Towards ACL semantics based on commitments and penalties

Leila Amgoud and **Florence Dupin de Saint-Cyr**[1]

Abstract.
The importance of defining a standard framework for agent communication languages (ACL) with a simple, clear, and a verifiable *semantics* has been widely recognized.
This paper proposes a *logic-based* semantics which is *social* in nature. The basic idea is to associate with each speech act a meaning in terms of the *commitment* induced by that speech act, and the *penalty* to be paid in case that commitment is violated. A *violation criterion* based on the existence of arguments is then defined per speech act. Moreover, we show that the proposed semantics satisfies some key properties that ensure the approach is well-founded. The logical setting makes the semantics verifiable.

1 Introduction

When building multi-agent systems, we take for granted the fact that agents which make up the system will need to engage in the different types of dialogues identified by Walton and Krabbe in [11], using a communication language (ACL). The definition of an ACL from the syntactic point of view (the different *speech acts*[2] that agents can perform during a dialogue) poses no problems. The situation is different when semantics is taken into account. Given that agents in a multi-agent system may be independently designed by different programmers, a clear understanding of semantics is essential. Indeed, any speech act should have a *unique interpretation*. Moreover, it should be *verifiable*, i.e. it should be possible to check whether a system conforms to a particular ACL or not [13]. Although a number of agent communication languages have been developed, obtaining a suitable formal semantics for ACLs which satisfies the above objective remains one of the greatest challenges of multi-agent theory.
Our aim is to define a semantics which prevents the shortcomings of existing approaches while keeping their benefits. The basic idea behind our semantics is that each speech act has a goal. For instance, behind a question, one expects an answer. Hence, during a dialogue, as soon as a speech act is uttered, a kind of *commitment* for achieving its goal is created. In the case of a question, by uttering such a speech act, a commitment for answering is created (here for the hearer). Note that this does not mean at all that the hearer should necessarily answer.
The new semantics is grounded on a computational logic framework, thus allowing automatic verification of compliance by means of proof procedures. More precisely, the semantics associates with each speech act a meaning in terms of the *commitment* induced by it, and a *penalty* to be paid in case that commitment is violated. A *violation criterion* based on the existence of arguments is then defined per speech act.
From a *syntactic* point of view, utterances are stored in *commitment stores* as in [7]. Each agent is supposed to be equipped with a commitment store visible to all agents.
Note that the aim of this paper is not to propose a dialogue protocol whose role is to ensure coherent dialogues, etc. However, the protocol of a dialogue system that uses our semantics should at least enforce agents to minimize their violation costs. The definition of such protocol is beyond the scope of this paper.
This paper is organized as follows: section 2 introduces the logical language used throughout the paper. Section 3 defines the new semantics. Some logical properties are presented in section 4, and an example is given in section 5. Section 6 compares our semantics with existing approaches.

2 The logical language

Throughout the paper, we consider a *propositional language* $\mathcal{L}$. $\vdash$ denotes classical inference and $\equiv$ logical equivalence. A knowledge base Σ is a set of formulas of $\mathcal{L}$. *Arguments* can be built from any knowledge base Σ:

Definition 1 (Argument) *An* argument *is a pair* (S, c) *where* c *is a formula of* $\mathcal{L}$ *and* $S \subseteq \Sigma$ *such that:*

1. *S is consistent,*
2. *$S \vdash c$,*
3. *S is minimal for set inclusion among the sets satisfying 1) and 2).*

S is the support *of the argument and c its* conclusion. $\mathtt{Arg}(\Sigma)$ *is the set of all the arguments that can be built from Σ.*

Given that a knowledge base Σ may be inconsistent, arguments may be conflicting too. In what follows we will use the "undercut" relation which is the most suitable in our case.

Definition 2 (Undercut) *Let* $A_1 = (H_1, c_1)$, $A_2 = (H_2, c_2) \in \mathtt{Arg}(\Sigma)$. A_1 undercuts A_2 *if* $\exists\, h'_2 \in H_2$ *such that* $c_1 \equiv \neg\, h'_2$.

Let $\mathcal{A} = \{ag_1, \ldots, ag_n\}$ be the set of *agents* involved in the system. Each agent is assumed to have a *role* allowing it to have the control over a subset of formulas in $\mathcal{L}$ ($\mathtt{Role}$: $\mathcal{A} \longmapsto 2^{\mathcal{L}}$). Roles are supposed to be visible to all agents. Thus they are not private.

A communication language is based on a set of *speech acts*. Let $\mathcal{S}$ denote that set. From $\mathcal{S}$ and $\mathcal{L}$, different moves can be built.

Definition 3 (Move) *If* $a \in \mathcal{S}$ *and either* $x \in \mathcal{L}$ *with* $x \not\vdash \bot$, *or* $x \in Arg(\mathcal{L})$ *then* $a{:}x$ *is a* move.

---

[1] IRIT - CNRS 118, route de Narbonne 31062 Toulouse Cedex 9, France amgoud, bannay@irit.fr

[2] The speech acts are also called *illocutionary* acts or *performatives*.

For a given move $a : x$, the function `Act` *returns the speech act* ($\texttt{Act}(a\!:\!x) = a$)*, and the function* `Content` *returns the content of the move* ($\texttt{Content}(a : x) = x$)*. Let $\mathcal{M}$ denote the set of all the possible moves that can be built from $\mathcal{S}$ and $\mathcal{L}$.*

For example, the move $\texttt{Question:}x$ (with x meaning that the sky is blue) is uttered to ask whether the sky is blue or not.

3 Semantics

The basic idea behind our semantics is to associate with each speech act a meaning in terms of the *commitment* induced by that speech act, and a *penalty* to be paid in case that commitment is violated. For each speech act, we define a *criterion* precising when the corresponding commitment is violated. These criteria are all based on the existence of arguments. The various moves uttered during a dialogue are stored in *commitment stores* which are visible to all agents.

3.1 Speech acts

We consider the following set of basic speech acts that are used in the literature for modeling the different types of dialogues identified by Walton and Krabbe in [11]:

$$\mathcal{S} = \{\texttt{Assert}, \texttt{Argue}, \texttt{Declare}, \texttt{Question}, \texttt{Request}, \texttt{Challenge}, \texttt{Promise}\}.$$

- `Assert` and `Argue` allow an agent to inform another agent about the state of the world. These acts are *assertive* according to the classification of Searle [9]. $\texttt{Assert:}x$ and $\texttt{Argue:}x$ differ in the syntactic form of x. In the case of $\texttt{Assert:}x$, x is a proposition ($x \in \mathcal{L}$) like "the weather is beautiful" or "it is my intention to hang a mirror". However, in $\texttt{Argue:}x$, x is an argument ($x \in Arg(\mathcal{L})$).
- $\texttt{Declare:}x$, with $x \in \mathcal{L}$, is a move that brings about a state of affairs that makes its content x true. It is a *declarative* act like "the auction is open" or "John and Mary are husband and wife".
- `Question`, `Request` and `Challenge` are *directive* acts that incite the agent who receives them to give an answer. The moves $\texttt{Question:}x$, $\texttt{Request:}x$ and $\texttt{Challenge:}x$ are all about the same kind of information, x is a *proposition* in the three cases ($x \in \mathcal{L}$). In $\texttt{Question:}x$, the agent asks for the truth value of x. In $\texttt{Request:}x$, the agent asks another agent to alter the value of x to true. $\texttt{Request:}x$ has then a more imperative character than $\texttt{Question:}x$ which does not ask the other agent to act on the world but only to give some information. A `Request` is used when an agent cannot, or prefers not to, achieve one of its goals alone. For instance, if ag_2 utters $\texttt{Request} : ag_2\_is\_paid$ then it means that ag_2 asks to be paid. By doing a $\texttt{Challenge:}x$ move an agent asks for a reason/argument in favor of the conclusion x.
- A `Promise` move commits the agent to some future course of action. It is a *commissive* act according to Searle [9]. The expression $\texttt{Promise:}x$ means that the agent is committed to make x true in the future, with $x \in \mathcal{L}$. For example, if ag_1 utters $\texttt{Promise:}ag_2\_is\_paid$, it means that ag_1 commits itself to ensure that ag_2 is paid in the future.
- In addition to the above speech acts, we will consider another act called `Retract` which does not belong to the different categories of speech acts defined by Searle. It can be seen as a meta-level act allowing agents to withdraw commitments already made. Allowing such a move makes it possible for agents to have a kind of non-monotonic behavior (i.e., to change their points of view, to revise their beliefs, etc.) without being sanctioned. Syntactically, $\texttt{Retract:}m$ is a move with $m \in \mathcal{M}$ being itself a possible move.

3.2 Commitments

In the scientific literature, one can find proposals where the semantics of an ACL is defined in terms of commitments. Examples of these are given by Colombetti [2] and Singh [10]. Colombetti and Singh argued that agents are social entities, involved in social interactions, so they are committed to what they say. In recent inter-agent communication approaches, the notions of dialogue games and (social) commitments are central. One rather influential dialogue game is DC, proposed by Hamblin [5]. DC associates with each player a *commitment store*, which holds its commitments during the dialogue. Commitments here are pieces of information given by players during the dialogue. Then, there are rules which define how commitment stores are updated. Take for instance assertion, it puts a propositional statement in the speaker's commitment store. What this basically means is that, when challenged, the speaker will have to justify his claim. But this does not presuppose that the challenge will come at the next turn in the dialogue.

We adopt this representation of commitments. Note that in this paper we are not interested in modeling the reasoning of the agent, we only consider what is said by each agent. The idea is to provide a semantics without worrying about agents mental states. Each agent is supposed to be equipped with a commitment store, accessible to all agents, that will contain the utterances it makes during the dialogue. A commitment store keeps tracks of two kinds of speech acts:

- Speech acts made by the agent itself such as assertions, promises and declarations.
- Speech acts received from other agents, such as requests, challenges and questions. For instance if an agent ag_i makes a request r to another agent ag_j, the request (r) is stored in the commitment store of ag_j. Hence, ag_j is said *committed* to answer to it.

Definition 4 (Commitment store) *A* commitment store CS_i *associated with an agent ag_i is a pair $CS_i = \langle A_i, O_i\rangle$ with:*
$A_i \subseteq \{m \in \mathcal{M} | \texttt{Act}(m) \in \{\texttt{Assert}, \texttt{Argue}, \texttt{Declare}, \texttt{Promise}\}\}$.
$O_i \subseteq \{m \in \mathcal{M} \mid \texttt{Act}(m) \in \{\texttt{Question}, \texttt{Request}, \texttt{Challenge}\}\}$.

A dialogue evolves from one step to another as soon as a move is uttered. In what follows, CS_i^s denotes the commitment store of agent i at step s. A commitment store is supposed to be *empty* at the beginning of a dialogue (i.e., at step 0). Hence, for all agent ag_i, $CS_i^0 = \emptyset$. Given a set X of moves, X^i denotes the moves of X that are uttered from step 0 to step i. Let us now introduce two functions `PROP` and $\texttt{PROP}_P$ that return sets of formulas as follows:

Definition 5 *Let $X \subseteq \mathcal{M}$.*

- $\texttt{PROP}(X)$ *is defined recursively by:*
  $$\texttt{PROP}(X^0) = \emptyset$$
  $$\texttt{PROP}(X^s) = \begin{cases} \texttt{PROP}(X^{s-1}) \cup \{x\} & \textit{if } m = \texttt{Assert:}x \\ \texttt{PROP}(X^{s-1}) \cup S & \textit{if } m = \texttt{Argue:}(S, c) \\ \texttt{PROP}(X^{s-1}) \diamond x & \textit{if } m = \texttt{Declare:}x \\ \texttt{PROP}(X^{s-1}) & \textit{else} \end{cases}$$
  where m is the move uttered at step s in X^s and $\diamond$ is an update operator described below.
- $\texttt{PROP}_P(X) = \{x \in \mathcal{L}$ *such that* $\exists \texttt{Promise:}x \in X\}$.

The above definition computes the set of formulas that represent the state of the world (according to what has been uttered during the dialogue). Note that Questions, Challenges and Requests are not considered in the definition because they don't describe the state of the world. Formulas that appear in assertions and arguments are directly considered. However, things are different with the formulas related to a move `Declare`. Indeed, by definition, after `Declare`:x the world evolves in such a way that x becomes true. Consequently, one has to update the whole set of propositions previously uttered. For that purpose, an update operator [12], denoted by $\diamond$, is needed. Several update operators have been introduced in the literature. The choice of the precise one to be used in our semantics is beyond the scope of this paper.

3.3 The notion of penalty

As said before, from each move a commitment is induced. It is natural to associate with each commitment a penalty that sanctions the agent when this commitment is violated. For the sake of simplicity, the penalty is supposed to depend on the speech act and not on the content of the move. Hence, each speech act in $\mathcal{S}$ is supposed to have a *cost* which is an integer: $\texttt{Cost} : \mathcal{S} \longmapsto \mathbb{N}$. Different speech acts may have different values. This captures the idea that some speech acts are more important than others. For instance, violating a promise may be more costly than not answering a question. With each commitment store is associated a penalty as follows:

Definition 6 (Penalty) *Let $CS_i = \langle A_i, O_i \rangle$ be a commitment store, and $X \subseteq A_i \cup O_i$. The* penalty *associated with X w.r.t. CS_i is*

$$c(X) = \sum_{m \in X} \texttt{Penalty}(m)$$

where $\texttt{Penalty}(m) = \texttt{Cost}(\texttt{Act}(m))$ *if the commitment m is violated in A_i and* $\texttt{Penalty}(m)$ *= 0 otherwise.*

Since a commitment store is empty at the beginning of a dialogue, its initial penalty is equal to 0. Moreover, at any step, the penalty of a given commitment store can be computed in a very simple way as shown in the next section.

3.4 Violation criteria

As shown before, a penalty is to be paid if a commitment is violated. This section presents in details when commitments induced from each speech act of $\mathcal{S}$ are violated. Subsequently, we suppose that the agent ag_i utters the move to the agent ag_j.

1. `Assert`:x
During a dialogue, an agent can assert that a given propositional formula is true. Then, this agent is not allowed to contradict itself during all the dialogue otherwise it will have to pay a penalty (except if it retracts that proposition). Indeed, a move `assert`:x is *violated* if the A_i part of the commitment store of the agent ag_i makes it possible to find an argument with a conclusion $\neg x$. Formally:

Definition 7 *A move* `Assert`:x *is* violated *iff*

$$\exists (S, \neg x) \in \texttt{Arg}(\texttt{PROP}(A_i)).$$

In order to avoid any form of wishful thinking, in the above definition, the promises are not taken into account when checking the violation of an assert move, even if they are stored in the A_i part of the commitment store.

2. `Argue`:x
During a dialogue, an agent can provide an argument x in favor of some conclusion. Then, this agent is not allowed to contradict itself in the sense that it cannot produce an undercutter against x.

Definition 8 *A move* `Argue`:x *is* violated *iff*

$$\exists (S', y) \in \texttt{Arg}(\texttt{PROP}(A_i)) \textit{ such that } (S', y) \textit{ undercuts}^{3}\ x.$$

As for assert moves and for the same reason, promises are not taken into account when looking for counter arguments.

3. `Declare`:x
During a dialogue, an agent can modify the state of a certain proposition x by declaring it true. The move `Declare`:x commits the honesty of the agent which carries it out in the sense that the agent should be empowered to modify the value of x. This capacity is defined by the role of the agent. For instance, it is not allowed for a simple citizen to marry people. Moreover, an agent can really modify this value only if there is no argument against performing that action. Formally:

Definition 9 *A move* `Declare`:x *is* violated *iff*

$$x \notin \texttt{Role}(ag_i) \textit{ or } \exists (S, \neg y) \in \texttt{Arg}(\texttt{Prop}(A_i)) \\ \textit{with } y \in \texttt{Precond}(x)$$

where $\texttt{Precond} : \mathcal{L} \rightarrow 2^{\mathcal{L}}$ *is a function that gives for any formula φ the pre-conditions for setting φ to true and that verifies:* $\texttt{Precond}(\bot) = \{\bot\}$ *and* $\texttt{Precond}(\top) = \emptyset$

The definition of `Precond` may come from law, it is supposed to be furnished (its definition is beyond the scope of this paper). For example, in order to open an auction, one should check whether the buyers are present. If a formula can never be set to true then the function `Precond` returns $\{\bot\}$. When, there is no pre-condition for setting the formula to true, the function returns $\emptyset$.

4. `Question`:x
During a dialogue, an agent may receive questions from other agents to which it should answer either positively or negatively. The absence of any argument in favor of x or $\neg x$ in the part A_j of the commitment store of the agent that receives the move means that the agent has not given any answer.

Definition 10 *A move* `Question`:x *is* violated *iff*

- $\nexists$ *(S, x)* $\in \texttt{Arg}(\texttt{PROP}(A_j))$ *and*
- $\nexists$ *$(S, \neg x)$* $\in \texttt{Arg}(\texttt{PROP}(A_j))$.

Again, promises are not considered when building arguments. Note that we check the existence of an argument in favor of x or $\neg x$ instead of just the existence of a proposition equivalent to x or to $\neg x$ in A_j. The reason is that the question can be answered implicitly via other assertions of the agent. In this setting, it is not possible to answer "I don't know" to a question. But, this could be easily handdled by introducing the speech act `Desinform`.

5. `Request`:x
An agent should give a positive or a negative answer to any request it receives from other agents.

---

[3] See Definition 2 in Section 2.

Definition 11 *A move* Request:x *is* violated *iff*

- $\nexists\ (S, x) \in \texttt{Arg}(\texttt{PROP}(A_j) \cup \texttt{PROP}_P(A_j))$ *and*
- $\nexists\ (S', \neg x) \in \texttt{Arg}(\texttt{PROP}(A_j) \cup \texttt{PROP}_P(A_j))$.

Note that to check whether a request is violated or not, we look for an argument in favor of x in both $\texttt{PROP}(A_j)$ and $\texttt{PROP}_P(A_j)$. The reason is that a request can get an answer in two ways: either because of a promise ensuring that in the future the requested proposition will be set to true or to false, or because it is already stated (either by declarations or assertions) to true or false.

6\. Challenge:x
Agents should provide arguments for any challenged proposition.

Definition 12 *A move* Challenge:x *is* violated *iff*

$$\nexists\ (S, x) \in \texttt{Arg}(\texttt{PROP}(A_j)) \text{ with } S \not\equiv x.$$

Let us take the example of agent which asserts x, after which the other agent makes a challenge on x. It is clear that the argument $(\{x\}, x)$ can be built from $\texttt{Arg}(\texttt{PROP}(A_j))$, however this is not an answer to the challenge. Thus in order to avoid such problem, the above definition requires that the argument presented after a challenge should be different from x.

7\. Promise:x
During a dialogue, an agent can make promises to other agents. This agent should pay a penalty in case it does not respect this promise. This can be checked on the part A_i of its commitment store. Indeed, if an argument in favor of proposition x can be built then the promise is honored otherwise it is considered violated.

Definition 13 *A move* Promise:x *is* violated *iff*

$$\nexists (S, x) \in \texttt{Arg}(\texttt{PROP}(A_i)).$$

8\. Retract:m
Agents may decide to retract some previously uttered moves. The advantage of this move is to allow them to revise their beliefs without being sanctioned. The Retract move is different from the others since it is never violated, thus $\texttt{Penalty}(\texttt{Retract}{:}m)$ = 0. Moreover, after such a move the commitment store is updated as follows:

Definition 14 *Let CS_i^s be the commitment store of an agent ag_i at step s. A move* $\texttt{Retract}(m)$ *at step $s+1$ has the following effect:*

$$CS_i^{s+1} = CS_i^s \setminus \{m\}$$

Note that retracting a move that has not been uttered has no effect.

4 Logical properties

The aim of this section is to show that the proposed semantics satisfies some key and desirable properties. The first property ensures that the semantics sanctions only bad behaviors of agents, and that any bad behavior is sanctioned.

Proposition 1

- *If $c(CS_i) > 0$, then $\exists\ m \in CS_i$ s.t m is violated.*
- *If $\exists\ m \in CS_i$ s.t m is violated, then $c(CS_i) > 0$.*

Another important result is the fact that if the total penalty of part A_i is null then all the stated information is consistent.

Proposition 2 *(Consistency) If $\sum_{m \in A_i} \texttt{Penalty}(m)$ = 0, then $\texttt{PROP}(A_i)$ is consistent.*

In [6], it has been shown that a propositional formula may be useful for explaining another formula in a given context. This property is called *novelty*. In what follows, we give its definition in terms of arguments.

Definition 15 *(Novelty) Let φ, ϕ be two propositional formulas, and Σ a set of formulas.*

- *φ is* new *for ϕ w.r.t. Σ iff:*
  - $\exists\ \langle S, \phi\rangle \in \texttt{Arg}(\Sigma \cup \varphi)$ *and* $\langle S, \phi\rangle \notin \texttt{Arg}(\Sigma)$*, or*
  - $\exists\ \langle S, \neg\phi\rangle \in \texttt{Arg}(\Sigma \cup \varphi)$ *and* $\langle S, \neg\phi\rangle \notin \texttt{Arg}(\Sigma)$
- *φ is said to be* independent *from ϕ w.r.t. Σ otherwise.*

We can show that if two formulas are independent w.r.t. the formulas of a commitment store, then the penalty of two moves conveying these formulas is decomposable. Formally:

Proposition 3 (Independence) *Let CS_i = $<A_i, O_i>$ be a commitment store, and let $m, m' \in \mathcal{M}$. If $Content(m)$ is independent from $Content(m')$ w.r.t. $\texttt{PROP}(A_i) \cup \texttt{PROP}_P(A_i)$, then*

$$c(\{m, m'\}) = c(\{m\}) + c(\{m'\})$$

5 Example

Let us study the following dialogue between two agents ag_1 and ag_2:

$ag_2 \rightarrow ag_1$: Do you think that Newspapers can publish (pub) the information X.

A_1	O_1
$\emptyset$	Question:pub

$c(CS_1)$ = Cost(Question)

$ag_1 \rightarrow ag_2$: No.

A_1	O_1
Assert:$\neg pub$	Question:pub

$c(CS_1)$ = 0

$ag_2 \rightarrow ag_1$: why?

A_1	O_1
Assert:$\neg pub$	Question:pub Challenge:$\neg pub$

$c(CS_1)$ = Cost(Challenge)

$ag_1 \rightarrow ag_2$: Because X concerns the private life of A (pri) and A does not agree to publish it (agr).

A_1	O_1
Assert:$\neg pub$ Argue:$(\{pri, \neg agr, pri \wedge \neg agr \rightarrow \neg pub\}, \neg pub)$	Question:pub Challenge:$\neg pub$

$c(CS_1)$ = 0

$ag_2 \rightarrow ag_1$: But A is a minister (min) and information about ministers are public.

A_2	O_2
Argue:$(\{min, min \rightarrow \neg pri\}, \neg pri)$	$\emptyset$

$c(CS_2)$ = 0

$ag_1 \rightarrow ag_2$: Yes, you are right.

A_1	O_1
Assert:$\neg pub$ Argue:$(\{pri, \neg agr, pri \wedge \neg agr \rightarrow \neg pub\}, \neg pub)$ Argue:$(\{min, min \rightarrow \neg pri\}, \neg pri)$	Question:pub Challenge:$\neg pub$

$c(CS_1)$ = 2 × Cost(Argue)

In the above example, the agent ag_1 answers the question and the challenge it received, thus there is no penalty to pay for those moves. However, this agent has presented an argument a = $<\{pri, \neg agr, pri \wedge \neg agr \rightarrow \neg pub\}, \neg pub>$, and accepted its undercutter b = $<\{min, min \rightarrow \neg pri\}, \neg pri>$. Consequently, the set $\texttt{PROP}(A_i)$ is inconsistent, and this makes it possible to even construct an undercutter c = $<\{pri, min \rightarrow \neg pri\}, \neg min>$ for the argument b. The agent has then to pay twice the cost of an Argue move. Note, however that from $\texttt{PROP}(A_i)$ it is not possible to construct an argument whose conclusion is $\neg pub$. This means that the agent is still coherent w.r.t. its assertion ($\neg pub$). Thus, there is no cost to pay for the assert move.

6 Related work

The first standard agent communication languages are KQML [3] and FIPA-ACL [4]. Both languages have been given a mentalistic semantics. The semantics is based on a notion of speech act close to the concept of illocutionary act as developed in speech act theory [9]. Such semantics assumes, more or less explicitly, some underlying hypothesis in particular, that agents are sincere and cooperative. While this may be well fitted for some special cases of interactions, it is obvious that negotiation dialogues are not cooperative. Another more important limitation of this approach is the fact that it is not verifiable since it is based on agents mental states. Our semantics does not refer at all to the mental states of the agents. Moreover, it treats another speech act, namely `Argue`, which allows agents to exchange arguments.

In the second approach, called *social* and developed in [2, 10], primacy is given to interactions among agents. The semantics is based on social *commitments* brought about by performing a speech act. For example, by affirming a data, an agent commits on the truth of that data. After a promise, the agent is committed carrying it out. There are several weaknesses of this approach and we summarize them in the three following points: 1) The definition of commitments complicates the agent architecture in the sense that it needs an ad hoc apparatus. Commitments are introduced especially for modeling communication. Thus agents should reason not only on their beliefs, etc, but also on commitments. In our approach, we didn't introduce any new language to treat commitments. 2) The level at which communication is treated is very abstract, and there is a considerable gap to fill in order to bring the model down to the level of implementation. However, the semantics presented in this paper can be implemented easily. 3) The concept of commitment is ambiguous and its semantics is not clear. According to the speech act, the semantics of the commitment differs. For example, by affirming a data, an agent commits on the truth of that data. The meaning of the commitment here is not clear. It may be that the agent can justify the data or can defend it against any attack, or that the agent is sincere. In our approach, the semantics of a commitment is very intuitive, unique and simple. Penalties are computed in a very simple way and at any time during a dialogue.

The last approach developed by Pitt and Mamdani in [8] and Alberti et al. in [1] is based on the notion of protocol. A protocol defines what sequences of moves are conventionally expected in a dialogue. The meaning of a speech act equates to the set of possible following answers. However, protocols are often technically finite state machines. This turns out to be too rigid in several circumstances. Current research aims at defining flexible protocols, which rely more on the *state of the dialogue*, and less on dialogue history. This state of dialogue is captured by the notion of commitment.

7 Conclusion and perspectives

This paper has introduced a new simple and verifiable ACL semantics. The interpretation of each speech act equates to the penalty to be paid in case the commitment induced by that speech act is violated. In this semantics, a violation criterion is given for each considered speech act. Note that in order to add a new speech act, one needs simply to define a new violation criterion associated with it. This semantics is based on propositional logic, and the violation criteria amount to compute arguments.

An extension of this work to first order logic is under study. Another interesting extension would be to handle explicitly time in order to be able to deal with deadlines for instance.

The notion of penalty may play a key role in defining agent's *reputation* and *trust* degrees. It is clear that an agent that pays a lot of penalties during dialogues may lose its credibility, and will no longer be trusted. Examining more deeply penalties can help to figure out agents profiles: cooperative agent, consistent agent, thoughtful agent (i.e., agent which respects its promises)...

Another possible refinement consists of introducing granularity in the definition of the function `Cost`. The basic idea is to take into account the content of moves when defining their costs. This captures the idea that, for instance, some asserted propositions are more important than others. For example, affirming that the weather is beautiful can be less important than affirming that the president is dead.

Our semantics satisfies interesting properties that show its well-foundedness. It also offers other advantages regarding dialogue protocols. For instance, one does not need to specify the different moves allowed after each move in the protocol itself. Agents only need to minimize the penalty to pay at the end of the dialogue. This give birth to very flexible protocols.

REFERENCES

[1] M. Alberti, A. Ciampolini, M. Gavanelli, E. Lamma, P. Mello, and P. Torroni, 'A social ACL semantics by deontic constraints', in *3rd Int. Central and Eastern European Conference on Multi-Agent Systems CEEMAS'2003, volume 2691 of LNAI*, pp. 204–213, (2003).

[2] M. Colombetti, 'A commitment-based approach to agent speech acts and conversations', in *Proceedings of the Workshop on Agent Languages and Conversation Policies. I4th International Conference on Autonomous Agents*, pp. 21–29, (2000).

[3] T. Finin, Y. Labrou, and J. Mayfield, 'Kqml as an agent communication language', in *J. Bradshaw, ed. Softaware agents, MIT Press, Cambridge.*, (1995).

[4] FIPA, 'Agent communication language', in *FIPA 97 Specification, Foundation for Intelligent Physical Agents.*, (1997).

[5] C. L. Hamblin, *Fallacies*, Methuen, London, UK, 1970.

[6] J. Lang, P. Liberatore, and P. Marquis, 'Conditional independence in propositional logic', *Artificial Intelligence*, **Volume 141(1-2)**, 79–121, (2002).

[7] J. MacKenzie, 'Question-begging in non-cumulative systems', *Journal of philosophical logic*, **8**, 117–133, (1979).

[8] J. Pitt and A. Mamdani, 'A protocol based semantics for an agent communication language', in *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99)*, pp. 486–491, (1999).

[9] J. R. Searle, 'Speech acts', in *Cambridge University Press*, (1969).

[10] M. P. Singh, 'A social semantics for agent communication languages', in *IJCAI'99 Workshop on Agent Communication Languages*, pp. 75–88, (1999).

[11] D. N. Walton and E. C. W. Krabbe, *Commitment in Dialogue: Basic Concepts of Interpersonal Reasoning*, State University of New York Press, Albany, NY, 1995.

[12] M. Winslett, *Updating Logical Databases*, Cambridge University Press, Cambridge, UK, 1990.

[13] M. Wooldridge, 'Reasoning about rational agents', in *Mitt Press*, (2000).