Formalizing double sided auctions in Coq

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

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

Abstract

In this paper we introduce a formal framework for analyzing double sided auction mechanisms in a theorem prover. In double sided auctions multiple buyers and sellers participate for trade. Any mechanism for double sided auctions to match buyers and sellers should satisfy certain properties of matching. For example, fairness, percieved-fairness, individual rationality are some of the important properties. These are critical 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 matching in a double sided auction. Finally, we use this framework to analyse properties of two important class of double sided auction mechanism. All the properties that we discuss in this paper are completely formalized in the Coq proof assistant.

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 has 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. This is known as algorithmic trading. 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 through a broker. These orders are matched by the stock exchange to execute trades. Most stock exchanges divide the trading activity into three main sessions known as pre-markets, continous markets and post markets. While in the pre-markets session an opening price of a product is discovered through double sided auction. In the continous markets session the incoming buyers and sellers are continously matched against each other on a priority basis. In the post-markets 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 [1]. In double sided auctions, auctioneer (e.g. stock exchange) collects buy and sell requests over a period. 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 period matches these orders based on their limit prices. This entire process is completed using a matching algorithm for double sided auctions.

Designing algorithms for double sided auctions is a well studied topic [2, 4, 3, 5]. 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. While an algorithm which clears each matched bid-ask pair at a single price is referred as uniform price algorithm. An algorithm which may clear each matched bid-ask pair at different prices is refered as dynamic price algorithm. There are other important properties besides the number of macthes which are considered while evaluating the effectiveness of a matching algorithm. For example, fairness, uniform pricing, individual rationality are some of the relevant features used to compare these matching algorithms. However, it is known that no single algorithm can posses all of these properties [4, 2].

In this paper, we describe 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 the product and all the products are indistinguishable as well as indivisible. We have used the Coq proof assistant to formally define the theory of double sided auctions. Furthermore, we use this theory to validate various properties of matching algorithms. We formally prove some important properties of two algorithms; a uniform price algorithm and a dynamic price algorithm.

2 Modeling double sided auctions

In this section we formally define various concepts involved in a double sided auction mechanism. The list data structure turns out to be very useful for decribing various properties of matching and the processes that operate on them in a double sided auction mechanism. In this section, we also describe some essential properties of lists which are used for stating important results on matching in a double sided auction.

2.1 Bid, Ask and limit price

An auction is a competetive event, where goods and services are sold to the highest bidders. In a double sided auction multiple buyers and sellers place their orders to buy or sell an item to an agent. The agent, known as auctioneer, 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. We can express bid as well ask using records containing two fields.

```
Record Bid: Type := Mk_bid { bp:> nat;    idb: nat }.
Record Ask: Type := Mk_ask { sp:> nat;    ida: nat }.
```

For a bid b, (bp b) is the limit price and (idb b) is its unique identity. Similarly for an ask a, (sp a) is the limit price and (ida a) is the unique identity of a. Note the use of coercion symbol `:>` in the first field of Bid. It declares bp as a function which is applied automatically to any term of type Bid that appears in a context where a term of type `nat` is expected. Hence, we can use the simple expression b instead of (bp b) to express the limit price of b. Similarly we can use a for the limit price of an ask a.

▶ Note. 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*.

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

2.2 Matching in Double Sided Auctions

In a double sided auction (DSA), the auctioneer collects all the buy and sell requests for a fixed duration. All the buy requests can be assumed to be present in a list B. Similarily, all the sell requests can be assumed to be present in a list A. At the time of auction, the auctioneer matches bids in B against asks in A. Furthermore, the auctioneer assigns a trade price to each matched bid-ask pair. This process results in a matching M, which consists of all the matched bid-ask pairs together with their trade prices. We represent matching as a list whose entries are of type `fill_type`.

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

In any matching M, a bid or an ask appears at most once. We say a bid-ask pair (b, a) is *matchable* if $b \geq a$ (i.e. $bp\ b \geq sp\ a$). Note that there might remain some bids in B which are not matched to any ask in a matching M. Similarly there might remain some asks in A which are not matched to any bid in M. The list of bids present in M is denoted as B_M and the list of asks present in M is denoted as A_M. For example, consider Fig. 1 which is a pictorial description of matching M between a list of bids B and a list of asks A. While the asks present in A is shown using left brackets and their limit prices. The bids present in B is shown using right brackets and their limit prices. All the matched bid-ask pair of M is then represented using matched brackets of same colors. For instance, the ask with limit price 53 is matched to the bid with limit price 69 in the matching M. Moreover, we can see that the bid with limit price 37 is not present in B_M since it is not matched to any ask in M.

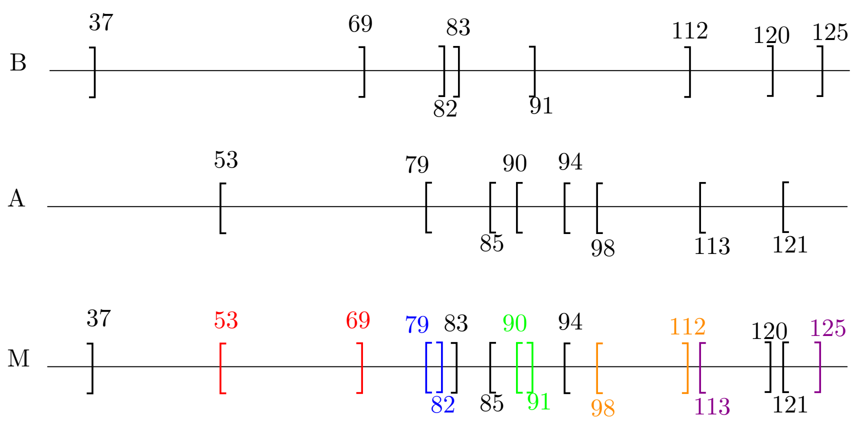


Figure 1 Bids in B and aks in A are represented using right and left brackets respectively. Every matched bid-ask pair in M is shown using the matched brackets of same colors. Note that the bids with limit prices 37, 83 and 120 are not matched to any ask in the matching M.

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

$B_M \subseteq B$, and (5) $A_M \subseteq A$.

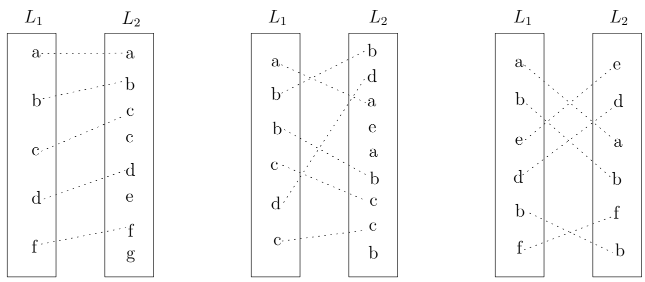
▶ **Definition 1.** *All_matchable M := ∀ m, In m M → (ask_of m) ≤ (bid_of m).*

▶ **Definition 2.** *matching_in B A M := All_matchable M ∧ NoDup* B_M *∧ NoDup* A_M $\wedge$ $B_M \subseteq B$ $\wedge$ $A_M \subseteq A$.

While term (*NoDup* B_M) in the above definition indicates that each bid is a request to trade one unit of item and the items are indivisible. We use the term $B_M \subseteq B$ to represent `Subset` relation between the lists B_M and B. It expresses the fact that each entry in the list B_M is also present in the list B.

Lists, sublist and permutation

The predicate `NoDup` and the `Subset` relation on lists are sufficient to define the notion of matching in a double sided auction. However, we need few more definitions to describe various proerties of matching as well as processes that operate on matching. In the following paragraphs we describe three such binary relations on lists namely `sublist`, `included` and `perm` which are essential for stating some useful results on matching in double sided auctions.



(a) sublist (b) included (c) perm

Figure 2 The dotted lines between entries of two lists confirm the presence of same entry 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 ocuurences in both the lists L_1 and L_2.

sublist L_1 L_2 **:** The notion of `sublist` is analogous to the subsequence relation on sequences. For the given lists L_1 and L_2 the expression `sublist` L_1 L_2 is `true` if every entry of L_1 is also present in L_2 and they apear in the same succesion. In Fig. 2(a) 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 lines intersect each other.

Let `T` be an arbitrary `eqType`. Then for any two lists l and s whoes elements are of type `T` we have following lemmas specifying the `sublist` relation.

▶ **Lemma 3.** *sublist_intro1 (a:T): sublist l s-> sublist l (a::s).*

▶ **Lemma 4.** *sublist_elim3a (a e:T): sublist (a::l)(e::s)-> sublist l s.*

▶ **Lemma 5.** *sublist_elim4: sublist l s -> (∀ a, count a l ≤ count a s).*

The term (*count a l*) in Lemma 5 represents the number of occurences of element a in the list l. Note the recursive nature of `sublist` as shown in Lemma 4. It usually makes inductive proofs easier for statements which contains `sublist` in the antecedent. Whereas, this is not true for the other relations (i.e. `included` and `perm`).

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

▶ **Lemma 6.** *included_intro: (∀ a, count a l ≤ count a s)-> included l s.*

▶ **Lemma 7.** *included_elim: included l s -> (∀ a, count a l ≤ count a s).*

▶ **Lemma 8.** *included_intro3: sublist l s -> included l s.*

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

▶ **Lemma 9.** *sorted_included_sublist: Sorted l -> Sorted s -> included l s -> sublist l s.*

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 in L_1. The notion of permutation for lists is analogus to the equality in multisets. In Fig 2(c) the list L_1 is perm of list L_2. We have following lemmas specifying the essential properties of the `perm` relation.

▶ **Lemma 10.** *perm_intro: (∀ a, count a l = count a s) -> perm l s.*

▶ **Lemma 11.** *perm_elim: perm l s -> (∀ a, count a l = count a s).*

▶ **Lemma 12.** *perm_sort: perm l s -> perm l (sort s).*

The term (*sort s*) in Lemma 12 represents the list s sorted using some ordering relation. Note that any permutation of a matching is also a matching. More precisely, we have the following invariance lemma on matching.

▶ **Lemma 13.** *match_inv: perm M M' -> perm B B' -> perm A A' -> matching_in B A M -> matching_in B' A' M'.*

▶ **Lemma 14.**

▶ **Lemma 15** (Lorem ipsum)**.** *Vestibulum sodales dolor et dui cursus iaculis. Nullam ullamcorper purus vel turpis lobortis eu tempus lorem semper. Proin facilisis gravida rutrum. Etiam sed sollicitudin lorem. Proin pellentesque risus at elit hendrerit pharetra. Integer at turpis varius libero rhoncus fermentum vitae vitae metus.*

Proof. Cras purus lorem, pulvinar et fermentum sagittis, suscipit quis magna.

▷ Claim 16. content...

Proof. content... ◁

◀

▶ **Corollary 17** (Curabitur pulvinar,)**.** *Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum. Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.*

▶ **Proposition 18.** *This is a proposition*

Proposition 18 and Proposition 18 . . .

2.3 Curabitur dictum felis id sapien

Curabitur dictum felis id sapien mollis ut venenatis tortor feugiat. Curabitur sed velit diam. Integer aliquam, nunc ac egestas lacinia, nibh est vehicula nibh, ac auctor velit tellus non arcu. Vestibulum lacinia ipsum vitae nisi ultrices eget gravida turpis laoreet. Duis rutrum dapibus ornare. Nulla vehicula vulputate iaculis. Proin a consequat neque. Donec ut rutrum urna. Morbi scelerisque turpis sed elit sagittis eu scelerisque quam condimentum. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Aenean nec faucibus leo. Cras ut nisl odio, non tincidunt lorem. Integer purus ligula, venenatis et convallis lacinia, scelerisque at erat. Fusce risus libero, convallis at fermentum in, dignissim sed sem. Ut dapibus orci vitae nisl viverra nec adipiscing tortor condimentum. Donec non suscipit lorem. Nam sit amet enim vitae nisl accumsan pretium.

2.4 Proin ac fermentum augue

Proin ac fermentum augue. Nullam bibendum enim sollicitudin tellus egestas lacinia euismod orci mollis. Nulla facilisi. Vivamus volutpat venenatis sapien, vitae feugiat arcu fringilla ac. Mauris sapien tortor, sagittis eget auctor at, vulputate pharetra magna. Sed congue, dui nec vulputate convallis, sem nunc adipiscing dui, vel venenatis mauris sem in dui. Praesent a pretium quam. Mauris non mauris sit amet eros rutrum aliquam id ut sapien. Nulla aliquet fringilla sagittis. Pellentesque eu metus posuere nunc tincidunt dignissim in tempor dolor. Nulla cursus aliquet enim. Cras sapien risus, accumsan eu cursus ut, commodo vel velit. Praesent aliquet consectetur ligula, vitae iaculis ligula interdum vel. Integer faucibus faucibus felis.

- Ut vitae diam augue.
- Integer lacus ante, pellentesque sed sollicitudin et, pulvinar adipiscing sem.
- Maecenas facilisis, leo quis tincidunt egestas, magna ipsum condimentum orci, vitae facilisis nibh turpis et elit.

▶ Remark 19. content...

3 Pellentesque quis tortor

Nec urna malesuada sollicitudin. Nulla facilisi. Vivamus aliquam tempus ligula eget ornare. Praesent eget magna ut turpis mattis cursus. Aliquam vel condimentum orci. Nunc congue, libero in gravida convallis , orci nibh sodales quam, id egestas felis mi nec nisi. Suspendisse tincidunt, est ac vestibulum posuere, justo odio bibendum urna, rutrum bibendum dolor sem nec tellus.

▶ **Lemma 20** (Quisque blandit tempus nunc). *Sed interdum nisl pretium non. Mauris sodales consequat risus vel consectetur. Aliquam erat volutpat. Nunc sed sapien ligula. Proin faucibus sapien luctus nisl feugiat convallis faucibus elit cursus. Nunc vestibulum nunc ac massa pretium pharetra. Nulla facilisis turpis id augue venenatis blandit. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.*

Fusce eu leo nisi. Cras eget orci neque, eleifend dapibus felis. Duis et leo dui. Nam vulputate, velit et laoreet porttitor, quam arcu facilisis dui, sed malesuada risus massa sit amet neque.

References

1 Daniel Friedman. The double auction market institution: A survey. 01 1993.

2 R Preston McAfee. A dominant strategy double auction. *Journal of economic Theory*, 56(2):434–450, 1992.

3 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.

4 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.

5 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.

A Styles of lists, enumerations, and descriptions

List of different predefined enumeration styles:

- \begin{itemize}...\end{itemize}
- ...
- ...

1. \begin{enumerate}...\end{enumerate}
2. ...
3. ...

(a) \begin{alphaenumerate}...\end{alphaenumerate}
(b) ...
(c) ...

(i) \begin{romanenumerate}...\end{romanenumerate}

(ii) ...
(iii) ...

(1) \begin{bracketenumerate}...\end{bracketenumerate}
(2) ...
(3) ...

Description 1 \begin{description} \item[Description 1] ...\end{description}
Description 2 Fusce eu leo nisi. Cras eget orci neque, eleifend dapibus felis. Duis et leo dui. Nam vulputate, velit et laoreet porttitor, quam arcu facilisis dui, sed malesuada risus massa sit amet neque.
Description 3 ...

B Theorem-like environments

List of different predefined enumeration styles:

▶ **Theorem 21.** *Fusce eu leo nisi. Cras eget orci neque, eleifend dapibus felis. Duis et leo dui. Nam vulputate, velit et laoreet porttitor, quam arcu facilisis dui, sed malesuada risus massa sit amet neque.*

▶ **Lemma 22.** *Fusce eu leo nisi. Cras eget orci neque, eleifend dapibus felis. Duis et leo dui. Nam vulputate, velit et laoreet porttitor, quam arcu facilisis dui, sed malesuada risus massa sit amet neque.*

▶ **Corollary 23.** *Fusce eu leo nisi. Cras eget orci neque, eleifend dapibus felis. Duis et leo dui. Nam vulputate, velit et laoreet porttitor, quam arcu facilisis dui, sed malesuada risus massa sit amet neque.*

▶ **Proposition 24.** *Fusce eu leo nisi. Cras eget orci neque, eleifend dapibus felis. Duis et leo dui. Nam vulputate, velit et laoreet porttitor, quam arcu facilisis dui, sed malesuada risus massa sit amet neque.*

▶ **Exercise 25.** *Fusce eu leo nisi. Cras eget orci neque, eleifend dapibus felis. Duis et leo dui. Nam vulputate, velit et laoreet porttitor, quam arcu facilisis dui, sed malesuada risus massa sit amet neque.*

▶ **Definition 26.** *Fusce eu leo nisi. Cras eget orci neque, eleifend dapibus felis. Duis et leo dui. Nam vulputate, velit et laoreet porttitor, quam arcu facilisis dui, sed malesuada risus massa sit amet neque.*

▶ **Example 27.** Fusce eu leo nisi. Cras eget orci neque, eleifend dapibus felis. Duis et leo dui. Nam vulputate, velit et laoreet porttitor, quam arcu facilisis dui, sed malesuada risus massa sit amet neque.

▶ Note 28. Fusce eu leo nisi. Cras eget orci neque, eleifend dapibus felis. Duis et leo dui. Nam vulputate, velit et laoreet porttitor, quam arcu facilisis dui, sed malesuada risus massa sit amet neque.

▶ Note. Fusce eu leo nisi. Cras eget orci neque, eleifend dapibus felis. Duis et leo dui. Nam vulputate, velit et laoreet porttitor, quam arcu facilisis dui, sed malesuada risus massa sit amet neque.

▶ Remark 29. Fusce eu leo nisi. Cras eget orci neque, eleifend dapibus felis. Duis et leo dui. Nam vulputate, velit et laoreet porttitor, quam arcu facilisis dui, sed malesuada risus massa sit amet neque.

▶ Remark. Fusce eu leo nisi. Cras eget orci neque, eleifend dapibus felis. Duis et leo dui. Nam vulputate, velit et laoreet porttitor, quam arcu facilisis dui, sed malesuada risus massa sit amet neque.

▷ Claim 30. Fusce eu leo nisi. Cras eget orci neque, eleifend dapibus felis. Duis et leo dui. Nam vulputate, velit et laoreet porttitor, quam arcu facilisis dui, sed malesuada risus massa sit amet neque.

▷ Claim. Fusce eu leo nisi. Cras eget orci neque, eleifend dapibus felis. Duis et leo dui. Nam vulputate, velit et laoreet porttitor, quam arcu facilisis dui, sed malesuada risus massa sit amet neque.

Proof. Fusce eu leo nisi. Cras eget orci neque, eleifend dapibus felis. Duis et leo dui. Nam vulputate, velit et laoreet porttitor, quam arcu facilisis dui, sed malesuada risus massa sit amet neque. ◀

Proof. Fusce eu leo nisi. Cras eget orci neque, eleifend dapibus felis. Duis et leo dui. Nam vulputate, velit et laoreet porttitor, quam arcu facilisis dui, sed malesuada risus massa sit amet neque. ◁