The Event Calculus on High-frequency Finance

Edward Tsang, Richard Olsen and Shaimaa Masry

Working Paper WP038-10
Centre for Computational Finance and Economic Agents (CCFEA)
University of Essex
Revised 22 October 2010

Abstract—To understand financial markets and prevent unnecessary crises, markets should be studied scientifically. With event calculus, this paper formalizes market clearance dynamics under simple market models. Using logic reduces ambiguity and enables rigorous reasoning. An algorithm for assessing risk is proposed. Real markets are more complex. This paper lays a solid foundation for studying market models.

Index Terms—High-frequency Finance, Financial markets, Event calculus.

1 Introduction

Financial markets are complex. Classical economics have been under serious challenge (e.g. see [10], [12]) to explain price action and volume flows in financial markets. One novel approach to market studies is to model the micro-behaviour of markets [12], [13]. The attempt is to observe micro-behaviour in the market with the aim to discover general dynamics [1], [6]. This approach is data-driven. Unlike classical economics, it does not depend on stringent assumptions, such as perfect rationality by the traders [14]. This new approach is still in its infancy. This paper looks at simple market models, and attempts to define the market dynamics formally.

The market can be described by states. The state of the market can be changed by events. In this paper, we limit our attention to buy and sell events initiated by market participants. Even though behaviour of the market participants may in general be unpredictable, certain inferences can be made. Given a set of buy and sell orders, the calculus can define state transitions. We can draw analogy with weather forecast, where although we may not know the long term weather changes, we can predict the immediate future given the current state; e.g. air flows from high pressure to low pressure regions.

Event calculus is useful for reasoning [4], [7], [8]. Shanahan states: "*The event calculus is a logical mechanism that infers what's true when given what happens when and what actions do*" [10] . This paper formalises the components relevant to the calculus for market transitions. It highlights the fact that the consequences of an order can be complex: the consequences are dependent on the positions and margins held by market participants. With this analysis, one can determine, for example, sizes of orders which can cause market crashes.

This paper formalises the obvious. But it is better to state the obvious with mathematical rigor rather than to leave it for potential ambiguity, which needs to be addressed repeatedly later in our research. Besides, what is obvious to some may not be obvious to others. Stating the obvious through event calculus enables us to study micro-behaviour rigorously.

- *E. Tsang is with the School of Computer Sceience and Electronic Engineering, University of Essex, CO4 3SQ. E-mail: edward@ essex.ac.uk*
- *R. Olsen is with the Olsen Ltd., Zurich, CH-8008. E-mail: richardo@olsen.ch*
- *S. Masry is with the Computational Finance and Economic Agents (CCFEA), University of Essex, CO4 3SQ. E-mail: smhmas@ essex.ac.uk*

2 Market Models

2.1 Model 1

This model is defined under a double auction market.

State + Orders → State

Where:

State = Queue_Profile = (Bid_Queue, Offer_Queue)

Bid_Queue = ((order$_1$, price$_1$, volume$_1$), (order$_2$, price$_2$, volume$_2$), , (order$_n$, price$_{bq}$, volume$_{bq}$))

Where price$_1$ > price$_2$ > … price$_{bq}$

Offer_Queue = ((order$_1$, price$_1$, volume$_1$), (order$_2$, price$_2$, volume$_2$), …, (order$_n$, price$_{oq}$, volume$_{oq}$))

Where price$_1$ < price$_2$ < … price$_{oq}$

The Bid_Queue comprises the bids to buy. The Offer_Queue comprises offers to sell. Buy (sell) orders having the same price are not merged.

Orders refer to a sequence of orders, where each order is either a bid or an offer, together with its volume.

Orders = (Order$_1$, Order$_2$, …, Order$_n$)

We assume that the orders are processed in sequence:

State + (Order$_1$, Order$_2$, …, Order$_n$) →
(State + Order$_1$) + (Order$_2$, …, Order$_n$)

For simplicity, we assume only two types of orders. A market order is to buy or sell at the market price. A limit order is to buy a certain volume up to a price specified, or to sell a certain volume above a price specified. For notional convenience, we write a market buy order as a limit buy order with the price set at infinity; a market sell order sets its price to minus infinity.

Order = (Order_No, Order_Type, Price, Volume)
Order_Type = bid | offer
Order_No = O_i

We define a symbol Inf, which stands for both infinity and minus infinity. We write a market buy order as (buy, Inf, Volume), a market sell order as (sell, Inf, Volume).

The calculus for clearance of a limit sell order can be defined below.

Let $Bid\_Queue_1 = ((O_1, P_1, V_1), (O_2, P_2, V_2) \ldots)$
$Offer\_Queue_1 = ((O_3, P_3, V_3), (O_4, P_4, V_4), \ldots)$
$Limit\_Order = (O_n, sell, P_n, V_n)$.

The calculus for a limit order is very simple. If the price of the sell order is less than or equal to at least the bid order at the head of the bid queue, the limit order can be fully or partially fulfilled. The sell order of volume V_n removes from the head of the Bid_Queue (P_1, V_1) the minimum of V_n or V_1. If V_n is greater than V_1, then the head of the Bid_Queue is removed. If the limit price is reached, clearing stops and the remaining unfulfilled sell order joins the offer queue. If the limit price is not yet reached, clearing continues with the remaining Bid_Queue until V is reduced to 0. If the price of the limit sell order is larger than the first bid order in the bid queue, then the sell limit order joins the offer queue. This can be formalised as follows.

$$((O_1, P_1, V_1), (O_2, P_2, V_2),\ldots) + (O_n, sell, P_n, V_n) \rightarrow$$
$$((O_3, P_3, V_3), (O_4, P_4, V_4), \ldots) \oplus (O_n, sell, P_n, V_n) \qquad \text{if } P_1 < P_n$$
$$((O_1, P_1, V_1 - (\min(V_1, V_n)), (O_2, P_2, V_2), \ldots) + (O_n, sell, P_n, V_n - (\min(V_1, V_n))) \qquad \text{if } P_1 \geq P_n$$

The + operation is recursive when $P_1 \geq P_n$; it stops when $P_1 < P_n$ or V_n is reduced to 0. Here $\oplus$ is the queue joining operator which simply put the orders in ascending order according to their prices.[1]

Cleared orders are removed from the bid queue:
$((O_1, P_1, 0), (O_2, P_2, V_2),\ldots) \rightarrow ((O_2, P_2, V_2), \ldots)$

Limit buy orders are handled symmetrically.
In the calculus above, the clearing of a market order is exactly the same as the limit order, except that market orders do not have limit prices and hence are always completely fulfilled. They do not join the bid or offer queues.

2.2 Example 1 for Model 1

With Model 1, the calculus for computing state transition is straight-forward. This example shows the state change for a given market order.

State 1.1 = (Bid_Queue1.1, Offer_Queue1.1)
Bid_Queue1.1 = ((O_1, 1.60, 2500), (O_2, 1.59, 2000), (O_3, 1.58, 2500), (O_4, 1.57, 1500), (O_5, 1.56, 4000))
Offer_Queue1.1 = ((O_6, 1.61, 3000), (O_7, 1.62, 2000), (O_8, 1.63, 1500))

Let Order1.1 = ($Order_9$, $Order_{10}$, $Order_{11}$), where
$Order_9$ = (O_9, sell, Inf, 5000)
$Order_{10}$ = (O_{10}, buy, 1.57, 1000)
$Order_{11}$ = (O_{11}, buy, 1.62, 6000)

With $Order_9$, which is a market order, the following transactions ensue:
2500 will be transacted at 1.60

This will result in the Bid_queue being reduced to:
(O_2, 1.59, 2000), (O_3, 1.58, 2500), (O_4, 1.57, 1500), (O_5, 1.56, 4000))
Next, the following two transactions will take place:
2000 will be transacted at 1.59
500 will be transacted at 1.58

The resulting state is:
State 1.2 = (Bid_Queue1.2, Offer_Queue1.2)
Bid_Queue1.2 = ((O_3, 1.58, 2000), (O_4, 1.57, 1500), (O_5, 1.56, 4000))
Offer_Queue1.2 = Offer_Queue1.1

With $Limit\_Order_{10}$, the offer queue is not changed as the price of the buy limit order is less than the price of the head of the offer queue. Since $Limit\_Order_{10}$ is not matched; it is added to the bid queue.

The resulting state is:
State 1.3 = (Bid_Queue1.3, Offer_Queue1.3)
Bid_Queue1.3 = ((O_3, 1.58, 2000), (O_4, 1.57, 1500), (O_{10}, 1.57, 1000), (O_5, 1.56, 4000))
Offer_Queue1.3 = Offer_Queue1.2

With $Limit\_Order_{11}$ (to buy 6000 with limit price 1.62), the offer queue is changed. Since the price 1.62 is greater than or equal to the first two orders in the offer queue, the following transactions will take place:
3000 will be transacted at 1.61
2000 will be transacted at 1.62

The remaining 1000 units will join the bid queue. Therefore, the resulting state is:
State 1.4 = (Bid_Queue1.4, Offer_Queue1.4)
Bid_Queue1.4 = ((O_{11}, 1.62, 1000), (O_3, 1.58, 2000),

[1] In functional programming convention, $\oplus$ is defined below:
$((P_1, V_1), (P_2, V_2), \ldots) \oplus (sell, P, V) \rightarrow$
$((P, V), (P_1, V_1), (P_2, V_2), \ldots) \qquad \text{if } P < P1$
$((P_1, V_1), ((P_2, V_2), \ldots) \oplus (sell, P, V))) \qquad \text{if } P \geq P1$

(O_4, 1.57, 1500), (O_{10}, 1.57, 1000),
(O_5, 1.56, 4000))
Offer_Queue1.4 = ((O_8, 1.63, 1500))

2.4 Model 2: When Positions and Margins are considered

The market dynamics will change when traders trade with margins. A trader with margin m, where 0<m≤1, will pay up only proportion m of the value that it trades. We make the following assumptions in our analysis:

Assumption 2.1. *For a trader with a short (long) position with margin m, its position is closed automatically when the price rises (falls) by more than m.*

Assumption 2.2. *All consequences of an automatic position closure take place before any new event occurs.*

***Assumption 2.3*[2].** *We assume that a position cannot be adjusted and is only only opened by a market or limit order. Position closure takes place automatically through margin calls. The relaxation of this assumption does not affect the generality of the results shown in our paper.*

Under this model, the description of a state must include traders' position profiles:

State = (Queue_Profile, Position_Profile)

Where:

Queue_Profile = (Bid_Queue, Offer_Queue)
Position_Profile = {Position | Position = (Position_Code, Position_Type, Volume, Value, Price, Margin)}
Position_No= P(O_i), where O_i is the Order_No of the order opening the position, given Assumption 2.3
Position_Type = long | short
Value = the value of the order(s) against which the opening position order has been matched.
Given:
Bid_Queue = ((O_1, P_1, V_1), (O_2, P_2, V_2), .., (O_{n-1}, P_{n-1}, V_{n-1}))
Order= (O_n, sell, Inf, V_n)
P(O_n)Value = (P_1* min(V_1,V_n)) +
(P_2* min(V_2, (V_n-min(V_n,V_2)))+...+
(P_{n-1}* min(V_{n-1},V_n-min(….)))
Price= Unit Price= Value/Volume

The clearance calculus is exactly the same as in Model 1, except that new events, namely new orders, can be triggered by state transitions.

Let TP = the last transaction price. Here, for simplicity, we assume that the last transaction price in "common sense". A more rigorous formalism should have it included in the state description. The martin-triggered set of new orders is NO:

NO = {(O_i, buy, Inf, V) | (P(O_i), short, Vol, Val P, m) ∈ Position_Profile such that P× (1+m) < TP} ∪ {(O_i, sell, Inf, V) | (P(O_i), long, Vol, Val P, m) ∈ Position_Profile such that P× (1–m) > TP}
Orders = Orders + NO

Here we make no assumption on how the set of new orders (NO) join the Orders queue; i.e. the "+" operator is yet to be defined. This is left to future refinement of the model.

2.5 Example 2 for Model 2: The effect of margin constraints

The following example shows the state transitions and how new events (which are limited to market orders in this model) are triggered.

Let:

State 2.1 = ((Bid_Queue2.1, Offer_Queue2.1), Positions2.1)
Bid_Queue2.1 = ((O_4, 1.60, 2500), (O_5, 1.59, 2000),
(O_6, 1.58, 2500), (O_7, 1.57, 1500),
(O_8, 1.56, 4000))
Offer_Queue2.1= ((O_9, 1.61, 3000), (O_{10}, 1.62, 2000),
(O_{11}, 1.63, 1500))
Positions2.1 = ((P(O_1), long, 4000, 6600, 1.65 4%),
(P(O_2), long, 2000, 3280, 1.64, 4%),
(P(O_3), long, 2000, 3280, 1.64, 5%))

For illustration purposes let us assume the following:

1. The position profile (Positions2.1) represents the current positions in the market created from previos orders.
2. Any new position in the market has a margin of 4%
3. Only one market order in the queue:

Order 2.1 = ((O_{12}, sell, Inf, 5000))

This is the same order that we used in Example 1. When it is cleared, as explained above, the bid queue will be changed. The state will be changed to:

State 2.2 = (Bid_Queue2.2, Offer_Queue2.2, Positions2.2)
Bid_Queue2.2 = ((O_6, 1.58, 2000), (O_7, 1.57, 1500),
(O_8, 1.56, 4000))
Offer_Quene2.2 = Offer_Queue2.1
Positions2.2 = ((P(O_1), long, 4000, 6600, 1.65 4%),
(P(O_2), long, 2000, 3280, 1.64, 4%),
(P(O_3), long, 2000, 3280,1.64, 5%),
(P(O_4), long, 2500, 4000, 1.60, 4%),
(P(O_5), long, 2000, 3180, 1.59, 4%),
(P(O_6), long, 500, 790, 1.58, 4%),
(P(O_{12}), short, 5000, 7970, 1.594, 4%))

Where:

P(O_{12}) Value = (1.6*2500)+(1.59*2000)+(1.58*500)= 7970
LastTP = 1.58 (the price of the last matched order in the Queue_Profile)

At this point, the bid queue and the position (long, 1.65, 4000, 4%) together will trigger a new market order. This is because 1.65×(1–4%) = 1.584, which is above the last transaction price, which was 1.580. Therefore, the margin is exceeded, and this position must be closed (Assumption 2.1). That means the order queue will be changed to:

Order 2.2 = ((O_{13}, sell, Inf, 4000))

[2] In a real market, a position is constructed via a set of orders. It can be openend, adjusted and closed by market and limit orders. Position closure takes place as a result of either a margin call or the trader's decision.

The following transactions take place:
2000 will be transacted at 1.58
1500 will be transacted at 1.57
500 will be transacted at 1.56

This will change the state to:
State 2.3 = ((Bid_Queue2.3, Offer_Queue2.3), Positions2.3)
Bid_Queue2.3 = ((O_8, 1.56, 3500))
Offer_Quene2.3 = Offer_Queue2.2
Positions2.3 = ((P(O_2), long, 2000, 3280, 1.64, 4%),
(P(O_3), long, 2000, 3280, 1.64, 5%),
(P(O_4), long, 2500, 4000, 1.60, 4%),
(P(O_5), long, 2000, 3180, 1.59, 4%),
(P(O_6), long, 2500, 3950, 1.58, 4%),
(P(O_{12}),short, 5000, 7970, 1.594, 4%),
(P(O_7), long, 1500, 2355, 1.57, 4%),
(P(O_8), long, 500, 780, 1.56, 4%))
Where:
LastTP = 1.56

Note that order O_6 has opened a new position P(O_6) in State2.2. However, it was only partially matched. In State2.3, O_6 is fully matched. Thus, we do not open a new position but we update the already opened position P(O_6).

The long position (long, 1.64, 2000, 4%) must be closed when the last transaction price (1.56 in this case) falls below its margin, which is 1.64×(1–4%) = 1.574. This means the order queue will be updated by the new market order:
Order 2.3 = (O_{14} sell, Inf, 2000)

When the order (sell, Inf, 2000) is matched, 2000 will be transacted at 1.56. This will reduce the state to:
State 2.4 = ((Bid_Queue2.4, Offer_Queue2.4), Positions2.4)
Bid_Queue2.4 = ((O_8, 1.56, 1500))
Offer_Queue2.4 = Offer_Queue2.3
Positions2.4 = ((P(O_3), long, 2000, 3280, 1.64, 5%),
(P(O_4), long, 2500, 4000, 1.60, 4%),
(P(O_5), long, 2000, 3180, 1.59, 4%),
(P(O_6), long, 2500, 3950, 1.58, 4%),
(P(O_{12}),short, 5000, 7970, 1.594, 4%),
(P(O_7), long, 1500, 2355, 1.57, 4%),
(P(O_8), long, 2500, 3900, 1.56, 4%))

Where:
LastTP = 1.56

Note that order O_8 has opened a new position P(O_8) in State2.3. However, O_8 was only partially matched. In State2.4, O_8 is fully cleared. Thus, we we update the already opened position P(O_8).

The position (long, 1.64, 2000, 5%) will only be closed when the price (LastTP) falls below 1.64×(1–5%) = 1.558. To summarize, a single market order of 5000 units led to the closure of two positions, which led to a total clearance of 11000 units, and a drop of 2.5% (from ≥1.60 to 1.56) in the market. It should be useful to compute, given a particular state of the market, how big an order is needed to drop the price by, say, 10%.

Besides, what would happen if the (long, 1.64, 2000, 5%) position has a 4% margin, instead of 5%? This will mean that this position has to be closed, but only 1500 of the 2000 will be bought (by the last bid in the queue); the remaining 500 units will not be cleared. The analysis of these properties goes beyond the scope of this simple calculus.

3 Consequential Closure

One can compute the consequential closure with respect to margin constraints. By doing so, one can evaluate the final state of any given event. For example, one would be able to say that "a market order to sell 6 million will lead to a price drop of 4%". One may also compute the condition for minimum price changes, e.g.
"What is the minimum size of a market sell order to lead to a price drop of r%?"

Answering questions like this would help to assess the stability of the market and value at risk. It could provide early warnings.

An algorithm as outlined below returns the volume of a market sell order that would lead the price to drop to or below price P_{drop}.

```
Function MinDrop(Queue_Profile, Position_Profile, P_drop);
  /* Let Queue_Profile = (Bid_Queue, Offer_Queue)
        Bid_Queue = ((P1, V1), (P2, V2), …, (Pbq, Vbq)) */
  i ← 1; Volume = 0;
  Repeat
        Queue_Profile' ← closure(Queue_Profile, Position_Profile,
            (offer, Inf, Volume));
        /* Let the bid queue for Queue_Profile' be ((P1', V1'), …) */
        If Vi < V1'
        Then {Volume ← Volume + Vi; i ← i + 1}
        Else Volume ← Volume + V1';
  Until P1' ≤ P_drop
Return Volume;
```

The procedure closure (Queue_Profile, Position_Profile, Order) computes the resulting Queue_Profile' after consequential closure is maintained using the calculus shown in the Model 2 Section.[3]

There exists a minimum k such that, for all the orders (P_i, V_i) at the front of the Bid_Queue, $P_{drop} \le P_i$ and Volume ≤ $V_1+V_2+...+V_k$. in the worst case, Function MinDrop has to go through all such (P_i, V_i)s.[4] Volume increases monotonically in Function MinDrop. Therefore this function must terminate.

Let M be the list of positions in the Position_Profile which margin calls are above P_{drop}. In the worst case, the procedure has to go through all of them. So each cycle of the Repeat loop will have complexity of |M|.

[3] Strictly speaking, the termination condition $P_1' \le P_{drop}$ should be replaced by LTP ≤ P_{drop}, where LTP is the Last transaction price which could be returned by the closure fuction. This is simplified for clarity. When the head of the queue in Queue_Profile' is below P_{drop}, any market order to sell will drop the price below P_{drop}. Therefore, the Volume returned is correct, which is our justification for the compromise.

[4] This is an upper-bound because any margin calls that might be triggered will absorb some of the volume.

Each "Then" part in each cycle of the Repeat loop would increase Volume to include one (P_i, V_i) pair. It is more complex to analyse the number of times that the "Else" part could be entered. In the worst case, each of the positions could bring the loop into the Else part through a margin call. Therefore, the complexity of the algorithm is bounded by $O(k \times |M|^2)$.

4 Market Making

The market maker absorbs a substantial amount of complexity (hence being rewarded). The market maker sets the "bid" and "ask" prices. The bid price is the price at which the market maker offers to buy; the ask price is the price at which the market maker offers to sell.

State = (Bid_price, Ask_price, MaxVol, Queue_Profile, Position_Profile)

Where:

Bid_price and Ask_price are the bid and ask prices quoted by the market maker;
MaxVol is the maximum volume that the market maker is willing to deal per order;
Queue_Profile and Position_Profile are the same as those defined in Model 2.

Here we assume that the clearing mechanism is completely automated. We define the clearing mechanism below. The key to the clearing mechanism is in the way that the market maker updates its bid and ask prices. In this paper, we make no assumption on f, which could vary from market maker to market maker; f should be a complex function.

Let Bid_price and Ask_price be the bid and ask prices in the current state, and Bid_price' and Ask_price' be the bid and ask prices in the next state. We generalize that the market maker sets the Bid_price' and Ask_price' with a function f, without specifying exactly what f is. f is a function that involves Bid_price, Ask_price, Queue_Profile, Position_Profile and many other factors, which may include the market maker's own position, bid and ask prices by the other market makers, the balance of payment between countries, interest rates, news and other economic indicators of the countries involved.

(Bid_price, Ask_price, MaxVol, (Bid_Q, Offer_Q), Positions) + (sell, P, V) $\rightarrow$
(Bid_price', Ask_price', MaxVol, (Bid_Q, Offer_Q), Positions) if $P \leq$ Bid_price & $V \leq$ MaxVol
(Bid_price', Ask_price', MaxVol, (Bid_Q, Offer_Q), Positions) + (sell, P, V–MaxVol) if $P \leq$ Bid_price & $V >$ MaxVol

(Bid_price', Ask_price', MaxVol, (Bid_Q, Offer_Q$\oplus$(sell, P, V)), Positions) if $P >$ Bid_price

The queue joining operator $\oplus$ is defined in the Model 1 Section.

If f could be written down mathematically, we would be able to define the event calculus in market making.

5 Liquidity

Artzner et al [3] proposed coherent measures of risk. This was challenged by Acerbi & Scandolo [1], [1] for not taking full consideration of liquidity risk. Acerbi & Scandolo introduced the marginal supply-demand curves (MSDCs) [1], which defines at any time instance the available prices of a given asset in the market. The attractiveness of their formalism is that liquidity risk is measured by market data; no assumptions are required. MSDCs are basically defined by queue profiles as defined above. Fig. 1 shows the MSDC in State 2.1. After clearing of Order 2.1, the market loses a certain amount of liquidity. This is shown by MSDC in Fig. 2.

The queue profile defines how liquid an asset is at any given time. Liquidity of an asset is therefore determined by how steep one ascends or descends in the MSDC. Following the above example, suppose at State 2.1, two traders bid 1.60 for 500 shares, and 1.59 for another 500 shares. Although the highest bid price is still 1.60, the new MSDC is actually steeper than the one shown in Fig. 1. This means, to sell over 1000 shares in this market (as opposed to the market shown in Fig. 1), the seller must be prepared to accept lower bids.

Unfortunately, the ordinary investors/traders who have no access to order books have no means of assessing their liquidity risk.[5] Therefore, market making provides investors/traders with market liquidity up to a certain limit (MaxVol in the Market Making Section). It also offers transparency in market liquidity. The MSDC under market making is shown in Fig. 3.

It is worth noting the obvious that, as a Queue Profile does not have to be symmetric, an asset could be highly liquid when one wants to buy, but illiquid when one wants to sell (and vice versa).

6 Conclusion

In this paper, we have defined a calculus for describing state changes in a market as a consequence of new orders coming in. We base our analysis on simple market models. This is no attempt to predict what new orders will arrive. The purpose of this paper is to lay the foundation for analyzing market states. We show that even with the simple calculus defined, we can ask important questions such as *"how big a sell order would push the price down by 10%?"* This research also supports Acerbi and Scandolo's call to measure liquidity risks with market data.

We acknowledge the fact that state changes in real markets are far more complex than what is described in this paper. It is up to the participants, including governing bodies, market makers and traders, to define the rules in an unambiguous mathematical and mechanical way. The aim is to create markets with properties that can be

[5] OANDA provides information on trader positions [9]. This could help conjecturing (with low confidence) marginal supply and demands (because eventually those in long positions have to sell, and those in short positions have to buy).

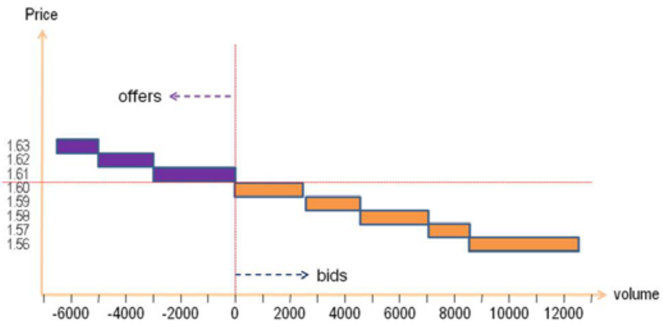

Fig. 1. The Marginal Supply-Demand Curve defined by the Queue Profile at State 2.1

studied scientifically. Eliminating black boxes and enabling scientific studies may be the best way to ensure stability and prevent financial crises. We intend to extend the

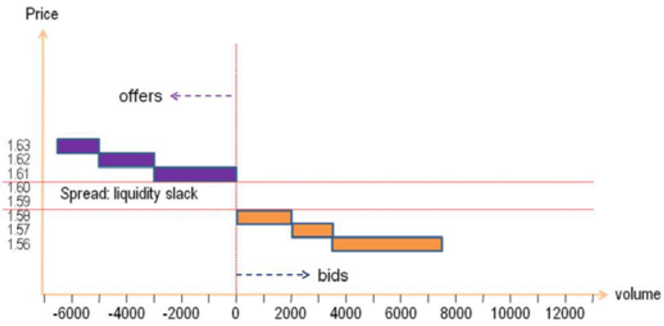

Fig. 2 The Marginal Supply-Demand Curve defined by the Queue Profile at State 2.2

calculus to cover more sophisticated market mechanisms and trading strategies.

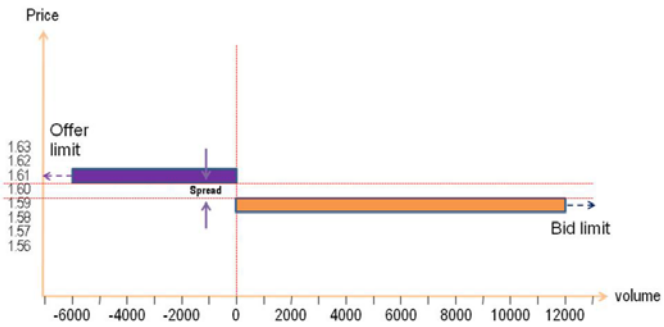

Fig. 1 The Marginal Supply-Demand Curve under market making

ACKNOWLEDGMENT

The authors wish to thank Monira Aloud for providing us with feedback and formatting the paper for us.

REFERENCES

[1] C. Acerbi and G. Scandolo, "Liquidity Risk Theory and Coherent Measures of Risk", *Quantitative Finance*, vol.8, pp. 681-692, 2008.

[2] C. Acerbi, "Portfolio Theory in Illiquid Markets", in Resti, A. (ed.), Pillar II in the new Basel Accord, Riskbooks, 2008, 241-272

[3] P. Artzner, F. Delbaen, J-M. Eber, and D. Heath, "Coherent measures of risk", *Math. Fin.*, vol.9, no.3, pp. 203-228, 1999.

[4] C-C. Chen, "Complex Event Types for Agent-Based Simulation", PhD dissertation, University College London, 2009

[5] M.M. Dacorogna, R. Gencay, U. Muller, R.B. Olsen, and O.V. Picktet, *An Introduction to High Frequency Finance*. Academic Press, 2001

[6] J.B. Glattfelder, A. Dupuis, and R. Olsen, "An Extensive Set of Scaling Laws and the FX Coastline", *Working Paper WP025-08*, Centre for Computational Finance and Economic Agents (CCFEA), University of Essex, September 2008.

[7] R. Kowalski and M. Sergot, "A Logic-Based Calculus of Events", *New Generation Computing*, vol.4, pp. 67-95, 1986.

[8] E.T. Mueller, "Aumated Commonsense Reasoning Using The Event Calculus", *Communications of the ACM*, vol.52, no. 1 , pp. 113-117, January 2009.

[9] OANDA http://fxtrade.oanda.com/tools/statistical_information/fx_open_position_summary

[10] R. Olsen, "Classical Economics: An Emperor with No Clothes", *Wilmott Magazine*, vol. 15, pp. 84-85, January 2005.

[11] M.P. Shanahan, "The Event Calculus Explained, in Artificial Intelligence Today", ed. M.J.Wooldridge and M.Veloso, *Springer Lecture Notes in Artificial Intelligence*, no. 1600, pp. 409-430, 1999.

[12] A. Shleifer, *Inefficient Markets: An Introduction to Behavioral Finance*. Oxford University Press, 2000

[13] S. Solomon, H. Levy, and M. Levy, *Microscopic Simulation of Financial Markets from Investor Behaviour to Market Phenomena*. Academic Press 2000

[14] E.P.K. Tsang, "Computational Intelligence Determines Effective Rationality", *International Journal on Automation and Control*, vol.5, no.1, pp. 63-66, January 2008.

Edward Tsang has a first degree in Business Administration (Major in Finance) and a PhD in Computer Science. He has broad interest in applied artificial intelligence, in particularly computational finance, heuristic search, constraint satisfaction and scheduling. He is currently a professor in computer science at the University of Essex where he leads the Computational Finance Group and Constraint Satisfaction and Optimization Group. He is also the Deputy Director of the Centre for Computational Finance and Economic Agents (CCFEA), an interdisciplinary centre. He chaired the Technical Committee for Computational Finance under the IEEE Computational Intelligence Society in 2004-2005.

Richard Olsen has a Master in Economics from Oxford University and a PhD in law from the University of Zurich. He has specialized in high frequney finance and has been a pioneer of this discipline. In 1995, he co-organized the first conference in the field. In 2001, he and his team published a book, 'Introduction to High Frequency Finance', Academic Press. He is CEO of Olsen Ltd, a systematic asset management company based in Zurich and chairman of OANDA, an Internet market maker of foreign exchange. He is visiting professor at the Centre for Computational Finance and Economic Ageants (CCFEA) since 2007.

Shaimaa Masry has a first degree in Management Information Systems and an MBA (Major IT). She is currently a PhD student in Computational Finance at the University of Essex. She is working on the High Frequency Finance Project. She has wide interest in computational finance, artificial intelligence and business intelligence.