Expressing Interaction in Combinatorial Auction through Social Integrity Constraints *

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

[1] ENDIF - 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|mmilano|ptorroni}@deis.unibo.it

Abstract. Combinatorial auctions are an interesting application of intelligent agents. They let the user express *complementarity* relations among the items for sale, and let the seller obtain higher revenues. On the other hand, the solving process, the so-called Winner Determination Problem (WDP) is NP-hard. This restricted the practical use of the framework, because of the fear to be, in some WDP instances, unable to meet reasonable deadlines. Recently, however, efficient solvers have been proposed, 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. The SOCS project addresses such issues, and provided a language, the *social integrity constraints*, for defining the allowed interaction moves, and a proof-procedure able to detect violations.
In this paper we show how to write a protocol for 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 extensions of the scenario, with more auctions in a same society, and suggest to verify whether two auctions interact. We also apply the NetBill protocol for the payment and delivery scheme.

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 one

* A preliminary version of this paper appeared in [1]. 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.*

of the favourite dealing protocols in the Internet. Now, the software agent technology seems an attractive paradigm to support auctions [2]: 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. As the rise of the Internet and electronic commerce continues, dynamic automated markets will be an increasingly important domain for agents.

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 [3,4]. The main drawback is that determining the winners is an NP-hard problem. This delayed the practical applications of combinatorial auctions, mainly for fear not to meet the given deadlines. Recently, however, solvers able to solve the winner determination problem in reasonable time have been developed.

Of course, another issue common to many e-commerce applications is *trust* [5]. 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 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 [6].

We implemented an abductive proof-procedure, called $\mathcal{S}$CIFF (extending the IFF), 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 [7] in the

ic_S. $\mathcal{S}$CIFF is sound [8] 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 [9].

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 to extend the framework to check the compliance of two interacting auctions, in a double auction scheme.

The paper is the extended version of a previous informal publication [1]. In Section 2 we recall the *SOCS* social model. We describe the combinatorial auction scenario in Section 3). We present new work on extensions of the given scenario (Section 4.1), and on experimentation (Section 5). We cite some related work and, finally, 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 [10,11]. More details can also be found in another paper in this same volume [12].

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

The knowledge in a society is given by:

- a *Social Organisation Knowledge Base (SOKB)*: an abductive logic program with constraints;
- a set $\mathcal{IC}_S$ of *Social Integrity Constraints* (ic_S): implications that can relate dynamic elements, 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 $\mathbf{E}(Description[, Time])$ for events that are expected to happen and as $\mathbf{EN}(Description[, Time])$ for events expected not to happen. Explicit negation can be applied to expectations, letting the user express concepts like *possibility* ($\neg\mathbf{EN}$) and *optionality* ($\neg\mathbf{E}$).

While $\mathbf{H}$ 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 [7] can be imposed on the variables occurring in expectations, in order to refine the meaning of the expectation, and to improve the propagation. For instance, in an auction context the atom:

$$\mathbf{E}(tell(Bidder, Auctioneer, bid(ItemList, Price), D), T_{bid}), T_{bid} < 10$$

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 $bid(ItemList, Price)$. The expectation is fulfilled if a matching event actually happens, satisfying the imposed constraints (i.e., the deadline $T_{bid} < 10$). D is a dialogue identifier, that can be useful to distinguish different instances of the same interaction scheme.

Negative expectations, instead, suggest actions that should not happen, in order for the protocol to be fulfilled. For this reason, their variables are quantified universally (if they do not occur in positive expectations as well). Constraints can be imposed also on universally quantified variables, through a solver we implemented. For example, $\mathbf{EN}(Bidder, Auctioneer, bid(ItemList, Price), D), T_{bid})$, $T_{bid} > 10$ means that no bidder is supposed to send any bid to any auctioneer after time 10. Another paper in this book [12] describes the implementation of the proof-procedure that gets happened events and raises such expectations.

3 The Combinatorial Auctions scenario

There exist different kinds of combinatorial auctions. In this paper, we consider *single unit reverse 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 $S_j \subseteq M$ is proposed to be bought at the price p_j, i.e., $B_j = (S_j, p_j)$. In a *reverse auction*, the auctioneer tries to buy a set of goods minimising the total cost.

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 [13], 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 [14], adding side constraints makes a Constraint Programming (CP) solver more appealing.

We address the problem by exploiting a module called *Auction solver* [15] 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 [16].

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 [16] 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 orders 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 was greater than the time used by the worst algorithm to solve the same problem, the selection tool would be useless.

We now give the general auction protocol in terms of ic_S in Section 3.2.

3.2 Auction Protocol

We first describe the auction protocol, then we give its implementation with social integrity constraints. The auction is declared open by

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

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*).

After the *openauction*, each bidder can place bids, i.e., declare the subset of the items it is interested in ($ItemList$), and the price (P) it is willing to pay:

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

Finally, the auctioneer replies to each of the bidders either *win* or *lose*:

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

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

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

Incorrect bids always lose; e.g., a bid for items not for sale must lose.

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

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

The auctioneer should answer to each bid within the deadline $T_{deadline}$.

$$\begin{array}{l}\mathbf{H}(tell(Bidder, Auc, bid(ItemList, P), D), \_)\wedge \\ \mathbf{H}(tell(Auc, \_, openauction(\_, T_{end}, T_{deadline}), D), \_) \\ \rightarrow \mathbf{E}(tell(Auc, Bidder, answer(Ans, Bidder, ItemList, P), D), T_{answer}) \\ \quad \wedge T_{answer} > T_{end} \wedge T_{answer} < T_{deadline} \wedge Ans :: [win, lose]\end{array} \tag{3}$$

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

$$\begin{array}{l}\mathbf{H}(tell(Auc, Bidder, answer(Ans_1, Bidder, ItemList, P), D), \_) \\ \rightarrow \mathbf{EN}(tell(Auc, Bidder, answer(Ans_2, Bidder, ItemList, P), D), \_) \\ \qquad \wedge Ans_1 \neq Ans_2\end{array} \tag{4}$$

Two different winning bids cannot contain the same item:

$$\begin{array}{l}\mathbf{H}(tell(Auc, Bidder_1, answer(win, Bidder_1, ItemList_1, \_), D), \_) \\ \wedge\mathbf{H}(tell(Bidder_2, Auc, bid(ItemList_2, P_2), D), \_) \\ \wedge Bidder_1 \neq Bidder_2 \wedge intersect(ItemList_1, ItemList_2) \\ \rightarrow \mathbf{EN}(tell(Auc, Bidder_2, answer(win, Bidder_2, ItemList_2, P_2), D), \_)\end{array} \tag{5}$$

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

4 Extensions

4.1 Double combinatorial auction

We extended the scenario to allow for multiple auctions in a same society. We assume that all the items are labelled with a unique name. Clearly, in such a situation, we must ensure that a computee will not bid (i.e., try to sell) the same item in two different auctions, as he will be able to provide only one. We add to the previous ic_S (1) to (5), the following:

$$\begin{array}{l}\mathbf{H}(tell(B, A1, bid(ItemList1, P1), D_1), \_)\wedge \\ \mathbf{H}(tell(B, A2, bid(ItemList2, P2), D_2), \_)\wedge \\ D_1 \neq D_2 \wedge intersect(ItemList1, ItemList2) \rightarrow \\ false.\end{array} \tag{6}$$

This allows for interesting bidding strategies and behaviours of the computees, such as the *double combinatorial auction*. Suppose, for example, that a computee opens an auction for a set of items. Another computee, that owns most of the requested goods but not all of them, could try to make its bid more appealing, by trying to get all of the requested items, and sell them all together. Think, for example, to a collection of items, like comics: the price that a bidder can be able to obtain is much higher if he sells a complete collection, so he will try to buy the missing issues. Then, the bidder may become auctioneer of a second auction. Possible bidders know that the second auction is more appealing,

because the bidder that misses only a few issues will probably be prepared to pay more for the few missing issues than the first auctioneer.

In such a scenario, the society must ensure that a bidder will not try to sell an item he still does not own, so the end of the second auction should not be later than the time allowed in the first auction for placing bids; i.e.:

$$\begin{array}{l}\mathbf{H}(tell(A, B, openauction(Items1, T_{end1}, T_{deadline1}), \_), \_) \wedge \\ \mathbf{H}(tell(B, C, openauction(Items2, T_{end2}, T_{deadline2}), \_), \_) \wedge \\ intersect(Items1, Items2) \wedge T_{deadline2} \geq T_{end1} \rightarrow \\ false.\end{array} \tag{7}$$

4.2 Combinatorial auction with Netbill

This protocol extends the tradition auction protocol with the payment at the end. NetBill [17] is a security and transaction protocol for the sale and delivery of low-priced information goods, such as software or journal articles. The protocol rules transactions between two actors: *merchant* and *customer*. Accounts for merchants and customers, linked to traditional financial accounts (like credit cards), are maintained by a NetBill server. The implementation of the NetBill protocol in Social Integrity Constraints can be found in [18].

In our case, it was possible to simply add the ic_S defining the NetBill protocol to the auction integrity constraints (1) to (5).

5 Experiments

We performed different types of experiments in the combinatorial auction scenario. First, we have tested the Auction Solver implemented in ILOG to test its efficiency for increasing number of bidders. Then, we experimented the proof-procedure with the protocols defined in Sections 3 and 4.

All the experiments have been performed on a Pentium 4, 2.4 GHz, 512 MB.

5.1 Experiments on the Auction Solver

In the Combinatorial Auction scenario we have to cope with a complex combinatorial optimisation problem: the Winner Determination Problem. Having an efficient, scalable and flexible tool that solves this problem is crucial for the efficiency of the overall system.

In this experiment, we exploit an Auction Solver implemented in ILOG solver [19] suitably wrapped in to Java. The auction solver is able to solve both winner evaluation problems with and without temporal side constraints.

Although the focus of this paper is more on showing the feasibility of such an implementation than comparing with existing platforms, we have some comparisons (shown in Figure 5.1) of the Auction Solver with Magnet [20], both with respect to the Simulated Annealing (SA) and with the Integer Programming (IP) models of Magnet. In [15] more extensive experimentation is reported,

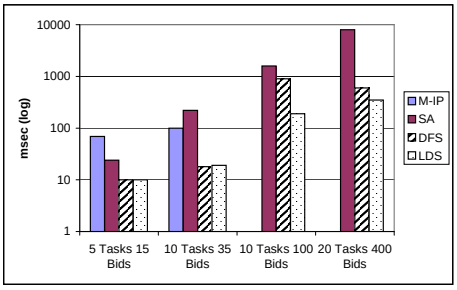


Fig. 1. Comparison with Magnet

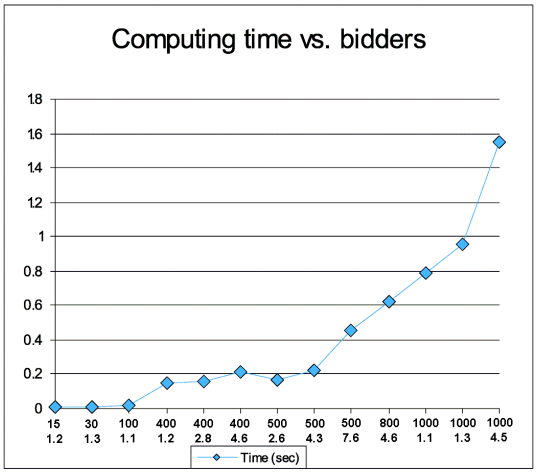


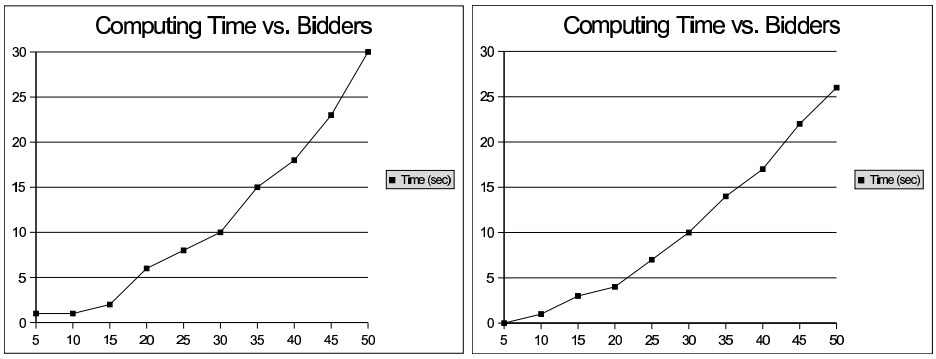
Fig. 2. Test of the Auction Solver performance

showing 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.

The Auction Solver is so efficient that it gives the possibility of scaling the auction size, and test the performances of the SOCS social infrastructure. Results are reported in Figure 2. We can see that auctions with 1000 bidders can be solved in less then 2 seconds. In the graph, we report the number of bids and the average number of tasks per bid. The results refer to the time for finding the optimal solution plus the proof of optimality.

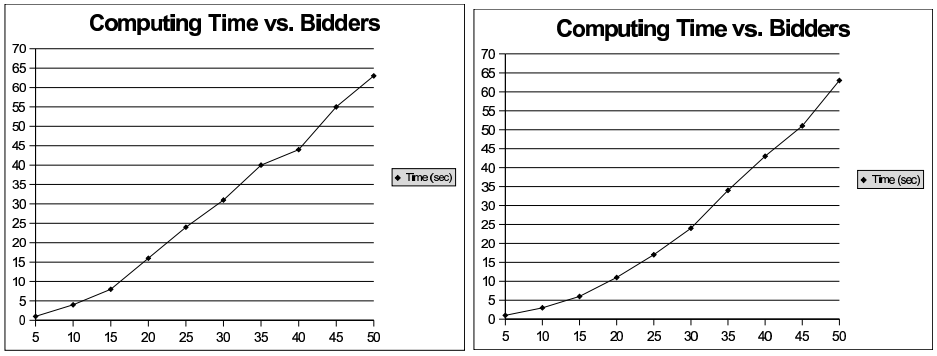
5.2 Experiments on the $\mathcal{S}$CIFF proof-procedure

We have arranged a set of experiments where the proof-procedure performances are tested for an increasing number of bidders. Bidders in fact send messages



(*a*) conformant protocol (*b*) non conformant protocol

Fig. 3. $\mathcal{S}$CIFF performance on a combinatorial auction



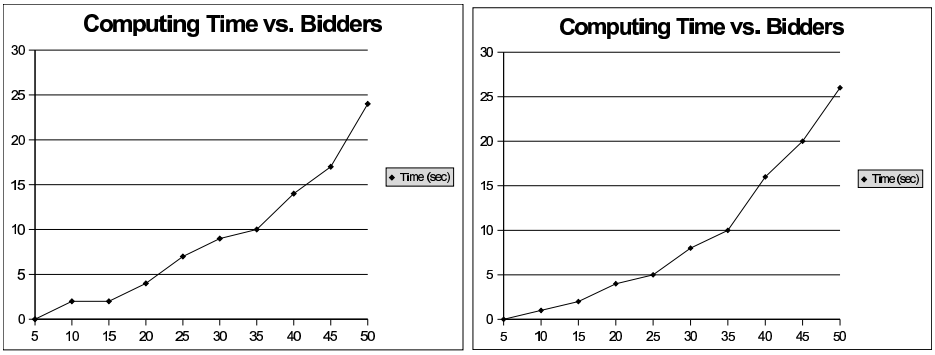
(*a*) conformant protocol (*b*) non conformant protocol

Fig. 4. $\mathcal{S}$CIFF performance on a double auction

which should be checked for conformance by the society and the more the number of bidders the more the number of messages to control. We have tested the two protocols described previously, plus an implementation that also considers the actual selling of the goods, through the NetBill protocol.

Concerning the single auction, we can see in Figure 3 that the tests provide an idea on how the proof scales for an increasing number of bidders and consequently of messages. We can see that results are good, since the prototype we implemented works well up to 50 bidders answering in half a minute.

As far as the double auction is concerned, it is intuitive that the number of exchanged messages is almost doubled with respect to a traditional combinatorial auction. We can see from the Figure 4 that the proof scales very well, being the time for testing conformance almost doubled with respect to a single auction.



(*a*) conformant protocol (*b*) non conformant protocol

Fig. 5. $\mathcal{S}$CIFF performance on an auction plus NetBill

As far as the auction plus the NetBill protocol, results are very good (Figure 5). In fact, the time for checking conformance is similar to that of the combinatorial auction.

As we can see from Figure 2 the Auction Solver is far more efficient since it scales up to 1000 bids within 2 seconds. Slower performances are achieved by the conformance checking of the society protocol (around 30 seconds for checking messages exchanged in auctions with 50 bidders). However, even if the components have different performances, from the testing, we can conclude that both components can be used in a real combinatorial auction scenario since time limits required for answering are much larger than sum of the answer times of the components.

Note that the implementation of the protocol currently considers only the combinatorial auction without temporal constraints. The full implementation with temporal constraints is left for future work.

6 Related Work

Different systems and methods have been proposed to solve a combinatorial auction in an efficient way, including dynamic programming techniques [13], approximate methods that look for a reasonably good allocation of bids [21,22], integer programming techniques [20,3], search algorithms [4].

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 [10,11]. We will cite here ISLANDER [23], 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*. 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 values, while in finite state machines there are only two. Also, we focused on *combinatorial* auctions; this provides nice features, widely documented in the literature, 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.

7 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 intelligent 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.

In future work, we will try other interaction schemes between the auction solver and the auctioneer agent; e.g., by having a centralised auction solver that serves more auctioneers. We are also interested in performing an extensive experimentation, to find how many auctioneers an auction solver can serve.

References

1. Alberti, M., Chesani, F., Gavanelli, M., Guerri, A., Lamma, E., Mello, P., Torroni, P.: Expressing interaction in combinatorial auction through social integrity constraints. In: Atti del Nono Convegno dell'Associazione Italiana per l'Intelligenza Artificiale, Perugia, Italia (2004)
2. Chavez, A., Maes, P.: Kasbah: An agent marketplace for buying and selling goods. In: Proc. of the 1^{st} International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM-96), London (1996) 75–90
3. Nisan, N.: Bidding and allocation in combinatorial auctions. [24] 1–12
4. Sandholm, T.: Algorithm for optimal winner determination in combinatorial auction. Artificial Intelligence **135** (2002) 1–54
5. Marsh, S.: Trust in distributed artificial intelligence. In Castelfranchi, C., Werner, E., eds.: Artificial Social Societies. Number 830 in LNAI, Springer-Verlag (1994)
6. Fung, T.H., Kowalski, R.A.: The IFF proof procedure for abductive logic programming. Journal of Logic Programming **33** (1997) 151–165
7. Jaffar, J., Maher, M.: Constraint logic programming: a survey. Journal of Logic Programming **19-20** (1994) 503–582

8. 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/informatica/tr/`.
9. Alberti, M., Chesani, F., Gavanelli, M., Lamma, E., Mello, P., Torroni, P.: Compliance verification of agent interaction: a logic-based tool. In Trappl, R., ed.: Proc. of the 17^{th} European Meeting on Cybernetics and Systems Research, Austrian Society for Cybernetic Studies (2004) 570–575
10. 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. Volume 2829 of LNAI, Springer-Verlag (2003) 287–299
11. 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
12. Alberti, M., Chesani, F., Gavanelli, M., Lamma, E.: The CHR-based implementation of a system for generation and confirmation of hypotheses. In Wolf, A., ed.: 19^{th} Workshop on (Constraint) Logic Programming. (2005) This volume.
13. Rothkopf, M., Pekec, A., R.M.Harstad: Computationally manageable combinational auctions. Management Science **44** (1998) 1131–1147
14. Sandholm, T., Suri, S., Gilpin, A., Levine, D.: Cabob: A fast optimal algorithm for combinatorial auctions. In Nebel, B., ed.: Proc. of IJCAI 01, (Morgan Kaufmann)
15. 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. Volume 2833 of LNCS, Springer Verlag (2003) 863–867
16. Guerri, A., Milano, M.: Learning techniques for automatic algorithm portfolio selection. In Lopez de Mantaras, R., Saitta, L., eds.: Proc. of ECAI. (2004)
17. Cox, B., Tygar, J., Sirbu, M.: NetBill security and transaction protocol. In: Proc. of the 1^{st} USENIX Workshop on Electronic Commerce, New York (1995)
18. Alberti, M., Gavanelli, M., Lamma, E., Mello, P., Torroni, P.: Specification and verification of agent interactions using social integrity constraints. Electronic Notes in Theoretical Computer Science **85** (2003)
19. ILOG S.A. France: ILOG Solver. 5.0 edn. (2003)
20. 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
21. Fujishima, Y., Leyton-Brown, K., Shoham, Y.: Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In Dean, T., ed.: Proc. of IJCAI 99, (Morgan Kaufmann) 548–553
22. Sakurai, Y., Yokoo, M., Kamei, K.: An efficient approximate algorithm for winner determination in combinatorial auctions. [24] 30–37
23. 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. Volume 2403 of LNCS, Springer Verlag (2002) 27–48
24. Jhingran, A., ed.: Proc. of the 2nd ACM Conference on Electronic Commerce (EC-00), October 17-20, 2000, Minneapolis, MN, USA. ACM Press (2000)