Persuasive Argumentation in a Medical Diagnosis Tutoring System

Amin Rahati and Froduald Kabanza
Department of Computer Science
University of Sherbrooke
Quebec, Canada J1K 2R1
amin.rahati@usherbrooke.ca,kabanza@usherbrooke.ca

Abstract

An important problem in intelligent tutoring systems and decision support systems in general concern the implementation of a convincing argumentation dialog between the system and the student or the user. The requirements for a convincing argumentation dialog can vary depending on the kinds of conflicts that arise in the exchanges between the system and the user. In this paper we discuss an approach computing and presenting persuasive arguments in an intelligent tutoring system for medical diagnosis. Although the focus of our application is on a simulator of medical diagnosis problems intended to provide exercises to medical students, the argumentation framework we propose could also inspire other types of decision support systems.

1 Introduction

In this paper we are concerned by the general problem of implementing verbal exchanges (*i.e.*, a dialog) between a user and a decision support system (DSS) intended to help her perform a cognitively complex task. A particular case of this is a simulator for medical diagnosis which simulates patients with particular health pathologies and support students who are asked to diagnose a selected patient by providing them with tutoring feedback. In such a system, a dialog must be implemented that determines when and how guiding help is provided to the student, that is, what to say to her, in what circumstances, and how to say it. A dialogue like this will be more effective if its makes it possible for the student to challenge the feedback given by the system [Yuan *et al.*, 2008].

A number of medical diagnosis tutoring systems have been proposed over the years, going from GUIDON in the late 1980 [Clancey, 1987] to SlideTutor [Crowley *et al.*, 2003], COMET [Suebnukarn and Haddawy, 2005] and TeachMed [Kabanza *et al.*, 2006] in more recent years. The interactions between these systems and the student can include tutoring hints and other useful forms of feedback that help the student realize her errors in her diagnosis approach and recover from them. However, these interactions have no built-in capability for providing convincing arguments to the student.

The capability to convince users is an important element in a dialog [Zukerman *et al.*, 1998; Restificar *et al.*, 1999], particularly when the dialog is an argumentation between the system and the user, that is, when the different utterances composing are moves for asserting, accepting or withdrawing an argument, or are questions or challenges to the opponent's argument [Walton and Krabbe, 1995; Reed, 1998]. The motivation of the work presented in this paper is to integrate an argumentation framework into a medical diagnosis system. Here we consider the TeachMed medical diagnosis tutoring system, but the approach we propose could also fit the other approaches. More specifically, we discuss an approach for determining how a system computes and presents a convincing argument to a user when there is a disagreement or conflict between them. In our case, this means a disagreement between TeachMed and the student.

Convincing and persuasive arguments can be computed through a modelling of user beliefs in a probabilistic or logical belief revision [Zukerman *et al.*, 1998; Restificar *et al.*, 1999]. Approaches of this kind have been so far limited to situations where the disagreement (conflict) between the parties (i.e., the student and the system in our case) results in a lack of knowledge about some facts or a flawed chain of reasoning from one of the parties. Yet we know that a conflict can also arise between the parties only because they disagree on what is more important in a given situation [Perelman and Olbrechts-Tyteca, 1969]. In particular two decision makers may agree on the values of decision variables at a given point in a decision making process (*e.g.*, they agree on the evidence variables in a patient diagnosis) but disagree on the actions that should be taken (e.g., they disagree on the hypotheses for a disease given the same evidences). In this kind of conflicting situations a persuasive argument would advocate an action that promotes the most important parameters. The capability of computing such arguments is very limited in existing approaches [Chesñevar *et al.,* 2006].

We use the concept of dialog game [Pilkington *et al.*, 1992; Maudet and Moore, 2001; Karunatillake *et al.*, 2009] to model the interactions between TeachMed and the student. Given that persuasive arguments used during verbal exchanges are often grounded in the trace of the diagnosis process performed by the user, we view the whole interaction between TeachMed and the student as arguments. Thus every action performed by the student is an argument even if the action is not actually an utterance intended for TeachMed but simply a step in the diagnosis process (*e.g.*, linking an evidence to a hypothesis). This idea of seeing the entire interaction between a system and a user (not just verbal utterances) as an argumentation is not unique. It is used for example in REACT, a DSS for medical care planning [Glasspool *et al.*, 2003].

The dialog game models the exchange aspect of the argumentation, that is, the taking of turns between the TeachMed and the student during their interactions. That way moves can be understood as transitions in the dialog game. In addition we need formalism for representing problem solving actions as arguments such that the system can reason about them to determine which argument can challenge them. So far researchers have used argument templates for this purpose [Glasspool *et al.*, 2003; Zeleznikow and Stranieri, 1995; Bench-capon, 1998]. Here we use a specific template proposed by Walton [Walton, 1996a] for modelling actions. The advantage of this template is that it allows the explicit representation of challenges to arguments through questions that are associated to templates. Test questions evaluate defeasibility of any instantiated argument of the template. We explain later how test questions can be used to specify the activation of counterarguments which can challenge the instantiated argument of the template.

Given a set of arguments including argument instantiated from user actions in a problem solving process as well as counter arguments generated by test questions, we need a mechanism to calculate the applicable (or persuasive) argument during a discussion with the user. Here we define an applicable argument as a counterargument against a user's argument which succeeds in defeating it (if any exists) and defending itself against possible challenging arguments. Using different definitions of defeat and attack among a set of arguments, different argumentation frameworks have been introduced so far for computing a subset of persuasive arguments. In our work preference among arguments is determined according to the value they promote and we compute persuasive arguments using a value-based argumentation framework introduced by [Bench-Capon, 2003].

Once a persuasive argument has been computed, the next step for the system is to decide how to represent the elements of the argument (*i.e.*, its premises and conclusion) in a way that can settle the conflict effectively. According to Walton [Walton, 1996b] a conflict is resolved when one participant commits to a proposition which indicates the position of another participant in an argumentation. To do so, we use some heuristic rules to engage the user in an argumentation dialog with the goal of making her to become committed to the conclusion of a persuasive argument if any exists.

The remainder of this paper is organized as follows. In the next section we start by introducing TeachMed. Then in section 3 we present the dialog game. After that we explain how the student's actions in a problem solving process are represented as arguments and how convincing arguments are calculated as responses to student actions. Then we present heuristic rules which are used to engage the student in a dialog to resolve conflicts using automatically computed persuasive arguments. Thereafter we provide an illustrative example of application of our approach integrated in TeachMed. Finally we conclude the paper by discussing about related research and future works.

2 Argumentative TeachMed

An intelligent tutoring system (ITS) is actually one particular kind of DSS, where the user being supported is a student and the provided support has to be pedagogical. We have chosen TeachMed, an ITS for teaching medical diagnosis [Kabanza *et al.*, 2006], as a testbed for our approach. TeachMed presents a simulated patient to a student, with some initial symptoms and background information. The learning task for the student is to investigate the health problem of the patient by gathering additional evidences through evidence-gathering actions (interrogations of the patient about his symptoms and background information such as family history or eating habits; lab tests; and physical exam) and by formulating hypotheses that explain the evidence collected so far. Queries and tests are selected from a list including noise queries. Each query has a pre-determined answer, specified in the virtual-patient model, which includes his vital signs, symptoms and results of lab tests or physical exam.

At the beginning, the student does not know what causes the pain. The learning objective for the student is to discover what most likely causes the pain. Initially, the student will formulate some hypotheses just after a few queries on the vital signs. As she gathers more evidence, she will eliminate some hypotheses, strengthen others and generate new ones. This process continues until she can narrow the list of hypothesis to one or two, that is, the final diagnosis.

The architecture of argumentative TeachMed has the following components: a user model which indicates evidence-gathering, hypothesis-formulation actions and utterances made by the user (in the form of commitments and history of played moves which are concepts that are explained later) and also a list of user's preference values in a problem solving task; an expert medical knowledge under the form of an influence diagram specifying the causal hypotheses-evidences relationships for the health case being solved and the utility of evidence-gathering actions; a pedagogic model, which consists of two modules, one for analyzing the user's arguments which determine when the ITS intervenes and another for presenting the calculated persuasive arguments that specify how the ITS intervenes and discusses with the user to help her.

In TeachMed the actions performed by the student are either diagnosis actions or requests questions for help. Diagnosis actions are in the form of a query for evidence gathering (asking a direct question to the patient about his symptoms, look up into vital signs, physical exam on a 3D model of the patient or lab tests), evidence recording in a table based upon answers by the patient simulator to evidence gathering queries, hypothesis recording in a table, and making links between evidences and hypotheses [Kabanza *et al.*, 2006]. We consider each user action as a potential source of conflict or disagreement with TeachMed which TeachMed tries to settle through an argumentation. More precisely, each diagnosis action made by the student is automatically asserted as an argu-

Figure 1: Medical Diagnostic Scenario

ment and TeachMed attempts to reject it if it can. On the other hand, each request for help made by the student is considered as argument and TeachMed tries to provide an explanation in favour or against it. In either case, the argumentation dialog is conducted with the goal of settling the conflict.

For instance, figure 1 is a modified scenario of a medical diagnostic scenario adapted from [Kabanza *et al.*, 2006]. The introduced modifications illustrate the argumentative capability that we are aiming for with the approach presented herein. In this scenario, the student is in the process of diagnosing a simulated patient and she has so far gathered a number of evidences and has formulated a list of hypotheses (*i.e.*, a differential diagnosis). After line 4 of this scenario TeachMed notices inconsistencies between the evidences and the hypotheses formulated by the student. At this point the system starts arguing with the student (line 5 to 15). The role of the system then should be to generate arguments intended to convince or persuade the student of her mistake. To avoid of complexities such as speech recognition in handling of the user's input, all the user's entries are done through some provided menus.

The overall interaction from the start of a diagnosis task by a student to the completion of the task is considered as a sequence of argumentation dialogs (*e.g.*, scenario of figure 1 constitutes of 5 argumentation dialogs). Each argumentation dialog is started by a problem solving action performed by the student (*e.g.*, actions performed in line 1 to 4) or her asking for help. In the former case the dialog ends when either the system accepts that action (for example by its silence and non-intervention as is the case with the reactions to the student's actions performed in line 1 to 3) or the student is convinced to revise her action. In the later case the dialog ends when the student indicates that she is convinced by accepting the system response (for instance in line 15 the student has been convinced to correct her action done in line 4) or the system cannot find more arguments.

If an argumentation dialog is to resolve the conflict between some participants then it should not go endlessly and

Figure 2: A dialog game

participants should be restricted on the argument they can put forward. Thus argumentation dialog should be regulated between participants according some protocol. In computational dialectic such protocol are known as *dialog games*. In the following section we elaborate our dialog game.

3 Dialog Game

A dialog game can be determined by defining available move types and the dialog state transition diagram (DSTD) [Bench-capon, 1998]. A move is described as a tuple $< T, P, S >$, where T indicates the move type, P the player (student or TeachMed), and S the propositional content of the argument. The following five types of moves are allowed in our dialog game: *assert*, *accept*, *withdraw*, *challenge* and *question*. Figure 3 outlines the complete description of these move types, rule or precondition which determines when a move is available and the effect of a played move on the dialog state. Preconditions and effects are defined based on concepts that are introduced later in this section.

The DSTD of a dialog game specifies the sequences of moves are allowed in argumentation dialog. We specify the whole DSTD as a recursion over DSTDs, such that we can have transitions between sub-DSTDs much like in a state chart [Harel, 1987; Drusinsky, 2006]. Figure 2 illustrates a DSTD with three possibly entry points (labeled 1). The topmost entry point is an assert dialog (AD) and it is meant for beginning an argumentation in a situation where the conflict is not caused by a question nor a challenge made by the opponent. The second entry point is a challenge dialog (CD) and it is to initiate a dialog for a challenge to an argument made by the opponent (CD). The last one is for a question dialog (QD) for initiating a dialog as question to the opponent.

A DSTD is run by a process called the *debate manager*, whose role it to keep track of the current control point of the dialog when a transition is made to a sub-dialog, using a stack much like a programming runtime environment keeps tracks of the recursive invocation of functions. The debate manager also manages the turn taking of the players giving control to TeachMed or the student whenever their respective move is

Figure 3: Move Types

the next. The debate manager keeps track of the history of moves made by players in a dialog-history list (DH). This is a path in the dialog tree.

Thus according to this example, the first entry point specifies that an opponent has three options to reply to an argument asserted by an proponent. She could accept the argument, and then the dialog terminates. She could also question (branch to QD) or challenge it by assert a counterargument against it (branch to CD). The remaining entry points are interpreted similarly. Note that in the case of a challenge, the dialog ends when the proponent withdraws the asserted argument. If the proponent was a student for example, this would mean she was convinced by the arguments put forward by TeachMed and hence withdraw her initial challenge. Before the argument is withdrawn there may have been some interaction of arguments and counter-arguments both ways (via the transition to AD). Once the dialogs ends, control is given back to the debate manager.

TeachMed controls its own moves, but it cannot control moves made by the student. The objective of a tutoring strategy is for TeachMed to decide upon its own moves such that to choose arguments which best help the students. The appropriate move will depend on the current situation and the type of the argument that initiated an argumentation, that is, the entry point in the dialog game. For instance if a student initiated an argumentation by challenging a tutoring hint provided by TeachMed, it will proceed by selecting arguments, when it is its turn, that are expected to end the dialog in a state where the student withdraws her argument, assuming that the student will persist arguing until he becomes convinced. [1]

We define a *commitment* as an argument asserted or accepted by a player [Mackenzie, 1979]. We also define a *debate state* as a tuple consisting of the dialog history, the current control point (*i.e.*, state) in the dialog game, and the list of commitments so far. All commitments are stored in a list called the *commitment store* (CS). An element in the list is a pair (P, S), meaning that player P has committed to the argument S. Initially CS is empty.

We assume that players are not limited to committing to only what they believe or to believe what they commit to. This simplification assumption not only helps us avoiding the use of a belief update framework. As argued by [Walton, 1996b], the commitment concept also provides a means to settle conflicts by making the opponent of a proposition become committed to it or the proponent of a proposition withdraw his commitment.

A dialog game only prescribes what types of moves are available for players at different point of an argumentation dialog. Since one of the participants is a system then we need to implement a mechanism for the system to be able to automatically select a move type among available move types and to compute content which is persuading to the user. Here we adopt a move selection strategy introduced by [Moore, 1993] and later advocated by others [Amgoud and Maudet, 2002; Yuan *et al.*, 2008].

4 Move Selection Strategy

The strategy for selecting a move at different points of an argumentation dialog depends on many parameters including the profiles and goals of participants, and available resources [Amgoud and Maudet, 2002]. For example, a strategy of a participant with an argumentative profile is to challenge others whenever possible [Amgoud and Parsons, 2001]. In education, the goal of a tutor is to help the student to solve a problem on his own as much as possible. Accordingly, a Socratic strategy may help students asking him questions that are expected to make him realize the right steps rather than providing him with direct explanations. On the other hand, if time is a critical resource then explanation is a better strategy than questioning.

Based on an analysis of natural critical discussions, Moore proposes three levels of strategy selection:

1. maintain or alter focus of discussion;
2. building own point of view or defeating the user's view;
3. adopting method to achieve the objective set at level 1 and 2.

Amgoud and Maudet [Amgoud and Maudet, 2002] argues that level 1 is appropriate for complex protocols which take into account concepts like relevance [Prakken, 2001]. Following a similar approach, we replace level 1 with the profile of participants in an argumentation. Defining the profile of participants affects the other levels in the move selec-

[1]We assume that if the student becomes frustrated with a length but helpless argumentation by TeachMed she will not withdraw her own argument to please TeachMed. She would abort the argumentation instead, an event not explicitly modeled in the previous example.

tion. Level 2 is implemented by making the user to accept the proposition which shows the system's point of view (i.e., building own point of view) or making her withdraw a proposition which indicates her point of view. It is also possible to have a strategy that combines both level 1 and level 2, for instance by building the system's point of view first, and then have the user withdraw her propositions which are against the system's point of view. Contrary to the two first levels which include some strategic goals, the third level refers to domain dependent tactics for achieving strategic goals.

Thus we use strategic goals to compute the content of a persuasive argument based on the student profile (levels 1 and 2). Then given the computed persuasive argument and a strategic goal, we use specific rules to decide how to represent the elements of this argument (level 3). For example in figure 1, at line 10, using these rules the system has started an inquiry to make the student to make commitments to reveal her point of view in the patient diagnosis process. At line 12 the system has asserted an argument to destroy the student's point of view. At line 14 the system has asserted another argument to build its own point of view on the next correct action for the patient diagnosis.

5 Representing Knowledge About Arguments

TeachMed needs formalism for representing problem solving actions as arguments and to reason on them in order to find what challenge arguments against them in different situations. For this, we adopt the formalism of argument templates [Glasspool *et al.*, 2003; Zeleznikow and Stranieri, 1995; Bench-capon, 1998; Walton, 1996a]. More specifically, we adopt the argument templates proposed by Walton [Walton, 1996a] , which are in fact general inference rules in the form of a pair (P, c) where from some premises P a conclusion c can be derived. The template modeling an action in a problem solving task has two premises, one about the goal g at the current problem solving state and another about the action a which is supposed to achieve the goal. The conclusion is performing the action a to achieve the goal g. Given a goal in the current circumstance (recognized for instance by using plan recognition techniques or by explicitly querying the user) this template is used to make instantiations of arguments from actions and utterances (that are questions, explanations or propositions for taking actions in task performing).

Utterances in an argumentation are not always action. They can also be either questions or explanations. In this case the arguments are represented as pairs $(-, c)$ where c is a proposition about domain knowledge.

The key point in Walton's templates is the definition of the concept of defeasibility of arguments instantiated from templates. This definition is articulated around a set of test questions (TQ) associated to each argument template. That way, an instantiated argument from a template is defeasible if it cannot stand against a posed TQ. The TQs are specified as rules stating the activation of arguments that can challenge a user action argument in problem solving process. The different elements composing a TQ are as follows:

Argument shows the challenged argument.

Figure 4: Implementation of some rules of TQs

Condition contains conditions (or premises) of the counterargument.

Conclusion shows the conclusion of the counterargument.

Variables contains the name of variables used in the condition and conclusion.

Data stores the actual data of variables.

Value contains the value that the counterargument promotes.

Figure 4 illustrates two TQ that we have implemented to verify user action arguments for the scenario in Figure 1. Rule 1 verifies arguments which express the relation between a gathered evidence and an hypothesis (*e.g.*, arguments instantiated from the student's actions of line 1 to 4). Rule 2 verifies arguments which are about changing current working hypothesis to another hypothesis (*e.g.*, argument instantiated the from student's action of line 9).

We model both types of conflicts mentioned so far in terms of disagreement on the values that are involved in the decision making process. In other words, correctness of an argument with respect to facts from the domain knowledge or chain of reasoning that causes conflict of the first type are considered as promoting some value which are involved in diagnosis process.

Let's note V the important values (*e.g.* determined by the user's profile) and their preferences in decision making for task performing. Also let's note val a function that assigns a value of set V to a given argument α instantiated from Walton's template or TQ rule. We have defined an analyzer module (AM) that, given this function and th TQ rules, verifies the defeasibility of a given argument α by producing counterarguments which may probably promote some value that is preferred to the value of α.

To generate counterarguments, AM uses TQ rules and a knowledge base which records the current problem solving state, the domain knowledge, and the user model. AM find a rule whose precondition is enabled by the knowledge base. If so, the rule fires as long as a new match for its premises can be found. Any fired rule promotes a predefined value associated to that rule and is considered as a counterargument against α. A counterargument is also represented by a pair (P, c) where the propositions composing the precondition of the fired rule constitute the premises P and the propositions in the conclusion represent c which is about the necessity of taking or not taking a specific action to achieve a specific goal. These counterarguments are candidates of convincing arguments.

The first cycle of applying rules in the AM starts when an argument is instantiated from the user new action in a problem solving process and finishes when all rules have been tested on this argument. Then the next cycle begins when the arguments generated in previous cycle are evaluated against rules. The running cycles of application of rules of AM continues until no more rules can fire.

To complete the description of calculating convincing arguments, we must provide AM with an algorithm to compute what arguments in the current problem solving state are convincing according to the values in V. Since v is determined by the user's profile this means the user profile will influence the process of specifying a convincing argument.

To compute persuasive argument we have adopted a value-based argumentation framework introduced by [Bench-Capon, 2003] which is an extension to the well known argumentation framework of Dung [Dung., 1995]. The aim of Dung's argumentation framework is to find acceptable (or undefeasible) arguments among a set of arguments AR when a clear definition of *attack* relation among them exists. According to [Dung., 1995], an argument $A \in AR$ is acceptable with respect to a set of arguments S iff for any attacker $B \in AR$ then B itself can be attacked by an argument in S . Then it calculates acceptable arguments of AR by determining the maximal (with respect to set inclusion) subset S of AR such that no two arguments in S attack each other, and all argument of S are acceptable to S.

Bench-Capon has changed the previous definition of acceptable arguments by taking into account the disagreement among different audiences on the preference among values that arguments promote [Bench-Capon, 2003]. Given a set of values V he defines a set of audiences sa where each $a \in sa$ represents a specific strict order of preferences $>_a$ on the values of V and also a function val which assigns a value of V to a given argument. Then given a specific audience a he defines an attack of argument $B \in AR$ on argument $A \in AR$ successful if $val(A) \not>_a val(B)$. Therefore, for audience a argument $A \in AR$ which is attacked by argument $B \in AR$ may be still acceptable with respect to set S iff the value that audience a gives to A is more than the value which it gives to B, no matter if B is attacked by any other argument in S.

As argued in [Bench-Capon, 2003] one preferred extension can be computed efficiently when no cycle exist. A cycle exists when two arguments A and B exist which promote the same values and attack each other. As a rational to resolve such cycles we assume that an argument is more persuasive for the user if it has bigger percentage of premises which the user has already committed to. And if the user has not committed to any of premises of two arguments then a random one is preferred since both have the same measure of persuasiveness on the user.

The convincing argument which is selected should be among the counterarguments which attack the user argument. In other words, from the counterarguments generated at the first cycle. One problem is to select among the counterarguments of cycle one if more than one emerge as acceptable argument. One solution is to prefer a counterargument which promotes the most important value. To do so, we assume that counterarguments of cycle one attack each other as well. As such, when preferred extension is calculated it can have only one counterargument which attacks user's argument. In the next section we explain system's decision making regarding how to represent the calculated convincing argument in the form of an argumentation dialog.

6 Presentation Module

The presentation module (PM) achieves the system's goal of argumentation by deciding how to represent a calculated convincing argument when it is the system's turn to say something. This is where the tactics of third level are taken into account. The PM lets the user win the argumentation as long as the AM cannot find a persuasive argument which can defeat the user argument. As such, PM will play a move of type *accept* to show the system has accepted the user argument (by remaining silence and letting the user continue). Otherwise, to settle the conflict with the user, following Walton's idea of conflict resolution [Walton, 1996b], the PM attempts to make the user commit to the conclusion of the calculated persuasive argument. To do so, the PM uses a stack named *argue agenda* (AG) and some *heuristic rules* (HR) which implement tactics in discussion with the user. As mentioned so far, tactics are domain dependent so HRs have been designed in a way to model a tutor which uses Socratic strategy. This means that, regarding the strategic goals of level 2, the system first tries to builds its own view on the conclusion which it desires essentially the user to become committed to it through questioning. But if the user gives the wrong answer to a question it may switch to destroy strategy to demolish the user's view first and thereafter to build its own view.

PM runs as long as AG is not empty and an HR can be applied. Each entry of AG presents a goal of the system for argumentation. Except the first entry others represent sub-goals of some goals in AG. A goal g is an argument where $(user, g) \notin CS$ and is realized if $(user, g) \in CS$ and $(user, \neg g) \notin CS$. When a goal is achieved it is poped from AG.

Initially, PM pushes the conclusion of the calculated convincing argument on AG as the first entry. Then using five HRs which implement a Socratic strategy, PM attempts to make the user to commit to the premises of this goal by questioning on those premises. If so then using the created rational force of these commitments PM persuades the user to accept the goal. This way the PM will let the system win by having the argumentation dialog ends in a dialog state where the user is able to recover from her mistake, or a state in which the user expresses satisfaction with respect to her question that originally triggered the dialog. In the following we elaborate HRs.

HR1 covers situations where the last move has been a question by the user. In this case the topmost entry on AG is the conclusion of calculated argument by AM as response. Therefore, if according to DH the system has not asserted the argument of the goal of the most top entry on AG then the system has to assert it (HR1.a). Otherwise, it means she does not know the premises that support the conclusion (for example, when the user repeats a question to state that she has not satisfied by the given answer). So if such premises exist which the user has not already committed to them then they are pushed on AG (HR1.b).

HR2 is applied when the last move has been the user acceptance of an argument asserted by the system or a withdrawn of an argument she has already asserted (in our approach the user acceptance of a counterargument put forwarded by the system in response to the student's argument is considered as a withdraw, *e.g.*, in line 13 the student has withdrawn of an argument she has asserted in line 11). In both cases, if the accepted argument is a sub-goal of an argument A which the system has already questioned and the student has given the wrong answer and the student has committed to all premises of A then the next move is to assert the argument A which is the most top entry of AG (HR2.a) because now the student have enough support to accept A. Otherwise, if AG is not empty then the system next move is to question about the goal of the most top entry of AG (HR2.b). In other words, the system questions a premise of A which the student has not yet committed to. But if AG is empty the argumentation dialog has finished and the system has won(HR2.c).

HR3 determines the system's next move where the last move has been an assertion by the user. If this assertion is a response to the system's question and the answer is correct then the system next move is to accept the user's argument (HR3.a). If it is not correct then to help the user to know the correct answer the system has to assert the correct response (HR3.b). But if the system's last move has been an assertion or a challenge and student's current assertion is a counterargument against it and AM has not been called so far to find a persuasive argument which can defeat it then it is called and since current focus of discussion has changed so all previous entries of AG are poped (HR3.c). If AM succeeds, and the user has not committed to the conclusion of the new computed persuasive argument then it is pushed on AG (HR3.d) and PM task is to achieve this goal. If AM succeeds but the user has already committed to the conclusion of the new persuasive argument then PM challenges the user for her inconsistency on these two commitments (HR3.e). If AM does not succeed, then the user is winner so the system has to accept this assertion and all the entries of AG are poped and discussion ends (HR3.f).

HR4 has been defined for two purposes. First to allow the system to take control of dialog flow and to lead argumentation after it accepts an argument A made by the user in response to its question. Second when the last move has been an assertion by the user which for that AM has been called and found a convincing argument with a conclusion pushed on AG as a goal. For the former case, if the accepted argument A attacks argument C or is support of an argument B which attacks argument C and the user has committed to C then the system's next move is to challenge the user for this inconsistency in commitment (HR4.a). Otherwise in both cases, if the goal on top of AG has some premises which according to DH the system has not posed questions of them and the user has not already committed to them then they are pushed on AG as sub-goals of new entries with the aim of making the user to commit to them(HR4.b). Otherwise the PM selects the top most goal of AG and asserts it if the user has already committed to all premises of the goal (HR4.c). Otherwise, PM initiates a sub-dialog by questioning the user about the goal of the most top entry on AG (HR4.d).

Figure 5: Generated arguments by rules of AM

7 Example

As mentioned so far, as a possible application, here we explain application of our approach on TeachMed, an ITS for teaching medical diagnosis [Kabanza *et al.*, 2006]. Assume that at the start of a tutoring session the user's level of skill and knowledge either is selected by herself from a menu (to give more freedom to the user) or is retrieved from a user profile. Then a set of pedagogic and quality values which are appropriate to evaluate the user's actions according to the specified level is recommended to her. The user is free to exclude any of these values. Suppose the user has selected five values $V = \{v_1, v_2, v_3, v_4, v_5\}$ with partial order $v_1 > v_2 > v_3 > v_4 > v_5$. Value v_1 states evidence gathering actions must be consistent with medical knowledge and the available patient information; value v_2 expresses that evidence gathering actions have (monetary or time) cost or are intrusive so those which have minimum cost and are less uncomfortable should be made first; value v_3 stipulates that working on a hypothesis should be continues as long as an evidence exists which is related to it; value v_4 expresses that diseases which are threats to the patient's life should be investigated first; value v_5 states that the most probable hypothesis should be selected as current working hypothesis.

It is in the TQs of AM that an argument made by the user is evaluated with respect to these values. To do so, as mentioned before, TQs are implemented in the form of rules which verify user's argument according to the values. These rules use information available by CS, DH and influence diagram for this purpose. Rule 1 of figure 4 represents an instance of verification of value v_1 in medical diagnosis. A condition in the precondition of this rule tests the relation between the evidence gathered by the user and the current working hypothesis to check whether an argument instantiated form the user's action is correct according to the influence diagram. If not, the conclusion of the counterargument generated from this rule states that this action should not be done. Rule 2 of figure 4 verifies the correctness of the user's argument according to value v_3 when she changes her current working hypothesis. If the precondition of this rule holds this means that there still remains evidence related to the current working hypothesis not yet gathered by the user. So a counterargument is generated with a conclusion stating that investigation of the current working hypothesis should be continued.

In the scenario of figure 1, based on the vital signs of the patient and information gathered from a short explanation which the patient has given about his pain, the student has produced three possible hypotheses: *Urinary infection* (UI), *Sexual Transmitted disease* (STD) and *Appendicitis* (AP). UI is the most probable hypothesis and AP is the least probable one but is threatening to the patient's life. Until line 4 of figure 1 the student has gathered some evidences which are related to UI so AM did not find any persuasive argument against these actions. Hence the system accepts these actions by remaining silence. But at line 4 the student has asked a query to gather evidence sexual partner (SP) which is not related to UI but is related to next probable hypothesis STD , yet there still remain two evidences related to UI but not gathered: *burning sensation on urination* (BSU) and *urinary lab test* (ULT). Figure 5(a) shows instantiated argument A from student's last action (of line 4) and an argument B generated by rule 1 of figure 4. The nodes show the conclusions of arguments. An arrow from an argument B to A indicates that B attacks A. According to [Bench-Capon, 2003], B is an acceptable persuasive argument since it promotes value v_1 which is preferred to value v_0 promoted by A. Value v_0 is a reserved value of V with the lowest preference which is used by the system for instantiated arguments from student's actions or utterances that does not promote any value. So argument B is given to PM to present it to the student.

As mentioned before, PM pushes the conclusion of B on AG as the goal of first entry. Then according to HR4.b premise "*Evidence BSU is not related to UI*" is pushed on AG. Then according HR4.d PM asks question in line 5 of the student. The system uses some simple text templates to display a text when it is to say something about an argument. Furthermore, to avoid of complexities utterance recognition, the student's utterances are entered through some options provided in some menus. Since the student's answer at line 6 is correct, according to HR3.a the system accepts the answer as indicated at line 7. And because this answer supports argument B of figure 5(a) which attacks argument A then according to the rule HR4.a the system challenges the user for her last query of evidence SP as shown in line 8.

The student's answer at line 9 indicates that she has changed her working hypothesis UI to the next probable hypothesis STD. According to HR3.c AM is called to find a persuasive argument against student's answer. Figure 5(b) outlines the conclusions of arguments generated by AM. The conclusions of some arguments like D and E as figure 5(b) shows may be identical but they are different in their premises and in the value that they promote. Argument A is an instantiated argument from Walton's template for student's last utterance. Arguments B, C and D are the first generated arguments by firing the rules of AM. Argument E is the last argument generated by the second cycle of running of AM.

According to figure 5(b) and the argumentation framework of [Bench-Capon, 2003], argument D is the only persuasive argument which is acceptable since value of D is preferred to values of argument C and A. Note that the value of D equals the value of B, but B is defeated by E. So E and D are acceptable. But as mentioned so far, a persuasive argument is selected from arguments which attack the student's argument. So the conclusion of argument D is pushed on AG according HR3.d. Then rule HR4.b is applied which pushes an instantiation of second premise of rule 2 of figure 4 with $ev = BSU$ and $whp = UI$. Thereafter, the only applicable rule is HR4.d which generates a question displayed in line 10. Since the student's answer at line 11 is incorrect, rule HR3.b is applied which as line 12 shows asserts the correct answer. Student's acceptation (line 13) will cause rule HR2.a to be applied and argument of line 14 to be asserted by the system. Finally since the student has accepted the last system's argument and AG is empty so rule HR2.c holds, argumentation dialog has finished and the system has won.

As the example shows, through an argumentation dialog the system has succeeded to oblige the student to reveal her understanding of current state of problem solving, to actively search in her knowledge to generate a convincing argument, reflect upon it, and finally revise her knowledge and improve her skill in problem solving as a result of the argumentation dialog outcome.

Different configurations of set V lead to different argumentation dialogs. For instance, given that v_5 is the most preferred value of V and probability of hypothesis STD has increased and it is currently the most probable hypothesis then student's counterargument in line 9 is acceptable. Or given that v_4 is the most preferred value of V then after line 9 argumentation dialog would unfolds in a way to persuade the student to investigate the hypothesis AP which is threatening to the patient life. Finally given that v_2 does not exist in V or it is less preferred with respect to the value of v_3 then any of the arguments B and D can be selected in random as a persuasive argument in discussion.

8 Discussion

Argumentation dialogs have been used so far in some DSS. Shankar [Shankar *et al.*, Spring 2006] presented an approach to render the knowledge involved in decision making for finding a solution in form of a well known argument structure defined by Toulmin [Toulmin, 1958] to enable the system to justify its decision or solution when the user asks for explanation. However, this work does not take into account convincing requirement of arguments w.r.t type of conflict in process of computing the persuasive argument.

Existing approaches which consider such requirements differ in the type of conflict which they address. Work by [Zukerman *et al.*, 1998] uses a probabilistic model of user belief to address the problem of computing the convincing argument when the conflict is due to the missing some step in chain of reasoning. Work by [Grasso *et al.*, 2000] addresses the problem of solving conflicts which are due to the different on values. But in contrast to our approach which computes a convincing argument by using argumentation techniques to promote values that the user holds they use a specific knowledge representation language to reason on values and to compute an argument which promotes the values that the system holds. In work [Chesñevar *et al.*, 2006] authors have presented an approach for computing persuasive argument according to the user preferences by integrating a specific argumentation framework which reason on information in the form of defeasible and strict rules in a declarative manner. Presenting information of user preferences in form of some rules makes this approach very complex to compute a persuasive argument when the order on preferences is important.

Our approach for computing convincing argument is somehow similar to approach of decision making by Atkinson [Atkinson *et al.*, 2006]. The main differences is that Atkinson uses Walton's template as a way to model a presumptive solution for decision making but we use Walton's template to instantiate user actions and utterances as some arguments to model interaction between the user and the system as argumentation. Also Atkinson uses TQs to verify and enhance the presumptive solution and to reach a decision but we use TQs as a way to evaluate user actions and utterance during interaction by taking into account the information of user model and domain knowledge.

As we mentioned before, one advantage of our approach is the ability of dynamic adaptation of calculating of convincing arguments w.r.t the user's preferences. We explained that changing the user profile or preference will result to different intervention of the system and different dialog argumentations as well. Also in our approach the strategies in the PM make the system able to recover from errors in the recognition of the user's goal during an argumentation dialog.

Our approach has some limitation that we intend to improve in future. Currently the PM generates simple text messages with some templates. Planning techniques can be used to generate more complex and flexible text messages. Also PM uses only one (premise-to-goal) strategy to conduct argumentation dialog which can be improved by adding more strategies. Finally the current system provides limited opportunity for the user to question the system. We intend to improve this limitation by providing more facilities for this purpose for example the user can change temporarily the preference order of values and then ask the system to provide a new convincing argument according to the new order.

9 Conclusion

In this paper we described an approach to modeling user interaction in a tutoring system for medical diagnosis in the form of argumentation using a dialog game and an argument template. We explained how to integrate a value based argumentation framework to find a convincing argument at given problem solving state and according to user preferences. To decide a move for the system when it needs to say something to the user during argumentation we presented some heuristic rules. While our approach has some limitations, it is promising for modelling argumentation dialog between the DSS and the user, specifically when arguments expressed by the system during a discussion have to be generated dynamically. Our system is in the preliminary stage of development and we intend to evaluate it by end-users in the future.

References

[Amgoud and Maudet, 2002] Leila Amgoud and Nicolas Maudet. Strategical considerations for argumentative agents (preliminary report). In *Proceeding of 9th International Workshop on Non-Monotonic Reasoning (NMR2002), Special session on Argument, Dialogue, and Decision*, pages 399–407, 2002.

[Amgoud and Parsons, 2001] Leila Amgoud and Simon Parsons. Agent dialogue with conflicting preferences. In *Proceedings of the International Workshop on Agent Theories, Architectures and Languages (ATAL01)*, pages 1–17, 2001.

[Atkinson *et al.*, 2006] Katie Atkinson, Trevor J. M. Bench-Capon, and Sanjay Modgil. Argumentation for decision support. In *Proceedings of the Seventeenth International Conference on Database and Expert Systems Applications (DEXA 2006)*, pages 822–831, Krakow, Poland, 2006. Lecture Notes in Computer Science (LNCS) 4080, Springer, Berlin.

[Bench-capon, 1998] Trevor J. M. Bench-capon. Specification and implementation of toulmin dialogue game. In *Proceedings of JURIX 98*, pages 5–20. GNI, 1998.

[Bench-Capon, 2003] Trevor J. M. Bench-Capon. Persuasion in practical argument using value-based argumentation frameworks. *Journal of Logic and Computation*, 13(3):429–448, 2003.

[Chesñevar *et al.*, 2006] Carlos Iván Chesñevar, Ana Gabriela Maguitman, and Guillermo Ricardo Simari. Argument-based critics and recommenders: a qualitative perspective on user support systems. *Data and Knowledge Engineering*, 59(2):293–319, 2006.

[Clancey, 1987] William J. Clancey. *Knowledge-Based Tutoring - The GUIDON Program*. Cambridge, MA: MIT Press, 1987.

[Crowley *et al.*, 2003] R. Crowley, O. Medvedeva, and D. Jukic. Slidetutor a model-tracing intelligent tutoring system for teaching microscopic diagnosis. In *Proceedings of the 11th International Conference on Artificial Intelligence in Education*, pages 157–164. IOS Press, 2003.

[Drusinsky, 2006] Doron Drusinsky. *Modeling and Verification Using UML Statecharts, First Edition : A Working Guide to Reactive System Design, Runtime Monitor-*

ing and Execution-based Model Checking. Newnes, April 2006.

[Dung., 1995] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321–358, 1995.

[Glasspool *et al.*, 2003] D. Glasspool, J. Fox, F. D. Castillo, and V. E. L. Monaghan. Interactive decision support for medical planning. In *Proceedings of the 9th Conference on Artificial Intelligence in Medicine Europe (AIME)*, pages 335–339, 2003.

[Grasso *et al.*, 2000] F. Grasso, A. Cawsey, and R. Jones. Dialectical argumentation to solve conflicts in advice giving: a case study in the promotion of healthy nutrition. *International Journal of Human Computer Studies*, 53(6):1077–1115, December 2000.

[Harel, 1987] David Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8(3):231–274, June 1987.

[Kabanza *et al.*, 2006] F. Kabanza, G. Bisson, A. Charneau, and T.-S. Jang. Implementing tutoring strategies into a patient simulator for clinical reasoning learning. *Journal of Artificial Intelligence in Medicine (AIIM)*, 38:79–96, 2006.

[Karunatillake *et al.*, 2009] Nishan C. Karunatillake, Nicholas R. Jennings, Iyad Rahwan, and Peter Mcburney. Dialogue games that agents play within a society. *Artificial Intelligence*, 173(9-10):935–981, 2009.

[Mackenzie, 1979] J. D. Mackenzie. Question-begging in non-cumulative systems. *Journal of Philosophical Logic*, pages 117–133, 1979.

[Maudet and Moore, 2001] N. Maudet and D. J. Moore. Dialogue games as dialogue models for interacting with, and via, computers. *Journal of Informal Logic*, 21(3):219–243, 2001.

[Moore, 1993] D. Moore. *Dialogue game theory for intelligent tutoring systems*. Ph.d. dissertation, Leeds Metropolitan University, Leeds, UK, 1993.

[Perelman and Olbrechts-Tyteca, 1969] C. Perelman and L. Olbrechts-Tyteca. *The New Rhetoric: A Treatise on Argumentation*. University of Notre Dame Press, Notre Dame, 1969.

[Pilkington *et al.*, 1992] R. Pilkington, J.R. Hartley, D. Hintze, and D.J. Moore. Learning to argue and arguing to learn: an interface for computer-based dialogue games. *International Journal of Intelligent Systems*, 3:275–295, 1992.

[Prakken, 2001] Henry Prakken. Relating protocols for dynamic dispute with logics for defeasible argumentation. *Synthese*, 127:187–219, 2001.

[Reed, 1998] C. Reed. Dialogue frames in agent communication. In *Proceeding of the Third International Conference on Multi-Agent Systems*, pages 246–253. IEEE Press, 1998.

[Restificar *et al.*, 1999] Angelo C. Restificar, Syed S. Ali, and Susan W. Mcroy. Arguer: Using argument schemas for argument detection and rebuttal in dialogs. In *Proceedings of User Modeling*, pages 315–317. Kluwer, 1999.

[Shankar *et al.*, Spring 2006] R. D. Shankar, S. W. Tu, and M. A. Musen. Medical arguments in an automated health care system. In *AAAI Symposia*, Stanford University, Spring 2006. IOS Press.

[Suebnukarn and Haddawy, 2005] S. Suebnukarn and P. Haddawy. Clinical-reasoning skill acquisition through intelligent group tutoring. In *Proc. of the 19th International Joint Conference on Artificial Intelligence (IJCAI-05)*, pages 1425–1430, July 2005.

[Toulmin, 1958] S. Toulmin. *The uses of arguments*. Cambridge University Press, 1958.

[Walton and Krabbe, 1995] D. Walton and E. Krabbe. *Commitment in Dialogue: Basic concept of interpersonal reasoning*. Albany NY: State University of New York Press, 1995.

[Walton, 1996a] D.N Walton. *Argumentation Schemes for Presumptive Reasoning*. Erbaum: Mahwah, NJ, 1996a.

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

[Yuan *et al.*, 2008] T. Yuan, D. Moore, and A. Grierson. A human-computer dialogue system for educational debate, a computational dialectics approach. *International Journal of Artificial Intelligence in Education*, 18:3–26, 2008.

[Zeleznikow and Stranieri, 1995] J. Zeleznikow and A. Stranieri. The split-up system: integrating neural networks and rule-based reasoning in the legal domain. In *Proceedings of the Fifth International Conference on Artificial Intelligence and Law (ICAIL)*, pages 185–194, New York, NY, USA, 1995. ACM Press.

[Zukerman *et al.*, 1998] I. Zukerman, R. McConachy, and K. Korb. Bayesian reasoning in an abductive mechanism for argument generation and analysis. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI)*, pages 833 – 838, 1998.