Journal of Universal Computer Science, vol. 14, no. 7 (2008), 1118-1135
submitted: 1/10/07, accepted: 21/1/08, appeared: 1/4/08

Formalizing Agent-Based English Auctions Using Finite State Process Algebra

Amelia Bădică
(University of Craiova, Romania
ameliabd@yahoo.com)

Costin Bădică
(University of Craiova, Romania
badica_costin@software.ucv.ro)

Abstract: The vision of global agent-based e-commerce environments that enable dynamic trading between business partners requires the study and development of suitable formal modeling frameworks. In particular, negotiation is a necessary and important activity to allow engagement of business parties in non-trivial business relationships. In this paper we propose a formal framework based on *finite state process algebra* for modeling and analysis of interaction protocols in agent-based negotiations. The approach is demonstrated by applying the framework to model agent interactions in a single-item English auction scenario.
Key Words: multi-agent system, formal specification, process algebra, English auction
Category: I.2.11, K.4.4, I.2.4

1 Introduction

E-commerce is a key service of modern information society. Agent-based e-commerce is a recent trend in e-commerce research. In this context, the ability of e-commerce software agents to discover remote markets and engage in commercial transactions is of primary importance.

Negotiations (and auctions in particular) are complex activities frequently encountered in modern e-commerce processes that are characterized by a tight interaction of the business parties ([Laudon and Traver 2004]). Their analysis and understanding, especially when negotiations are automatized using software agents ([Lomuscio et al. 2002]), requires the study and development of suitable formal modeling frameworks.

Moreover, sound development of intelligent software agent systems requires the application of principled methodologies, as for example those proposed by [Padgham and Winikoff 2004] and [Cervenka and Trencansky 2007]. It is our belief that principled approaches should be based on formal representation languages and therefore formal methods will have to play a significant role in this area.

The development of formal frameworks for modeling agent interactions, including those encountered in negotiations and auctions, generated a lot of interest during the last years ([Bartolini et al. 2005], [van Eijk et al. 2003], [Esterline and Rorie 2001], [Hillston and Kloul 2001], [Weiliang et al. 2002], and [Rouff et al., 2006]). A similar interest has been manifested in formalizing business process notations to describe socio-economic activities ([Bădică et al. 2003], [Dong and Sheng 2003], [Feng et al. 2005], [Karamanolis et al. 2000], and [Puhlmann 2006]).

Many of these approaches utilize process algebras as foundational formalisms. Following the trend, this paper proposes a formal framework for modeling and analysis of interaction protocols in agent-based negotiations using *finite state process algebra* (FSP hereafter) ([Magee and Kramer 2006]). The approach is applied to model an English auction – a non-trivial and well-known auction mechanism, but it can be easily applied to other types of negotiations, like Dutch auctions, iterative bargaining, a.o (see [Wurman et al. 2001] for more examples). The benefits of our proposal are twofold: i) it allows formal verification of the system against a set of qualitative properties (see subsection 5.2), and it can be adapted to derive quantitative performance measures (like throughput, see [Hillston and Kloul 2001]); ii) it allows the development of formal models that can be used as a basis for the sound implementation of negotiation agents by mapping local processes (see FSP models in section 4) to agent behaviors.

The paper is structured as follows. We start in section 2 with a brief review of FSP and available modeling tool. In section 3 we introduce our negotiation model and show how it can be instantiated for English auctions. Then in section 4 we present detailed FSP models of agents in the negotiation system: negotiation host, buyer and seller. We follow in section 5 with an experimental evaluation and verification of our model. The last part of the paper discusses concluding remarks and future works.

2 Background on FSP

FSP is an algebraic specification technique of concurrent and cooperating computational processes as finite state labeled transition systems (LTS hereafter).

Definition 1. (labeled transition system) Let $\mathcal{S}$ be the universal set of states, $\mathcal{L}$ be the universal set of action labels and τ be the internal unobservable action. A *finite LTS* is a quadruple $P = \langle S, A, \Delta, q \rangle$ s.t.: i) $S \subseteq \mathcal{S}$ is a finite set of states; ii) $A = \alpha P \cup \{\tau\}$, where $\alpha P \subseteq \mathcal{L}$ denotes the alphabet of P; iii) $\Delta \subseteq S \times A \times S$ is the transition relation that maps a state and an action to another state; iv) $q \in S$ is the initial state of P.

LTS models are suitable for specifying discrete-event systems. However, descriptions, either visual or textual of LTS models as labeled directed graphs are impractical for more than a few states. For this reason the FSP process algebra has been proposed ([Magee and Kramer 2006]). FSP uses the following constructs: prefix, choice, parallel composition, re-labeling and definition (see [Magee and Kramer 2006] for more details).

i) *Prefix.* The process $a \to P$ performs the action a and then behaves like P. The prefix operator specifies sequential execution of actions.

ii) *Choice.* The process $P \mid Q$ behaves either like P or like Q. If both are enabled then the choice is non-deterministic.

iii) *Parallel composition.* The composite process $P \parallel Q$ specifies the interaction between processes P and Q on the common set of actions in their alphabets αP and αQ. This means that for actions outside set $\alpha P \cap \alpha Q$, P and Q proceed independently, but for actions in $\alpha P \cap \alpha Q$, P and Q must cooperate and proceed together.

iv) *Re-labeling.* Re-labeling functions applied to a process term change the names of the action labels. The process P/L where $L = \{nl_1/ol_1, \ldots, nl_k/ol_k\}$, $ol_i \in L$, $nl_i \subseteq L$, behaves like P excepting that any action ol_i appears to an external observer as any of the actions in the set nl_i, for all $1 \leq i \leq k$.

v) *Definition.* A definition $A = P_A$ associates the behavior of the process term P_A with the name A. You can then use A in process terms to describe more complex behaviors. Thus A is interpreted as the name of a re-usable process component.

The syntax of FSP is introduced in two steps: i) definition of sequential processes; ii) definition of composite processes. The key point is to not arbitrarily mix choices and parallel compositions in order to preserve the finiteness of the state space ([Magee and Kramer 2006]).

The set of process names is partitioned into the sets $\mathcal{P}_S$ of sequential process names and $\mathcal{P}_C$ of composite process names[3]. Let $\mathcal{L}$ be the set of all action labels.

Definition 2. (sequential process) A *sequential process term* is defined according to the following rules: a) END is a sequential process term denoting an empty process that engages in no further actions; b) If $SPN \in \mathcal{P}_S$ then SPN is a sequential process term; c) If $a_i \in \mathcal{L}$ and SP_i are sequential process terms, $1 \leq i \leq k$, then $a_1 \to SP_1|a_2 \to SP_2|\ldots|a_k \to SP_k$ is a sequential process term.

A sequence $SPN_1 = SP_1, \ldots, SPN_p = SP_p$ s.t. $SPN_i \in \mathcal{P}_S$ and SP_i are sequential process terms, $1 \leq i \leq p$, defines a sequential process with name SPN_1. Definitions of SPN_i, $2 \leq i \leq p$, are called *local definitions.*

[3] Elements of $\mathcal{P}_C$ are distinguished from elements of $\mathcal{P}_S$ by prefixing with $\parallel$ (see appendix)

Definition 3. (composite process) A *composite process term* is defined with the following rules: a) If $PN \in \mathcal{P}_C \cup \mathcal{P}_S$ then PN is a composite process term; b) If CP_i are composite process terms, $1 \leq i \leq k$, then $CP_1 \parallel CP_2 \parallel \ldots \parallel CP_k$ is a composite process term; c) If CP is a composite process term and L is a re-labeling function then CP/L is a composite process term.

If $CPN \in \mathcal{P}_C$ and CP is a composite process term then $CPN = CP$ defines a composite process with name CPN.

A FSP model consists of a finite set of sequential and/or composite process definitions such that no process name occurring in the right-hand side of a composite process definition was left undefined. FSP has an operational semantics given via a LTS. The mapping of a FSP term to a LTS is described in detail in [Magee and Kramer 2006] and it follows the intuitive meaning of FSP constructs introduced in this section.

3 Agent Negotiation Model

We understand automated negotiations as a process by which a group of software agents communicate with each other to reach a mutually acceptable agreement on some matter ([Lomuscio et al. 2002]). In this paper we focus our attention on *auctions* – a particular form of negotiation where resource allocations and prices are determined by bids exchanged between participants according to a given set of rules ([McAfee and McMillan 1987]).

In automated negotiations (including auctions) it is important to distinguish between *negotiation protocols* (or *mechanisms*) and *negotiation strategies*. The protocol comprises public "rules of encounter" between negotiation participants by specifying the requirements that enable them to interact and negotiate. The strategy defines the private behavior of participants aiming at achieving their desired outcome ([Wooldridge 2002]).

Our negotiation model follows the generic software framework for automated negotiation proposed by [Bartolini et al. 2005] and it is specialized for the particular case of English auctions following implementation details using JADE [JADE] and JESS [JESS] that were reported in [Bădică et al. 2006b], [Bădică 2007c]. So this work can also be seen as an attempt to formalize behavior and interactions of negotiation agents, as defined by that implementation.

Authors of [Bartolini et al. 2005] sketched a software framework for implementing agent negotiations that comprises: (1) negotiation infrastructure, (2) generic negotiation protocol and (3) taxonomy of declarative rules. The *negotiation infrastructure* defines roles of negotiation participants (eg. *Buyer* or *Seller* in an auction) and of a negotiation host – a specialized arbitrator middle-agent as described in [Bădică 2007a]. According to the *generic negotiation protocol*

([Bartolini et al. 2005]), participants exchange proposals (or bids) via a common space (or market) that is governed by an authoritative entity – the negotiation host (or market maker). Status information describing negotiation state and intermediary information is automatically forwarded by the host to all entitled participants according to the information revealing policy of that particular negotiation ([Bartolini et al. 2005], [Bădică 2007c]). *Negotiation rules* deal with the semantic constraints of a particular negotiation mechanism (e.g. English auctions). Rules are used for checking validity of proposals and sequences of exchanged messages, updating of negotiation status and informing participants, and controlling agreement formation and negotiation termination.

Formal modeling of an agent-based English auction requires a precise description of the generic negotiation protocol and of semantic constraints specific to English auctions. In order to introduce the agent negotiation model, we follow [Bartolini et al. 2005] by representing messages using FIPA ACL ([FIPA]).

The *generic negotiation protocol* controls how messages are exchanged by the host and participants by facilitating the following negotiation activities: (1) admission to negotiation, (2) proposal (or bid) submission, (3) informing participants about the change of negotiation state, (4) agreement formation and (5) negotiation termination.

Admission to negotiation. This activity starts when a new participant requests admission by sending a PROPOSE message to the host. The host grants (or not) participant admission responding with either an ACCEPT-PROPOSAL or a REJECT-PROPOSAL message. In particular, the first admission request (always submitted by a seller participant in an English auction) initiates the negotiation.

Proposal submission. Participants may enter the phase of submitting bids after they were admitted to the negotiation. The generic negotiation protocol states that a participant will be notified by the host if his proposal was either accepted (with an ACCEPT-PROPOSAL) or rejected (with an REJECT-PROPOSAL).

Informing participants. The negotiation protocol requires that participants will always be notified (with INFORM messages) about any update of the negotiation state that is visible to them according to the visibility rules.

Agreement formation can be triggered at any time during negotiation. When agreement formation rules signal that an agreement was reached, the protocol states that participants involved in the agreement will be notified by the host with INFORM messages.

Negotiation termination can be triggered at any time during negotiation. When negotiation termination rules signal that the negotiation process reached its final state, the protocol states that all participants will be notified by the host with INFORM messages.

We now follow with a brief and concise description of English auctions. Technically, English auctions are single-item, first-price, open-cry, ascending auctions ([Laudon and Traver 2004],[Wooldridge 2002]). In an English auction there is a single item sold by a single seller and many buyers bidding against one another for buying the item until the auction terminates. Usually, there is a time limit for ending the auction (either a total time limit or a certain inactivity period), a seller reservation price that must be met by the winning bid for the item to be sold and a minimum value of the bid increment. A new bid must be higher than the currently highest bid plus the bid increment in order to be accepted. All the bids are visible to all the auction participants, while seller reservation price is private to the auction.

4 FSP Model of Agent Negotiation

4.1 Negotiation Structure

A *negotiation structure* defines a general framework that statically constraints a given negotiation. It consists of a set of roles that contains a *negotiation host* role and one or more *negotiation participant* roles.

The *negotiation host* role orchestrates the negotiation and coordinates negotiators by employing the generic negotiation protocol.

A *negotiation participant* role describes the behavior of a negotiator that plays an active role in the negotiation. Usually, two negotiation participant roles are defined – *buyer* and *seller*. For example, in an English auction there is a single *seller* participant and one or more *buyer* participants, while in an reverse English auction there is a single participant with role *buyer* and one or more participants with role *seller*.

A negotiation process is always initiated by a certain participant known as *negotiation initiator*. Usually is required that the initiator has a given negotiation role – *negotiation initiator* role. For example, in an English auction the initiator has always role *seller*, while in a reverse English auction the initiator has always role *buyer*.

Focusing our discussion on auctions for buying and selling goods, a negotiation structure can be formally defined as follows:

Definition 4. (negotiation structure) A *negotiation structure* is a triple $N = \langle Host, Seller, Buyer, Initiator \rangle$ such that: i) $Host$ is the *negotiation host* role; ii) $Seller$ is the *seller* role that defines behavior of participants selling goods in the auction; iii) $Buyer$ is the *buyer* role that defines behavior of participants buying goods in the auction; iv) $Initiator$ is the role that is allowed to initiate the auction – either *buyer* or *seller*, i.e. $Initiator \in \{Buyer, Seller\}$.

In our formal modeling framework behaviors of negotiation roles are described using FSP. Therefore we shall have FSP processes describing $Host$, $Seller$ and $Buyer$ roles. A participant behavior is defined by instantiating his role. Finally, the behavior of the negotiation system is defined using parallel composition of roles for each negotiation participant, including of course the negotiation host.

4.2 Negotiation Host

In what follows we shall assume that our negotiation host is able to handle a single negotiation at a certain time. In other words, the negotiation host functions as a *one-at-a-time server*. In order to handle multiple negotiations concurrently, several negotiation host instances must be ran concurrently. However, as focus of this paper is to formally describe a single negotiation, we do not explore further this path.

Following the generic negotiation protocol described in the previous subsection, a negotiation will consist of a series of stages. In what follows these stages are particularized for the case of an English auction:. Note that submissions of FIPA ACL messages are modeled using suitable FSP actions, as follows:

i) *initiation* – the negotiation is initiated by the seller using the $init$ action; note that initiation acts also as a registration of the seller agent participant; initiation is either accepted (action $accept\_init$) or rejected (action $reject\_init$) by the host;

ii) *buyer registration* – each buyer agent must register with the negotiation using $register$ action before she is allowed to submit bids; registration is granted (action $accept\_registration$) or not (action $reject\_registraton$) by the negotiation host;

iii) *bids submission* – each registered buyer agent is allowed to submit bids using the *bid* action. Bids can be either accepted (action $accept\_bid$) or rejected (action $reject\_bid$) by the host. When a certain bid is accepted, the other registered buyer participants are notified accordingly by the host using action $inform$. Additionally to the generic negotiation introduced in the previous section, we have also chosen to model the event corresponding to the buyer decision to cancel bidding – action $cancel\_bid$.

iv) *agreement formation* – when the host observes a certain period of bidding inactivity, it triggers negotiation termination via action *stop*. This event subsequently triggers agreement formation. In this stage the host checks if an agreement can be generated. If no buyer has registered before the negotiation terminated then no agreement can be made and action no_win with no parameter is executed. However, if at least one buyer has successfully

Table 1: *Server* process that describes the negotiation host role.

$$
\begin{array}{ll}
Server & = init \rightarrow AnswerInit, \\
AnswerInit & = accept\_init \rightarrow ServerBid(\perp, \emptyset) | \\
 & \quad reject\_init \rightarrow Server, \\
ServerBid(chb, Bs) & = bid(b \in Bs) \rightarrow AnswerBid(b, chb, Bs) | \\
 & \quad stop \rightarrow ServerAgreement(chb) | \\
 & \quad register(b' \notin Bs) \rightarrow AnswerReg(b', chb, Bs), \\
AnswerReg(b', chb, Bs) & = accept\_registration(b') \rightarrow ServerBid(chb, Bs \cup \{b'\}) | \\
 & \quad reject\_registration(b') \rightarrow ServerBid(chb, Bs) | \\
AnswerBid(b, chb, Bs) & = accept\_bid(b) \rightarrow InformBuyers(b, Bs) | \\
 & \quad reject\_bid(b) \rightarrow ServerBid(chb, Bs), \\
InformBuyers(b, Bs) & = inform(b_1) \rightarrow inform(b_2) \rightarrow \ldots \rightarrow \\
 & \quad inform(b_k) \rightarrow ServerBid(b, Bs), \\
ServerAgeement(\perp) & = no\_win \rightarrow Server, \\
ServerAgeement(chb) & = win(chb) \rightarrow Server | \\
 & \quad no\_win(chb) \rightarrow Server.
\end{array}
$$

submitted an accepted bid then the host will have to decide if there is a winner (action *win*) or not (action *no_win* with parameter) depending on if the currently highest bid overbids or not the seller reservation price.

Note that message contents (i.e. bid value or submission time), with the exception of buyer identities, are ignored in our model. Moreover, as the FSP model deliberately ignores message contents, focusing only on the interaction patterns between negotiation participants, the application of negotiation rules is only assumed and implicitly incorporated in the model. For example, in the FSP model: i) negotiation is always initiated by the seller, ii) seller only initiates the auction but she does not submit any other bids, iii) applications of bid validity rules, negotiation termination rules and agreement formation rules are not explicitly shown, while results of their applications are modeled as a nondeterministic choices.

Negotiation host behavior is described as the *Server* process model shown in table 1. Note that *Server* process has a cyclic behavior and thus it runs infinitely, being able to handle an infinite sequence of negotiations, one negotiation at a time.

In a real setting, participant agents (buyers and sellers) can be created and destroyed dynamically. In our model we assume there is a given set of buyers and a single seller that are created when the system is started. Buyers are able to dynamically register to negotiations. Whenever a new negotiation finishes, a new one can be immediately initiated by the seller agent and buyers are required to register again in order to be able to participate and bid for buying the sold product.

Assuming each buyer agent has a unique name, let $\mathcal{B}$ be the set of all names of buyer agents that were created when the system was initiated and let be $\perp$ a name not in $\mathcal{B}$. Definition of the $Server$ process is using several indexed families of local processes:

- $ServerBid(b, B)$ such that $b \in B \cup \{\perp\}, B \subseteq \mathcal{B}$. Here b records the buyer associated with currently highest bid and B denotes the set of registered buyers. The condition $b \in B \cup \{\perp\}$ means that either no buyer agent has submitted a bid in the current negotiation ($b =\perp$) or the buyer agent that submitted the currently highest bid must have already registered with the negotiation before the submission, i.e. $b \in B$. Note that when the server detects a certain period of bidding inactivity she will trigger negotiation termination and agreement formation – action *stop*.
- $AnswerBid(b_1, b_2, B)$ such that $b_1 \in B, b_2 \in B \cup \{\perp\}, B \subseteq \mathcal{B}$. Here b_1 denotes the buyer that submitted the most recent bid, b_2 denotes the buyer associated with currently highest bid and B denotes the set of registered buyers. The fact that $b_1 \in B$ means that the most recently submitted bid comes from a registered buyer. The fact that $b_2 \in B \cup \{\perp\}$ means that either the currently highest bid has not been submitted yet ($b_2 =\perp$) or it was submitted by a registered buyer ($b_2 \in B$).
- $AnswerReg(b_1, b_2, B)$ such that $b_1 \in \mathcal{B} \setminus B, b_2 \in B \cup \{\perp\}, B \subseteq \mathcal{B}$. Here b_1 denotes the buyer that requested registration with the current negotiation, b_2 denotes the buyer associated with currently highest bid and B denotes the set of registered buyers. The fact that $b_1 \in \mathcal{B} \setminus B$ means that the registration request comes from a buyer that is not yet registered with the negotiation. The fact that $b_2 \in B \cup \{\perp\}$ means that either the currently highest bid has not been submitted yet ($b_2 =\perp$) or it was submitted by a registered buyer ($b_2 \in B$).
- $InformBuyers(b, B)$ such that $b \in B, B \subseteq \mathcal{B}$. Here b denotes the buyer that submitted an accepted bid and B denotes the set of registered buyers. The fact that $b \in B$ means that the bid that was accepted comes from a buyer that has registered with the negotiation. This process is responsible with notifying registered buyers that a new bid has been accepted by the host – action $inform(b)$. Note that in the FSP model of the negotiation host shown in table 1, b_i, $1 \leq i \leq k$ are defined such that $Bs \setminus \{b\} = \{b_1, b_2, \ldots, b_k\}$.
- $ServerAgreement(b)$ such that $b \in \mathcal{B} \cup \{\perp\}$. Here b denotes the buyer that submitted the currently highest bid. Process $ServerAgreement(b)$ is executed when $Server$ detected that negotiation can be terminated because of bidding inactivity. This process checks if an agreement can be made. If there is no buyer that submitted an accepted bid, i.e. $b =\perp$ then there is no agreement – action no_win. Otherwise either there is an agreement –

Table 2: $Buyer$ and $Seller$ processes.

$$
\begin{aligned}
Buyer &= register \rightarrow BuyerRegister| \\
&\quad\; inform \rightarrow Buyer, \\
BuyerRegister &= accept\_registration \rightarrow BuyerBid| \\
&\quad\; reject\_registration \rightarrow Buyer, \\
BuyerBid &= bid \rightarrow WaitBid| \\
&\quad\; cancel\_bid \rightarrow Buyer| \\
&\quad\; inform \rightarrow BuyerBid, \\
WaitBid &= accept\_bid \rightarrow Wait| \\
&\quad\; reject\_bid \rightarrow BuyerBid| \\
&\quad\; inform \rightarrow BuyerBid, \\
Wait &= inform \rightarrow BuyerBid| \\
&\quad\; end \rightarrow Buyer.
\end{aligned}
$$

$$
\begin{aligned}
Seller &= init \rightarrow WaitInit, \\
WaitInit &= accept\_init \rightarrow WaitEnd| \\
&\quad\; reject\_init \rightarrow Seller, \\
WaitEnd &= end \rightarrow Seller.
\end{aligned}
$$

action $win(b)$ or the currently highest bid does not meet the requirements for generating an agreement – action $no\_win(b)$.

4.3 *Buyer* Role

The $Buyer$ role is defined as a cyclic FSP process. Note that a buyer agent must first register to the negotiation before starting to submit bids. If registration is granted, she can start bidding according to its private strategy – action bid. Here we have chosen a very simple strategy: each buyer agent submits a first bid immediately after it is granted admission to the negotiation and subsequently, whenever it gets a notification that another participant issued a bid that was accepted by the host. Additionally, each buyer participant has its own valuation of the negotiated product. If the current value that the buyer decided to bid exceeds her private valuation then the proposal submission is canceled – action $cancel\_bid$, i.e. product became “too expensive”. Note that after a buyer agent submitted a bid that was accepted, she will enter a state waiting for a notification that either another successful bid was submitted or that she eventually was the last submitter of a successful bid in the current auction (i.e. a potentially winning bid, depending on if the bid value was higher than the seller reservation price) – see action end. Note that execution of the end action also means that the negotiation was terminated by the server.

Table 3: $System$ process as parallel composition of negotiation host, buyers and seller processes.

$$
\begin{aligned}
BuyerAgent_1 = {} & Buyer/\{bid_1/bid, reject\_bid_1/reject\_bid, accept\_bid_1/accept\_bid,\\
& inform_1/inform, cancel\_bid_1/cancel\_bid, \{win_1, no\_win_1\}/end,\\
& register_1/register, accept\_registration_1/accept\_registration,\\
& reject\_registration_1/reject\_registration\}.\\
BuyerAgent_2 = {} & Buyer/\{bid_2/bid, reject\_bid_2/reject\_bid, accept\_bid_2/accept\_bid,\\
& inform2/inform, cancel\_bid_2/cancel\_bid, \{win_2, no\_win_2\}/end,\\
& register_2/register, accept\_registration_2/accept\_registration,\\
& reject\_registration_2/reject\_registration\}.\\
SellerAgent \;\; = {} & Seller/\{\{no\_win, win_1, no\_win_1, win_2, no\_win_2\}/end\}.\\
System \;\; = {} & Server||SellerAgent||BuyerAgent_1||BuyerAgent_2.
\end{aligned}
$$

4.4 *Seller* Role

The $Seller$ role is also defined as a cyclic FSP process. The seller agent initiates the auction – action $init$ and then, assuming initiation was successful, she waits for the auction to terminate – action end, before issuing a new initiation request.

4.5 Negotiation System

Let us assume that our system is initialized by creating 2 buyer agents, i.e. $\mathcal{B} = \{b_1, b_2\}$, and one seller agent. Buyer and seller agents are created by instantiating $Buyer$ and respectively $Seller$ roles. Note that instantiation of $Buyer$ roles assumes also indexing of actions bid, $reject\_bid$, $accept\_bid$, $inform$, $cancel\_bid$, $register$, $accept\_registration$, $reject\_registration$ with buyer's name and also renaming action end with an indexed set of actions $\{win, no\_win\}$. Similarly, instantiation of $Seller$ role assumes renaming action end with a set of actions denoting various ways the auction may terminate: without a winner assuming no buyer submitted an accepted bid – no_win, with or without a winner assuming at least one buyer submitted an accepted bid – indexed set of actions $\{win, no\_win\}$. Finally, instantiation of $Server$ role requires no renaming, as the names of the buyer agents were supposed known in the definition of $Server$ process.

Negotiation system is defined as parallel composition of negotiation host, seller agent and buyer agents processes – see table 3.

5 Experiments

We have conducted a series of experiments with the FSP models introduced in section 4. The main goal was to check various qualitative properties of agent interactions in the negotiation system. As a side effect we have also recorded the size of the model expressed as number of states and transitions, depending on the number of negotiation participants.

5.1 Experimental Setup

Firstly we had to express the general models shown in tables 1, 2 and 3 using the FSP language supported by the LTSA tool.

An initial mapping was proposed in [Bădică 2007b]. However it has the drawback that the mapping of the *Server* process is not scalable with respect to the number of buyers. This problem is caused by the way we had chosen to map local processes indexed with subsets. A subset index was mapped to a sequence of integer indexes representing the subset elements; for example subset $\{1, 3\}$ was represented as [1][3]. Using this approach the mapping would have to use at least one local process definition for subsets with 1 element, at least one local process definition for subsets with 2 elements, etc. This situation is not acceptable when the number of participants is increasing.

Therefore we propose an improved mapping inspired by the solution we have already explored in [Bădică 2007a]. Assuming that we have n buyers and that each buyer is represented by a unique integer from the set $\{1, 2, \ldots, n\}$, a subset index is mapped to a sequence of n $\{0, 1\}$-valued integer indexes; for example subset $\{1, 3\}$ is represented as [1][0][1]. $\perp$ symbol is mapped to 0. We obtain the mapping conventions:

- $ServerBid(chb, Bs)$ is mapped to $ServerBid[chb][i_1][i_2]\ldots[i_n]$ such that $chb \in \{0, 1, 2, \ldots, n\}$, $i_j = 0$ if $j \notin Bs$ and $i_j = 1$ if $j \in Bs$ for all $j \in \{1, 2, \ldots, n\}$;
- $AnswerReg(b, chb, Bs)$ is mapped to $AnswerReg[b][chb][i_1][i_2]\ldots[i_n]$ such that $b \in \{1, 2, \ldots, n\}$, $chb \in \{0, 1, 2, \ldots, n\}$, $i_j = 0$ if $j \notin Bs$ and $i_j = 1$ if $j \in Bs$ for all $j \in \{1, 2, \ldots, n\}$; $AnswerBid$ is mapped in a similar way;
- $InformBuyers(b, Bs)$ is mapped to $InformBuyers[b][i_1][i_2]\ldots[i_n]$ such that $b \in \{1, 2, \ldots, n\}$, $i_j = 0$ if $j \notin Bs$ and $i_j = 1$ if $j \in Bs$ for all $j \in \{1, 2, \ldots, n\}$.
- $ServerAgreement(chb)$ is mapped to $ServerAgreement[chb]$ such that $chb \in \{0, 1, 2, \ldots, n\}$. Here

Following these conventions, for example if $n = 3$ then local process $AnswerBid(1, 3, \{1, 3\})$ becomes $AnswerBid[1][3][1][0][1]$. The complete mapping of our FSP model of the English auction system for $n = 3$ buyers is shown below.

```
const N = 3

Buyer =
  (register -> BuyerRegister | inform -> Buyer),
BuyerRegister =
  (accept_registration -> BuyerBid | reject_registration -> Buyer),
BuyerBid =
  (bid -> WaitBid | cancel_bid -> Buyer | inform -> BuyerBid),
WaitBid =
  (accept_bid -> Wait | reject_bid -> BuyerBid | inform -> BuyerBid),
Wait =
  (inform -> BuyerBid | end -> Buyer).

Seller =
  (init -> WaitInit),
WaitInit =
  (accept_init -> WaitEnd | reject_init -> Seller),
WaitEnd =
  (end -> Seller).

Server =
  (init -> AnswerInit),
AnswerInit =
  (accept_init -> ServerBid[0][0][0][0] | reject_init -> Server),
ServerBid[chb:0..N][i1:0..1][i2:0..1][i3:0..1] = (
  when i1 == 1 bid[1] -> AnswerBid[1][chb][i1][i2][i3] |
  when i1 == 0 register[1] -> AnswerReg[1][chb][i1][i2][i3] |
  when i2 == 1 bid[2] -> AnswerBid[2][chb][i1][i2][i3] |
  when i2 == 0 register[2] -> AnswerReg[2][chb][i1][i2][i3] |
  when i3 == 1 bid[3] -> AnswerBid[3][chb][i1][i2][i3] |
  when i3 == 0 register[3] -> AnswerReg[3][chb][i1][i2][i3] |
  stop -> ServerAgreement[chb]),
AnswerReg[b:1..N][chb:0..N][i1:0..1][i2:0..1][i3:0..1] = (
  when b == 1 accept_registration[1] -> ServerBid[chb][1][i2][i3] |
  when b == 2 accept_registration[2] -> ServerBid[chb][i1][1][i3] |
  when b == 3 accept_registration[3] -> ServerBid[chb][i1][i2][1] |
  reject_registration[b] -> ServerBid[chb][i1][i2][i3]),
AnswerBid[b:1..N][chb:0..N][i1:0..1][i2:0..1][i3:0..1] = (
  accept_bid[b] -> InformBuyers[b][i1][i2][i3] |
  reject_bid[b] -> ServerBid[chb][i1][i2][i3]),
InformBuyers[b:1..N][i1:0..1][i2:0..1][i3:0..1] =
  InformBuyers[b][1][i1][i2][i3],
InformBuyers[b:1..N][i:1..N][i1:0..1][i2:0..1][i3:0..1] =
  if (i==1 && i!=b && i1==1) then (inform[1] -> InformBuyers[b][2][i1][i2][i3])
  else if (i==1 && (i==b || i1==0)) then InformBuyers[b][2][i1][i2][i3]
  else if (i==2 && i!=b && i2==1) then (inform[2] -> InformBuyers[b][3][i1][i2][i3])
  else if (i==2 && (i==b || i2==0)) then InformBuyers[b][3][i1][i2][i3]
  else if (i==3 && i!=b && i3==1) then (inform[3] -> ServerBid[b][i1][i2][i3])
  else ServerBid[b][i1][i2][i3],
ServerAgreement[0] =
  (no_win -> Server),
ServerAgreement[chb:1..N] =
  (win[chb] -> Server | no_win[chb] -> Server).

||BuyerI(I=1) =
  Buyer/{bid[I]/bid,reject_bid[I]/reject_bid,accept_bid[I]/accept_bid,inform[I]/inform,
         cancel_bid[I]/cancel_bid,{win[I],no_win[I]}/end,register[I]/register,
         accept_registration[I]/accept_registration,
         reject_registration[I]/reject_registration}.

||System = (
  Server || Seller/{{win[b:1..N],no_win[1..N], no_win}/end} ||
  (forall [i:1..N] BuyerI(i))).
```

There are few interesting notes about this FSP model:

– The model is parameterized with the number of buyers – constant N;

– $Buyer$ and $Seller$ processes are identical to our previous solution from [Bădică 2007b]; only the mapping of $Server$ process has been updated;

– Buyers are now defined as an indexed family of processes $BuyerI(I)$, rather than separately (as in [Bădică 2007b]). This is further exploited by the definition of the $System$ process using $forall$ construct.

5.2 Properties of the Negotiation System

Techniques discussed in [Karamanolis et al. 2000] for workflow analysis can be also applied to analyze our negotiation system. These techniques include interactive step-by-step simulation and model verification against safety and liveness properties.

Interactive simulation allows us to perform a manually controlled step-by-step execution of the negotiation system. While this feature may give a "feeling" about how the system would behave before actually being implemented, it is quite limited for large applications. Additionally, the trace facility can only eventually detect abnormal behaviors, failing to prove that the system behaves correctly for all its possible executions.

LTSA tool supports a more powerful way of checking a target system using *safety and progress properties* ([Magee and Kramer 2006]).

A *safety property* is defined as a deterministic process P asserting that any of the system traces in the alphabet of P is correct, i.e. are accepted by P. If an error state is reachable in the LTS of its composition with the target system then the safety property is violated; additionally, LTSA provides one execution trace that violates the property. This is useful for the modeler to correct the model.

In what follows we consider some safety properties that we found useful to check for our negotiation system.

The negotiation system is free of deadlocks – being deadlock free is a basic property of a distributed system that is automatically checked by the LTSA tool.

When a new participant requests admission to negotiation, negotiation host either grants or not participant admission, before the participant can request again admission to negotiation.

$$\begin{array}{ll} \textbf{property}\ Admission = & register(b) \rightarrow CheckAdmission(b), \\ CheckAdmission(b) = & accept\_registration(b) \rightarrow Admission | \\ & reject\_registration(b) \rightarrow Admission. \end{array}$$

A registered participant may submit bids, then cancel bidding, before she can re-enter negotiation and start bidding again.

$$\begin{array}{ll} \textbf{property}\ Bid(b) = & accept\_registration(b) \rightarrow ContinueBidding(b), \\ ContinueBidding(b) = & bid(b) \rightarrow ContinueBidding(b) | \\ & cancel\_bid(b) \rightarrow Bid(b). \end{array}$$

Table 4: Size of the LTS of the negotiation system

# buyers	# states	# transitions
2	66	158
3	370	1053
4	2160	7056
5	12400	45712
6	68992	283808
7	372032	1690416

When a participant submits a new bid, the bid is either accepted or rejected by the negotiation host before the participant can submit a new bid.

$$\begin{array}{ll} \textbf{property}\ Proposal = & bid(b) \rightarrow CheckBid(b), \\ CheckBid(b) = & accept\_bid(b) \rightarrow Proposal | \\ & reject\_bid(b) \rightarrow Proposal. \end{array}$$

Whenever a negotiation is successfully started it must also safely terminate with or without a winner, before a new negotiation can be started.

$$\begin{array}{ll} \textbf{property}\ SafeTermination = & accept\_init \rightarrow \\ & \{win(b \in \mathcal{B}), no\_win(b \in \mathcal{B}), no\_win\} \rightarrow \\ & SafeTermination. \end{array}$$

A *progress property* is defined as a finite set of actions and it requires that any infinite execution of the target system contains at least one of the actions in this set infinitely often.

We have checked our negotiation system against the *default progress property* that asserts each action in the process alphabet will be executed infinitely often ([Magee and Kramer 2006])[1].

We have also determined the LTS of a negotiation system with $n = 2, 3, \ldots, 7$ buyers using LTSA tool ([Magee and Kramer 2006]). The results are summarized in table 4.

6 Conclusions and Future Work

In this paper we proposed a formal framework for modeling agent interactions in agent-based English auctions using finite state process algebra. This works bridges the gap between the sound analysis of agent interactions using formal methods and the implementation of agent-based English auctions using available

[1] Note that progress properties are checked under the *fair choice assumption* ([Magee and Kramer 2006])

agent platforms. As future work we plan to extend this approach by capturing also the semantic aspects of the agent interaction protocols into a unified framework. Furthermore we intend to apply this framework to model more complex agent systems including: i) negotiation servers that are able to coordinate several negotiations that are ran in parallel; ii) agent systems that incorporate also middle-agents for carrying out for example matchmaking processes that are frequently needed in e-commerce.

Acknowledgement

Work of Amelia Bădică and Costin Bădică has been partially funded by the following two CNCSIS[2] grants:

1. 94/2005: "HiperProc: Hypermedia Techniques for Knowledge-Based Representation of Business Processes" and
2. 185/2006: "Technologies and Intelligent Software Tools for Automated Construction of E-Catalogues of Products Using Knowledge Acquisition from the Web".

References

[Bădică 2007a] Bădică, C., Bădiţă, A., Liţoiu, V.: "Middle-Agents Interactions As Finite State Processes: Overview and Example"; Proceedings of 16^{th} IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises, WETICE'2007, 2007, Paris, France. IEEE Computer Society Press (2007) 12-17.

[Bădică 2007b] Bădică, C., Bădiţă, A.: "Formal modeling of agent-based English auctions using finite state process algebra"; Agent and Multi-Agent Systems: Technologies and Applications. Proceedings of KES-AMSTA-2007, Wroclaw, Poland. Lect. Notes in Artif. Intel. 4496, Springer Verlag (April 2007) 248-257.

[Bădică 2007c] Bădică, C., Ganzha, M., Paprzycki, M.: "Implementing Rule-Based Automated Price Negotiation in an Agent System"; Journal of Universal Computer Science, vol. 13, no. 2, (2007) 244-266.

[Bădică et al. 2006b] Bădică, C., Bădiţă, A., Ganzha, M., Iordache, A., Paprzycki, M: "Implementing rule-based mechanisms for agent-based price negotiations"; Proceedings of the 21^{st} Annual ACM Symposium on Applied Computing, SAC, Dijon, France. ACM Press, New York, NY, (April 2006) 96-100.

[Bădică et al. 2005d] Bădică, C., Bădiţă, A., Ganzha, M., Iordache, A., Paprzycki, M.: "Rule-Based Framework for Automated Negotiation: Initial Implementation"; A. Adi, S. Stoutenburg, S. Tabet (eds.): Proc. RuleML, Galway, Ireland. Lect. Notes in Comp. Sci. 3791, Springer Verlag (November 2005) 193-198.

[Bădică et al. 2003] Bădică, C., Bădică, A., Liţoiu, V.: "Role Activity Diagrams as Finite State Processes"; Proceedings of the 2^{nd} International Symposium on Parallel Distributed Computing ISPDC'03, Ljubljana, Slovenia. IEEE Computer Society Press, (October 2003) 15-22.

[2] `http://www.cncsis.ro`

[Bartolini et al. 2005] Bartolini, C., Preist, C., Jennings, N.R.: "A Software Framework for Automated Negotiation"; Proc. of SELMAS. Lect. Notes in Comp. Sci. 3390, Springer, Berlin (2005) 213-235.

[Cervenka and Trencansky 2007] Cervenka, R., Trencansky, I.: "The Agent Modeling Language – AML. A Comprehensive Approach to Modeling Multi-Agent Systems Series: Whitestein"; Series in Software Agent Technologies and Autonomic Computing, Springer, (2007).

[Dong and Sheng 2003] Dong, Y., Sheng, Z.: "Using pi-Calculus to Formalize UML Activity Diagram"; Proceedings of the 10^{th} IEEE International Conference on Engineering of Computer-Based Systems ECBS 2003. IEEE Computer Society (2003) 47-54.

[van Eijk et al. 2003] van Eijk, R.M., de Boer, F.S., van der Hoek, W., Meyer, J.-J.Ch.: "Process Algebra for Agent Communication: A General Semantic Approach"; Communication in Multiagent Systems 2003. Lect. Notes in Comp. Sci. 1650, Springer Verlag (2003) 113-128.

[Esterline and Rorie 2001] Esterline, A.C., Rorie, T.: "Using the pi-Calculus to Model Multiagent Systems"; Formal Approaches to Agent-Based Systems, First International Workshop, FAABS'2000, Greenbelt, MD, USA. Lect. Notes in Comp. Sci. 1871, Springer Verlag (2001) 164-179.

[Feng et al. 2005] Feng, Z., Yin, J., Zhang, H., Dong, J.: "Inter-organizational business process modeling for electronic commerce based on pi-calculus"; Proceedings of the Intternational Conference on Services Systems and Services Management, ICSSSM'2005, Chongqing, China. IEEE Press vol.2 (2005) 966-970.

[FIPA] FIPA: Foundation for Physical Agents. See `http://www.fipa.org`.

[Hillston and Kloul 2001] Hillston, J., Kloul, L.: "Performance investigation of an online auction system"; Concurrency and Computation: Practice and Experience, 13(1) (2001) 23-41.

[Karamanolis et al. 2000] Karamanolis, C., Giannakapoulou, D., Magee, J., Wheater, S.: "Model Checking of Workflow Schemas"; Proceedings of the 4th International conference on Enterprise Distributed Object Computing, EDOC'00. IEEE Computer Society (2000), 170-181.

[JADE] JADE: Java Agent Development Framework. See `http://jade.cselt.it`.

[JESS] JESS: Java Expert System Shell. See `http://herzberg.ca.sandia.gov/jess/`.

[Laudon and Traver 2004] Laudon, K.C., Traver, C.G.: "E-commerce. business. technology. society" (2^{nd} ed.). Pearson Addison-Wesley, (2004).

[Lomuscio et al. 2002] Lomuscio, A.R., Wooldridge, M., Jennings, N.R.: "A classification scheme for negotiation in electronic commerce"; F. Dignum, C. Sierra (Eds.): Agent Mediated Electronic Commerce: The European AgentLink Perspective, Lect. Notes in Comp. Sci. 1991, Springer, Berlin (2002) 19-33.

[Magee and Kramer 2006] Magee, J., Kramer, J.: "Concurrency. State Models and Java Programs 2^{nd} ed."; John Wiley & Sons, (2006).

[McAfee and McMillan 1987] McAfee, R.P., McMillan, J.: "Auctions and bidding"; Journal of Economic Literature, 25(2) (1987) 699-738.

[Padgham and Winikoff 2004] Padgham, L., Winikoff, M.: "Developing Intelligent Agent Systems. A practical guide"; John Wiley & Sons (2004).

[Puhlmann 2006] Puhlmann, F.: "Why Do We Actually Need the Pi-Calculus for Business Process Management"; Proceedings of the 9^{th} International Conference on Business Information Systems, BIS'2006, Klagenfurt, Austria. Lecture Notes in Informatics 85, GI (2006) 77-89.

[Rouff et al., 2006] Rouff, C.A., Hinchey, M., Rash, J., Truszkowski, W., and Gordon-Spears, D. (Eds.): "Agent Technology from a Formal Perspective"; NASA Monographs in Systems and Software Engineering, Springer (2006).

[Weiliang et al. 2002] Weiliang, M., Xiaodong, W., Huanye, S.: "A Configurable Auction Framework for Open Agent Systems[[; Joint Workshop Agent Technology and Software Engineering/Agent Infrastructure, Tools and Applications, Net.ObjectDays 2002, Erfurt, Germany, (2002).

[Wooldridge 2002] Wooldridge, M.: "An Introduction to MultiAgent Systems", John Wiley & Sons, (2002).

[Wurman et al. 2001] Wurman, P.R., Wellman, M.P., Walsh, W.E.: "A Parameterization of the Auction Design Space"; Games and Economic Behavior, 35, 1/2, (2001) 271-303.