



THE APPROACH TO PROGRAMMING AGENT-BASED SYSTEMS

**Dmitry Cheremisinov
Liudmila Cheremisinova**

**The United Institute of Informatics Problems of
National Academy of Sciences of Belarus**

**8-th International Workshop on Boolean Problems
September 18 - 19, 2008, Freiberg (Sachsen)**



Obstacles to the Widespread Adoption of Multi-Agent Technologies

- ❖ the lack of systematic methodologies enabling designers to clearly specify and structure their applications as MASs;
- ❖ the lack of widely available MAS toolkits
- ❖ flexible tools are needed enabling designers
 - to specify an agent's problem-solving behaviour,
 - to specify agents interaction,
 - to visualize, debug and implement the behaviour of the agents and the entire MAS.



What is suggested

- ❖ The purpose is to explore the possibility of applying the language PRALU, proposed for description of concurrent logical control algorithms, for design and simulation real-time multi-agent systems
- ❖ The methodology of programming agents in PRALU is offered that is based on two-block architecture: control and the functional blocks



Agents

M.Wooldridge and N.R.Jennings:

Agent is a hardware or software-based computer system that enjoys the following properties:

- ❖ ***autonomy***: agents operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state;
- ❖ ***social ability***: agents interact with other agents (and possibly humans) via some kind of *agent-communication language*;
- ❖ ***reactivity***: agents perceive their environment, and respond in a timely fashion to changes that occur in it;
- ❖ ***pro-activeness***: agents do not simply act in response to their environment, they are able to exhibit goal-directed behaviour by *taking the initiative*.



Multi-Agent Systems (MAS)

MAS is a concurrent system consisting of a number of autonomous, reactive and internally-motivated agents acting in a decentralized environment.

MAS can operate if the agents are able to exchange information in the form of messages and have a common understanding of them.

- ❖ **agent communication languages** – for specification of a domain specific vocabulary (an ontology) and messages that can be exchanged between agents (KQML, FIPA ACL)
- ❖ **interaction protocols** specify sequences in which messages should be arranged in agent interactions



Agent Interaction Protocols

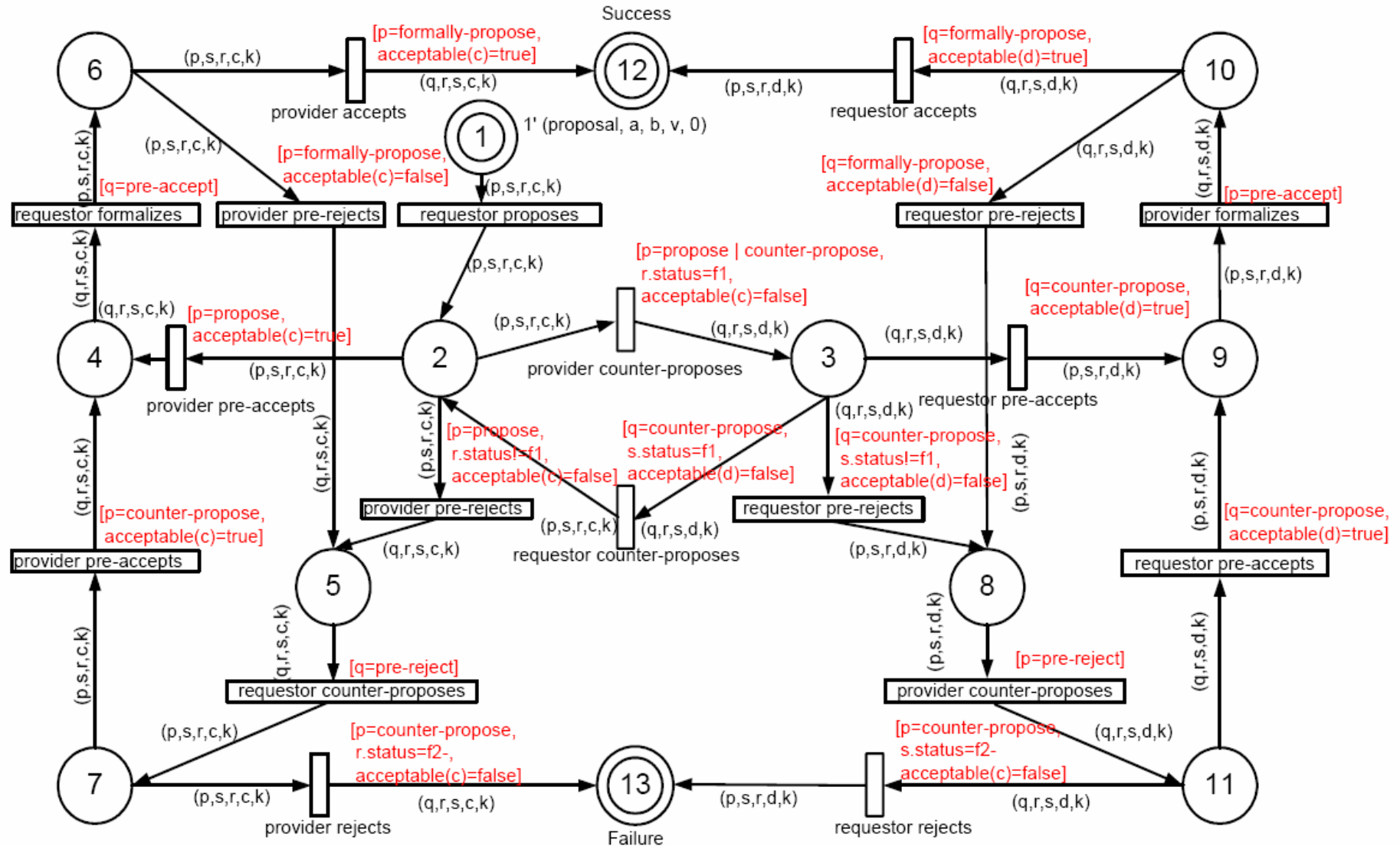
- ❖ **Deterministic finite automata (DFA)**
- ❖ **Message flow diagram (used by FIPA (Foundation for Intelligent Physical Agents))**
- ❖ **Statecharts**
- ❖ **UML (Unified Modelling Language) and Agent UML interaction diagrams**
- ❖ **Colored Petri Net (CPN)**

Petri Net models

- ❖ **are considered today to be one of the best ways to model agent interaction protocols;**
- ❖ **provide an appropriate mathematical formalism for the description and analysis of distributed and concurrent systems.**



CPN Protocol of two Agents Conversation





Special Language PRALU

PRALU language

- ❖ has its **background in the Petri net theory** (expanded nets of free choice – EFC–nets);
- ❖ possesses special means for **keeping track of the current states** of the conversation, receiving messages and initiating responses;
- ❖ **combines properties** of “cause-effect” models with Petri nets;
- ❖ is well suited for representation of the interactions involved in **concurrent system**;
- ❖ is **simple** enough for understanding;
- ❖ supports **hierarchical descriptions**;
- ❖ supports **graphical** and **symbolic forms** of descriptions;
- ❖ is supported by powerful **software** developed for verification, simulation, hardware and software implementation.



Algorithms in PRALU

2 basic operations:

acting operations “ $\rightarrow A$ ” (producing event $A = 1$)

waiting operations “ $- p$ ” (waiting for event $p = 1$)

“ $- x_j z_k$ ” and “ $\rightarrow y_p z_q$ ” “ $-(a > b + c)$ ” and “ $\rightarrow (a := b + c)$ ”

Additional operations: suppression (“ $\rightarrow*$ ”, “ $\rightarrow*\gamma$ ”, “ $\rightarrow'\gamma$ ”), arithmetic, timeout (“ $-n$ ”) and counting (“ $\rightarrow(x+)$ ”, “ $\rightarrow(x-)$ ”, “ $-(x=n)$ ”) operations

Algorithm in PRALU is presented as an unordered set of chains:

$$\mu_j: - p_j L_j \rightarrow v_j,$$

L_j is a linear algorithm, ($L_j = “- p \rightarrow A \rightarrow B - q \rightarrow C”$)

μ_j, v_j – initial and terminal chain labels: $\mu_j, v_j \in M = \{1, 2, \dots, m\}$,

“ $\rightarrow v_j$ ” – transition operation

“ $2: -p \rightarrow A \rightarrow B - q \rightarrow C \rightarrow 3.4.5$ ”, “ $7.8: -5 \rightarrow D \rightarrow 3.4.5$ ”

PRALU-Algorithms Execution

Chains are fulfilled both **serially** and **concurrently** depending on the variable starting set $N_t \subseteq M$:

$\alpha_j = \mu_j: -p_j L_j \rightarrow v_j$ is activated if $\mu_j \subseteq N_t$ and $p_j = 1$

After that $N_t \Rightarrow N_t \setminus \mu_j \Rightarrow (N_t \setminus \mu_j) \cup v_j$

**Concurrent
branching**

1: ...→2.3

2: ...

3: ...

**Merging
conc.branching**

2: ...→4

3: ...→5

4.5: ...

**Alternative
branching**

1: - a...→2

- \bar{a} ...→3

**Converging
altern.branching**

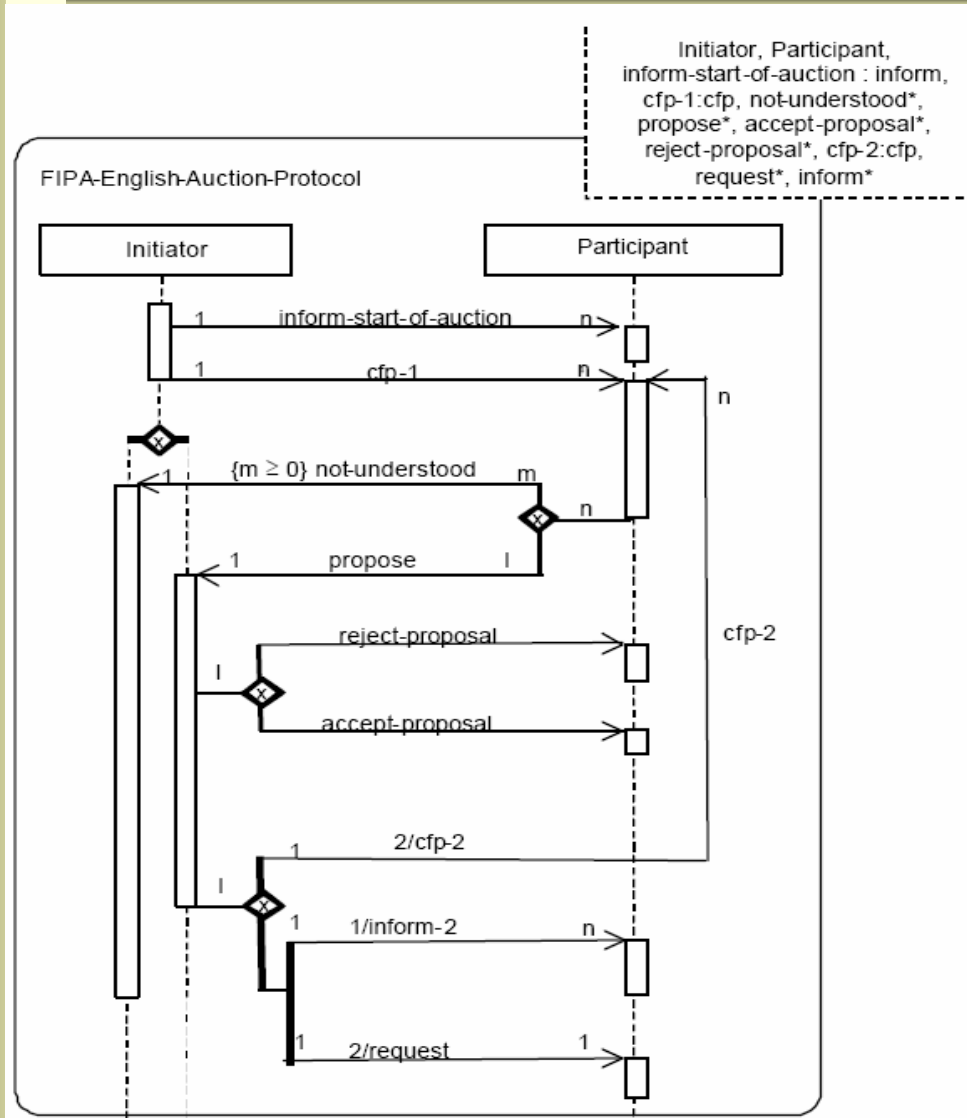
2: ...→4

3: ...→4

4: ...



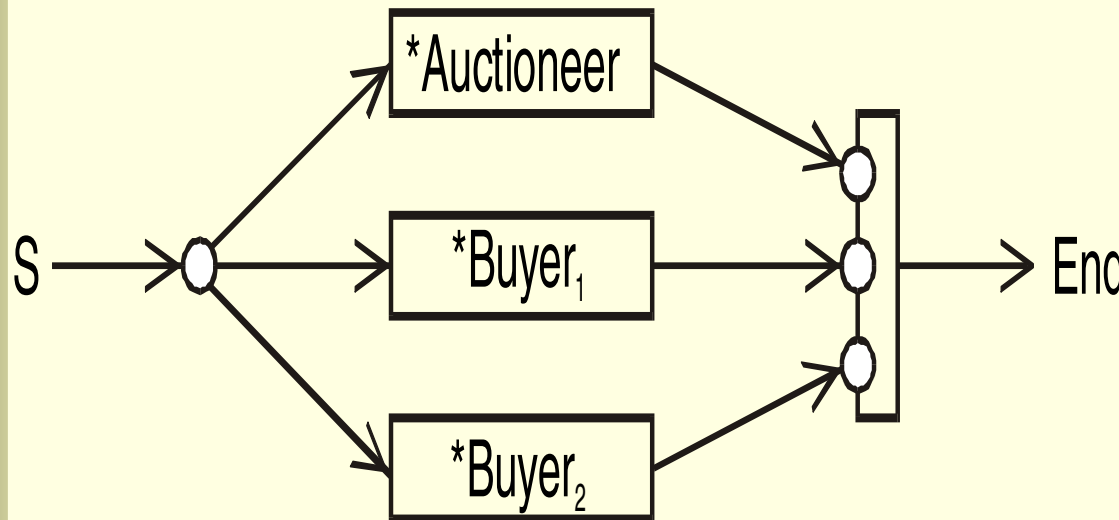
The Example of English Auction (FIPA Diagram)



- ❖ The auctioneer seeks to find the market price of a good by initially proposing to buyers a price below than that of the supposed market value and then gradually raising the price
- ❖ Each time the price is announced, the auctioneer waits if any buyer will signal their willingness to pay the price
- ❖ The auction continues until no buyers are ready to pay the proposed price, then the auction ends



English Auction Interaction Protocol in PRALU



Main_process ()

1: →2.3.4

2: →*Auctioneer →5

3: →*Buyer₁ →6

4: →*Buyer₂ →7

5.6.7: →.

- ❖ There are participants (and interaction protocols) of 2 types: **Auctioneer** and **Buyers**:
- ❖ The processes Auctioneer and Buyers are **executed concurrently**
- ❖ PRALU-blocks are **exchanging** with values of **global logical variables**
- ❖ Each block has some sets of **input** and **output variables**



Interaction Protocol of Auctioneer: Symbolic Form

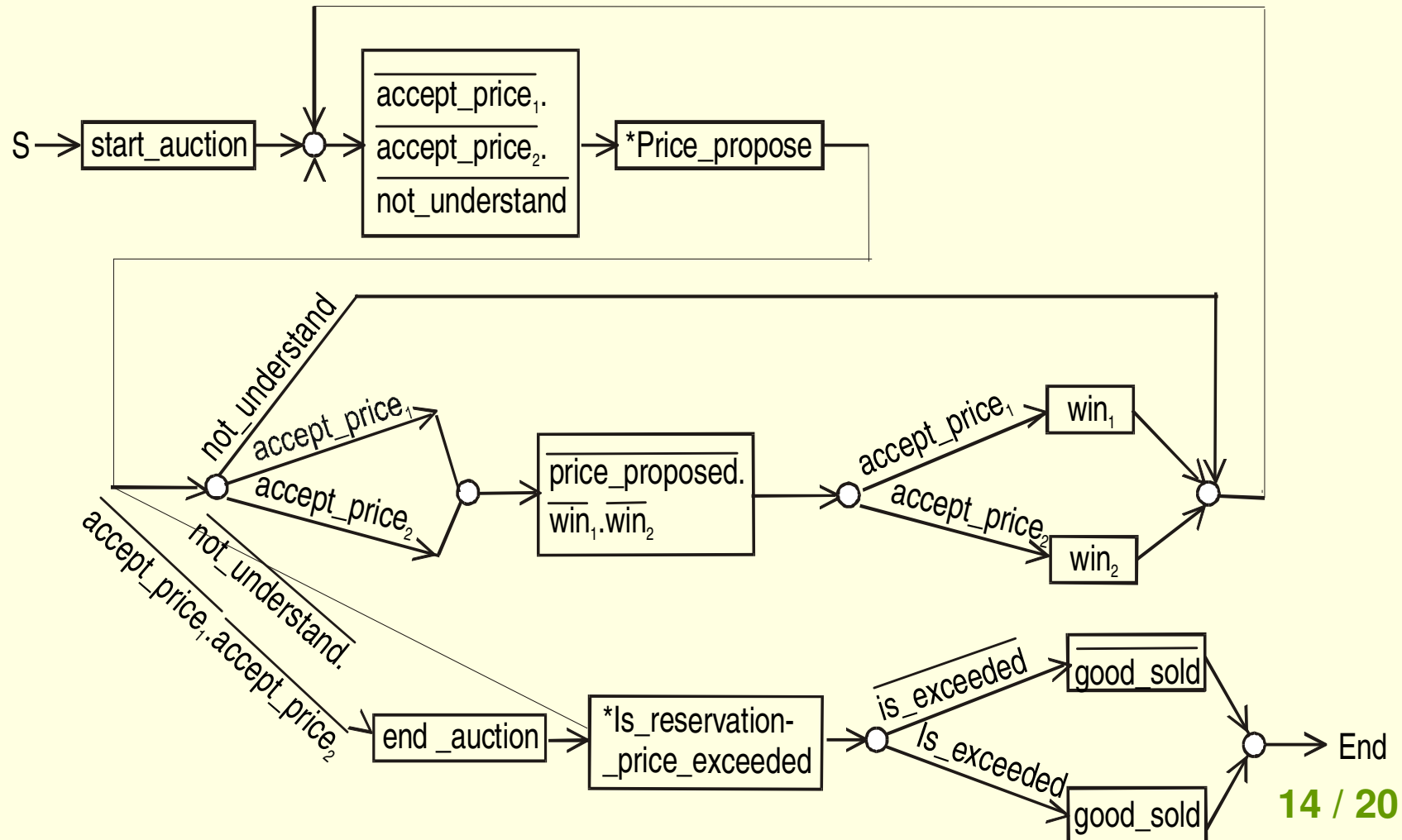
**Auctioneer (accept_price₁, accept_price₂, not_understand /
start_auction, price_proposed, end_auction)**

- 1: →start_auction →2** **auction beginning**
- 2: →'accept_price₁.'accept_price₂.'not_understand
→*Price_propose(/price_proposed) →3** **complex acting operation**
- 3: -not_understand →2**
- accept_price₁ →4
- accept_price₂ →4
-'not_understand.'accept_price₁.'accept_price₂ →end_auction
→*Is_reservation_price_exceeded(/is_exceeded) →6
- 4: →'price_proposed.'win₁.'win₂ →5**
- 5: - accept_price₁ →win₁ →2**
- accept_price₂ →win₂ →2
- 6: - is_exceeded →good_sold →7**
-' is_exceeded →'good_sold →7
- 7: →.**



Interaction Protocol of Auctioneer: Graphic Form

Auctioneer (accept_price₁, accept_price₂, not_understand /
start_auction, price_proposed, end_auction)





Interaction Protocol of a Buyer: Symbolic Form

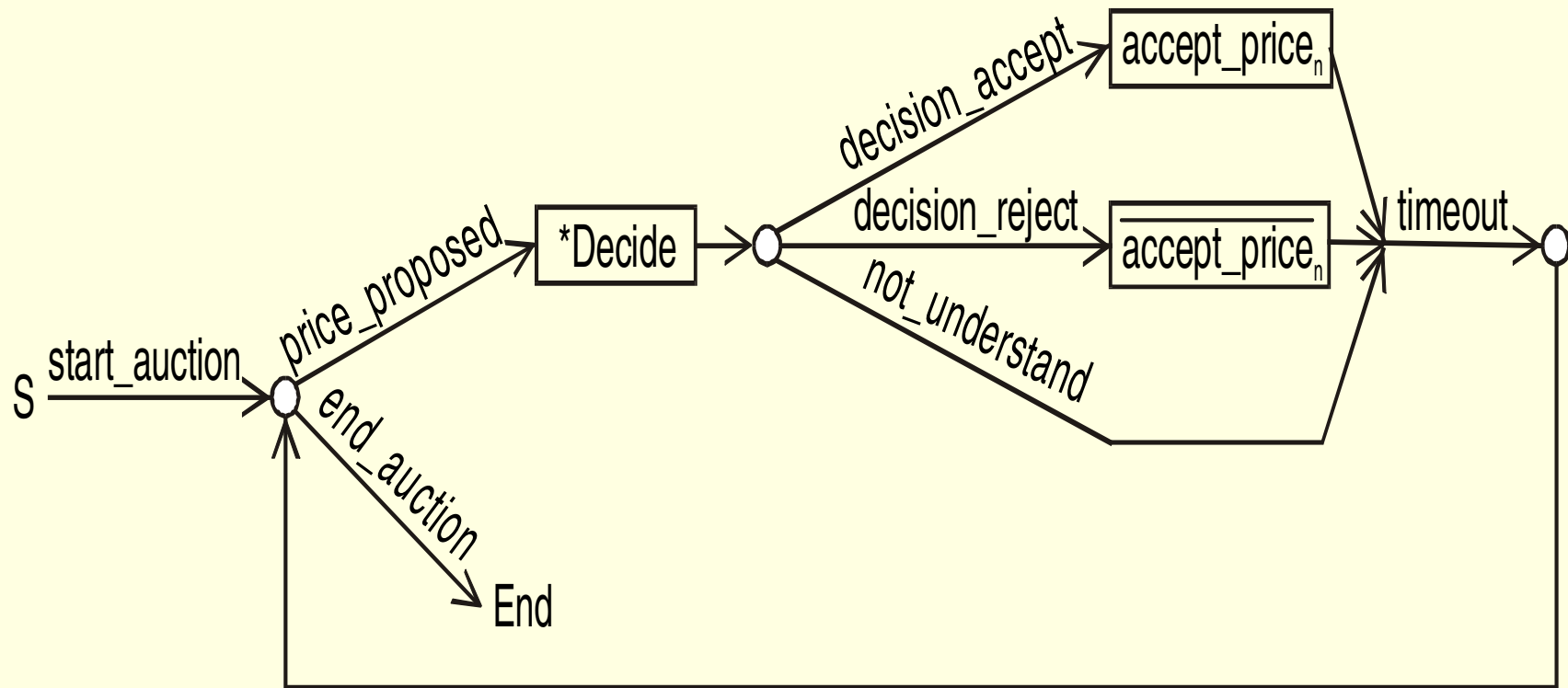
**Buyer_n (start_auction, price_proposed, end_auction /
accept_price_n, not_understand)**

- 1: -start_auction →2** **beginning start of auction**
- 2: -price_proposed →*Decide(/decision_accept,decision_reject) →3**
-end_auction →. **complex acting operation**
- 3: -decision_accept → accept_pricen →4**
-decision_reject →'accept_pricen →4
-not_understand →4
- 4: -timeout →2** **waiting for "timeout" unit times**



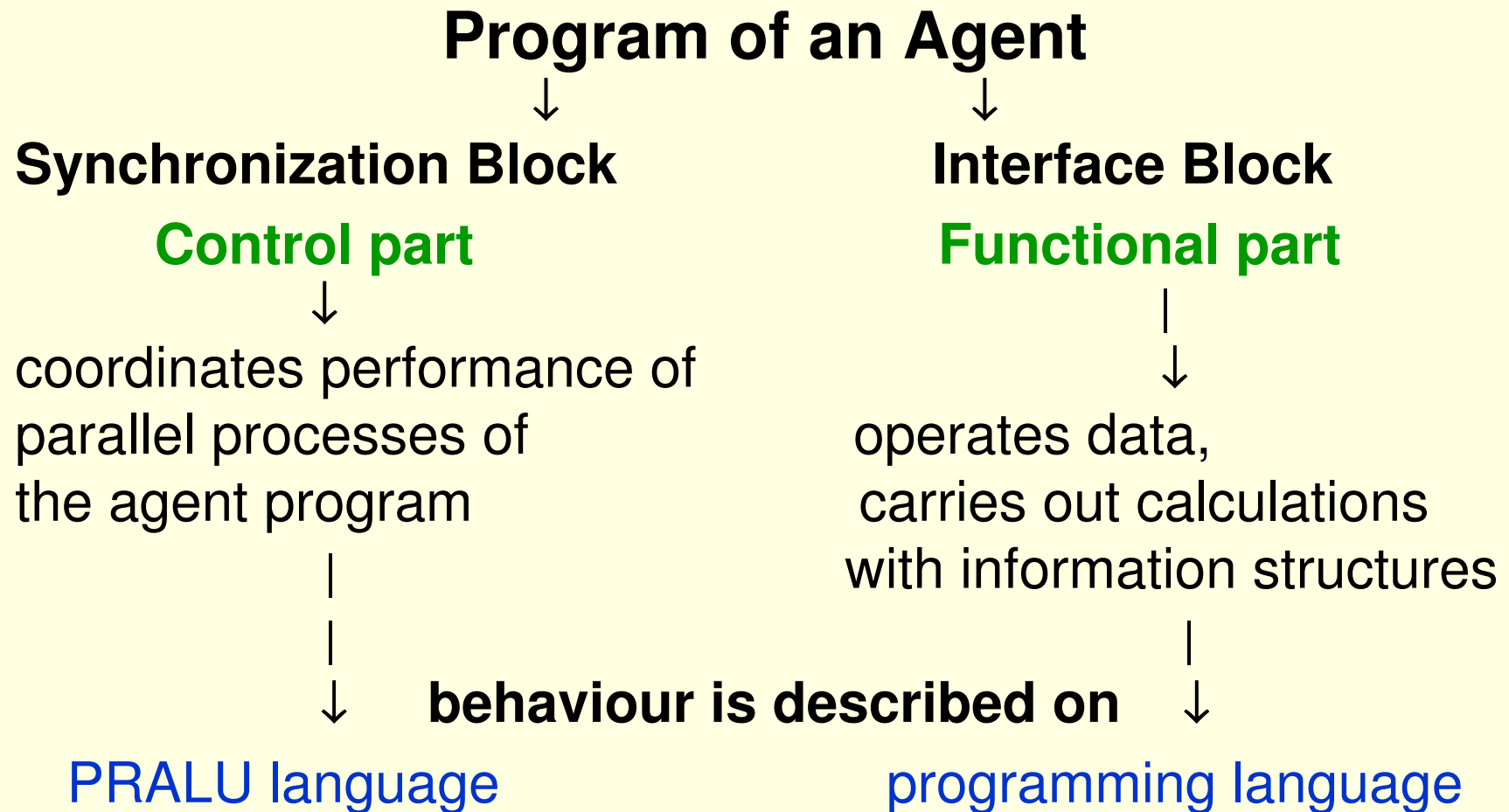
Interaction Protocol of a Buyer: Graphic Form

Buyer_n (start_auction, price_proposed, end_auction /
accept_price_n, not_understand)



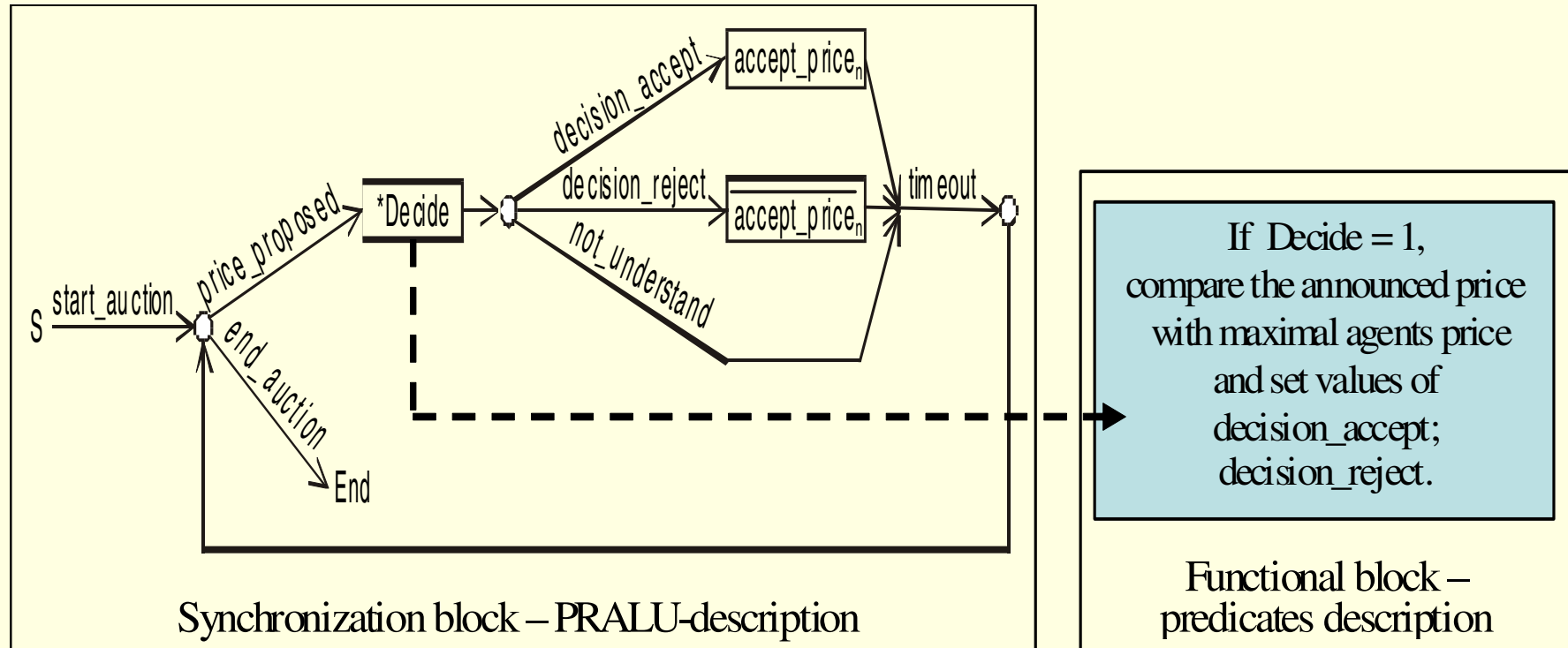


Methodology of Programming Agents on PRALU





Methodology of Programming Agents on PRALU



Functional part realizes predicates that prescribe performance of some actions. The appropriate logical variable is introduced for each predicate, that is set in true value to start the process of the predicate calculation.



Multi-Agent systems design using PRALU

1. Splitting at functional level the specification of MAS to be designed into control and functional parts.
2. Development and analysis of PRALU-description of the control part of MAS, being based on informal specification of its interaction protocol.
3. Verification of logic consistency of the MAS behaviour by checking correctness and simulation of PRALU-description.
4. Program implementation of the functional part of MAS.
5. Program implementation of the control part of MAS: translating PRALU-description of MAS on programming language
6. Binding control and functional parts of the MAS program.
7. Testing of the generated programs.



Conclusion

- ❖ It is shown that PRALU language is well suited for specifying and modelling interaction protocols of MAS;
- ❖ The methodology of programming agents in PRALU is suggested that is based on two-block architecture and that:
 - ensures structuring the process of MAS designing by means of separating control part of MAS specification from calculation part;
 - allows the designer to concentrate on more complex stage of MAS designing – its interaction protocol .

Thank you for your attention