

Building Intelligent Dialog Systems ^{*†}

Susan W. McRoy[‡] Syed S. Ali Angelo Restificar Songsak Channarukul

Natural Language and Knowledge Representation Research Group

University of Wisconsin-Milwaukee

Abstract

We overview our recent work in specifying and building intelligent dialog systems that collaborate with users for a task. As part of this work we have specified and built systems for: giving medical students an opportunity to practice their decision making skills in English (B2); performing template-based natural language generation (YAG); detecting and rebutting arguments (ARGUER); recognizing and repairing misunderstandings (RRM); and assessing and augmenting patients' health knowledge (PEAS). All of these systems make use of rich models of dialog for human-computer communication.

Keywords: intelligent dialog systems, argumentation, natural language generation, misunderstanding, and repair

1 Introduction

Computers have given people the power to process vast quantities of information, quickly and effectively. However, computers tend to respond to requests for information by presenting either too much information (such as a recommendation to buy a stock, accompanied by tables of historical data and glitzy graphics), or virtually no information (such as a recommendation to buy a stock with little or no justification). By contrast, people avoid these extremes through dialog, whereby information is presented and processed incrementally, assessment (did you understand what I meant?) is ongoing, and participants can direct the course of the dialog.

Computers have made possible this problem of “information overload.” Through computers, people everywhere have access to an immense amount of information; however, the information that is useful is lost among the majority that is not. The interaction can resemble that of Figure 1, where the user can be overwhelmed (or underwhelmed).

^{*}This work was supported by the National Science Foundation, under grants IRI-9701617 and IRI-9523666 and by a gift from Intel Corporation.

[†]This article is an edited version of an article that will appear in *intelligence*, Spring 1999, 1(10), and is copyright ©1999 by the Association for Computing Machinery, Inc. Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Publications Dept, ACM Inc., fax +1 (212) 869-0481, or permissions@acm.org.

[‡]Contact author's address and email: Department of EECS, University of Wisconsin-Milwaukee, 3200 N. Cramer Street, Milwaukee, WI 53211, mcroy@uwm.edu



Figure 1: The Worst-Case Example of Human-Computer Communication

Part of the difficulty is that the flow of information between people and computers is usually very different from the flow of information between two people. Computers often present large quantities of information and expect relatively little feedback. People tend to communicate information to each other incrementally. After each unit of communication, a speaker will pause briefly to allow listeners to provide feedback on what has just been said or to add his or her own contribution to the dialog. When given feedback, a speaker will interpret the feedback with respect to his beliefs and goals. For example, he or she will consider whether the listener understands, whether the communication is having the desired effect, and how subsequent communication might be made more effective. Ideally we want the communication to be balanced as shown in Figure 2.

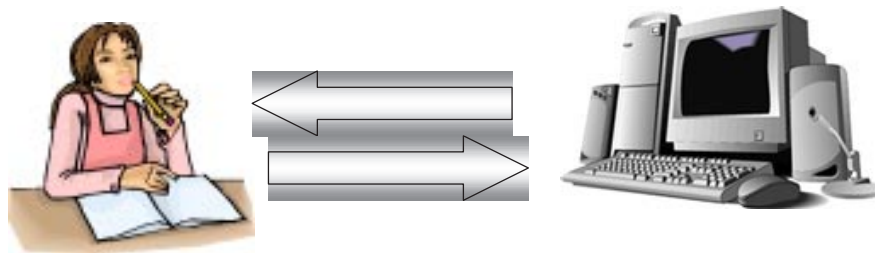


Figure 2: The Better-Case Example of Human-Computer Communication

Computers tend to provide a lot of information all at once, providing few, if any, opportunities for feedback from the user. Moreover, a computer system will rarely attempt to “understand” what feedback it does allow, so that only feedback anticipated by the software designer impacts system performance. Indirect feedback, such as asking the same question twice, is not

interpreted; thus, the typical computer system will simply repeat its pre-programmed response, rather than attempting to address the underlying difficulty. People would rarely make this mistake (unless they had reason to believe that their initial utterance was not heard).

1.1 Dialog *vs.* Presentation

A dialog is two-way interaction between two agents that communicate. People communicate efficiently and effectively using dialog. Computers do not, typically, engage in dialog with people. Intelligent dialog systems (IDS) are concerned with the effective management of an incremental dialog between a computer system and its user. An IDS must deal with both input and output and must monitor the effectiveness of its actions. During the course of an interaction, content to be presented, as well as the system's model of the user (for example the user's apparent level of expertise), changes dynamically. Intelligent dialog requires the representation of goals, intentions, and beliefs including beliefs about the effectiveness of the interaction. The intelligence of a dialog system lies in the traditional utility of a dialog model, which includes the ability to interpret a variety of communicative forms whose meaning depends on the context, including fragments, anaphora, and follow-up questions. In our view, dialog systems must also recognize situations that require dynamic adaptation, including misunderstanding, non-understanding, and argumentation.

By contrast, intelligent presentation systems are concerned with the effective presentation of a fixed content subject to constraints (for example, the user's apparent expertise and preferred presentation style). Their "intelligence" lies in sophisticated domain and presentation models used to adapt presentations. However, such presentations are planned and displayed in their entirety (without intermediate feedback from the user) and thus do not allow the system to monitor the effectiveness of the presentation to deal with ambiguity, misunderstanding or non-understanding.

1.2 An Architecture for Intelligent Dialog Systems

The general goal of our work is to investigate computational models of dialog that can support effective interaction between people and computer systems. Our research also aims to provide computational methods for integrating and using this information to produce relevant utterances and to identify and resolve communication problems as they arise.

Our work thus addresses two important issues for the design of intelligent dialog systems. The first concern is to specify what tasks an IDS must support. The second concern is to specify what knowledge or content an IDS needs to represent. Our answers to these questions are related. In our view, the reasoning tasks that an IDS must support include reasoning about the coherence of the dialog—in particular it must respond in a way that will be perceived as coherent and must recognize potential sources of communication breakdown, such as misunderstanding or argumentation. An IDS must also maintain a model of the user's beliefs and interests as well as a history of the communication between the user and the system. To support these tasks, an IDS must include representations of the linguistic, intentional, and social knowledge that people use to understand dialog. Moreover, because interactive, incremental dialog is a time-constrained process, an IDS must be able represent and reason about knowledge at multiple levels of abstraction, allowing it to allocate its reasoning efforts opportunistically.

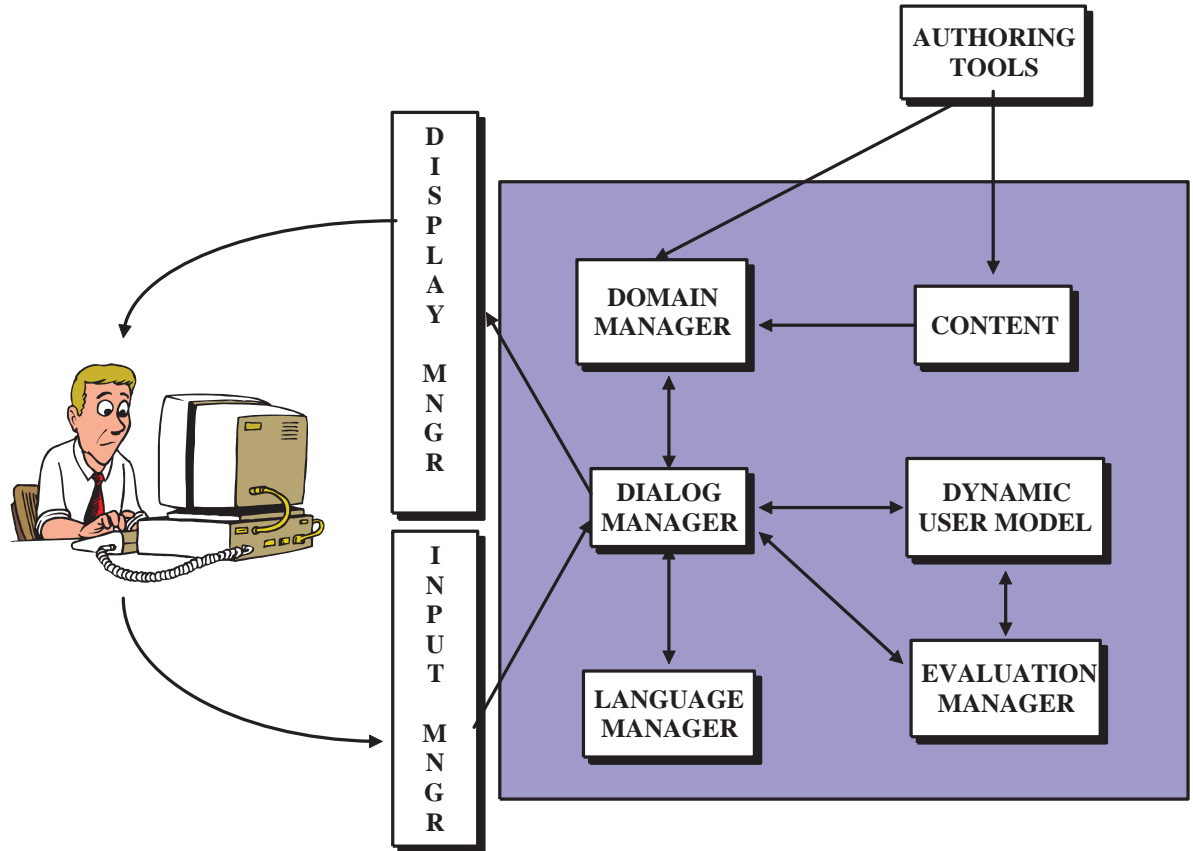


Figure 3: Architecture of an Intelligent Dialog System

Our architecture for Intelligent Dialog Systems is shown in Figure 3. The **INPUT MANAGER** and **DISPLAY MANAGER** deal with input and output, respectively. The input modalities would include typed text, spoken text, mouse clicks, and drawing. The output modalities would include text, graphics, speech and video. The **DIALOG MANAGER** is the component through which all input and output passes. This is important because the system must have a record of everything that occurred (both user and system-initiated). If the user chooses to input language, the **LANGUAGE MANAGER** is handed the text to parse and build the appropriate representation which is then interpreted by the dialog manager. The **DOMAIN MANAGER** component will be comprised of general rules of the task as well as specific information associated with how the **CONTENT** is to be presented. The content will be generated, prior to system use, by the use of **AUTHORING TOOLS** that allow the rapid development of the content. Based on the ongoing interaction, as well as information provided by the user, **USER BACKGROUND & PREFERENCES** are tracked. The status of the interaction is evaluated incrementally by the **EVALUATION MANAGER**, which affects the ongoing dialog and user model.

This architecture builds on our prior work, where the user is on a “thin” client personal computer interacting with a server that contains all the components described.

2 Specific Projects and Their Significance

The Natural Language and Knowledge Representation Research Group (<http://tigger.cs.uwm.edu/~mcroy/nl-kr.html>) at the University of Wisconsin-Milwaukee has been working on a number of projects that address issues needed for designing Intelligent Dialog Systems.

- B2 gives medical students an opportunity to practice their decision making skills by communicating in English;
- YAG allows a system to generate English language utterances;
- ARGUER recognizes and rebuts arguments;
- RRM recognizes and repairs misunderstandings;
- PEAS provides an architecture for assessing and augmenting patients' health knowledge; one component is LEAF, a dynamically customized medical history form.

These projects are researching issues in communication in natural language, dialog management, and the use of user models to tailor the interaction between the user and the computer. In particular, B2 and YAG support flexible, yet time-constrained interaction, by incorporating representations of knowledge at multiple levels of abstraction, for natural language input and output, respectively. ARGUER and RRM reason about the coherence of dialog and include representations of the type of knowledge needed to recognize and address arguments and misunderstanding, respectively. PEAS is an architecture for using knowledge about the user to tailor the interaction between the user and the computer system; LEAF (Layman Education and Activation Form) is a component within this architecture.

2.1 B2 (<http://tigger.cs.uwm.edu/~b2>)

B2 is a general purpose tutoring system that allows students to practice their decision-making skills in a number of domains (McRoy, 1998; McRoy et al., 1997; McRoy et al., 1998a). B2 supports mixed-initiative interaction using a combination of typed English utterances and point-and-click based communication using a mouse. To address the ambiguity that occurs in these utterances, without sacrificing generality, B2 analyzes user inputs incrementally and opportunistically. In particular, it uses a parser with a linguistically based grammar to process the student's typed inputs and produce a structure that captures syntactic marking and bracketing, along with some conceptual information. We call this a *mixed-depth* representation. Encoding decisions that require reasoning about the domain or about the discourse context are left to subsequent processing.

Subsequent processing incorporates the mixed-depth representations into a five-level model of discourse, shown in Figure 4. These levels capture everything that the student and the system have said, as well as how their utterances extend the ongoing discourse. They are needed to allow the system to interpret context-dependent utterances such as *Why?* or to deal with unexpected utterances, such as misunderstandings or arguments.

interpretation of exchanges
exchanges (pairs of interpretations)
system's interpretation of each utterance
sequence of utterances
utterance level

Figure 4: Five Levels of Representation

The utterance level is a representation of what the user typed or selected with a mouse, as produced by the parser. The second level corresponds to the sequence of utterances, which enables the system to reason about temporal ordering constraints. (This level is comparable to the linguistic structure in the tripartite model of (Grosz and Sidner, 1986)). The third level comprises the system's interpretation of each utterance. Each utterance event (from level 1) will have an associated system interpretation, which corresponds to a communicative act (such as question or command) which may reference entities from the underlying task (such as a specific medical case). The fourth and fifth levels of the discourse model are exchanges and interpretations of exchanges, respectively. A *conversational exchange* is a pair of interpreted events that fit one of the conventional structures for dialog (*e.g.*, QUESTION-ANSWER). The interpretation of an exchange indicates how the exchange fits with previous ones, such as whether it manifests understanding, misunderstanding, agreement, or disagreement.

Figure 5 shows a glossed (for space reasons) portion of the representations built during a short dialog (the underlying knowledge representation used is the SNePS semantic network formalism (Shapiro et al., 1994; Shapiro and Rapaport, 1992; Shapiro and Rapaport, 1987)). The yellow circles (called nodes) represent the sequence level, the blue nodes the utterance level (glossed as text here), the pink nodes are B2's interpretation of the utterances, and the green nodes are the exchange structure (the interpretation of exchanges level is not shown). This particular dialog illustrates a system misunderstanding of the question, *Why does a positive HIDA suggest gallstones*, as a request for the system to explain the probability relations between gallstones and HIDA, then a request by the user for a repair, followed by the system's re-interpretation of the question and its subsequent generation of a new response. (B2 uses RRM to accomplish this repair; see Section 2.4).

2.2 YAG (<http://tigger.cs.uwm.edu/~yag>)

YAG (Yet Another Generator) generates natural language in real-time, as required by Intelligent Dialog Systems. YAG combines template-based approaches for the representation of text with knowledge-based methods for representing content (*i.e.*, inputs are concepts or propositions along with optional annotations to specify syntactic constraints).

YAG's architecture is layered and is shown in Figure 6. The inner layer (**surface-1**) takes an input that contains parameters used to fill (instantiate) a template, as well as the name of the template to be instantiated. This information is used to generate a text. This layer is

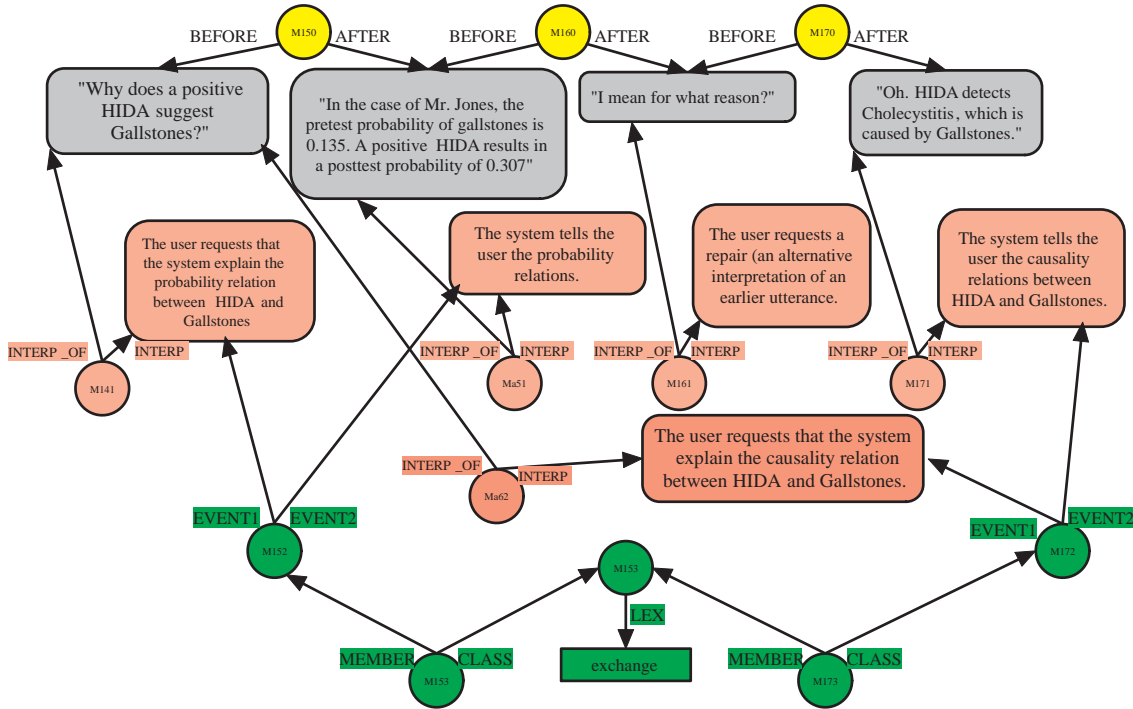


Figure 5: Level Representations in a B2 Dialog

domain independent. The outer layer (**surface-2**) accepts knowledge representation structures (here, semantic networks) used by an application. The knowledge representation is mapped into a structure suitable for the inner layer and is passed to **surface-1** for further processing. The outer layer can be customized to a domain.

Templates are declarative representations of text structure. They can contain realization information, including syntactic information and conceptual information, at different levels of abstraction. A template is a pre-defined form that is filled by the information specified by either the user or the application. Figure 7 is an example template from YAG. Each form in the template (a list) is a rule that expresses how to realize a surface constituent. This particular template would be used to generate a sentence for expressing propositions of the form `has-property(agent, pname, pval)`, such as `has-property(John, age, 20)`.

In the template in Figure 7, if **agent** = "John", **pname** = "age", and **pval** = "20", the surfaced text will be "John's age is 20.". The template has four parts, for generating the subject, verb, property, and punctuation, in order. The first rule, which is a condition rule, has two alternatives. (Such alternatives are called *options*; in each option, the condition is given last.) In the first rule, the second option is chosen because **pname** has been specified. Within this option, the **agent** is generated as a possessive, "John's ", followed by the value of the feature **pname** (which is the string "age"). Next, the verb rule is executed. The verb "be" together with its features, **SUBJECT** = **agent** and **TENSE** = **present** generates the verb "is". The third rule is another condition rule. The first option fails, because no **property** is specified. The second option applies because **pval** has been specified (**pval** = 20). Thus, this third rule generates the value of the feature **pval** (i.e. 20). The final rule always returns

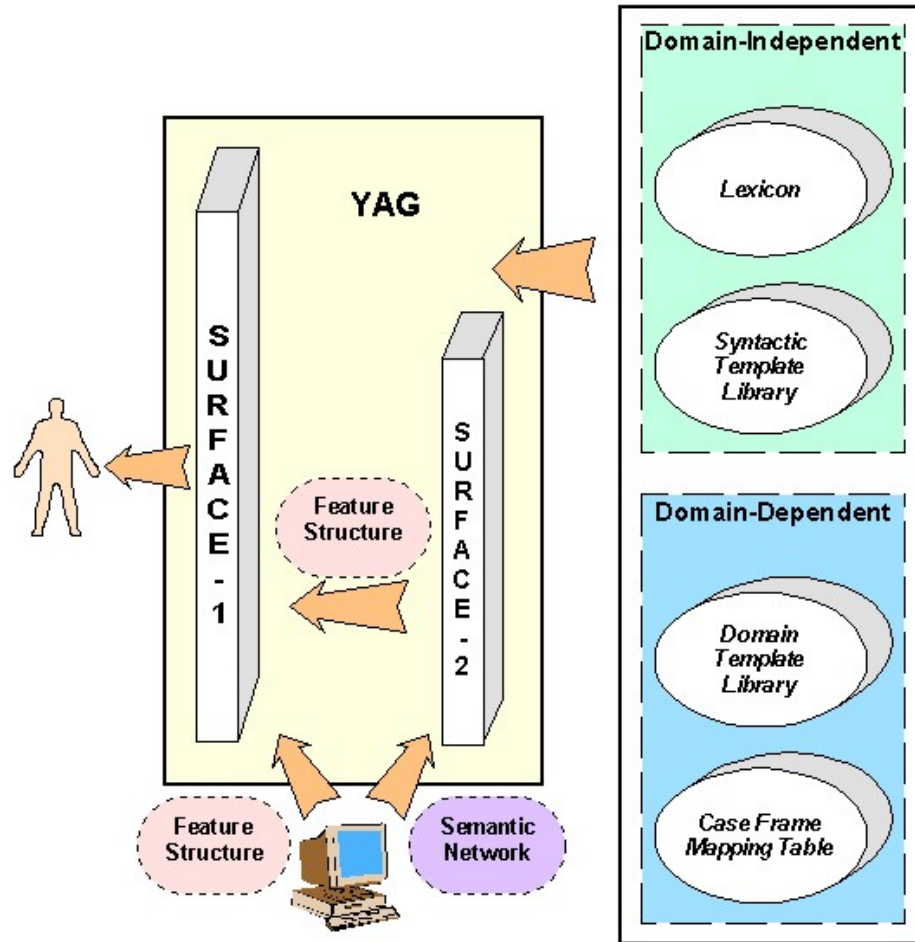


Figure 6: Architecture of YAG

the period punctuation. Finally, all outputs are concatenated in order and returned as a string "John's age is 20."

The advantages of template-based approaches include:

- Speed: Templates do not require full grammar unification and are stored in a template library indexed using a hash table.
- Comprehensibility: Templates are intrinsically declarative.
- Flexibility: Templates are very general. They can be parameterized to realize different text.
- Extensibility: Templates can reuse other templates to realize complex text.
- Reusability: General purpose templates (comprising a general-purpose, template-based generation grammar) can be defined to realize domain-independent text.

To speed the process of template development, YAG includes a graphical tool for developing new templates called TATTOO. TATTOO (Template Authoring and Testing TOOL) allows

```

((C ((0 (E agent (SUBJECTIVE))
      (equal pname NIL))
     (0 ((E agent (POSSESSIVE))
        (F pname)
        (not (equal pname NIL))))))
 (V "be" ((SUBJECT . agent) (TENSE . present)))
 (C ((0 (F property)
      (not (equal property NIL)))
     (0 (F pval)
        (not (equal pval NIL))))))
(S ".")
)

```

Figure 7: An Example Template

users to graphically construct templates that can be tested immediately with just a mouse click. TATTOO includes facilities for template re-use and syntax-checking. Figure 8 illustrates the development of the template of Figure 7.

2.3 ARGUER: (<http://tigger.cs.uwm.edu/~arguer>)

Intelligent dialog systems must be prepared to deal with argument. ARGUER (Argument Detection and Rebuttal) is an argumentation system that allows an Intelligent Dialog System to recognize when another agent disagrees with it and to attempt to convince the other agent to adopt the system's view. The approach involves the use of agent models and user-adapted interaction techniques to process utterances and generate appropriate responses.

Figure 9 depicts the architecture of ARGUER. The system has access to the Domain Knowledge Base (domain knowledge), Argumentation Knowledge Base (argumentation knowledge, *i.e.* knowledge about the structure of arguments), and the Commonsense Knowledge Base. The system is presumed to be the domain expert. It has access to the user's belief models (because it is the system's models of the users). Depending on whose turn it is, the system attempts to interpret whether the current utterance attacks or supports previous claims (which may be any prior utterance).

When the user inputs an utterance, the system will attempt to interpret the user's utterance as an attack or support on a prior utterance of the system. It does so by asking, What does the user's utterance attack? and second, What does the user's utterance support? All reasoning to answer these questions occurs in the user's belief model and makes use of all the knowledge sources therein. The system's utterances will attempt to attack or support the previous utterances of the user (and failing that provide supporting arguments to prior system utterances). In addition to the knowledge sources it has access to, the reasoning also takes into consideration the system's model of the user (user model).

The underlying principle for detecting arguments in ARGUER is to find a general case of an argument schema into which the meaning representation of an utterance can be matched.

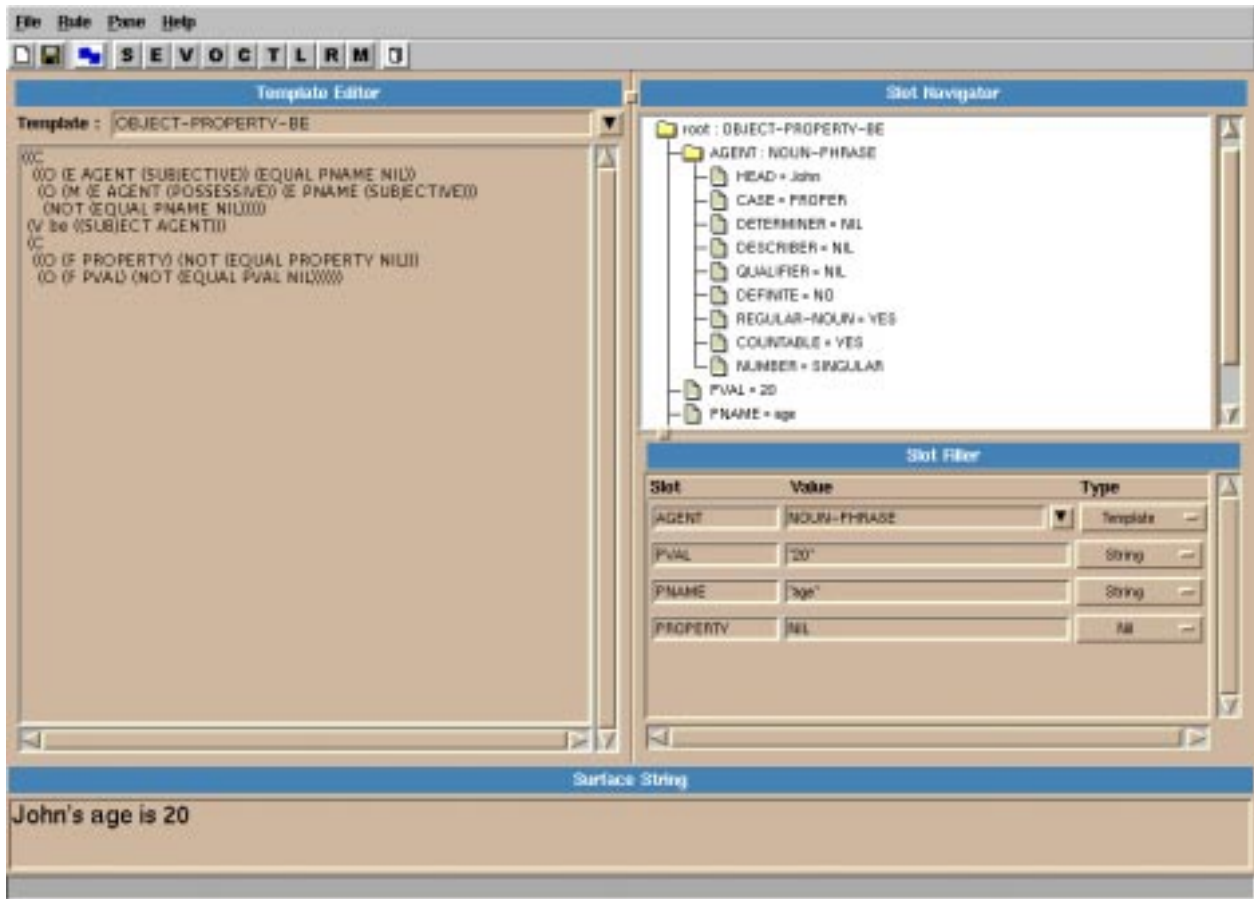


Figure 8: TATTOO Template Composer

Argument detection and rebuttal are established via argument schema rules. An argument schema rule characterizes the structure of an argument. If a matching schema is found, the corresponding variables in the argument schema rule is instantiated, thereby establishing attack or support relations.

The example shown in Figure 10 illustrates an argument. In the figure, S1 and S2 are utterances of the system. U1 is the user's utterance. The argument schema can be used to detect the argument apparent from the first pair, S1 and U1. In the argumentation knowledge-base, we have an argument schema rule: If X is an utterance implying Y, then NOT Y is an attack to X. Rules in the common-sense and domain knowledge bases allow us to derive that requiring w of A (which follows from the imperative form of S1) implies the need of A for w. S1 is the utterance implying Y = "there is a need for a blood pressure check." Thus, using the above argument schema rule, (NOT Y) = "there is no need" (for a blood pressure check) is an attack to X = S1.

The argument schema can also be used to generate the rebuttal. Suppose a participant in the dialog says S1. Using the argument schema rule describe above, Y is instantiated. This in turn allows us to use NOT Y as an attack to X = S1 and hence the other agent, to rebut S1, can say, "There is no need".

The use of argument schemata for argument detection and rebuttal allows argument rela-

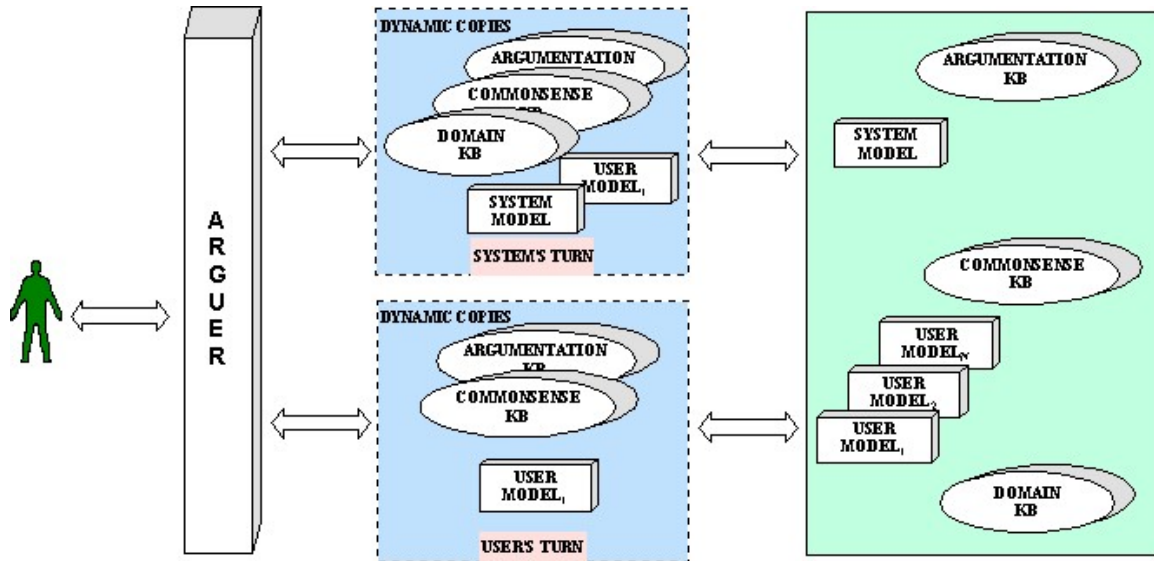


Figure 9: Architecture of ARGUER

S1 Have your blood pressure checked.

U1 There is no need.

S2 Uncontrolled high blood pressure can lead to heart attack, heart failure, stroke or kidney failure.

Figure 10: An Example Argument

tions between propositions to be established dynamically. Moreover, the method is incremental in that it allows processing of each piece of the utterance and uses only a part of the argument to continue.

2.4 RRM

RRM (The Recognition and Repair of Speech Act Misunderstandings) provides a unified account of speech-act production, interpretation, and repair (McRoy, 1995; McRoy and Hirst, 1995). These tasks are essential to the management of intelligent dialogs, because opportunities for errors in communication are unavoidable:

- The user's attention might not be focused on the aspect of the presentation that the system expects.
- The user might not have the same understanding of what a verbal description or a graphical image is meant to convey.
- The user might lack some of the requisite knowledge of the domain necessary to interpret a proposed explanation.

Thus, any computer system that communicates must be able to cope with the possibility of miscommunication—including misunderstanding, non-understanding, and misinterpretation. In *misunderstanding*, one participant obtains an interpretation that she believes is complete and correct, but which is, however, not the one that the other speaker intended her to obtain. In *non-understanding*, a participant either fails to obtain any interpretation at all, or obtains more than one interpretation, with no way to choose among them. In *misinterpretation*, the most likely interpretation of a participant’s utterance suggests that their beliefs about the world are unexpectedly out of alignment with the other’s. All three forms of miscommunication can eventually lead to repair in a dialog; however, misinterpretations and non-understandings are typically recognized immediately, whereas a participant is not aware, at least initially, when a misunderstanding occurs. Additionally, misinterpretation can be a source of misunderstanding.

RRM addresses possible misunderstandings (as well as expected interpretations), while respecting the time-constraints of Intelligent Dialog Systems, by combining intentional and social accounts of interaction to capture the expectations that help constrain interpretation. Intentions help focus the system on interpretations that support tasks that the user or system is pursuing; social conventions help focus the system on interpretations that fit into known types of conversational exchanges (see Section 2.1). The approach to interpretation is abductive—the system must try to explain how communicative acts performed by the user relate to prior discourse. These explanations require the system to make (possibly unsound, but reasonable) assumptions about the user’s goals, expectations, or misunderstanding. An action is considered a manifestation of misunderstanding if there is no coherent link apparent and there is a reason for supposing that misunderstanding has occurred.

2.5 PEAS (<http://tigger.cs.uwm.edu/~peas>)

The Patient Education and Activation System (PEAS) is a design for a collection of tools that help prepare people for discussions with a doctor and the choices that they will be asked to make (McRoy et al., 1998b). These tools support the following tasks: Assessing the user’s health background, concerns, and preferences (knowledge acquisition); providing customized/filtered information about health-related topics (knowledge delivery/information retrieval); and evaluating users’ knowledge of health topics (testing).

The most-developed of these tools is LEAF (Layman Education and Activation Form, <http://tigger.cs.uwm.edu/~alp/LEAFV1.1/leaf.html>), which aims to create a more informed and “activated patient” by extending the normal activity of filling in a medical history form to include educational activities that help patients understand the terminology of the form and suggest topics that they might want to discuss with their doctor. For example, if the patient has indicated that they have a preventable medical condition, the system will offer information about prevention and treatment strategies. It will also suggest questions that she might ask her doctor. Unlike a brochure, the presentation of this information is incremental, and interactive, allowing the system to monitor the patient’s attention and level of understanding and adapt the interaction appropriately. LEAF will also filter out irrelevant parts of the form, which, because they are irrelevant, are most likely to contain terminology that is unfamiliar or confusing. A prototype version of LEAF is available over the Internet, for demonstration purposes.

3 Summary

We have presented the recent work of the Natural Language and Knowledge Representation Research Group (NLKRRG) at the University of Wisconsin-Milwaukee. This work is specifying and building intelligent dialog systems that collaborate with users. We have specified and built systems for: giving medical students an opportunity to practice their decision making skills in English (B2); performing template-based natural language generation (YAG); detecting and rebutting arguments (ARGUER); recognizing and repairing misunderstandings (RRM); and assessing and augmenting patients' health knowledge (PEAS). All of these systems make use of rich models of dialog for human-computer communication.

Additional information about these projects is available on the web at the URLs provided.

References

- Grosz, B. J. and Sidner, C. L. (1986). Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204.
- McRoy, S. (1998). Achieving Robust Human-Computer Communication. *International Journal of Human-Computer Studies*, 48:681–704.
- McRoy, S., Haller, S., and Ali, S. S. (1997). Uniform Knowledge Representation for NLP in the B2 System. *Natural Language Engineering*, 3(2):123–145.
- McRoy, S. W. (1995). Misunderstanding and the negotiation of meaning. *Knowledge-based Systems*, 8(2–3):126–134.
- McRoy, S. W., Haller, S. M., and Ali, S. S. (1998a). Mixed Depth Representations for Dialog Processing. In *Proceedings of Cognitive Science '98*, pages 687–692. Lawrence Erlbaum Associates.
- McRoy, S. W. and Hirst, G. (1995). The repair of speech act misunderstandings by abductive inference. *Computational Linguistics*, 21(4):435–478.
- McRoy, S. W., Liu-Perez, A., and Ali, S. S. (1998b). Interactive Computerized Health Care Education. *Journal of the American Medical Informatics Association*, 5(4):76–104.
- Shapiro, S., Rapaport, W., Cho, S.-H., Choi, J., Feit, E., Haller, S., Kankiewicz, J., and Kumar, D. (1994). A dictionary of SNePS case frames. Online techreport available at URL: <http://www.cs.buffalo.edu/pub/sneps/WWW/Manuals/dictionary.ps>.
- Shapiro, S. C. and Rapaport, W. J. (1987). SNePS considered as a fully intensional propositional semantic network. In Cercone, N. and McCalla, G., editors, *The Knowledge Frontier*, pages 263–315. Springer-Verlag, New York.
- Shapiro, S. C. and Rapaport, W. J. (1992). The SNePS family. *Computers & Mathematics with Applications*, 23(2–5):243–275.