Automated Planning of Tutorial Dialogues

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

***Abstract*—Managing a dialogue between a student and an intelligent tutoring system is a challenging problem for many applications. It has often been argued and demonstrated that adaptive dialogues between a user and a computer can be generated automatically, using automated planning techniques to plan speech acts. To date such plan-based dialogue generation approaches have relied on deterministic planning algorithms. Consequently they can only handle sequential dialogue structures. In this paper we describe a new approach for automatically planning more general tree-like dialogue structures, by using a nondeterministic planner with incomplete knowledge and sensing. Our approach takes into account incomplete information about the user's knowledge by including queries that the computer can ask to the user to gather missing information that is necessary for an effective feedback. We illustrate our system with an application to an intelligent tutoring system for medical diagnosis.**

***Index Terms*—Dialogue processing, Believe revision and update, Planning under uncertainty, Intelligent Tutoring Systems**

I. Introduction

Adaptive and believable dialogues constitute an important aspect of intelligent capabilities in the interactions between a computer and a user, for many applications. This is particularly true for intelligent tutoring systems (ITS), that is, systems that try to teach a subject by providing a support similar to one by human teachers. In this context, dialogues specify what the system tells the student, when and how, to support his learning process in the most effective way. Dialogues can be involved to guide a student solve a particular problem, to help him recover from a mistake, present him new knowledge, or provide any other kind of feedback [1], [2], [3], [2].

Since such dialogues depend on the current learning situation (e.g., the current step in a problem solving process), the interactions that have to take place between the student and the system during the dialogue cannot be exhaustively specified off-line. Most current applications involve only partial dialogues that cover some learning situations identified in advance.

Automated planning has long been presented as a promising technique for automatically generating more adaptive user-computer dialogues [4], [5], [6], [7], [8]. The basic principle is to model speech acts as actions executed by an agent and dialogues as plans (i.e., combinations of actions) for achieving some communication goal. That way, existing planning algorithms can be applied to generated dialogue plans. Despite the recent development of automated planning algorithms that can deal with uncertainty, plan-based dialogue generation techniques still rely on deterministic planning systems. They do not take into account uncertainty about input from the user. Uncertainty continues to be handled solely at the level of the global artificial intelligence (AI) architecture, which invokes the dialogue planner each time the user input is not as planned for in the dialogue.

It is difficult to anticipate all mistakes a student will make when interacting with an ITS. We can't predict either the student's responses to queries made by the ITS (for instance when asking him whether he wants some help or not). If a deterministic planner is used to plan a dialogue for such situations, it will have to commit to only one of the possible course of events at the planning time, leaving the responsibility to the AI engine to replan should the choices made at the planning time happen to be wrong when the dialogue is ran. Such frequent re-planning can be a source of inefficiency in managing the interactions between the student and the ITS.

In this paper we describe a more robust planning approach which models the problem of generating a dialogue as one of planning with incomplete knowledge about the user. That way, the dialogue planner produces dialogues having a tree-like structure involving conditional branches on probable user inputs. A generated dialogue also involves queries to the user aimed at gathering information necessary to decide upon the next course of action in the dialogue plan. (e.g., the system may have to ask questions to the student to know what's wrong with him). Our dialogue planner is an application of the PKS planning algorithm, originally introduced by Bacchus and Petrick [9].

The remainder of the paper is organized as follows. In the next section we discuss our case study of an ITS for medical diagnosis. Then we give an overview of the dialogue system architecture, followed by a detailed description of its main components. Finally we present some scenarios, followed by a discussion on related works and a conclusion.

II. Motivation: Dialogues in an ITS for Medical Diagnosis

This work is motivated by the automatic generation of dialogues between a student and an ITS that teaches him to diagnose clinical problems. ITS for clinical diagnosis is a research area with a growing interest and several interesting prototypes have been recently developed, including [1], [3], [10]. In all these systems the dialogues are manually specified to cover a limited number of interactions between the student

Collected Evidence : Acute lower abdominal pain Working Hypothesis : Urinary infection	
1. Student Query: Patient:	Any fever? I don't know.
2. Student Query: Patient:	Could you describe your pain? Is it like a cramp, a burning sensation? It's not a cramp but it's very painful.
3. Student Query: Patient:	Is it worse on one side? No.
4. Student Query: Patient:	How many times did you urinate since the beginning of your pain? I don't know. More often I think.
5. Student Query: Patient:	Do you have a burning sensation on urination? No.
6. Student Query:	Do you have a sexual partner?
7. Tutor: Student:	Is this question relevant to the hypothesis you are working with?(Yes / No, I'm examining another hypothesis.) No, I'm examining another hypothesis.
8. Tutor: Student:	Did you finish working with urinary infection? (Yes/No) Yes.
9. Tutor: Student:	Are you sure? (Yes/No) Yes.
10. Tutor:	Change your working hypothesis before proceeding.

Fig. 1. Interaction scenario

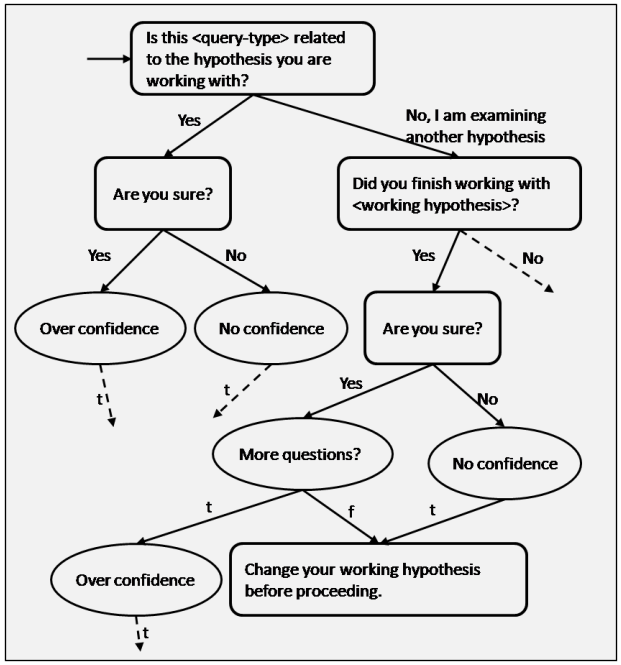

Fig. 2. A dialogue finite state machine. t = true; f = false.

and the system. Our system is an extension of TeachMed [10] to allow automatically planned dialogues.

A typical scenario begins with a student selecting a virtual patient having a particular disease with the objective to generate a correct diagnosis. The student makes a diagnosis by performing an investigation. He asks queries to the virtual patient about the different symptoms, life style and family background. He can

also make queries in terms of a physical exam on a 3D model of the patient e.g., reflexes) or in terms of lab tests (e.g., blood samples). Queries and tests are selected from a list including noise queries. Each query has an answer specified in the virtual-patient model, which includes his vital signs, symptoms and results of lab tests or physical exam.

Figure 1 illustrates an excerpt for queries on a virtual patient complaining of an abdominal pain. At the beginning, the student does not know what causes the pain. He asks queries to formulate some initial hypotheses. As more queries are asked, he will eliminate some hypotheses, strengthen others and generate new ones. This process continues until he can narrow the list of hypothesis to one or two, that is, the final diagnosis.

Some lab queries might be costly (e.g., MRI) or intrusive, so they are made if only necessary. On the other hand, forgetting to perform a particular test may lead to missing a pathology that threats the patient's health. The interview cannot go on endlessly either, so the student must ask as few questions as necessary. All these constraints call for appropriate feedback depending on how the student is behaving. Feedback is also necessary to help a student become aware of his mistakes and to recover from them. For instance, the student may forget to formulate a relevant hypothesis from the evidences he has collected so far, or fail to discard one that should be discarded. Such mistakes suggest a lack of appropriate clinical knowledge or skill by the student. Feedback in an ITS is not just meant to raise mistakes committed by the student, but most importantly to engage the student in a pedagogic dialogue that will make him understand the reasons behind those mistakes and foster the required clinical skills to avoid them later in similar situations.

TeachMed uses an influence diagram (ID) [11] to model the causal relationship between symptoms and diseases and the utility of queries. Given the evidence collected by the student at the current stage and the current differential diagnosis, inference on the ID is used to determine the next queries or tests

with high added value of information (e.g., queries or tests that would increase the likelihood of a current hypothesis or discard one). These queries are then compared to those made by the student to provide feedback if there are significant discrepancy. On the other hand, given the evidence collected so far, the system can determine the most likely hypotheses in the ID and compare with the differential diagnosis formulated by the student so far to provide feedback accordingly.

To monitor the student's solving trace, TeachMed uses production rules, such that the preconditions of a rule are matched by situations requiring intervention (e.g., the student has failed to identify an hypothesis from the evidences he has collected so far) and the postcondition is a trigger of a feedback dialogue for handling the situation.

In the original TeachMed version [10], the dialogue is modelled explicitly as a finite state machine (FSM); that is, a graph with two types of states: display state and internal state. In a display state, the system displays a multiple-

choice message to the student and waits for him to answer by one choice, then moves to the corresponding transition; messages involve place holders (delimited by $<$ and $>$ in the example below) to be filled by values of corresponding variables at the time they are displayed. In an internal state, the system is making some computations, producing a symbol as output, and moving to a state along a transition matching the output. Figure 2 shows a dialogue for a student perceived to be changing his hypothesis without explicitly updating his hypothesis table. Display states are drawn with a rectangle containing part of the displayed message. Internal states are drawn in ovals. Entry states are indicated by an arrow with no origin state.

For the extension of TeachMed described here, feedback rules have a postcondition that is communication goal, rather than an FSM dialogue, so that the actual FSM dialogue is generated online by a planner. This permits more complex and adaptive dialogues, for which the structure is determined dynamically, rather than being enforced off-line. Before describing the planner, it is interesting to illustrate a situation leading to the activation of a dialogue such as in Figure 2.

Consider again the scenario in Figure 1. Initially, the student is given a virtual patient to diagnose. The interaction between the student and the TeachMed begins with a straightforward question-answer loop as follows : the student is querying the virtual patient and getting answers that are specified in the patient model. After a few questions and corresponding answers, the student is working on the hypothesis of urinary infection. Until step 5, the student has been making queries that are related to this hypothesis, trying to confirm it or discard it. The ITS knows what current hypothesis the student is working on because he must specify it in an appropriate window. On step 6, the ITS determines (through inference on the ID) that that the query is not related to the working hypothesis (Urinary infection), but is related to another hypothesis (sexually transmitted disease, (STD)). A feedback rule matching this situation triggers a call to the planner with a goal of remedying the situation. As a remedy, the planner then generates the dialogue FSM in Figure 2. Then the interaction in Figure 1 resumes in step 7, this time being driven by the dialogue FSM in Figure 2.

As the message displayed on step 7 indicates, the placeholder <query-type> is replaced by "question" because the value of this variable is "question" (meaning "interview question"). The displayed message is a concatenation of the message in the entry state with labels on outgoing transitions. The remainder of the dialogue can be followed easily from Figure 2, given the choices made by the student when answering multiple choice questions displayed by the dialogue process.

In the original TeachMed version [10], there are many such dialogues covering the different kinds of feedback. Some of these dialogues are quite complex to specify, considering that the purpose is not to provide the answer to the student immediately, but to check first whether the perceived situation is indeed correct, to engage in a dialogue where the student's confidence will be tested (e.g., by the question "are you sure?" , but this can get more complex), and then to encourage the student to identify the error himself.

With our new version, the virtual patient designer does not have to enumerate all possible dialogues. He only needs to specify generic speech acts involved in the different dialogues by using templates of speech acts (i.e., planning operators), specifying when they are invoked and what are their effects on the student's belief state. It then becomes the role of the planner to generate a particular FSM dialogue that is a tree of speech acts for a given communication goal. This is not only reduces costs in specifying medical diagnosis learning objects, but also makes the system more adaptive.

III. Dialogue Planner

The dialogue planner applies PKS planning algorithms [9]. At the planning time, the planner may have incomplete knowledge about the student's belief state (e.g.: What hypothesis is the student working with? Does the student know that this symptom can be caused by this particular pathology?). Since the planner is able to reason about the uncertainty on the student's belief state, it can generate dialogue plans that include questions to the student, aimed at acquiring necessary information at the execution time to run the dialogue succesfully. These questions can be seen as "sensing actions" made by the ITS. The ITS is an agent executing a dialogue plan and it senses its environment (the student) by asking questions.

Hence in our context a dialogue FSM is a plan generated by PKS. The input for generating such a plan is an initial state, a goal, and plan operators. The initial state contains facts from the student's model and the current solving step that re relevant; in particular; these include facts that matched the precondition of the rule that triggers the goal for the dialogue.

A. Goals

A goal is a disjunctive formula on the student belief state. Predicates express basic facts about the student's belief state. We use $StBel(f, x_1, \ldots, x_n)$ to express the student belief. The first argument denotes the kind of the belief and the remaining arguments are objects concerned by the belief. For instance, we use $StBel(causes, U, SP)$ to express the fact that the student believes there is causal relationship between urinary infection (U) and having a sexual partner (SP). We also use $StBel(change, U)$ to means that student believes he should change his working to urinary infection (U) (i.e., he should start asking queries on the virtual patient that will help him confirm or discard the new working hypothesis).

Thus the goal $(not \quad StBel(causes, U, SP) \quad | \quad StBel(change, U))$ is achieved by an FSM such that by running it, the final student's belief state will satisfy the formula. Remember the FSM is triggered because the ITS has noticed that the student has asked a query not related to the current working hypothesis (urinary infection). So either the student is wrongly thinking that the query is related to urinary infection, or in he is actually focussing on a new hypothesis (but he forgot to specify the new working hypothesis in the appropriate window on the ITS interface). An dialogue satisfying the previous dialogue will make the student realize which one of the two situations he should normally be in.

Speech Act	Group	Text	Preconditions	Effects
InformNoRelation(<hp>, <q>)	E	This <q> is not related to the <hp>	K(StAnswer(askConfCauses,<hp>,<q>)) & notK(casualRelation(<hp>,<q>))	add(Kf,not StBel(causes,<hp>,<q>))
CommandUpdateWhp(<hp>)	O	Update your Working hypothesis.	(K(StAnswer(askConfFin,<hp>)) | K(not StAnswer(askConfFin,<hp>))) & K(WorkingHP(<hp>)) & K(Finished(<hp>))	add(Kf,StBel(change,<hp>))
AskRelation(<hp>, <q>)	I	Is this <q> related to the hypothesis you are working with?(Yes/No)	K(StAsked(question,<q>)) & K(WorkingHP(<hp>)) & notK(casualRelation(<hp>,<q>)) & notK(StBel(causes, <hp>, <q>)) & notK(StHeared(askCauses,<hp>,<q>))	add(Kw, StAnswer(askCauses,<hp>,<q>)) add(Kf,StHeared(askCauses,<hp>,<q>))
AskFinishedWhp(<hp>)	I	Did you finish working with <hp>?(Yes/No)	K(not StAnswer(askCauses,<hp>,<q>))& notK(StHeared(askFin,<hp>)) & notK(not StBel(finished, <hp>)) & notK(StBel(finished, <hp>))	add(Kw, StAnswer(askFin,<hp>)) add(Kf,StHeared(askFin,<hp>))
AskConfFinished(<hp>)	I	Are you sure?(Yes/No)	K(StAnswer(askFin,<hp>)) & notK(StBel(confFin,<hp>)) & notK(not StBel(confFin,<hp>)) & notK(StHeared(askConfFin,<hp>))	add(Kw,StAnswer(askConfFin,<hp>)) add(Kf,StHeared(askConfFin,<hp>))
AskConfRelation(<hp>,<q>)	I	Are you sure?(Yes/No)	K(StAnswer(askCauses,<hp>,<q>)) & notK(StBel(confCauses,<hp>,<q>)) & notK(not StBel(ConfCauses,<hp>,<q>)) & notK(StHeared(askConfCauses,<hp>,<q>))	add(Kw,StAnswer(askConfCauses,<hp>,<q>)) add(Kf,StHeared(askConfCauses,<hp>,<q>))

Fig. 3. Brief description of speech acts used in dialogue plan generation. E = Explanatory, O = Ordering, I = Interrogative

B. Belief State

PKS planner produces a plan (in this case, a tutorial dialogue plan) by searching a space of believes. During search, the planner is in a particular belief state. The space is explored by applying templates of speech acts to generate successors of the current belief state in forward chaining process. One of the interesting

features of PKS is that each belief state is partitioned into four sets, allowing to make efficient logic inference during the search for a plan [9]. Two of these components are relevant for our application:

- A set of known facts ($\mathbf{K_f}$), that is, the set of facts for which the planner knows the true at the planning time in the current state.
- A set of knowable facts ($\mathbf{K_W}$), that is, the set of facts for which the planner does not know the truth at the planning time, but for which it has planned sensing actions that will determine the truth at the run time.

For our application, known facts are facts in the initial state and thereafter facts that are entailed by explanatory and ordering speech acts. For instance, initially $StAsked(Question, SP)$ and $WorkingHP(U)$ are in $\mathbf{K_f}$ if the student has asked a question about the sexual-partner evidence and working hypothesis is urinary infection. If at planning time, the planner applies a planning operator that models an order instructing the student to do something (e.g., change your working hypothesis) the effect of will be an update on $\mathbf{K_f}$. Knowable facts are facts entailed by interrogative speech acts. For instance, if the planner applies an operator modelling a query to the student (e.g., asking the student whether he has finished working with Urinary hypothesis), the effect is an update on the $\mathbf{K_W}$ component of the successor state to reflect that the student's answer (yes or no) will be known at the run time when the query action is actually executed.

C. Planning Operators

Display states in the FSM correspond to speech acts. To generate them, the planner requires as input planning operators that specify templates of speech acts. Each planning operator describes the situation in which a type of speech act is appropriate (i.e, the precondition of the planning operator) and its effect on the current state. There are three types of speech acts:

- *Explanatory speech acts* display an explanation to student.
- *Ordering speech acts* give orders to the student about the next step in the clinical problem solving.
- *Interrogative speech acts* ask questions to the student.

Explanatory and ordering speech acts deterministic and always affect only the $\mathbf{K_f}$ component of states. Interrogative speech acts are non deterministic and affect both $\mathbf{K_f}$ and $\mathbf{K_W}$ state components. Nondeterminism here accounts for the possible answers by the students for a question represented by the interrogative speech act. This nondeterminism induces a conditional branch in the dialogue plan structure.

Figure 3 illustrates a simple planning operators. PKS uses the logic modal operator K to express the planner's knowledge about something. For any first-order formula, $K(\phi)$ means that the planner knows ϕ. Symbols between braces ($< >$) are to be interpreted as template placeholders (or predicate variables). A planning operator is applied to a state by replacing the

Algorithm 1 Dialogue planner algorithm

```
1.  begin  DP(s, p, g)
2.    loop
3.        if GoalSatisfied(s, g) then return p
4.          if K_w ≠ ∅ then
5.              Pick(α) : α ∈ K_w
6.              Branch (s, α, s_1, s_2)
7.              C := {DP(s_1, ∅, g), DP(s_2, ∅, g)}
8.              if failure ∈ C then return failure
9.              else return p, C
10.         applicable ← {A | PrecondsSatisfied(A, s)}
11.         if applicable = ∅ then return failure
12.         Pick(A) : A ∈ applicable
13.         s ← ApplyEffects(A, s)
14.         p ← p, A
15. end
```

variables with values that make the operator's precondition true in the current state. Each substitution gives an action (that is, a fully instantiated planning operator), which creates a successor of the current state by applying its effects. The *effects* column describes the updates made by the application of the operator to a current state. The *add* keyword in the effects means that the effect is added to the indicated state component. The *text* column shows the text that is displayed when the speech act is executed.

D. Planning Algorithm

PKS planning algorithm is shown in Table 1. Its input are: an initial belief state (s); an empty plan (p); and a communicative goal (g). A part from the conditional if statement on line 4 and its nested statement (lines 5 to 9), this is similar to forward-search algorithm. The if statement on line 4 checks the $\mathbf{K_W}$ state component for an entry; if one exist (say α) it is removed from $\mathbf{K_W}$ (i.e, PICK statement, line 5) and then two new belief states s_1 and s_2 are created from the current state s by adding α and $\neg\alpha$ (i.e., BRANCH statement, 6); then (through lines 7 to 9) two recursive invocations of the algorithm take place. If the goal holds in all the created branches by the recursive calls, then after returning to the top-level invocation of the algorithm the nested list p is returned as the solution plan, otherwise a $failure$ is returned. A correctness proof of this algorithm can be found in [9].

Figure 4 shows a portion of the plan generated from: an initial state whose $\mathbf{K_f}$ component is $\{StAsked(question, SP)$, $WorkingHP(U)$, $Finished(U)\}$ (the later fact is one of the facts that shows the planner knowledge about the problem and means that planner knows urinary infection has finished, we ignored the other facts of initial state that are not relevant to our example) and the other components ($\mathbf{K_W}$, $\mathbf{K_V}$, $\mathbf{K_X}$) are empty; the goal is $(not\ StBel(causes, U, SP) \mid StBel(change, U))$; and the planning operators are in Figure 3. Nodes in the plan of Figure 4 correspond to instantiations of planning operators. Transitions correspond to conditional branches spawned by the update of $\mathbf{K_W}$ components. One obtains the dialogue in Figure 2 from the previous plan by replacing the action in nodes by corresponding message texts as given in the description of planning operators (Figure 3). This dialogue was produced in 1 second on a Pentium 1.8 GHZ.

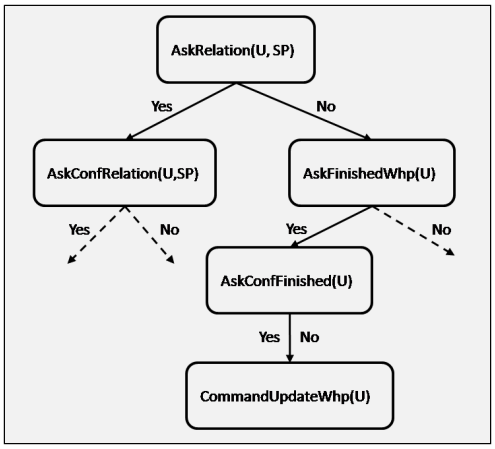

Fig. 4. A part of the tree structure representation of generated dialogue plan

IV. Conclusion and Related work

In this paper we showed a new dialogue planning approach that is able to take into account the uncertainty about user input in the dialogue process and incomplete knowledge about the users knowledge of problem or task in planning process. This is a significant shift from previous dialogue planning approaches which use deterministic approaches. As mentioned in the introduction, a lot of previous work have considered the problem of automatically generating user-computer dialogues by using planning techniques [4], [5], [6], [7], [8]. APE [7] is a recent system that uses a deterministic dialogue planner that handles the uncertainty of user input by replanning. BEETLE [8] uses a 3-tier planning architecture to handle different aspects in dialogue generation, but still a deterministic planner is used.

Currently speech acts are modelled by planning operators with corresponding template messages. More natural messages could be displayed by using automatically generated texts rather than templates, by integrating a discourse planning method [12], [5], [4], [13]. In fact, we view dialogue planning at high level where one is interested in planning the structure of the dialogues, that is, the turns for the participants in the dialogues. Discourse planning would occur at a lower level of speech acts (i.e., each speech acts becomes itself the output of a planning or some other automated generation process). For instance, if a system has different ways of communicating a message, discourse planning would help determine the best approach.

References

[1] R. Ganeshan, W. Johnson, E. Shaw, and B. P. Wood, "Tutoring diagnostic problem solving," in *Proceedings of the Fifth Int'l Conf. on Intelligent Tutoring Systems*, 2000.

[2] K. VanLehn, C. Lynch, K. Schulze, J. Shapiro, and R. Shelby, "The andes physics tutoring system: Lessons learned," *International Journal of Artificial Intelligence in Education*, vol. 15, no. 3, 2005.

[3] S. Suebnukarn and P. Haddawy, "Clinical-reasoning skill acquisition through intelligent group tutoring," in *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI2005)*, Edinburgh, UK, July 2005, pp. 1425–1430.

[4] M. Pollack, "A model of plan inference that distinguishes between the beliefs of actors and observers," in *Proceedings of the 24th Meeting of the Association for Computational Linguistics*. Morristown, New Jersey: Association for Computational Linguistics, 1986, pp. 207–215.
[5] P. Cohen and H. Levesque, "Intention is choice with commitment," *Artificial Intelligence*, vol. 42, pp. 213–261, 1990.
[6] A. Cawsey, "Planning interactive explanations," *International Journal of Man-Machine Studies*, vol. 38, Issue 2, pp. 169–199, February 1993.
[7] R. Freedman, "Plan-based dialogue management in a physics tutor," in *Proceedings of the Sixth Applied Natural Language Processing Conference (ANLP)*, Seattle, 2000.
[8] M. G. Core, J. D. Moore, and C. Zinn, "Supporting constructive learning with a feedback planner," *AAAI Fall Symp. Building Dialogue Systems for Tutorial Applications,Cape Cod, MA*, vol. 15, no. 3, November 2000.
[9] R. P. A. Petrick and F. Bacchus, "Extending the knowledge-based approach to planning with incomplete information and sensing," in *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS-04)*, 2004, pp. 2–11.
[10] F. Kabanza and G. Bisson, "Clinical reasoning learning with simulated patients," pp. 385–394, 2005.
[11] R. E. Neapolitan, *Learning Bayesian Networks*. Prentice Hall, 2003.
[12] M. Bratman, *Intentions, Plans, and Practical Reason*. Harvard University Press, 1987.
[13] R. M. Young, J. Moore, and M. Pollack, "Towards a principled representation of discourse plans," in *Proceedings of the Sixteenth Conference of the Cognitive Science Society*, Atlanta, GA, 1994.