

Property Propagation Rules for Prioritizing and Synchronizing Trading Activities

Yain–Whar Si, David Edmond, Arthur H.M. ter Hofstede, Marlon Dumas
Centre for Information Technology Innovation
Queensland University of Technology
GPO Box 2434, Brisbane Qld 4001, Australia
{y.si,d.edmond,a.terhofstede,m.dumas}@qut.edu.au

Abstract

With the growing number of marketplaces and trading partners in the e-commerce environment, software tools designed to act on behalf of human traders are increasingly used to automate trading activities. This paper describes a model for constructing trading engines which are capable of concurrently participating in multiple inter-related negotiations with heterogeneous protocols. These tree-structured engines are configured by means of a single generic synchronization construct which enables the incremental composition of complex trading schemes, including a number of well known strategies from the financial trading domain. The construct is augmented by a priority-based scheduling algorithm which selects a set of nodes for negotiation based on their estimated profit, the time remaining and the desired degree of concurrency. The model also provides iterative negotiation, which is essential in any complex trading environment.

Keywords: trading activities, negotiation protocols, concurrent negotiations, synchronization

1 Introduction

Electronic marketplaces, especially over the Internet, allow an increasing number of trading activities to be automated. Online auction houses (eBay, Yahoo), online exchanges (World Chemical Exchange, e-STEEL), and electronic communication networks (Instinet, Island) now provide the basic infrastructure for programmatic product discovery, quote polling, auctioning, bidding, order placement, trade settlement, etc. Already, several tools for trading partners discovery, price tracking, and automated bidding (among others) have emerged.

The next step in this evolution is the automation of *composite* trading activities. While a trading activity is usually

deployed to interact with at least one trading partner for buying/selling a unit of an item, a composite trading activity may need to interact with multiple trading partners and marketplaces concurrently, to trade in multiple units, to comply with temporal constraints, and to deal with specialized knowledge about market mechanisms and domain areas. Because of the subtle interactions between these characteristics, executing composite trading activities require rigorous planning and control, guided by carefully designed strategies.

Trading strategies can be viewed as guidelines and requirements to perform trading activities. Previous studies on the design of strategies for composite trading activities (e.g., [6] [13] [3]) assume an identical negotiation protocol across all trading activities (e.g., English auction). They are therefore not applicable to activities involving different negotiation protocols. Our work addresses this issue by proposing a common interface which is used to abstract the internal dynamics of trading activities.

Based on this common interface, a coordination model is presented, which can be used to configure trading engines for concurrently participating in multiple negotiations. In this model, composite trading activities are defined as assemblages of elementary and other composite trading activities governed by certain synchronization constraints.

An elementary trading activity handles a negotiation with a given marketplace or trading partner. It acts as a “wrapper”, in the sense that it hides (or rather abstracts from) the specificities of the negotiation protocol imposed by the trading partner. It also takes local decisions as to whether a given proposal is acceptable or not. A composite trading activity is specified using a generic synchronization construct. A composite trading activity aggregates and coordinates a number of other (elementary or composite) trading activities. Trading activities are also defined in such a way that the trader can dynamically update the constraints during the negotiation, thereby enabling the elemen-

tary trading activities to renegotiate at any time.

The execution of a composite trading activity is guided by a scheduling algorithm which selects a set of nodes for negotiation based on the estimated profit and the time factor. The nodes with higher profitability and less remaining negotiation time are given higher priorities. Using this priority-based scheduling algorithm, the model optimizes the overall profit and minimizes the negotiation duration.

A brief introduction to the concept of elementary trading activities, composite trading activities, and synchronization is given in section 2. The generic synchronization construct, the invariants, and the interface among of the components of the synchronization model are presented in section 3. The interface includes propagations of instructions, status, and price related properties. Rules for the propagation of instructions and status are given in section 4 and 5. Rules for propagation of price related properties (including profit estimation) are presented in section 6. The prioritization and generation of negotiation schedules based on the properties is presented in section 7. An application example of the synchronization construct to trading scenarios is given in section 8. In section 9 we briefly review related work before summarizing our ideas in section 10.

2 Interrelated trading activities

A key element in specifying trading strategies is the understanding of how basic elements of a trading activity can be defined. An *Elementary Trading Activity* (ETA) is defined as a set of operations required for the purpose of reaching a trading agreement (i.e. a deal). These operations are typically structured in three phases: discovery of trading partners, negotiation, and trade settlement. In this paper, we focus on the negotiation, where synchronization between several interrelated trading activities is required. An ETA is described by the following attributes: the action to be taken (eg. buy or sell), the description of the item (eg. name, the number of units), the description of the trading partners, the negotiation protocol deployed, and the temporal constraints. An example of an ETA from the financial trading domain is: "Negotiate with seller A to buy 2000 units of BHP with the price of \$10 using the bargaining protocol. The trade should be executed before 12:00 13-Nov-2001 and after 10:00 13-Nov-2001."

A *Composite Trading Activity* (CTA) may comprise one or more ETAs. For instance, a buy-sell CTA may include two ETAs, one for buying and one for selling. ETAs within a composite trading activity can be interrelated or independent. There are at least two possible types of relationships among two interrelated trading activities, *complementary* and *alternative*.

Complementary: Two trading activities are complementary [11] when both of them have to be successful or none of

them should be successful. For instance, in *bundle trading* (often found in the financial trading domain), a trader simultaneously purchases or sells an entire portfolio or a cross-section of a portfolio [4]. Index fund managers, index arbitrageurs, hedgers, and equity managers frequently employ bundle trading to maintain the diversification in their portfolio holdings. For example, a fund manager would like to sell 60% of shares from 10000 units of BHP stock and 70% of shares from 20000 units of NOKIA stock. In order to re-balance his/her portfolio, the fund manager is determined not to sell any stocks at all if one of these trades is not successful. In other words, all selling activities should be successful or none of them should be successful.

Alternative: Two trading activities are in an alternative (also known as "substitutive" [11]) relationship when only one of the activities has to be successful or none of them should be successful. For instance, the manager of a paper production factory may concurrently negotiate with two suppliers for 500 tons of pulp and be planning to choose the one with the lower acceptable offer. In this case, only one of the two ETAs will be successful. Alternative ETAs also occur in financial trading. In the US equity market, a significant amount of trades are carried out through one-to-one bargaining between brokers (e.g., in the so-called *upstairs markets* [9]). In this context, a trader can be involved in two alternative activities in which the same security is sought from two different brokers: these negotiations are synchronized so that at the end, the trader chooses the cheaper offer.

To the best of our knowledge, there is no tool supporting concurrent negotiations involving complementary and alternative trading activities. CTAs such as bundle trading and upstairs markets negotiation are still being carried out manually. Furthermore, a CTA can involve both complementary and alternative activities. An example is a fund manager planning a bundle trade which involves simultaneous purchase and sale of several stocks according to the following plan:

- buy either 10000 units of BHP from seller S1 or 5000 units of Yahoo from seller S2 and
- sell either 20000 units of NOKIA to buyer B1 or 6000 units of Amazon to buyer B2.

This example suggests that a CTA can be viewed as a composition of ETAs and other CTAs as shown in figure 1.

ETAs within the same CTA can in principle be executed concurrently. However, there are occasions during the negotiation when the ETAs need to be "synchronized" so that a global decision is taken. For example, in figure 1, ETAs (4) and (5) need to be synchronized to make sure that only one of them will be successful. ETAs (6) and (7) also need to be synchronized. In addition, both buying and selling activities, which are depicted as CTAs (2) and (3), need to be synchronized so that both of them will be successful or none

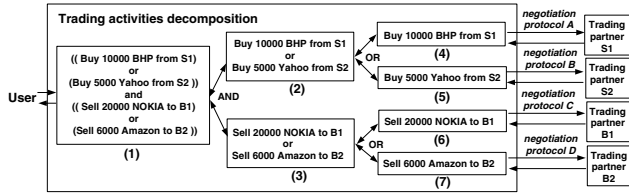


Figure 1. Nested bundle trading activity with heterogeneous negotiation protocols

will. These observations suggest a model where trading activities are represented as trees in which the leaves denote ETAs and the nodes denote CTAs which are instantiations of synchronization constructs.

3 Synchronization model

We introduce a generic **synchronization construct** which is simple yet highly adaptable to different trading requirements. Given (C_1, \dots, C_m) ($m > 0$) nodes to be synchronized, the synchronization construct **[Min..Max] OUT OF m** is informally defined as “**given m concurrent trading activities, at least Min activities and at most Max activities are to be successful or none of them should succeed**”. The formal semantics of the synchronization construct with $Min = Max$ is given in [14] using predicate/transition nets (PrT-nets). The bundle trading example given in figure 1 can be modelled using synchronization constructs as depicted in figure 2. The CTAs from figure 1 are replaced by appropriate constructs.

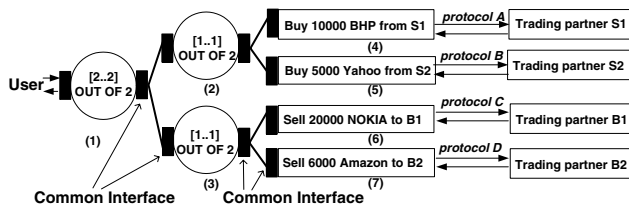


Figure 2. Bundle trading with synchronization constructs

An ETA (leaf node) can be either a seller or a buyer. In the context of negotiation between an ETA and a trading partner, we can classify the interactions between a ETA and its trading partner(s) into three groups according to following criteria; (a) only the ETA will send proposals, (b) only the trading partner will send proposals, and (c) both parties may send proposals. For instance, a buyer ETA in a Dutch auction is not required to send proposals whereas a buyer ETA in an English auction may send proposals to the trading partner (in this case, the auctioneer). Once the proposal is

accepted, it is required to be honored by the sender. The **polarity** of an ETA with a negotiation protocol which requires sending of proposals is said to be *binding*. The polarity of an ETA is defined as *non-binding* if the associated protocol is not required to send proposals. For instance, a buyer ETA with the Dutch auction protocol is non-binding whereas a seller ETA with the Dutch auction protocol is binding.

States: At any time, a node can be in one of six possible states: idle and available for synchronization, negotiating, stalled in negotiation but able to renegotiate under new constraints, ready-to-accept, successful in negotiation, and negotiation has ended and unable to renegotiate or continue. The possible transitions among these states are depicted in figure 3.

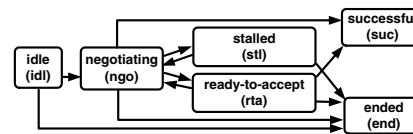


Figure 3. Possible states of a node

In order to avoid inconsistencies, it is crucial to define the invariants which must hold throughout the execution of a synchronization model. We define the following notation which will be used in specifying invariants.

Notation: Suppose that an internal node N is connected with a set of child nodes $N.Chi = \{C_1, \dots, C_m\}$ and $N.Min$ and $N.Max$ are the minimum and maximum number of child nodes to be achieved. Given an internal node N , the following functions return the set of child nodes which have non-binding and binding polarity respectively.

$$nonbd(N) = \{c \in N.Chi | c.Pol = nonbd\} \quad (1)$$

$$bd(N) = \{c \in N.Chi | c.Pol = bd\} \quad (2)$$

We define the following functions which return the set of child nodes of a particular state.

$$ngo(N) = \{c \in N.Chi | c.State = ngo\} \quad (3)$$

$$suc(N) = \{c \in N.Chi | c.State = suc\} \quad (4)$$

$$stl(N) = \{c \in N.Chi | c.State = stl\} \quad (5)$$

$$rta(N) = \{c \in N.Chi | c.State = rta\} \quad (6)$$

$$idl(N) = \{c \in N.Chi | c.State = idl\} \quad (7)$$

$$end(N) = \{c \in N.Chi | c.State = end\} \quad (8)$$

We also define functions which return the set of child nodes based on the negotiating state and the polarity.

$$ngo_nonbd(N) = ngo(N) \cap nonbd(N) \quad (9)$$

$$ngo_bd(N) = ngo(N) \cap bd(N) \quad (10)$$

Structural invariants: The following condition describes the valid range for the minimum and the maximum number of child nodes to be achieved.

$$\mathbf{C1.} \ 0 < N.Min \leq N.Max \leq |N.Chi|$$

We must also ensure that sufficient non-binding child nodes are available before the negotiation process begins. Non-binding child nodes are essential to achieve the required number of child nodes without actually being over-committed. For instance, suppose an internal node has two non-binding child nodes and one binding child and the synchronization assigned is [3..3] OUT OF 3. To achieve at least three child nodes or none, an internal node may start negotiating with two non-binding child nodes. Suppose that during the negotiation, both these child nodes have received acceptable proposals from their respective trading partners. As a result, their states are changed into ready-to-accept. Now the internal node can authorize the binding child node to negotiate. If the binding child node is successful, the two non-binding child nodes are instructed to accept. In this way, the minimum number of child nodes is guaranteed. If the binding child fails, the two waiting non-binding child nodes are instructed to abort so that none of them will be successful. This example demonstrates that there must be at least $(N.Min - 1)$ non-binding child nodes to fulfill the minimum number of child nodes for achievement. This condition is captured by the following precondition.

$$\mathbf{C2.} \ |nonbd(N)| \geq N.Min - 1$$

Negotiation invariants: There are certain conditions which must hold throughout the negotiation. The following invariant states that if an internal node has some successful child nodes, it must have sufficient ready-to-accept child nodes to guarantee the successful achievement of the minimum number of child nodes ($N.Min$).

$$\mathbf{I1.} \ |suc(N)| > 0 \Rightarrow |rta(N)| \geq N.Min - |suc(N)|$$

The following invariant states that if there are insufficient child nodes to guarantee the successful completion of $N.Min$ child nodes, no child node with binding polarity should be asked to negotiate.

$$\mathbf{I2.} \ (|suc(N)|=0 \wedge |rta(N)| < N.Min - 1) \Rightarrow |ngo_bd(N)|=0$$

To prevent the total number of successful child nodes exceeding the maximum requirement, the total number of currently negotiating child nodes which have binding polarity must not exceed the number of child nodes which are still allowed to be successful. This condition is captured by the following invariant.

$$\mathbf{I3.} \ |ngo_bd(N)| \leq N.Max - |suc(N)|$$

Common interface: ETAs within the synchronization model may employ different protocols to negotiate with different trading partners. Negotiation protocols frame the interactions between negotiating parties: what deals can be made and what sequences of offers are allowed [10]. Each negotiation protocol has its own distinct characteristics. For instance, in a Dutch auction the auctioneer begins with an initial high price and the price descends until someone states a desire to buy at the current price. The sequences of inter-

actions between two participants in a negotiation are different from one protocol to another. Therefore, a generic homogeneous interface is required to invoke, monitor, and control these trading activities.

In the generic interface (depicted as filled rectangles in figure 2), two types of messages are defined: **instructions** and **reports**. The type of instructions which can be sent are: (1) end all ongoing negotiation processes (below), (2) start all negotiation processes (below), (3) renegotiate all processes which are in stalled, negotiating or ready-to-accept state, and (4) accept negotiation processes which are ready to accept. There are two types of reports: **status reports** (successful, ended, ready-to-accept, stalled, polarity) and **property reports** (limit price, quote, profit, temporal constraints). Instructions are always propagated downwards and reports are propagated upwards. The following sections deal with the propagation of instructions, states, and property reports.

4 Propagation of instructions

Instructions are issued by the user or the internal nodes and propagated downwards. The negotiation process begins when the user issues the 'start' instruction to the root. When an idle internal node receives a 'start' instruction from its parent node (or the user), it determines which child nodes should be instructed for negotiation using a priority-based algorithm described in section 8 and sends 'start' instructions to those child nodes. The state of the internal node is then changed into 'negotiating'. This process is repeated at every internal node until instructions reach the leaf level. When a leaf node receives a 'start' instruction, it changes its state into 'negotiating' and begins negotiating with the designated trading partner(s).

During the negotiation, the user may issue the 'renegotiate' instruction with a new set of constraints (eg. new limit price) to the root of the tree. An internal node will only process the 'renegotiate' instruction if it is currently in negotiating, stalled, or ready-to-accept state. Otherwise, the instruction is simply ignored. If the internal node is in one of the above states, it immediately sends 'renegotiate' instructions to those of its child nodes which are also in negotiating, stalled, or ready-to-accept states. This process is repeated at every affected internal node until instructions reach the leaf level. When a leaf node receives a 'renegotiate' instruction, it sets its state to 'negotiating' and proceeds the negotiation with the new set of constraints.

If an internal node is in the ready-to-accept state and receives an 'accept' instruction from its parent node (or the user), it sends the 'accept' instructions to those of its child nodes which are also in the ready-to-accept state. The internal node also changes its state into 'successful'. This process is repeated until the instructions reach the leaf level.

When a leaf node receives an ‘accept’ instruction, it changes its state into ‘successful’ and sends the ‘accept–proposal’ message to its trading partner(s).

During the negotiation, the user may decide to terminate the negotiation process by issuing the ‘end’ instruction to the root. If an internal node receives an ‘end’ instruction, it changes its state into ‘ended’ and forwards the instruction to all of its affected child nodes. When a leaf node receives an ‘end’ instruction, it withdraws from the current negotiating process and changes its own state into ‘ended’.

5 Propagation of state and polarity

The current negotiation status and the polarities are determined at the leaf nodes and propagated upwards. Before the negotiation process begins, an internal node as well as all its child nodes are in the idle state ($|idl(N)| = |N.Chi|$).

State transition and polarity of leaf nodes: An idle leaf node changes its state into ‘negotiating’ when it receives a ‘start’ instruction from its parent node. During the negotiation, a node with binding polarity may send proposals (or bids) to its trading partners. If the proposal is accepted, the state of the node is changed into ‘successful’. If the polarity is non-binding, the node may only receive proposals from the trading partners. If the proposal received by the node is acceptable with respect to the limit price, the state of the node is changed into ‘ready–to–accept’.

The state of the non-binding node becomes ‘stalled’ when the proposal received from the trading partner is greater than the limit price (for a buying node), or the proposal received is less than the limit price (for a selling node). The state of the binding node becomes ‘stalled’ when the next proposal to be sent to the trading partner no longer satisfies the limit set by the user. A node also becomes ‘stalled’ when it cannot proceed due to the temporal constraints set by the user. The state of a node is changed into ‘ended’ when it receives an ‘end–of–negotiation’ message from its trading partner or an ‘end’ instruction is received from its parent node. Whenever a node changes into a new state, it is immediately reported to its parent node.

The polarity of a leaf node is defined by: (a) the intended action of the node (“buy” or “sell”), and (b) the negotiation protocols used (e.g. English auction, Dutch auction, continuous double auction). Since the polarity of a leaf node does not change, it is only reported once to the parent node before the negotiation begins.

Internal nodes: Based on the states and polarities of its child nodes, an internal node derives its own state and polarity based on the following rules. A node is considered as successful if and only if the number of successful child

nodes is between the desired minimum and maximum.

S1. $N.State = suc \Leftrightarrow N.Min \leq |suc(N)| \leq N.Max$

A node is considered as stalled if and only if the number of child nodes which are in the ended state is sufficiently low that it is still possible to continue, and the total number of child nodes which are in the ended or stalled state is sufficiently high so that it is impossible to achieve the minimum number of child nodes required to be successful, unless some negotiation parameters are changed.

S2. $N.State = stl \Leftrightarrow |end(N)| + |stl(N)| > |N.Chi| - N.Min \geq |end(N)|$

A node is considered as ended if and only if the total number of child nodes in the ended state is greater than or equal to the maximum number of child nodes allowed to be in the ended state.

S3. $N.State = end \Leftrightarrow |end(N)| > |N.Chi| - N.Min$

A node is considered as ready–to–accept if and only if it does not have enough successful child nodes and there are sufficient ready–to–accept child nodes to achieve the minimum requirement ($N.Min$).

S4. $N.State = rta \Leftrightarrow |rta(N)| \geq N.Min \wedge |suc(N)| < N.Min$

A node is considered as negotiating if and only if there is at least one negotiating child node and the node itself is not in successful, stalled, ready–to–accept, or ended state.

S5. $N.State = ngo \Leftrightarrow |ngo(N)| > 0 \wedge N.State \notin \{suc, stl, rta, end\}$

A node is non-binding if and only if it can be asked to negotiate in a non-binding way. (To negotiate in a non-binding way is to act in such a way that a successful state cannot be reached without first going through the ready–to–accept state.) A node is defined as non-binding if and only if there are no negotiating child nodes with binding polarity and there are sufficient non-binding child nodes (not in the ended state) to achieve the minimum requirement.

P1. $N.Pol = nonbd \Leftrightarrow |ngo_bd(N)| = 0 \wedge \{|c \in N.Chi | c.State \neq end \wedge c.Pol = nonbd\}| \geq N.Min$

Since there are only two possible polarities, we define a node as binding if its polarity is not non-binding. After an internal node has derived its own state and polarity, it reports them to its parent node. The process is repeated whenever there is a change in the state of its child nodes.

6 Propagation of price related properties

During the execution of a synchronization model, there is a need to compare or select nodes in terms of their price related properties such as limit price, quote, transaction cost, and estimated profit. Since those properties are directly available at the leaf nodes, it is necessary to define rules to propagate these properties from leaf nodes to the upper levels of the tree. The calculation of price related properties for an internal node is based on the concept of

valid negotiation arrangement: *a set of child nodes capable of changing the node into a successful state.*

Definition 6.1. A valid negotiation arrangement v of an internal node N is a subset of $N.Chi$ such that:

1. v only consists of child nodes from idle, stalled, negotiating, successful or ready-to-accept states.
 $|\{c \in v | c.State = end\}| = 0$
2. v must include all the successful child nodes.
 $suc(N) \subseteq v$
3. The cardinality of v must be between the minimum and maximum number of child nodes required to be successful. $N.Min \leq |v| \leq N.Max$
4. The total number of binding child nodes within v must not violate the invariants previously defined. According to **I2**, the total number of binding child nodes in v is zero if there are insufficient child nodes to guarantee the achievement of the minimum requirement.
 $(|suc(N)| = 0 \wedge |rta(N)| < N.Min - 1) \Rightarrow bd(v) = 0$
5. According to **I3**, the total number of binding child nodes in v must not exceed the number of child nodes which are still allowed to be achieved.
 $bd(v) \leq N.Max - |suc(N)|$

Definition 6.2. A valid negotiation arrangement lv of an internal node N is a lower bound valid negotiation arrangement iff: $|lv| = N.Min$

Limit price: The limit price of a leaf node is set by the trader prior to the negotiation. Its value is static throughout the negotiation unless it is modified by the user. The limit price of a *selling* leaf node indicates the minimum amount the user is expected to gain whereas the limit price of a *buying* leaf node indicates the maximum amount that the user is willing to spend. *The limit price of a selling leaf node is set to be positive and the limit price of a buying leaf node is set to be negative.* For the purpose of simplification, we define a predicate $itn(N)$ which is true if the node N is an internal node ($itn(N) \equiv |N.Chi| > 0$). The limit price of an internal node is 0 if its state is ended.

L1. $(itn(N) \wedge N.State = end) \Rightarrow N.Lim = 0$

The limit price of an internal node whose state is successful is equivalent to the sum of its successful child nodes' limit prices.

L2. $(itn(N) \wedge N.State = suc) \Rightarrow N.Lim = \sum_{c \in suc(N)} c.Lim$

A lower bound valid negotiation arrangement is a valid negotiation arrangement which is only able to achieve the minimum requirement ($N.Min$). For a given set of child nodes, there can be more than one lower bound valid negotiation arrangement. The limit price of an internal node represents the minimum amount the node is expecting to gain or the maximum amount the node is willing to spend by achieving the minimum requirement. Therefore the

limit price of an internal node which state is idle, negotiating, stalled or ready-to-accept is equivalent to the sum of the limit price of the child nodes within a negotiation arrangement which is the minimum among all available lower bound valid negotiation arrangements. Suppose that $lbset(N)$ is the set of all lower bound valid negotiation arrangements of N .

L3. $(itn(N) \wedge N.State \notin \{end, suc\}) \Rightarrow N.Lim = \min_{v \in lbset(N)} (\sum_{c \in v} c.Lim)$

Profit: The profit of a node in general is an indication of how attractive the current quote is with respect to the limit price. The profit of a leaf node returns a positive value when the current quote is higher than the limit price for a selling action or the current price is lower than the limit price for a buying action. The profit of a leaf node returns a negative value when the current quote is lower than the limit price for a selling action or the current quote is higher than the limit price for a buying action. Positive profit represents a desirable position whereas a negative profit represent an undesirable position to the trader. The profit also need to take into account the transaction cost. We define the profit of a node (regardless of whether it is a leaf or an internal node) as follows:

F1. $N.Pro = N.Quo - N.Lim - N.Tsc$

Maximum profit valid negotiation arrangement: Suppose that $vna(N)$ is the set of all valid negotiation arrangements of N . We define the function $mpv(N)$ of an internal node N , which returns a valid negotiation arrangement that maximizes the overall profit of N as follows:

$mpv(N)$ yields a randomly chosen $v \in vna(N)$ such that
 $\sum_{c \in v} c.Pro = \max_{i \in vna(N)} (\sum_{j \in i} j.Pro)$

Transaction cost: Transaction cost is one of the important issues in any trading situation. For instance, in financial trading, transaction costs imposed by the brokers and exchanges pose a significant overhead to both buyers and sellers. In our model, we assume that every leaf node has information about the estimated transaction cost. We also assume that transaction cost is zero for any unsuccessful deals. The following rule defines the transaction cost of an internal node which state is ended.

T1. $(itn(N) \wedge N.State = end) \Rightarrow N.Tsc = 0$

The transaction cost of an internal node whose state is successful is equivalent to the sum of its successful child nodes' transaction costs. This situation is captured by the following rule:

T2. $(itn(N) \wedge N.State = suc) \Rightarrow N.Tsc = \sum_{c \in suc(N)} c.Tsc$

The estimated transaction cost of an internal node whose state is idle, negotiating, stalled or ready-to-accept is equivalent to the sum of the transaction costs of the elements of the maximum profit valid negotiation arrangement.

$$\mathbf{T3.} (itn(N) \wedge N.State \notin \{end, suc\}) \Rightarrow N.Tsc = \sum_{c \in mpv(N)} c.Tsc$$

Quote: The quote at a seller leaf node is equivalent to the highest proposal received from a buyer trading partner and the quote at a buyer leaf node is equivalent to the lowest proposal received from a seller trading partner. Before any negotiation process begins, elementary trading activities (leaf nodes) will obtain the initial quotes from the trading partners. For instance, in an English auction, the initial price declared by the auctioneer can be treated as an initial quote. In a Dutch auction, the auctioneer may declare a relatively high initial price which can also be treated as an initial quote by the bidders. In bargaining protocols, the trading activities may request their respective trading partners to send initial quotes before the negotiation process begins. *The quote of a selling leaf node is set to be positive and the quote of a buying leaf node is set to be negative.* The following rule defines the quote of an internal node which state is ended.

$$\mathbf{Q1.} (itn(N) \wedge N.State = end) \Rightarrow N.Quo = 0$$

The quote of an internal node which state is successful is equivalent to the sum of its successful child nodes' quotes.

$$\mathbf{Q2.} (itn(N) \wedge N.State = suc) \Rightarrow N.Quo = \sum_{c \in suc(N)} c.Quo$$

The quote of an internal node which state is idle, negotiating, stalled, or ready-to-accept is equivalent to the sum of the quote values of the elements of the maximum profit valid negotiation arrangement.

$$\mathbf{Q3.} (itn(N) \wedge N.State \notin \{end, suc\}) \Rightarrow N.Quo = \sum_{c \in mpv(N)} c.Quo$$

7 Prioritization and scheduling

When a trader is able to negotiate with several trading partners to fulfill an objective, then naturally a schedule is required to decide with which partners to negotiate. When this principle is applied to a tree structured trading activity, each internal node has to decide which of its child nodes should be allowed to negotiate. As a result, a scheduling mechanism is required to choose the combination of activities (or nodes) which is likely to produce an optimum result.

7.1 Prioritizing child nodes for negotiation

Nodes will be selected by their parent nodes for negotiation based on the *priority value* θ . Nodes with higher priority value are more likely to be selected for negotiation. The priority value θ_N is computed as a weighted sum of the priorities relative to the expected profit and the time factor. The overall priority θ_N of node N is defined as follows:

$$\mathbf{R1.} \theta_N = \omega_P \theta_N^P + \omega_T \theta_N^T$$

ω_P and ω_T being the weights assigned by the user to each of the two factors. Weights are assigned to the root of the tree and replicated to all internal nodes. The user may require

that $\omega_P + \omega_T = 1$, although this is not a requirement of the model.

Priority based on profit factor (P): Profit factor allows an internal node to prioritize its child nodes in terms of their profit value. The profit factor of a node is the normalization of the profit of that node with respect to the other siblings of node N . Suppose that Z is the parent of N and abs is the absolute value function:

$$\mathbf{R2.} N \in Z.Chi \Rightarrow \theta_N^P = \frac{N.Pro + abs(\min_{c \in Z.Chi} c.Pro)}{\sum_{c \in Z.Chi} (c.Pro + abs(\min_{c \in Z.Chi} c.Pro))}$$

Suppose that an internal node has four child nodes which profits are -4, -2, 4, and 6. According to R2, their respective priority value are 0, 0.1, 0.4, and 0.5.

Priority based on external time factor (T): We assume that all the leaf nodes (elementary trading activities) have information about their respective trading partner end time. This concept of end time is propagated up to the root of the tree based on the following rule.

$$\mathbf{R3.} N.EndTime = \max_{c \in N.Chi} c.EndTime$$

The external time factor allows an internal node to prioritize its child nodes with respect to the end time. Nodes with fast approaching end times are given higher priorities. Suppose that *currenttime* and $N.EndTime$ are represented as the number of time units relative to some common origin.

$$\mathbf{R4.} currenttime < N.EndTime \Rightarrow \theta_N^T = \frac{currenttime}{N.EndTime}$$

If the current time has passed the deadline, the priority becomes 0.

$$\mathbf{R5.} currenttime \geq N.EndTime \Rightarrow \theta_N^T = 0$$

7.2 Total number of child nodes for negotiation

Once child nodes are prioritized, a set of child nodes (called *the negotiation schedule*) has to be selected for negotiation. In certain situations, the total number of child nodes selected can have significant impact on the market. Suppose that a trader is planning to buy a large number of identical items (e.g. 100000 units of a particular stock) by deploying five alternative non-binding trading activities which are assigned to five different sellers. If all activities are selected for negotiation, it may significantly increase the apparent demand of the item, which is probably visible to all market participants. In addition, withdrawing from the negotiation for those activities which fail to secure the deals may also affect his/her reputation. In order to avoid this situation, the *degree of concurrency factor* (CF) is used to control the total number of child nodes which are going to be selected for negotiation at one time. Just as with the profit and time factors in the previous section, the CF is assigned to the root of the tree by the user before the negotiation process begins and replicated to all internal nodes. We require that $0 \leq CF \leq 1$.

Non-binding child nodes: The total number of non-binding child nodes for negotiation φ_{nb} is equal to the number of child nodes still required to be successful (to achieve Min) plus an additional number of child nodes determined based on the concurrency factor. Suppose that x is the minimum number of child nodes that need to be selected in order to satisfy the minimum requirement and y is the total number of non-binding child nodes able to negotiate.

$$x = \max((N.Min - |suc(N)| - |rta(N)|), 0)$$

$$y = |\{c \in N.Chi | c.Pol = nonbd \wedge c.State \in \{idl, ngo\}\}|$$

If the number of available non-binding child nodes exceeds the number of child nodes needed to satisfy the minimum requirement, the number of non-binding child nodes to be instantiated is equal to the number of child nodes required to satisfy the minimum requirement plus the additional number of child nodes dictated by the concurrency factor.

$$\mathbf{N1.} \quad y \geq x \Rightarrow \varphi_{nb}^N = x + \lfloor (y - x) \times CF_N \rfloor$$

If the number of available non-binding child nodes is less than the minimum requirement, all the available non-binding child nodes should be instantiated for negotiation.

$$\mathbf{N2.} \quad y < x \Rightarrow \varphi_{nb}^N = y$$

Number of binding child nodes for negotiation: According to **I2**, the total number of binding child nodes for instantiation φ_b should be zero if there is are sufficient successful or ready-to-accept child nodes to guarantee the achievement of the minimum requirement ($N.Min$).

$$\mathbf{N3.} \quad (|suc(N)| = 0 \wedge |rta(N)| < N.Min - 1) \Rightarrow \varphi_b^N = 0$$

If there are sufficient successful or ready-to-accept child nodes to guarantee the achievement of the minimum requirement, the total number of binding child nodes for instantiation φ_b is bounded by $N.Max$. Suppose that x is the number of child nodes that can still be achieved and y is the number of binding child nodes available.

$$x = N.Max - |suc(N)|$$

$$y = |\{c \in N.Chi | c.Pol = bd \wedge c.State \in \{idl, ngo\}\}|$$

Using the degree of concurrency, the total number of binding child nodes for negotiation can be defined as follows:

$$\mathbf{N4.} \quad \neg(|suc(N)| = 0 \wedge |rta(N)| < N.Min - 1) \Rightarrow \varphi_b^N = \lfloor \min(x, y) \times CF_N \rfloor$$

7.3 Generating the negotiation schedule

Based on the priority value θ and the total number of child nodes to be instantiated, the algorithm to generate the negotiation schedule S_N of an internal node N is described as follows:

- Create an empty schedule S_N ;
- Create sets L_b and L_{nb} which contain all binding and non-binding child nodes c with $c.State \in \{idl, ngo\}$;
- Calculate φ_b^N and φ_{nb}^N ;
- For($i = 1$ to φ_b^N) {Remove child node C_i with the highest

- priority value from L_b and add C_i to S_N };
- For($j = 1$ to φ_{nb}^N) {Remove child node C_j with the highest priority value from L_{nb} and add C_j to S_N };
- Return S_N ;

Notice that the negotiation schedule can be empty if there are not enough idle or negotiating nodes to achieve the minimum number of child nodes. In this case, the user will be notified that some of the nodes in the stalled state need to be “unstalled” before the negotiation can proceed.

8 Case study

Suppose that a fund manager is planning to rebalance a portfolio. The research department has recently downgraded some stocks from the banking and telecommunications sectors based on their weaker than expected quarterly reports and has upgraded some of the stocks from the property, transportation and energy sectors due to their strong earning reports. The fund manager’s current assets are calculated based on the number of units available and the limit price per unit decided by the fund manager. The fund manager is planning to reduce the assets in the banking sector by 70% and the telecommunications sector by 100%. The fund manager is planning to buy six stocks from the upgraded sectors with the capital gained from the selling.

In the banking sector, the fund manager has 8000 units of HSBC, 20000 of Dah Sing Financial, and 27000 units of Wing Lung Bank. The fund manager is planning to sell any two combination of shares so that the banking asset will be reduced by approximately 70%. Synchronization construct $n1$ in figure 4 ensures that two selling activities will be successful or none of them will be successful. In the telecommunications sector, the fund manager has 30000 units of China Mobile and 160000 units of China Unicom. In order to reduce by 100%, both stocks will be sold. Synchronization construct $n2$ in figure 4 ensures that both selling activities will be successful or none of them will be successful. Synchronization construct $n6$ ensures that both $n1$ and $n2$ will be successful or none of them will be successful.

For the upgraded sectors (property, transportation, and energy), the fund manager is planning to buy one of the stocks from each sector. Synchronization constructs $n3$, $n4$, and $n5$ ensure that only one of the buying activities in each sector will be successful. Subsequently, synchronization construct $n7$ ensures that all buying activities will be successful or none of them will be successful. As part of the portfolio rebalancing, the fund manager is determined not to transact any deals unless all buying and selling requirements are satisfied. Synchronization construct $n8$ ([2..2] OUT OF 2) from figure 4 ensures that all buying and selling activities will be successful or none of them will be successful.

We have collected intra-day data from the Hong Kong Stock Market for the stocks described in our example.

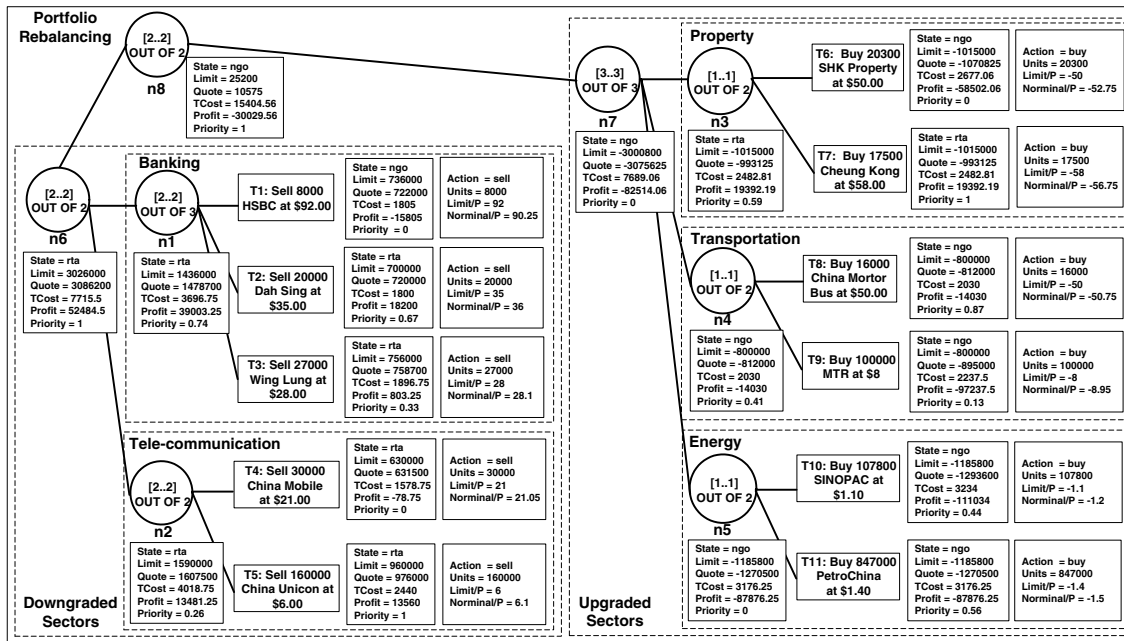


Figure 4. Data being propagated at 10:10AM on November 28, 2002

Minute by minute historical nominal prices are used as the quotes and fed into the synchronization model. The snapshot of the data propagation within a particular trading day is depicted in figure 4. Transaction cost at leaf nodes is estimated as 0.25% of quote value. Since it is financially infeasible to test (bid) with the real market, we simplify the procedure in the propagation of state. If the quote of a selling leaf node is greater than the limit price or the quote of a buying leaf node is less than the limit price per unit, the state is set to be ready-to-accept. Otherwise, the leaf node is defined as negotiating. Based on the state of the leaf nodes, the state of the internal nodes are determined according to the rules defined in section 5. For instance, in figure 4, $n1$ is in ready-to-accept state since at least two of its children ($T2$ and $T3$) are in ready-to-accept states. However, $n8$ is in negotiating state since only one of its child nodes ($n6$) is in ready-to-accept state.

Limits, quotes, transaction costs, and profits are calculated based on the rules described in section 6 and propagated up to the root. The priority value indicates how desirable a node is compared to its siblings in terms of potential profit. We omitted the calculation of the external time factor since all the trading activities have a uniform end time imposed by the market. For instance, trading task $T2$ in figure 4 has the highest priority since the profit of $T2$ is the highest among its siblings. This example shows how price related properties can be propagated during the synchronization of interrelated trading activities. These properties allow the model to make a projection of the maximum profit.

9 Related work

The Michigan Internet AuctionBot [15] is one of the earliest prototypes allowing bidding by human bidders as well as software agents. The AuctionBot is a configurable auction server which manages a large number of simultaneous auctions. In the first Trading Agent Competition (TAC) [8], participant agents bid in simultaneous English auctions for hotel rooms. These hotel rooms have to be packaged with flights and entertainment tickets so as to maximize their utility. The scenario from the competition differs from ours, in that the agents in [8] are free to purchase any number of hotel rooms whereas our synchronization construct is designed to achieve a specified range of deals.

Preist et al. [13] have proposed a coordination algorithm for an agent bidding in multiple English auctions for m identical units of an item. The algorithm ensures that the agent makes "at most" m purchases at the end. Their approach differs from ours in at least 2 ways. First, in [13] there is no mechanism to allow dynamic revision of constraints during the negotiation. Second, our approach provides modular compositions of complex trading activities.

Anthony et al. [2] have proposed a series of tactics as sets of decision functions for calculating future bids for an agent bidding in multiple heterogeneous auctions for a single item. The agent considered in [2] selects one auction at a time to bid for a single item. In contrast to [2], our approach performs concurrent negotiations to achieve multiple units of the same or different items.

Traditional database transaction processing [7] and advanced transaction models [5] address issues such as consistency, correctness and recovery for transactional tasks. Although the negotiation phase of a trading task is clearly different from a transactional task over a database, the multi-process coordination model described in this paper can be seen as an extension of the nested transaction model [12] which allows for repeated locking of the resources by elementary trading activities and internal nodes. Although the ready-to-accept state during the negotiation is somewhat similar to placing a lock on a data item in a database, the analogy stops there, since locking a data item does not have the same implications as locking a deal with a trading partner. Unlike transactional tasks over databases, rolling back or compensating a committed trading activity is impractical due to the possibility of financial loss.

To the best of our knowledge, the only commercial solution that provides a language for specifying trading strategies is TradeStation [1], which is intended for financial applications. EasyLanguage from TradeStation Technologies is a high-level programming language designed for analyzing securities data time-series, implementing trading rules and associated actions. However, it does not provide a direct solution to implement strategies which involve interrelated trading activities. For instance, there are no provisions to express conditional bids (e.g. complementary bids).

10 Conclusion

This paper described a novel model for synchronizing interrelated trading activities involving different forms of negotiation. A special interface is defined to homogenize the trading activities involved. In order to specify trading strategies, a generic synchronization construct is introduced. The synchronization construct also provides iterative negotiation capabilities which are essential in any complex trading environment. Rules for propagation of price and time related properties have been formalized. A priority-based scheduling algorithm which selects child nodes for negotiation based on the overall profit and the external time factor has been discussed.

Acknowledgment This work is partly funded by the Australian Research Council Grant “Self-describing transactions operating in a large, open, heterogeneous and distributed environment” involving the Queensland University of Technology and GBST Holdings Pty Ltd. The first author is also funded by the University of Macau.

References

[1] Easy Language, TradeStation Technologies. <http://www.tradestationopenplatform.com>.
 [2] P. Anthony, W. Hall, V. D. Dang, and N. R. Jennings. Autonomous agents for participating in multiple online auc-

tions. In *Proceedings of IJCAI Workshop on E-Business and the Intelligent Web*, 2001.
 [3] M. Dumas, L. Aldred, G. Governatori, A. H. ter Hofstede, and N. Russell. A probabilistic approach to automated bidding in alternative auctions. In *Proceedings of the 11th International Conference on the World Wide Web (WWW)*, pages 99–108. ACM Press, May 2002.
 [4] P. C. Dunn. Package trading. In W. H. Wagner, editor, *The Complete Guide to Securities Transactions, Enhancing Investment Performance and Controlling Costs*. John Wiley & Sons, 1989. 171-184.
 [5] A. K. Elmagarmid, J. G. M. Yungho Leu, and O. Bukhres. Introduction to advanced transaction models. In A. K. Elmagarmid, editor, *Database transaction models for advanced applications*, pages 33–47. Morgan Kaufmann Publishers, 1992.
 [6] E. Gimenez-Funes, L. Godo, J. A. Rodriguez-Aguilar, and P. Garcia-Calves. Designing bidding strategies for trading agents in electronic auctions. In Y. Demazeau, editor, *Proceedings of the 3rd International Conference on Multi-Agent Systems, (ICMAS'98)*. IEEE, 1998.
 [7] J. Gray and A. Reuter. *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann Publishers Inc, San Mateo, California, 1993.
 [8] A. Greenwald and P. Stone. Autonomous bidding agents in the trading agent competition. *IEEE Internet Computing*, 5(2):52–60, March–April 2001.
 [9] D. B. Keim and A. Madhavan. The upstairs market for large-block transactions: Analysis and measurement of price effects. *The Review of Financial Studies*, 9(1):1–36, Spring 1996.
 [10] A. R. Lomuscio, M. Woodridge, and N. R. Jennings. A classification scheme for negotiation in electronic commerce. In F. Dignum and C. Sierra, editors, *Agent-Mediated Electronic Commerce: The European AgentLink Perspective*, LNAI 1991, pages 19–33. Springer Verlag, 2001.
 [11] Y. Matsumoto and S. Fujita. An auction agent for bidding on combinations of items. In *Proceedings of the 5th International Conference on Autonomous Agents, AGENTS'01*, pages 552–559. ACM, 2001.
 [12] J. Moss. Nested transactions and reliable distributed computing. In *Proceedings of the 2nd. Symposium on Reliability in Distributed Software and Database Systems, Pittsburgh, PA*, pages 33–39. IEEE CS Press, 1982.
 [13] C. Preist, A. Byde, and C. Bartolini. Economic dynamics of agents in multiple auctions. In *Proceedings of the 5th International Conference on Autonomous Agents*, pages 545–551, Montreal, Canada, May 2001. ACM Press.
 [14] Y.-W. Si, D. Edmond, A. H. ter Hofstede, and M. Dumas. A model for the configurable composition and synchronization of complex trading activities. In *Proceedings of the 2003 ACM Symposium on Applied Computing, Melbourne, Florida, USA*, pages 595–602. ACM Press, 2003.
 [15] P. R. Wurman, M. P. Wellman, and W. E. Walsh. The Michigan Internet AuctionBot: a configurable auction server for human and software agents. In *Proceedings of the 2nd International Conference on Autonomous Agents*, pages 301–308. ACM, 1998.