UDC 164=111

*N.O. Garanina*

# ELEUSIS: PERFECT RECALL FOR INDUCTIVE REASONING

This paper formalizes a variant of inductive game Eleusis. A model of the game is an interpreted system with perfect recall agents for the players and the dealer. The pecularity of Eleusis multi-agent system is that the agents have to guess the behavior of the system, rather than some static information about the system. We express some Eleusis rules and properties of the system by the formulas of propositional knowledge logic and branching time *Act*-CTL-$K_n$.

MULTIAGENT SYSTEMS; EPISTEMIC LOGIC; ELEUSIS; PERFECT RECALL; INTERPRETED SYSTEMS.

*Н.О. Гаранина*

# ЭЛЕВСИН: АБСОЛЮТНАЯ ПАМЯТЬ ДЛЯ ИНДУКТИВНОГО ВЫВОДА

Формализован вариант индуктивной карточной игры Элевсин. Моделью игры является интерпретированная система с агентами с абсолютной памятью, реализующими игроков и раздающего. Особенностью мультиагентной системы Элевсин является то, что агенты должны вычислить поведение самой системы, а не какую-либо статическую информацию. Некоторые правила игры Элевсин и свойства построенной мультиагентной системы выражены формулами логики знаний и действий *Act*-CTL-$K_n$.

МУЛЬТИАГЕНТНЫЕ СИСТЕМЫ; ЛОГИКА ЗНАНИЙ; ЭЛЕВСИН; АБСОЛЮТНАЯ ПАМЯТЬ; ИНТЕРПРЕТИРОВАННЫЕ СИСТЕМЫ.

---

There are a lot of puzzles that give clear comprehension of various notions from a theory of epistemic logics. For example, the Coordinated Attack Problem [3] shows unreachability of common knowledge in the environment with time gaps; in Muddy Children Puzzle [3] knowledge is based on the quantity of observations, while in Dining Cryptographers Problem [9] knowledge is based on the quality of observations; Mars Robot Puzzle [14] illustrates knowledge taken by double-checking observations; False Coin Puzzle [11] demonstrates knowledge acquisition. In the last two puzzles and in the puzzle from this paper perfect recall for agents is crucial[1].

To the best of our knowledge, epistemic logics was not used for a definition of inductive knowledge of agents before. In this paper we formalize inductive game Eleusis. This game was introduced by Robert Abbot in 1956. We give a description of the first variant of the game from [4]:

Eleusis (pronounced ee-loo-sis) is a game for three or more players. It makes use of the standard deck of playing cards. Players take turns at being the dealer, who has no part in the actual play except to serve as a kind of umpire. He deals to the other players until one card remains. This is placed face up in the center of the table as the first card of the «starter pile». To make sure that players receive equal hands, the dealer must remove a certain number of cards before dealing. For three players (including the dealer, who of course does not get a hand) he removes one card; for four players, no cards; five players, three cards, and so on. The removed cards are set aside without being shown.

After the cards are dealt and the «starter card» is in place, the dealer makes up a secret rule that determines what cards can be played on the starter pile. It is this rule that corresponds to a law of science; the players may think of the dealer as Nature, or, if they prefer, as God. The dealer writes his rule on a piece of paper,

[1] For the Dining Cryptographers this notion is not so important because substantial information (even or odd number of differences) could be coded by a boolean variable.

which he folds and puts aside. This is for later checking to make sure that the dealer does not upset Nature's uniformity by changing his rule. For each player the object of the game is to get rid of as many cards as possible. This can be done rapidly by any player who correctly guesses the secret rule.

An example of a very simple rule is: «If the top card of the starter pile is red, play a black card. If the top card is black, play a red card».

There are many studies of this game, but they refer to such aspects as induction algorithms for players discovering the rule [2], machine learning technique [10] or the computational complexity of various decision problems that arise in Eleusis [7]. Multi-agent systems and epistemic logics were not in the focus of the game researchers before.

We study Eleusis in trace-based synchronous perfect recall multi-agent systems. In such systems agents have memory: their knowledge depends on the states passed and on the previous actions. We can describe this kind of agents because semantics of knowledge is defined on traces, i. e. finite sequences of states and actions, and every agent can distinguish traces with different sequences of information available for it. Each element of the trace represents a state of the system at some moment of time. A model of this game differs from other multi-agent systems as in this system the agent has complete information about the system and tries to guess modus operandi of another agent or of the environment.

For formalization of Eleusis world we use a Propositional Logic of Knowledge and Branching Time *Act*-CTL-K$_n$ [12] and a notion of an interpreted system [6, 3] enriched by perfect recall [8]. Properties of these systems could be checked by some model checker of multi-agent systems with perfect recall, for example, MCMAS [9].

## Background Logics and Models

First, we would like to give definitions to combined Propositional Logic of Knowledge and Branching Time *Act*-CTL-K$_n$ from [12] briefly. This logic is a fusion of Propositional Logic of Knowledge (PLK) [3] and Computational Tree Logic (CTL) [1] extended by action symbols. Usually, semantics of *Act*-CTL-K$_n$ is defined in terms of a satisfiability relation $\models$ in environments that are a special kind of labeled transition systems. In this paper we prefer to use interpreted systems which are a specific form of environments.

Let us reformulate the definition of interpreted systems from [6]. We denote a set of agents by $A = \{1, ..., n\}$ ($n \in N$), for each agent $i \in A$ a set of local states by $L_i$ and possible actions by $Acts_i$, and a set of local states and actions for the environment by $L_e$ and $Acts_e$ respectively. Let $Act = Acts_1 \times ... \times Acts_n \times Acts_e$ be the set of joint actions. We assume that a set of protocols $P_i : L_i \to 2^{Acts_i}$, for $i \in A$, represents the behavior of every agent, and a protocol $P_e : L_e \to 2^{Acts_e}$ for the environment.

A formal definition of interpreted systems is as follows:

**Definition 1.** (*of interpreted systems*). Given a set of agents $A = \{1, ..., n\}$ *an interpreted system* is a tuple $M = (G; t; P; \sim_1, ..., \sim_n; \iota; V)$, where

(1) $G$ is the set of the global states for the system $G = L_1 \times ... \times L_n \times L_e$; $\iota \in G$ is the initial state;

(2) a transition function $t : G \times Act \to G$ models computations in the system;

(3) a joint protocol $P : G \to 2^{Act}$ defines the behavior of the system such as $P = \langle P_1, ..., P_n, P_e \rangle$, where $P_i$ and $P_e$ are protocols of agents and environment;

(4) $\sim_i \subseteq G \times G$ ($i \in A$) is an epistemic indistinguishability relation for each agent $i \in A$ defined by $s \sim_i s'$ iff $l_i(s') = l_i(s)$, where the function $l_i : G \to L_i$ returns the local state of agent $i$ from a global state $s$;

(5) $V : G \to 2^{Prp}$ is a valuation function for a set of propositional variables *Prp* such as *true* $\in V(s)$ for all $s \in G$. $V$ assigns to each state a set of propositional variables that are assumed to be true at that state.

For every $a \in Act$ an *a-run* is a maximal sequence of states $gs = s_1 ... s_j s_{j+1} ...$ such as $t(s_j, a) = s_{j+1}$ for all $j > 0$.

Every indistinguishability relation expresses the fact that an agent has incomplete information about system states.

**Definition 2.** (*of Act-CTL-K$_n$ syntax*). *Syntax* of *Act*-CTL-K$_n$ consists of formulas that are constructed from Boolean constants {*true*, *false*}, propositional variables, connectives ¬,

$\wedge$, $\vee$, and the following modalities. Let $i \in A$, $a \in Act$, $\varphi$ and $\psi$ be formulas. Then formulas with the modalities are:

- knowledge modalities: $K_i\varphi$ and $S_i\varphi$ (they are read as `an agent $i$ knows' and `an agent $i$ supposes');
- action modalities: $\boldsymbol{AX}^a\varphi$, $\boldsymbol{EX}^a\varphi$, $\boldsymbol{AG}^a\varphi$, $\boldsymbol{EG}^a\varphi$, $\boldsymbol{AF}^a\varphi$, $\boldsymbol{EF}^a\varphi$, $\boldsymbol{A}\varphi\boldsymbol{U}^a\psi$, and $\boldsymbol{E}\varphi\boldsymbol{U}^a\psi$ ($\boldsymbol{A}$ is read as `for all futures', $\boldsymbol{E}$ – `for some futures', $\boldsymbol{X}$ – `next time', $\boldsymbol{G}$ – `always', $\boldsymbol{F}$ – `sometime', $\boldsymbol{U}$ – `until', and a sup-index $a$ – `in a-run(s)').

Syntax of *Act*-CTL-$K_n$ combines modalities of PLK [3] and CTL [1] with action symbols. Semantics of *Act*-CTL-$K_n$ follows semantics of these logics.

**Definition 3.** (*of Act-CTL-$K_n$ semantics*). A satisfiability relation $\models$ between interpreted systems, states, and formulas is defined inductively with respect to a structure of formulas. For Boolean constants, propositional variables, and connectives a satisfiability relation is standard. For the action modalities semantics is almost the same as for the standard CTL-modalities, but is formulated for $a$-runs. For the knowledge modalities we define the semantics as follows. Let $s \in G$, $i \in A$, $\varphi$ be a formula, then

- $s \models_M (K_i\varphi)$ iff for every $s'$ : $s \sim_i s'$ implies $s' \models_M \varphi$;
- $s \models_M (S_i\varphi)$ iff for some $s'$ : $s \sim_i s'$ and $s' \models_M \varphi$.

*Semantics of a formula* $\varphi$ in an interpreted system $M$ is the set of all worlds of $E$ that satisfies this formula $\varphi$: $M(\varphi) = \{s \mid s \models_M \varphi\}$.

Further we consider just *Act*-CTL-$K_n$ normal formulas in which negation is used in literals only. Every *Act*-CTL-$K_n$ formula is equivalent to some normal formula due to «De Morgan» laws.

We use trace-based perfect recall synchronous (PRS) interpreted systems generated from background interpreted systems. In PRS environments (1) states are sequences of states of initial interpreted systems with a history of generating actions; (2) there are transitions from one sequence to another with an action $a$ made by extending the sequence with a state reachable by $a$ from the last state of the sequence; (3) perfect recall protocols take into account local traces of agents when modelling their actions; (4) an agent does not distinguish such sequences if the background system performs the same sequence of actions, and the agent cannot distinguish the sequences state by state; (5) propositional variables are evaluated at the last state of sequences with respect to their evaluations in the background interpreted systems.

**Definition 4.** (*of a PRS-system*). Let $M$ be an interpreted system $M = (G; t; P; \sim_1, ..., \sim_n; \iota; V)$. A trace-based *Perfect Recall Synchronous interpreted system* generated by $M$ is another interpreted system $prs(M) = (G_{prs}; t_{prs}; P_{prs}; \approx_1, ..., \approx_n; \iota_{prs}; V_{prs})$[2]:

(1) $G_{prs}$ is the set of all pairs ($gs$, $as$), where non-empty $gs \in G^*$, $as \in Act^*$, $|gs| = |as| + 1$, and $t(gs_j, as_j) = gs_{j+1}$ for every $j \in [1..|as|]$; $ls_i(gs) \in L_i^*$ is a sequence of local states of agent $i \in A$ with $(ls_i)_j = l_i(gs_j)$; $\iota_{prs} \in G_{prs}$ is the initial state;

Let us assume that ($gs$, $as$), ($gs'$, $as'$) $\in G_{prs}$:

(2) a transition function $t_{prs} : G^{prs} \times Act \to G_{prs}$ is defined as follows: for every $a \in Act$: $t_{prs}((gs, as), a) = (gs', as')$ iff $as' = as^\wedge a$, $gs' = gs^\wedge s'$, and $t(g_{|gs|}, a) = s'$;

(3) a joint protocol $P_{prs}(M) : G_{prs} \to 2^{Act}$ such as $P_{prs} = \langle (P_1)_{prs}, ..., (P_n)_{prs}, (P_e)_{prs} \rangle$, where every $(P_i)_{prs}$: $L_i^* \to 2^{Acts}i$ does not depend on $P_i$[3] and $(P_e)_{prs} : L_e^* \to 2^{Acts}e$ such as $(P_e)_{prs}(ls_e \wedge s_e) = P_e(s_e)$;

(4) for every $i \in A$: ($gs$, $as$) $\approx_i$ ($gs'$, $as'$) iff $as = as'$ and $gs_j \sim_i gs'_j$ for every $j \in [1..|gs|]$;

(5) for every $p \in Prp$: ($gs$, $as$) $\in V_{prs}(p)$ iff $gs_{|gs|} \in V(p)$.

In PRS-systems agents have some kind of memory because an awareness expressed by an indistinguishability relation depends on the history of the system evolution. Actions of perfect recall agents are knowledge-based because protocols of agents take into account the previous states and actions.

## Eleusis Model

In order to define the background Eleusis interpreted system $E$-$M$ we have to determine the following features: (1) agents and an environment; (2) their local states and

[2] In the definition, for every set $S$ let $S^*$ be the set of all finite sequences over $S$ and the operation $\wedge$ stand for the concatenation of finite words.

[3] We model behavior according to information history, but not to the single local state.

generated global states; (3) local agent actions and generated global actions; (4) a transition function; (5) protocols of agents and the environment; (6) indistinguishability relations for agents; (7) propositional variables and their evaluation.

(1) Agents and an environment. Let players of the game and the dealer be agents. The environment may not be specified in our case. Without loss of generality, we consider two players – Alice and Bob, and the dealer Clare.

(2) Local and global states. It is not reasonable to fix special local states for agents because they have complete information about the system. We define the global state using the following local variables:

$Value = \{2, 3, 4, 5, 6, 7, 8, 9, 10, Kn, Q, K, A\}$ is a set of card values;

$Suit = \{\spadesuit, \clubsuit, \diamondsuit, \heartsuit\}$ is a set of suits;

$Cards = Value \times Suit$ is a set of cards;

$Obs \in Cards$ is the last observable card in the dealer's pile;

$Prt \in Cards \cup \{\varnothing\}$ is a pretender card to satisfy the rule put by some player;

$Deck_A$, $Deck_B \subset Cards$ are the decks of Alice and Bob which they have to get rid of;

$Turn \in \{A, B\}$ determines whose move is next;

$Moves \in [0..n]$ is a number of dealer replies. Let $n$ be big enough to discover the rule by players.

Then every global state is $(Obs, Prt, Deck_A, Deck_B, Turn, Moves) \in G$, where $G = Cards \times (Cards \cup \{\varnothing\}) \times 2^{Cards} \times 2^{Cards} \times \{A, B\} \times [0..n]$.

We set $\iota = (Obs, \varnothing, Deck_A, Deck_B, Turn, 0)$, where $Obs \in Cards$, $Deck_A$, $Deck_B \subset Cards \setminus \{some\ card\}$[4] as initial state, and $Turn \in \{A, B\}$. Further we use the following notation: for every card $C \in Cards$ let $C.Val$ be a value of $C$ and $C.Suit$ be a suit of $C$.

(3) Local and global actions. Let us define the following actions of agents:

*turn* is an action of agent-players. The result of this action is a new pretender card;

*accept* is an action of the agent-dealer used for validation that the pretender card satisfies the rule. The result is a new observable card and a decreased deck of a player that gives the pretender;

*reject* is an action of the agent-dealer used for disproof that the pretender card satisfies the rule. The result is passing the move.

Let us note that in this system agents cannot act parallelly because their actions depend on each other. Hence global actions are rather a PDL-program [5] of actions $move = turn; (accept \cup reject)$ (to make an experiment then to give a result) than a product $turn \times \{accept, reject\}$. It is easy to make a product by the PDL-program move introducing lazy action *skip* which does nothing for every agent.

(4) A transition function. The transition function for actions is given in terms of pre- and post-conditions of local state variables: $(pre_1, ..., pre_5) \rightarrow (post_1, ..., post_5)$, where for every $j \in [1..5]$ (1) $pre_j$ is a precondition for the corresponding variable; the precondition of omitted variable means that the corresponding variable has any value; (2) $post_j$ is a postcondition for the corresponding variable; the postcondition of omitted variable means that the corresponding variable has the same value as in the precondition. Let $card \in Cards$ and $other\_cards \subset Cards$. We describe turn and reject for Alice only, because for Bob these actions are similar.

1. *turn*:

$(Prt = \varnothing,\ Deck_A = \{card\} \cup other\_cards,\ Turn = A) \rightarrow (Prt = card,\ Deck_A = other\_cards)$;

2. *accept*:

$(Prt = card) \rightarrow (Obs = card,\ Prt = \varnothing,\ Moves = Moves + 1)$;

3. *reject*:

$(Prt = card,\ Deck_A = other\_cards;\ Turn = A) \rightarrow$

$(Prt = \varnothing,\ Deck_A = \{card\} \cup other\_cards,\ Turn = B,\ Moves = Moves + 1)$.

(5) Protocols of agents. In our model protocol for players is easy: they perform any action in the state. They try an arbitrary card they have. We choose this simple behavior of players because the definition of actions corresponds to reasoning too cumbersome for this short paper. This is a topic for our future work. Nevertheless, it is possible to formulate and analyse some interesting properties of the Eleusis system.

[4] In order to provide our agents with an equal quantity of cards, the dealer has to remove one card from the initial deck.

The protocol for dealer actions depends very much on the chosen rule. In many cases it must be a perfect recall protocol because the dealer relies on the previously accepted cards when accepting or rejecting cards. Some of these protocols are given in next section.

(6) Indistinguishability relations. As game agents have complete information about the system in our variant of Eleusis, we assume that every indistinguishability relation is equality.

(7) Propositional variables and their evaluation. In our case a set of propositional variables strictly depends on formulas to be verified. Hence we define them later using values of local variables. Thus a valuation function $V$ is defined naturally.

The corresponding perfect recall interpreted system for Eleusis $E\text{-}prs(M)$ can be made from $E\text{-}M$ by Definition 4. Perfect recall is really needed for agent-players because in order to discover the rule they have not only to consider accepted cards, but to remember rejected cards. Several perfect recall protocols for the dealer depending on the rules are given in the section below.

### Eleusis Rules

The most interesting thing in Eleusis and its formalization is the accepting rule. We consider the rules that use card descriptions only[5], i. e. their values and suits (colors and faces are subsets of suits and values). There are variants of Eleusis where rules could be defined as a function from sequences to boolean values [7]. Time modalities next $\boldsymbol{X}$, globally $\boldsymbol{G}$ and until $\boldsymbol{U}$ are sufficient for a wide class of Eleusis rules[6]. We formulate several rules from [4] using these time modalities. The corresponding dealer protocols are described too. Let us define the following sets of cards: $Even = \{card \mid card.Val \in \{2, 4, 6, 8, 10, Q\}\}$, $Odd = \{card \mid card.Val \in \{3, 5, 7, 9, Kn, K, A\}\}$, $Black = \{card \mid card.Suit \in \{\spadesuit, \clubsuit\}\}$, and $Red = \{card \mid card.Suit \in \{\diamondsuit, \heartsuit\}\}$. Let $v \in Value$ and $s \in Suit$.

In dealer actions the above PDL-program $move = turn; (accept \cup reject)$ follows the Dealer protocol corresponding to the rule of the game. The dealer protocol is a function from sequences of the system states to a set of dealer actions enabled at these sequences. For sequences we use the following regular notation. Let $p$ be a propositional condition for system variables that uses set inclusion, subsets, equalities, inequalities of system variables, and boolean connectives. For simplicity we consider $p$ to be a «propositional» state from a set of system states that satisfies with this propositional condition $p$:

- a propositional state $p$ is a sequence of states;
- $p;q$ is a sequence of states which is concatenation of sequences of states from $p$ and $q$;
- $p^*$ is a sequence of states which is a finite repeat of states from $p$.

Let $state$ be a propositional state from a complete set of system states $G$. Let us assume that by the definition of propositional states in sequence $state^*$ states can differ. It is easy to see that the set of states in which the dealer rejects a pretender card is the complement to the set of states in which the dealer accepts the pretender card. We shall describe both cases.

1) Alternatively even and odd cards:

$\boldsymbol{AG}^{move}((Obs \in Even \rightarrow \boldsymbol{AX}^{move}\ Obs \in Odd) \wedge (Obs \in Odd \rightarrow \boldsymbol{AX}^{move} Obs \in Even))$.

The dealer protocol:

$state^*; ((obs \in Even \wedge prt \in Odd) \vee (obs \in Odd \wedge prt \in Even)) \rightarrow accept;$

$state^*; ((obs \in Even \wedge prt \in Even) \vee (obs \in Odd \wedge prt \in Odd)) \rightarrow reject.$

2) The card played must have either the same suit or the same value as the card on top of the pile:

$\boldsymbol{AG}^{move}((Obs.Val = v \wedge Obs.Suit = s) \rightarrow \boldsymbol{AX}^{move}(Obs.Val = v \vee Obs.Suit = s))$.

The dealer protocol:

$state^*; (Obs.Val = prt.Val \vee Obs.Suit = prt.Suit) \rightarrow accept;$

$state^*; (Obs.Val \neq prt.Val \wedge Obs.Suit \neq prt.Suit) \rightarrow reject.$

3) If the top two cards are of the same color, play a card from ace to 7. If they are of different colors, play a card from 7 to king:

[5] It is possible to devise rules taking into account some external features like names of players or the color of their hair. These rules correspond to the experiments in which the effect of a experimenting person and his equipment is considerable.

[6] We do not need F in future because Eleusis rule determines all cards in sequences step by step.

$\mathbf{AG}^{move}(((Obs \in Black) \rightarrow \mathbf{AX}^{move}((Obs \in Black) \wedge \mathbf{AX}^{move}(Obs.Val \in [A..7])) \vee (Obs \in Black) \rightarrow \mathbf{AX}^{move}((Obs \in Red) \wedge \mathbf{AX}^{move}(Obs.Val \in [8..K]))) \vee ((Obs \in Red) \rightarrow \mathbf{AX}^{move}((Obs \in Red) \wedge \mathbf{AX}^{move}(Obs.Val \in [A..7])) \vee (Obs \in Red) \rightarrow \mathbf{AX}^{move}((Obs \in Black) \wedge \mathbf{AX}^{move}(Obs.Val \in [8..K]))))$

The dealer protocol:

$state^*$; $obs \in Black$; $(obs \in Black \wedge prt.Val \in [A..7]) \rightarrow accept$;

$state^*$; $obs \in Red$; $(obs \in Red \wedge prt.Val \in [A..7]) \rightarrow accept$;

$state^*$; $obs \in Black$; $(obs \in Red \wedge prt.Val \in [8..K]) \rightarrow accept$;

$state^*$; $obs \in Red$; $(obs \in Black \wedge prt.Val \in [8..K]) \rightarrow accept$;

$state^*$; $obs \in Black$; $(obs \in Black \wedge prt.Val \in [8..K]) \rightarrow reject$;

$state^*$; $obs \in Red$; $(obs \in Red \wedge prt.Val \in [8..K]) \rightarrow reject$;

$state^*$; $obs \in Black$; $(obs \in Red \wedge prt.Val \in [A..7]) \rightarrow reject$;

$state^*$; $obs \in Red$; $(obs \in Black \wedge prt.Val \in [A..7]) \rightarrow reject$.

4) If the second card from the top is red, play a card with a value equal to or higher than this card. If the second card is black, play a card of equal or lower value:

$\mathbf{AG}^{move}(((Obs \in Red \wedge Obs.Val = v) \rightarrow \mathbf{AX}^{move}(Obs \in Cards) \wedge \mathbf{AX}^{move}(Obs.Val \geq v)) \vee ((Obs \in Black \wedge Obs.Val = v) \rightarrow \mathbf{AX}^{move}(Obs \in Cards) \wedge \mathbf{AX}^{move}(Obs.Val \leq v)))$

The dealer protocol:

$state^*$; $(obs \in Red \wedge Obs.Val = v)$; $(prt.Val \geq v) \rightarrow accept$;

$state^*$; $(obs \in Black \wedge Obs.Val = v)$; $(prt.Val \leq v) \rightarrow accept$;

$state^*$; $(obs \in Red \wedge Obs.Val = v)$; $(prt.Val \leq v) \rightarrow reject$;

$state^*$; $(obs \in Black \wedge Obs.Val = v)$; $(prt.Val \geq v) \rightarrow reject$.

All above rules from [4] are applicable over a fixed number of cards. The next rule uses indefinite, but finite number of cards.

5) Change the color of cards after an ace:

$\mathbf{AG}^{move}((Obs \in Black)\ \mathbf{AU}^{move}(Obs.Val = A \wedge \mathbf{AX}^{move}\ Obs \in Red) \vee (Obs \in Red)\ \mathbf{AU}^{move}\ (Obs.Val = A \wedge \mathbf{AX}^{move}\ Obs \in Black))$.

The dealer protocol:

$(obs \in Black)^*$; $(obs \in Black \wedge Obs.Val = A)$; $(prt \in Red) \rightarrow accept$;

$(obs \in Red)^*$; $(obs \in Red \wedge Obs.Val = A)$; $(prt \in Black) \rightarrow accept$;

$(obs \in Black)^*$; $(obs \in Black \wedge Obs.Val = A)$; $(prt \in Black) \rightarrow reject$;

$(obs \in Red)^*$; $(obs \in Red \wedge Obs.Val = A)$; $(prt \in Red) \rightarrow reject$.

After expressing the accepting rule *Rule* we can formulate the features of the Eleusis model. For example:

1) $\mathbf{AF}^{move}\ (K_A Rule \vee K_B Rule)$ – there is a moment in the future when some player knows the rule;

2) $\mathbf{AF}^{move}(K_A Rule \wedge \neg K_B Rule)$ – there is a moment in the future when Alice knows the rule earlier than Bob;

3) $(Turn = A) \rightarrow \mathbf{AF}^{move}\ (K_A Rule \wedge \neg K_B Rule)$ – if Alice moves first then she is a winner;

4) $\neg(K_A Rule \vee K_B Rule)\ \mathbf{AU}^{move}\ (K_A Rule \vee K_B Rule) \wedge (Moves < m)$ – the rule can be discovered in less than $m$ steps.

Let us define that time complexity of the model checking these formulas depends on the complexity of the rule. The dealer can take into account a finite part of the sequence which determines accepting cards: in this case the model checking the complexity corresponds to the linear complexity of checking *Act*-CTL-$K_n$ formulas in forgetful semantics [13]. But when the dealer makes the decision on pretender cards considering all sequence, then it corresponds to true perfect recall semantics and the model checking properties of the system has non-elementary complexity [12].

In this paper we consider a simple variant of the inductive game Eleusis. A model of the game is an interpreted system with perfect recall agents for the players and the dealer. This is an example of a multi-agent system in which agents have to guess the behavior of the system rather than the information about it. We express some Eleusis rules by formulas of branching time logic Act-CTL and some properties of the system by formulas of *Act*-CTL-$K_n$ with knowledge modalities. Model checking of these properties depends on the complexity of the Eleusis rule. It can be linear or non-elementary according to the size of Eleusis model.

We do not give protocols of players guessing the rule because it is too complex for a short

paper. In the future we plan to develop a simple protocol and try to do model checking of the corresponding Eleusis system with some perfect recall model checker.

The research has been supported by Russian Foundation for Basic Research (grant 13-01-00643) and by Siberian Branch of Russian Academy of Science (Integration Grant n.15/10 «Mathematical and Methodological Aspects of Intellectual Information Systems»).

## REFERENCES / СПИСОК ЛИТЕРАТУРЫ

1. **Clarke E.M., Grumberg O., Peled D.** Model Checking. MIT Press, 1999.

2. **Dietterich T.** Applying General Induction Methods to the Card Game Eleusis. *Proc. of the Ist Annual National Conference on Artificial Intelligence.* Stanford University. AAAI Press/MIT Press, 1980, pp. 218–220.

3. **Fagin R., Halpern J.Y., Moses Y., Vardi M.Y.** Reasoning about Knowledge. MIT Press, 1995.

4. **Gardner M.** The Second Scientific American Book of Mathematical Puzzles and Diversions. University of Chicago Press Edition, 1987, 251 p.

5. **Harel D.** First-Order Dynamic Logic. *Lecture Notes in Computer Science*, 1979, Vol. 68.

6. **Kacprzak M., Lomuscio A., Penczek W.** Verification of Multi-agent Systems via Unbounded Model Checking. *Proc. of the 3 Internat. Joint Conference on Autonomous Agents and Multi-agent Systems*, 2004, Vol. 2, pp. 638–645.

7. **Kurzen L.** Eleusis: Complexity and Interaction in Inductive Inference. *Proc. of the 2 ILCLI Internat. Workshop on Logic and Philosophy of Know-ledge, Communication and Action*, Donostia – San Sebastian, Spain. 2010. The University of the Basque Country Press, 2010.

8. **Lomuscio A. R., van der Meyden R., Ryan M.** Knowledge in Multi-agent Systems: Initial Configurations and Broadcast. *ACM Transactions on Computational Logic*, 2000, Vol. 1, No. 2, pp. 247–284.

9. **van der Meyden R., Su K.** Symbolic Model Checking the Knowledge of the Dining Cryptographers. *Proc. of the 17th IEEE Workshop on Computer Security Foundations*, 2004, pp. 280–291.

10. **De Raedt L., Van Laer W.** Inductive Constraint Logic. *Proc. of 6th International Conference Algorithmic Learning Theory,* Fukuoka, Japan, 1995, Springer, Lecture Notes in Computer Science, 1995, Vol. 997, pp. 80–94.

11. **Shilov N.V., Yi K.** How to find a coin: propositional program logics made easy. *In Current Trends in Theoretical Computer Science, World Scientific*, Vol. 2, 2004, pp. 181–213.

12. **Shilov N.V., Garanina N.O., and Choe K.-M.** Update and Abstraction in Model Checking of Knowledge and Branching Time. *Fundameta Informaticae*, 2006, Vol. 72, No. 1–3, pp. 347–361.

13. **Shilov N.V., Garanina N.O.** Combining Knowledge And Fixpoints. Preprint, Novosibirsk, 2002, No. 98, 50 p.

14. **Shilov N., Garanina N., Bodin E.** Multi-agent approach to a Dijkstra problem. *In Proc. of Workshop on Concurrency, Specification and Programming,* Helenenau, 2010. Humboldt-Universitat zu Berlin. Informatik-Bericht, No. 237, Vol. 1, pp. 73–84.

**GARANINA, Natalia O.** *Laboratory of Theoretical Programming, A.P. Ershov Institute of Informatics Systems, Siberian Branch of the Russian Academy of Sciences.*

630090, Acad. Lavrentjev Ave. 6, Novosibirsk, Russia.

E-mail: garanina@iis.nsk.su

**ГАРАНИНА Наталья Олеговна** – *научный сотрудник лаборатории теоретического программирования Института систем информатики имени А.П. Ершова Сибирского отделения РАН.*

630090, Россия, г. Новосибирск, пр. Акад. Лаврентьева, д. 6.

E-mail: garanina@iis.nsk.su