SOCIETY FOR SCIENCE AND EDUCATION
UNITED KINGDOM

TMLAI TRANSACTIONS ON MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE

Volume 5 No. 3
ISSN 2054-7390

Dialogue Based Decision Making in Online Trading

[1,2]Wei Bai, [2]Emmanuel M.Tadjouddine, [1]Terry R.Payne, [2]Gangmin Li
[1]Department of Computer Science, University of Liverpool, England, UK;
[2]Department of Computer Science and Software Engineering, Xi'an Jiaotong-Liverpool University, SIP, SuZhou, China;
Wei.bai07@student.xjtlu.edu.cn; T.R.Payne@liverpool.ac.uk; Emmanuel.Tadjouddine@xjtlu.edu.cn; Gangmin.Li@xjtlu.edu.cn

ABSTRACT

Software agents, acting on behalf of humans, have been identified as an important solution for future electronic markets. Such agents can make their own decisions based on given prior preferences and the market environment. These preferences can be described using Web Ontology Languages (OWL), while the market mechanism can be represented in a machine-understandable way by utilizing the technique of Semantic Web Services (SWS). Besides, SWS enables agents to automatically discover, select, compose and invoke services. To extend the dependability and interactivity of SWS, we have utilized dialogue games and the Proof-Carrying Code (PCC) to enable buyers to interact with sellers, so that desirable properties for an online auction market can be automatically certified. Our decision-making framework combines formal proofs with informal evidence collected by web services in a dialogue game between a seller and a buyer. We have implemented our approach and experimental results have demonstrated the feasibility as well as the validity of this framework as an enabler for a buyer agent to enter or not an online auction.

***Keywords*:** Dialogue games; Online auction; Semantic Web Services; Software agent; Decision making.

1 Introduction

We consider E-commerce scenarios wherein software agents can buy or sell goods on behalf of their owners. To enable software agents to participate in such online trading, the trading mechanism should be presented in a machine understandable way. The Semantic Web provides an approach to enable agents to read and interpret a trading mechanism as an online service for which static and dynamic information can be explicitly described using ontology languages. For example, the Web Ontology Language (OWL) [22], which is based on description logic, can be used to express classes and relationships among them. The Semantic Markup for Web Services (OWL-S) [17], which is focused on the process description of a service, can be used to describe the procedures of the trading mechanism. However, the message exchange in the architecture of Semantic Web Services is restricted as a client-server or request-response pattern. To extend the interactivity of the Semantic Web Services, argumentation is introduced to support message exchange via dialogues. By using dialogue games, agents can assert, challenge and justify their arguments according to their knowledge [28].

DOI: 10.14738/tmlai.53.3390
Publication Date: 2nd July, 2017
URL: http://dx.doi.org/10.14738/tmlai.53.3390

Dialogue games are rule-governed interactions among software agents [20], wherein each agent presents its ideas by making "moves" based on a set of rules. Since the common agent communication languages, such as FIPA ACL [11], lack certain locutions to express justifications for statements, additional locutions are proposed to extend the FIPA ACL, so that argumentation can be supported in a dialogue [19]. Argumentation via dialogue games permit agents to carry out various types of interactions, such as information-seeking, inquiry, persuasion or negotiation. Agents can construct a dialogue by dynamically adjusting the content and sequences of utterances as the discussion ensues. In our framework, we have used an inquiry dialogue to make two agents take turns in asserting, questioning, accepting, or rejecting statements. The goal of an inquiry dialogue is to find out whether a statement is true or false or show that there is insufficient evidence to accept a statement [29]. In our work, evidence can be formal proof or an informal statement (e.g. statistical evidence) for a desirable property of the trading mechanism. An example of a desirable property of an auction mechanism could be that the highest buyer wins or that biding its true valuation is the optimal strategy for a buyer. Formal proofs are constructed using the COQ [10] theorem prover within the PCC (Proof-Carrying Code) paradigm [23]. In our online auction scenario, PCC enables the auctioneer to develop proofs for properties of interest and the buyer to check the correctness of a given proof [5]. By considering the set of evidences collected in a dialogue for a set of desirable properties, a buyer agent as a service consumer can evaluate the quality of a service and make decision as to whether to enter or leave a service.

On one hand, in the language architecture of Semantic Web [14], each language in the above layer extends the capabilities of the layer below, as shown in **Figure 1**. For example, OWL extends RDFS by providing more semantic features like cardinality, union, intersection and more reasoning possibilities. The top two layers in this Semantic Web stack are the Proof and Trust layers. In general, proofs can be constructed to increase trustworthiness in the system. However, the current system does not support yet a proof construction or procedure for building up trust. On the other hand, Interactive theorem proving provides an approach to develop formal proof by man-machine collaboration. It is an important approach which using formal methods to verify the correctness of a protocol or a mathematical statement. In an Interactive Theorem Prover (ITP), human can create definitions, theorems and generate proofs using an interactive proof editor. The ITP also supports automatic checking of proofs. In our work, we use COQ [10], which is an ITP, to generate and check the proofs of desirable properties of an online auction.

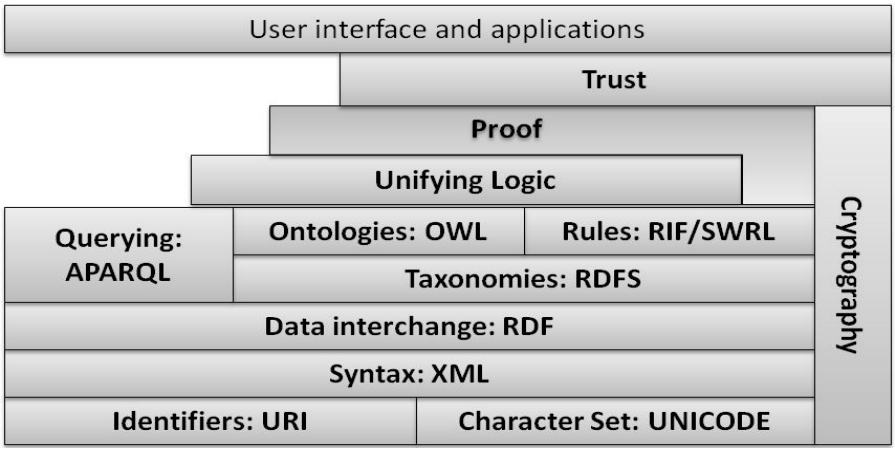

Figure 1. Semantic Web Stack

The contributions in this paper are three-fold:

- We have integrated dialogue games within the PCC paradigm to increase trust in an agents-mediated online auction. As a consequence, we have extended the interactivity of agents' communication wherein formal proofs can be used as arguments in a dialogue.
- Since, not all desirable properties of an online auction mechanism will have associated formal proofs, we have allowed for informal or empirical evidence related to the QoS (Quality of Service). For example, an auctioneer may claim that it does not have a proof that its mechanism is free from cheating, but 98% of its consumers never complained about being cheated. We have designed a dialogue game wherein formal and informal evidence can be used as arguments.
- We have constructed a decision model over the formal and empirical evidences allowing a buyer agent, with predefined expectations, to decide whether to join or not an auction.

The proposed dialogue game framework is implemented in JADE [6], which is a widely used tool to implement multi-agent systems. It provides mechanisms to create agents, enable agents to execute tasks and make agents communicate with each other.

The remainder of this paper is organized as follows: In section 2, we will introduce the technique of Semantic Web Services and give an example of an English Auction which is written in the language of OWL-S. The proposed framework is presented in Section 3. In Section 4, we present an example that implement our framework, and is then evaluated in Section 5. Related work is presented in Section 6, before concluding in Section 7.

2 Semantic Web Services

The Semantic Web [7] not only enables greater access to content but also to services on the Web. Semantic We Services is a technology that combines Semantic Web and Web services to develop new web applications. Web services technology is based on a set of standard protocols such as UDDI (Universal Description, Discovery and Integration), SOAP (Simple Object Access Protocol), and WSDL (Web Services Description Language). However, these standards do not support automated Web services. To improve the automatic properties of Web services, Semantic Web Services (SWS) are created [9]. OWL-S [17] and WSMO (Web Service Modeling Ontology) [25] are two standards for the SWS technology. OWL-S is composed of three main parts: the *service profile* for advertising and discovering services; the *process model*, which gives a detailed description of a service operation; and the *grounding*, which provides details on how to interoperate with a service via messages. WSMO provides a conceptional framework for semantically describing all relevant aspects of Web services to facilitate the automation of discovering, combining and invoking electronic services over the Web. WSMO comprises four main elements: *ontologies*, which provide the terminology used by other WSMO elements, Web service *descriptions*, which describe the functional and behavioral aspects of a Web service, *goals* that represent the user's desires, and *mediators*, which aim at automatically handling interoperability problems between different WSMO elements. Current SWS technique can help us build a system that enables agents to publish, discover and invoke services in an open environment (e.g. the Internet).

2.1 A Scenario: An English Auction

In this paper, we use OWL-S to build up the Web service. OWL-S can be used together with other Semantic Web languages, such as OWL DL [22] and SWRL [15], to describe the properties and capabilities of a Web service in unambiguous, computer-interpretable form. To set up a Semantic Web service, the first step is

to build the ontology of a specific area. The ontology can be used to formally describe the semantics of terms representing an area of knowledge and give explicit meaning to the information. This enables automated reasoning, semantic search and knowledge management of the specific area. For example, the ontology of auction domain can contain the constructs of classes, relations, axioms, individuals and assertions. We give a framework of the auction ontology in **Figure 2**.

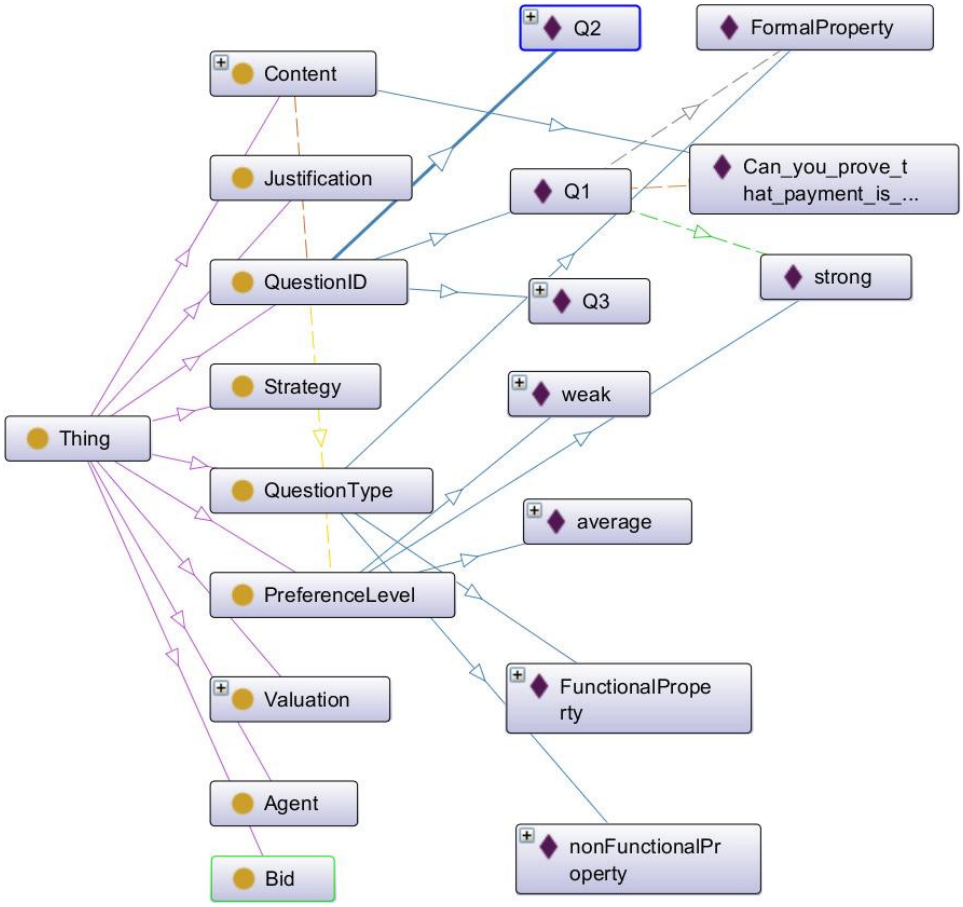

Figure 2. A Fragment of the Auction Ontology

The auction domain ontology not only defines basic information of an agent (e.g. bid, valuation, strategy), but also the preference of an agent. These preferences can be used to help agents make decisions during a trading dialogue. The following code illustrates the ontology about a question named Q1. Q1 has an object property *hasContent*, and a data property with name *hasWeight*.

```
<ClassAssertion>
    <Class IRI="#QuestionID"/>
    <NamedIndividual IRI="#Q1"/>
</ClassAssertion>
 <ObjectPropertyAssertion>
    <ObjectProperty IRI="#hasContent"/>
    <NamedIndividual IRI="#Q1"/>
```

```
    <NamedIndividual IRI="#Can_you_prove_that_payment_is_the_highest_bid"/>
  <DataPropertyAssertion>
  <DataProperty IRI="#hasWeight"/>
    <NamedIndividual IRI="#Q1"/>
    <Literal datatypeIRI="&xsd;double">0.5</Literal>
  </DataPropertyAssertion>
```

After the construction of domain ontology, the auction mechanism should be described using the OWL-S ontology. In such a trading mechanism, functional description should be defined: inputs, outputs, preconditions and results (IOPEs). The *inputs* are the objects that should be provided to invoke the service; the *outputs* are the objects that the service produces; the *preconditions* are the propositions that should be true prior to service invocation; and the *results* consists of effects and outputs. The following code is an example showing that, the precondition for running an auction is that at least one agent must have some bid.

```
<process:hasPrecondition>
 <expr:SWRL-Condition>
        <expr:expressionObject>
              <swrl:AtomList>
              <rdf:first>
              <swrl:DatavaluedPropertyAtom>
              <swrl:propertyPredicate
         rdf:resource="&mybid;#hasBid"/>
              <swrl:argument1 rdf:resource="&this;#agent"/>
              <swrl:argument2 rdf:resource="&this;#abid"/>
          </swrl:DatavaluedPropertyAtom>
              </rdf:first>
              <rdf:rest rdf:resource="&rdf;#nil"/>
              </swrl:AtomList>
        </expr:expressionObject>
  </expr:SWRL-Condition>
</process:hasPrecondition>
```

In OWL-S, a process represents a specification of the means a buyer agent uses to interact with a service. In our scenario, we use two kinds of processes: one is *atomic process,* which corresponds to a single interchange of a request message and a response message; the other is *composite process*, which consists of a series of processes linked together by control flows and data flows. The control flow describes the relations between the executions of different sub-processes. The control constructs include *sequence,*

split, if-then-else and iterate etc. Data flow specify how information is transferred from one process to another process. In our example of an English auction, we can define one *if-then-else* branch to describe that when a new bid is greater than current bid (a local variable), we update the value of current bid with the new bid. This process can be simply described as follows.

```
<process:CompositeProcess>
 <process:composedOf>
        <process:If-Then-Else rdf:ID="CompareBid">
                <process:ifCondition>
                        <expr:SWRL-Condition>
                        swrlb:lessThan(#currentBid,#newBid)
                        </expr:SWRL-Condition>
                </process:ifCondition>
                <process:then>
 <!-- Update the value of #currentBid -->
     ...
                </process:then>
                <process:else>
<!-- Keep the value of #currentBid as usual-->
     ...
                </process:else>
  </process:If-Then-Else>
 </process:composedOf>
</process:CompositeProcess>
```

In our setting, we consider the scenario that buyers communicate with the seller to decide whether to join an online auction, Therefore, we do not need to define the grounding of the service. OWL-S can be used to build complex business solutions by describing the functional, non-functional properties of a service, so that agents can perform automatic reasoning on these descriptions. Dialogue games, which can help software agents interact rationally by providing support or counterexample for a conclusion, make this reasoning more flexible for greater interaction between an auctioneer and a buyer.

3 Our Dialogue Game Framework

Online auction web sites, such as eBay, have attracted millions of users around the world to sell, bid and buy goods. In our specific scenario, software agents are assumed to be capable of buying or selling goods through online auction houses. In this scenario, how can buyer agents choose the appropriate auction house? A buyer agent will have properties of interest. These properties need to be kept in the auction

mechanism for the agent to join, bid, and buy items. Our framework is aimed at enabling a potential buyer (e.g., software agent) to interact with the auctioneer before deciding whether to join an auction. These interactions between an auctioneer and a buyer are carried out within a dialogue game wherein the buyer agent can query whether desirable properties hold and request associated evidences. To enable trust, the auctioneer uses the PCC paradigm to convince buyers that a service has some desirable properties, as shown in our previous work [4,5]. This enables the auctioneer to specify and develop formal proofs for those properties. Then, the buyer uses a proof checker to check that a given proof of a well specified property of the auction is correct. Besides, a buyer can object to the auctioneer under the condition that a counterexample is found by the proof checker. Thus, trust can be established between an auctioneer and a buyer. **Figure 3** illustrates our framework that adapts PCC to certify auction properties. At the producer or auctioneers side, we have the specifications of auction mechanism along with the proofs of desirable properties in a machine-checkable formalism in the form of a COQ file. The certification procedure works as follows. The buyer agent arriving at the auction house can download its specification and the claimed proof of a desirable property. Then, the buyer requests the proof checker *coqhk*, which is a standalone verifier for COQ proofs, to the auctioneer. The auctioneer provides the address of the proof checker which is stored by a trusted third party (e.g., the home page of COQ) to the buyer. After the proof checker is installed to the consumer side, the buyer can perform all verifications of claimed properties of the auction before deciding to join and with which bidding strategy.

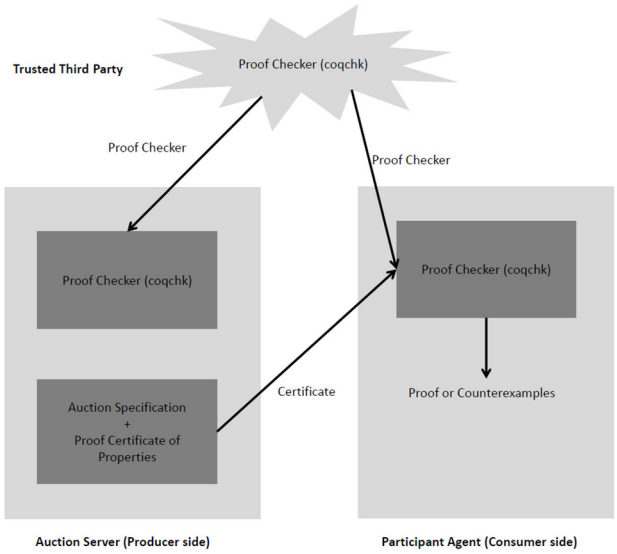

Figure 3. Applying PCC to Certify Auction Properties

We have integrated the PCC paradigm within our dialogue model as illustrated by the framework in **Figure 4**. In this framework, both the auctioneer and the buyer share the same question ontology for general online auction services. The auctioneer holds an OWL-S ontology which provides a machine understandable description of the auction mechanism. Evidence for a claimed property may have an informal or formal justification. Informal justification relies upon data collected by the service. In the case of properties with formal proofs, the related specification of the OWL-S description can be translated into COQ specifications so that proofs of these properties can be developed from within COQ. This kind of sound language translation is described in our previous work [3]. The proof certificates for the established desirable properties are local to the auctioneer. This strengthens the interactivity between a buyer and

an auctioneer and reduces knowledge disclosing. The proofs will be disclosed to a buyer when related questions are proposed in a dialogue game. The dialogue game is used to enable a buyer agent to find out about properties and certificates. This dialogue is a two-person game, which means that only one buyer can communicate with the auctioneer at a time.

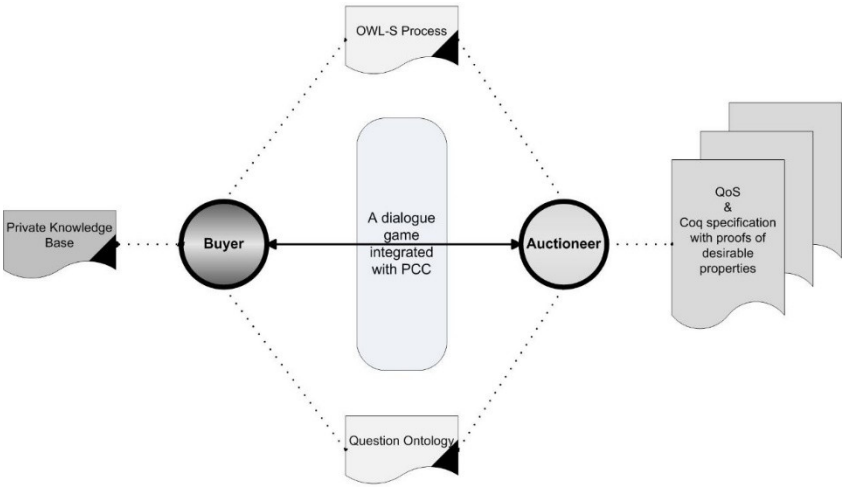

Figure 4. The dialectical approach framework.

3.1 The Formal Dialogue Model

The proposed inquiry dialogue consists of several *locutions* or *moves*, *pre-conditions* that indicate the rules that must be satisfied before a move, and the *post-conditions* that describe the actions that will occur after a move. We have restricted the number of participants in the inquiry dialogue to two. Let P be the participants in the dialogue. A participant is either a sender or a recipient.

A dialogue D is simply defined by a sequence of moves between participants. One move represents a message exchange made from one participant to the other. As the dialogue progresses, each move is indexed by a timepoint, which is denoted by a natural number, and only one move can be made at each timepoint. In our inquiry dialogue model, seven types of moves are defined. They are *open, assert, question, justify, accept, reject* and *close*, and the type of each legal move should be one of them.

Definition 1. A ***dialogue***, denoted D, is a sequence of moves $[m_r, \dots, m_t]$, where $\mathrm{r}, \mathrm{t} \in \mathrm{N}, \mathrm{r} < \mathrm{t}$, involving two participants $P_i \in P, i = \{1, 2\}$, such that:

1. the first move of the dialogue, m_r is type *open*,
2. the last move of the dialogue, $m_t$is of type *close*,
3. $\mathrm{Sender}(m_s) \in \mathrm{P}(\mathrm{r} \leq \mathrm{s} \leq \mathrm{t})$,
4. $\mathrm{Sender}(m_s) \neq \mathrm{Sender}(m_{s+1})(\mathrm{r} \leq \mathrm{s} \leq \mathrm{t})$.

The first move of a dialogue D must always be an *open* move (condition 1), while the last move should be a *close* move (condition 2). Each move of the dialogue must be performed by a participant of the dialogue (condition 3). Finally, participants take turns to make moves (condition 4).

The dialogue assumes that each participant holds a commitment store that records its statements in a dialogue. The commitment store of participant P_i is defined as a private-write, public-read record containing all the commitments incurred by P_i, but the content of this commitment store can only be written using the moves made by P_i.

Definition 2. A ***commitment store*** is a set of beliefs denoted as CS_x^t, where $x \in P$ is an agent and $t \in N$ is a timepoint.

The commitment store of P_i is created when the agent enters a dialogue and persists until the dialogue terminates.

Definition 3. A ***proof*** is an argument from hypotheses to a conclusion and each step of the argument follows the laws of logic.

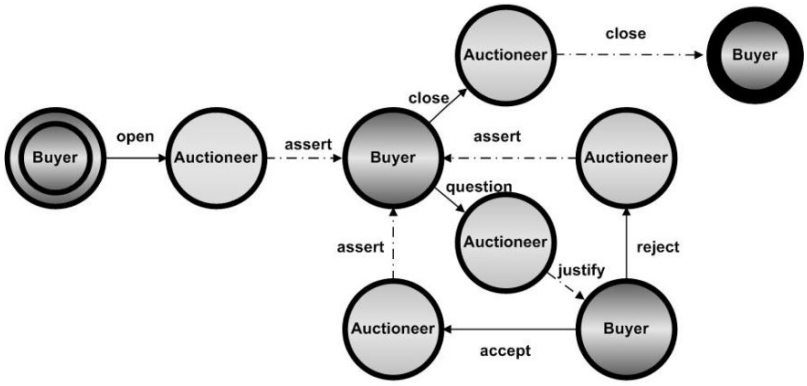

Figure 5. The State Diagram of the dialogue.

The state diagram of the dialogue is given in **Figure 5.** A buyer can open a dialogue by using the *open* move. Then, the auctioneer can give an assertion to the buyer. The buyer can choose to close the dialogue on the condition that he does not have any issues to raise with the auctioneer. Otherwise, the buyer can give a move of type *question* to query on the issues that he is concerned about. The auctioneer must give a justification to the buyer for each specific question. These are three cases in the justification process: 1) the auctioneer has a formal proof for the answer to the question at hand; 2) the auctioneer has an informal evidence for the answer; or 3) the auctioneer does not know the answer for the question. In the first case, the buyer will use the PCC paradigm to check the correctness of the formal proof, if this proof is certified, then the buyer gives an *accept* move, otherwise, the proof checker will generate a counterexample and deny the property that related to the question, then the buyer makes the move of *reject*. In the second case, the buyer will compare the informal evidence with a reasonable expected value for the issue of interest and will accept it with some score. In the last case, the buyer will give a *reject* move to the auctioneer. The auctioneer can give assertions to the buyer, so that the dialogue can carry on until both participants agreed to close the dialogue.

The components of a dialogue include:

- $\sum$: The knowledge base, or beliefs of each agent.
- $CS \in \sum$: an agent's commitment store that refers to the statements that have been made in the dialogue.
- a: auctioneer.
- b: buyer.
- x: either auctioneer or buyer.
- ρ: the last sender in the current dialogue.

The locutions used in the dialogue model are defined as follows.

Definition 4. $open(b)$: buyer b opens a dialogue.

- Pre-conditions:
  1. $\mathrm{b} \notin \mathrm{P}$, where P is the set of dialogue participants.
- Post-conditions:
  1. $P' = P \cup \{b\}$
  2. $CS^b = \emptyset$
  3. $\rho = \mathrm{b}$

A buyer b can open a dialogue. The precondition of this locution is that the buyer is not a participant of this dialogue. The postcondition is that buyer b becomes a participant of the dialogue (post-condition 1) and his commitment store is created (post-condition 2).

Definition 5. $assert(a, b, \phi)$: auctioneer a gives an assertion to the buyer.

- Pre-conditions:
  1. $\mathrm{a} \neq \rho$
  2. $\mathrm{locutiontype}(l_{s-1}) \in \{\mathrm{open}, \mathrm{accept}, \mathrm{reject}\}$
  3. $\phi \in \sum^a$
- Post-conditions:
  1. $CS^{a'} = \{\phi\} \cup CS^a$
  2. $\rho = a$

The auctioneer a should not have uttered the previous move (pre-condition 1). The *assert* move should be given under the condition that a buyer has open a dialogue, the justification is accepted or rejected by the buyer (pre-condition 2). A belief ϕ from the beliefs store of the auctioneer will be asserted by the auctioneer to ask the buyer to propose a question (pre-condition 3). Once the assert has been uttered, the belief ϕ will be added to the commitment store of the auctioneer (post-condition 1).

Definition 6. $question(b, a, \phi)$: buyer b asks a question about property ϕ to the auctioneer a.

- Pre-conditions:
  1. $\mathrm{b} \neq \rho$
  2. $\mathrm{locutiontype}(l_{s-1}) \in \{\mathrm{assert}\}$
  3. $\phi \notin \sum^b$
  4. $\phi \notin CS^b$
- Post-conditions:
  1. $\phi \notin \sum^b$
  2. $\phi \in CS^b$
  3. $\rho = \mathrm{b}$

A buyer can ask questions to the auctioneer after the auctioneer has uttered an assertion (pre-conditions 1 and 2). The property ϕ does not contain the knowledge base of b (pre-condition 3) and the buyer has not proposed questions about this property (pre-condition 4). Once b has uttered the question, the knowledge base of b does not change (post-condition 1), and the commitment store of b has been updated (post-condition 2).

Definition 7. $justify(a, b, \phi)$: auctioneer a justifies the property ϕ for buyer b.

- Pre-conditions:

1. $\mathrm{a} \neq \rho$
2. $\mathrm{locutiontype}(l_{s-1}) \in \{\mathrm{question}\}$
3. $\phi \in CS^b$
4. $\phi \notin \sum^b$

- Post-conditions:
  1. $\phi \notin \sum^b$
  2. $\phi \in CS^a$
  3. $\rho = a$

After receiving a question about property ϕ (pre-condition 2 & 3), the auctioneer a should provide a justification to this property. The knowledge base of the buyer and commitment store remains the same in this move (post-condition 1). The justification of this property is added to the commitment store of the auctioneer (post-condition 2).

Definition 8. $accept(b, a, \phi)$: buyer b accepts the justification of property ϕ.

- Pre-conditions:
  1. $\mathrm{b} \neq \rho$
  2. $\mathrm{locutiontype}(l_{s-1}) \in \{\mathrm{justify}\}$
  3. $\phi \notin \sum^b$
  4. $\phi \in CS^a$
- Post-conditions:
  1. $\phi \in \sum^b$
  2. $\phi \in CS^a$
  3. $\rho = \mathrm{b}$

The *accept* move is used to accept the justification for property ϕ in the preceding *justify* move (pre-condition 2). The property ϕ is not contained in the knowledge base of the buyer before the move of *accept* (pre-condition 3), and the auctioneer has provided the justification for this property (pre-condition 4). Property ϕ becomes the element of the knowledge base of b after this move (post-condition 1). The commitment store of auctioneer a does not change in the move of *accept* (post-condition 2).

Definition 9. $reject(b, a, \phi^{att}, \phi)$: buyer b rejects the property ϕ using ϕ^{att}.

- Pre-conditions:
  1. $\mathrm{b} \neq \rho$
  2. $\mathrm{locutiontype}(l_{s-1}) \in \{\mathrm{justify}\}$
  3. $\mathrm{attack}(\phi^{att}, \phi)$
  4. $\phi \notin \sum^b$
  5. $\phi \in CS^a$
- Post-conditions:
  1. $\phi \notin \sum^b$
  2. $\phi \in CS^a$
  3. $\rho = \mathrm{b}$

Buyer can raise a rejection by giving an evidence ϕ^{att} to previously declared property ϕ in response to the move *justify* (pre-condition 2). There is an evidence ϕ^{att} which attacks ϕ (pre-condition 3). The knowledge base of buyer b and commitment store of auctioneer *a* does not change (pre-condition 4 & 5 and post-condition 1 & 2).

Definition 10. $close(p)$: participant p closes a dialogue.

- Pre-conditions:
  1. $\mathrm{p} \neq \rho$
  2. $\mathrm{locutiontype}(l_{s-1}) \in \{\mathrm{assert}, \mathrm{close}\}$
  3. $\forall \phi \in \sum^{b}, \phi \in CS^{b}$
- Post-conditions:
  1. If matched-close $\mathrm{P} = \emptyset$

The participant can only close a dialogue after an *assert* or *close* move (pre-condition 2). The buyer chooses to close the dialogue when all the questions in his knowledge base have been proposed (pre-condition 3). When both participants have agreed to close the dialogue, they will be removed from the dialogue (post-condition 1).

3.2 Decision Model and Processes of the Dialogue

In our setting, both the auctioneer and the buyer agents share the same knowledge base, which includes the ontologies of the online auction and related specifications. Both agents can understand each other's messages but have a private knowledge base. The set of questions are private to the buyer and the set of answers associated to the questions are private to the auctioneer. However, when a question or answer is proposed, it becomes public to both participants. Each question is associated with a *preference level*, which determines the order in which the questions are proposed by the buyer. The preference level $\mathrm{E} = \{\mathrm{strong}, \mathrm{average}, \mathrm{weak}\}$ is then used to drive the dialogue forward.

A *Weighted Sum* model is used in the dialogue game. The output of the *Weighted Sum* model is a binary set of *O*, whose elements are {*Yes, No*}. Decision is made by evaluating a bunch of properties, which are represented as the set of *H*. We assume that every property is bind with a score s_i and a relative weight w_i which indicates the importance of a property. The value of s_i is determined by a function which depends on the type of evidence provided. If we have formal evidence, then the scoring function will return either negative value in the case of the proof cannot be accepted by the proof-checker, or a full score otherwise. If the evidence is empirical, then the scoring function is in the form of intervals, see **Section 4** for more details. It is the usual practice to set the summation of weights to 1 in the weighted sun model, such that $\sum_{i=1}^{n} w_i = 1$ wherein n is the number of elements in H. The final score fs is calculated as follows:

$$fs = \sum_{i=1}^{n} w_i s_i$$

, where n is also the number of elements in H.

The buyer has a reasonable expectation in the form of an $admissibility\ threshold\ \varepsilon$ beyond which the final score will lead the buyer to join the auction. In other words, if $\mathrm{fs} \geq \varepsilon$, then the buyer will choose *Yes* to join the auction at hand. The threshold ε may come from experience or be derived from historical data.

Algorithm 1 The processes of an inquiry dialogue

1. buyer opens an inquiry dialogue uses an $open$ move
2. auctioneer gives an $assert$ to ask buyer to propose a question
3. **while** buyer has question(s) that has(have) not been proposed **do**
4. buyer selects a question which with the highest preference level
5. buyer asks this question using the move whose type is $question$
6. Auctioneer gives a justification using the move whose type is of $justify$
7. **if** the justification is unknown or has failed to be checked **then**
8. buyer gives a $reject$ move
9. **else**
10. buyer gives an $accept$ move
11. **end if**
12. auctioneer gives an $assert$ to ask buyer to propose a question
13. **end while**
14. buyer gives a $close$ move
15. auctioneer gives a $close$ move to terminate the dialogue

The inquiry dialogue is described by Algorithm 1. A dialogue starts with a move opening an inquiry dialogue, that has a matched-close to terminate the dialogue and whose moves conform to the rules for each locution described in Section 3.1. The $buyer$ starts an inquiry dialogue by giving an $open$ move, then the auctioneer propose an $assert$ move to ask $buyer$ to propose questions. The $buyer$ choose a question which has not been proposed that has the highest preference level from his knowledge base and then uses the move $question$ to propose it. After receiving a question, the auctioneer should respond by a $justify$ move. If the content of a justification is unknown, or a formal proof that is failed to be checked. The $buyer$ will give a $reject$ move, otherwise he should accept the justification use an $accept$ move. The $auctioneer$ should give an assertion to ask $buyer$ to propose a new question in the end of the loop. A dialogue can be closed when both participants agree to terminate it. All the proposed questions and related justifications are stored in the commitment store of the dialogue. Besides, each question can only be asked once.

4 Inquiry Dialogue Example

We have illustrated the proposed inquiry dialogue by means of an example where a $buyer$ talks to an $auctioneer$ to decide whether to join an online auction service. The shared question ontology contains three types of questions: 1) questions about those properties whose answers could be formal proofs of a service; 2) questions about the functional properties (e.g. inner operation of a service) of a service; and 3) questions about the non-functional properties (e.g. QoS) of a service. In this example, seven questions are proposed as listed in **Table 1.**

Table 1. Question in the shared ontology.

QuestionID	hasType	hasContent
Q1	FormalProperty	Can you prove that the payment is the highest bid?
Q2	FunctionalProperty	What's the payment method?
Q3	FunctionalProperty	What's the delivery company for your product?
Q4	FunctionalProperty	What's the Input of the service?
Q5	nonFunctionalProperty	What's the Reputation of your service?
Q6	nonFunctionalProperty	What's the ResponseTime of your service?
Q7	FormalProperty	Can you prove that the winner has the highest bid?

In Table 1, each question is marked by an ID, and the related type and content of each question are defined use the OWL $objectProperty$ relationship. Users can extend this ontology by adding questions as the classification of types.

The $buyer$ holds a private knowledge base that contains the questions he wants to inquire the dialogue. The preference level and weight of each question are shown in **Table 2.** As mentioned above the $buyer$ will choose the questions in order based on the preference level and calculate the scores of the justifications using the variable of weight, w. The summation of all the weights in this table equals to one.

Table 2. Questions in the knowledge base of the buyer.

QuestionID	Preference Level	Weight (w)	hasContent
Q1	strong	0.3	Can you prove that the payment is the highest bid?
Q2	average	0.15	What's the payment method?
Q3	weak	0.1	What's the delivery company for your product?
Q5	strong	0.2	What's the Reputation of your service?
Q6	average	0.15	What's the ResponseTime of your service?
Q7	weak	0.1	Can you prove that the winner has the highest bid?

The $auctioneer$ uses **Table 3** to search for the answers for each query, a $buyer$ cannot directly visit this table unless he queries the $auctioneer$. All the questions in the shared ontology should be contained in this table, though there may exist questions that do not have related answers. **Table 4** shows the grading standards of the $buyer$ for the justifications. The highest score for each question is 100, the lowest score for the formal question is -50 when the proof provided by the auctioneer is failed to check, and the lowest score of other questions is 0. As this grading table is subjective, a $buyer$ can grade the justification based on his own preference. However, for the question whose justification is a formal proof, the $buyer$ should give full marks to it when the proof is successfully checked.

Table 3. Justifications of questions hold by the auctioneer.

QuestionID	hasContent	Justification
Q1	Can you prove that the payment is the highest bid?	FormalProof_PEquHB
Q2	What's the payment method?	Visa DEBIT
Q3	What's the delivery company for your product?	DHL
Q4	What's the Input of the service?	BuyerID & Bid
Q5	What's the Reputation of your service?	0.85
Q6	What's the ResponseTime of your service?	unknown
Q7	Can you prove that the winner has the highest bid?	FormalProof_WhatHB

Table 4. Grading Table of the Buyer.

QuestionID	Justification	Score
Q1	Formal Proof Successes	100
	Formal Proof Fails	-50
	Do Not Have Formal Proof	0
Q2	Alipay	100
	Visa DEBIT	80
	Bank Cards	50
	Others	0
Q3	SF EXPRESS	100
	DHL	80
	EMS	70
	Others	0
Q5	$[0.90,1.00]$	100
	$[0.70,0.90)$	80
	$[0.50,0.70)$	50
	$[0.00,0.50)$	0
	unknown	0
Q6	$(0.00\text{ns}, 0.03\text{ns})$	100
	$[0.03\text{ns}, 0.05\text{ns})$	80
	$[0.05\text{ns}, 0.10\text{ns})$	50
	$[0.10\text{ns}, +\infty)$	0
	unknown	0
Q7	Formal Proof Successes	100
	Formal Proof Fails	-50
	Do Not Have Formal Proof	0

An example dialogue between a buyer and an auctioneer is presented as in **Figure 6**. The turn order in the dialogue is deterministic, the $buyer$ should open a dialogue in **Move 1.** Then the $auctioneer$ and $buyer$ give assertions one by one.

Moves 2-5: The $auctioneer$ asks $buyer$ to ask a question. $Buyer$ searches in his knowledge base of questions and find out a question that with the highest preference, which is Q1. Then the $auctioneer$ searches the justification for the question from **Table 3**. This justification is a formal proof, so $auctioneer$ should send the address of a proof checker (in the case that the $buyer$ does not have the proof checker) and related proof files to the $buyer$. The $buyer$ then downloads and uses the proof checker to check the correctness of the proof, and gets positive feedback. Finally, the $buyer$ accepts the justification.

Moves 6-9: The $auctioneer$ asks $buyer$ to ask another question. The $buyer$ finds question Q5, which has the highest preference level within the remaining questions. The $auctioneer$ gives the justification with value 0.85 and the $buyer$ accepts it.

Moves 10-13: After the $auctioneer$ requests the $buyer$ to ask a question. The $buyer$ proposes question Q2 and receives the justification with value of "Visa DEBIT". The $buyer$ accepts the justification in this round.

Moves 14-17: In this round, the $buyer$ raises question Q6, the $auctioneer$ does not have a justification for this question, so the content of this justification becomes *unknown*. Then the $buyer$ rejects this justification.

(1) *buyer* → *auctioneer*: *open(buyer)*
(2) *auctioneer* → *buyer*: *assert(auctioneer, buyer, You can ask a question.)*
(3) *buyer* → *auctioneer*: *question(buyer,auctioneer,Q1)*
(4) *auctioneer* → *buyer*: *justify(auctioneer,buyer,FormalProof_PEquHB)*
(In the justification process, the *auctioneer* sends the proof to the *buyer* and the proof is successfully checked by using the COQ proof checker.)
(5) *buyer* → *auctioneer*: *accept(buyer,auctioneer,FormalProof_PEquHB)*
(6) *auctioneer* → *buyer*: *assert(auctioneer, buyer, You can ask a question.)*
(7) *buyer* → *auctioneer*: *question(buyer,auctioneer,Q5)*
(8) *auctioneer* → *buyer*: *justify(auctioneer,buyer,0.85)*
(9) *buyer* → *auctioneer*: *accept(buyer,auctioneer,0.85)*
(10) *auctioneer* → *buyer*: *assert(auctioneer, buyer, You can ask a question.)*
(11) *buyer* → *auctioneer*: *question(buyer,auctioneer,Q2)*
(12) *auctioneer* → *buyer*: *justify(auctioneer,buyer,Visa DEBIT)*
(13) *buyer* → *auctioneer*: *accept(buyer,auctioneer, Visa DEBIT)*
(14) *auctioneer* → *buyer*: *assert(auctioneer, buyer, You can ask a question.)*
(15) *buyer* → *auctioneer*: *question(buyer,auctioneer,Q6)*
(16) *auctioneer* → *buyer*: *justify(auctioneer,buyer, unknown)*
(17) *buyer* → *auctioneer*: *reject(buyer,auctioneer, unknown)*
(18) *auctioneer* → *buyer*: *assert(auctioneer, buyer, You can ask a question.)*
(19) *buyer* → *auctioneer*: *question(buyer,auctioneer,Q3)*
(20) *auctioneer* → *buyer*: *justify(auctioneer,buyer, DHL)*
(21) *buyer* → *auctioneer*: *accept(buyer,auctioneer,DHL)*
(22) *auctioneer* → *buyer*: *assert(auctioneer, buyer, You can ask a question.)*
(23) *buyer* → *auctioneer*: *question(buyer,auctioneer,Q7)*
(24) *auctioneer* → *buyer*: *justify(auctioneer,buyer,FormalProof_WhasHB)*
(In the justification process, the *auctioneer* sends the proof to the buyer. The *buyer* failed to checked the proof by using the COQ proof checker, which means the proof provided by the *auctioneer* is wrong. The *buyer* finds an attack to this property.)
(25) *buyer* → *auctioneer*: *reject(buyer,auctioneer,FormalProof_WhasHB)*
(26) *auctioneer* → *buyer*: *assert(auctioneer, buyer, You can ask a question.)*
(27) *buyer* → *auctioneer*: *close(buyer)*
(28) *auctioneer* → *buyer*: *close(auctioneer)*

Figure 6. An Example of the Inquiry Dialogue

Moves 18-21: The $buyer$ proposes another question in his knowledge base and accepts the justification that is given by the $auctioneer$.

Moves 22-25: The $auctioneer$ asks $buyer$ to ask a question. The $buyer$ proposes the last question from his knowledge base, which is Q7. Then the $auctioneer$ sends the proof to the $buyer$. The *buyer* uses the proof checker to check the correctness of the proof and gets a counterexample, which means the proof provided by the auctioneer is wrong. In this case, the $buyer$ finds an attack of this property. Finally, the $buyer$ rejects the justification.

Moves 26-28: The $auctioneer$ asks $buyer$ to propose a new question. However, all the questions in the buyer's knowledge base have been raised. The $buyer$ chooses to close the dialogue and the $auctioneer$ agrees to close the dialogue.

After the termination of the dialogue, the $buyer$ calculates the scores for each justification. The final score $fs = 0.3 * 100 + 0.15 * 80 + 0.1 * 80 + 0.2 * 80 + 0.15 * 0 + 0.1 * (-50) = 61$ We assume that the $admissibility\ threshold$, ε, of the buyer is 70. As $fs < \varepsilon$, the *buyer* decides to not join the auction house.

5 Empirical Evaluation

We have implemented the proposed dialogue model in the JADE platform. We have added the proposed locutions to the original FIPA ACL messages, so that the original methods (e.g., $addReceiver$) can be called directly. The pre/postconditions of each locution are hard-coded in the program to detect the state of a dialogue. The combination rules which are illustrated in Figure 5 are implemented using the behavior control mechanisms of JADE. When a buyer needs to check a proof, the program will use a command to call the external software COQ and check the correctness of a proof. In our dialogue model, only two participants are allowed.

In the first experiment, there were five different auctioneers, from A_1 to A_5, which is shown in the first column of **Table 6**. Each of the auctioneers holds an auction service. A buyer talks with each of them based on the questions $\{Q_1, Q_2, Q_3, Q_5, Q_6\}$. This table shows the grades given for each question as well as the total scores, the grading is based on **Table 4**. For the auction hold by A_1, because the value of the total score is greater than the admissibility threshold, ε, whose value is 70, the buyer decides to join this auction after the dialogue. For A_4, the score for Q_1 is -50 which means that the result of the proof check failed. This table shows that a buyer can make decisions for different auction services based on the same questions and grading system through dialogues.

Table 6. Decision making for different auction services.

AuctioneerID	Q1 (w=0.35)	Q2 (w=0.15)	Q3 (w=0.1)	Q5 (w=0.25)	Q6 (w=0.15)	Total Score	Decision ($\varepsilon = 70$)
A1	100	100	80	100	80	95	Y
A2	0	50	70	80	0	34.5	N
A3	100	50	80	80	100	85.5	Y
A4	-50	80	100	50	50	24.5	N
A5	100	50	70	50	50	69	N

Table 7 shows the result of our second experiment. We kept the value of final scores (the second column) of each service the same as in **Table 6**. Then, by changing the admissibility threshold from 60 to 100, the number of services that are rejected has increased. The value of ε reflects the subjective belief of a buyer. If the buyer has higher expectations from the service. Then the value of ε will be higher.

Table 7. Decision making for different auction services with different admissibility threshold.

AuctioneerID	Total Score	Decision ($\varepsilon = 60$)	Decision ($\varepsilon = 70$)	Decision ($\varepsilon = 80$)	Decision ($\varepsilon = 90$)	Decision ($\varepsilon = 100$)
A1	95	Y	Y	Y	Y	N
A2	34.5	N	N	N	N	N
A3	85.5	Y	Y	Y	N	N
A4	24.5	N	N	N	N	N
A5	69	Y	N	N	N	N

In the third experiment, we assume that there is only one auctioneer and three questions (Q_1, Q_2, Q_5). There are three buyers (B_1, B_2, B_3) whose weights for each question are different. We use the same

grading **Table 4** to score each question as in the previous experiments. The admissibility threshold of all buyers is set to 70. The last column of **Table 8** shows that B_2 and B_3 choose to join the auction while B_1 refused to. This table shows that, even if each buyer gets the same justification from the same auctioneer and they have the same admissibility threshold, the weights of questions can influence the result of the decision making.

Table 8. Decision making for the same auction service with different weight.

B1	Q1(w=0.20)	Q2(w=0.50)	Q5(w=0.30)	Total Score	Decision ($\varepsilon = 70$)
	100	50	80	69	N
B2	Q1(w=0.30)	Q1(w=0.50)	Q1(w=0.20)	Total Score	Decision ($\varepsilon = 70$)
	100	50	80	71	Y
B3	Q1(w=0.40)	Q1(w=0.35)	Q1(w=0.25)	Total Score	Decision ($\varepsilon = 70$)
	100	50	80	77.5	Y

The protocol proposed in our work satisfied a set of desiderata [21].

- *Stated dialogue purpose:* The purpose of the dialogue is to enable buyer agents to make decision by querying the auctioneer. All participants are aware of this purpose before they enter in the dialogue.
- *Diversity of individual purposes:* The dialogue model lets agents achieve their own purposes in terms of inquiry with peers.
- *Inclusiveness:* There is no elimination of agents.
- *Transparency:* The protocol syntax and semantics are public and available to all participants, along with the combination rules of locutions.
- *Fairness:* The locutions and protocol are the same for all participants, which means that no agents have any privileges in the society.
- *Rule*-consistency*:* All protocol rules are consistent with the syntax and semantics.
- *Separation of Syntax and Semantics:* The syntax and semantics of the proposed locutions are separately defined.
- *Encouragement of resolution:* The dialogue will terminate based on the dialogue driven algorithm and termination rules.
- *Discouragement of disruption:* The commitment rules preclude disruptive behaviors. For example, the buyer agent cannot ask the same question more than once.
- *System simplicity:* There are seven locutions in this model. Only a set of locutions are permitted to utter in each stage of the dialogue. Agents take turns to make locutions in bipartite dialogues.
- *Computational Simplicity:* In our dialogue system, we use the COQ proof checker to check a proof, which reduces the complexity compared to generate a proof. The length of the dialogue resulting from the protocol is: $1 + 4|H| + 3$, where $|H|$ is the number of questions in the knowledge base of a buyer.

6 Related Work

The formal study of human argument and dialogue has been proposed for modeling agent interactions for more than a decade [20]. In the seminal work of Walton and Krabbe [29], a typology of primary dialogue types is provided to model human dialogues. Dialogues are categorized as six primary types:

Information-seeking Dialogues (where participants aim to exchange knowledge); *Inquiry Dialogues* (where participants aim to find or destroy a proof of hypothesis); *Persuasion Dialogues* (where participants aim to resolve conflicts of opinions); *Negotiation Dialogues* (where participants aim to make a deal on the conflicts of interests); *Deliberation Dialogues* (where participants aim to decide best available course of action) and *Eristic Dialogues* (where participants quarrel verbally that aim to vent grievances). Most of the dialogue among humans or agents may involve mixtures of these dialogue types, which are called embedded.

In formal dialogue games, players interact with each other by making utterances according to previously defined rules. In the work of McBurrney and Parsons [18], five dialogue game rules have been identified. These rules include: *Commencement Rules* which define the circumstance under which the dialogue commences; *Locutions*, which define the rules that indicate what are permitted; *Combination Rules*, which define the contexts under which locutions are permitted or not; *Commitments*, which define the rules that indicate the circumstances under which participants express commitment to a proposition; and *Termination Rules*, which define the circumstances under which the dialogue ends. FIPA ACL [11] is a standard language for agents to communicate in. FIPA ACL contains 22 locutions (e.g., inform, request, refuse and agree) which make agents share knowledge and negotiate contracts. However, FIPA ACL does not support argumentation statements. A protocol named *Fatio* with five locutions (*assert, question, challenge, justify and retract*) has been proposed for argumentation by [19]. In our work, we have presented a dialogue model, which is in accordance with the rules for a dialogue game. The difference between our work and the protocol of *Fatio* is that we focus on the type of inquiry dialogue. In [8], a formal inquiry dialogue system based on Defeasible Logic Programming [12] has been presented. In this inquiry dialogue system, three locutions *open*, *assert* and *close* are defined for generating dialogues. Besides, they also define an exhaustive strategy to decide which move to make. We defined our inquiry dialogue by adding more locutions to the model and we also introduced the idea of attack to enable agents to reject unsatisfied justifications. In the work of [24], an inquiry dialogue has been proposed to enable agents to negotiate over ontological correspondences. In this dialogue, agents can not only make *assert* moves to assert beliefs, they also can *object* to a belief by providing an attack and *accept* or *reject* beliefs. In the ArguGRID [27] project, Web service, agents and argumentation technique have been combined to support decision making and negotiations inside Virtual Organizations. ArgSCIFF [28] is a project that aims to make Web service reasoning more visible to potential users by using dialogues for service interaction. It extends the kind of request-response interaction among Web services and makes agents justify the interaction outputs. The difference between their work and our approach is that we utilize the PCC paradigm to the dialogue to enable agents to automatically check formal proofs that provided by the service provider. An automate negotiation approach has been presented among agents in an open environment in [26]. In this work, protocols are expressed in terms of a shared ontology. Based on the shared ontology, agents can learn to tune strategies based on existing algorithms. We extend their work by introducing dialogue games into the scenario. Besides，dialogue games has been applied for shared decision making in a human-robot team [1]. In this paper, we extend our previous work [2] for decision making in the scenario of automatic online trading.

In our previous work, we have represented some properties (e.g., truthful bidding is a dominant strategy in a Vickrey auction) in the theorem prover COQ [5]. We have implemented the PCC [23] paradigm to certify the desirable properties of online auction services, which are written in OWL-S. PCC is a paradigm

that enables computer system to automatically ensure that a computer code provided by a foreign agent is safe for installation and execution. We use the interactive theorem prover COQ because it has been developed for more than twenty years [10] and is widely used for formal proof development in a variety of contexts related to software and hardware correctness and safety. COQ has been used to develop and certify a compiler [16], and a fully computer-checked proof of the Four Colour Theorem has been created by [13]. In the work of [4], we have formalized an auction service using OWL-S, then we translate this representation into a program that is written in COQ. The semantics of these expressions is preserved in this transaction. The, we use proof tactics to prove desirable properties of these expressions within the theorem prover COQ.

7 Summary

In this paper, we have proposed a dialogue game to enable buyer agents to automatically query an auctioneer before deciding whether to join an online auction. We have formally described this inquiry dialogue model by defining the rules of locutions and commitments. The auctioneer and the buyer share a common knowledge enabling them to communicate and make sense of each other's arguments. But they also have private knowledge. For example, the questions related to properties of interest to the buyer are not known to the auctioneer until they have been revealed through the dialogue. The buyer has a ranking function over the questions in the form of a preference level, which is used to drive the dialogue forward. A scoring function over possible answers in line with predefined expectations is used to decide whether to join or not an auction.

A noticeable feature of our dialogue game is that it uses formal proofs as arguments. Thus, it combines formal evidence with informal evidence in an interaction. We have implemented our dialogue framework from within JADE wherein we have integrated the COQ theorem prover in the Proof-Carrying Code paradigm enabling an agent to check whether a proof of a given auction desirable property is correct or not. Experimental results have demonstrated the feasibility as well as the validity of our approach. Future work includes the extension of the range of desirable properties to be proven in the system and extensive evaluation of our approach. More importantly, we would like to simulate virtual markets with these kind of intelligent software agents to shed some light on real life markets namely our human attitudes towards online shopping.

REFERENCES

[1] Azhar, M.Q. and E.I. Sklar. *Analysis of empirical results on argumentation-based dialogue to support shared decision making in a human-robot team*. in *Robot and Human Interactive Communication (RO-MAN), 2016 25th IEEE International Symposium on*. 2016. IEEE.

[2] Bai, W., E. Tadjouddine, and T. Payne. A Dialectical Approach to Enable Decision Making in Online Trading. in European Conference on Multi-Agent Systems. 2015. Springer International Publishing.

[3] Bai, W. and E.M. Tadjouddine, *Automated program translation in certifying online auctions.* ETAPS/VpPT, 2015: p. 11-18.

[4] Bai, W., E.M. Tadjouddine, and Y. Guo, Enabling Automatic Certification of Online Auctions. arXiv preprint arXiv:1404.0854, 2014.

[5] Bai, W., et al. A proof-carrying code approach to certificate auction mechanisms. in International Workshop on Formal Aspects of Component Software. 2013. Springer.

[6] Bellifemine, F.L., G. Caire, and D. Greenwood, Developing multi-agent systems with JADE. Vol. 7. 2007: John Wiley & Sons.

[7] Berners-Lee, T., J. Hendler, and O. Lassila, The semantic web. Scientific american, 2001. 284(5): p. 28-37.

[8] Black, E. and A. Hunter, An inquiry dialogue system. Autonomous Agents and Multi-Agent Systems, 2009. 19(2): p. 173-209.

[9] Christensen, E., et al., Web services description language (WSDL) 1.1, 2001.

[10] Dowek, G., et al., The COQ proof assistant user's guide: Version 5.6, 1991, INRIA.

[11] Fipa, A., Fipa acl message structure specification. Foundation for Intelligent Physical Agents, http://www. fipa. org/specs/fipa00061/SC00061G. html (30.6. 2004), 2002.

[12] García, A.J. and G.R. Simari, Defeasible logic programming: An argumentative approach. Theory and practice of logic programming, 2004. 4(1+ 2): p. 95-138.

[13] Gonthier, G., The four colour theorem: Engineering of a formal proof, in Computer Mathematics2008, Springer. p. 333-333.

[14] Horrocks, I., et al. Semantic web architecture: Stack or two towers? in International Workshop on Principles and Practice of Semantic Web Reasoning. 2005. Springer.

[15] Horrocks, I., et al., SWRL: A semantic web rule language combining OWL and RuleML. W3C Member submission, 2004. 21: p. 79.

[16] Leroy, X., Formal verification of a realistic compiler. Communications of the ACM, 2009. 52(7): p. 107-115.

[17] Martin, D., et al., OWL-S: Semantic markup for web services. W3C member submission, 2004. 22: p. 2007-04.

[18] McBurney, P. and S. Parsons, Games that agents play: A formal framework for dialogues between autonomous agents. Journal of logic, language and information, 2002. 11(3): p. 315-334.

[19] McBurney, P. and S. Parsons. Locutions for argumentation in agent interaction protocols. in International Workshop on Agent Communication. 2004. Springer.

[20] McBurney, P. and S. Parsons, Dialogue games for agent argumentation, in Argumentation in artificial intelligence2009, Springer. p. 261-280.

[21] McBurney, P., S. Parsons, and M. Wooldridge. Desiderata for agent argumentation protocols. in Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1. 2002. ACM.

[22] McGuinness, D.L. and F. Van Harmelen, OWL web ontology language overview. W3C recommendation, 2004. 10(10): p. 2004.

[23] Necula, G.C., Proof-carrying code. design and implementation, in Proof and system-reliability2002, Springer. p. 261-288.

[24] Payne, T.R. and V. Tamma. Negotiating over ontological correspondences with asymmetric and incomplete knowledge. in Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems. 2014. International Foundation for Autonomous Agents and Multiagent Systems.

[25] Roman, D., et al., Web service modeling ontology. Applied ontology, 2005. 1(1): p. 77-106.

[26] Tamma, V., et al., Ontologies for supporting negotiation in e-commerce. Engineering applications of artificial intelligence, 2005. 18(2): p. 223-236.

[27] Toni, F., et al. The ArguGRID platform: an overview. in International Workshop on Grid Economics and Business Models. 2008. Springer.

[28] Torroni, P., M. Gavanelli, and F. Chesani, Argumentation in the semantic web. IEEE Intelligent Systems, 2007. 22(6).

[29] Walton, D. and E.C. Krabbe, Commitment in dialogue, 1995, State University of New York Press Albany.