Agent utterances in the dialogue

No Author Given

No Institute Given

The dialogue model constructed so far allows the user to choose any utterance of the five modeled utterances at each speaking slot. For this reason, The agent has to be able de to respond consistently regardless to the utterance chosen by the user. In this report, I present my model of utterances responses that the agent can choose for each type of received utterance. This model take in account the current mental state when the user utterance was received and its perception of the relationship. We focus on the relation of dominance that takes three values: {Dominant, submissive, peer}.

In the following, I present the agent possible responses for each utterance.

1 State Preference (C, less, more)

A user can state a preference on values of a certain criterion *C*. As explained, the semantic of the utterance allows the speaker to express three different cases:

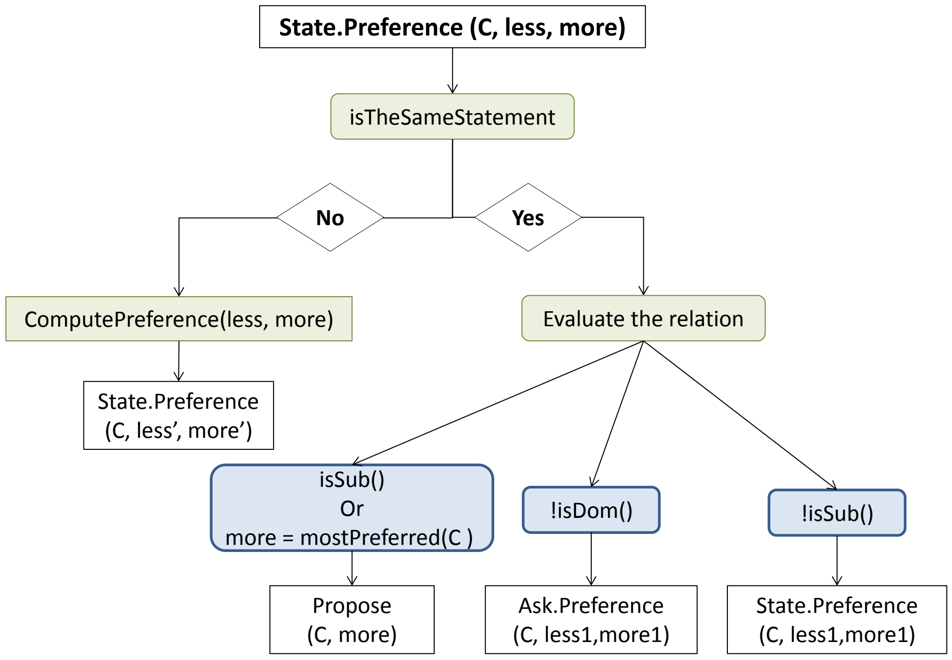
State.Preference (C, less, more)
isTheSameStatement
No
Yes
ComputePreference(less, more)
Evaluate the relation
State.Preference (C, less', more')
isSub() Or more = mostPreferred(C)
!isDom()
!isSub()
Propose (C, more)
Ask.Preference (C, less1,more1)
State.Preference (C, less1,more1)

Fig. 1: agent possible answers to a state Preference

– StatePreference(C, less, more) : I like ”less” less than ”more”
– StatePreference(C, *, more): I like the most More
– StatePreference(C, less, *): I like the least less.

The agent analyses the values of the received utterance. First, the agent checks that its last utterance is not a “StatePreference” about the expressed values, which is done with the method *isTheSameStatement*. The algorithm of this method is depicted bellow:

```
1: procedure isTheSameStatement($(less, more)$)
2:     $lastAgentUtt \leftarrow getLastAgentUtterance()$
3:     $lastUserUtt \leftarrow getLastUserUtterance()$
4:     if ($lastAgentUtt == null$) and ($lastUserUtt == null$) then return $false$
5:     if ($lastAgentUtt.getMore() == null$) then
6:         if $lastAgentUtt.getLess() == lastUserUtt.getLess()$ or $lastAgentUtt.getLess() == lastUserUtt.getMore()$ then return $true$
7:     if ($lastUserUtt.getMore() == null$) then
8:         if $lastUserUtt.getLess() == lastAgentUtt.getLess()$ or $lastUserUtt.getLess() == lastAgentUtt.getMore()$ then return $true$
9:     if ($lastAgentUtt.getLess() == null$) then
10:        if $lastAgentUtt.getMore() == lastUserUtt.getLess()$ or $lastAgentUtt.getMore() == lastUserUtt.getMore()$ then return $true$
11:    if ($lastUserUtt.getLess() == null$) then
12:        if $lastUserUtt.getMore() == lastAgentUtt.getLess()$ or $lastUserUtt.getMore() == lastAgentUtt.getMore()$ then return $true$
```

Fig. 2: Pseudocode for hybrid reactive HTN execution and recovery system.

to and depending on the perception of the relationship, the agent will choose an adequate answer. The possibles responses are depicted in Fig. 6, and explained in the following:

1. **StatePreference (C, less, more)**:The agent can react to the the stated values by observing its preferences on the stated criteria. This utterance is selected if the agent’s previous utterance doesn’t concern a statement about (less, more). The condition ”sameStatement” is computed with the following algorithm:
   This condition avoid the agent to fall in infinite loop. Depending on the values of the input preference, the agent will respond with one the the following values.
   – input: *(*, more)* → output: *(more, mostPreferred)* if *more* ≠ *mostPreffered*, else *(*, more)*.
   – input: *(less, *)* → output: *(leastPreferred,less)* if *less* ≠ *leastPreffered*, else *(less, *)*.
   – input: *(less, more)* → Output: *(less, more)* if(score(less) < score(more)), *else (more,less)*.

2. **StatePreference (C, less1, more1)**: The agent can state a preference about other values than expressed in the user utterance provided that the agent is not submissive. In the case of all the preferences related to the criterion *C* are already expressed (all the preferences are in OAS). The agent expresses a preference about another criterion *C1* such that $C \neq C1$.
3. **AskPreference(C, less1, more1)**: THis response is chosen in the case where the agent is submissive so he doesn't want to propose new values. Therefore, he uses the utterance ask to invite the user to choose values. In the same way as the previous responses, if the agent is aware of all the user preferences about the criterion *C*, he will ask the user about his preferences about another criterion *C1*.
4. **Propose(C, more)**: the choice of the propose utterance depends on the perception of the relationship of dominance :
   – if the agent is not submissive; then the value "more" of the expressed preference has to match the most preferred value of the agent preferences on the criterion *C* *(i.e more = mostPreferred)*. Thus, the agent constructs a proposal that takes as value its mostPreferred value of *C*.
   – If the agent is submissive, he prioritizes the user preferences. Thus, if the statement is of the type (StatePreference (*, more)). The agent proposes to choose the value "more" for *C*.

2 Ask Preference (C, less, more)

In the case of user asks the agent about a certain preference. The only possible response is to express its opinion about the asked preference. The content of the response is generated using a method that calculates the values from the input of the ask utterance. We distinguish the following cases, illustrated in Fig. 3:

1. If *(C, less, more)=(*,*)* thus *reactToAsk(C,less,more) = mostPreferred(C)*. The agent express its most preferred value of the criterion *C*.
   – **For example :**
   – U: what kind of cuisine do you prefer ?
   – A: I like Japanese cuisine. (which corresponds to the agent mostPreferred value for the criterion *Cuisine*).
2. If *(less=*)* and *(more* $\neq$ **)*, thus *reactToAsk(*,more)=(*,more)* if *(more=mostPreferred)*, else *reactToAsk(*,more)=(*,mostPreferred)*.
   – For example :
   – U: Do you like Chinese cuisine ?
   – A: I like Chinese less than Japanese cuisine. (which corresponds to the agent mostPreferred value for the criterion *Cuisine*).
3. In the same idea, if *(more=*)* and *(less* $\neq$ **)*, thus *reactToAsk(less,*)=(less,*)* if *(less=leastPreferred)*, else *reactToAsk(less,*)=(leastPreferred,*)*.

3 Propose (value)

When the agent receive a proposal, he can either accept it or reject this proposal. However, we model different ways to express the rejection of a proposal that is

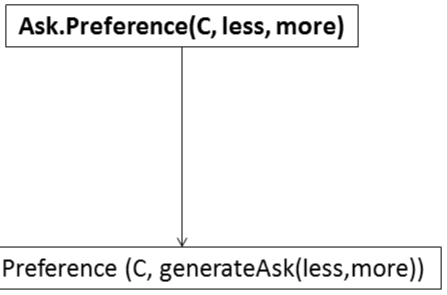
Ask.Preference(C, less, more)
State.Preference (C, generateAsk(less,more))

Fig. 3: agent possible answers to an ask Preference

influenced by the perception of the relationship. The value of proposal can be either a criterion or an option. (see Fig. 4)

1. **Accept(value)**: A proposal is accepted by the agent only if its score of preference is acceptable. To calculate the acceptance of a proposal, we modeled a simple method that checks if the score of the value of proposal is acceptable. In the case of a criterion proposal preference if the score of a criterion is positive then it is acceptable. While options are sorted by their utility rate, and an option is acceptable if its rank in the list is bellow the middle. Note that, when an option is accepted, the negotiation is closed.
2. **Reject(value)**: We suppose that a submissive person is not comfortable with expressing a clear reject. Therefore, we decide that only a non submissive agent can express a reject if the proposed values are not acceptable.
3. **StatePreference(C,value,*)**: a submissive agent can express an implicit reject using a statePreference. In the case of a criterion proposal, the proposed value is assigned to the less argument of the preference and the more is calculated from the preference base of the agent, such that $more \notin Rejected$. In the case of option, we extract the criterion that gets the worst score and we generate a preference in the same way than the criterion proposal.
4. **Propose(value 2)** : This utterance is defined to allow the agent to counter propose another value if the value proposed by the user is not acceptable. This utterance is allowed only for a non submissive agent such that $value2 \neq$

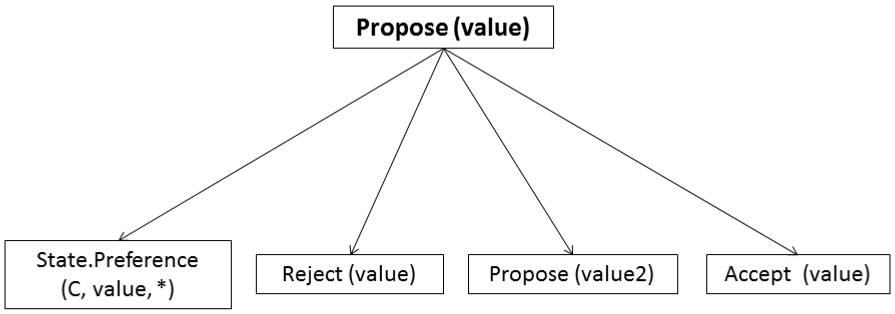

Fig. 4: agent possible answers to a propose utterance

value and *value*2 $\notin$ *Rejected*. This value can either be a criterion or an option.

4 Reject (value)

A user can reject a proposal that he doesn't like. Thus, the agent has to react in order to either propose other values or to collect more knowledge about the user preferences. However we defined an additional case for the dominant agent. (see Fig. 5)

1. **Propose(value 2)** : if the agent is not submissive, he can propose another proposal which can be either a criterion or an option. The proposed value concerns its most preferred value. The current most preferred value is calculated by taking into account the current state of negotiation (its ignores the rejected criteria and options).
2. **AskPreference(C,less,more)**: if the agent is submissive and receives a reject, he would gather more knowledge about the user preferences. Therefore, he'll ask the user about a preference that doesn't exist in other preferences.
3. **StatePreference(C,less,more)**: The agent continues the negotiation and express its preference concerning the current discussed criterion before the reject.

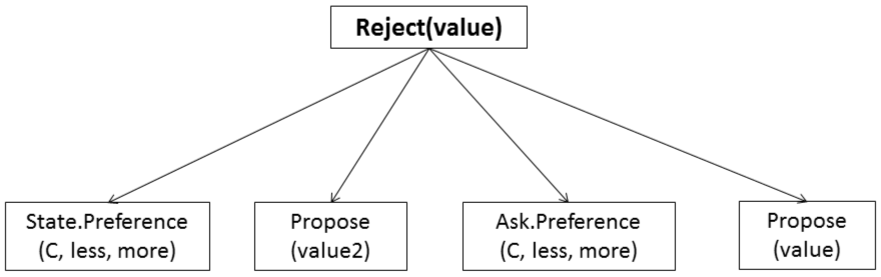
Reject(value)
State.Preference (C, less, more)
Propose (value2)
Ask.Preference (C, less, more)
Propose (value)

Fig. 5: agent possible answers to reject utterance

4. **Propose(value)**: When the agent is dominant and the rejected value is its most preferred value, we modeled a case where the agent will counter propose with the rejected proposal until the user accepts it.

5 Accept (value)

When a user accept proposal, the agent response will depends on the content of the utterance. If the value is an option, then the negotiation is over and the agent has to close the dialogue. Otherwise, the agent has to open a negotiation on the other criteria, or propose an option that contains the accepted criterion. In the following, we present the possible responses for an accept value where a value concerns a criterion.

1. **Propose(value 2)** : if the agent is dominant, he can open a negotiation on an other criterion, by proposing a value.
2. **AskPreference(C1,less,more)**: if the agent is submissive, he would open a new negotiation by asking the user his preferences about the criterion *C1*, where type(value) *not* = *C1*.
3. **StatePreference(C1,less,more)**: The agent continues the negotiation and express its preference concerning the new discussed criterion.

This propositions remain a first test and ameliorations will be proposed soon.

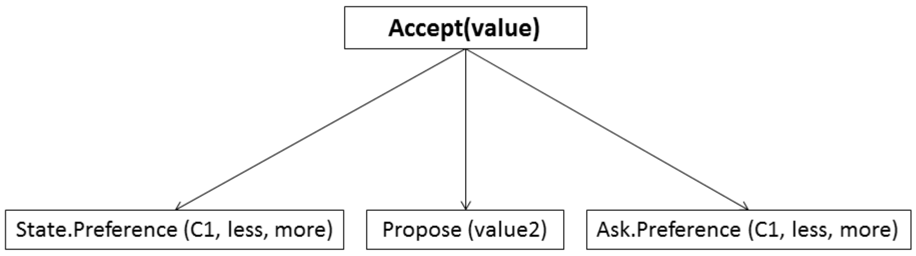
Accept(value)
State.Preference (C1, less, more)
Propose (value2)
Ask.Preference (C1, less, more)

Fig. 6: agent possible answers to a accept utterance