Argument Detection and Rebuttal in Dialog[1]

Angelo Restificar[1] Syed S. Ali[2] Susan W. McRoy[1]
angelo@uwm.edu syali@uwm.edu mcroy@uwm.edu
[1]Electrical Engineering and Computer Science
[2]Mathematical Sciences
University of Wisconsin-Milwaukee

Abstract

A method is proposed for argumentation on the basis of information that characterizes the structure of arguments. The proposed method can be used both to detect arguments and to generate candidate arguments for rebuttal. No assumption of *a priori* knowledge about *attack* and *support* relations between propositions, advanced by the agents participating in a dialog, is made. More importantly, by using the method, the relations are dynamically established while the dialog is taking place. This allows incremental processing since the agent need only consider the current utterance advanced by the dialog participant, not necessarily the entire argument, to be able to continue processing.

Introduction and Motivation

Argument detection is an important task in building an intelligent system that can understand and engage in an argument. In an intelligent dialog system (IDS)[6], an interactive system that tailors its response according to the user's needs and intentions, it is necessary to detect whether an utterance given by the user is an argument against an utterance advanced by the system. Two agents, *e.g.* the system and the user, may engage in a conversation and may not always agree. Each of them may attempt to resolve issues either by attacking an agent's claim or by defending its position. Thus, an IDS must be able to determine whether a proposition advanced by an agent in a dialog attacks a claim currently held by the other agent, supports it, or does neither.

The method we propose here, used by our system ARGUER, finds an argument schema which matches the deep meaning representation of a proposition. The schemata characterize general patterns of argument, similar to those studied by [16, 3]. These types of arguments are reasonable but defeasible—they incorporate plausible assumptions in the absence of complete information. ARGUER uses a truth-maintenance system to revise its beliefs. These schemata are used to establish *support* or *attack* relations among the propositions expressed by agents over the course of a dialog. The schemata allow ARGUER to recognize attack or support relations between an utterance and any prior utterance, not just an immediately preceding one. Separate models of the agent's beliefs are maintained, both for the system and the user. Hence, a proposition believed by the system may not be necessarily believed by the user. The system generates a response by taking into consideration the beliefs held by the user.

The work that we describe here focuses on the problem of relating a proposition to an existing knowledge base for argument detection and rebuttal. By using argument schemas, we will show how a system can detect and rebut arguments, automatically. The relation of the input proposition to existing knowledge is established while the dialog is ongoing. This allows a dialog participant to consider only the current utterance advanced by the other participant, not necessarily the entire argument, to continue processing.

[1]This work was supported by the National Science Foundation, under grants IRI-9701617 and IRI-9523666.

Background

Argument and Argument Relations

During a dialog, whenever a participant challenges an utterance of another participant, we say that an *argument* has started. The propositions that are usually advanced by the participants during the exchange are either an *attack* to a proposition (*i.e.*, by attacking the merits of the opponent's claim), or a *support* to a previously held claim. An argument relation is either an *attack* or a *support*. We use rules, called *argument schema rules*, that characterize the structure of arguments and are used to determine whether an input proposition attacks or supports existing knowledge (or does neither). Thus during the exchange, it is possible to keep track of the relationships between utterances by maintaining a labeled directed acyclic graph whose nodes are the propositions corresponding to the utterances and whose arcs are labeled as either *attack* or *support*. This labeled directed graph is called an *argument graph* [4]. In this paper, we refer to an argument as any node in the argument graph.

Relation to Prior Work

Argumentation has attracted the interest of researchers in philosophy [16], law [10] and artificial intelligence. Inside the field of artificial intelligence, investigations ranged from understanding arguments [1, 4, 11], building interactive systems [15, 17] and to the use of argumentation to deal with issues in default logic and nonmonotonic reasoning [8, 5].

Our current work deals with issues closely related to understanding arguments in an interactive environment, more specifically, in a dialog. Recent work in interactive argumentation systems include IACAS [15] and NAG [17]. IACAS allows the users to start a dispute and find arguments. However, it does not address the issue of maintaining separate belief models for each participant in a dialog. NAG, on the other hand, uses tagged words from the input and uses Bayesian networks to find the best set of nodes that can be formed as an argument for a proposition. Neither system, however, addresses argument detection; they deal with issues different from the ones that concern us. [7] focus on a different, but related, problem within dialog, the collaborative construction of a plan. This work considers the problem of deciding whether the system should accept or reject a user's proposal, on the basis of its consistency with the system's beliefs and the evidence offered by the user. Unlike ARGUER's approach to deciding whether some evidence supports a belief, which relies on domain-independent schemata, their system's approach relies on domain-specific patterns of evidence. Other systems like ABDUL/ILANA [4] and HERMES [9] do not address the issue of how attack and support relations between propositions may be established computationally. Alvarado's OpEd [1] although designed to understand arguments is limited to processing editorial text and does not address the issues we are concerned in dialog processing.

In ARGUER, the relations between propositions are dynamically established while the dialog is taking place. The use of argument schemas allows *incremental* processing of arguments. Since relations between propositions can be established dynamically, the system may not always have to wait until the complete argument is presented. Moreover, the method is *symmetrical* because it can be used for interpretation or generation of arguments. This is important because the system can have the role of observer or participant. Currently, ARGUER does not address the problem of choosing among a set of possible arguments when generating an utterance; this has been the topic of work by [17], for example. We are currently working on a model of preference that will address this concern.

Argument Detection and Rebuttal

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. In ARGUER, argument detection and rebuttal are established via argument schema rules. An argument schema rule characterizes the structure of an argument. The method finds all the schemata into which the deep meaning representation of a proposition fits. If a match is found, the corresponding variables in the argument schema rules are instantiated, thereby establishing attack or support relation. For example, given a proposition α corresponding to `Tweety is a bird`, the schema, NOT α, corresponding to `Tweety is not a bird`, characterizes an argument against α. A simple argument schema rule that allows ARGUER to establish an attack relation between the two propositions is: α *attacks* NOT α.

The sample dialog of Figure 1 will be used throughout the discussion in this paper. The domain knowledge used in the dialog is extracted from the American Heart Association Screener Technician Manual[2], a manual for teaching people how to measure blood pressure and what to say to patients. Two participating agents in the dialog, the system and the user, argue about the need for a blood pressure check. S1, S2 and S3 are the system's utterances while U1 and U2 are those of the user.

```
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.
U2 But I feel healthy.
S3 Unfortunately, there are no signs or
   symptoms that tell whether your blood
   pressure is elevated.
```

Figure 1: Sample blood pressure dialog

We refer to an argument as any proposition corresponding to a node in the argument graph (see Section), e.g. in Figure 1, S1, U1, S2, U2 and S3 are possible arguments. S1 tells the user that he/she must have a blood pressure check. The user responds by telling the system that there is no need (U1). The system then tells the user of the possible consequences if the user does not have a blood pressure check, *i.e.*, uncontrolled high blood pressure can lead to heart attack, heart failure, stroke or kidney failure (S2). Then the user responds that there is nothing wrong with himself/herself: 'But I feel healthy' (U2). The system then responds using S3 that, unfortunately, a person might have a high blood pressure

even if no symptoms are felt.

The Types of Knowledge

ARGUER's knowledge base contains at least three types of information: domain knowledge, common sense knowledge and argumentation knowledge. This information is needed to detect arguments and generate possible responses to them. ARGUER needs domain knowledge to reason about information specific to the topic being talked about. For example, the knowledge associating signs or symptoms with an illness is domain knowledge essential in understanding and responding during dialogs about health. General facts are considered as common-sense knowledge, *e.g.* requiring x from A may imply a need of x by A. In addition, information about the structure of arguments and their various forms is used to detect arguments advanced by agents during a dialog. The argumentation knowledge consists of the argument schema rules.

The Knowledge Representation

ARGUER represents all its knowledge in a uniform framework known as a propositional semantic network. A propositional semantic network is a framework for representing the concepts of a cognitive agent who is capable of using language (hence the term *semantic*). The particular knowledge representation system that is used by ARGUER is SNePS [14] which provide facilities for building and finding nodes as well as for (first- and second-order) reasoning, truth-maintenance, and knowledge partitioning (for user- and system-models). Reasoning in ARGUER is computationally tractable because of the knowledge partitioning; only relevant portions of the knowledge base(s) are used for argumentation.

For brevity and clarity, in this paper, we shall use equivalent logic form representations in our sample interactions.

Models of Belief

ARGUER uses a separate model of belief for each participant in the dialog. In Figure 1, the system plays the role of the medical expert and the user is the patient. The propositions that the system believes may not be necessarily believed by the user. The system, in responding to propositions advanced by the user, takes into account the beliefs currently held in the (system's) user model (as well as its own beliefs). We define a belief model M of an agent A as follows:

Definition 1 (Belief Model)
Let $T = \{\sigma_i \mid \sigma_i$ *is a proposition believed by an agent* $A, i = 1 \ldots n\}$ *and* $T' = \{\sigma_j \mid \sigma_j$ *can be logically derived from* T*, iteratively*$\}$*. The belief model of agent* A*, denoted* M_A*, is* $T \cup T'$*. Furthermore, if a proposition* $\sigma_k \notin M_A$ *then we say that agent* A *is* ***ignorant*** *about* σ_k*.*

According to Definition 1, the belief model of an agent is the set containing all the propositions believed by the agent together with all its logical consequences[2]. If a proposition does not belong to an agent's model, it is not necessarily false. It may just be the case that the agent is ignorant about it.

During the dialog, as the utterance's meaning representation becomes available, general cases (corresponding to argument schema rules) are used to establish the argument relations. In our method, this process is dynamic and no *a priori* knowledge of argument relations between propositions is assumed. For brevity, we consider only two argument schemas in this paper that are necessary for the examples that we discuss. However, our model is extensible and does support other argument schemas.

Argument Schemas

Suppose someone makes a claim. A common way of challenging that claim is by saying 'No, it is not the case that ...'. Let us refer to the first speaker as the *proponent* and the second speaker, *opponent*. The proponent then is obliged to prove or show evidence that would support the claim, i.e. the burden of proof lies with the proponent [16]. In Fragment 1, we intuitively know that U1 is a challenge to S1. By having a general case of argument schema that captures this situation, it is possible to establish computationally that U1 is a challenge to a preceding claim made by the other agent, *i.e.*, S1. The same argument schema could also be used for rebuttal.

Fragment 1

```
S1 Have your blood pressure checked.
U1 There is no need.
```

In Fragment 1, S1 tells the user that there is a need for a blood pressure check. This first utterance makes the *claim* (by implication). The user however responds by challenging the claim and saying, U1, that there is no need. By using an argument schema rule, in this case the rule R1b below, our system can establish that U1 is an *attack* to S1.

[2]This is a simplifying assumption. We are working on a resource-limited model of belief.

Argument Schema 1 (Negation)

Let α and β be propositions, then:

R1a: If α is an utterance then NOT α is an attack to α.

R1b: If α is an utterance implying β, then NOT β is an attack to α.

In Fragment 1, U1 attacks S1. The utterance 'Have your blood pressure checked' implies that 'There is a need for a blood pressure check.' We assume that there exists common-sense and domain knowledge that allows us to derive this knowledge. Rules in the common-sense and domain knowledge allow us to derive that 'requiring x of A' (which follows from the imperative form of S1) implies 'the need of A for x'. The rule R1b in Argument Schema 1 allows us to establish that the utterance U1: There is no need, is a proposition that attacks the utterance S1: Have your blood pressure checked. In this case α is S1 and NOT β is U1.

Conversely, suppose the user said S1 and the system wants to generate an attack. Using either R1a or R1b allows us to generate a response that attacks the preceding claim. If α is S1, then there are two possible attacks to α: (1) NOT α via R1a and (2) NOT β via R1b. The best possible response could then be selected using some preference criteria. The method of selecting the best response is a part of our ongoing work (see Section).

We consider a more complex case in the next fragment of the dialog of Figure 1.

Fragment 2

U1 `There is no need.`
S2 `Uncontrolled high blood pressure can lead to heart attack, heart failure, stroke or kidney failure.`

In Fragment 2, one way to detect that S2 is an attack to U1 is to consider the consequences of U1. Not having a blood pressure check can lead to uncontrolled high blood pressure which can lead to events like heart attack or stroke. These events in turn can lead to a fatal event, *i.e.,* death, which is possibly unacceptable to the user.

There are two different types of unacceptability that we consider here. One type is subjective, *i.e.,* an agent subjectively determines a proposition to be unacceptable (such as the possibility of his/her death). The other type is motivated by reason, *i.e.,* a proposition is unacceptable to the agent because the agent believes its negation is true. For example, the statement `It is unacceptable to me that Tweety can fly` because `I believe that Tweety can not fly` is of the second type. We define the notion of unacceptability as follows:

Definition 2 (Unacceptable)

*Let M_A be a belief model of agent A, δ be a proposition, and C,D be the set of facts and rules of the common-sense and domain knowledge, respectively. Let the symbol, $\vdash$, denote logical derivation. Furthermore let $\mathbf{uc}(\delta)$ be a proposition equivalent to the statement "δ is not acceptable". The proposition δ is **unacceptable** with respect to M_A if and only if (1) $(M_A \cup C \cup D) \vdash NOT\ \delta$, or (2) $(M_A \cup C \cup D) \vdash \mathbf{uc}(\delta)$.*

Definition 2 allows two types of unacceptability. The condition in (1) means we do not accept the proposition because we believe its negation. The condition in (2) means we do not accept the proposition because we simply believe it is subjectively unacceptable. In Fragment 2, α is U1 and δ, the state which is unacceptable to the user, is implied by S2. Any proposition leads to δ from α, attacks α. Our model uses both types of unacceptability[3].

Argument Schema 2

Let α, β, γ and δ be propositions and let the symbol, $\Rightarrow$, denote logical implication, then:

R2: If $\alpha \Rightarrow (\beta \Rightarrow \gamma) \Rightarrow \delta$, where δ is unacceptable to the agent uttering α then $(\beta \Rightarrow \gamma)$ is an attack to the utterance α.

Argument Schema 2 means that any set of rules or propositions that leads to a state that is unacceptable to the speaker is an attack to what has just been uttered. The flow of reasoning is the following: Suppose that there is no need for a blood pressure check. If blood pressure is not checked, blood pressure may become high. High blood pressure can lead to heart attack, heart failure, stroke or kidney failure. Having a heart attack, heart failure, stroke or kidney failure is unacceptable to the agent.

Using R2, the system detects that the proposition 'Uncontrolled high blood pressure can lead to heart attack, heart failure, stroke or kidney failure' is an attack to the utterance 'There is no need'. Moreover, R2 can be used to find possible responses to U1. Possible propositions that attack U1 are (1) 'If a blood pressure is not checked, blood pressure may become high' and (2) 'High blood pressure can lead to heart attack, heart failure, stroke or kidney failure'. These propositions enable the flow of reasoning to reach a state that is unacceptable to the speaker.

A Sample Interaction

In this section, we describe an implementation of the method, used in ARGUER, that detects arguments

[3]We are investigating argumentation issues when there is an inconsistency between the user model and the domain and common sense knowledge.

and also finds possible ways to respond. The problem that we are attempting to address here is a part of a larger project to understand various issues involved in robust human-machine communication [12]. We will use a portion of Figure 1 to show how the proposed method works.

ARGUER's knowledge base consists of common-sense knowledge, domain knowledge and argumentation knowledge. The knowledge base and the utterance of the participating agents in the dialog are represented uniformly. On the implementation level, the uniform representation allows a common method of accessing and processing information of different types, which in this case include processing information from domain knowledge, argumentation knowledge and common-sense knowledge. Advantages of using uniform representation are discussed in [12].

We are currently extending ARGUER to process natural language (using the general-purpose grammar of B2 and a template-based generation system [12, 13]). For clarity, the example(s) below are shown in English, ARGUER assumes the interpreted logic-based representations shown for input (and outputs similar representations).

An Example In the following example, S1, U1 and S2 are utterances of the agents participating in the dialog. S1 and S2 are utterances of the system while U1 is an utterance of the user. We show how ARGUER detects correctly that U1 attacks S1, and S2 attacks U1. Moreover, after each utterance (U1 and S1) we show how ARGUER can use the same method to generate possible rebuttals.

Fragment 3

```
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.
```

Utterance S1 is interpreted as: agent 'system' requiring agent 'user' to have a blood pressure check. (We assume that U1 has been modified to the form 'There is no need for a blood pressure check'.) We represent the utterance S1 as:

$$\text{S1:} \quad \textit{Require(system, user, *check-BP)}$$

The variable **check-BP* represents an unknown agent performing a blood pressure check on the agent *user*. The utterance U1 is true in the user's model, *i.e.*, as far as the user is concerned, the user does not need a blood pressure check. This is represented as:

$$\text{U1:} \quad \neg \textit{Need(user, *check-BP)}$$

Since models for the user and the system are separately maintained, this proposition is not believed by the system, but can be used to detect an argument.

Detecting an argument: U1 attacks S1 To detect that U1 attacks S1, ARGUER asks the question

$$\textit{Attacks(Require(system, user, *check-BP),} \\ \neg \textit{Need(user, *check-BP))}$$

(*i.e.*, is U1 an attack on S1?) and uses two rules to answer:

1. Rule R1b: If α is an utterance implying β, then NOT β is an attack to α.

$$\text{R1b:} \quad \forall \alpha, \beta \; \textit{Derived}(\alpha, \beta) \rightarrow \textit{Attacks}(\alpha, \neg\beta)$$

2. a common-sense rule, C1, that states that if the system requires some act of an user, then the user needs the required act.

$$\text{C1:} \; \forall u, a \; \textit{Require(system, u, a)} \rightarrow \textit{Need(u, a)}$$

When rule C1 is instantiated and used (with S1), ARGUER deduces the consequent

$$\textit{Need(user, *check-BP)}$$

and a meta-level proposition that represents the reasoning path of a ground rule from the antecedent to the consequent:

$$\textit{Derived(Require(system, user, *check-BP),} \\ \textit{Need(user, *check-BP))}$$

This meta-level proposition is then used in argument schema R1b (with α = *Require(system, user, *check-BP)* and β = *Need(user, *check-BP)*) to conclude that the user is attacking the system's utterance:

$$\textit{Attacks(Require(system, user, *check-BP),} \\ \neg \textit{Need(user, *check-BP))}$$

Generating an argument U1 that attacks S1 To generate an argument that attacks S1 (possibly to preempt a user argument), ARGUER might ask the question

$$\textit{Attacks(Require(system, user, *check-BP),} \; *X)$$

(where $*X$ is a free unbound variable) and deduces possible attacking propositions to S1. As before, this uses rules R1b and C1 to find possible values for $*X$. In this example, it once again finds:

$$\textit{Attacks(Require(system, user, *check-BP),} \\ \neg \textit{Need(user, *check-BP))}$$

We note that this method can find *all* arguments that attack S1 (*i.e.*, not just U1), dynamically and with the currently available information and state of the user model.

Generating a rebuttal: S2 attacks U1 To generate a rebuttal to U1, ARGUER asks the question:

*Attacks(¬Need(user, *check-BP), *X)*

namely, what attacks U1. In this example, these rules are used:

1. R2: $\forall\ \alpha, \beta, \gamma, \delta$ (*Derived(*α, β*)* $\wedge$ *Derived(*β, γ*)* $\wedge$ *Derived(*γ, δ*)* $\wedge$ *uc(*δ*))* $\rightarrow$ *Attacks(*α, *Derived(*β, γ*))*

2. C2: $\forall a, x\ \neg Need(a, x) \rightarrow \neg Do(a, x)$

3. D1: $\forall a$ $\neg$*Do*(a, **check-BP*) $\rightarrow$ *Possible(attribute(a, BP, high))*

4. D2: $\forall$ a *Possible(attribute(a, BP, high))* $\rightarrow$ *Possible(event(a, {stroke, heart-failure, heart-attack, kidney-failure}))*

5. D3: *uc(Possible(event(a, {stroke, heart-failure, heart-attack, kidney-failure})))*

Note that the simplest case of R2 is when $\alpha = \beta$ and $\gamma = \delta$ (the user's utterance immediately leads to a consequent that is unacceptable). C2 is a common sense rule that states that if a agent does not need to do an act, the agent will not do it. D1 is a domain rule that states that if an agent does not have their blood pressure checked then high blood pressure is possible. D2 is a domain rule that states that an agents having possible high blood pressure can lead to a possible stroke, heart attack or failure, or kidney failure. D3 is a domain fact that states that these outcomes are unacceptable to the user.

To answer its query, ARGUER reasons as follows:

- *Derived(¬Need(user, *check-BP), ¬Do(a, *check-BP))* From C2 and U1.
- *Derived(¬Do(user,*check-BP), Possible(attribute(user, BP, high)))* From D1 and C2.
- *Derived(Possible(attribute(user, BP, high)), Possible(event(user, {stroke, heart-failure, heart-attack, kidney-failure})))* From D2, D1 and C2.

Using R2 and D3 with this final derivation, ARGUER can conclude:

*Attacks(¬Need(user, *check-BP),*
Derived(Possible(attribute(user, BP, high)),
Possible(event(user,{stroke, heart-failure,
heart-attack, kidney-failure}))))

From this, ARGUER decides that S2 is an attack to U1:

S2: *Derived(Possible(attribute(user, BP, high)),*
Possible(event(user,{stroke,heart-failure,
heart-attack, kidney-failure})))

Moreover, because the method is symmetrical, it could be used to detect that S2 attacks U1, to predict possible rebuttals to S2, or to answer the question "What attacks what?"

Summary

We have shown the use of argument schema both for detecting arguments and generating possible rebuttals in dialogs. The method is implemented as ARGUER and allows us to establish argument relations between propositions while the dialog is ongoing. Moreover, the method we have proposed here is *incremental* in that it allows processing of each piece of the utterance and selects only a part of the argument to continue.

References

[1] S. Alvarado. *Understanding Editorial Text: A Computer Model of Argument Comprehension.* Kluwer Academic, 1990.

[2] American Heart Association, Milwaukee, WI. *Blood Pressure Measurement Education Program Screener Technician Manual*, 1998.

[3] K. D. Ashley. *Modeling Legal Argument: Reasoning with Cases and Hypotheticals.* MIT Press, Cambridge, MA, 1990.

[4] L. Birnbaum, M. Flowers, and R. McGuire. Towards an AI Model of Argumentation. In *Proceedings of the AAAI-80*, pages 313–315, Stanford, CA, 1980.

[5] A. Bondarenko, P. M. Dung, R. A. Kowalski, and F. Toni. An Abstract, Argumentation-Theoretic Approach to Default Reasoning. *Artificial Intelligence*, 93:63–101, 1997.

[6] M. Bordegoni, G. Faconti, T. Y. Maybury, T. Rist, S. Ruggieri, P. Trahanias, and M. Wilson. A Standard Reference Model for Intelligent Multimedia Representation Systems. *The International Journal on the Development and Applications of Standards for Computers, Data Communications and Interfaces*, 1997.

[7] Jennifer Chu-Caroll and Sandra Carberry. Generating Information-Sharing Subdialogues in Expert-User Consultation. In *Proceedings of the 14th IJCAI*, pages 1243–1250, 1995.

[8] P. M. Dung. The Acceptability of Arguments and its Fundamental Role in Non-Monotonic Reasoning, Logic Programming and N-Person Games. *Artificial Intelligence*, pages 321–357, 1995.

[9] N. Karacapilidis and D. Papadias. Hermes: Supporting Argumentative Discourse in Multi-Agent Decision Making. In *Proceedings of the AAAI-98*, pages 827–832, Madison, WI 1998.

[10] R. P. Loui and J. Norman. Rationales and Argument Moves. *Artificial Intelligence and Law*, 1993.

[11] R. McGuire, L. Birnbaum, and M. Flowers. Opportunistic Processing in Arguments. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 58–60, Vancouver, B.C., 1981.

[12] Susan McRoy, Syed S. Ali, and Susan Haller. Uniform Knowledge Representation for NLP in the B2 System. *Journal of Natural Language Engineering*, 3(2):123–145, 1997.

[13] Susan W. McRoy, Syed S. Ali, and Susan M. Haller. Mixed Depth Representations for Dialog Processing. In *Proceedings of the 20th Annual Conference of the Cognitive Science Society (CogSci '98)*, pages 687–692. Lawrence Erlbaum Associates, 1998.

[14] Stuart C. Shapiro and William J. Rapaport. The SNePS family. *Computers & Mathematics with Applications*, 23(2–5), 1992.

[15] G. Vreeswijk. IACAS: An Implementation of Chisholm's Principles of Knowledge. In *Proceedings of the 2nd Dutch/German Workshop on Nonmonotonic Reasoning*, pages 225–234, 1995.

[16] D. Walton. *Argument Structure: A Pragmatic Theory*. University of Toronto Press, 1996.

[17] I. Zukerman, R. McConachy, and K. Korb. Bayesian Reasoning in an Abductive Mechanism for Argument Generation and Analysis. In *Proceedings of the AAAI-98*, pages 833–838, Madison, Wisconsin, July 1998.