Dialogue Management for Conversational Case-Based Reasoning

Karl Branting, James Lester, and Bradford Mott

LiveWire Logic, Inc.
2700 Gateway Centre Blvd., Suite 900
Morrisville, NC 27560
{branting,lester,mott}@livewirelogic.com

Abstract. Two key objectives of conversational case-based reasoning (CCBR) systems are (1) eliciting case facts in a manner that minimizes the user's burden in terms of resources such as time, information cost, and cognitive load, and (2) integrating CBR with other problem solving modalities. This paper proposes an architecture that addresses both these goals by integrating CBR with a discourse-oriented dialogue engine. The dialogue engine determines when CBR or other problem-solving techniques are needed to achieve pending discourse goals. Conversely, the CBR component has the full resources of a dialogue engine to handle topic changes, interruptions, clarification questions by either the user or the system, and other speech acts that arise in problem-solving dialogues.

1 Introduction

Conversational case-based reasoning (CCBR) is intended to improve the interaction between users and CBR systems by eliciting case facts in a manner that minimizes the user's burden in terms of resources such as time, information cost, and cognitive load. The goal of improving the interaction between users and CBR systems gives rise to a number of distinct issues:

1. Minimizing the number or cost of questions by determining the most informative question to ask at each stage of an interaction.
2. Giving users control over the degree to which initiative is held by the system or user.
3. Enabling the system to explain both why a question was asked and how the system's answer was reached.
4. Handling interruptions (temporary topic shifts) and subgoals (providing information the user needs to answer a system question, or eliciting information needed to answer a question the user is unable to answer).
5. Permitting users to ask clarifying questions.
6. Enabling the system to ask users clarifying questions when necessary.
7. Integrating CBR with other problem-solving modalities that can answer questions that would otherwise have to be posed to the user.

Most previous work in CCBR has focused on the first of these issues, minimizing the number of questions asked by the system. Approaches to minimizing questions that have been explored include inferring answers to redundant questions [Aha et al., 1998], recognizing the earliest point in the dialogue at which no more questions are required [McSherry, 2003], and ordering questions by information gain [Doyle and Cunningham, 2000,McSherry, 2001] or similarity variance [Kohlmaier et al., 2001].

Recent research has focused on CCBR as a dialogue amenable to the standard tools of discourse analysis. For example, [Göker and Thompson, 2000] identified a set of dialogue operators applicable to CCBR, and [Bridge, 2002] modeled the interactions in CCBR through a dialogue grammar. As CBR becomes increasingly embedded in general problem-solving agent architectures, rather than in stand-alone applications, these issues will become increasingly important.

This paper presents a general architecture for CCBR called the *Discourse Goal Stack Model* (DGSM). The next section briefly summarizes the key issues in dialogue management. Section 3 outlines the DGSM architecture. Section 4 describes an implementation of this architecture in RealDialog, a conversational agent for customer relationship management.

2 Dialogue Management

A long-term goal of the computational linguistics community has been to devise a conversational agent capable of interacting with humans in two-way natural language dialogue. Although a general natural language understanding facility is not in sight, with ever increasing compute cycles at their disposal, designers of conversational agents are approaching their goal at an accelerating pace. In addition to pursuing the fundamental research goals of creating a domain-independent architecture that can provide the language functionalities required by a conversational agent, computational linguists are motivated by the promise of creating conversational interfaces that can serve as the front-end to other, often complex, automated reasoning systems. By augmenting automated reasoning systems with dialogue functionalities, conversational interfaces can facilitate collaborative, mixed-initiative interactions in which problem-solving responsibilities are shared by the user and the application. We believe that a conversational CBR architecture that provides tightly integrated dialogue capabilities can take advantage of the communicative functionalities of conversational agents.

In the broadest formulation of the problem, conversational agents engage in spoken dialogues that are mixed-initiative, i.e., either the human or the agent can have control of the dialogue at a particular “turn” [Seneff, 2002]. These dialogues are characterized by all of the complexities that typify human-human conversations. For example, human-human dialogues frequently exploit the discourse context to effectively communicate using *anaphora* (using context-based referring expressions such as pronouns) and *ellipsis* (employing phrases that omit key syntactic components that are implicit).

Creating an end-to-end spoken dialogue system requires solving two families of problems: speech processing and natural language processing. In the classic architecture, speech recognition and speech synthesis modules bracket the natural language pipeline. The natural language pipeline itself proceeds from natural language understanding through dialogue management and closes the loop with natural language generation. Dialogue management lies at the heart of conversational agent architectures. Dialogue managers are assigned responsibility for two key problems: (1) ensuring that conversations are coherent across multiple interactions, and (2) supporting mixed-initiative interactions that achieve both the user's and the system's goals [Rudnicky and Xu, 1999]. In this work, we draw exclusively from the natural language work on conversational agents.

One can distinguish three fundamental architectures for performing dialogue management [Allen et al., 2000,Rudnicky and Xu, 1999]. First, *graph-based* architectures (sometimes referred to as "call-flow based systems") employ finite state machines to guide all interactions. Graph-based approaches offer the advantage of well-structured dialogues whose give-and-take can be clearly anticipated in advance. If a designer can lay out questions and expected alternate possible responses in a tree, e.g., making an operator-assisted long distance call, then at runtime the conversational agent can respond effectively to each of the possible "moves" that the user can make. However, graph-based approaches suffer from a rigidity that prohibits them from dealing well with unexpected conversational moves. Unless the designer can know in advance with high confidence what possible structure the dialogues will have, graph-based dialogue managers will encounter unexpected statements, questions, and imperatives of users and will fail to react in a manner that is helpful. This limitation is particularly problematic when dialogues are to be mixed-initiative and user-initiated topic shifts are the norm rather than the exception.

Second, *slot-filling* architectures (sometimes referred to as "frame-based" architectures) employ a feature vector with values to be determined during the course of the conversation. Slot-filling dialogue managers permit a broader range of conversations and do not impose the same topic ordering restrictions that graph-based systems do. For example, a simple travel reservation system could use a slot-filling architecture to determine the time of departure and arrival, travel dates, and seating preferences that a prospective passenger might request. Slot-filling architectures work well for conversational agents designed to identify values for a relatively small set of slots. However, they are ineffective for more complicated tasks that require the user and the agent to collaboratively create complex artifacts, e.g., forming a mission plan, creating a multi-faceted travel itinerary, or synthesizing a design [Allen et al., 2001] and where issues such as intent recognition and plan recognition are central.

Third, *plan-based* architectures offer the most general dialogue management capabilities [Allen et al., 2001]. The field has not yet converged on a single plan-based architecture, but many efforts have yielded dialogue managers with one or more of the following features. They may employ a planner to create domain-specific plans, the execution of which will solve the user's problem,[1] they may use an agenda to adaptively drive all conversations, or they may incorporate an array of domain-specific goal-handlers to perform arbitrary computations required to expand particular components of a plan or script that represents the evolving solution. In our work we draw on each of these approaches.

3 The Discourse Goal Stack Model

The Discourse Goal Stack Model (DGSM) is based on a view of CCBR as a specialized form of goal-oriented dialogue. The central tasks of CCBR—selecting appropriate cases, eliciting case descriptions, and responding to requests for clarification or topic changes—can all be viewed as handling specific types of discourse goals. We address these goals through a goal stack that (1) permits all dialogue goals to be handled in a uniform fashion and (2) handles interruptions and subgoals, even when interleaved or nested to arbitrary depth.

DGSM builds on the observation that there is a symmetry between the discourse goals that trigger CBR and the discourse goals that arise in CBR when the facts of a problem description are being elicited. When a system is engaged in a dialogue with a user, the user may make direct or indirect requests for information that can only be answered if the system elicits additional information from the user. For example, if the user requests information that can be provided through CBR, such as diagnostic or product selection information, the system generally must elicit facts specific to the user's request, such as symptoms or product requirements. Invoking CBR is thus one way among many of satisfying the discourse goal of providing information to a customer.

Similarly, during CBR a user may fail to understand a question, be unable or unwilling to find out the answer, or temporarily change the subject. Each of these events gives rise to new system discourse goals, such as answering the user's request for clarification or satisfying the system's need to clarify a question.

DGSM consists of a goal stack, a collection of discourse goal types, a forest of augmented transition networks (ATNs) in which nodes are discourse goals and arcs are speech acts by the user or the system, and a goal handler responsible for determining the appropriate action to take in response to the goal at the top of the stack. The goal handler selects from among the following actions, based on the value of the current top-of-stack and the most recent speech act by the user:

1. If the current goal corresponds to a node in an ATN and the user's utterance is recognized as a speech act matching a transition from that node, the goal handler pops the stack and pushes the node at the end of the transition.
2. If the current goal corresponds to a node in an ATN but the transition contains a speech act by the system, the speech act is generated, the stack is popped, and the top of stack is replaced by the node at the end of the transition.

[1] Similar techniques are used in non-natural-language-based approaches to mixed-initiative interaction, e.g., [Rich and Sidner, 1998].

3. If the user's utterance doesn't correspond to a transition from the state at the top-of-stack but matches an initial transition in another ATN, the utterance is interpreted as a change of topic. The state at the end of the transition is therefore pushed onto the stack.
4. If the state at the top-of-stack is the end state of an ATN, it is popped.
5. If the top-of-stack is a goal that can only be achieved by an external module, such as the case-based reasoner, a constraint-satisfaction problem solver, or an inference engine, the module is invoked. External modules may themselves generate discourse goals.

The algorithm for DGSM's goal handler is depicted in Figure 1.

4 Case-Based Reasoning in DGSM

In DGSM, CBR is invoked when the goal handler encounters a *selection goal*, that is, a goal that requires selecting one element from a set of entities, such as diagnoses or inventory items, based at least in part on information provided by, or specific to, the user. DGSM is consistent with the standard CCBR model [Aha et al., 1998] in assuming that each case is specified by a unique set of attribute/value pairs. Associated with each attribute is question text and type information specifying acceptable answers (described in more detail below in Section 4.1).

When a selection goal is encountered, the selection handler instantiates a *caseCollection* object with a collection of initial hypotheses corresponding to the selection goal. For example, if the system interpreted a statement by the user as a request to troubleshoot a printer[2] and the system had a collection of cases corresponding to the goal of selecting a printer diagnostic state, a caseCollection would be instantiated and the CBR module invoked by pushing its start state onto the stack.

Initially, all cases associated with the specific selection goal are candidates. The CBR module iteratively selects the question that discriminates best among the current candidate cases and poses it to the user until a unique case remains or there are no more questions that can discriminate among the remaining candidates. If there is a unique case, it is reported to the user; otherwise, the system reports a failure.

Figure 2 depicts the CBR module. Circles represent discourse goals, squares and diamonds represent procedures and branches, respectively, unitalicized arc labels represent the propositional content of speech acts, and the arc labeled "Call *Directed Elicitation(Q)*" causes the Directed Elicitation ATN (shown in Figure 3) to be invoked by pushing its start state onto the goal stack.

Because there is a unique set of hypotheses for each selection goal, DGSM is not limited to CBR in a single domain, but can handle an arbitrary number of distinct selection goals.

[2] The techniques for interpreting statements by users used in RealDialog are set forth below in Section 5.

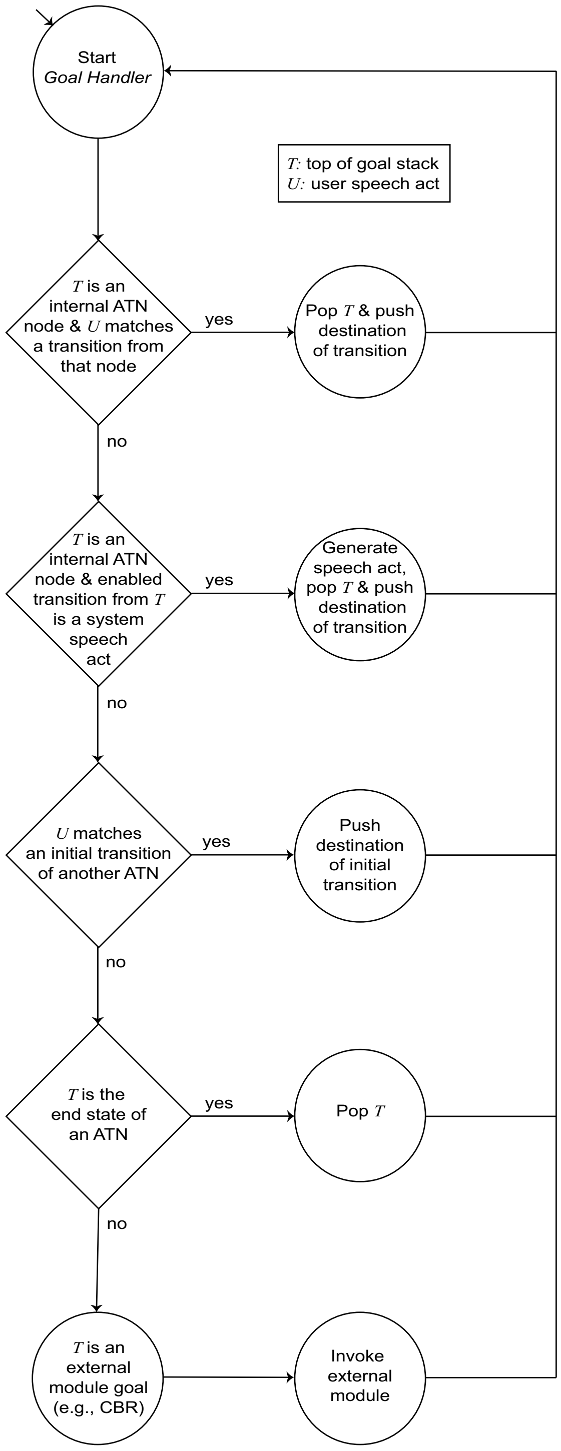

Start
Goal Handler
T: top of goal stack
U: user speech act
T is an internal ATN node & *U* matches a transition from that node
yes
Pop *T* & push destination of transition
no
T is an internal ATN node & enabled transition from *T* is a system speech act
yes
Generate speech act, pop *T* & push destination of transition
no
U matches an initial transition of another ATN
yes
Push destination of initial transition
no
T is the end state of an ATN
yes
Pop *T*
no
T is an external module goal (e.g., CBR)
Invoke external module

Fig. 1. The DGSM goal handler.

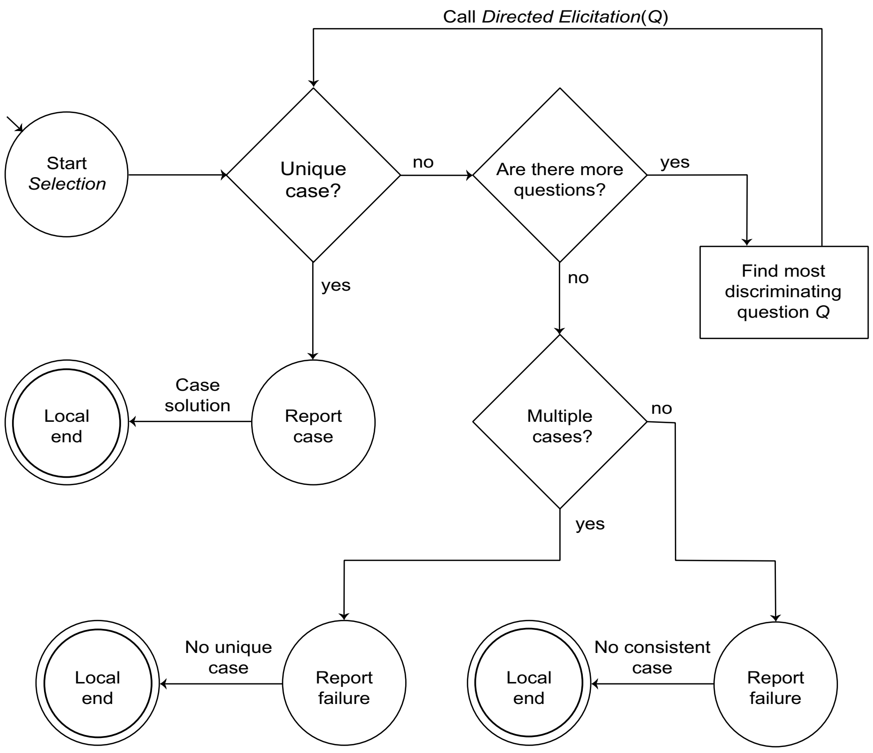


Fig. 2. The DGSM CBR module.

4.1 Directed Elicitation

Directed elicitation is a general mechanism for leading the user to provide information specific to a selection goal while permitting interruptions and clarification questions by either the system or the user. As mentioned above, each case attribute is associated with a question text and a specification of the acceptable answer type. After the CBR module has selected the attribute that discriminates best among the current hypotheses, it poses the question corresponding to the attribute and invokes directed elicitation with the required answer type.

Figure 3 illustrates the structure of directed-elicitation ATNs. A directed-elicitation ATN is invoked by pushing its start state onto the goal stack. The goal handler asks the question corresponding to the transition from this start state, then compares the user's utterance with the transition that is expected from the *get answer* state. If the utterance expresses a value of the expected type found, the value is recorded in a *conversation variable*, a global variable representing information specific to the current dialogue. If the utterance doesn't match the expected transition, then the goal handler searches for alternative ATNs with initial transitions that match the utterance. If one is found, the utterance is interpreted as an interruption, and the start state of the ATN with the matching initial transition is pushed onto the stack. When the local end state of this ATN is reached, it is popped and the dialogue context in which the interruption occurred is restored.

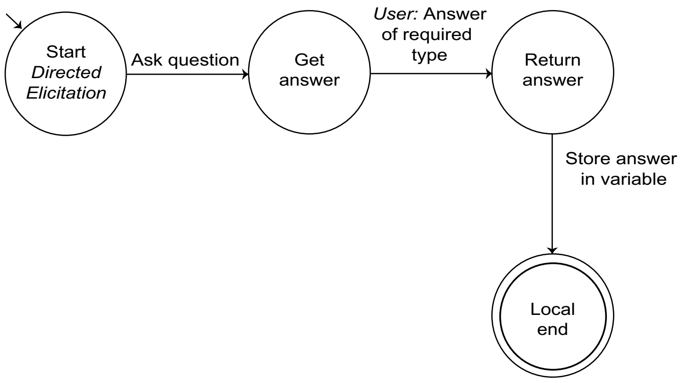


Fig. 3. The structure of directed-elicitation ATNs. A separate directed elicitation ATN exists for each question type.

Typical directed-elicitation ATN types include the following:

– Yes/no
– Selection from list
– Free text
– Integer

- Real
- Integer range
- Date
- Date range
- Date list

In addition to a directed-elicitation ATN, it is desirable to have a termination ATN, as shown in Figure 4, to permit the user to end the CBR episode without ending the dialogue with the system.

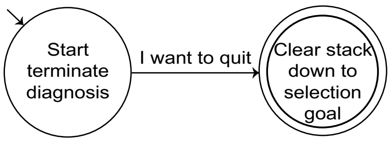


Fig. 4. An ATN to permit a CBR episode to be terminated without ending the user's dialogue with the system.

4.2 Example

The following highly simplified example illustrates how the DGSM model permits interruptions to be handled during a CBR interaction. Suppose that there is a diagnostic case library containing the following cases:

1. influenza
   - fever: true
   - rash: false
2. measles
   - fever: true
   - rash: true
3. dermatitis
   - fever: false
   - rash: true
4. food poisoning
   - fever: false
   - nausea: true

Suppose that the dialogue system includes the ATN shown in Figure 5, which is intended to recognize when a request for diagnosis has been made and to trigger a diagnosis selection.

Consider the following dialogue between a user and a system:

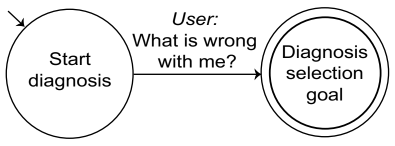


Fig. 5. An ATN to identify when a request for diagnosis has been made and to trigger a diagnosis selection.

```
...
User: What is wrong with me?
Agent: Do you have a fever?
User: How do I tell if I have a fever?
Agent: Measure your temperature with a thermometer.
User: How high of a temperature is a fever?
Agent: A temperature above 37 is considered a fever.
User: Then I don't have a fever.
Agent: Do you have a rash?
...
```

In order for CBR to be triggered, the system must recognize that the user's first utterance matches the transition shown in Figure 5. When the goal handler recognizes the utterance matching the initial (and only) transition of the Figure 5 ATN, it pushes the "Diagnosis selection goal" state onto the stack. The goal handler instantiates a caseCollection object with the cases corresponding to this diagnosis goal, i.e., the four cases shown above, pops the goal, and pushes the first state of the CBR module onto the stack.

Since there is not a unique case and there are multiple remaining questions, the CBR module finds the most discriminating attribute, in this case *fever*, poses the question text to the user ("Do you have a fever?"), and pushes the start state of the directed elicitation ATN for *Yes/No* onto the stack.

The user's answer, "How do I tell if I have a fever?", is not of the expected type (it is neither an affirmative nor a negative). If the system contains the ATN shown in Figure 6, the goal handler (following the third branch in the diagram shown in Figure 1) will push the second state of the top ATN in Figure 6 onto the stack, produce the text corresponding to the transition from that state, "Measure your temperature with a thermometer," and pop the local end state, returning the stack to a state in which the top-of-stack is the second state in the directed elicitation ATN.

The user's statement, "How high of a temperature is a fever?", once again does not match the transition from the state at the top of the goal stack, so again the system finds an ATN whose initial transition matches the user's utterance, i.e., the lower ATN in Figure 6, pushes the second state in this ATN onto the stack, and produces the text corresponding to the transition from that state, "A temperature above 37 is considered a fever."

The user's statement, "Then I don't have a fever," matches the transition in the directed-elicitation ATN corresponding because it is a negative. The negative response is recorded in a conversation variable, the local end state of the directed-elicitation ATN is popped, and the "Call directed elicitation" transition in Figure 2 is completed, returning the CBR module to a state in which it tests for a unique hypothesis. Since there is still no unique hypothesis and there is at least one remaining question, directed elicitation is invoked again, this time with the *rash* attribute, giving rise to the system statement "Do you have a rash?"

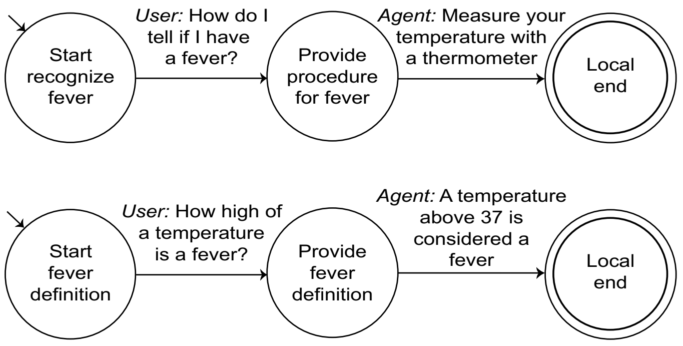

Start recognize fever
User: How do I tell if I have a fever?
Provide procedure for fever
Agent: Measure your temperature with a thermometer
Local end
Start fever definition
User: How high of a temperature is a fever?
Provide fever definition
Agent: A temperature above 37 is considered a fever
Local end

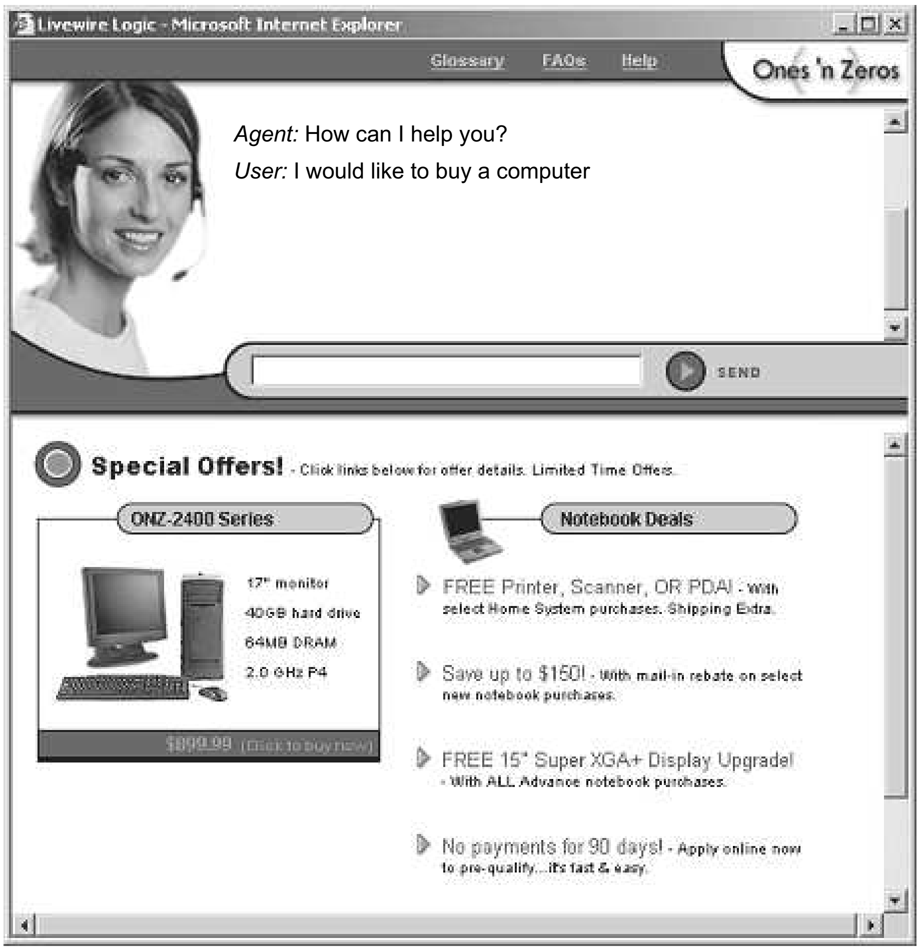
Fig. 6. ATNs providing the procedure to recognize a fever and the temperature threshold for a fever.

This example illustrates how topic shifts introduced by the user's need for additional information to help answer the system's question are handled in a simple and general fashion in a goal-stack architecture.

5 The Implementation of DGSM

Any implementation of DGSM must specify the modality in which users and agents interact and the manner in which user utterances are interpreted. DGSM is implemented in RealDialog, a web-based conversational agent for customer relationship management. While RealDialog is an enterprise software system in use at a number of companies, the implementation of DGSM in RealDialog is an experimental component that is currently in the prototype phase and has not yet been used in commercial installations. RealDialog's interface is shown in Figure 7. Users type queries into a text field, and answers are displayed in a conversation area. Optionally, additional information can be displayed in a web-display panel.

The full details of the interpretation of utterances are beyond the scope of this paper (see [Lester et al., 2004] for a general discussion of utterance interpretation in conversational agent architectures). However, the basic steps are as follows. The first step is tokenization of the user's statement, that is, division of the input in a series of distinct lexical entities. Tokenization includes



Agent: How can I help you?

User: I would like to buy a computer

Special Offers! - Click links below for offer details. Limited Time Offers.

ONZ-2400 Series

17" monitor
40GB hard drive
64MB DRAM
2.0 GHz P4

$899.99 (Click to buy now)

Notebook Deals

- FREE Printer, Scanner, OR PDA! - With select Home System purchases. Shipping Extra.
- Save up to $150! - With mail-in rebate on select new notebook purchases.
- FREE 15" Super XGA+ Display Upgrade! - With ALL Advance notebook purchases.
- No payments for 90 days! - Apply online now to pre-qualify...it's fast & easy.

Fig. 7. The RealDialog interface.

spell-correction and interpretation of apostrophes. The second step is syntactic analysis. In RealDialog, this consists of part-of-speech tagging and parsing. The result of the tokenization, tagging, and parsing is a parse tree.

Parse trees often require *reference resolution*, interpretation of referring expressions, such as the "it" in, "I would like to buy it now." A related problem is interpretation of *ellipsis*, that is, material omitted from a statement but implicit in the conversational context.

In general, a dialogue system must perform some form of pragmatic analysis, that is, determining the speech or communication act [Searle, 1969] that the utterance performs. For example, "Can you reach the salt?" is in the form of a question, but its pragmatic effect is a request for the salt. "I would like to buy it now" is in the form of a declaration, but its pragmatic effect is also a request. Similarly, the pragmatic effect of "Yes" is a request for more in response to "Would you like some more" but the opposite in response to "Have you had enough?"

In a stack-based dialogue architecture like DGSM, pragmatic analysis is typically performed implicitly as a side-effect of the locality of ATNs. For example, the meaning of a Yes/No answer obtained through directed elicitation depends on the question to which the user's utterance is a response. In general, by comparing the user's utterance to the transitions from the current top-of-state, a stack architecture biases the interpretation of an utterance toward the meaning that is most appropriate in the current context.

RealDialog has been in commercial use in enterprise installations since 2002. The primary customers are large commercial enterprises with extensive call centers. RealDialog has been employed both in "outward-facing" deployments, in which it is available to users visiting the business's web site, and "inward-facing deployments" in which it is used by customer service representatives to help find the answers to users' questions more efficiently. The primary functionality used by these applications is simply one-step question answering, but the CCBR component is an implemented component of the system.

6 Summary and Future work

This paper has described an architecture that integrates CBR with a discourse-oriented dialogue engine. This architecture permits CBR or other problem-solving techniques to be selected when needed to achieve pending discourse goals and, conversely, makes the full resources of a dialogue engine available to CBR component to handle topic changes, interruptions, clarification questions by either the user or the system, and other speech acts that arise in problem-solving dialogues.

The DGSM described in this paper is a first step in the integration of CBR with discourse-oriented dialogue engines. In the enumeration of CCBR issues in the introduction, DGSM addresses Issue 2—giving users control over the degree to which the initiative is held by the system or the user—by (1) taking the initiative from the user in response to discourse goals that require questions to be answered by the user but (2) permitting the user to seize the initiative at any point. Issues 4 and 5 are addressed by handling clarifying questions and other interruptions. Issue 7—integrating CBR with other problem-solving modalities– is addressed by embedding the CBR module in a dialogue system that treats all goals in a uniform fashion. Under this approach, a single goal handler can invoke whatever problem-solving modules have been implemented in a given system.

However, several issues are not addressed by DGSM. DGSM does not in itself help with Issue 3, enabling the system to explain either why a question was asked and how the system's answer was reached, and it is completely independent of Issue 1, minimizing questions. Issue 6, enabling the system to ask users clarifying questions, can be addressed in the DGSM framework. However, it is a complex problem of interpretation to recognize when an utterance is relevant but ambiguous, equivocal, or vague (and therefore in need of clarification), as opposed to simply incoherent or irrelevant (and therefore a deviation from the topic).

As noted above, the implementation of DGSM in RealDialog is an experimental component that has not been used in commercial installations. It does not include constraint relaxing dialogues as proposed in [Göker and Thompson, 2000] to recover from situations in which no cases are consistent with the attribute/value pairs specified by the user. RealDialog's tool suite does not currently include an adequate case editor, and the criteria for selecting the most discriminating case attribute is not customizable. However, RealDialog illustrates how CBR can be integrated into a goal-stack architecture and how the resulting integration can significantly improve the flexibility and robustness of conversational case-based reasoning.

References

[Aha et al., 1998] Aha, D. W., Maney, T., and Breslow, L. A. (1998). Supporting dialogue inferencing in conversational case-based reasoning. *Proceedings of the Fourth European Workshop on Case-Based Reasoning, Lecture Notes in Computer Science*, 1488:262–270.

[Allen et al., 2000] Allen, J., Byron, D., Dzikovska, M., Ferguson, G., Galescu, L., and Stent, A. (2000). An architecture for a generic dialogue shell. *NLENG: Natural Language Engineering, Cambridge University Press*, 6.

[Allen et al., 2001] Allen, J. F., Ferguson, G., and Stent, A. (2001). An architecture for more realistic conversational systems. In *Proceedings of the International Conference on Intelligent User Interfaces*, pages 1–8.

[Bridge, 2002] Bridge, D. (2002). Towards conversational recommender systems: A dialogue grammar approach. In Aha, D., editor, *Proceedings of the Workshop in Mixed-Initiative Case-Based Reasoning, Workshop Programme at the Sixth European Conference in Case-Based Reasoning*, pages 9–22.

[Doyle and Cunningham, 2000] Doyle, M. and Cunningham, P. (2000). A dynamic approach to reducing dialog in on-line decision guides. In *European Workshop on Case-Based Reasoning (EWCBR)*, pages 49–60.

[Göker and Thompson, 2000] Göker, M. and Thompson, C. (2000). Personalized conversational case-based recommendation. In *Proceedings of the 5th European Workshop on Case-Based Reasoning*, Trento, Italy. Springer.

[Kohlmaier et al., 2001] Kohlmaier, A., Schmitt, S., and Bergmann, R. (2001). A similiarity-based approach to attribute selection in user-adaptive sales dialogs. In Aha, D. and Watson, I., editors, *Fourth International Conference on Case-Based Reasoning, ICCBR 2001, Lecture Notes in Artificial Intelligence 2080*, pages 306–320, Vancouver, BC, Canada. Springer.

[Lester et al., 2004] Lester, J., Branting, K., and Mott, B. (2004). Conversational agents. In Singh, M., editor, *Practical Handbook of Internet Computing*. CRC Press.

[McSherry, 2001] McSherry, D. (2001). Minimizing dialog length in interactive case-based reasoning. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 993–998, Seattle, Washington, USA.

[McSherry, 2003] McSherry, D. (2003). Increasing dialogue efficiency in case-based reasoning without loss of solution quality. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 121–126, Acapulco, Mexico.

[Rich and Sidner, 1998] Rich, C. and Sidner, C. L. (1998). COLLAGEN: A collaboration manager for software interface agents. *User Modeling and User-Adapted Interaction*, 8(3-4):315–350.

[Rudnicky and Xu, 1999] Rudnicky, A. and Xu, W. (1999). An agenda-based dialog management architecture for spoken language systems. In *Proceedings of Workshop on Automatic Speech Recognition and Understanding*, Keystone, CO.

[Searle, 1969] Searle, J. (1969). *Speech Acts: An Essay in the Philosophy of Language.* Cambridge University Press, Cambridge.

[Seneff, 2002] Seneff, S. (2002). Response planning and generation in the Mercury flight reservation system. *Computer speech and language*, pages 283–312.