Expressing Interaction in Combinatorial Auction through Social Integrity Constraints *

Marco Alberti[1], Federico Chesani[2], Marco Gavanelli[1], Alessio Guerri[2], Evelina Lamma[1], Paola Mello[2], and Paolo Torroni[2]

[1] Dip. di Ingegneria - Università di Ferrara - Via Saragat, 1 - 44100 Ferrara, Italy.
{malberti|mgavanelli|elamma}@ing.unife.it
[2] DEIS - Università di Bologna - Viale Risorgimento, 2 - 40136 Bologna, Italy.
{fchesani|aguerri|pmello|ptorroni}@deis.unibo.it

Abstract. Combinatorial Auctions are an attractive application of intelligent agents; their applications are countless and are shown to provide good revenues. On the other hand, one of the issues they raise is the computational complexity of the solving process (the Winner Determination Problem, WDP), that delayed their practical use. Recently, efficient solvers have been applied to the WDP, so the framework starts to be viable.
A second issue, common to many agent systems, is *trust*: in order for an agent system to be used, the users must *trust* both their representative and the other agents inhabiting the society. Malicious agents must be found, and their violations discovered. The SOCS project addresses such issues, and provided a language, the *social integrity constraints*, for defining the allowed interaction moves, together with a proof-procedure able to detect violations.
In this paper we show how to write a protocol for the combinatorial auctions by using social integrity constraints. In the devised protocol, the auctioneer interacts with an external solver for the winner determination problem. We also suggest some solutions for a further, challenging issue: defining a protocol that contains the concept of *optimal allocation* and checking efficiently that the allocation proposed by the auctioneer is indeed optimal.

1 Introduction

Auctions have been practically used for centuries in human commerce, and their properties have been studied in detail from economic, social and computer science viewpoints. The raising of electronic commerce has pushed auctions as

* The authors wish to thank Michela Milano for her precious help. This work is partially funded by the Information Society Technologies programme of the European Commission under the IST-2001-32530 project in the context of the Global Computing initiative of the FET (Future Emerging Technology) initiative and by the MIUR COFIN 2003 projects *La Gestione e la negoziazione automatica dei diritti sulle opere dell'ingegno digitali: aspetti giuridici e informatici* and *Sviluppo e verifica di sistemi multiagente basati sulla logica*.

one of the favourite dealing protocols in the Internet. Now, the software agent technology seems an attractive paradigm to support auctions [1]: agents acting on behalf of end-users could reduce the effort required to complete auction activities. Agents are intrinsically autonomous and can be easily personalised to embody end-user preferences. In addition, they are adaptive and capable of learning from both past experience and their environment, in order to cope with changing operating conditions and evolving user requirements [2]. In fact, while in the past bidders were only humans, recent Internet auction servers [3] allow software agents to participate in the auction on behalf of end-users, and some of them even have a built-in support for mobile agents [4]. As the rise of the Internet and electronic commerce continues, dynamic automated markets will be an increasingly important domain for agents.

Depending on the kind of auction, the auctioneer either sells a set of goods trying to maximise the profit, or buys a set of goods minimising the cost. Bidders have the goal to obtain (respectively, sell) the goods under convenient price conditions.

Combinatorial auctions are types of auctions that give more expressiveness to the bidders: in fact, bidders can place bids on sets of items, expressing complementarity and, in some cases, also substitutability among the items [5,6]. On the other hand, determining the winners is an NP-hard problem, so combinatorial auctions were not typically used due to the lack of efficient solvers. Recently, however, various solvers have been developed, that are able to solve the winner determination problem in reasonable time.

Of course, another issue in e-commerce and, in particular, in electronic auctions, is *trust* [7]. Amongst the various aspects of trust in MASs (often related to credibility levels between agents), we find utterly important that human users trust their representatives: in order for the system to be used at all, each user must trust its representative agent in the auction. The agent must be well specified, and a formal proof of a correspondence between specification and implementation is, at least, desirable. Also, even if the agents are compliant to their specifications, the compliance to the social rules and protocols must be provable, in order to avoid, or, at least, detect malicious behaviours.

A typical answer to such issues is to model-check the agents with respect to both their specifications and requirements coming from the society. However, this is not always possible in open environments: agents could join the society at all times and their specifications could be unavailable to the society. Thus, the correct behaviour of agents can be checked only from the external in an open environment: by monitoring the communicative actions of the agents.

The *SOCS* project [8] addresses these issues by providing formal definitions both for the agents, that are based on Computational Logics, and are thus called *Computees*, and for the society in an open environment.

In this paper, we focus on the societal aspects, and on the compliance of the computees (or, in general, agents) to protocols and social rules. These can be easily expressed in a logic language, the *Social Integrity Constraints* (ic_S) that

are an extension of the integrity constraints widely used in Abductive Logic Programming, and, in particular, extend those of the IFF proof-procedure [9].

We implemented an abductive proof-procedure, called $\mathcal{S}$CIFF (extending the IFF [9]), that is able to check the compliance to protocols and social rules given a history of communicative actions. Besides a posteriori check of compliance, $\mathcal{S}$CIFF also accepts dynamically incoming events, so it can check compliance during the evolution of the societal interaction, and raise violations as soon as possible. $\mathcal{S}$CIFF extends the IFF in a number of directions: it provides a richer syntax, it caters for interactive event assimilation, it supports fulfillment check and violation detection, and it embodies CLP-like constraints [10] in the ic_S. $\mathcal{S}$CIFF is sound [11] with respect to the declarative semantics of the society model, in its abductive interpretation. The $\mathcal{S}$CIFF has been implemented and integrated into a Java-Prolog-CHR based tool [12].

In this paper, we show a definition of the combinatorial auction protocol in Social Integrity Constraints. Since the solving process is NP-hard, we exploit an efficient solver for the Winner Determination Problem. Finally, we propose a challenging extension: defining a protocol conformance checking architecture that contains the concept of *optimal allocation* for the Winner Determination Problem. The protocol conformance architecture exploits again an efficient auction solver to check the optimality of an allocation.

The paper is structured as follows. In Section 2 we briefly recall the *SOCS* social model. We describe the combinatorial auction scenario in Section 3, and we propose other solutions in Section 4. Finally, we cite some related work and we conclude.

2 *SOCS* social model

We sketch, for the sake of readability, the *SOCS* social model; the reader is referred to previous publications for more details on the syntax and semantics of the language [14,15].

The society knowledge is physically memorised in a device, called the *Society Infrastructure*, that has reasoning capabilities and can use the society knowledge to infer new information. We assume that the society infrastructure is time by time aware of social events that dynamically happen in the environment (*happened* events). They are represented as ground atoms

$$\mathbf{H}(Event[, Time]).$$

Social events are typically communication actions amongst computees, but may also involve *objects*: computees may need to interact with servers that are in the environment, and that typically do not show the features that distinguish agents from other software entities (like autonomy or proactivity). Some of the objects could be certified by the society and be considered always trustable, i.e., their replies will be considered always correct (provided that the requests are correct). In human commerce there also exist certified objects: for example, a grocer has a certified balance that is used to prove to the buyer that the quantity of goods

is correct. The client will typically trust the balance, but has the right to check that the goods posed on the balance are indeed the requested ones.

The knowledge in a society is given by:

- a *Social Organisation Knowledge Base (SOKB)*: a logic program;
- a set $\mathcal{IC}_S$ of *Social Integrity Constraints* (ic_S): implications that can relate elements in the dynamic part, CLP constraints and predicates defined in the SOKB.

The "normative elements" are encoded in the ic_S. Based on the available history of events, and on the ic_S-based specification, the society can define what the "expected social events" are, i.e., what events are expected (*not*) to happen. The expected events, called *social expectations*, reflect the "ideal" behaviour of the agents. Social expectations are represented as atoms $\mathbf{E}(Event[, Time])$ for events that are expected to happen and as $\mathbf{EN}(Event[, Time])$ for events expected not to happen.

While **H** atoms are always ground, the arguments of expectations can contain variables. Intuitively, an $\mathbf{E}(X)$ atom indicates a wish about an event $\mathbf{H}(Y)$ which unifies with it: X/Y. CLP constraints [10] can be imposed on the variables occurring in expectations.

For instance, in an auction context the atom:

$$\mathbf{E}(tell(Bidder, Auctioneer, bid(ItemList, Price), D), T_{bid})$$

stands for an expectation about a communicative act *tell* made by a computee ($Bidder$), addressed to another computee ($Auctioneer$), at a time T_{bid}, with subject $bid(ItemList, Price)$. Although **H**, **E**, and **EN** atoms can contain any term as argument, in this paper we will use the performative

$$tell(Sender, Receiver(s), Content, D).$$

The addressee of *tell* can be one or more receivers. The dialogue identifier (D), although not always important, can be useful to distinguish different instances of the same interaction scheme; for example, there could be two (or more) auctions in parallel, so one may want to distinguish the auction instance the bid refers to.

3 The Combinatorial Auctions scenario

There exist different kinds of combinatorial auctions. In this paper, we consider *single unit auctions*. In a *single unit auction*, the auctioneer wants to sell a set M of goods/tasks maximising the profit. Goods are distinguishable. Each bidder j posts a bid B_j where a set S_j of goods/tasks $S \subseteq M$ is proposed to be bought at the price p_j, i.e., $B_j = (S_j, p_j)$.

3.1 The Auction Solver

Besides the usual constraints of a combinatorial auction (i.e., two winning bids cannot have elements in common), some real-life auction scenarios also have the so called *side constraints*: other constraints that should be satisfied by the winning bids. One typical example is when the auctioneer needs to allocate tasks, that have precedence relations. Bidders propose to execute (bunches of) tasks, each with an associated time window, at a given price. The auctioneer will try to find a set of tasks that will cover the whole manufacturing process satisfying the time precedence constraints and minimising the cost.

The Winner Determination Problem in combinatorial auctions is NP-hard [16], so it cannot be addressed naively, but we need to exploit smart solving techniques. While the pure WDP is best solved with an Integer Programming (IP) solver [17], adding side constraints makes a Constraint Programming (CP) solver more appealing.

We address the problem by exploiting a module called *Auction solver* [18] that embeds two different algorithms both able to solve efficiently the WDP: one is a pure IP-based approach, and the other is an Hybrid approach based on a CP model with a variable selection heuristic based on the variables reduced costs deriving from the Linear Relaxation of the IP model. For a complete description of the IP and CP models, see [19].

The results obtained using the two algorithms strongly depend on the instance structure. The module embeds an automatic Machine Learning based portfolio selection algorithm, able to select the best algorithm on the basis of few structural features of the problem instance. Guerri and Milano [19] show that the method is able to select the best algorithm in the 90% of the cases, and that the time spent to decide which of the available algorithms fits best with the current instance is always order of magnitude lower with respect to the search time difference between the two algorithms. This is a fundamental assumption; in fact, if the sum of the times used to extract the features and to solve the problem with the best algorithm is greater than the time used by the worst algorithm to solve the same problem, the algorithm selection tool becomes completely useless.

So, one of the key elements in our architecture is the Auction Solver, that can be integrated in a number of different ways. We first give the general auction protocol in terms of ic_S in Section 3.2, then we propose one such scenario, and outline some alternative solutions that will be investigated in the future.

3.2 Auction Protocol

To start with, we can use ic_S to check that the auction protocol is indeed respected, without caring for the NP-hard aspects of the protocol. I.e., we will not check that the result provided by the auctioneer is indeed the optimal solution of the WDP.

At this level, the auction protocol is the one depicted in Figure 1. If a computee wants to sell a list of items, it can declare they are for sale with an

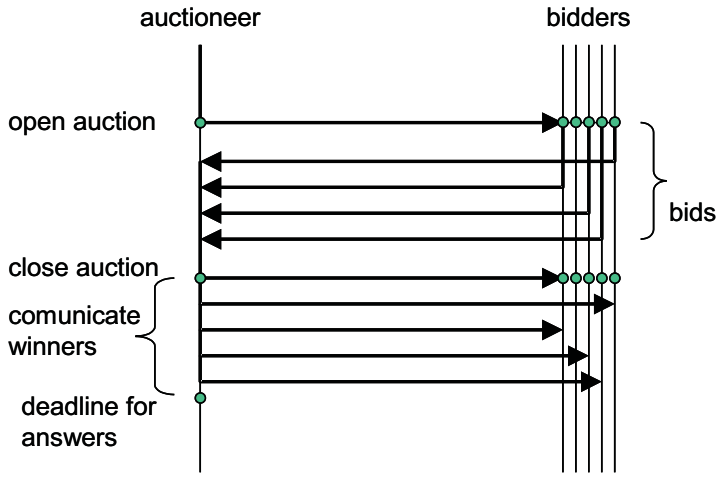


Fig. 1. Auction Protocol

openauction:

$$\mathbf{H}(tell(Auc, Bidders, openauction(ItemList, T_{end}, T_{Deadline}), D), T_{open}),$$

containing, as parameters, the deadlines for sending valid bids (T_{end}), and for the winners declaration ($T_{deadline}$). In an open society, bidders can come without registration, so the addressee of the *openauction* is not fundamental (bidders could join even if not explicitly invited with an *openauction* from the auctioneer, but they might have received the call from other computees, or from a blackboard). Of course, in semi-open societies [20], where there is a *gatekeeper* filtering the agents that want to enter the society, more ic_S can be imposed to define a registration protocol and a gatekeeper role.

After the *openauction*, bidders can stat placing bids, i.e., sending messages to the auctioneer declaring the subset of the items they are interested in ($ItemList$), and the price (P) they are willing to pay for such a set:

$$\mathbf{H}(tell(Bidder, Auc, bid(ItemList, P), D), T_{bid}).$$

The auction terminates a time T_{end}, and the auctioneer declares the auction closed:

$$\mathbf{H}(tell(Auc, Bidder, closeauction, D), T_{end}).$$

Finally, the auctioneer sends to each of the bidders the result of the WDP by uttering either *win* or *lose*:

$$\mathbf{H}(tell(Auc, Bidder, answer(win/lose, Bidder, ItemList, P), D), T_{answer}).$$

We can now show how the protocol is defined by means of Social Integrity Constraints.

The auction protocol in Social Integrity Constraints. Each time a bidding event happens, the auctioneer should have sent an *openauction* event:

$$\begin{array}{l}\mathbf{H}(tell(Bidder, Auc, bid(\_, \_), D), T_{bid}) \rightarrow \\ \mathbf{E}(tell(Auc, \_, openauction(\_, T_{end}, \_), D), T_{open}) \wedge \\ T_{open} < T_{bid} \wedge T_{bid} \leq T_{end}\end{array} \quad (1)$$

Incorrect bids always lose; e.g., a bid for items not for sale must lose. Indeed, the answer lose refers also to not acceptable bids.

$$\begin{array}{l}\mathbf{H}(tell(Auc, \_, openauction(Items, \_, \_), D), \_) \wedge \\ \mathbf{H}(tell(Bidder, Auc, bid(ItemBids, P), D), \_) \wedge \\ not\ included(ItemBid, Items) \\ \rightarrow \mathbf{E}(tell(Auc, Bidder, answer(lose, Bidder, ItemBids, P), D), \_)\end{array} \quad (2)$$

$$\begin{array}{l}included([], \_). \\ included([H|T], L) :- member(H, L), included(T, L).\end{array}$$

The auction should be closed at time T_{end}

$$\begin{array}{l}\mathbf{H}(tell(Auc, Bidder, openauction(\_, T_{end}, \_), D), \_) \\ \rightarrow \mathbf{E}(tell(Auc, Bidder, closeauction, D), T_{end})\end{array} \quad (3)$$

The auctioneer should answer to each bid. The answer should be sent after the auction is closed within the deadline $T_{deadline}$.

$$\begin{array}{l}\mathbf{H}(tell(Bidder, Auc, bid(ItemList, P), D), \_) \wedge \\ \mathbf{H}(tell(Auc, \_, openauction(\_, T_{end}, T_{deadline}), D), \_) \\ \rightarrow \mathbf{E}(tell(Auc, Bidder, answer(Ans, Bidder, ItemList, P), D), T_{answer}) \\ \quad \wedge T_{answer} > T_{end} \wedge T_{answer} < T_{deadline} \wedge Ans :: [win, lose]\end{array} \quad (4)$$

A bidder should not receive for the same auction on the same bid two conflicting answers:

$$\begin{array}{l}\mathbf{H}(tell(Auc, Bidder, answer(Ans_1, Bidder, ItemList, P), D), \_) \\ \rightarrow \mathbf{EN}(tell(Auc, Bidder, answer(Ans_2, Bidder, ItemList, P), D), \_) \\ \qquad \wedge Ans_1 \neq Ans_2\end{array} \quad (5)$$

Two different winning bids cannot contain the same item:

$$\begin{array}{l}\mathbf{H}(tell(Auc, Bidder_1, answer(win, Bidder_1, ItemList_1, \_), D), \_) \\ \wedge \mathbf{H}(tell(Bidder_2, Auc, bid(ItemList_2, P_2), D), \_) \\ \wedge Bidder_1 \neq Bidder_2 \\ \wedge intersect(ItemList_1, ItemList_2) \rightarrow \\ \mathbf{EN}(tell(Auc, Bidder_2, answer(win, Bidder_2, ItemList_2, P_2), D), \_)\end{array} \quad (6)$$

$$\begin{array}{l}intersect([X|\_], L) :- member(X, L). \\ intersect([\_|Tx], L) :- intersect(Tx, L).\end{array}$$

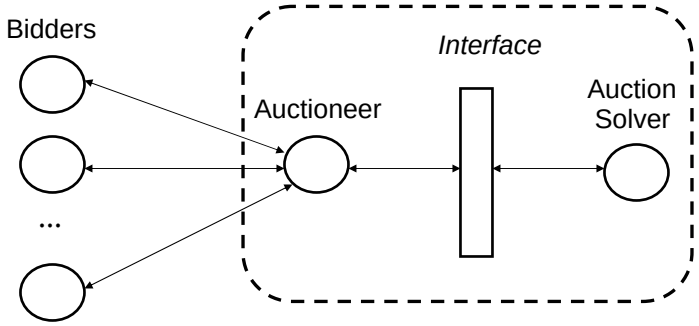


Fig. 2. Computees and object involved in a Combinatorial Auctions

Example. Suppose that a bidder tries to force the auctioneer to sell an item that was not offered in the *openauction*; e.g., the history could be:

$$\mathbf{H}(tell(auct, bidder_1, openauction([pc, monitor, mouse], 10, 20), 1), 1)$$
$$\mathbf{H}(tell(bidder_1, auct, bid([keyboard], 10€), 1), 3)$$

In this case, the protocol defines the correct behaviour of the auctioneer: it raises the expectation

$$\mathbf{E}(tell(auct, bidder_1, answer(lose, bidder_1, [keyboard], 10€), 1, \_).$$

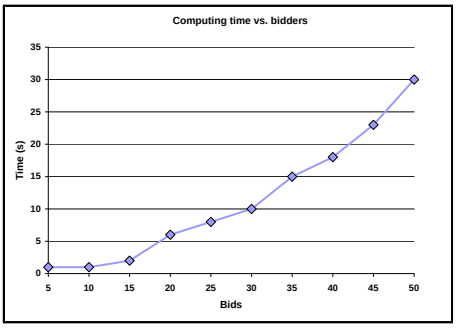
If the auctioneer does not give a matching reply, the proof-procedure will raise a violation.

3.3 Implementation with the Auction Solver

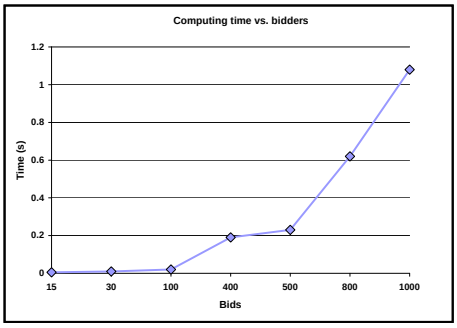
In this scenario, the Auction Solver is conceived as a passive object while the auctioneer is a computee. Their interaction is not monitored by the society, therefore at society level they are indeed considered as a unique entity as shown by the dashed circle in Figure 2. Obviously, more than one auctioneer can cohabit in the same society, each of them conceptually embedding its own instance of the Auction Solver object.

The auctioneer provides the auction solver a WDP instance and receives its optimal solution. The auction solver has been wrapped into Java, so as to define a simple interface for providing a problem instance to the solver and receive the solution of the problem. In this case the auctioneer should collect the data of the instance and send them to the auction solver. In a sense, in this case we are inserting part of the auction protocol (the optimality part) in the knowledge base of the auctioneer, similar to the idea of protocol conformance enforcing given by Endriss et al. [21].

We have some preliminary results for this scenario, shown in Figure 3, where the computation times of the $\mathcal{S}$CIFF proof-procedure and of the Auction Solver



(*a*) Time taken by the $\mathcal{S}$CIFF proof-procedure to verify compliance



(*b*) Time taken by the Auction Solver to solve the WDP

Fig. 3.

are shown. Experiments were performed on a 2.4 GHz Intel Pentium 4 with 512 Mb RAM. In [18] the authors show that the Auction Solver module outperforms, both in search time and in anytime solution quality, any other commercial solver in a WDP with temporal precedence constraints.

4 Other possible solutions

There are obviously other solutions for this problem, that will be taken into account in future work. In particular, we can, at society level, monitor the communications between the auctioneer and the auction solver.

Since the auctioneer is autonomous, it might decide to take a sub-optimal decision, for various reasons: e.g., the auctioneer might have been bribed by some bidders to make them win. Since the WDP is hard, we assume that only a specific solver can efficiently find the optimum: it is unlikely that the auctioneer can find it within the given deadlines without relying on the auction solver. It could be sensible to have, in the society, the auction solver as a *trusted object*: its replies are supposed to be correct. If the society does not trust the auctioneer but only the passive object *auction solver*, we have various possibilities; we show two of them.

In the first, the auctioneer is separated from the auction solver, and the society checks (besides the compliance to the protocol given in Section 3.2), that the auctioneer sends indeed to the auction solver the same bids it receives (not a fake problem in which, e.g., some bids were excluded to make some bidder win) and that the replies of the auctioneer are indeed the same computed by the solver (Figure 4):

$$\begin{aligned}\mathbf{H}(tell(Bidder, Auc, bid(ItemList, P), D), \_) \rightarrow \\ \mathbf{E}(tell(Auc, solver, bid(ItemList, P), D), \_)\end{aligned} \tag{7}$$

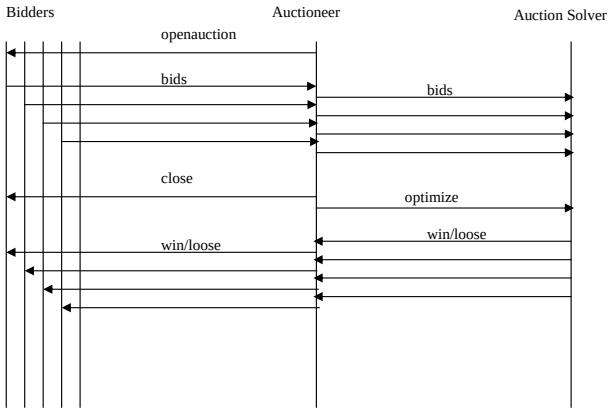


Fig. 4. Auction protocol when the society checks the communications between the auctioneer and the auction solver

$$\mathbf{H}(tell(solver, Auc, answer(Ans, Bidder, ItemList, P), D), \_) \rightarrow \\ \mathbf{E}(tell(Auc, Bidder, answer(Ans, Bidder, ItemList, P), D), \_) \quad (8)$$

In the second solution, we do not want the society to force the auctioneer to use a given object, or algorithm, to solve the WDP, but we leave it free to choose autonomously how to solve the problem. It might use our solver, or it might do it in other ways. In such a case, however, the auctioneer must face an external check: a controller can check that the provided solution is indeed correct. In order to check the validity of a solution, a computee takes the role of *controller*, and uses the trusted object *auction solver* in order to prove to the society the correctness of the solution (Figure 5).

5 Related Work

Combinatorial Auctions are increasingly studied since they have the advantage that bidders can bid on combinations of items. This leads to more efficient allocation than traditional auctions where the bidders valuations are only additive. The drawback is that evaluating bids and determining the winning bids is a NP-hard problem. However, there are different systems and methods to solve a combinatorial auction in an efficient way. The methods used to solve the problem exploit

- dynamic programming techniques [16]
- approximate methods that look for a reasonably good allocation of bids [22,23]
- integer programming techniques [24,5]

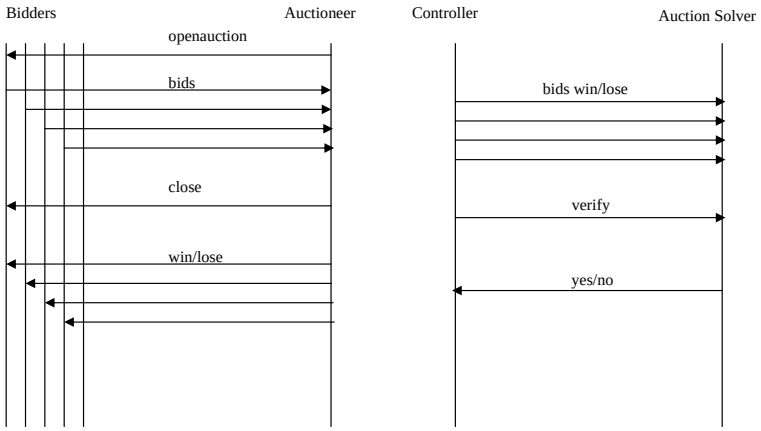


Fig. 5. Auction protocol: the society controls the auctioneer via a controller interacting with the auction solver

– search algorithms [6].

Various works are related to the $\mathcal{S}$CIFF proof procedure and the checking of compliance to protocols; a full discussion can be found in previous publications [14,15]. We will cite here ISLANDER [25], a tool to specify protocols in a system ruled by electronic institutions that has been applied to a Dutch auction (and other scenarios). Their formalism is multi-levelled: agents have *roles*, agents playing a role are constrained to follow *protocols* when they belong to a *scene*; agents can move from a scene to another by means of *transitions*. As in various works, protocols are defined by means of transition graphs, in a finite state machine. Our definition of protocols is wider than finite state machines, and leaves more freedom degrees to the agents. In our model, an event could be *expected to happen*, *expected not to happen* or have no expectations upon, thus there can be three possible states, while in finite state machines there are only two states. Moreover, they apply the model to the Dutch auction, while we focus on combinatorial auctions; this extends the possibilities, as widely documented in the literature on combinatorial auctions, but also makes the solving problem NP-hard. For this reason, a general purpose proof-procedure that checks the compliance to the protocol could be inefficient. We proposed a specialised solver and integrated it in our system.

6 Conclusions

Combinatorial auctions are recently starting to withdraw from the set of *practically unusable* applications as more efficient solvers are being produced for the winner determination problem. One of their natural applications involve intel-

ligent agents as both bidders and auctioneers, but this raises the problem of humans trusting their representatives, and the other agents in the society.

Through the tools provided by the *SOCS* project, we give means for the user to specify the fair and trusty behaviour, and a proof-procedure for detecting the unworthy and fallacious one. We defined the combinatorial auctions protocol through social integrity constraints, also exploiting an efficient solver for the winner determination problem.

Future work will concern trying other interaction schemes between the auction solver and the auctioneer agent; for example, by having a centralised auction solver that serves more auctioneers. We are also interested in continuing experimentation, in order to find how many auctioneers an auction solver can efficiently serve.

References

1. Chavez, A., Maes, P., Kasbah: An agent marketplace for buying and selling goods. In: Proceedings of the 1st International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology. (1996)
2. Guttman, R., Moukas, A., Maes, P.: Agent-mediated electronic commerce: A survey. Knowledge Engineering Review **13(2)** (1998) 143–147
3. Wurman, P., Wellman, M., Walsh, W.: The michigan internet auctionbot: A configurable auction server for human and software agents. In: Proceedings of the Second International Conference on Autonomous Agents (Agents-98). (1998)
4. Sandholm, T.: eMediator: a next generation electronic commerce server. In: Proceedings of the Fourth International Conference on Autonomous Agents (Agents-2000). (2000)
5. Nisan, N.: Bidding and allocation in combinatorial auctions. [26] 1–12
6. Sandholm, T.: Algorithm for optimal winner determination in combinatorial auction. Artificial Intelligence **135** (2002)
7. Marsh, S.: Trust in distributed artificial intelligence. In Castelfranchi, C., Werner, E., eds.: Artificial Social Societies. Number 830 in Lecture Notes in Artificial Intelligence, Springer-Verlag (1994) 94–112
8. SOCS: Societies Of ComputeeS: a computational logic model for the description, analysis and verification of global and open societies of heterogeneous computees. `http://lia.deis.unibo.it/Research/SOCS/`.
9. Fung, T.H., Kowalski, R.A.: The IFF proof procedure for abductive logic programming. Journal of Logic Programming **33** (1997) 151–165
10. Jaffar, J., Maher, M.: Constraint logic programming: a survey. Journal of Logic Programming **19-20** (1994) 503–582
11. Alberti, M., Gavanelli, M., Lamma, E., Mello, P., Torroni, P.: Specification and verification of interaction protocols: a computational logic approach based on abduction. Technical Report CS-2003-03, Dipartimento di Ingegneria di Ferrara, Ferrara, Italy (2003) Available at `http://www.ing.unife.it/aree_ricerca/informazione/cs/technical_reports`.
12. Alberti, M., Chesani, F., Gavanelli, M., Lamma, E., Mello, P., Torroni, P.: Compliance verification of agent interaction: a logic-based tool. [27] 570–575
13. Stathis, K., Kakas, A.C., Lu, W., Demetriou, N., Endriss, U., Bracciali, A.: PROSOCS: a platform for programming software agents in computational logic. [27] 523–528

14. Alberti, M., Gavanelli, M., Lamma, E., Mello, P., Torroni, P.: An Abductive Interpretation for Open Societies. In Cappelli, A., Turini, F., eds.: AI*IA 2003: Advances in Artificial Intelligence, Proceedings of the 8th Congress of the Italian Association for Artificial Intelligence, Pisa. Volume 2829 of Lecture Notes in Artificial Intelligence., Springer-Verlag (2003) 287–299 http://www-aiia2003.di.unipi.it.
15. Alberti, M., Gavanelli, M., Lamma, E., Mello, P., Torroni, P.: Modeling interactions using *Social Integrity Constraints*: A resource sharing case study. In Leite, J.A., Omicini, A., Sterling, L., Torroni, P., eds.: Declarative Agent Languages and Technologies. Volume 2990 of LNAI. Springer-Verlag (2004) 243–262 1st International Workshop (DALT 2003), Melbourne, Australia, July 15, 2003. Revised Selected and Invited Papers.
16. Rothkopf, M., Pekec, A., R.M.Harstad: Computationally manageable combinatorial auctions. Management Science **44** (1998) 1131–1147
17. Sandholm, T., Suri, S., Gilpin, A., Levine, D.: Cabob: A fast optimal algorithm for combinatorial auctions. In Nebel, B., ed.: Proceedings of the 17th International Joint Conference on Artificial Intelligence, Seattle, Washington, USA, Morgan Kaufmann Publishers (2001)
18. Guerri, A., Milano, M.: Exploring CP-IP based techniques for the bid evaluation in combinatorial auctions. In Rossi, F., ed.: Principles and Practice of Constraint Programming - CP 2003, 9th International Conference, CP 2003, Kinsale, Ireland, September 29 - October 3, 2003, Proceedings. Volume 2833 of Lecture Notes in Computer Science., Springer (2003) 863–867
19. Guerri, A., Milano, M.: Learning techniques for automatic algorithm portfolio selection. In Lopez de Mantaras, R., Saitta, L., eds.: Proceedings of the 16th European Conference on Artificial Intelligence, IOS Press (2004) to appear.
20. Davidsson, P.: Categories of artificial societies. In Omicini, A., Petta, P., Tolksdorf, R., eds.: Engineering Societies in the Agents World II. Volume 2203 of Lecture Notes in Artificial Intelligence., Springer-Verlag (2001) 1–9 http://link.springer.de/link/service/series/0558/papers/2203/22030001.pdf.
21. Endriss, U., Maudet, N., Sadri, F., Toni, F.: Protocol conformance for logic-based agents. In Gottlob, G., Walsh, T., eds.: Proceedings of the 18th International Joint Conference on Artificial Intelligence, Morgan Kaufmann Publishers (2003)
22. Fujishima, Y., Leyton-Brown, K., Shoham, Y.: Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In: Proceedings of the 16th International Joint Conference on Artificial Intelligence. Volume 1., Morgan Kaufmann Publishers (1999) 548–553
23. Sakurai, Y., Yokoo, M., Kamei, K.: An efficient approximate algorithm for winner determination in combinatorial auctions. [26] 30–37
24. Collins, J., Gini, M.: An integer programming formulation of the bid evaluation problem for coordinated tasks. In Dietrich, B., Vohra, R.V., eds.: Mathematics of the Internet: E-Auction and Markets. Volume 127 of IMA Volumes in Mathematics and its Applications. Springer-Verlag, New York (2001) 59–74
25. Sierra, C., Noriega, P.: Agent-mediated interaction. From auctions to negotiation and argumentation. In d'Inverno, M., Luck, M., Fisher, M., Preist, C., eds.: Foundations and Applications of Multi-Agent Systems, UKMAS Workshop 1996-2000, Selected Papers. Volume 2403 of Lecture Notes in Computer Science., Springer (2002) 27–48
26. Jhingran, A., ed.: Proceedings of the 2nd ACM Conference on Electronic Commerce (EC-00), October 17-20, 2000, Minneapolis, MN, USA. ACM Press (2000)

27. Trappl, R., ed.: Proceedings of the 17th European Meeting on Cybernetics and Systems Research, Vol. II, Symposium "From Agent Theory to Agent Implementation" (AT2AI-4). Austrian Society for Cybernetic Studies, Vienna, Austria (2004)