Formal Verification of Knowledge Based Programs

Carla Delgado, Mario Benevides and Michel Carlini

COPPE/Sistemas - Universidade Federal do Rio de Janeiro (UFRJ)
P.O. Box 68.511 - 21.945-970 - Rio de Janeiro - RJ - Brazil
{delgado, mario, mcar}@cos.ufrj.br

Abstract. In this work we investigate two approaches for representing and reasoning about knowledge evolution in *M*ulti-*A*gent *S*ystem (*MAS*), and presents a translation method from one formalism to another. We use a formal *L*anguage for *T*ime and *K*nowledge [7] as a specification language for the whole system, and for each agent, we use *K*nowledge *B*ased *P*rograms, [6]. After introducing both approaches, we propose a translation method to translate a representation in *KBP* into a set of *TKL* formulas. The translation method presented provides an strategy to model and verify *MAS* without being attached to local or global representations, giving the alternative to switch from one representation to another. In order to illustrate usefulness of the translation method two *MAS* are presented: the Muddy Children Puzzle and the Bit Exchange Protocol.

Keywords: formal verification, modal logic, knowledge based programs.

1 Introduction

When representing a *M*ulti-*A*gent *S*ystem (*MAS*) we have two main approaches. We can look at the system by means of agents' individual behavior or as a global unity reacting to the surrounding environment. In this work we investigate two methods and their relationship.

For the global approach, we use a formal *L*anguage for *T*ime and *K*nowledge, *TKL*, to specify the global behavior of *MAS*s . *TKL* is a logic language capable of representing knowledge evolution in a group of agents. A complete axiomatic characterization of these notions is given in [7].

For the individual agent's approach, we use *K*nowledge *B*ased *P*rograms (*KBP*) [6], which aims to describe the activities of a *MAS* considering agent's knowledge and abstracting from implementation details. In this approach a *MAS* is a collection of *KBP*s, one to each agent, that executes in accordance with knowledge tests applied to the agent's local state.

Using *TKL* formulas, it is possible to formally verify interesting global properties of the system, as deadlock, agreement, global stability, distributed knowledge and everybody's knowledge. On the other hand, there is a great variety of problems that can be represented by a KBP. The ability to build a *KBP* from a *MAS*' specification written in *TKL* would be interesting in order to model the behavior of each agent in the system as an autonomous process, and to show the relationship between knowledge and actions. Conversely, if we obtain an specification of a *KBP* expressed in *TKL* from the *KBP*s, we would get both a concise global representation of the whole system and a framework for checking global properties using theorem provers available [2].

In this paper we propose a set of rules to generate a specification in *TKL* formulas from a set of local *KBP*s composing a *MAS*. Each rule basically describes how to generate formulas in *TKL* from items of specific sections of the *KBP* system. The complete translation is obtained by applying some applicable rules of transformation to each *KPB* that belongs to the system. To formalize the translation method, we define these rules as functions from the *KBP*s items into sets of formulas written in *TKL*.

The methods of translation presented in this work offers an strategy to model and verify *MAS*s in two levels: local and global, allowing one to switch from one representation to another. Sections 2 and 3 present *TKL* and the *KBP* formalisms. Section 4, defines the translation methods with application examples. Section 5 sketches the proofs of theorems about important properties of the translation methods, and in section 6 we state our conclusions.

2 A Language for Knowledge and Time

In order to be able to reason about knowledge in a *MAS* it is necessary to assume that agents are capable of reasoning about the world and also about other agents' knowledge. As *MAS* is a dynamic entity, we need a representation of knowledge that captures the evolving knowledge spread out among agents. A complete axiomatic characterization of the notion of knowledge and common knowledge, and an accurate analysis of the role played by time in *MAS*' evolution was given in [7].

The language is a propositional multi-modal language. For each agent i, there is a modality K_i representing knowledge from agent's i point of view. Intuitively, the formula $K_i\varphi$ indicates that "agent i knows φ". For simplicity, the language may also include modality B_i, dual of K_i, which means that "agent i believes φ is true". We also have modal temporal operator $\bigcirc\varphi$, which means that φ is true at next moment. Other temporal modalities are $\Box$ (always) and *Until*. The modal operators for knowledge in a group of agents, Distributed Knowledge D_G, Everybody's Knowledge E_G and Common Knowledge C_G, are defined in the usual way [6].

A brief formal description for the *Temporal Knowledge Language* (*TKL*):

$\varphi ::= p| \perp |\neg\varphi|\varphi \wedge \psi|\varphi \vee \psi|\varphi \rightarrow \psi|K_i\varphi|B_i\varphi| \bigcirc \varphi|\varphi\, Until\, \psi|\Box\varphi$, where p ranges over the propositional symbols and i over the set of agents.

A model for *TKL* is a tuple $M = \langle S, \omega, l, R_1, R_2, \cdots\rangle$ where S is set of states or possible worlds (an infinite sequence of states will be called a *history*), $\omega \in S^N$ is the real history, l is a labelling function $l : S \rightarrow 2^P$, and for every agent i, R_i is an infinite sequence of equivalence relations, i. e. for any $k \in N$, on R_i^k is an equivalence relation on S_k^N, the set of all possible histories at instant k. The intuitive meaning for R_i^k is that any two histories ω and τ are equivalent with respect to i and k if the knowledge that agent i has gathered about the world and its history up to instant k is not enough for her to distinguish between ω and τ.

Formula φ is $True$ for a history σ and instant $k \in N$ at a model M:

1. $M, k, \sigma \models p$ iff $p \in l(\sigma_k)$;
2. $M, k, \sigma \models \neg\varphi$ iff not $M, k, \sigma \models \varphi$;
3. $M, k, \sigma \models \varphi \wedge \psi$ iff $M, k, \sigma \models \varphi$ and $M, k, \sigma \models \psi$;
4. $M, k, \sigma \models K_i\varphi$ iff $M, k, \tau \models \varphi$ for all histories τ such that $(\sigma, \tau) \in R_i^k$;
5. $M, k, \sigma \models B_i\varphi$ iff $M, k, \tau \models \varphi$ for some history τ such that $(\sigma, \tau) \in R_i^k$;
6. $M, k, \sigma \models \bigcirc\varphi$ iff $M, k+1, \sigma \models \varphi$;
7. $M, k, \sigma \models \Box\varphi$ iff for all $n \geq k$ $M, n, \sigma \models \varphi$;
8. $M, k, \sigma \models \varphi$ *Until* ψ iff for all $n \geq k$ such that $M, n, \sigma \not\models \varphi$ there is some m, $k \leq m \leq n$ such that $M, m, \sigma \models \psi$.

A complete axiomatization for this logic is presented in [7].

3 Knowledge-Based Programs

Most models for distributed systems suggest that each entity is a program that executes its actions in accordance with tests applied to agent's local state [8, 1]. Each test together with its associated actions is called a *guard*. *K*nowledge *B*ased *P*rograms were defined in [6] intending to describe the activities of a *MAS* considering agent's knowledge and abstracting from implementation details. This can be done including knowledge tests to agent's local state, together with conventional tests.

A knowledge test for agent i is a boolean combination of formulas of the form $K_i\varphi$ or $\neg K_i\varphi$, where φ can be an arbitrary formula that may include other modal operators, like common knowledge, other agents knowledge and temporal ones [6] (for simplification, the dual of K_i, B_i, may also be used in knowledge tests). The agent selects an action based on the result of applying the standard test to local state and knowledge test to knowledge state. Standard programs are a special cases of *KBP*s.

Standard Program	*KBP*
Program *Prog*$_i$	**Program** *KBP*$_i$
initial: *initial conditions*	**initial:** *initial conditions, initial knowledge*
repeat	**repeat**
case of	**case of**
(a) *test_1* **do** *action_1*	**(a)** *test_1* $\wedge$ *knowledge_test_1* **do** *action_1*
(b) *test_2* **do** *action_2*	**(b)** *test_2* $\wedge$ *knowledge_test_2* **do** *action_2*
...	
end case	**end case**
until *termination_test*	**until** *knowledge_termination_test*
end	**end**

*KBP*s provide a interesting level of abstraction to represent the relation between knowledge and actions. Depending on the granularity of knowledge and actions representation, a *KBP* may have different levels of abstraction. This is illustrated in the example below. The Muddy Children Puzzle offers an interesting study about the knowledge involved in a MAS. It consists of a system formed by n children and their father.

We start with n children playing together. The father tell them that they should not get dirty. It happens that some of the children get mud on their foreheads. Each child can see the mud on the others forehead, but not on his own. After some time, the father returns and says: "At least one of you has mud on your forehead". Then, he keeps asking the following question, over and over: "Does any of you know whether you have mud on your own forehead?" Suppose that all children are perceptive, intelligent, truthful, and they answer simultaneously. If there are k muddy children, in the first $k-1$ times, all children will answer 'NO', but in then the k^{th} time the muddy children will all answer 'YES'.

The behavior of each child on the Puzzle can be modelled as a *KBP*, where proposition childheard$_i$ stands for "Child i just heard father's question", and proposition p_i stands for "child i is muddy":

```
program $MC_i$        %% version 1
  initial: $K_i p_j$, for j={% set of muddy children agent i sees%}
           $K_i(\bigvee_{x=1..n} p_x)$ %% *At least one child is muddy*
  repeat
    case of
    (a) childheard$_i \wedge (K_i p_i \vee K_i \neg p_i)$ do say 'YES'
    (b) childheard$_i \wedge (\neg K_i p_i \wedge \neg K_i \neg p_i)$ do say 'NO'
    end case
  until $K_i p_i \vee K_i \neg p_i$
end
```

The *KBP* above is too abstract and not very interesting as far as knowledge representation is concerned. It does not represent knowledge evolution in time. We would like to have a representation of a *KBP* showing how pieces of knowledge are acquired and how an agent produces new consistent knowledge. Below, we present a more detailed *KBP* for the same puzzle.

```
program $MC_i$        %% version 2
  initial: $K_i p_j$, for j={% set of muddy children agent i sees%}
           $K_i(\bigvee_{x=1..n} p_x)$ %% At least one child is muddy
  repeat
    case of
    (a) $\text{initial}_i \land \text{childheard}_i \land K_i p_i$ do say 'YES'
    (b) $\text{initial}_i \land \text{childheard}_i \land \neg K_i p_i$ do say 'NO'
    (c) $\text{childheard\_yes}_i \land \neg K_i p_i$ do learn $K_i \neg p_i$
    (d) $\text{childheard}_i \land \neg\ \text{childheard\_yes}_i \land$
          $B_i(\neg p_i \land B_1(\neg p_1 \land \cdots \land B_{k-2}(\neg p_{k-2} \land B_{k-1}(\neg p_{k-1} \land K_k p_k)) \cdots)))$ do
            learn $B_i(\neg p_i \land B_1(\neg p_1 \land \cdots \land B_{k-2}(\neg p_{k-2} \land K_{k-1}(p_{k-1} \land K_k p_k)) \cdots)))$
            say 'NO'
    (e) $\text{childheard}_i \land \neg\ \text{childheard\_yes}_i \land B_i(\neg p_i \land K_j p_j)$, for $j <> i$ do
            learn $K_i p_i$
            say 'YES'
    end case
  until $K_i p_i \lor K_i \neg p_i$
end
```

Propositions childheard$_i$ and p_i have the same meaning of the previous example, and propositions initial$_i$ and childheard_yes$_i$ stands for "Program MC$_i$ has just begun" and "Child i heard answer Yes from any other child" respectively. The command *learn* denotes the action of acquiring a piece of knowledge. A detailed explanation of the *KBP* presented above can be found at [3]. It would be interesting to provide some explanation about the formula in guard (d), when the father asks if somebody already knows if there is mud on its forehead, the state of knowledge of each agent changes from:

$B_i(\neg p_i \land B_1(\neg p_1 \land B_2(\neg p_2 \land \cdots \land B_{k-2}(\neg p_{k-2} \land B_{k-1}(\neg p_{k-1} \land K_k p_k)) \cdots)))$
to: $B_i(\neg p_i \land B_1(\neg p_1 \land B_2(\neg p_2 \land \cdots \land B_{k-2}(\neg p_{k-2} \land K_{k-1}(p_{k-1} \land K_k p_k)) \cdots)))$
which means that as agents learn more and decrease their uncertainty, some of their believes are confirmed and turned into knowledge.

The latter version of the *KBP* presents much more information about the way agents reason about knowledge, abstracting from how messages are sent or received, how the local knowledge base of each agent is implemented, and other implementation details. Now, it is possible to understand how knowledge evolve on the system. The complete specification of the *MAS* can be obtained by the composition of all MC$_i$ programs.

4 Translating *KBP*s into *TKL* formulas

There is a great variety of problems that can be represented as a set of *KBP*s. It is an interesting tool to model the behavior of each agent in the system as an autonomous and local process, showing the relations between knowledge and actions. On the other hand, by producing an specification as a set of formulas written in *TKL*, we get a concise representation of the system and a framework for checking global properties such as deadlock detection, agreement, global stability, distributed and everybody's knowledge using theorem provers available

[2]. In this section we propose a set of rules to generate a specification in *TKL* from a set of local *KBP*s.

The translation process consists on applying some rules of transformation to each *KPB* that belongs to the system. Each rule basically describes how to generate formulas in *TKL* from specific items of the *KBP*. To formalize the translation method, we define these rules as functions from *KBP* items into a set of *TKL* formulas. First, we give an overview of the complete process.

Formulas at the initial section of a *KBP* are put in the set of formulas corresponding to the *MAS* specification. They represent facts that are true at the beginning of the system. If a formula is preceded by operator K_i, for some i, then it represents local knowledge for agent i, and does not interfere with other agents' knowledge. On the other hand, formulas that are not preceded by modal operator for knowledge represents global information.

Example: **Bit Sender KBP**

keeps sending a bit to it's partner, Bit Receiver KBP, until convinced the bit was correctly received

```
 program $BtS$
    initial: $K_s(bit = 0)$, assuming that zero is the initial value of the bit
             $K_s(bit = 0 \vee bit = 1)$
    repeat
       case of
       (a) $\neg K_s K_r(Bit)$[a] do sendbit
       end case
    until $K_s K_r(Bit)$
 end
```

Looking at initial section, we can straightly extract formulas $K_s(bit = 0)$ and $K_s(bit = 0 \vee bit = 1)$ for TKL specification

[a] the formula $K_r Bit$ stands for $K_r(bit = 0) \vee K_r(bit = 1)$

The guards of the *KBP* correspond to the preconditions of a section of actions (or commands). Actions affects the system and consequently other agents, so when representing state change rules in a *MAS*, we generate, for each guard of the program, a rule of the kind: *cause* $\rightarrow$ *consequence*; where *cause* corresponds to the precondition implied by the guard, and *consequence* to the effect of the actions associated to that guard. The test of a guard is already *TKL* formula, but actions have to be translated; it is necessary to identify the effect caused by each action and translate this effect into *TKL*.

Example: **Bit Sender KBP**
From guard (a) we identify: **cause**: $\neg K_s K_r(Bit)$ **consequence:** sendbit
cause is already a TKL formula, but the action sendbit needs translation.

Propositions, evaluated at knowledge tests on the *KBP*s, represent important information about state changes from the point of view of each agent. Actions are the only way to change truth value of propositions at knowledge tests. So, the execution of an action at a *KBP* is translated into a set of *TKL* formulas that change their values as consequence of action execution. Besides, actions in the *KBP* representing agent i can affect agent i itself or other agents in the system; actions that affect the agent itself have their results reflected at the right moment they are performed, but actions that affect other agents will only be perceived in the next turn of the system.

Example: **Bit Sender KBP**
Action sendbit is a communication action. When talking about a system where messages are safely delivered, it's consequence is that agent receiver will learn the bit value. As it is an action made by one agent that concerns another, it must be preceded by the temporal operator: $\neg K_s K_r(Bit) \rightarrow \bigcirc K_r(Bit)$

Finally, we tackle the translation of the knowledge termination test. A *KBP* finishes its execution when a termination condition is achieved, what corresponds to say that the agent being modelled by this *KBP* is not capable of doing any other action or perceiving anything else. Whenever all *KBP*s of the system has achieved its termination condition, it should finish its computation and whole system must halt. This can also be represented by a rule *cause* $\rightarrow$ *consequence*, where *cause* corresponds to the conjunction of all *KBP*s termination condition formulas ($\bigwedge_{\forall i}$ *knowledge_termination_test*$_i$), and *consequence* to the formula $\Box \perp$.

Example: **Bit Sender KBP**
Termination condition will be made from the conjunction of Bit Sender *KBP* termination test and the other *KBP*s involved, i. e., the Receiver's.
$K_s K_r(Bit) \wedge$ *knowledge_termination_test*$_{receiver}$ $\rightarrow \Box \perp$

This rules can be summarized as follows.

- Each formula at the initial section of each *KBP* becomes a formula of the *MAS* specification.
- For each guard (on each *KBP*) of the form: *test_i* $\wedge$ *knowledge_test_i* **do** *action*$_i^1$, *action*$_i^2$ $\cdots$, generate formula: *test_i* $\wedge$ *knowledge_test_i* $\rightarrow$ *effect_of_action*$_i^1$ $\wedge$ *effect_of_action*$_i^2$ $\cdots$
- For the termination conditions of all *KBP*s of the system, generate formula: $\bigwedge_{\forall i}$*knowledge_termination_test*$_i$ $\rightarrow \Box \perp$

The formula *effect_of_action*$_i^n$ represents the fact that becomes true as consequence of the action being performed. *effect_of_action*$_i^n$ is a *TKL* formula involving propositional operators and operators for individual knowledge, if the action affects only the agent who performed it; if the action affects other agents, it will turn into a formula preceded by temporal operator $\bigcirc$, due to the fact that it will only be perceived at the next turn of the system.

These rules can be formalized by a translation function Ψ_{MAS} that maps *KBP*s into a set of formulas written in *TKL*. The function $\Psi_{MAS} : P \rightarrow 2^S$, where P is a set of *KBP*s and 2^S is the power set of formulas of *TKL*. Ψ_{MAS} can be defined in terms of a subfunction ψ_{mas} from *KBP* items into formulas in *TKL*, based on the rules explained above.

For each KBP of the system:
ψ_{mas}: initial section translation
For each knowledge logic formula φ at initial section: $\psi_{mas}(\varphi) = \varphi$
ψ_{mas}: guards translation
For each guard G of the form: *test_i* $\wedge$ *knowledge_test_i* **do** $action_i^1$ $action_i^2$ $\cdots$
$\psi_{mas}(G) =$ *knowledge_test_i* $\rightarrow$
effect_of_action$_i^n$, if action$_i^n$ affects only agent i
$\bigcirc$*effect_of_action*$_i^n$, otherwise
(for all associated actions)
ψ_{mas}: termination condition translation
For knowledge logic formula *knowledge_termination_test* from all *KBP*s
$\psi_{mas}(knowledge\_termination\_test) = \bigwedge_{\forall i} knowledge\_termination\_test_i \rightarrow \Box \perp$

In section 3, we present two *KBP*s for the Muddy Children Puzzle. The first one is too abstract and the second one quite detailed. If we extract a specification from the former, it would be also too abstract and not very useful to reasoning about knowledge evolution. From the later, we can obtain a more detailed specification and prove interesting properties about how knowledge evolves at the *KBP*. In order to illustrate the use of translation function Ψ_{MAS} we apply it to the *KBP* of second example for the Muddy Children Puzzle, MC$_i$ version 2.

Applying Ψ_{MAS} to MC$_i$ version 2:

$\psi_{mas}(K_i p_j)$: $K_i p_j$, for j={% set of muddy children agent i sees%}
$\psi_{mas}(K_i(\bigvee_{x=1..n} p_x))$: $K_i(\bigvee_{x=1..n} p_x)$
$\psi_{mas}(a)$: $(\text{initial}_i \wedge \text{childheard}_i \wedge K_i p_i) \rightarrow \bigcirc \bigwedge_{j \in G} \text{childheard\_yes}_j$
$\psi_{mas}(b)$: $(\text{initial}_i \wedge \text{childheard}_i \wedge \neg K_i p_i) \rightarrow \bigcirc \bigwedge_{j \in G} \text{childheard\_no}_j$
$\psi_{mas}(c)$: $(\text{childheard\_yes}_i \wedge \neg K_i p_i) \rightarrow K_i \neg p_i$
$\psi_{mas}(d)$: $(\text{childheard}_i \wedge \neg\, \text{childheard\_yes}_i \wedge$
$B_i(\neg p_i \wedge B_1(\neg p_1 \wedge ... \wedge B_{k-2}(\neg p_{k-2} \wedge B_{k-1}(\neg p_{k-1} \wedge K_k p_k))...)))) \rightarrow$
$B_i(\neg p_i \wedge B_1(\neg p_1 \wedge ... \wedge B_{k-2}(\neg p_{k-2} \wedge K_{k-1}(p_{k-1} \wedge K_k p_k))...))) \wedge$
$\bigcirc \bigwedge_{j \in G} \text{childheard\_no}_j$
$\psi_{mas}(e)$: $(\text{childheard}_i \wedge \neg\, \text{childheard\_yes}_i \wedge B_i(\neg p_i \wedge K_j p_j)$, for $j <> i) \rightarrow$
$K_i p_i \wedge \bigcirc \bigwedge_{j \in G} \text{childheard\_yes}_j$
$\psi_{mas}(K_i p_i \vee K_i \neg p_i)$: $\bigwedge_{\forall i}(K_i p_i \vee K_i \neg p_i) \rightarrow \Box \perp$

With this specification in hands, we can prove some interesting properties:

For a group of n children where k have muddy foreheads:
For any turn $m \leq k$, no children hears any affirmative answer.

$$\bigwedge_i \underbrace{\bigcirc \bigcirc \cdots \bigcirc}_{m\ times} \neg childheard\_yes_i,\ \ for\ i = 1\ to\ n$$

After k turns if child j is dirty, she knows it.

$$p_j \rightarrow \underbrace{\bigcirc \bigcirc \cdots \bigcirc}_{k\ times} K_j p_j$$

What leads to an affirmative answer that will be heard soon, letting everyone aware about one's own forehead after the k^{th} turn.

$$\bigwedge_i \underbrace{\bigcirc \bigcirc \cdots \bigcirc}_{k+1\ times} childheard\_yes_i \wedge (K_i p_i \vee K_i \neg p_i),\ \ for\ i = 1\ to\ n$$

A child that is not dirty will only became aware of it after she hears an affirmative answer.

$$\neg p_i \rightarrow (\neg K_i(p_i \vee \neg p_i)\ until\ childheard\_yes_i)$$

5 Relation Between KBP and *TKL* formulas Specification

An important property of *MAS* specifications written in *TKL* obtained from a set of *KBP*s using the translation methods is stated by the following theorem:

Theorem 1. *Let S be a set of TKL formulas, obtained applying function Ψ_{MAS} to a KBP representing one agent in a MAS. All executions of P satisfies S.*

Proof. The proof is by induction on the structure of program P. All formulas in S are obtained applying a rule ψ_{mas} to a *KBP* item, so to preserve the validity implied by the execution of P.

6 Conclusions

In this work we investigate two different approaches for representing and reasoning about knowledge in a *MAS*. We use a formal *L*anguage for *T*ime and *K*nowledge [7] as specification language to represent global properties of the system, and for each individual agent, we use *K*nowledge *B*ased *P*rograms [6]. The latter is more suitable for modelling the behavior of each agent in the system as an autonomous process and observe how interaction takes place; the latter has the advantage of providing a concise representation of the whole system and the possibility of checking global properties using theorem provers available [2].

This paper proposes a translation method, which consists of a set of rules to generate a specification in *TKL* formulas from a set of local *KBP*s composing a *MAS*. The translation method presented provides an strategy to model and verify global properties of a *MAS* presented as a set of *KBP*s.

Two examples of *MAS* were presented in order to illustrate the translation method: the Muddy Children Puzzle and the Bit Exchange Protocol. The Muddy Children Puzzle illustrates the relations between knowledge and time in a group of agents, and the Bit Exchange Protocol evidences the role played by interaction in *MAS*s. This approach could be used to model many practical problems related to *MAS*s, as for example, the implementation of protocols of communication and games.

As future work, we could define another translation method to accomplish the inverse process, that is the construction of a set of *KBP*s from a set of formulas corresponding to a specification of a *MAS* written in *TKL*. But it seems to be a more complex translation, as *KBP*s have hidden semantic properties that are embedded in its structure.

References

1. Barbosa, V.: An Introduction to Distributed Algorithms. MIT Press (1996).
2. Dixon, C., Nalon, C., Fisher, M.: Tableaux for Temporal Logics of Knowledge: Synchronous Systems of Perfect Recall Or No Learning. Proceedings of 10^{th} International Symposium on Temporal Representation and Reasoning and Fourth International Conference on Temporal Logic. Cairns, Queensland, Australia(2003).
3. Delgado, C., Benevides, M.: Dynamic Knowledge Logics. Proceedings of Encontro Nacional de Inteligência Artificial (ENIA). Fortaleza, Brazil (2001)
4. Fagin, R., Vardi, M.: Knowledge and Implicit Knowledge in a Distributed Environment. In: Proceedings of the Conference on Theoretical Aspects of Reasoning about Knowledge, (1986) 187–206
5. Fagin, R., Halpern, J., Moses, Y. and others: Reasoning About Knowledge. MIT Press (1995).
6. Fagin, R., Halpern, J., Moses, Y. and others: Knowledge-Based Programs. Distributed Computing, Vol. 10(4), (1997) 199–225
7. Lehmann, D.: Knowledge, common knowledge, and related puzzles. In Proc. 3rd Ann. ACM Conf. on Principles of Distributed Computing, pages 62–67, 1984.
8. Lamport, L.: Time, clocks and the ordering of events in a distributed system. Communications of the ACM, Vol 21(7), (1978) 558–565.