Verifying Dominant Strategy Equilibria in Auctions

Emmanuel M. Tadjouddine [1] & Frank Guerin [2]

CSD Report No AUCS/TR0704
Computing Sciences Department
University of Aberdeen
Aberdeen, AB24 3UE, UK
April 2007
{etadjoud,fguerin}@csd.abdn.ac.uk

Abstract

Future agent mediated eCommerce will involve open systems of agents interoperating between different institutions, where different auction protocols may be in use. We argue that in order to achieve this agents will need a method to automatically verify the properties of a previously unseen auction protocol; for example, they may wish to verify that it is fair and robust to deception. We are therefore interested in the problem of automatically verifying the game-theoretic properties of a given auction mechanism, especially the property of strategyproofness. In this paper we show how the `Alloy` model checker can be used to automatically verify such properties. We illustrate the approach via two examples: a simple two player Vickrey auction and a quantity restricted multi-unit auction using the Vickrey-Clarke-Groves mechanism.

1 Introduction

Future agent mediated eCommerce will involve open systems of agents interoperating between different institutions, where different auction protocols may be in use; the roaming agents will need to understand the rules of engagement in foreign institutions. This *semantic interoperation* for agents in different institutions is recognised as a "major challenge facing computer scientists" [7]. We look at the special case of agents engaged in auctions. Agent auctions have two aspects which we need to consider: (i) they specify a set of rules which the agents must follow, specifying when they are allowed to bid, how the winner should be determined and the payments to be made; (ii) they usually have certain desirable game theoretic properties such as incentive compatibility, encouraging agents to bid truthfully. Other properties can be *collusion-proofness* meaning agents cannot collude to achieve a certain outcome or *false-name bidding free* meaning agents cannot manipulate the outcome by using fictitious names. However, in here, we focus on the properties of a solution concept of the game, in particular dominant strategy equilibrium.

To enable agents to understand the rules of an unseen auction we need to consider both of these aspects. The first can be handled by agreeing on a standard language

[1] funded by UK EPSRC under grant EP/D02949X/1, Computing Sciences Department, University of Aberdeen

[2] Computing Sciences Department, University of Aberdeen

in which the rules of the auction can be written and published. There are a number of candidate languages in the literature [17, 9]. The second aspect is considerably more problematic. Understanding the rules of an auction is not enough; an agent needs to know some of the properties of the auction (e.g. individual rationality, incentive compatibility) in order to make a decision about whether or not to participate, and what strategy to use. There is also a recursive nature to this understanding of the game-theoretic properties; most game theoretic solutions rely on the equilibrium property of a game, and this equilibrium rests on common knowledge assumptions; if the common knowledge of the equilibrium is not achieved, then agents cannot expect it to be played [13].

Assuming that the auction mechanism is published along with a claim that it is strategyproof (i.e. truthful bidding is optimal), we aim to verify that truthful bidding is indeed strategyproof. Such verification is important for selfish and rational agents engaged in financial transactions. An agent cannot just trust the auctioneer and then choose the recommended strategy. To illustrate the verification procedure, we consider the Vickrey auction and a quantity-restricted multi-unit auction (QRMUA) and verify these equilibrium properties via the `Alloy` model checker based on first-order relational logic, see `http://alloy.mit.edu/`.

Related Work. In [28], it is argued that the triple KRA (Knowledge, Rationality, Action) is a powerful tool for reasoning on multi-agent systems. Among other examples, the idea of checking equilibria for normal form games with complete information by using the KRA triple and the backward induction technique [10, p. 58] is presented. In [22], a WHILE-language is extended to represent game theory mechanisms with complete information for multiple agents and a Hoare-like calculus with pre- and post-conditions is used to verify simple mechanisms such as the Solomon's dilemma or the Dutch auction. In [11, 12], the SPL(Simple Programming Language) [18] is extended into SMPL (Simple Mechanism Programming Language) in order to represent games in open multi-agent systems where agents have implicit preferences. The equilibrium is checked using an exhaustive algorithm that builds the entire game tree and uses a backward induction procedure to check equilibrium properties at each tree node. In [26], the game is constrained so as it can be represented as a set of Presburger formulae and the equilibrium property also expressed a Presburger formula is checked using fixed-point approach. The hereby cited work did not address the issues of carrying out a dominant strategy equilibrium verification for a given game.

The contributions of this paper are the followings. i) We have provided a polynomial time algorithm solving the QRMUA, integrated it into the VCG mechanism, and proved some properties on the revenue of the resulting pricing rule. ii) We have also shown by example that there are certain restricted cases of combinatorial auctions where it is feasible for agents to automatically check the relevant game theoretic properties, and hence that it is feasible for agents to interoperate between different auction houses and work with previously unseen specifications, for simple auctions.

In Section 2, we introduce our notations, and describe the combinatorial auction problem and the Vickrey-Clarke-Groves (VCG) mechanism. In Section 3, we describe how model checking can be used to verify the game theoretic properties of an auction.

In Section 4, we illustrate this approach using a simple Vickrey auction. In Section 5, we look at the case of a simple type of combinatorial auction; we formulate a winner determination algorithm for it, and check the game theoretic properties of a VCG implementation of it. Section 6 concludes.

2 Background and Model

In a *combinatorial auction*, there is a set I of m items to be sold to n potential buyers. A bid is formulated as a pair $(B(x), b(x))$ in which $B(x) \subseteq I$ is a bundle of items and $b(x) \in \mathbb{R}^+$ is the price offer for the items in B. Given a set of k bids, $X = \{(B(x_1), b(x_1)), (B(x_2), b(x_2)), \ldots, (B(x_k), b(x_k))\}$ the *combinatorial auction problem*(CAP) is to find a set $X_0 \subseteq X$ such that $\sum_{x \in X_0} b(x)$ is maximal, subject to the constraints that for all $x_i, x_j \in X_0 : \; B(x_i) \cap B(x_j) = \emptyset$ meaning an item can be found in only one accepted bid. We assume *free disposal* meaning that items may remain unallocated at the end of the auction.

Unfortunately the CAP is NP-hard [23]. Consequently, approximation algorithms [24, 25, 3] are used to find a near-optimal solution or restrictions are used to find tractable instances of the CAP [27] hence providing polynomial time algorithms for a restricted class of combinatorial auctions. A combinatorial auction can be *sub-additive* (for all bundles $B_i, B_j \subseteq I$ such that $B_i \cap B_j = \emptyset$, the price offer for $B_i \cup B_j$ is less than or equals to the sum of the price offers for B_i and B_j) or *super-additive* (for all $B_i, B_j \subseteq I$ such that $B_i \cap B_j = \emptyset$, the price offer for $B_i \cup B_j$ is greater than or equals to the sum of the price offers for B_i and B_j).

To set up the auction, we use a game theory *mechanism* [21, 5]: a decision procedure that determines the set of winners for the auction according to some desired objective. An objective may be that the mechanism should maximise the social welfare. Such a mechanism is termed *efficient*. For open multi-agent systems, another desirable property for the mechanism designer can be *strategyproofness* (truth telling is a dominant strategy). A mechanism that is strategyproof has a dominant strategy equilibrium. A well known class of mechanisms that is efficient and strategyproof is the Vickrey-Clarke-Groves (VCG), see for example [19] and [2]. The VCG mechanism is performed by finding (i) the allocation that maximises the social welfare and (ii) a pricing rule allowing each winner to benefit from a discount according to his contribution to the overall value for the auction. To formalise the VCG mechanism, let us introduce the following notations:

- $\mathcal{X}$ is the set possible allocations
- $v_i(x)$ is the true valuation of $x \in \mathcal{X}$ for bidder i
- $b_i(x)$ is the bidding value of $x \in \mathcal{X}$ for bidder i
- $x^* \in \text{argmax}_{x \in \mathcal{X}} \sum_{i=1}^{n} b_i(x)$ is the optimal allocation for the submitted bids.
- $x^*_{-i} \in \text{argmax}_{x \in \mathcal{X}} \sum_{j \neq i}^{n} b_j(x)$ is the optimal allocation if agent i were not to bid.

- u_i is the utility function for bidder i.

The VCG payment p_i for bidder i is defined as

$$\begin{aligned} p_i &= b_i(x^*) - \left(\sum_{j=1}^{n} b_j(x^*) - \sum_{j=1,\ j\neq i}^{n} b_j(x^*_{-i})\right) \\ &= \sum_{j=1,\ j\neq i}^{n} b_j(x^*_{-i}) - \sum_{j=1,\ j\neq i}^{n} b_j(x^*), \end{aligned} \tag{1}$$

and then, the utility u_i for agent i is a quasi-linear function of its valuation v_i for the received bundle and payment p_i.

$$u_i(v_i, p_i) = v_i - p_i.$$

Let p_i^* and p_i be the payments for agent i when it bids its true valuation v_i and any number b_i respectively. Note p_i, p_i^* are functions of b_{-i}. The strategyproofness of the mechanism amounts to the following verification:

$$\forall i,\ \forall v_i,\ \forall b_i\ u_i(v_i, p_i^*(b_{-i})) \geq u_i(v_i, p_i(b_{-i})). \tag{2}$$

In the equation (1), the quantity $\sum_{j=1, j\neq i}^{n} b_j(x^*_{-i})$ is called the *Clark tax*. Another interesting property of this VCG mechanism is that it is *weakly budget balanced* [4] meaning the sum of all payments is greater than or equal to zero. For the VCG properties (e.g. strategyproof) to hold, the auctioneer must solve $n+1$ hard combinatorial optimisation problems (the optimal allocation in the presence of all bidders followed by n optimal allocations with each bidder removed) exactly. In this work, we consider tractable instances of the CAP to which we apply the VCG mechanism. We formally specify the auction using a programming language, and then we verify some game-theoretic properties of the auction. These properties are simply first-order logic formulae within the auction model, e.g. the winner is always the highest bidder or the auction mechanism is incentive compatible or strategyproof.

3 Property Verification via Model Checking

To verify game-theoretic properties of a given auction, we use the `Alloy` model checker based on first order relational logic [15, 16, 8]. We then model the basic entities of the auction as well as the relations between them. Next we express the operations and the auction properties that we wish to check. To verify a property, the `Alloy` analyser starts by negating it and then looking for an instance where the negated property is true within a range of small number of models according to a user-defined scope. If the model is a correct representation of the auction, such an instance represents a counter-example to the auction property. If no counter-example is found, the negated formula may still be true in a larger scope since first-order logic is undecidable and the `Alloy` tool achieves tractability by restriction to a finite universe. It is worth noting the `Alloy` analysis is intractable asymptotically but according to Daniel Jackson's *small scope hypothesis* that states "Many bugs have small counter-examples". In other words, "negative answers tend to occur in small models already, boosting the confidence we may have in a positive answer" [14, p. 143].

4 A Motivating Example: The Vickrey Auction

In this section we motivate the problem by encoding the Vickrey auction [29] using the `Alloy` modelling language. We show how an agentcan employ simple IF-constructs to negate the well known game theoretic properties of the Vickrey auction, e.g. incentive compatibility; hence an agent would be able to check certain auction properties before entering it.

4.1 Representation

In a Vickrey auction, n agents bid for a single item. Each agent has a private valuation v of the item. The winner is the highest bidder, gets the item but pays the second highest bid p, getting the utility $u = v - p$. A losing bidder pays nothing and has a zero utility. It is well known that this mechanism is incentive compatible. We wish to enable the potential bidders to check such a claim for instance. Assuming a bid is a non negative integer, the Vickrey auction for two bidders can be modelled in `Alloy` as follows:

```
sig Bidder {
val: Int,
bid: some Int,
u: some Int,
w: Int
}


fact constr {
   all p: Bidder |
   int [p.w] >= 0 and  int [p.w] =< 1
   and int[p.bid]>=0 and int[p.bid]=<50
   and int[p.val]>=0 and  int[p.val]=<50
}


pred show (b:Bidder ) {
 #b.bid=1
}

run show
```

This model uses a built-in `Alloy` *signature* `Int` and defines the signature `Bidder`. The fields within the signature represent binary relations between signature types. For example, the field `val` is a binary relation mapping `Bidder` types to exactly one `Int`. The field `bid` maps `Bidder` types to one or more `Int`s. In short, a bidder has a single valuation `val`, can place a bid `bid` within a range of values to win (`w=1`) or lose (`w=0`) the auction with some utility value `u`. The fact `constr` encodes simple constraints on the model stating the range of the `Bidder`'s fields `bid, val, w` are $(0, 50)$, $(0, 50)$ and $(0, 1)$ respectively. The predicate `show` can be used to simulate the model.

4.2 Coding the Vickrey Mechanism

Once the basic entities of our system modelled, the winner determination algorithm `wda` as well as the utility function related to the Vickrey auction are coded using `Alloy` predicates.

```
pred wda(p1,p2,p1',p2':Bidder) {
 memcp[p1,p1']
 memcp[p2,p2']
 let x1=p1.bid, x2=p2.bid |
(int[x1] >= int[x2] ) =>
  int p1'.u=int[p1.val]-int[x2]
  and int[p2'.u] = 0
  and int[p1'.w]=1 and int[p2'.w]=0
else (
  int[p2'.u]=int[p2.val]-int[x1]
  and int[p1'.u] = 0
  and int[p1'.w]=0 && int[p2'.w]=1 )
 }


pred memcp (p1, p1': Bidder) {
  p1'.bid=p1.bid and p1'.val=p1.val
 }
```

The predicate `wda` takes as arguments two bidders `p1, p2` before the run of the auction and two bidders after `p1', p2'`. The body of this predicate calls another predicate `memcp` that copies the fields `bid, val` across two `Bidder` types. It says that the bids and values of the two bidders remained unchanged. However, the highest bidder wins the auction; his utility being the difference between his valuation and the loser's bid whilst the lowest bidder loses with a utility zero.

4.3 Checking Auction Properties

Having modelled the auction and its operations describing its dynamic behaviour, we can ask for example the following questions. Is the winner always the highest bidder? Is the mechanism strategyproof? Such properties can be checked using `Alloy` *assertions*, which are formulae the `Alloy` analyser can check whether they are valid by seeking their counter-examples.

Before encoding these equiulibrium properties in `Alloy`, let us express them in first order logic without the `Alloy` syntax. A bidder is an entity p of type `Bidder` having a single valuation val(p), a bid bid(p), a boolean variable $w(p)$ indicating if it wins or loses, and a utility $u(p)$ associated with the outcomes of the auction. Moreover, the winner determination algorithm as well as the utility function an be coded using predicates as seen in Section 4.2. The predicate wda(p_1, p_2, p'_1, p'_2) takes as arguments two bidders p_1, p_2 before the run of the auction and two bidders p'_1, p'_2 after. The bids and values of the two bidders remained unchanged. However, their utilities are computed according to the Vickery mechanism using a conditional expression with a test on who has the highest bid.

The assertion `winIsHighBidder` can then be expressed as follows:

$$\forall p_1, p_1', p_2, p_2' \texttt{:Bidder}, \mathrm{wda}(p_1, p_2, p_1', p_2') \wedge w(p_1') = 1 \rightarrow \mathrm{bid}(p_1') \geq \mathrm{bid}(p_2')$$

meaning after the run of the auction, the winner has the highest bid. The assertion `isDSE` for checking the strategyproofness can be expressed as follows:

$$\begin{aligned}&\forall p_1, p_2, p_2^t, p_1^* \texttt{:Bidder}, \forall p_1', p_2', p_2^{t'}, p_1^{*'} \texttt{:Bidder},\\ &\mathrm{bid}(p_1^*) = \mathrm{val}(p_1^*) \wedge \mathrm{val}(p_1^*) = \mathrm{val}(p_1) \wedge \mathrm{val}(p_2^t) = \mathrm{val}(p_2) \wedge \mathrm{bid}(p_2^t) = \mathrm{bid}(p_2)\\ &\wedge \mathrm{wda}(p_1, p_2, p_1', p_2') \wedge \mathrm{wda}(p_1^*, p_2^t, p_1^{*'}, p_2^{t'}) \rightarrow u(p_1^{*'}) \geq u(p_1').\end{aligned}$$

This considers 4 bidders p_1, p_2, p_1^*, p_2^t where p_2^t is a clone of p_2 and p_1, p_1^* have the same valuation but p_1^* bids truthfully and then checks the utilities of bidders p_1, p_1^* after the run of the auction, say $p_1', p_1^{*'}$.

In the `Alloy` modelling language, the above two assertions can be encoded as follows.

```
assert winIshighBidder {
all p1,p1',p2,p2':Bidder {
  wda[p1,p2,p1',p2'] && int[p1'.w] = 1
  => int[p1'.bid] >= int[p2'.bid]
  }
}

assert isDSE {
all p1,p2,p2t,p1s:Bidder,
  p1',p2',p2t',p1s':Bidder {
  p1s.bid=p1s.val && p1s.val=p1.val
  && p2t.val=p2.val && p2t.bid=p2.bid
  && wda[p1,p2,p1',p2']
  && wda[p1s, p2t,p1s',p2t']
  => int[p1s'.u] >= int[p1'.u]
  }
}

check isDSE for 50 but 4 Bidder
check winIsHighBidder for 4 but 2 Bidder
```

The `Alloy` assertion `winIsHighBidder` encodes a formula to check for the validity of the assertion: the winner is always the highest bidder. The assertion `isDSE` checks if the encoded model of the Vickrey mechanism is strategyproof. This is expressed by considering 4 bidders `p1, p2, p1s, p2t` where `p2t` is a clone of `p2` and `p1, p1s` have the same valuation but `p1s` bids truthfully and checks the utilities of `p1, p1s`. Eventually, we add *check* commands to verify each assertion within a finite scope. For example, the `isDSE` assertion is checked for 50 `Int` values and 4 `Bidder` types.

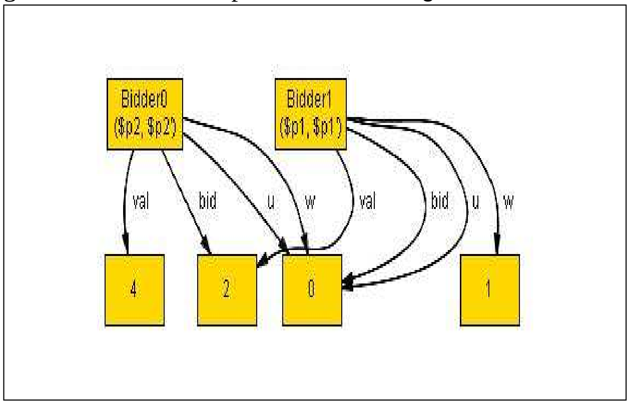
This analysis was carried out on a PC Pentium Dual Processor 2.99 GHz with 2GB RAM, running Windows XP. The two assertions were found to be valid within the specified scope, which makes us believe their correctness. To give an idea of the runtime, the `isDSE` analysis took 312 CPU seconds to complete. But how much data have been processed? With 50 `Int`, there are 50^4 possible values for each bidder. An exhaustive search for all 4 bidders give 50^{16} cases to consider. Fortunately, `Alloy` uses tree optimisation techniques that can discard large search subspaces without fully inspecting them [15, 16].

Interestingly, if we change the line `4` of the `wda` predicate by the following:

```
(int[x1]>=int[x2] || int[x2]=2) =>
```

then, neither of the two assertions is valid. For example, `Alloy` took 0.2 CPU second to find the counter-example shown by the diagram in Figure 1, for the model checking of `winIsHighBidder`. Figure 1 shows two bidders `Bidder0` valuing the item for 4 and bidding 2 and `Bidder1` valuing the item for 2 and bidding 0. `Bidder1` is the highest bidder but loses the auction as indicated by its field $(w{=}0)$.

Figure 1: counter-example for `winIsHighBidder` assertion



5 Quantity-Restriction in Multi-Unit Auctions

Quantity-Restriction in Multi-Unit Auctions (QRMUA) is a combinatorial auction in which each potential buyer submits bids for a number of identical items but restricts the number of items he wishes to buy. It has the property that the price offer for a bundle by a buyer is the sum of his bids on the individual items in the bundle. It is also sub-additive since a buyer will get for free any extra item allocated to him beyond the number of items he wished to obtain. Obviously, there is a maximum number of items to be allocated by the auctioneer. This is a special case of the quantity-restriction in multi-object auctions of [27], which is shown to be tractable by reducing it to a b-matching problem [6, p. 186] in a bipartite graph. The QRMUA can be illustrated by the auction example for reservation of seats in a particular flight given by [27].

Let m be the size of I e.g., the set of seats in the airplane and n the number of potential buyers with their required numbers of seats $m_1, m_2, \ldots, m_n$ respectively. The price offer for each bundle of m_i seats is $m_i * b_i$, where b_i is the price offer for a single seat. Let a_i be the number of seats allocated to agent i. The allocation consists in choosing disjoint subsets $\{B_i \subseteq I | i \in Winners\}$ of winning bids such that $\sum_{i \in Winners} a_i b_i$ is maximal, where $a_i = \min(|B_i|, m_i)$. This allocation problem can be formulated in terms of integer linear programming (ILP) as follows:

$$\max \sum_{i=1}^{n} a_i b_i \qquad \text{subject to} \quad \begin{cases} a_i \leq m_i, \ i = 1, n \\ \sum_{i=1}^{n} a_i \leq m \end{cases} \tag{3}$$

Because ILP is generally intractable, we may use a similar argument to that of [20] to relax the ILP to linear programming, which is tractable, and then show that the resulting solution is integral. We recover the tractability result by providing a simple polynomial algorithm (see Algorithm 1) that solves the optimisation problem of equation (3).

Our algorithm takes as inputs n bids $b_1, b_2, \ldots, b_n$, n values $m_1, m_2, \ldots, m_n$ representing the numbers of required seats for each bidder and the total number of seats m. It returns an allocation $a_1, a_2, \ldots, a_t$ that maximises the quantity $\sum_{i=1}^{n} a_i b_i$. This will enable us to easily encode QRMUA into the `Alloy` modelling language.

Algorithm 1 (Allocation for QRMUA)

Sort $b_i, i = 1, n$ in decreasing order
Assume $b_1 \geq b_2 \geq \ldots \geq b_n$
Set $i := 1;\ r := m;\ s := 0$
While $(r > 0$ and $i \leq n)$ Do
    If $r \geq m_i$ Then
        Set $a_i := m_i;\ r = r - a_i$
    Else
        Set $a_i := r;\ r := 0$
    End If
    Set $s := s + a_i b_i;\ i = i + 1$
End Do
Set $t := i - 1$
If $r > 0$ Then
    Choose randomly $j,\ 1 \leq j \leq t$
    Set $a_j := a_j + r$
End If
At the end, return the values $(a_i)_{i=1,t}$ and the sum s

Lemma 1 *Algorithm 1 correctly solves the ILP of equation (3) in $\mathcal{O}(n \log n)$ time.*

Proof: Algorithm 1 can sort the n bids in $\mathcal{O}(n \log n)$ using the quicksort procedure and then takes at most n steps to terminate, thus giving time complexity $\mathcal{O}(n \log n)$.

We need to prove this algorithm finds the optimal allocation hence solving the optimisation problem of equation (3). The proof is by induction on the number m of items. The case m=1 is trivial. Assume the algorithm is correct for m items, we shall prove it remains so for m+1 items. Our assumption means we have found an allocation $a_1, a_2, \ldots, a_t$ such that $s = \sum_{i=1,t} a_i b_i$ is maximum with $b_1 \geq b_2 \geq \ldots \geq b_n$. Let us consider the case $t < n$. We have $a_i = m_i$ for $i = 1, \ldots, t{-}1$. If $a_t < m_t$ then our algorithm will find the value of a_t as being $a_t + 1$ and the sum s becomes $s + b_t$ with b_t the maximum value of the remaining bids $b_t, \ldots, b_n$. Therefore, $s{+}b_t$ is the maximum possible we can get w.r.t. the constraints of the problem (any seat allocated to a bidder beyond his required number of seats is free for that bidder). If $a_t = m_t$ then, the m+1th seat will be allocated to the next highest bidder t+1 and the sum becomes $s{+}b_{t+1}$, which is the maximum possible. Similarly, we can show that the case $t = n$ leads to the optimal allocation with the sum $s{+}b_n$ if $a_n < m_n$ or simply the sum s otherwise. □

5.1 The VCG Mechanism Applied to QRMUA

Assuming each bidder has a private valuation v_i and bids b_i for a single seat, we can use Algorithm 1 to find the n+1 optima x^* and $x^*_{-i}, i = 1, \ldots, n$ in the VCG payments of equation (1). To illustrate, let us consider the case where $n = 2, m = 8, m_1 = 5, m_2 = 4$. Assume bidders bid their true valuations b_1=v_1=4 and b_2=v_2=3. We have the following payments:

$$\begin{array}{llll} p_1 = & b_2(x^*_{-1}) - b_2(x^*) = & 4*3 - 3*3 = & 3 \\ p_2 = & b_1(x^*_{-2}) - b_1(x^*) = & 5*4 - 5*4 = & 0 \end{array}$$

Bidder 2 gets its bundle for free and the auctioneer makes the modest revenue of 3.

Lemma 2 *By applying the VCG mechanism to QRMUA, we have the following:*

1. *If $m_1 + m_2 + \ldots + m_n \leq m$ then the VCG payments are zero. The bidders get the bundles for free.*

2. *If $m_1 + m_2 + \ldots + m_n > m$ then there exists a VCG payment that is greater than zero.*

Proof: For the first part of the lemma, the total sum from the optimal allocation in the presence of all bidders is $\sum_{j=1}^{n} m_j b_j$. The total sum from the optimal allocation if i is not bidding is $\sum_{j \neq i}^{n} m_j b_j$ and the reported bid for i is $m_i b_i$. Consequently the VCG payments $p_i = 0$. For the second part of the lemma, let us assume without loss of generality $b_1 \geq b_2 \geq \ldots \geq b_n$, we shall prove that $p_1 > 0$. Since $m_1{+}m_2{+}\ldots{+}m_n > m$, Algorithm 1 will find an optimal allocation $x^* = a_1, \ldots, a_t$ with $t \leq n$ such that $\sum_{i=1}^{t} a_i = m$ and an optimal allocation $x^*_{-1} = a'_2, \ldots, a'_{t'}$ with $t \leq t' \leq n$ such that $\sum_{i=2}^{t'} a'_i = m$. We then have the payment $p_1 = \sum_{j=2}^{t'} a'_j b_j - \sum_{j=2}^{t} a_j b_j$. Let us consider the case $t' = t$. It follows $a_j = a'_j = m_j$ for $j = 2 \ldots t{-}1$ and $a'_t = a_1 + a_t$. Consequently $a'_t > a_t$ hence $p_1 = (a'_t - a_t) b_t > 0$. If $t' > t$, then we have $a_j = a'_j = m_j$ for $j = 2 \ldots t$. Consequently $p_1 = \sum_{j=t+1}^{t'} a'_j b_j > 0$. □

This low or even zero revenue for the auctioneer is one of the shortcomings of the VCG mechanism, see [1] for more VCG criticisms. Ideally, an auctioneer aims to get maximum revenue. Nonetheless, this VCG mechanism with the Clark tax has the attractive property of having a weak budget balance, which is vital in an auction setting. For an optimal allocation $a_1, a_2, \ldots, a_t$ with $t \leq n$, the winning bidders get the utilities $u_i = a_i v_i - p_i$. The losing bidders have zero utility. Having set up a tractable CAP, typically the QRMUA with an allocation and pricing rules, we wish to prove certain properties related to the underlying mechanism by using the `Alloy` modelling language.

5.2 Analysis of the Resulting Auction

As seen for the Vickrey auction, we model the basic entities, their relationships and the dynamic behaviour (the winner determination algorithm) of the resulting QRMUA within `Alloy` as follows. For clarity, we omit the details of the `Alloy` coding but provide the key stages of the modelling. The items are identical and are represented by an `Alloy` signature `Item`. An `Alloy` model of the bidder contains a field `bid`, which is a set of pairs of bundle and price, along with a private value `val` for each individual item, a number of allocated items `alloc` and a utility `u`, which is an integer. Note that, for this particular auction, the bids are additive since the price of a bundle is simply the sum of the prices of its individual items. Furthermore, the complexity of expressing additive bids in the OR bidding language is linear, see for example [20]. Therefore, we can use the OR language to express each agent's preferences.

Figure 2: An `Alloy` metamodel for the QRMUA

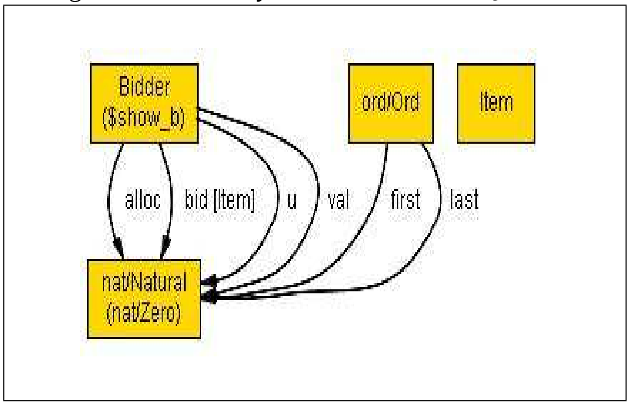


Figure 2 shows a simulation of our QRMUA model for one item and one bidder. It shows for instance the bid `bid[Item]` as a mapping from items to `Natural`, which is the set of non negative integers as provided by the `Alloy` module `natural`. To encode the winner determination using our Algorithm 1, we require the capability of multiplying two integers. This is not provided by `Alloy` but there is an `Alloy` utility module called `natural` allowing to multiply two non negative integers. It views a non negative integer as a set of its predecessors and then multiplying two non negative integers coincide with the cardinality of the Cartesian product of the two sets representing the two numbers. We therefore restrict our checking to the set of non negative integers and for two bidders. The winner determination algorithm is implemented by a predicate `pred wda (p1,p2,p1',p2':Bidder, it:set Item)` that determines the allocation for each bidder, its VCG payment and its associated utility. We then wish to verify the implemented auction mechanism is strategyproof or has a dominant strategy equilibrium.

5.3 Verifying Dominant Strategy Equilibrium

As shown for the Vickrey auction, the dominant strategy equilibrium property for QR-MUA is expressed by a first order logic formula. Unlike the Vickrey auction, here we add the conditionals `p1.bid+p2.bid=it` and `disj[p1.bid,p2.bid]` expressing the fact that the two sets of items for the two bidders form a partition of the set `it` of all items.

The strategyproofness property was verified for different scopes. To give an idea of the runtime, its verification for an auction composed of 20 items took 7 min and 10 s of CPU time to complete. Though, this is not a complete proof of correctness, this suggests that the property is likely to hold for an unbounded model. More importantly, whenever we introduced flaws in the auction mechanism, `Alloy` found a counter-example to the strategyproofness property.

It is worth noting that by expressing the auction mechanism and its game-theoretic properties using first order logic, the checking remains undecidable in general. An alternative would be to restrict to a subset of first order logic that is decidable, e.g., the Presburger formulae [26]. However, even for normal form games, checking a dominant strategy equilibrium requires a time complexity that is exponential in the number of players. The `Alloy` approach combines model checking that is tractable with first order logic formula enabling us to find counter-example for wrong formulae and prove likely correct properties for some finite scope. However, for normal form games, `Alloy` will be able to provide a definite proof of a property by scanning through all the possible actions of each player for a fixed number of players.

6 Conclusions

In this paper we have explained how an agent in an eCommerce scenario could automatically check some interesting properties of a published auction specification. The main property of interest is strategyproofness (i.e. the protocol is immune to counter-speculation and strategic bidding). Our approach to verification uses the Alloy model checker. In particular we have shown, by example, that there are certain restricted cases of combinatorial auctions where it is feasible for agents to check for strategyproofness. The main lesson learnt is that while verifying this property is feasible, it is quite computationally difficult even for relatively simple auction types. In the future we intend to undertake a theoretical investigation of the limits of this type of automated property checking. We expect that some classes of auctions will prove difficult to verify automatically, and that some auctions will only be verifiable if their complexity is limited (e.g. limited number of items, bidders and bundle types). When these limits are determined, they can be used to inform the design of mechanisms for agent scenarios where semantic interoperation is desired.

Acknowledgement

We thank the UK EPSRC for funding this project under grant EP/D02949X/1.

References

[1] L. M. Ausbel and P. Milgrom. The lovely but lonely vickrey auction. In P. Cramton et al., editor, *Combinatorial Auctions*. MIT Press, 2006.

[2] R. Cavallo. Optimal decision-making with minimal waste: strategyproof redistribution of VCG payments. In *AAMAS*, pages 882–889, 2006.

[3] J. Cerquides, U. Endriss, A. Giovannucci, and J. A. Rodríguez-Aguilar. Bidding languages and winner determination for mixed multi-unit combinatorial auctions. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-2007)*, pages 1221–1226, January 2007.

[4] E. H. Clarke. Multipart pricing of public goods. *Public Choice*, 11:17–33, 1971.

[5] V. Conitzer and T. Sandholm. Incremental mechanism design. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-2007)*, pages 1251–1256, January 2007.

[6] W. J. Cook, W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver. *Combinatorial optimization*. Wiley, 605 Third Avenue, New York, US, 1998.

[7] R. K. Dash, N. R. Jennings, and D. C. Parkes. Computational-mechanism design: A call to arms. *IEEE Intelligent Systems*, pages 40–47, November 2003. Special Issue on Agents and Markets.

[8] G. Dennis, R. Seater, D. Rayside, and D. Jackson. Automating commutativity analysis at the design level. In *ISSTA*, pages 165–174, 2004.

[9] M. Genesereth and N. Love. General game playing: Game description language specification, 2005.

[10] R. Gibbons. *A Primer in Game Theory*. Prentice Hall, 1992.

[11] F. Guerin. An algorithmic approach to specifying and verifying subgame perfect equilibria. In *Proceedings of the Eighth Workshop on Game Theoretic and Decision Theoretic Agents (GTDT-2006), Hakodate, Japan*. AAMAS 2006, 2006.

[12] F. Guerin. Applying game theory mechanisms in open agent systems with complete information. *Journal of Autonomous Agents and Multi-Agent Systems*, page publised online, November 2006.

[13] F. Guerin and E. M. Tadjouddine. Realising common knowledge assumptions in agent auctions. In *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pages 579–586, Hong Kong, China, December 2006.

[14] M. R. A. Huth and M. D. Ryan. *Logic in Computer Science: Modelling and Reasoning about Systems*. Cambridge University Press, Cambridge, England, 2000.

[15] D. Jackson. Automating first-order relational logic. In *SIGSOFT FSE*, pages 130–139, 2000.

[16] D. Jackson. Alloy: a lightweight object modelling notation. *ACM Trans. Softw. Eng. Methodol.*, 11(2):256–290, 2002. See also http://alloy.mit.edu.

[17] D. Koller and A. Pfeffer. Representations and solutions for game-theoretic problems. *Artificial Intelligence*, 94(1):167–215, 1997.

[18] Z. Manna and A. Pnueli. *Temporal Verification of Reactive Systems (Safety), vol. 2*. Springer-Verlag, New York, Inc., 1995.

[19] P. Milgrom. *Putting Auction Theory to Work*. Cambridge University Press, 2004.

[20] N. Nisan. Bidding and allocation in combinatorial auctions. In *EC '00: Proceedings of the 2nd ACM conference on Electronic commerce*, pages 1–12, New York, NY, USA, 2000. ACM Press.

[21] D. C. Parkes. Auction design with costly preference elicitation. *Annals of Mathematics and AI*, 44:269–302, 2004.

[22] M. Pauly. Programming and verifying subgame perfect mechanisms. *Journal of Logic and Computation*, 15(3):295–316, 2005.

[23] M. H. Rothkopf, A. Pekec, and R. M. Harstad. Computationally manageable combinatorial auctions. *Management Science*, 44(8):1131–1147, 1998.

[24] T. Sandholm. An algorithm for optimal winner determination in combinatorial auctions. In *IJCAI*, pages 542–547, 1999.

[25] T. Sandholm, S. Suri, A. Gilpin, and D. Levine. Winner determination in combinatorial auction generalizations. In *AAMAS*, pages 69–76, 2002.

[26] E. M. Tadjouddine and F. Guerin. Verifying equilibria in games of complete information represented by presburger formulas. In *Proceedings of the LOFT'06, July 13-15, Liverpool, UK*. LOFT 2006, 2006.

[27] M. Tennenholtz. Some tractable combinatorial auctions. In *AAAI/IAAI*, pages 98–103, 2000.

[28] W. Van der Hoek. Knowledge, rationality and action. In *Proceedings of the AAMAS'04, New York, USA*. AAMAS 2004, 2004.

[29] W. Vickrey. Counter-speculation, auctions, and competitive sealed tenders. *Journal of Finance*, 41(33):8–37, 1961.