Journal of mathematics and computer science 10 (2014), 266-274

Contents list available at JMCS

Journal of Mathematics and Computer Science

Journal Homepage: www.tjmcs.com

Negotiation planning of autonomous agents in multi-agent environment

Nahid Sadeghpoor[1], A. Rahati[2]
[1,2] *University of Sistan and Baluchestan, Zahdan, Iran*
sadeghpoor@pgs.usb.ac.ir

Article history:
Received April 2014
Accepted May 2014
Available online May 2014

Abstract

In negotiation process, when a conflict arises between people in a society, each will try to pose some arguments to resolve the conflict and convince the other party. Successful termination of many of these interactions is not achieved only via exchange of a series of single-message suggestions and requires that an arguer offer compelling justifications to prove his/her claim. Obviously, before negotiation is started, decision should be made about the terms and arguments that can convince the other side and terminate negotiations successfully. In other words, an exact plan of arguments and their order should be prepared. The same is true about a multi - agent environment where each agent needs planning and executing a sequence of actions to achieve its goals. Some actions in these sequences are under the direct control of the planner agent and others require negotiations with other agents. Negotiation is usually carried out during the plan execution, however, it can be considered during the planning stage, as in real life. In this paper, we present a new approach that takes into account incomplete information about opponent's position in negotiation by using a nondeterministic planner with incomplete knowlage and sensing actions. We evaluate our approach by solving a well known negotiation scenario.

Keywords: Autonomous agents, conflict, multi-agent environment, negotiation planning.

1. Introduction

A multi-agent environment consists of some multiple independent agents that interact in a domain, where each agent needs to plan a sequence of actions in order to achieve its goals[1]. Some of these actions are under the direct control of agents but executing those that are not requires agents interact with each other to ask agents who has the direct control to perform actions on behalf of them[2]. As such,, negotiation can be simulated by communication among agents in a multi-agent environment, in particular by modeling dialogue between two agents with conflicting goals when they try to reach an agreement in both competitive and collaborative scenarios [3, 8]. In this context, negotiation process is considered as a dialogue where parties exchange some proposals and response to proposals to comunitcate and consequently to reach a mutually acceptable agreement.

In the multi-agent literature, various interaction and decision mechanisms for automated negotiation have been proposed [4-7]. Among them, argumentation-based negotiation has been more successful since it provides a fundamental method for modeling interaction between agents in multi-agent systems and in

particulare for modeling the negotiation process among agents. Much similar to a real negotiation scenario between people, in addition to the information uttered with a proposal, these approaches allow agents to exchange some additional information required to justify a claim or to put forward as a new argument and reject each others' proposals[9, 10].

In the multi-agent literature from past to now, an agent traditionally builds a plan that determines a sequence of actions with no regard for the negotiation; i.e. it builds a plan, starts to execute it and wherever needs to execute actions that are not under its control, starts negotiation. But, if the negotiation fails at this time, that means the current plan has failed and the agent must build a new one. In order to reduce the failures at execution time, Monteserin and Amadi proposed a new way in their work [8]. They argued that agents' negotiation process can be planned before the negotiation takes place. In this study we develop their work by suing a more robust planning approach which considers agents' incomplete knowledge about their counterpart in negotionation. In particular, based on the negotiation framework that defined in [11], our planning approach is able to take into account the possible opponent's responses during negotiation planning, such as accept or reject a proposal. So, in our approach the negotiation plan has a tree like structure involving conditional branches on responses of the opponents. Our planner is an application of the PKS planning algorithm, introduced by Bacchus and Petrick [12, 15].

This paper is organized in the following way. In the next section, we present motivation of this work by describing a well known negotiation scenario between some agents. Thereafter, we show how determinisctic planners can resolve such disagreement among agents by taking into account only one course of (negotiation) actions in planning time. In section 3, we decribe our nondeterministic negotiation planning algorithm that can overcome this shortcome by modeling incomplete knowedge through some sensing action and specific way of knowledge modeling. Finally, we state our conclusions and suggest future work.

2. Motivation: negotiation in a multi-agent environment to solve conflicts

This work is motivated by the automatic generation of arguments between two agents that they utter during the negotiation process to solve conflict. To better understanding the aim of this work, consider a well-known and widespread negotiation scenario used in other studies [8, 13]. This scenario outlines a situation in that there is some scare resources available to some agents how needs them to achieve their goals. Obviously, conflicts will arise among these agents when they need same resources and it must be resolved through negotiation. Scenario describes a home as an environment in which three agents named Picty, Mirry and Chairy are working. Each of them has a specific goal, some beliefs and resources as shown in Table 1.

Table 1. Goals, beliefs and set of resources.

Agents	Goals	Resources	Beliefs
Picty	hang a picture	a picture, a screwdriver, a screw, a hammer.	I can hang the picture by using a nail, a hammer and a chair. I believe that it is possible to hang the mirror with a screwdriver and screw.
Mirry	hang a mirror hold a glue	a mirror, a nail, a glue, a hammer.	I can hang the mirror by using a nail and a hammer.
Chairy	fix a broken chair	a broken chair	I can fix my broken chair by using a glue.

Given the information of Table 1, the following three conflicts between Picty, Mirry and Chairy are detected:

1) Picty needs the nail, and Mirry needs the hammer.
2) Picty needs the chair, but Chairy must fix this at first.
3) Chairy needs a glue, but Mirry wants to keep it [8].

If Picty is the planner agent who needs to achieve its goal by considering the above conflicts, so it must first get nail from Mirry, then get glue from Mirry and give it to Chairy, because Picty needs a fixed chair for its goal. Thereafter Chairy repairs the chair and will give it to Picty and finally it can hang the picture. Figure 1.a shows this plan by using actions given in Table 2. It is readily seen that when Picty starts to execute this plan, the first two actions must be negotiated. So, if the negotiation for these actions fails in the execution time then the plan has failed and the agent must replan an alternative one.

This problem can be more complex when the number of agents in the environment increases, for example by adding some new agents ($adhy_i$) that each one has a glue and their goal is to keep it[8]. As such, Picty can give glue to several agents by performing action *GiveResourceTo (agent, Chairy, glue)* of Table 2. Suppose Picty chooses an agent named $Adhy_1$ for doing this action and starts to execute the plan depicted in Figure 1.b. As we know, negotiation fails for this action, because it has conflict with $Adhy_1$'s goal. Hence, Picty needs re-planning.

Table 2. Actions that Picty needs to achieve its goal.

Action	Preconditions	Effects
hangAPicture(Agent, Picture, Hammer, Nail, Chair)	iam(Agent), cando(Agent, hangAPicture), has(Agent, picture(Picture)), has(Agent, hammer(Hammer)), has(Agent, nail(Nail)), has(Agent, chair(Chair))	pictureHanging(Picture), not(has(Agent, nail(Nail)))
giveResourceTo(AgentS, AgentD, Resource)	isagent(AgentD),isagent(AgentS), notEqual(AgentS,AgentD), has(AgentS, Resource), do(AgentS, giveResourceTo (AgentS, AgentD, Resource))	has (AgentD, Resource), not (has (AgentS, Resource))).
fixAChair(Agent, BrokenChair, Glue),	isagent(Agent), cando(Agent, fixA Chair), has(Agent, broken Chair (Broken Chair)), has(Agent,glue(Glue)),do(Agent,fixAChair (Agent, BrokenChair, Glue)),	has (Agent, chair(BrokenChair)), not(has(Agent, glue(Glue)))]).

Now, we want to reduce these failures when planning the negotiation among agents by taking into account alternative arguments that an opponent may put forwards in response to the argument of planner agent and also making decision in advance (before executing the negotiation plan) about which agents, when and how can negotiate with planner agent.

Consider the negotiation scenario for after executing action *GiveResourceTo (Mirry, Picty, nail (n1))* as shown in table 3. Picty wants to take nail from Mirry. If Mirry accept without any condition then negotiation has succeeded and the plan can be continued. But, if Mirry reject the request made by Picty, then Picty must persuade him by providing some justification based on his knowledge about mirry's beliefs and goals. Figure 2 shows this negotiation between two agents as a finite state machine (FSM). Display states are drawn with a rectangle containing the arguments and justification by Picty and Mirry and responses or counterargument are shown by arrows. We want to simulate this FSM by planning approaches that we use in next section and build a negotiation plan.

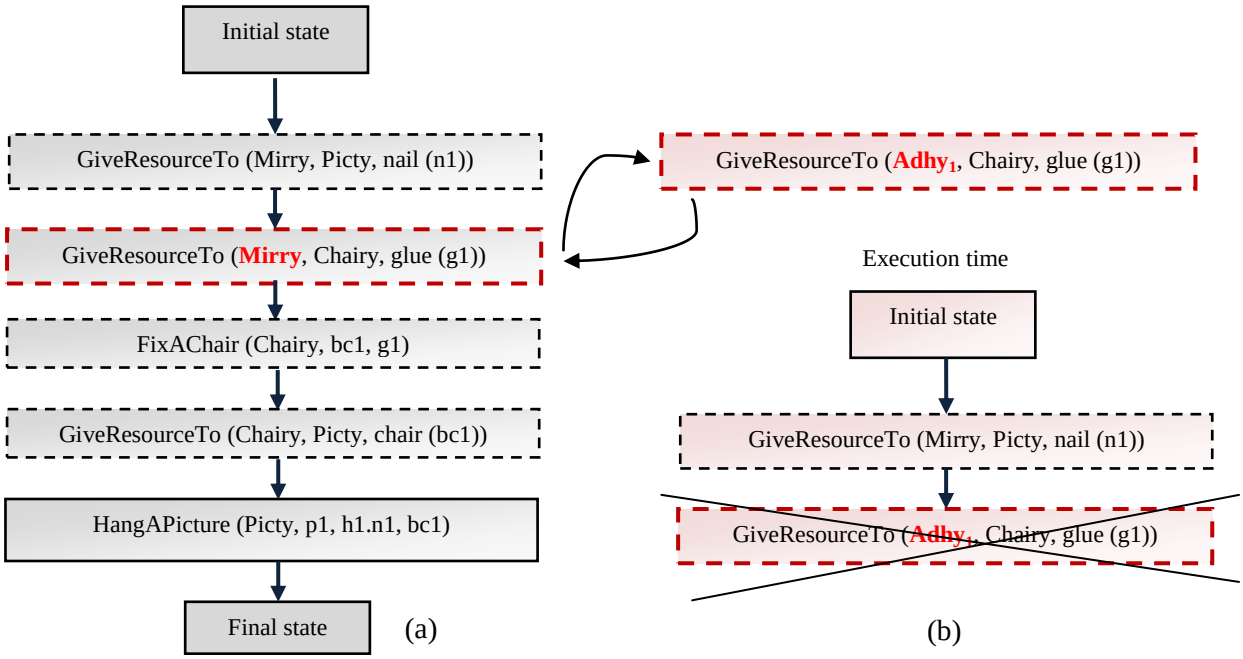

Figure1. plan of picty where: a) Mirry is only agent that can do action two, b) adhy$_i$ and Mirry are agents that can do this action.

Table.3. Negotiation process between Picty and Mirry

Picty:	I propose you give me your nail (accept/reject)
Mirry:	No, I can't because I need it for achieving my goal.
Picty:	But you can hang mirror with screw and screwdriver. So I propose you give me your nail and do your goal with these 2 things. (accept/ reject)
Mirry:	Ok, I accept and give you the nail.

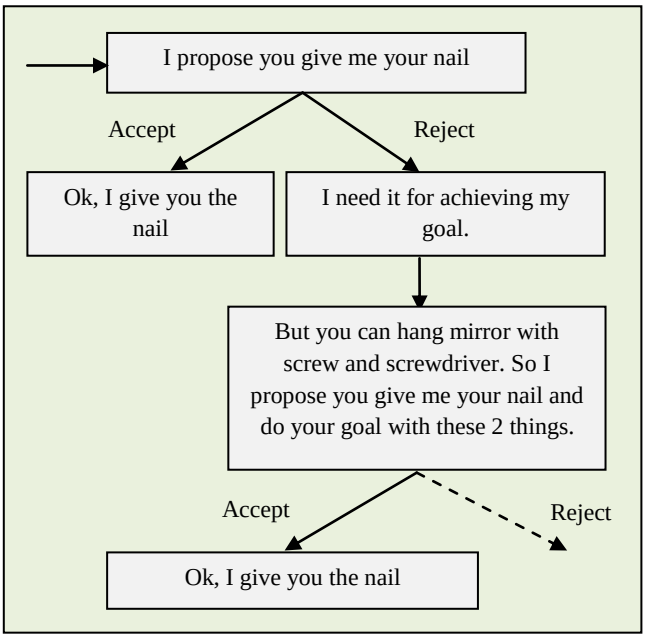

Figure.2. Negotiation final state machine

3. Negotiation planner

The negotiation planner applies PKS planning algorithms [12, 15]. This planner must take into account incomplete knowledge of the planner agent about the opponent's belief state and construct a tree-like plan by using some sensing actions. In this plan, each branch is constructed according to one of the response of the opponent (accept/reject) to planner request and sensing actions specified which of them should be executed at the execution time. Hence in our approach, a negotiation FSM is a plan generated by PKS. The inputs for generating such a plan are an initial state, a goal or final state and negotiation actions. The

planner will use these inputs to search for a plan and solve the planning problem. In our work, the initial state contains facts that planner agent knows about environment, opponents and itself. On other words, it describes the world where the conflict takes place in order to generate negotiation plan. Final state or goal of any negotiation planning problem is where according agent's planner knowledge about itself and opponents' belief state represents conflict resolution or mutual agreement among all agents.
We use some logic predicates to represent some simple facts that are expressed by negotiation language. These predicates are shown in Table 4.

Table.4. negotiation language

Predicate	Description
iam(X)	X is the negotiator agent.
isagent(X)	X is an agent
believe(X, B)	X believes B.
isgoal(X, G)	Agent X has G in its goals
prefer(X, G1, G2)	G1 and G2 are X's goals, and X prefers to fulfill G1 over G2.
imply(A, B)	A implies B
cando(X, A)	X can do action A.
do(X, A)	X will do action A
has(X, S)	X has resource S.
wasgoal(X, G)	X pursued goal G in the past.
did(X, A)	X did action A in the past.
fulfilled(X, G, A)	in the past, X fulfilled goal G by doing action A.
Propose(X,Y,B)	X propose to agent Y do/believe B.
Propose_with_just(X,Y,B)	X proposes to agent Y do/believe B with express some justification**.**
Prososе_with_reward(X,Y,A,B)	X proposes to agent Y to do/believe B with give it a reward.
Prppose_with_threat(X,Y,A,B)	X proposes to agent Y to do/believe B by threaten it.
Opponent_answer(X,Y,A,G)	This predicate shows feedback of the opponent Y. If it is true means it accept X's request A**.**

3.1. Agent's Knowledge

When a planner has a complete knowledge about its world by making close world assumption, it is sufficient to use a single database, D, to represent the agent's knowledge. But, in our study, we assume that the agent knowledge about its opponents and environment is incomplete. So, it is necessary to develop a representation that can model this incomplete information and sensing actions as well as the knowledge that an agent will come to know at some point in the future. **For this purpose,** according to Petrik & Baccuch work [12], we use a collection of databases that each one will be used to maintain a particular type of knowledge. Two of these databases are relevant to our application:

1) k_f: a database that maintains a set of facts for which the planner knows the true at the planning time.
2) k_w: a database that maintains a set of facts for which the planner does not know their truth at the planning time, but for which it has planned sensing actions that will determine the truth at the execution time[8,14].

In this work, we use k_w to model incomplete knowledge of planner agent about the opponent's position in negotiation, for example, wheter opponent's accept a request for an action or reject it.

3.2. Actions

Planning actions represent arguments that agents can utter during negotiation with its counterparts. By considering the display states in the FSM, there are two general actions or argument types:

1) Planner agent's arguments: these arguments (actions) are used by an agent to utter its proposals or some justification to persuade its opponent.
2) Opponent's arguments: these are arguments that are uttered as response to a proposal. Since these arguments are unknown at the planning time, so, we have used sensing actions for this purpose. In another words, the effect of these actions are placed in k_w that cause some branches to be created in plan, one branch correspond to each possible counterpart's response such as acceptance or rejection. In execution time, effect of these actions is used to know which branch of the plan must be continued, to remain on the correct track of plan and avoid replanning that may be necessary in previous planning approaches that consider only one course of actions during planning. [12].

Table.5 illustrates some planning actions used in this work. Each action has a precondition and an effect part. For an action to be added in a plan, its preconditions must be satisfied, if so, it is assumed that the world will be modified by the effect of this action. *K(φ)* in the preconditions means that the planner knows *φ*[12,15,14]. Also, the *add* keyword in the effects means that the planner agent's current state of beliefs will change according to facts stated after it.

Table.5. some simple planning actions

Actions	Preconditions	Effects
propose(X, Y, action A, Goal)	*k (iam(X)), k (isagent(Y)), k (isgoal (Y, Goal)), k (believe (Y, imply (do(Y, action A), not (goal)))), k (not (isgoal (X, Goal))), k (not (believe (X, imply (do(Y, action A), not (goal)))),*	*add (k_f , propose (X, Y, not (imply (do (Y, action A), not (goal))))).*
Sense _ proposes(X, Y, action A, Goal)	*k (iam(X)), k (isagent(Y)), k ((Propose (X, Y, not (imply (do(Y, action A), not (goal))) or (Propose _ with_just (X, Y, not (imply (do (Y, action A), not (goal))))), not (k (opponent _ answer (X, Y, action A, Goal)))*	*add (k_w , opponent_answer (X, Y, ation A,goal))*
accept_propose (X, Y, action A, Goal)	*k (iam(X)), k (isagent(Y)), k (opponent_answer (X, Y, action A, Goal)))*	*add (k_f , do(Y, action A), add (k_f , not (believe (Y, imply (do (Y, action A), not (goal)))). del (k_f , (believe (Y, imply (do(Y, action A), not (goal))))), del (k_f , (Propose (X, Y, not (imply (do (Y, action A), not (goal))) or (Propose_with_just (X, Y, not (imply (do (Y, action A), not (goal))))),*
assert_justif (X, Y, action A, action B, Goal)	*k (iam(X)), k (isagent(Y)), k (notequal (action A, action B)), k(believe (X, imply (do(Y, action B), Goal)), k (believe(X, imply (Resources, do (Y, action B)))), k (action (action B, C, P, E)). k (believe(X, imply (do(X, action A), not (goal)))). k(not(opponent_answer (X, Y, action A, Goal))).*	*add (k_f , Propose_with_justif (X, Y, not (imply (do(Y, action A), not (goal)))),del (k_f , Propose (X, Y, not (imply (do(Y, action A), not (goal)))).*

3.3. Negotiation algorithm

PKS negotiation planner algorithm is shown in Algorithm 1. It has three inputs: initial knowledge state (s), goal of negotiation (g) and an empty plan (π). The *if* statement in line 4 checks k_w database, if it incluses an element like α, then the planner picks it up and creates two branches s_1 and s_2 from the current state s, such that one contains α and the other ~α (line 6). Thereafter, algorithm continues recursively for each branch (line 7). Finally, if the goal holds in all the created branches, π is returned as the solution plan; otherwise, a failure is returned. More information about the planning algorithm can be found in the references [12,14, 15].

Algorithm 1: negotiation planning algorithm

```
1.  Begin  NP (s, g,  π);
2.  Loop
3.    IF  goldsatisfied(s; g) THEN return  π
4.        if  k_w≠φ then
5.          PICK(α) : α ϵ  k_w
6.          BRANCH (s; α ; s1; s2)
7.          C := {NP(s1; φ; g); NP(s2;φ; g)}
8.          if failure  ϵ  C then return failure
9.          else return  π.C
10.    Action ←{A | precondsatisfied(A; s)}
11.    if Action = φ then return failure
12.    PICK(A) : A  ϵ Action
13.     s← Effect(A; s)
14.     π  ϵ  π.A
15.  end
```

Fig.5. shows a part of the picty's negotiation plan, generated to model the negotiation process between Picty and Mirry as described by the scenario outlined in Table 3. The initial state, as indicated in Figure 4, contains some predicates that represent planner agent's knowledge and beliefs about the conflicting situation. They are all some known fact in k_f database and according to them, the planner starts to generate a negotiation plan.

isagent(picty),isagent(mirry), has (picty,picture(p1)), has (picty, screwdriver(sd1)), has (picty, screw(s1)), has (picty, hammer (h1)), has(mirry,mirror (m1)), has (mirry, nail(n1)), has(mirry,glue(g1)), cando (picty,hangApicture), cando(mirry,hangmirror). believe (picty, imply([has (picty, hammer(h1)), has (picty, nail (n1)),has (picty,Chair(bc1)), has (picty,picture(p1))], do (picty, hangAPicture(picty, p1,h1, n1, bc1)))). believe (picty, imply ([has(mirry, mirror(m1)), has(mirry, screwdriver(sd1)), has(mirry, screw(s1))], do(mirry, hangAMirror(mirry, m1, sd1, s1)))). believe (mirry, imply([has(mirry, mirror(m1)), has(mirry, hammer (h1)), has(mirry, nail(n1))],do(mirry, hangAMirror(mirry, m1, h1, n1)))).believe(picty, imply(do(picty, giveResourceTo(picty, mirry, hammer(h1))), not(pictureHanging (p1))). believe(mirry, imply(do(mirry, giveResourceTo(mirry, picty, nail(n1))), not (mirrorHanging(m1)))). isgoal (mirry, mirrorHanging(m1)). isgoal (picty, pictureHanging(p1)).

Figuere.4. facts describing initial state of a negotitatin dialogue planning

Goal of this negotiation planning problem is *do (mirry, giveResourceTo (mirry, picty, nail (n1)))* that express a situation in that Mirry is persuaded to give nail (*n1*) to Picty. Using these information and actions defined in Table 4, planning algoirthm constructs a negotiation plan to solve conflict between Mirry and Picty, as can be seen in a part of a plan shown in Figure 5. According to this plan, at first Picty proposes to Mirry to give nail to him; since, Mirry's response to this proposal is unknown at the planning time, so a sensing action named *Sense-propos* can be use to create two branches in the plan one for acceptation and another for rejection made by Mirry. If Mirry accept the proposal, then the goal has been achieved , therwise Picty has convince Mirry to do so by expressing some justifications.

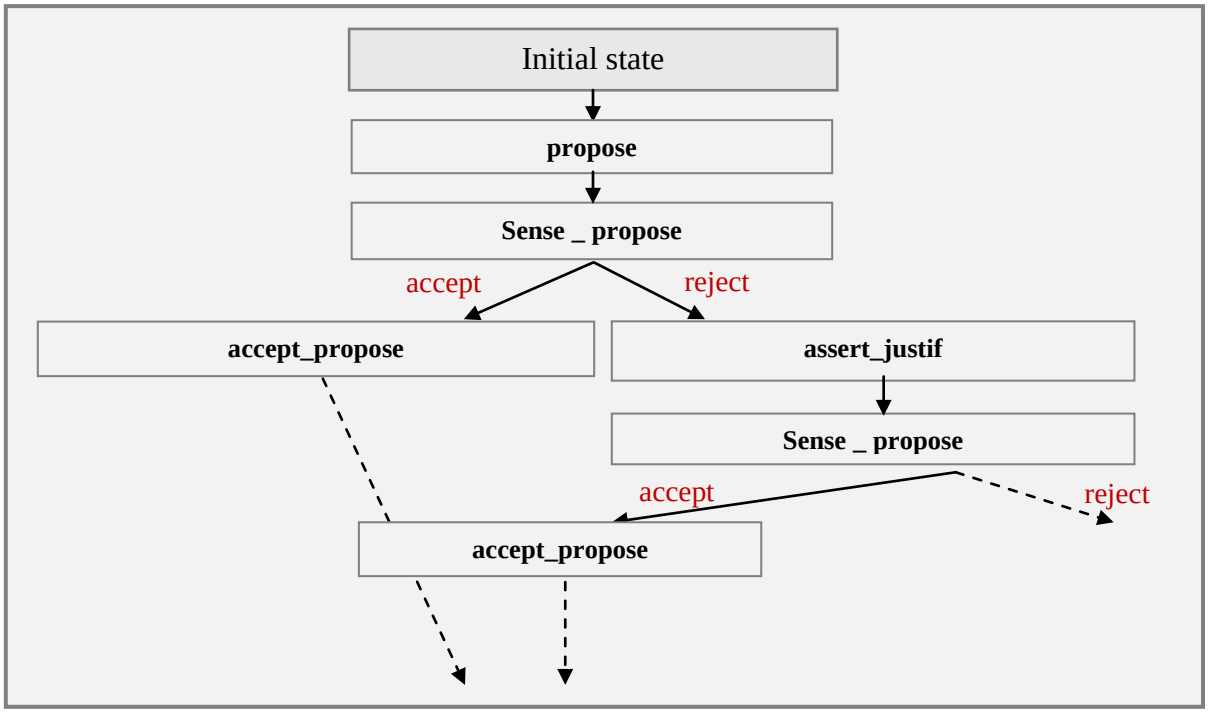

Figure.5. part of the picty negotiation plan

4. Conclusion

In this paper, we showed a negotiation planning approach that is able to take into account incomplete knowledge about the opponent's position in negotiation for conflict resolution by inserting some sensing action in the generated dialogue plan to sense the oppnent's response (such as accept/reject) to a proposal uttered by the agent planner. As such, our planning approach is more robust in compare to other existing approaches that only consider one course of actions when generating a negotiation plan.

As the future work we intend to use other PKS planning databases as well to enable us model facts with more than two values; as such we can generate negotiation plan for situations that opponent's reponse can be more than two states.

References

[1] F. Wu, S. Zilberstein, X. Chen, Online planning for multi-agent systems with bounded communication, Artificial Intelligence 175, (2011), 487–511.

[2] S.Kraus, K. Sycara,A. Evenchik , Reaching agreements through argumentation: a logical model and Implementation, Artif. Intell. 104 (1–2), (1998), 1–69.

[3] E. Oliva, Argumentation and Artifacts for Intelligent Multi-agent Systems , Doctorate in Electronics, Computer Science and Telecommunications, March (2008).
[4] S. Kraus, Strategic Negotiation in Multi-Agent Environments. MIT Press, Cambridge MA, USA, (2001).
[5] L. Amgoud, Y. Dimopoulos, and P. Moraitis. A unified and general framework for argumentation-based negotiation. AAMAS, ACM Press, (2006), 1018-1025.
[6] S. Ramchurn, N.R. Jennings, C. Sierra, Persuasive negotiation for autonomous agents: a rhetorical approach, in: C. Reed, F. Grasso, G. Carenini (Eds.), Proc, IJCAWorkshop on Computational Models of Natural Argument, Acapulco, Mexico, (2003), 9-17.
[7] P. Faratin. Automated Service Negotiation Between Autonomous Computational Agents. PhD thesis, University of London, Queen Mary and Westeld College, Department of Electronic Engineering, (2000).
[8] A. Monteserin, A. Amandi, Argumentation–based negotiation planning for autonomous Agents, Decision Support Systems 51, (2011), 532–548.
[9] I. Rahwan, S. D. Ramchurn, N. R. Jennings, P. McBurney, S. Parsons, and L. Sonenberg. Argumentation based negotiation. Knowledge Engineering Review (toappear), 18(4), (2003b), 343- 375.
[10] N. C. Karunatillake, Argumentation–Based Negotiation in a Social Context , A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy in the School of Electronics and Computer Science Intelligence, Agents, Multimedia Group, October (2006).
[11] N.C. Karunatillake, N.R. Jennings, I. Rahwan, P. McBurney, Dialogue Games that Agents Play within a Society, (2009).
[12] Ronald P. A. Petrick, F. Bacchus, A Knowledge-Based Approach to Planning with Incomplete Information and sensing, American Association for Artificial Intelligence, (2004).
[13] I. Rahwan, L. Sonenberg, P. McBurney, Bargaining and argument-based negotiation: some preliminary comparisons, in: I. Rahwan, P. Moraitis, C. Reed (Eds.), Argumentation in Multi-Agent Systems: Proc. 1st Inter. Workshop, ArgMAS'04: Expanded and Invited Contributions, LNAI, 3366, Springer-Verlag, Berlin, Germany, (2005), 176-191.
[14] A. Rahati, F. Kabanza. Automated Planning of Tutorial Dialogues. Department of Computer Science University of Sherbrookee Canada. amin.rahati, kabanza@usherbrooke.ca.
[15] Ronald P. A. Petrick. Representing an Agent' s Incomplete Knowledge for Planning, A thesis presented to the University of Waterloo in fulfilment of the thesis requirement for the degree of Master of Mathematics in Computer Science, 1998.