

# Formalizing double sided auctions in Coq

Suneel Sarswat

Tata Institute of Fundamental Research, India  
suneel.sarswat@gmail.com

Abhishek Kr Singh

Tata Institute of Fundamental Research, India  
abhishek.uor@gmail.com

---

## Abstract

In this paper we introduce a formal framework for analysing double sided auction mechanisms in a theorem prover. In financial markets double sided auctions are used for price discovery. In a double sided auction multiple buyers and sellers participate for trade. Any mechanism for double sided auctions to match buyers and sellers should satisfy certain properties. For example, fairness, perceived-fairness, and individual rationality are some of the important properties and to verify them we need a formal setting. We formally define all these notions in a theorem prover. This provides us a formal setting in which we prove some useful results on matchings in double sided auctions. Finally, we use this framework to verify properties of two important classes of double sided auction mechanisms. All the properties we discuss in this paper are completely formalized in the Coq proof assistant without adding any axiom to it. This is the first work formalizing matching algorithms using double sided auctions for markets in a theorem prover.

**2012 ACM Subject Classification** Information systems → Online auctions; Software and its engineering → Formal software verification; Theory of computation → Algorithmic mechanism design; Theory of computation → Computational pricing and auctions; Theory of computation → Program verification; Theory of computation → Automated reasoning

**Keywords and phrases** Coq, formalization, auction, matching, financial markets

**Digital Object Identifier** 10.4230/LIPIcs..2019.

## 1 Introduction

Trading is a principal component of all modern economy. Over the century more and more complex instruments (for example, index, future, options etc.) are being introduced to trade in the financial markets. With the arrival of computer assisted trading, the volume and liquidity in the markets have improved significantly. Today all big stock exchanges use computer algorithms (matching algorithms) to match buy requests (demands) with sell requests (supplies) of traders. Computer algorithms are also used by many traders to place orders in the markets<sup>1</sup>. As a result of all this the markets has become complex and large. Hence, the analysis of markets is no more feasible without the help of computers.

A potential trader (buyer or seller) places orders in the markets. These orders are matched by the stock exchange to execute trades. Most stock exchanges divide the trading activity into three main sessions known as per-market, continuous market and post market. In the pre-market session an opening price of a product is discovered through a double sided auction. In the continuous market session the incoming buyers and sellers are continuously matched against each other on a priority basis. In the post-market session clearing of the remaining orders is done and a closing price is discovered.

A double sided auction mechanism allows multiple buyers and sellers to trade simultaneously [5]. In double sided auctions, the auctioneer (e.g. a stock exchange) collects buy and

---

<sup>1</sup> This is known as algorithmic trading.



sell requests over a period of time. Each potential trader places the orders with a limit price: below which a seller will not sell and above which a buyer will not buy. The exchange at the end of this time period matches these orders based on their limit prices. This entire process is completed using a double sided auction matching algorithm.

Designing algorithms for double sided auctions is a well studied topic [6, 9, 7, 10]. A major emphasis of many of these algorithms is to maximize the number of matches or maximize the profit of the auctioneer. Note that an increase in the number of matches increases the liquidity in the markets. A matching algorithm can produce a matching with a uniform price or a matching with dynamic prices. An algorithm which clears each matched bid-ask pair at a single price is called a uniform price algorithm. An algorithm which may clear each matched bid-ask pair at different prices is called a dynamic price algorithm. There are other important properties besides the number of matches which are considered while evaluating the effectiveness of a matching algorithm. For example, fairness, uniform pricing, and individual rationality are some of the relevant features used to compare these matching algorithms. However, it is known that no single algorithm can possess all of these properties simultaneously [9, 6].

In this paper, we present a formal framework to analyze double sided auctions using a theorem prover. For this work, we assume that each trader wishes to trade a single unit of product and the product is indivisible. In Section 2 we formally define the theory of double sided auctions in the Coq proof assistant. In Section 3 we define and prove some important properties of matching algorithms in double sided auctions. In particular we present a dynamic price matching algorithm which produces a maximum as well as a fair matching. In Section 4 we describe a uniform price matching algorithm used for price discovery in financial markets. Moreover, we prove that it produces a matching which is maximum among all possible uniform matchings. We summarize the work in Section 5 with an overview of possible future works. The Coq formalization for this paper is available at [1].

## 2 Modeling double sided auctions

To formalize the notion of matching in a double sided auction we use list data structure. List is also used to define various processes that operate on a matching. However, to express the properties of these processes we need some relations on lists which are analogous to relations on multisets. In this section we formally define these relations which are further used for stating some important results on matching in a double sided auction.

### 2.1 Bid, Ask and limit price

An auction is a competitive event, where goods and services are sold to the highest bidders. In any double sided auction multiple buyers and sellers place their orders to buy or sell a unit of underlying product. The auctioneer then matches these buy-sell requests based on their *limit prices*. While the limit price for a buy order (i.e. *bid*), is the price above which the buyer doesn't want to buy one quantity of the item. The limit price of a sell order (i.e. *ask*), is the price below which the seller doesn't want to sell one quantity of the item. In this work we assume that each bid is a buy request for one unit of item. Similarly each ask is a sell request for one unit of item. If a trader wishes to buy or sell multiple units, he can create multiple bids or asks with different *ids*.

We can express bid as well ask using records containing two fields.

```
Record Bid: Type := Mk_bid { bp:> nat;    idb: nat }.
```

88     `Record Ask: Type := Mk_ask { sp:> nat;     ida: nat }.`

89     For a bid  $b$ ,  $(bp\ b)$  is the limit price and  $(idb\ b)$  is its unique identity. Similarly for an ask  
 90      $a$ ,  $(sp\ a)$  is the limit price and  $(ida\ a)$  is the unique identity of  $a$ . Note the use of coercion  
 91     symbol `:>` in the first field of *Bid*. It declares  $bp$  as an implicit function which is applied to  
 92     any term of type *Bid* appearing in a context where a natural number is expected. Hence,  
 93     now onward we can simply use  $b$  instead of  $(bp\ b)$  to express the limit price of  $b$ . Similarly  
 94     we can use  $a$  for the limit price of an ask  $a$ .

95     Since equality for both the fields of *Bid* as well as *Ask* is decidable (i.e. `nat: eqType`),  
 96     the equality on *Bid* as well as *Ask* can also be proved decidable. This is achieved by declaring  
 97     two canonical instances `bid_eqType` and `ask_eqType` which connect *Bid* and *Ask* to the  
 98     `eqType`.

## 99     2.2     Matching in Double Sided Auctions

100     In a double sided auction (DSA), the auctioneer collects all the buy and sell requests for  
 101     a fixed duration. All the buy requests can be assumed to be present in a list  $B$ . Similarly,  
 102     all the sell requests can be assumed to be present in a list  $A$ . At the time of auction, the  
 103     auctioneer matches bids in  $B$  against asks in  $A$ . We say a bid-ask pair  $(b, a)$  is *matchable* if  
 104      $b \geq a$  (i.e.  $bp\ b \geq sp\ a$ ). Furthermore, the auctioneer assigns a trade price to each matched  
 105     bid-ask pair. This process results in a matching  $M$ , which consists of all the matched bid-ask  
 106     pairs together with their trade prices. We define matching as a list whose entries are of type  
 107     `fill_type`.

108     `Record fill_type: Type:=     Mk_fill {bid_of: Bid;     ask_of: Ask;     tp: nat}`

109     In a matching  $M$ , a bid or an ask appears at most once. Note that there might be some  
 110     bids in  $B$  which are not matched to any ask in  $M$ . Similarly there might be some asks in  $A$   
 111     which are not matched to any bid in  $M$ . The list of bids present in  $M$  is denoted as  $B_M$   
 112     and the list of asks present in  $M$  is denoted as  $A_M$ . For example, Fig. 1 shows a matching  
 113      $M$  between list of bids  $B$  and list of asks  $A$ . While the asks present in  $A$  is shown using  
 114     left brackets and corresponding limit prices. The bids present in  $B$  is represented using  
 115     right brackets and corresponding limit prices. All the matched bid-ask pair in  $M$  is then  
 116     represented using brackets of same colors. Note that the bid with limit price 37 is not present  
 117     in  $B_M$  since it is not matched to any ask in  $M$ .

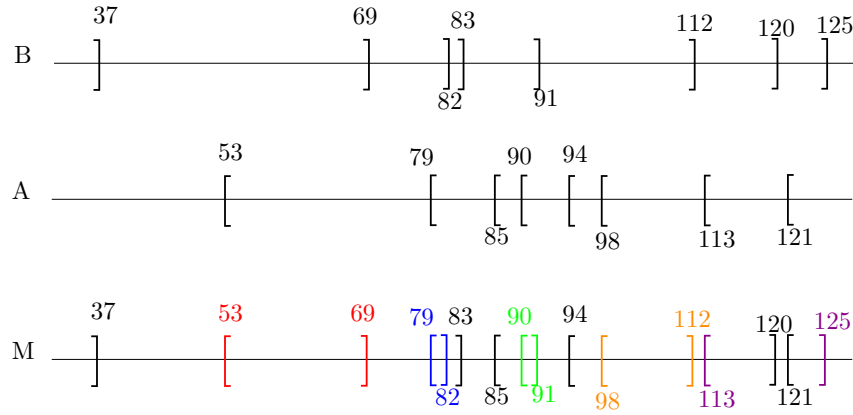
118     More precisely, for a given list of bids  $B$  and list of asks  $A$ ,  $M$  is a matching iff, (1) All  
 119     the bid-ask pairs in  $M$  are matchable, (2)  $B_M$  is duplicate-free, (3)  $A_M$  is duplicate-free, (4)  
 120      $B_M \subseteq B$ , and (5)  $A_M \subseteq A$ .

121     ► **Definition 1.**  $matching\_in\ B\ A\ M := All\_matchable\ M \wedge NoDup\ B_M \wedge NoDup\ A_M$   
 122      $\wedge B_M \subseteq B \wedge A_M \subseteq A$ .

123     While the term  $NoDup\ B_M$  in above definition indicates that each bid is a request to trade  
 124     one unit of item and the items are indivisible. We use the expression  $B_M \subseteq B$  to denote  
 125     the term  $(Subset\ B_M\ B)$ . It expresses the fact that each element in the list  $B_M$  is also  
 126     present in the list  $B$ .

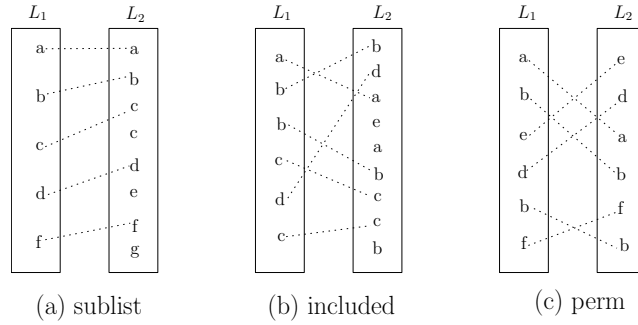
## 127     2.3     Lists, sublist and permutation

128     While the predicates `NoDup` and `Subset` are sufficient to express the notion of a matching. We  
 129     need more definitions to describe the properties of matching in double sided auctions. In the



■ **Figure 1** Bids in  $B$  and asks in  $A$  are represented using right and left brackets respectively. Every matched bid-ask pair in  $M$  is shown using brackets of same colors. Bids with limit prices 37, 83 and 120 are not matched to any ask in the matching  $M$ .

130 following paragraphs we describe three binary relations on lists namely **sublist**, **included**  
 131 and **perm** which are then used for stating important results on matching in a double sided  
 132 auction.



■ **Figure 2** The dotted lines between the entries of lists confirm the presence of these entries in both the lists. (a) If  $L_1$  is **sublist** of  $L_2$  then no two dotted lines can intersect. (b) A list  $L_1$  is **included** in  $L_2$  if every entry in  $L_1$  is also present in  $L_2$ . (c) Two lists  $L_1$  and  $L_2$  are permutation of each other if each entry has same number of occurrences in both  $L_1$  and  $L_2$ .

133 **sublist**  $L_1 L_2$  : The notion of **sublist** is analogous to the subsequence relation on sequences.  
 134 For the given lists  $L_1$  and  $L_2$  the term (**sublist**  $L_1 L_2$ ) evaluates to **true** if every entry of  
 135  $L_1$  is also present in  $L_2$  and they appear in the same succession. For example, in Fig. 2(a)  
 136 the list  $L_1$  is a **sublist** of  $L_2$  since there is a line incident on each entry of  $L_1$  and no two  
 137 lines intersect each other.

138 More precisely, for any two lists  $l$  and  $s$  whose elements are of type  $T$  we have following  
 139 lemmas specifying the **sublist** relation.

140 ► **Lemma 2.**  $\text{sublist\_intro1 } (a:T): \text{sublist } l \ s \rightarrow \text{sublist } l \ (a::s).$

141 ► **Lemma 3.**  $\text{sublist\_elim3a } (a \ e:T): \text{sublist } (a::l)(e::s) \rightarrow \text{sublist } l \ s.$

142 ► **Lemma 4.**  $\text{sublist\_elim4}: \text{sublist } l \ s \rightarrow (\forall a, \text{count } a \ l \leq \text{count } a \ s).$

143 The term (*count a l*) in Lemma 4 represents the number of occurrences of element *a* in  
 144 the list *l*. Note the recursive nature of **sublist** as evident in Lemma 3. It makes inductive  
 145 reasoning easier for the statements which contain **sublist** in the antecedent. However, this  
 146 is not true for the other relations (i.e. **included** and **perm**).

147 **included**  $L_1 L_2$  : A list  $L_1$  is **included** in list  $L_2$  if every entry of  $L_1$  is also present in  $L_2$ .  
 148 The notion of **included** is analogous to the subset relation on multisets. In Fig 2(b) the list  
 149  $L_1$  is **included** in  $L_2$  since there is a line incident on each entry of  $L_1$ . More precisely, we  
 150 have following lemmas specifying the **included** relation.

151 ► **Lemma 5.** *included\_intro*:  $(\forall a, \text{count } a \ l \leq \text{count } a \ s) \rightarrow \text{included } l \ s$ .

152 ► **Lemma 6.** *included\_elim*:  $\text{included } l \ s \rightarrow (\forall a, \text{count } a \ l \leq \text{count } a \ s)$ .

153 ► **Lemma 7.** *included\_intro3*:  $\text{sublist } l \ s \rightarrow \text{included } l \ s$ .

154 Note that if *l* is **sublist** of *s* then *l* is also **included** in *s* but not the vice versa. However,  
 155 if both the lists *l* and *s* are sorted based on some ordering on type T then *l* is **sublist** of *s*  
 156 whenever *l* is **included** in *s*.

157 ► **Lemma 8.** *sorted\_included\_sublist*:  $\text{Sorted } l \rightarrow \text{Sorted } s \rightarrow \text{included } l \ s \rightarrow$   
 158  $\text{sublist } l \ s$ .

159 **perm**  $L_1 L_2$  : A list  $L_1$  is permutation of list  $L_2$  iff  $L_1$  is included in  $L_2$  and  $L_2$  is included  
 160 in  $L_1$ . The notion of permutation for lists is similar to the equality in multisets. In Fig 2(c)  
 161 the list  $L_1$  is perm of list  $L_2$ . We have following lemmas specifying the essential properties  
 162 of the **perm** relation.

163 ► **Lemma 9.** *perm\_intro*:  $(\forall a, \text{count } a \ l = \text{count } a \ s) \rightarrow \text{perm } l \ s$ .

164 ► **Lemma 10.** *perm\_elim*:  $\text{perm } l \ s \rightarrow (\forall a, \text{count } a \ l = \text{count } a \ s)$ .

165 ► **Lemma 11.** *perm\_sort*(*e*:  $T \rightarrow T \rightarrow \text{bool}$ ):  $\text{perm } l \ s \rightarrow \text{perm } l \ (\text{sort } e \ s)$ .

166 The term (*sort s*) in Lemma 11 represents the list *s* sorted using an ordering relation  
 167 *e*. The definition of matching as a list is necessary for describing processes that operate on  
 168 it. However, while describing various properties of a matching we can always consider it as  
 169 a collection. For example, consider the following lemma which states that the property of  
 170 being a matching is invariant over permutation.

171 ► **Lemma 12.** *match\_inv*:  $\text{perm } M \ M' \rightarrow \text{perm } B \ B' \rightarrow \text{perm } A \ A' \rightarrow \text{matching\_in}$   
 172  $B \ A \ M \rightarrow \text{matching\_in } B' \ A' \ M'$ .

173 We use  $B_M$  to represent the list of bids from *B* that are matched in *M*. Similarly we use  
 174 notation  $P_M$  to represent a list containing trade prices of matched bid-ask pair in *M*. While  
 175 proving various properties of a matching *M* we very often base our arguments solely on the  
 176 information present in  $B_M$ ,  $A_M$  and  $P_M$ . Therefore it is useful to have lemmas establishing  
 177 the interaction of  $B_M$ ,  $A_M$  and  $P_M$  with above mentioned relations on lists.

178 ► **Lemma 13.** *included\_M\_imp\_included\_bids*:  $\text{included } M \ M' \rightarrow \text{included } B_M \ B_{M'}$   
 179  $.$

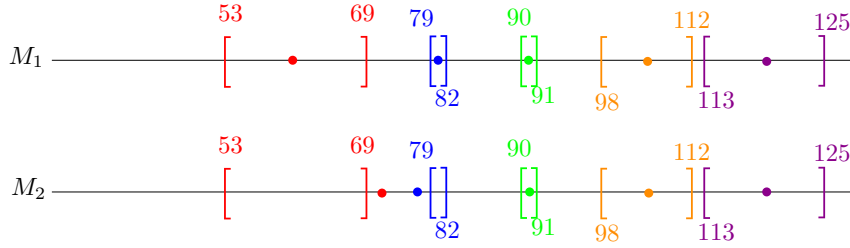
180 ► **Lemma 14.** *included\_M\_imp\_included\_asks*:  $\text{included } M \ M' \rightarrow \text{included } A_M \ A_{M'}$

181 ► **Lemma 15.** *included\_M\_imp\_included\_tps*:  $\text{included } M \ M' \rightarrow \text{included } P_M \ P_{M'}$

182 ► **Lemma 16.** *sorted\_nodup\_is\_sublistB*:  $\text{Sorted } B \rightarrow \text{Sorted } B' \rightarrow \text{NoDup } B \rightarrow$   
 183  $\text{NoDup } B' \rightarrow B \subseteq B' \rightarrow \text{sublist } P_B \ P_{B'}$ .

### 3 Analysis of Double sided auctions

Usually in a double sided auctions mechanism, the profit of an auctioneer is the difference between the limit prices of matched bid-ask pair. In this work we do not consider analysis of profit for the auctioneer. Therefore the buyer of matched bid-ask pair pays the same amount which the seller receives. This price for a matched bid-ask pair is called the trade price for that pair. Since the limit price for a buyer is the price above which she doesn't want to buy, the trade price for this buyer is expected to be below its limit price. Similarly the limit price for a seller is the price below which he doesn't want to sell, hence the trade price for this seller is expected to be below its limit price. Therefore in any matching it is desired that the trade price of a bid-ask pair lies between their limit prices. A matching which has this property is called an *indivial rational (IR)* matching. Note that any matching can be converted to an IR matching without altering it's bid-ask pair (See Fig 3).



**Figure 3** The colored dots represent the trade prices at which the corresponding matched bid-ask pairs are traded. While the matching  $M_2$  is not IR since some dots lie outside the corresponding matched bid ask-pair. The matching  $M_1$  is IR because trade prices for every matched bid-ask pair lie inside the interval. Note that the matching  $M_1$  and  $M_2$  contains exactly same bid-ask pairs.

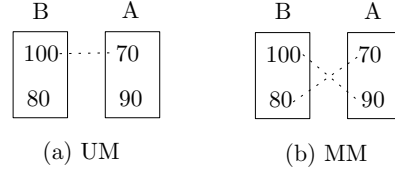
The number of matched bid-ask pairs produced by a matching algorithm is crucial in the design of a double sided auction mechanism. Increasing the number of matched bid-ask pairs increases the liquidity in market. Therefore, producing a maximum matching is an important aspect of double sided auction mechanism design. For a given list of bids  $B$  and list of asks  $A$  we say a matching  $M$  is maximum if no other matching  $M'$  on the same  $B$  and  $A$  contains more number of matched bid-ask pairs than  $M$ .

**Definition 17.**  $Is\_MM\ M\ B\ A := (matching\_in\ B\ A\ M) \wedge (\forall\ M',\ matching\_in\ B\ A\ M' \rightarrow |M'| \leq |M|).$

In certain situations, to produce a maximum matching, different bid-ask pair must be assigned different trade prices. For example see Fig 4. To get a maximum matching of size two it is necessary to trade both the matched bid-ask pair at different prices. However, different prices for the same product in the same market simultaneously leads to dissatisfaction amongst some of the traders. A mechanism which clears all the matched bid-ask pairs at same trade price is called a *uniform matching*. It is also known as *perceived-fairness*.

#### 3.1 Fairness

A bid with higher limit price is more competitive compared to bids with lower limit prices. Similarly an ask with lower limit price is more competitive compared to asks with higher limit prices. In a competitive market, like double sided auction, it is necessary to prioritise more competitive traders for matching. A matching which prioritise more competitive traders is



■ **Figure 4** In this figure two bids with limit prices 100 and 80 respectively are matched against two asks of limit price 70 and 90. There is only one matching  $M_2$  of size two possible and it is not uniform.

215 called a *fair* matching. Consider the following predicates `fair_on_bids` and `fair_on_asks`  
 216 which are used to describe a fair matching.

217 ► **Definition 18.**  $\text{fair\_on\_bids } M B := \forall b b', \text{In } b B \wedge \text{In } b' B \rightarrow b > b' \rightarrow \text{In } b' B_M \rightarrow \text{In } b B_M.$   
 218

219 ► **Definition 19.**  $\text{fair\_on\_asks } M A := \forall s s', \text{In } s A \wedge \text{In } s' A \rightarrow s < s' \rightarrow \text{In } s' A_M \rightarrow \text{In } s A_M.$   
 220

221 ► **Definition 20.**  $\text{Is\_fair } M B A := \text{fair\_on\_asks } M A \wedge \text{fair\_on\_bids } M B.$

222 Here, the predicate `fair_on_bids`  $M B$  states that the matching  $M$  is fair for the list of  
 223 buyers  $B$ . Similarly, the predicate `fair_on_asks`  $M A$  states that the matching  $M$  is fair for  
 224 the list of sellers  $A$ . A matching which is fair on both the traders (i.e.  $B$  and  $A$ ) is expressed  
 225 using the predicate `Is_fair`  $M B A$ .

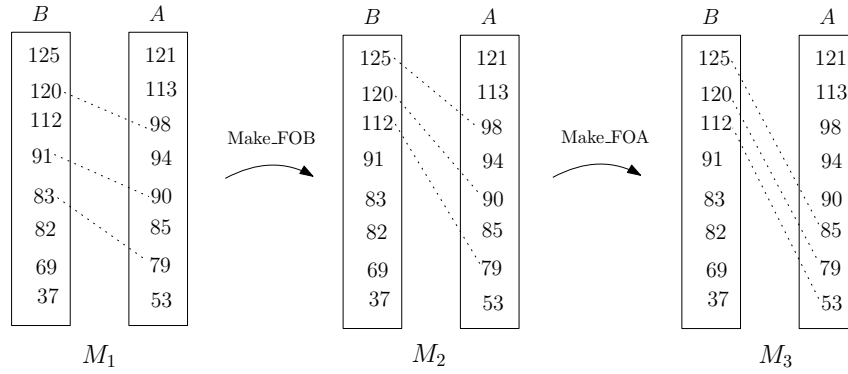
226 Unlike the uniform matching, a fair matching can always be achieved without compro-  
 227 mising the size of matching. We can accomplish this by converting any matching into a fair  
 228 matching without changing its size. For example, consider the following function `make_FOB`.

```
229 Fixpoint Make_FOB (M:list fill_type) (B: list Bid):=
230   match (M,B) with
231   | (nil,_) => nil
232   | (m::M',nil) => nil
233   | (m::M',b::B') => (Mk_fill b (ask_of m) (tp m))::(Make_FOB M' B')
234 end.
```

235 The function `make_FOB` produces a `fair_on_bids` matching from a given matching  $M$   
 236 and a list of bids  $B$ , both sorted in decreasing order of bid prices (See Fig 5). The function  
 237 `make_FOB` is a recursive function and at each step it replaces the largest bid in  $M$  with the  
 238 largest bid available in  $B$ . Since at any moment the largest bid in  $B$  is bigger than the largest  
 239 bid in  $M$ , the new bid-ask pair is still matchable. Note that `make_FOB` doesn't change any of  
 240 the ask in  $M$  and due to the recursive nature of `make_FOB` on  $B$ , a bid is not repeated in the  
 241 process of replacement. This ensure that the new  $B_M$  is duplicate-free. Once a matching is  
 242 modified to a fair matching on bids, we use similar function `make_FOA` on this matching to  
 243 produce a fair on ask matching. Hence the final result is a fair matching.

244 More precisely, for the function `make_FOB` and `make_FOA` we have the following lemmas  
 245 proving it fair on bids and fair on asks respectively.

246 ► **Lemma 21.**  $\text{mfob\_fair\_on\_bid } M B: (\text{Sorted } M) \rightarrow (\text{Sorted } B) \rightarrow \text{sublist } P_{B_M}$   
 247  $P_B \rightarrow \text{fair\_on\_bids } (\text{Make\_FOB } M B) B.$



■ **Figure 5** The dotted lines in this figure represent matched bid-ask pairs in matching  $M_1$ ,  $M_2$  and  $M_3$ . In the first step function `make_FOB` operates on  $M_1$  recursively. At each step it picks the top bid-ask pair, say  $(b, a)$  in  $M_1$  and replaces the bid  $b$  with most competitive bid available in  $B$ . The result of this process is a `fair_on_bids` matching  $M_2$ . In a similar way the function `make_FOA` changes  $M_2$  into a fair on ask matching  $M_3$ .

248 ► **Lemma 22.** `mfob_fair_on_ask M A: (Sorted M) -> (Sorted A) -> sublist PAM`  
 249 `PA -> fair_on_asks (Make_FOA M A) A.`

250 ► **Theorem 23.** `exists_fair_matching (Nb: NoDup B)(Na: NoDup A): matching_in`  
 251 `B A M -> ( ∃ M', matching_in B A M' ∧ Is_fair M' B A ∧ |M| = |M'| ).`

252 Proof of Theorem 23 depends on Lemma 22 and Lemma 21. Furthermore, Lemma 22  
 253 and Lemma 21 can be proved using induction on the size of M.

## 254 3.2 Maximum Matching

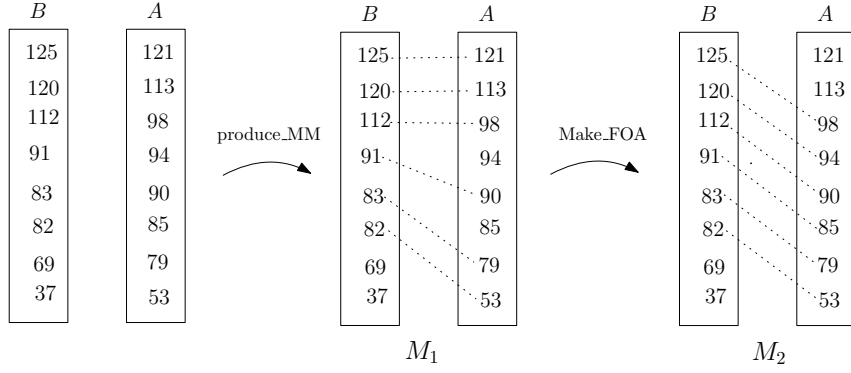
255 The liquidity in any market is a measure of how quickly one can trade in the market without  
 256 much cost. A highly liquid market boosts the investor's confidence in the market. One way  
 257 to increase the liquidity in a double sided auction is to maximize the number of matched  
 258 bid-ask pair. In the previous section we have seen that any matching can be changed to a  
 259 fair matching without altering its size. Therefore, we can have a maximum matching without  
 260 compromising on the fairness of the matching. In this section we describe a matching which  
 261 fair as well as maximum. For a given list of bid  $B$  and list of ask  $A$ , a maximum and fair  
 262 matching can be achieved in two steps. In first we have function `produce_MM` which produces  
 263 a matching which is maximum and fair on bids. In the next step we apply `make_FOA` to this  
 264 maximum matching to produce a fair on ask matching (See Fig 6).

```

265 Fixpoint produce_MM (B:list Bid) (A: list Ask): (list fill_type) :=
266   match (B, A) with
267   | (nil, _) => nil
268   | (b::B', nil) => nil
269   | (b::B', a::A') => match (a <= b) with
270     | true => ({|bid_of:= b; ask_of:= a; tp:=(bp b)|})::(produce_MM B' A')
271     | false => produce_MM B A'
272   end
273 end.
```

274 At each iteration the above function generates a matchable bid-ask pair (See Fig 6). Due  
 275 to the recursive nature of function `produce_MM` on both  $B$  and  $A$ , it never pairs any bid with





■ **Figure 6** In the first step, the function `produce_MM` operates recursively on the list of bids  $B$  and list of asks  $A$ . At each step the function `produce_MM` selects the most competitive available bid and then pairs it with the largest matchable ask. Note that the output of this function is fair on bid since it doesn't leave any bid from top. In the second step, the function `make_FOA` converts the  $M_1$  into fair matching  $M_2$ .

more than two asks. This ensures that the list of bids in matching (i.e.  $B_M$ ) is duplicate-free. Note that the function `produce_MM` tries to match a bid until it finds a matchable ask. The function terminates when either all the bids are matched or it encounters a bid for which no matchable ask is available. Therefore, the function `produce_MM` produces a matching from a given lists of bids  $B$  and a list of asks  $A$ , both sorted in decreasing order by there limit prices.

The following theorem states that when the function `produce_MM` is given a list of bids  $B$  and list of asks  $A$ , both sorted in decreasing order by limit prices, then it produces a maximum matching.

► **Theorem 24.**  $produce\_MM\_is\_MM(Nb:NoDup\ B)(Na:NoDup\ A): Sorted\ B \rightarrow Sorted\ A \rightarrow Is\_MM\ (produce\_MM\ B\ A)\ B\ A.$

**Proof:** We prove this result using induction on the size of list  $A$ .

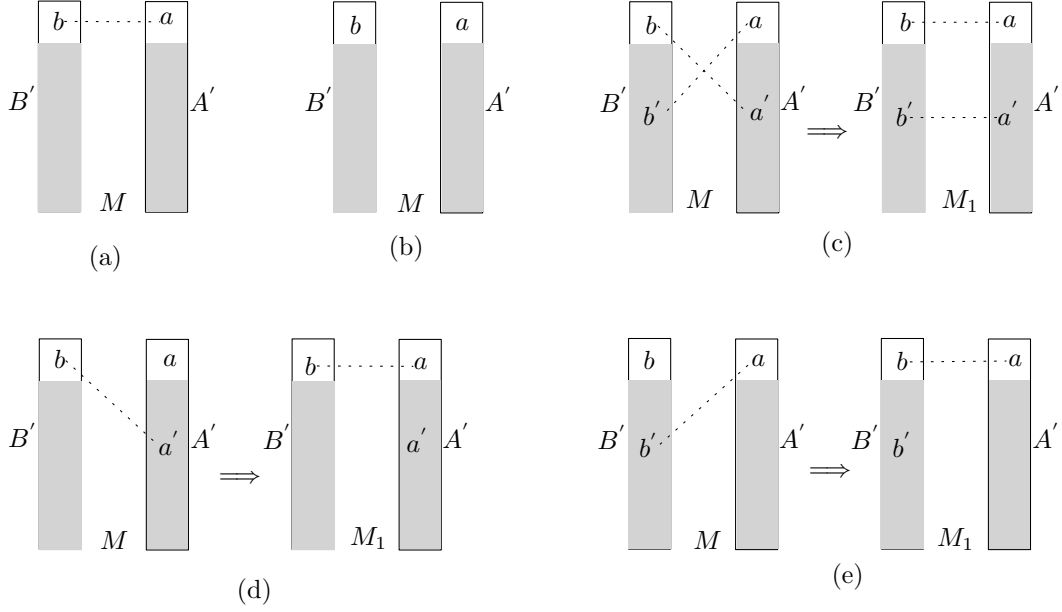
■ Induction hypothesis (IH):  $\forall A', |A'| < |A| \rightarrow \forall B, Sorted\ B \rightarrow Sorted\ A' \rightarrow Is\_MM\ (produce\_MM\ B\ A')\ B\ A'.$

Let  $M$  be an arbitrary matching on the list of bids  $B$  and list of asks  $A$ . Moreover, assume that  $b$  and  $a$  are the topmost bid and ask present in  $B$  and  $A$  respectively (i.e.  $A = (a :: A')$  and  $B = (b :: B')$ ). We need to prove that  $|M| \leq |produce\_MM\ B\ A|$ . We prove this inequality in the following two cases.

■ **Case-1** ( $b < a$ ): In this case the limit price of  $a$  is strictly more than the limit price of  $b$ . In this case the function `produce_MM` computes a matching on  $B$  and  $A'$ . Note that due to the induction hypotheses (i.e. IH) this is a maximum matching for  $B$  and  $A'$ . Since the limit price of ask  $a$  is more than the most competitive bid  $b$  in  $B$  it cannot be present in any matching of  $B$  and  $A$ . Therefore a maximum matching on  $B$  and  $A'$  is also a maximum matching on  $B$  and  $A$ . Hence we have  $|M| \leq |produce\_MM\ B\ A|$ .

■ **Case-2** ( $a \leq b$ ): In this case the function `produce_MM` produces a matching of size  $m + 1$  where  $m$  is the size of matching `produce_MM`  $B'\ A'$ . Hence we need to prove that  $|M| \leq m + 1$ . Note that due to induction hypothesis (i.e. IH) the matching `produce_MM`  $B'\ A'$  is a maximum matching on  $B'$  and  $A'$ . Hence no matching on  $B'$  and  $A'$  can have size bigger than  $m$ . Without loss of generality we can assume that  $M$  is also sorted in decreasing order of bid prices. Now we further split this case into the following five sub

cases (see Fig 7).



■ **Figure 7** This figure shows all the five sub cases of Case-2 (i.e. when  $b \geq a$ ). The dotted line shows presence of the connected pair in matching  $M$ . Both the list of bids  $B$  and list of asks  $A$  are sorted in decreasing order of their limit prices. Moreover, we assume  $B = b :: B'$  and  $A = a :: A'$ .

- 307 ■ **Case-2A** ( $M = (b, a) :: M'$ ) : In this case bid  $b$  is matched to ask  $a$  in the matching  
 308  $M$  (see Fig 7 (a)). Note that  $M'$  is a matching on  $B'$  and  $A'$ . Since  $M' \leq m$  we have  
 309  $|M| = |M'| + 1 \leq m + 1$ .
- 310 ■ **Case-2B** ( $b \notin B_M \wedge a \notin A_M$ ) : In this case neither bid  $b$  nor ask  $a$  is present in  
 311 matching  $M$  (see Fig 7 (b)). Therefore  $M$  is a matching on  $B'$  and  $A'$ . Hence we have  
 312  $|M| \leq m < m + 1$ .
- 313 ■ **Case-2C** ( $(b, a') \in M \wedge (b', a) \in M$ ) : In this case we have  $(b, a') \in M$  and  $(b', a) \in M$   
 314 where  $a' \in A'$  and  $b' \in B'$ . We can obtain another matching  $M_1$  of same size as  $M$   
 315 (see Fig 7 (c)) where  $(b, a) \in M_1$  and  $(b', a') \in M_1$ . Note that all other entries of  $M_1$   
 316 is same as  $M$ . Therefore we have  $M_1 = (b, a) :: M'$  where  $M'$  is a matching on  $B'$  and  
 317  $A'$ . Since  $|M'| \leq m$  we have  $|M| = |M_1| \leq m + 1$ .
- 318 ■ **Case-2D**: ( $(b, a') \in M \wedge a \notin A_M$ ) : In this case we have  $(b, a') \in M$  and  $a \notin A_M$  where  
 319  $a' \in A'$ . We can obtain another matching  $M_1$  of same size as  $M$  (see Fig 7 (d)) where  
 320  $(b, a) \in M_1$ . Therefore we have  $M_1 = (b, a) :: M'$  where  $M'$  is a matching on  $B'$  and  
 321  $A'$ . Since  $|M'| \leq m$  we have  $|M| = |M_1| \leq m + 1$ .
- 322 ■ **Case-2E**: ( $(b', a) \in M \wedge b \notin B_M$ ) : In this case we have  $(b', a) \in M$  and  $b \notin B_M$  where  
 323  $b' \in B'$ . We can obtain another matching  $M_1$  of same size as  $M$  (see Fig 7 (e)) where  
 324  $(b, a) \in M_1$ . Therefore we have  $M_1 = (b, a) :: M'$  where  $M'$  is a matching on  $B'$  and  
 325  $A'$ . Since  $|M'| \leq m$  we have  $|M| = |M_1| \leq m + 1$ .

326 Note that all the cases in the above proof correspond to predicates which can be expressed  
 327 using membership predicate on lists. Since we have decidable equality on the elements of  
 328 list all these predicates are also decidable. Hence, we can do case analysis on them without  
 329 assuming any axiom.  $\square$

Now that we proved maximality property of `produce_MM` one can produce a fair as well as maximum matching by applying the functions `Make_FOA` and `Make_FOB` to the output of `produce_MM`. More precisely, for a given list of bids  $B$  and list of asks  $A$ , we have following result stating that there exists a matching which is both maximum and fair.

► **Theorem 25.**  *$exists\_fair\_maximum (B: list Bid)(A: list Ask): \exists M, (Is\_fair M B A \wedge Is\_MM M B A).$*

## 4 Matching in financial markets

An exchange is an organized financial market. There are various types of exchanges for example stock exchange, commodity exchange, foreign exchange etc. The job of an exchange is to facilitate trading between buyers and sellers for the products which are registered in the exchange. Many exchanges operate during a fix duration in the day. Some exchanges divide the trading activities into multiple sessions for various reasons. Most stock exchanges hold trading into two main session; pre-market (or auction session) and continuous market (or regular trading session). During the pre-market session an exchange collects all the bids and asks for fix duration and then apply double sided auction mechanism to these orders. At the end of the pre-market session an opening price for the product is discovered. During the regular sessions a bid (ask) is matched against the existing asks (bids) immediately. If the bid (ask) is not matchable it is placed in a priority queue based on its limit price.

The pre-market session reduces uncertainty and volatility for the regular sessions of trading. To avoid failure on behalf of the traders the exchange must match all the bids and asks within their limit prices. So any matching produced during this session must be individual rational. One of the most important aspects of the pre-market session is to discover the price of underlying product based on the total demand and supply. Furthermore, the exchange must be fair towards all the traders. So the matching produced during this session should be a fair matching. This means discovering a unique price at which maximum number of bid-ask pairs can be traded. This price is also known as equilibrium price.

Most exchanges matches the bids and asks during the pre-market session at equilibrium price. We describe an algorithm which produces an equilibrium price. The algorithm `UM` produces a individual rational matching which is fair and maximum among all uniform matchings.

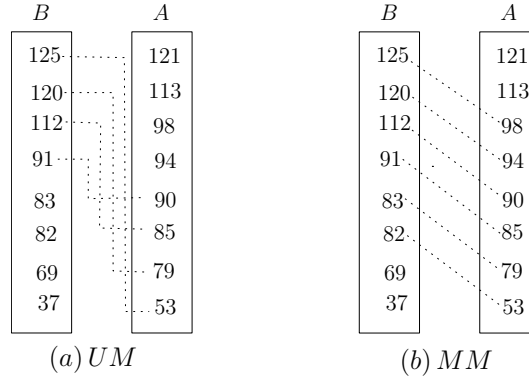
```

360 Fixpoint produce_UM (B:list Bid) (A:list Ask) :=
361   match (B,A) with
362   |(nil, _) => nil
363   |(_,nil)=> nil
364   |(b::B',a::A') => match (a <= b) with
365   |false =>nil
366   |true  => ({|bid_of:= b; ask_of:= a; tp:=(bp b)|}):produce_UM B' A'
367   end
368 end.
369 Definition uniform_price B A := bp (bid_of (last (UM_matching B A))).
370 Definition UM B A:= replace_column (UM_matching B A) (uniform_price B A).
```

The function `produce_UM` produces bid-ask pairs, `uniform_price` discover the uniform price and finally `UM` produces a uniform matching. The function `produce_UM` is a recursive function which matches the largest available bid in  $B$  with the smallest available ask in  $A$  at each iteration (See Fig 8). The function `produce_UM` terminates at a step when the most

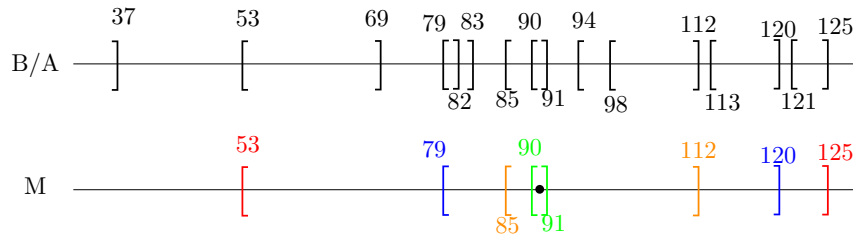
375 competitive bid available in  $B$  is not matchable with any available ask in  $A$ . The following  
 376 theorem states that when the function `produce_UM` is given a list of bids  $B$  and list of asks  
 377  $A$ , where  $B$  is sorted in decreasing order by limit prices and  $A$  is sorted in increasing order  
 378 by limit prices, it produces a maximum matching among all uniform matchings.

379 ► **Theorem 26.** *UM\_is\_maximal\_Uniform* ( $B$ : list Bid) ( $A$ : list Ask): Sorted  $B \rightarrow$   
 380 Sorted  $A \rightarrow \forall M$ : list fill\_type, Is\_uniform  $M \rightarrow |M| \leq |(\text{UM } B \ A)|$



■ **Figure 8** (a) The dotted lines indicate all the bid-ask pair produced by function `produce_UM`. In each iteration function `produce_UM` matches the largest available bid in  $B$  with the smallest available ask in  $A$ . (b) The dotted lines here indicate a maximum matching for the list of bids  $B$  and list of asks  $A$ . Note that in this case the matching produced by `UM` is not a maximum matching.

381 **Proof:** Let  $M$  be any arbitrary IR and uniform matching on the list of bids  $B$  and list  
 382 of asks  $A$  where each matched bid-ask pair is traded at price  $t$ . We need to prove that  $m \leq$   
 383  $|(\text{UM } B \ A)|$  where  $m$  is the number of matched bid-ask pairs in the matching  $M$ . Observe  
 384 that in any individual rational and uniform matching the number of bids above trade price  
 385 is same as the number of asks below the trade price (See Fig 9). Therefore, there are at least  
 386  $m$  bids above  $t$  and  $m$  asks below  $t$  in  $B$  and  $A$  respectively.



■ **Figure 9** Trade price  $p$  for the matching  $M$  is shown using a dot that lies between the ask with limit price 90 and bid with limit price 91. Note that since  $M$  is individual rational the number of matched asks below the trade price  $p$  is same as number of matched bids above the trade price  $p$ .

387 Since at each step the function `produce_UM` pairs the largest bid available in  $B$  with the  
 388 smallest ask available in  $A$  it must produce at least  $m$  bid-ask pairs. Hence for the list of  
 389 bids  $B$  and list of asks  $A$  the function `UM` produces a uniform matching which of size at least  
 390  $m$ . □

## 5 Conclusion

There are several algorithms known for double sided auctions [4]. For many applications of double sided auctions, fairness, individual rationality and uniformity are some of the essential properties required. Theorem provers can be a useful tool in the analysis of these properties [3, 2]. Formalizing an algorithm for double sided auctions in a theorem prover increases its reliability.

Previously there have been two works that formalizes some of the concepts in auctions and an attempt to formalize financial markets [8] in a theorem prover [?, ?]. Prior to our work, to the best of our knowledge, there has been no work formalizing matchings in markets. This is the first work formalizing matching algorithms using double sided auctions for markets in a theorem prover.

We formally define various notions of double sided auctions in the Coq Proof Assistant. Moreover, we prove some general results on matchings in a double sided auction. We use lists to define the notion of a matching. To express the properties of various processes operating on a matching we need some relations on lists. We define these relations and develop a library of facts on these relations which is further used to prove important results on matchings in a double sided auction. Finally, we use this formal setting to verify properties of two important classes of matching algorithms known as dynamic price and uniform price algorithms. In this work, we assume that each trader wishes to trade a single unit of product and the product is indivisible. In future this work can be extended to accommodate trades involving multiple units of an item. Another interesting direction of work would be extending this framework to include the analysis of continuous markets as well.

## References

- 1 Coq formalization of auctions. <https://github.com/suneel-sarswat/auction>.
- 2 The Coq Standard Library. <https://coq.inria.fr/library/>.
- 3 The coq proof assistant, version 8.7.1, December 15 2017. URL: <https://hal.inria.fr/hal-01673716>.
- 4 Kaustubh Deshmukh, Andrew V. Goldberg, Jason D. Hartline, and Anna R. Karlin. Truthful and competitive double auctions. *Lecture Notes in Computer Science*, 2461:361–??, 2002.
- 5 Daniel Friedman. The double auction market institution: A survey. 01 1993.
- 6 R Preston McAfee. A dominant strategy double auction. *Journal of economic Theory*, 56(2):434–450, 1992.
- 7 Jinzhong Niu and Simon Parsons. Maximizing matching in double-sided auctions. In Maria L. Gini, Onn Shehory, Takayuki Ito, and Catholijn M. Jonker, editors, *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '13, Saint Paul, MN, USA, May 6-10, 2013*, pages 1283–1284. IFAAMAS, 2013. URL: <http://dl.acm.org/citation.cfm?id=2484920>.
- 8 Grant Olney Passmore and Denis Ignatovich. Formal verification of financial algorithms. In Leonardo de Moura, editor, *Automated Deduction - CADE 26 - 26th International Conference on Automated Deduction, Gothenburg, Sweden, August 6-11, 2017, Proceedings*, volume 10395 of *Lecture Notes in Computer Science*, pages 26–41. Springer, 2017.
- 9 Peter R. Wurman, William E. Walsh, and Michael P. Wellman. Flexible double auctions for electronic commerce: theory and implementation. *Decision Support Systems*, 24(1):17–27, 1998.
- 10 Dengji Zhao, Dongmo Zhang, Md Khan, and Laurent Perrussel. Maximal matching for double auction. In Jiuyong Li, editor, *Australasian Conference on Artificial Intelligence*, volume 6464 of *Lecture Notes in Computer Science*, pages 516–525. Springer, 2010.