Intelligent Information Management, 2009, 1, 65-72
doi:10.4236/iim.2009.12011 Published Online November 2009 (http://www.scirp.org/journal/iim)

The Specification of Agent Interaction in Multi-Agent Systems

Dmitri CHEREMISINOV
United Institute of Informatics Problems, National Academy of Sciences of Belarus, Minsk, Belarus
Email: *cher@newman.bas-net.by*

Abstract: The problem of the description of interaction between agents in a multi-agent system (MAS) in the form of dialogues of negotiations is considered. For formalization of the description of interaction at a level of steps of the dialogue which is carried out in common by two spatially divided agents, the concept of synchronization of processes is analyzed. The approach to formalization of the description of conditions of synchronization when both the independent behaviour, and the communications of agents can be presented at a level of logic is offered. It is shown, that the collective behavior of agents can be described by the synthetic temporal logic that unite linear and branching time temporal logics.

Keywords: interaction protocol, temporal logic, parallel algorithm

1. Introduction

A multi-agent system [1] consists of set of agents that operate together in order to achieve some goals. Such system can be considered as the organization of agents (by analogy to the human organization) or, in other words, as some artificial society. Protocols play the central role in the organization of this society. The protocol of interaction can be stipulated by the designer of investigated system, or the agents who are taking part in information interchange agree about the application of a protocol before interaction took place.

The interaction protocol of agents defines the rules that govern the dialogue between agents in multi-agent system. The central problem of the interactions that took place in open systems and not being cooperative (dialogues of negotiations) is the problem of conformity check between agent behavior and interaction protocol. The implementation of conformity check encounters the trouble what is the identification of steps of dialogue. Recognition of the step which is carried out by two spatially divided agents jointly, require analyzing the concept of interaction of processes.

The basis of formal models of protocols is cooperating sequential processes. Fundamental features these models differ are the degree of synchronization of behaviour of participants of interaction. This paper is the attempt to analyze by logic the concept of synchronization in these frameworks.

2. Formalization of Concept of Interaction Event

Usually collective behaviour of system of agents is described as dialogue of agents which communicate by means of sending and receiving messages. On each step of activity the agent carries out some action depending on the internal state and the received message. As a result the action changes the internal state and sends messages to other agents. The collaboration of agents emphasise autonomy and cooperation (with other agents) in order to perform tasks for their owners. Collaboration behaviour of system of agents is unreachable when the agent doesn't know anything about individual behaviour of other agents. Particularly because the agents are autonomous and cannot be assumed to be benevolent, agents must know that others must do to act in certain ways without requiring that one examine the internal reasoning (or the source code) of the agents. Speaking informally, the protocol is formal representation of knowledge of the agent about individual behaviour of other agents into the scope of joint task. There is still a need for a proper formalism for protocols that is suitable for automated implementation. Suppose we are given a protocol specification, *PS*. One way of obtaining a concrete agent program from *PS* is to treat it as an *executable specification,* and *interpret* the specification directly in order to generate the agent's behavior. If models for the specification language can be given a computational interpretation, then model building can be viewed as executing the specification (Figure 1).

This knowledge is inexpedient to treat as parameter of an agent program from speed consideration. This knowledge must be realized by programmer as the agent program. In this scheme, we take our abstract representation of the knowledge, and transform it into a concrete program via some synthesis process.

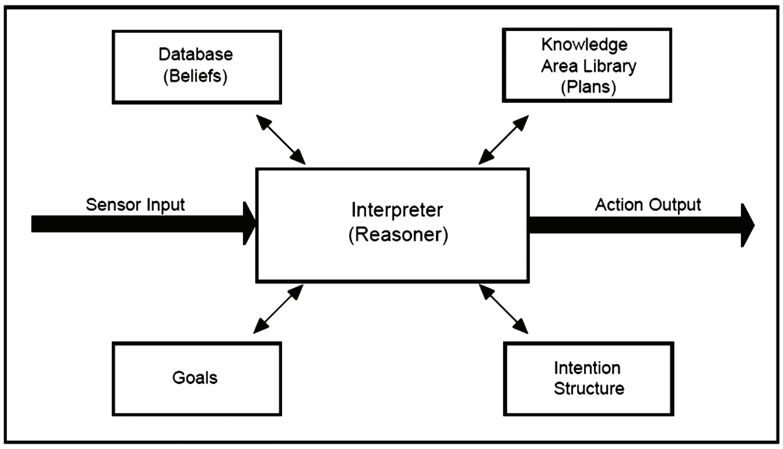

Figure 1. Parameters of universal agent

For the specification of agent independent behaviour are widely used formalisms of high level abstractness, for example, such as temporal logic. At the same time the communications between agents is specified by means of concepts of a realization level, such as mail boxes and messages. One of problems of such segregated approach to interactions consists that it is extremely difficult to model the interactions between agents though the independent behaviour of the agent is described completely. This problem arises due to absence of agent model unifying aspects of independent behaviour and the communications. The main reason of absence of the general model is the lack of conceptual basis unifying all abstraction, connected with collective behaviour of agents.

The independent behaviour of multi-agent system agents are characterized by processes. Process is defined by the full description of potential behaviour of the agent. The behaviour of process consists of events. Thus, suitable axiomatization of concept of event is required for the analysis of concept of interaction of processes.

The event serves as the concept to abstract from physical time at the description of behaviour of system. Widely widespread axiomatization of event is connected with the assumption, that events have no duration [2]. The behaviour multi-agent system consists of events – steps of dialogue between agents – and is consecutive in this sense. For recognition of the step which is carried out by two spatially divided agents jointly, it is impossible to bypass the concept of parallelism.

The models of parallelism known in the literature is possible roughly divided into two classes: 1) models, in which concurrent execution of two processes described by interleaving of (atomic) events of those processes; 2) models in which causal dependencies between events are set explicitly. Interleaving models are focused on systems with events are considered instant and indivisible. In this case the act of interaction is complete event which describe participation of all processes cooperating in this act [2]. This act as the step of dialogue are carried by two spatially divided agents represent event which should have duration and structure.

There is popular opinion the concept of the event having duration is reduced to concept of instant event. The following formulation of these assumptions is taken from Hoare [3] "The actual occurrence of each event in the life of an object should be regarded as an instantaneous or an atomic action without duration. Extended or time-consuming actions should be represented by a pair of events the first denoting its start and the second denoting its finish."

Now it is known that this opinion is erroneous, and behaviour in the events having duration is not reduced to the behaviour expressed through instant events. Mutual irreducibility the concept of the event having duration to concept of instant event is proved formally and constructively [4]. The formal proof is based on incomparability of formalism describing event systems [5]. Systems of the events having duration are described by the causal relation (branching-time temporal logic), systems of instant events – relation of consequence and parallelism (linear-time temporal logic).

3. Structure of Interaction Event

The elementary structure of durational event that is a step of dialogue is pair of durational events which constitute a step densely without a time interval between. First event can be interpreted as "pronouncing" of the message by one agent; the second event can be interpreted as "perception" of this message by other participant of dialogue. The basic feature of this structure is the assumption of density of an event composition and that constituting events belong to behaviour of different agents (Figure 2). Absence of a time interval between members of pair durational events designates instant event *of synchronization* of processes.

3.1. The Event Model of Synchronization

Abstracting from function of agents, it is possible to consider synchronization of their behaviour as the goal of interaction. Thus, the step of dialogue is a composition of *three* events, *two* events are durational events, and *one* is instant event. However constituting event of interaction still remains event which occur simultaneously in *different* processes.

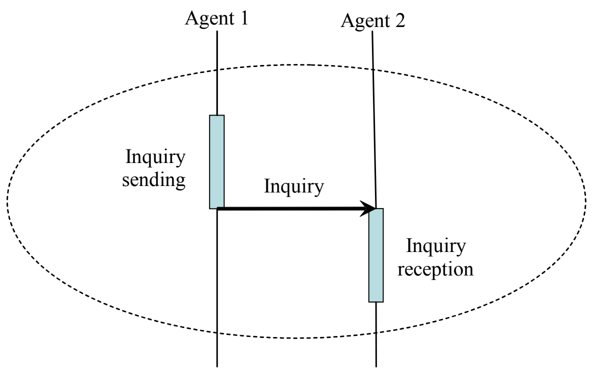

Figure 2. Structure of the communication event

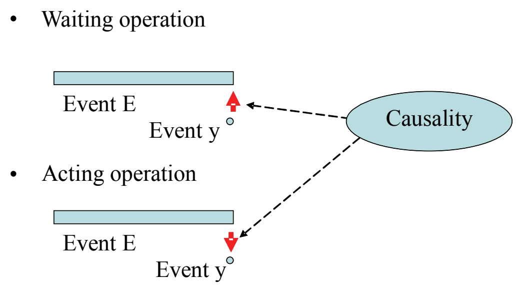

Figure 3. Structure of operations

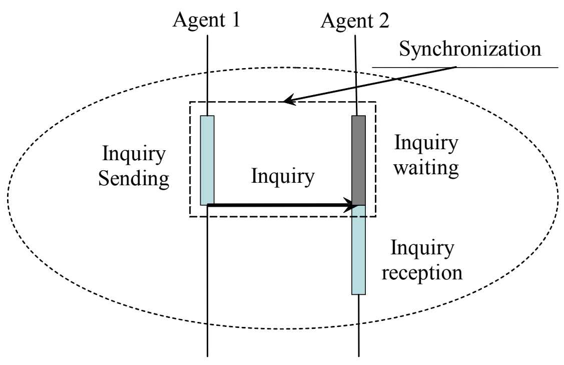

Figure 4. Synchronization of behaviour of agents

On the one hand synchronization of agent behaviour occurs during the rare moments. In the rest of the time communicating agents behave independently from each other. On the other hand, processes should interchange information about current states to ensure synchronization. Formally it can be reached by splitting of all events constituting agent behaviour on internal and external events. Only external events of the agent can be “visible” to other agents. In this case the specification of the agent behaviour is the causal relation on set of possible events, in particular, this relation describe the reasons of occurrence of external and internal events.

Let composition durational internal event E and instant external event y is *operation*. A composition $y \rightarrow E$ durational and instant events is waiting operation that wait external event y, and a composition $E \rightarrow y$ is acting operation which effect is realization of external event y (Figure 3). It is necessary to note, that order of events in both operations is the same: the first goes durational event, then goes instant event. These compositions allow to consider dependences between events of a composition as cause and effect because from physical reasons event-consequence occurs behind event-reason without overlapping on time. Waiting operation is durational event end reason of its termination is y. Acting operation is durational the event and the reason of y is its termination. The symbol “$\rightarrow$” we interpret as cause and effect dependence between events. Such treatment of waiting and acting operations is a basis of formal semantics PRALU language [6] in which conjunction of Boolean variables describe external event of operations.

PRALU language in this interpretation represents the synthetic temporal logic uniting linear and branching time temporal logics supplied by the assumption of density of time [7]. Temporal formulas of this logic are interpreted as the statement concerning order of events of two sorts: instant and durational.

The composition considered above allows to describe independent behaviour of agents. Parallel execution of waiting operation by one agent and acting operation by other agent leads to synchronization of behaviour of agents during the moment of instant event y (Figure 4). By definition the effect of waiting operation is its termination at moment of instant external event.

The line of “life” of the agent consists of pairs waiting and acting operations. The border between them serves as the synchronization event. Here the action consists from “perception” of the accepted message and “pronouncing” new message. Obviously, the occurrence of synchronization depends on duration of acting operations.

3.2. The Interaction as an Event of an Environment

The step of dialogue can be considered as the *other* composition of three events; *one of which* is durational, and others *two* are instant (Figure 5). In this model of a dialogue step the durational event is directly interaction. This event, having duration, should have a physical basis. Without loss of a generality it is possible to consider that event of interaction occurs in *an environment* of agents. In this model event of interaction becomes not distributed, but this event is *local* in an environment.

4. Concept of an Environment

From consideration of physical realizations of the distributed systems follows that the synchronization achieve by the special organization of system of cooperating agents. Two general types of the system organization for achievement of synchronization are known: *synchronous* and *asynchronous* systems. The standard definition of distinction of these types of systems consists that synchronous systems have shared "clock", and it is said to be *asynchronous* if each agent has its own independent clock. It is obvious, that these shared "clock" are an accessory of an environment of agents.

In traditional interaction theories CCS [3] or CSP [8] concept of an environment is used implicitly, hence it is not formalized. CCS and CSP use on concept of an environment which consists in the following. The environment is considered simply other agent. Other words, the environment for the given agent includes all other agents of the system operating in parallel with this agent. In this case the agent and an environment are objects of identical nature. It is obvious, that this assumption of properties of an environment is not well from the point of view to specify the interaction for achievement of synchronization. Such approach is justified by following. It is considered that the concept of an environment concern with realization of system and is not representable at a level of behaviour.

Our purpose consists in offering formal model of interaction which is not concern with realization of system. We consider an environment essentially distinct from agents. The basis of this approach is the representation about interaction as the communication act consisting from

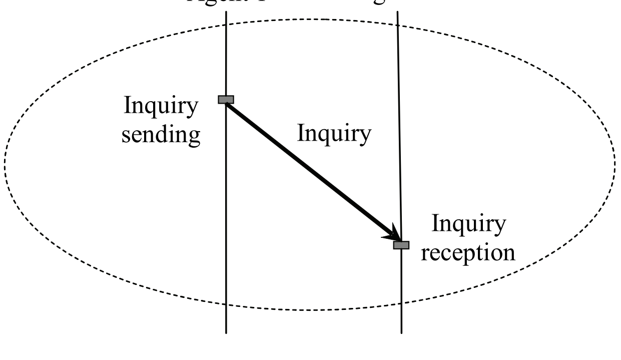

Figure 5. Synchronization event as environment event

sending and receiving of the messages. This formalization of interaction originates from Shannon's paper about the theory of communication [9] in which interaction is considered as a way of transfer of the message from sender to a receiver via a medium also called transfer environment. Physical realization of an environment can be the computer program, the device or the physical environment.

Obviously, synchronization of agent behaviour is impossible without fixing data which are transferred during their interaction [10] by an environment. Thus, environment serves as model of transport system for delivery of messages. Other words environment represents the memory that is shared of all agents. This memory is known as a global state of multi-agent system. In its most simple form, the communications can be based on the fixed set of differing signals. In case of binary signals the representation of a global state is the set of the boolean variables which values are possible signals. In case of structural signals the environment of agents usually refers *as message passing system*. The concept of an environment is closely concern with *autonomy* of agents. Autonomy has its focus on freely choosing between actions and on acting independently. Autonomy means that the agents receive the information only through an environment.

System, in which the behaviour of an environment is deterministic, refers to *closed system*. In case of closed system it is supposed, that the reasons of all events are inside of system and its behaviour is self controlled completely. If the behaviour of an environment is nondeterministic, system refers to *opened system*. Unlike the memory considered in the theory of automatic devices, the behaviour of memory of an environment of the open system can depend on uncontrollable conditions.

The specification of interaction in the form of the description of message passing system does the description of autonomous behaviour of the separate agent not closed because this description is not enough for understanding of the complete behaviour of the agent. Obviously, most important property of the message passing system is restriction on length of durational events, imposed by this system.

5. Time as Logic Concept

In the previous part of this paper the time was considered as the logic concept expressed by relations between events through sequence and order of events. Time is discrete, because there is an observable time quantization by the events that fixed in behaviour of an environment. If we do not impose restrictions on duration of events in all components of multi-agent system, the time is not measured. The *measured* time means, that each event in history of system behaviour is accompanied with number that express either duration of event, or specifying the moment of time when it occur. Synchronization of behaviour of agents means the measured time.

Measured time can be realized, if we assume, that duration of all simultaneously executed acting operations in a multi-agent system is identical. This duration is natural for accepting for the unit of time. In this case in the closed systems duration of waiting operations are expressed by an integer $i \geq 1$. The assumption that duration of all simultaneously executed acting operations is identical holds in synchronous system. Obviously, this assumption defines the pairs of interacting waiting and acting operations by number of a step of time. Synchronous system keeps the assumption that time is discrete, dense and measured.

Other assumption to realize measured time is that any operation was carried out in parallel to itself is illegal. In this case realization of any operation in history of functioning of the agent is accompanied with counter number of this realization. The formal proof of this statement is in [7]. The function which calculate a counter number of operation realization (from the start of system) when this operation starts can be used for measurement of time. Interaction occurs only in pairs of waiting and acting operations which have the same count number. This is known as a *rendezvous* condition. Asynchronous system keeps the assumption that time is discrete and measured, but removes the assumption that duration of all simultaneously executed acting operations is identical. They measuring time principle is differ from synchronous type system.

6. Programming of Agents in the PRALU Language

The central problem in a multiagent system is that of coordinating the work of the agents, a process that may be understood as that of assuring that the sequences of actions performed by the agents is consistent and coordinated.

The specification of a conversation in a multiagent system is determined by two components, messages and protocols. The former are described in communication languages, such as Knowledge Query and Manipulation Language (KQML) [11], Foundation for Intelligent Physical Agents (FIPA) ACL [12], and others. A dictionary and messages which may be exchanged by agents are described in the communication languages.

A protocol is a set of interaction rules that serve to coordinate the work of several agents. We consider protocol as common knowledge by which agents achieve some goals. Formal models of protocols have been studied within the context of the theory of distributed computations. The fundamental feature that serves to distinguish these models is the degree of synchronization of behavior of the participants in an interaction. If we abstract from the function of the agents, the objective of interaction turns out to be synchronization of the behavior of the interacting agents, since excessive synchronization entirely undermines the feasibility of joint operation of the agents. Achievement of synchronization requires specialized organization of interacting processes.

The sequences consisting of action and waiting operations are linear algorithms in PRALU. For instance, the following expression means: wait for p and execute A, execute B, then wait for q and execute C: "$-p\to A\to B$ $-q\to C$".

In general, a PRALU algorithm can be presented as an unordered set of chains α_j in the form $\mu_j{:}{-}p_jL_j\to\nu_j$, where L_j is a linear algorithm, μ_j and ν_j denote the initial and the terminal chain labels represented by some subsets of integers from the set $M=\{1,2,\ldots,m\}$: $\mu_j, \nu_j \subset M$ and the expression "$\to\nu_i$" presents the transition operation: to the chains with labels from ν_j.

Chains can be fulfilled both serially and in parallel. The order in which they should be fulfilled is determined by the variable starting set $N_t \subseteq M$ (its initial value $N_0 = \{1\}$ as a rule): a chain $\alpha_j = \mu_j{:}{-}p_jL_j\to\nu_j$ (that was passive) is activated if $\mu_j \subseteq N_t$ and $p_j = 1$. After executing the operations of the algorithm L_j, N_t gets a new value $N_{t+1} = (N_t \setminus \mu_j) \cup \nu_j$.

6.1. Representing Agent Interaction Protocols in PRALU

For an example of an interaction protocol, consider an English auction [12]. The auctioneer seeks to find the market price of a good by initially proposing a price below that of the supposed market value and then gradually raising the price. Each time the price is announced, the auctioneer waits to see if any buyers will signal their willingness to pay the proposed price. As soon as one buyer indicates that it will accept the price, the auctioneer issues a new call for bids with an incremented price and continues until no buyers are prepared to pay the proposed price. If the last price that was accepted by a buyer exceeds the auctioneer's (privately known) reservation price, the good is sold to that buyer for the agreed price. If the last accepted price is less than the reservation price, the good is not sold.

In the case of the auction there are participants of two types: the Auctioneer and Buyers. So, we have two kinds of interaction protocols – those of Auctioneer and of Buyers. The last participants are peer and should be described with identical interaction protocols.

Interaction protocol as a whole can be represented in PRALU as three complex acting operations – blocks. Each block has some sets of inputs and outputs that are enumerated in brackets following the block name (the other variables of a block are its internal). Initialization of a complex acting operation is depicted by the fragment such as "→∗Buyer". The operation Buyer exists in as many copies as the number of participants of the auction, so the copies of the operation differ in their indexes only.

The modeling of the process of auction begins with the execution of “Main_process” triggering event that initiates the interaction protocol execution. Here the processes Auctioneer and Buyer$_n$s are executed concurrently. For the sake of simplicity we limit the number of buyers to two. The process Auctioneer starts with sending the first message (start_auction) that is waited by others participants to continue communication. Below PRALU description of the auction interaction protocol is shown.

```
  Main_process ()
  1: → 2.3.4
  2: →*Auctioneer →5
  3: →*Buyer$_1$ →6
  4: →*Buyer$_2$ →7
  5.6.7: →.
  Buyer$_n$ (start_auction, price_proposed, end_auction /
accept_price$_n$, not_understand)
  1: – start_auction →2
  2: – price_proposed →*Decide(/        decision_accept,
decision_reject) →3
       – end_auction →.
  3: –decision_accept → accept_price$_n$ →4
       – decision_reject →′accept_price$_n$ →4
       – not_understand →4
  4: – timeout →2
  Auctioneer (accept_price$_1$, accept_price$_2$, not_under-
stand / start_auction, price_proposed, end_auction)
  1: → start_auction →2
  2:→′accept_price$_1$.′accept_price$_1$.′not_understand
     →* Price_propose(/price_proposed) →3
  3:– not_understand →2
       – accept_ price$_1$ →4
       – accept_ price$_2$ →4
       –′not_understand.′accept_price$_1$.′accept_price$_2$
→ end_auction → *Is_reservation_price_exceeded
( /  is_exceeded)→6
  4: →′price_proposed.′win$_1$.′win$_2$ →5
  5: – accept_ price$_1$ →win1 →2
       – accept_ price$_2$ →win2 →2
  6: – is_exceeded →good_sold →7
       – ′is_exceeded →′good_sold →7
  7: →.
```

It is assumed that all unformalized operations are referred to as acting operations that set values of logical variables assigned to them. For example, Buyer’s operation “Decide” decides for accepting or rejecting the announced price. Depending on adopted decision, it outputs true value of logical variable “decision_accept” or “decision_reject”. In a similar, Auctioneers operation “Price_propose” proposes an initial price or increments the charged price outputting true value of logical variable “price_proposed”; the operation “Is_reservation_price_exceeded” verifies if the price accepted by a buyer exceeds the auctioneer’s reservation price outputting true or false value of logical variable “is_exceeded”.

The operation “–timeout” (where timeout is integer number) means waiting for timeout unit times before doing something followed it. The operation “**→.**” is interpreted as the transition to an end of a process described by the block. When the processes of Auctioneer and all Buyers reach their end in the Main_process the transition to its end is executed.

Within the BDI architecture [13] agents are associated with beliefs (typically about the environment and other agents), desires or goals to achieve, and intentions or plans to act upon to achieve its desires. BDI agent consists of the sets of beliefs *B*, plans *P*, situations *E*, actions *A* and intentions *I*. When the agent notes a change in its environment, it decides, that there an event takes place representing some situation from *E*. Registration of the event consists in changing a state of the agents “mind”: a choice of some belief from *B*. According to it and the desire (defined by some plan from *P*) the agent intends to execute some intention representing some sequence of actions from *A* to achieve the goal chosen from *I*. Thus, planned actions are defined by the chosen plan from *P*. After they are carried out, the current situation of the environment will be changed.

In traditional parallel programming languages the basic concepts are data and control of data calculation. Data are represented by values of variables, and the control is specified by a set of processes which transform local memory states defining variable values. The PRALU program of a BDI agent is a set of plans defining actions by means of which the agent should reach a goal of its functioning. The plan consists of head labels, a body and tail labels. The body of the plan is a sequence of actions, by means of which the agent should reach a goal of its functioning, and conditions which the agent should check up. The head and tail labels of the plan symbolize intentions. Critical concept for the behaviour of the agent is the concept of active intention. The plan will be carried out only when all intentions from its head are active. After executing the plan all intentions from the head become inactive and intentions from the tail quite the contrary become active. The current set of active intentions always is not empty, the body of a plan and a set of tail intentions can be empty.

6.2. The Programming Methodology

We suggest the natural methodology of designing agent program. It supposes splitting the program into two parts: a synchronization block and a functional block (Figure 6). The first block coordinates plan execution of the agent program, that is, it controls the agent behavior. The synchronization block should be described in PRALU

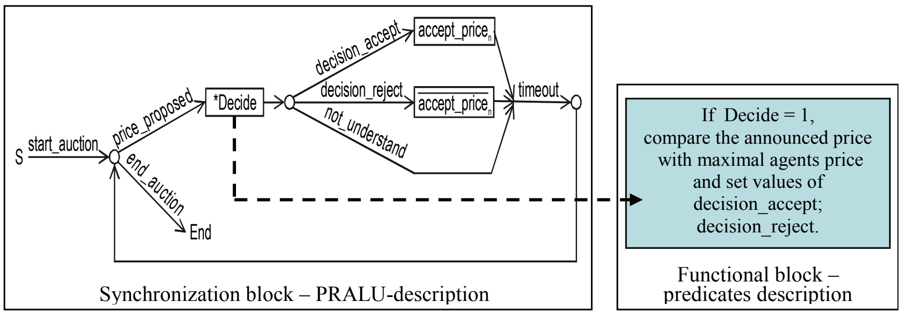
S
start_auction
price_proposed
*Decide
end_auction
End
decision_accept
decision_reject
not_understand
accept_price
accept_price
timeout
If Decide = 1, compare the announced price with maximal agents price and set values of decision_accept; decision_reject.
Synchronization block – PRALU-description
Functional block – predicates description

Figure 6. The program of the agent buyer in English auction

language. The functional block operates data and carries out calculations. This block is realized in procedural language.

The functional part is presented by predicates. The predicate is the conditions describing memory states of the agent program and the external environment or the order directing performance of some actions. The appropriate logic variable is introduced for each predicate in synchronization block.

Such an approach allows separating development of a synchronizing part of the PRALU-description from the functional one. When designing MAS, the implementation of the functional part can be delayed concentrating designer's attention on working out the synchronization block implementing interaction protocol on PRALU.

The PRALU-description compiled by PRALU compiler on the intermediate language applied in the simulation program [7] too. The program generated by the PRALU compiler consists of two interacting blocks– 1) calculation of reactions, 2) control of sensors and actuating mechanisms of the system. The heart of the control structure of the program of reactions calculation is an infinite loop; it consists of entering input signals into computer memory, calculating reactions and outputting signals to actuating mechanisms. The predicates of a functional part are considered as "the additional equipment" of MAS software in such model of the program organization. The program compiled from PRALU has semantics of measured time with of a rendezvous condition [5].

The majority of known systems for logic agent programming use model of calculations of Prolog. Prolog to find set of all decisions of a question, traverse a tree of search, tries many variants, which is not included into the decision, and comes back to earlier state in case of failure, trying other branch. This process is very expensive of space and time. Computing efficiency offered methodology is comparable to efficiency of the programs written in C.

7. Conclusions

The independent behaviour of agent in the majority of multi-agent system models is described by a formalism of high level abstractness, but the communications is specified by the concepts that close to realization. The difference of levels of the description does not allow to model the communications between agents at that level in which their independent behaviour is described. This problem arises because of absence of agent models that unify aspects of local behaviour and the communications.

In the present work we suggest to describe the synchronization conditions by specification of event properties which have been not concerned with the realization of these events. Our approach allows to specify both the independent behaviour and the communications at a level of logic. It is shown, that the collective behaviour of agents can be described by the synthetic temporal logic that unity the linear and branching time temporal logics. Such synthetic logic is an interpretation of PRALU language.

The suggested agent programming methodology ensures structuring the process of MAS designing by means of separating calculation part of MAS specification from control part. That allows the designer to concentrate on more complex stage of MAS designing – its interaction protocol. It is shown that language PRALU is very suitable for specifying interaction protocols of MAS and implementation of suggested methodology. Powerful theory and software has been developed that provides correctness verifying, simulation, hardware and software PRALU-description's implementation [6].

REFERENCES

[1] V. S. Subrahmanian, P. Bonatti, J. Dix, *et al.*, "Heterogeneous Agent Systems," MIT Press, 2000.

[2] S. D. Brookes, C. A. R. Hoare, and A. D. Roscoe, "A theory of communicating sequential processes," Journal of the ACM, Vol. 31, No. 3, pp. 560–599, 1984.

[3] S. D. Brookes, C. A. R. Hoare, and A. D. Roscoe, "A theory of communicating sequential processes," Journal of the ACM, Vol. 31, No. 3, pp. 560–599, 1984.

[4] C. A. R. Hoare, “Communicating sequential processes,” Prentice Hall International Series in Computer Science, 1985.

[5] R. Van Glabbeek and F. Vaandrager, “The difference between Splitting in *n* and *n+1*,” Report CS-R9553, Centre for Mathematics and Computer Science, Amsterdam 1995, Abstract in: Proceedings 3rd Workshop on Concurrency and Compositionality, Goslar, March 5-8, 1991 (E. Best & G. Rozenberg, eds.), GMD-Studien Nr. 191, Sankt Augustin, Germany 1991. Information and Computation, Vol. 136, No. 2, pp. 109–142, 1997.

[6] D. I. Cheremisinov, “The real difference between linear and branching temporal logics,” Workshop on Discrete-Event System Design DESDes’04, University of Zielona Gora Press, Poland, pp. 103–108, 2004.

[7] A. D. Zakrevski, “Parallel algorithms of logic control,” Minsk, Belarus, pp. 202, 1999. (in Russian)

[8] D. I. Cheremisinov and L. Cheremisinova, “Specifying agent interaction protocols with Parallel control algorithms,” Proceedings of 11th International Conference of Knowledge-Dialog-Solution (KDS’05), Varna, Bulgaria, pp. 496–503, June 20–30, 2005.

[9] R. Milner, “A calculus of communication systems,” LNCS’92, Springer Verlag, 1980.

[10] C. E. Shannon, “A mathematical theory of communication,” Bell System Technical Journal, Vol. 27, pp. 379–423, pp. 623–656, 1948.

[11] J. Odell, H. V. D. Parunak, M. Fleischer, and S. Brueckner, “Modeling agents and their environment,” Proceedings of the 3 International Workshop on Agent Oriented Software Engineering, Lecture Notes in Computer Science, Springer Verlag (Berlin, D), Vol. 2585, pp. 16–31, 2003.

[12] F. Finin, Y. Labrou and J. Mayfield, “KQML as an agent communication language,” edited by J. M. Bradshaw, Software Agents, MIT Press, pp. 291–316, 1997.

[13] Foundation for Intelligent Physical Agents (FIPA), http://www.fipa.org.

[14] V. Lesser, “Cooperative multiagent systems: A personal view of the state of the art,” in: IEEE Transactions of Knowledge and Data Engineering, Vol. 11, No. 1, pp. 133–142, 1999.