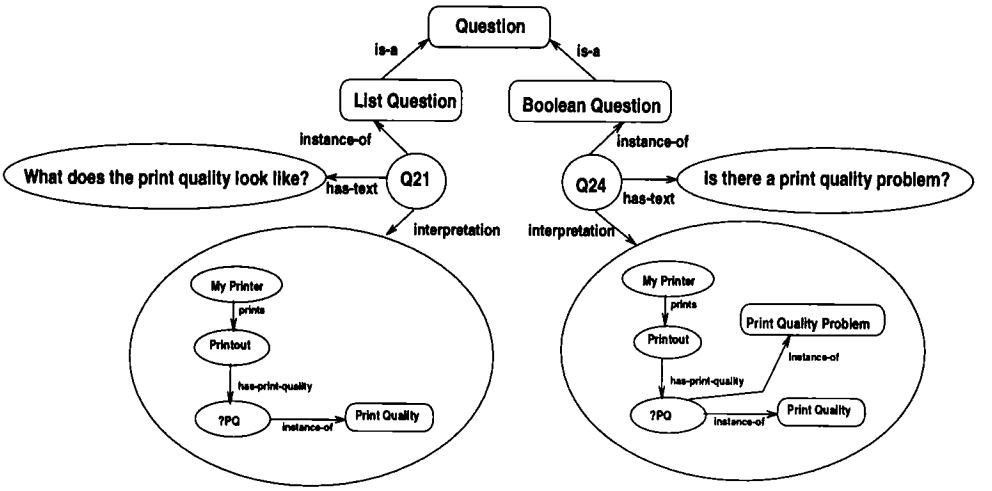

Figure 4: Partial Question Model for the Printer Troubleshooting Library

Querying Parka

Evaluating Parka-DB queries requires a relational knowledge base. In this context, a Parka-DB KB, which consists of binary assertions, consists of the implication rules derivable from the library model and the user's processed problem description. For example, assertions corresponding to text implication rules can be easily generated. Suppose that a naive text rule existed, stating that if the text description contained the phrase "black streaks," then Q21 should be answered with "Black Streaks." The corresponding assertions are:

(has_text Q21_Black_Streaks 'black streaks')
(QA_question Q21_Black_Streaks Q21)
(answer Q21 'unknown')
(QA_answer Q21_Black_Streaks 'Black Streaks')

Next, suppose that the user's text does indeed include the phrase "black streaks" or a synonymous phrase as identified by the text pre-processor (Figure 1). Finally, assume that a KB containing the text implication assertions shown above is given, along with a string matching facility. Then the query shown in Figure 5, which invokes a string matching test on the user's initial text description, can be used to derive the following bindings:

{ ?Phrase/'black streaks'
?QA/Q21_Black_Streaks,
?Text/'black streaks'
?Q/Q21,
?A/'Black Streaks'}.

Thus, this procedure can derive the answer "Black Streaks" for question Q21.

The assertions corresponding to the two chaining implication rules shown in Figure 6 are:

(chaining_implies Q21_Black_Streaks Q24_Yes)
(QA_question Q21_Black_Streaks Q21)

```
{ (text_implies ?Phrase ?QA),      ;; Locate phrase for ⟨Q,A⟩ pair
  (has_text ?QA ?Text),            ;; Find text for this ⟨Q,A⟩ pair
  (stringMatch ?Text ?Phrase)      ;; Match retrieved phrase in this text
  (QA_question ?QA ?Q),            ;; ?Q is the question to answer
  (answer ?Q 'unknown'),           ;; Check that it's not currently answered
  (QA_answer ?QA ?A)       }       ;; then retrieve implied answer.
```

Figure 5: Parka-DB Query for Retrieving Implied Answers from Text Implication Rules

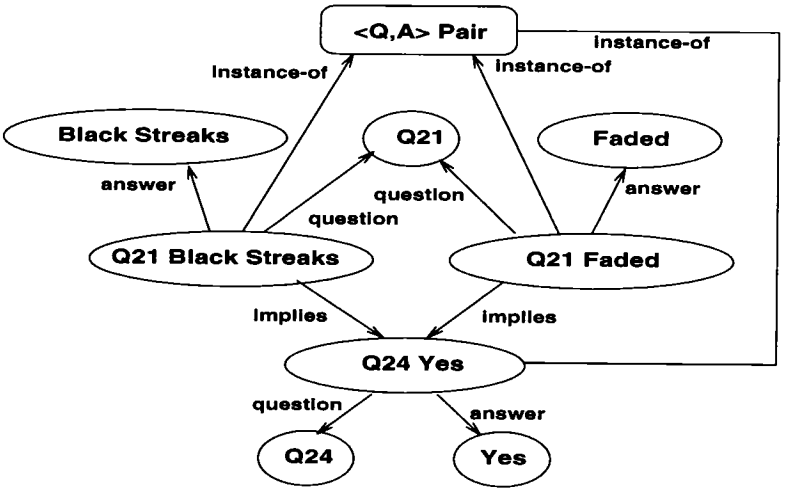

Figure 6: Two Chaining Implication Rules for the Printer Troubleshooting Library

(QA_answer Q21_Black_Streaks 'Black Streaks')
(QA_question Q24_Yes Q24)
(QA_answer Q24_Yes 'Yes')
(chaining_implies Q21_Faded Q24_Yes)
(QA_question Q21_Faded Q21)
(QA_answer Q21_Faded 'Faded')

If we assume that the problem description currently contains the single ⟨question,answer⟩ pair (i.e., *(answer Q21 'Black Streaks')*), and that all other questions have a default answer of 'unknown', then this KB can be queried for implied answers as shown in Figure 7. Given this KB and query, Parka-DB will find the following bindings:

```
{ ?QAx/Q21_Black_Streaks,
  ?Q1/Q21,
  ?A1/'Black Streaks',
  ?QAy/Q24_Yes,
  ?Q2/Q24,
  ?A2/'Yes'}.
```

Thus, this approach should find the answer *Yes* for Question 24, given only that the user included the phrase "black streaks" in their problem description. Using this approach, other implied answers can also be retrieved.

Related Research

Few publications exist on CCBR, and those that do tend not to focus on integrating model-based inferencing. For example, Aha and Breslow (1997) describe the use of a knowledge-poor learning approach to revise case libraries so as to improve their case retrieval behavior. Racine and Yang (1997) describe how to maintain conversational case bases. Trott and Leng (1997) describe how to structure the library development process using the KADS methodology.

Several researchers have integrated model-based approaches in a CBR framework. For example, Vilain et al. (1990) introduced a representation language that supports a hybrid analytical and case-based approach intended to increase learning rates and reduce the brittleness of induced generalizations for classification tasks. Navinchandra et al. (1991) describe index transformation techniques in CADET, a CBR system that solves mechanical design tasks by representing causal relations between problem variables. Hastings et al. (1995) describe CARMA, which uses a model-based

```
{ (QA_question ?QAx ?Q1),          ;; Find question of first ⟨q,a⟩ pair
  (QA_answer ?QAx ?A1),            ;; Find answer of that pair
  (answer ?Q1 ?A1),                ;; Check: This is also its current answer
  (QA_question ?QAy ?Q2),          ;; ?Q2 is the question we want to answer
  (answer ?Q2 'unknown'),          ;; Check: It's not currently answered
  (chaining_implies ?QAx ?QAy),    ;; If there is an implied answer,
  (QA_answer ?QAy ?A2)         }   ;; then retrieve it.
```

Figure 7: Query for Retrieving Implied Answers from Chaining Implication Rules

adaptation component to increase the predictive accuracy of a case-based reasoner. Several other examples exist. However, to our knowledge, no previous effort has focused on integrating model-based components to improve the performance of a CCBR approach.

Research Status and Agenda

We are presently integrating NaCoDAE (Breslow & Aha, 1997) with Parka-DB (Hendler et al., 1996). We plan to test this integration on a completed printer troubleshooting library model and on models for other case libraries. We intend to compare the case retrieval performance of our integrated system with one in which users explicitly enter implication rules.

Currently, we have generated implication rules for the printer troubleshooting task and used Parka-DB to derive implied answers from sample user problem descriptions. Our next steps include developing code to link NaCoDAE with Parka-DB, implementing a user interface for domain model capture, and expanding the current integration with WordNet (Miller, 1995). We anticipate that an initial prototype implementation for the integrated environment will be completed by the spring of 1998.

Important research issues include defining how to graphically relate question interpretations to derive chaining rules, how to process problem description text for use in triggering text implication rules, thesaurus/canonicalization concerns, and how to semi-automate the task of creating a library model from text in case descriptions, questions, answers, and actions.

Acknowledgements

Thanks to Len Breslow for his suggestions and comments on this paper, and to Merwyn Taylor, David Rager, and Jim Hendler for their assistance on obtaining and using Parka-DB. This research was supported by a grant from the Office of Naval Research and a Cooperative Research and Development Agreement (NCRADA-NRL-97-160) with Inference Corporation.

References

Aha, D. W., & Breslow, L. A. (1997). Refining conversational case libraries. *Proceedings of the Second International Conference on Case-Based Reasoning* (pp. 267–278). Providence, RI: Springer.

Breslow, L., & Aha, D. W. (1997). *NaCoDAE: Navy Conversational Decision Aids Environment* (Technical Report AIC-97-018). Washington, DC: Naval Research Laboratory, Navy Center for Applied Research in Artificial Intelligence.

Hastings, J., D., Branting, L. K., & Lockwood, J. A. (1995). Case adaptation using an incomplete causal model. *Proceedings of the First International Conference on Case-Based Reasoning* (pp. 181–192). Sesimbra, Portugal: Springer-Verlag.

Hendler, J., Stoffel, K., & Taylor, M. (1996). *Advances in high performance knowledge representation* (Technical Report CS-TR-3672). College Park, MD: University of Maryland, Department of Computer Science.

Miller G.A. (1995). WordNet: A Lexical Database for English. *Communications of the ACM, 38*(11), 39–41.

Navinchandra, D., Sycara, K., & Narasimhan, S. (1991). A transformational approach to case based synthesis. *Artificial Intelligence in Engineering Design and Manufacturing, 5.*

Racine, K., & Yang, Q. (1997). Maintaining unstructured case bases. *Proceedings of the Second International Conference on Case-Based Reasoning* (pp. 553–564). Providence, RI: Springer.

Trott, J. R., & Leng, B. (1997). An engineering approach for troubleshooting case bases. *Proceedings of the Second International Conference on Case-Based Reasoning* (pp. 178–189). Providence, RI: Springer.

Vilain, M., Koton, P., & Chase, M. P. (1990). On analytical and similarity-based classification. *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 867–874). Boston, MA: AAAI Press.

Watson, I. (1997). Applying Case-Based Reasoning: Techniques for Enterprise Systems. San Francisco: Morgan Kaufmann.