Model-Checking Alternating-time Temporal Logic with Strategies Based on Common Knowledge Is Undecidable

Raluca Diaconu\textsuperscript{1,2} and Cătălin Dima\textsuperscript{2}

\textsuperscript{1} Department of Computer Science, “Al.I.Cuza” University of Iași, Iași 700506, Romania
\textsuperscript{2} LACL, Université Paris Est-Créteil, 61 av. du G-ral de Gaulle, 94010 Créteil, France

Abstract. We present a semantics for the Alternating-time Temporal Logic (ATL) with imperfect information, in which the participants in a coalition choose their strategies such that each agent’s choice is the same in all states that form the common knowledge for the coalition. We show that ATL with this semantics has an undecidable model-checking problem if the semantics is also synchronous and the agents have perfect recall.

1 Introduction

Alternating-time Temporal Logic (ATL) (Alur et al., 1998, 2002) is a modal logic that generalizes CTL. It is defined over concurrent game structures with one or more players. A formula $\langle A \rangle \phi$ in ATL expresses that a coalition $A$ of agents can cooperate to ensure that the formula $\phi$ holds.

Different semantics have been given for the cooperation modalities, depending on whether the knowledge that each agent has of the current state of the game is complete or not (i.e. semantics with complete, resp. incomplete information), whether agents can use knowledge of the past game states when deciding on their next move (i.e. semantics based on perfect, resp. imperfect recall), and whether the agents in the coalition are aware of their possibility to enforce their goal, or they must be told so – i.e. de re vs. de dicto semantics (Jamroga and van der Hoek, 2004). Other variants of the semantics of ATL are studied in e.g. (Schobbens, 2004; Jamroga and Agotnes, 2007), see also (Bulling et al., 2010) for a recent survey on the topic.

In this paper we study a variation of the semantics for ATL, in which agents in coalitions choose their strategies based on the common knowledge, available inside the coalition, of the system history. More precisely, when an agent chooses her next move, she uses not only the information she observes of the system history, but also the knowledge she has about the knowledge of the other agents in the coalition, and the knowledge she has about the knowledge the other agents have about their knowledge, etc. This is a variation of the ATL\textsubscript{iR} logic studied in (Schobbens, 2004), see also (Jamroga and van der Hoek, 2004; Bulling et al., 2010). However, in our semantics, all the agents in a coalition utilize the same information about the system state: the common knowledge, for that coalition, of the system state. This is where our semantics differs from the one in (Schobbens, 2004; Jamroga and van der Hoek, 2004; Bulling et al., 2010), where the information which is used by each agent in a coalition depends only on her individual knowledge of the system state, and thus might be different for two distinct agents in the same coalition. Our semantics is therefore closer to the one studied in (Dima et al., 2010), where the strategies used by the agents of a coalition are based on the distributed knowledge inside the coalition.

Group strategies based on common knowledge have the following property: after the strategy is fixed and shared between the agents in the coalition, at each subsequent time instant,
each agent knows exactly what is the sequence of actions that any other agent in the coalition has issued up to that instant. This means that if, e.g., for trust purposes, agent Alice wants to check whether agent Bob in the same coalition has applied the appropriate sequence of actions that he was supposed to, and this is checked by e.g. verifying Bob’s log, the access that Alice gets to Bob’s log does not give her any extra information about Bob’s local state at any moment, besides the information that Alice already had when the coalition was created and the joint strategy was fixed. If strategies based on individual knowledge were used within coalitions, then such a scenario would lead to information leak since e.g. when Alice gets informed about the actual sequence of actions taken by Bob, she might deduce some information about Bob’s initial local state, since Bob’s strategy might differ when applied in distinct states that are identically observed by Alice.

In this paper we only concentrate on the simplest setting of ATL with strategies based on common knowledge and do not model the trust part of the above scenario. The main result of our paper is that, for ATL with common-knowledge strategies, the model-checking problem is undecidable. Our proof is an adaptation of (van der Meyden, 1998; Shilov and Garanina, 2002), where it is shown that the model checking problem is undecidable for CTL with common knowledge, with a synchronous and perfect recall semantics, see also (van der Meyden and Shilov, 1999). Note that the results from the cited papers cannot be used directly since the common knowledge operator cannot be expressed in ATL with our semantics. The only connection is the following: for systems in which agents have a single choice in each state, \( \langle A \rangle \Box \phi \) is equivalent with \( C_A \Box \phi \), where \( C_A \) is the common knowledge operator for the group \( A \).

The rest of the paper is organized as follows: in the next section we recall the game arenas framework for interpreting ATL, and present the three variants of group strategies that can be used for interpretation the coalition operators: individual strategies, strategies based on distributed knowledge, and strategies based on common knowledge, and we give the relationship between the three types of strategies. In the third section we recall the semantics of ATL and introduce the new semantics based on group strategies with common knowledge. The fourth section contains our undecidability proof. We end with a section with conclusions.

2 Background

In this section we give some basic definitions and fix some notations. We recall here the notations for game arenas with emphasis on the equivalence relations. We present the three types of strategies: group strategies based on individual knowledge, group strategies based on distributed knowledge, and group strategies based on common knowledge, and give the relationship between them.

Definition 1. A game arena \( \Gamma = (Ag, Q, (Q_a)_{a \in Ag}, Q_e, (C_a)_{a \in Ag}, \delta, \lambda, Q_0) \) is a tuple where

- \( Ag \) is a finite set whose elements are called agents; subsets \( A \subseteq Ag \) are called coalitions.
- \( Q_a \) is a finite set of local states for agent \( a \), and \( Q_e \) is a finite set of environment states.
- \( Q = \bigotimes_{a \in Ag} Q_a \times Q_e \) is the set of global states.
- \( C_a \) is a finite set of actions available to agent \( a \). We denote by \( C_A \) the set of actions available for the coalition \( A \), \( C_A = \bigotimes_{a \in A} C_a \), and \( C = C_{Ag} \).
- \( \delta \subseteq Q \times C \times Q \) is the transition relation.
- \( Q_0 \subseteq Q \) is the set of initial states.
We write \( q \sim_a q' \) for a transition \( (q, c, q') \in \delta \). Also, given a global state \( q \), we denote \( q|_a \) its \( a \)-component and call it the \( a \)-projection of \( q \).

A run \( \rho \) is a (finite or infinite) sequence of transitions agreeing on intermediate states, i.e. \( \rho = (q_{i-1} \xrightarrow{c_i} q_i)_{1 \leq i \leq n} \) with \( n \in \mathbb{N} \) or \( \rho = (q_{i-1} \xrightarrow{c_i} q_i)_{1 \leq i \leq \infty} \). The length of a run \( \rho \), denoted \( |\rho| \), is the number of its transitions, \( \eta \), in the case of finite runs, or \( \infty \) for infinite runs. A run \( \rho \) is initialized if \( q_0 \in Q \).

We denote by \( \text{Runs}^f(\Gamma) \) the set of finite initialized runs and by \( \text{Runs}^\omega(\Gamma) \) the set of infinite initialized runs of \( \Gamma \). When the game arena is understood from the context, we only use the notations \( \text{Runs}^f \), resp. \( \text{Runs}^\omega \). \( \rho[i] \) denotes the state on \( i \)-th position in the run \( \rho \), and \( \rho[0\ldots i] \) denotes the prefix of \( \rho \) of length \( i \), for all \( i, 0 \leq i \leq |\rho| \). Note that, if \( \rho \in \text{Runs}^f(\Gamma) \) or \( \rho \in \text{Runs}^\omega(\Gamma) \), then we have \( \rho[0\ldots i] \in \text{Runs}^f(\Gamma) \), for all \( i, 0 \leq i < |\rho| \).

Two states are observationally equivalent for agent \( a \) if the same atomic propositions observable by agent \( a \) have the same truth values in both states. Denote this relation for states \( \sim_a \), \( \sim_a \subseteq Q \times Q \). Thus, formally,

\[
q \sim_a q' \text{ iff } q|_a = q'|_a
\]  

We extend the relation \( \sim_a \) for finite runs in the following way: given two runs of equal length, \( |\rho| = |\rho'| \),

\[
\rho \sim_a \rho' \text{ iff } \rho[i] \sim_a \rho'[i] \text{ for all } 1 \leq i \leq |\rho|.
\]  

The distributed knowledge equivalence relation is an extension of this relation to coalitions that are “willing to exchange” information about their local states. On states, the distributed knowledge relation inside the set of agents (coalition) \( A \) is defined as follows:

\[
q \approx_A q' \text{ iff } q \sim_a q' \text{ for all } a \in A.
\]  

This relation is then extended straightforwardly on runs as follows: given two runs of equal length, \( |\rho| = |\rho'| \),

\[
\rho \approx_A \rho' \text{ iff } \rho[i] \approx_A \rho'[i] \text{ for all } 1 \leq i \leq |\rho|.
\]  

The common knowledge equivalence relation on states is a different extension of the observability relations \( \sim_a \) to coalitions in which agents are “not willing to share” the information about their local state with the other agents in the coalition. This relation is denoted by \( \sim^C_A \) and is formally defined as follows:

\[
\sim^C_A = \left( \bigcup_{a \in A} \sim_a \right)^*.
\]  

Note that when \( \text{card}(Q/\sim^C_A) = \text{card}(Q) \) for all coalitions \( A \subseteq Ag \), the system is equivalent to a system with complete information.

\( \sim^C_A \) can be extended on runs in the following way:

\[
\rho \approx^C_A \rho' \text{ iff } \exists n \in \mathbb{N} \text{ s.t. } \exists \rho_0, \rho_1, \ldots, \rho_n \in \text{Runs}^\omega \text{ and } \exists a_1, \ldots, a_n \in A \\
\text{s.t. } \rho = \rho_0, \rho' = \rho_n, \rho_i \sim_{a_{i+1}} \rho_{i+1}, \forall 0 \leq i < n.
\]
Definition 2 ((Dima et al., 2010)). A strategy with distributed knowledge (or dk-strategy for short) for a set of agents (coalition) $A$ is any mapping $\sigma: (Q/\sim_A)^* \rightarrow C_A$.

An individual strategy for an agent $a$ is a strategy for the set of agents $\{a\}$.

We write $\Sigma_{dk}(A, \Gamma)$ for the set of all strategies of coalition $A$ in game arena $\Gamma$. Given a dk-strategy $\sigma \in \Sigma_{dk}(A, \Gamma)$, we define the projection on agent $a \in A$ of $\sigma$ as the mapping $\sigma^a_\Gamma: (Q/\sim_A)^* \rightarrow C_a$ with $\sigma^a_\Gamma(\alpha) = \sigma(\alpha)^a_\Gamma$ for all $\alpha \in (Q/\sim_A)^*$.

Furthermore, $\Sigma_{ind}(A, \Gamma)$ denotes the set of tuples of individual strategies for agents in $A$, i.e.

$$\Sigma_{ind}(A, \Gamma) = \{(\sigma_a)_{a \in A} | \sigma_a \in \Sigma_{dk}(\{a\}, \Gamma)\}.$$ 

In the sequel the term ind-strategy for a coalition $A$ denotes a tuple of individual strategies, one for each agent in $A$. In an ind-strategy, each agent plans his future actions based solely on his information about the current state, without any other interaction with the other agents or reference to the knowledge the other agents have about the current state.

Note that in a strategy with distributed knowledge for a coalition with at least two agents, the agents’ actions depend on the common observations of all agents in the coalition. Therefore, implementing such strategies requires some type of communication between agents. One of the suggested communication patterns in (Dima et al., 2010) is the use of an “arbiter” that gathers the information about the local states of each agent in the coalition, and then issues to each agent its action according to the dk-strategy under play. The more classical notion of coalition in e.g. (Jamroga and van der Hoek, 2004; Bulling et al., 2010) is to consider that a coalition acts together simply when each agent in the coalition follows an individual strategy for his play.

Definition 3. A strategy with common knowledge (or ck-strategy for short) for a set of agents $A$ is a mapping $\sigma: (Q/\sim_A^C)^* \rightarrow C_A$.

A ck-strategy maps each history of common knowledge observations for coalition $A$ to a tuple of actions for the given coalition. The intuition is that each agent chooses the same action for histories (i.e. sequences of observations) that might be different according to their individual knowledge, but are considered identical w.r.t. the common knowledge of the group $A$.

Similarly with the case of dk-strategies, we define the projection on agent $a \in A$ of a ck-strategy $\sigma \in \Sigma_{ck}(A, \Gamma)$ as the mapping $\sigma^a_\Gamma: (Q/\sim_A^C)^* \rightarrow C_a$ with $\sigma^a_\Gamma(\alpha) = \sigma(\alpha)^a_\Gamma$ for all $\alpha \in (Q/\sim_A^C)^*$.

A dk-strategy $\sigma$ is compatible with a run $\rho = (q_i \xrightarrow{c_{i+1}} q_{i+1})_{i \geq 0}$ if for all $i \leq |\rho|$, $\sigma(\rho[0..i]_{\sim A}) = c_{i+1}|A$.

Similarly, a ck-strategy $\sigma$ is compatible with $\rho$ if for all $i \leq |\rho|$, $\sigma(\rho[0..i]_{\sim A^C}) = c_{i+1}|A$.

Remark 1. In many practical situations, it is more convenient to define the local states of the agents by means of some subsets of atomic propositions $\Pi_a \subseteq \Pi$ which are assumed to be observable to agent $a$, subsets might not be pairwise disjoint. In this framework, the labeling of any two global states $q, q'$ whose $a$-projection is the same, i.e. $q|^a_\Gamma = q'|^a_\Gamma$, has the property that $\lambda(q) \cap \Pi_a = \lambda(q') \cap \Pi_a$. Then, for each global state $q$, we identify, by abuse of notation, the local state of $a$ in $q$ (i.e. $q|^a_\Gamma$) with $\lambda(q) \cap \Pi_a$. 

For a given set of agents $A \subseteq Ag$, we denote $\Pi_A = \bigcup_{a \in A} \Pi_a$. We also denote $\lambda_A : Q \rightarrow 2^{\Pi_A}$ the function defined by $\lambda_A(q) = \lambda(q) \cap \Pi_A$, and, by abuse of notation, we write $\lambda_a$ for $\lambda_{\{a\}}$.

Note also that, within this framework, a dk-strategy is a mapping $\sigma : (2^{\Pi_A})^* \rightarrow A$.

We will utilize this variant of game arenas in the third section of this work.

The following results follow directly from the definitions:

**Proposition 1.** The following strict inclusions hold:

$$\Sigma_{ck}(A, \Gamma) \not\subseteq \Sigma_{ind}(A, \Gamma) \not\subseteq \Sigma_{dk}(A, \Gamma)$$

### 3 Syntax and Semantics of ATL and $ATL^{prs}_C$

We recall here the syntax of ATL, and present the three variants of its semantics, corresponding to the utilization of ind-strategies, dk-strategies or ck-strategies for interpreting coalition operators.

The syntax of ATL is defined by the following grammar:

$$\phi ::= p \mid \phi \land \phi \mid \neg \phi \mid \llbracket A \rrbracket \circ \phi \mid \llbracket A \rrbracket \phi \cup \phi \mid \llbracket A \rrbracket \phi \wedge \phi$$

where $p \in \Pi$ and $A \in Ag$.

Formulas of the type $\llbracket A \rrbracket \phi$ read as “coalition $A$ can enforce $\phi$”. The operator $\llbracket \cdot \rrbracket$ is a path quantifier and $\circ$ (“next”), $\cup$ (“until”), $\wedge$ (“weak until”) are temporal operators.

The usual derived operators can be obtained as follows:

$$\llbracket A \rrbracket \Box \phi \equiv \llbracket A \rrbracket \phi \ \text{false} \quad \llbracket A \rrbracket \Diamond \phi \equiv \llbracket A \rrbracket \phi \ \text{true} \uparrow \phi$$

Formulas of the type $\llbracket A \rrbracket \phi$ reads as “coalition $A$ cannot avoid” $\phi$.

ATL is interpreted over concurrent game structures. Three different interpretations can be given, according to the possibility given to coalitions to utilize ind-strategies, dk-strategies or ck-strategies. We denote $\models_{ind}$, $\models_{dk}$ and $\models_{ck}$ the three variants of semantics.

Formally, given a game structure $\Gamma$, an infinite run $\rho \in \text{Runs}^\omega(\Gamma)$, a point $i \in \mathbb{N}$, we put:

1. $(\Gamma, \rho, i) \models_{a} p$ if $p \in \lambda(\rho[i])$ for any $a \in \{\text{ind}, \text{ck}, \text{dk}\}$.
2. $(\Gamma, \rho, i) \models_{a} \phi_1 \land \phi_2$ if $(\Gamma, \rho, i) \models_{a} \phi_1$ and $(\Gamma, \rho, i) \models_{a} \phi_2$, again for any $a \in \{\text{ind}, \text{ck}, \text{dk}\}$.
3. $(\Gamma, \rho, i) \models_{a} \neg \phi$ if $(\Gamma, \rho, i) \not\models_{a} \phi$, for any $a \in \{\text{ind}, \text{ck}, \text{dk}\}$.
4. $(\Gamma, \rho, i) \models_{\text{ind}} \llbracket A \rrbracket \circ \phi$ if there exists $\sigma \in \Sigma_{\text{ind}}(A, \Gamma)$ such that $(\Gamma, \rho', i+1) \models_{a} \phi$ for all runs $\rho'$ which are compatible with $\sigma$ and satisfy $\rho'[0 \cdots i] = \rho[0 \cdots i]$.
5. $(\Gamma, \rho, i) \models_{\text{ind}} \llbracket A \rrbracket \phi_1 \cup \phi_2$ if there exists $\sigma \in \Sigma_{\text{ind}}(A, \Gamma)$ such that for all runs $\rho'$ which are compatible with $\sigma$ and satisfy $\rho'[0 \cdots i] = \rho[0 \cdots i]$ there exists $j \geq i$ such that $(\Gamma, \rho', j) \models_{\text{ind}} \phi_2$ and $(\Gamma, \rho', k) \models_{\text{ind}} \phi_1$, for all $k, i \leq k \leq j - 1$.
6. $(\Gamma, \rho, i) \models_{\text{ind}} \llbracket A \rrbracket \phi_1 \wedge \phi_2$ if there exists $\sigma \in \Sigma_{\text{ind}}(A, \Gamma)$ such that for all runs $\rho'$ which are compatible with $\sigma$ and satisfy $\rho'[0 \cdots i] = \rho[0 \cdots i]$ there exists $j \geq i$ such that $(\Gamma, \rho', j) \models_{\text{ind}} \phi_2$ and $(\Gamma, \rho', k) \models_{\text{ind}} \phi_1$, for all $k, i \leq k \leq j - 1$, or $(\Gamma, \rho', j) \models_{\text{ind}} \phi_1$, for all $j \in \mathbb{N}, j \geq i$.

For the case of $\models_{ck}$, the semantics of the coalition operators must be rewritten as follows:

1. $(\Gamma, \rho, i) \models_{\text{ck}} \llbracket A \rrbracket \circ \phi$ if there exists $\sigma \in \Sigma_{\text{ck}}(A, \Gamma)$ such that $(\Gamma, \rho', i+1) \models_{\text{ck}} \phi$ for all runs $\rho'$ which are compatible with $\sigma$ and satisfy $\rho'[0 \cdots i] \sim_{A} \rho[0 \cdots i]$.
the configurations as runs in the game arena. The runs consist of states which encode the use temporal operators to describe an arbitrary long but finite computation. “guess” the amount of space required by the computation. Instead of asynchrony, we shall also (van der Meyden and Shilov, 1999). In (van der Meyden, 1998) asynchrony is used to

Theorem 1. The model checking problem for \( \text{ATL} \) is undecidable.

Proof. We reduce the Halting problem for Turing machines that start with an empty tape to the model checking problem for \( \text{ATL} \). The proof is structured as follows: first we present how to encode a Turing machine into a game arena, second we explain how the transitions are simulated in the model and then we prove the correctness of the theorem statement.

In the balance of the paper, \( \text{ATL} \) denotes the logic whose semantics is given by \( \equiv_{ck} \).

Finally, the \( \equiv_{dk} \) semantics is defined similarly with \( \equiv_{ck} \), with the difference that all references to \( \sim_A \) are replaced with references to \( \sim_A \), and each quantification over strategies is restricted to elements from \( \Sigma_{ck}(A, \Gamma) \).

We say that a formula \( \phi \) is satisfied in the game arena \( \Gamma \), written \( \Gamma \models_{\alpha} \phi \), if \( (\Gamma, \rho, 0) \models_{\alpha} \phi \) for all \( \rho \in \text{Runs}^{\omega}(\Gamma) \).

We note that the usual notation \( \text{ATL} \) denotes the logic in which the semantics is given by \( \equiv_{ind} \). Also, in (Dima et al., 2010), \( \text{ATL}_{prs} \) denotes the logic in which the semantics is given by \( \equiv_{dk} \). This is a modification the \( \text{ATL} \) logic with operators of the type \( \langle A \rangle \) from (Jamroga and van der Hoek, 2004), by ensuring that strategies are not only feasible when the coalition is created, but also during the whole existence of the coalition. A similar relationship can be observed between the semantics \( \equiv_{ck} \) and the operators of the type \( \langle A \rangle \) from (Jamroga and van der Hoek, 2004).

In this section we present our undecidability result for the model checking problem in \( \text{ATL} \). The problem statement is the following:

Problem 1. (The Model Checking Problem for \( \text{ATL} \)) Given a game arena \( \Gamma \) and an \( \text{ATL} \) formula \( \phi \), decide whether \( \Gamma \models_{\alpha} \phi \).

Theorem 1. The model checking problem for \( \text{ATL} \) is undecidable.

Proof. We reduce the Halting problem for Turing machines that start with an empty tape to the model checking problem for \( \text{ATL} \). The proof is structured as follows: first we present how to encode a Turing machine into a game arena, second we explain how the transitions are simulated in the model and then we prove the correctness of the theorem statement.

W.l.o.g. we restrict the halting problem to Turing machines satisfying the following properties:

1. they never write a blank symbol;
2. they have a single initial state, denoted \( q_0 \);
3. during a computation the machines never return to the initial state \( q_0 \);
4. they halt when they reach a final state, i.e. there are no transitions leaving a final state.

The construction is similar to (van der Meyden, 1998; Shilov and Garanina, 2002), see also (van der Meyden and Shilov, 1999). In (van der Meyden, 1998) asynchrony is used to "guess" the amount of space required by the computation. Instead of asynchrony, we shall use temporal operators to describe an arbitrary long but finite computation.

Intuitively, we encode the configurations of the Turing machine and the transitions between the configurations as runs in the game arena. The runs consist of states which encode the
contents of the tape cells and the positions of the R/W head. Some extra bits of information are needed, like the fact that some state represents a cell which is at the left (or the right) of the head position, or that some states represent the transition which is applied on a tape cell, encoding both the previous and/or the next tape symbol for that transition, and the direction where the head moves after taking the transition.

Also some states represent the left and right limit of the tape space. We use a special state marking the right limit as a guess of the amount of space that is needed for simulating a Turing machine which halts when starting with an empty tape.

The transitions between configurations of the Turing machine are also encoded as runs in the game arena. Then, the observability relations of one agent in the arena is utilized for connecting a run encoding a configuration with a run encoding a transition between configurations, which is then connected with the run encoding the next configuration with the aid of the observability relation of the second agent.

Finally, checking that the constructed game arena can simulate a halting run of the given Turing machine is done by checking the satisfiability of a reachability formula, saying that the two agents in the game cannot avoid (in the sense of choosing some strategy with common knowledge which applies to any identically observable history) the situation in which the Turing machine halts.

Formally, for a Turing machine $M = \{Q, \Gamma, \delta, \beta, \lambda, q_0, F_M\}$ we construct a game arena denoted $\Gamma = (Ag, Q, (Q_a)_{a \in Ag}, (C_a)_{a \in Ag}, \delta, \lambda, Q_0)$ for two agents, in which the set of global states is:

$$Q = \Sigma^M \cup \Sigma^M \cup \overline{\Sigma^M} \cup \overline{\Sigma^M} \cup \{\beta\} \cup (\Sigma^M \times Q^M) \cup \{\epsilon_L, \epsilon_R, \overline{\epsilon_L}, \overline{\epsilon_R}\}.$$  

(The local states will be defined by identifying which sets of the atomic propositions that can be seen by each agent, as in Remark 1 above.)

The sets of states $\Sigma^M$, $\overline{\Sigma^M}$, $\overline{\Sigma^M}$ and $\overline{\Sigma^M}$ are four copies of the set of states of $M$. The states in $\Sigma^M = \{s \mid s \in \Sigma^M, \beta\}$ correspond to the symbols preceding the head. The states in $\overline{\Sigma^M} = \{s \mid s \in \Sigma^M\}$ correspond to the symbols that follow the head. The blank symbol can never appear on the tape segment at the left of the head since we have only Turing machines that do not write blank symbols. We shall call the states in $\overline{\Sigma^M}$ overlined states. We shall use them to encode final configurations. The state $\beta$ is used to enforce the system to have an initial configuration. In a run this state will always follow the state $\langle \beta, q_0 \rangle$, where $q_0$ is the initial state. We use the tuples in $(\Sigma^M \times Q^M \times \Sigma^M \times \{\text{left, right}\} \times \{\text{prev, next}\})$ to encode transitions of the Turing machine. The labels left, right indicate whether the head will move to the left or, respectively, to the right, and the labels prev and next mark the previous position of the head, respectively the next position where the head is moving.

The symbols in the last line of (5) represent delimiters in order to assure that the machine does not move its head off the “guessed” space. In a run, the state $\epsilon_L$ will encode the left margin of the tape and the state $\epsilon_R$, the right margin. Likewise, $\overline{\epsilon_L}, \overline{\epsilon_R}$ and $\overline{\epsilon_L}, \overline{\epsilon_R}$ are delimiters corresponding to the initial and, respectively, to the final configuration.

The set of initial states is $Q_0 = \{\epsilon_L, \epsilon_R, \overline{\epsilon_L}\}$ and the set of actions consists of a singleton set $C_1 = C_2 = C = \{\text{act}\}$.

The transition relation is given by the following rules:
Transitions that encode a sequence of symbols on the machine’s tape are the following:

for all \( s, s' \in \Sigma^M \), \( s' \xrightarrow{\text{act}} s' \in \delta \), \( s \xrightarrow{\text{act}} \beta \in \delta \), \( \overline{s} \xrightarrow{\text{act}} \overline{s'} \in \delta \), \( \overline{s} \xrightarrow{\text{act}} \overline{s'} \in \delta \), \( \overline{\beta} \xrightarrow{\text{act}} \overline{\beta} \in \delta \).

A state in the set \( \Sigma^M \times Q^M \) is preceded by the states in \( \Sigma^M \) and followed by states in \( \Sigma^M \); this is modeled by the following transitions: for all \( q \in Q^M \cup \{q_0\} \), \( s' \xrightarrow{\text{act}} s', \beta \in \delta \), \( s', \beta \in \delta \).

The same rule applies for the final configuration: for all \( s, s' \in \Sigma^M \), \( s \neq \beta \), for all \( q_f \in F^M \), \( \overline{s} \xrightarrow{\text{act}} \overline{s'}, \beta \in \delta \), \( \overline{s'} \xrightarrow{\text{act}} \overline{s'} \in \delta \).

In a run encoding a transition we mark the position where the head was and the position where the head is moving. If the head moves to left then for all symbols \( s, s', s_n, s_p \in \Sigma^M \) and for all states \( q_p, q_n \in Q^M \setminus (F^M \cup \{q_0\}) \) for which \( (s_n, q_n, L) \in \delta_M(s_p, q_p) \) holds, \( s' \xrightarrow{\text{act}} (s_n, q_n, \text{left}, \text{next}) \in \delta \), \( s, q_n, \text{left}, \text{next} \xrightarrow{\text{act}} s, q_p, \text{left}, \text{prev} \in \delta \), \( s, q_p, \text{left}, \text{prev} \xrightarrow{\text{act}} s' \in \delta \).

When the head moves to right we have \( s' \xrightarrow{\text{act}} (s_p, q_p, \text{right}, \text{prev}) \in \delta \), \( s, q_p, \text{right}, \text{prev} \xrightarrow{\text{act}} s, q \in \delta \). In a run encoding a transition the next state will be the leftmost encoding the transition. \( \epsilon_L \xrightarrow{\text{act}} (s, q, \text{right}, \text{prev}) \in \delta \), \( \epsilon_L \xrightarrow{\text{act}} (s, q, \text{left}, \text{next}) \in \delta \).

A dual situation holds for the final state, \( \epsilon_R \): for all symbols \( s \in \Sigma^M \), and for all states \( q \in Q^M \), \( s \xrightarrow{\text{act}} \epsilon_R \in \delta \), \( s, q \xrightarrow{\text{act}} \epsilon_R \in \delta \), \( s, q, \text{right}, \text{next} \xrightarrow{\text{act}} \epsilon_R \in \delta \), \( s, q, \text{left}, \text{prev} \xrightarrow{\text{act}} \epsilon_R \in \delta \).

The same rules apply for the runs encoding the final configuration. In this situation the head must be on a final state: \( \overline{s} \xrightarrow{\text{act}} \overline{s} \in \delta \), \( \overline{s} \xrightarrow{\text{act}} \overline{s} \in \delta \), \( \epsilon_L \xrightarrow{\text{act}} (\beta, q_0) \in \delta \). In order to force the game arena to encode the initial configuration we distinguish between \( \beta \) and \( \overline{\beta} \). \( \beta \) will only follow an initial state. \( \beta, q_0 \xrightarrow{\text{act}} \beta \in \delta \), \( \beta \xrightarrow{\text{act}} \beta \in \delta \), \( \beta \xrightarrow{\text{act}} \epsilon_R \in \delta \).

Once the final state is reached it is never left. \( \epsilon_R \xrightarrow{\text{act}} \epsilon_R \), \( \epsilon_R \xrightarrow{\text{act}} \epsilon_R \), \( \epsilon_R \xrightarrow{\text{act}} \epsilon_R \).

Finally, nothing else belongs to \( \delta \).

The set of atomic propositions is

\[ \Pi = \{ p_1, p_2, p_3 \} \cup \{ p_s, p_{s_1}, p_{q_1}, p_{s_2}, p_{q_2} \mid s \in \Sigma^M, q \in Q \} \].

The valuation function is defined by the following rules:

\[ p_1 \in \lambda_1(x), p_1 \in \lambda_2(x) \text{ for all } x \in \{ \beta, \epsilon_L, \epsilon_R \} \]  \hspace{1cm} (6)

\[ p_2 \in \lambda_1(x), p_2 \in \lambda_2(x) \text{ for all } x \in \{ \overline{s}, \overline{s}, \{ s, q \}, \overline{s}, \epsilon_L, \epsilon_R \mid s \in \Sigma^M, q \in Q^M \} \]  \hspace{1cm} (7)

\[ p_3 \in \lambda_1(\epsilon_R), p_1 \in \lambda_2(\epsilon_R) \]  \hspace{1cm} (8)
Propositions \( p_1, p_2, p_3 \in \Pi \) are interpreted to \( false \) in any other state.

The extra propositions are used for clarifying the observability relation for each agent, namely, for all \( s \in \Sigma^M \) and \( q \in Q^M \):

- \( p_{s,1} \in \lambda_1(\varsigma) \), for all \\
  \( \varsigma \in \{ (s, s') \mid s \in \Sigma^M \} \cup \{ (s, q, \text{left, next}) \mid s', s'' \in \Sigma^M, q' \in Q^M, (s', q, L) \in \delta^M(s', q') \} \cup \{ (s, q, \text{right, next}) \mid s', s'' \in \Sigma^M, q' \in Q^M, (s', q, R) \in \delta^M(s', q') \} \cup \{ (s, q, \text{right, prev}) \mid s' \in \Sigma^M, q' \in Q^M, (s', q', R) \in \delta^M(s, q) \} \cup \{ (s, q, \text{left, prev}) \mid s' \in \Sigma^M, q' \in Q^M, (s', q', L) \in \delta^M(s, q) \} ;

- \( p_{q,1} \in \lambda_2(\varsigma) \), for all \\
  \( \varsigma \in \{ (s, q) \mid s \in \Sigma^M \} \cup \{ (s, q, \text{right, prev}) \mid s' \in \Sigma^M, q' \in Q^M, (s', q', R) \in \delta^M(s, q) \} \cup \{ (s, q, \text{left, prev}) \mid s' \in \Sigma^M, q' \in Q^M, (s', q', L) \in \delta^M(s, q) \} ;

- \( p_{s,2} \in \lambda_2(\varsigma) \), for all \\
  \( \varsigma \in \{ (s, q) \mid s \in \Sigma^M \} \cup \{ (s, q, \text{left, next}) \mid s' \in \Sigma^M, q' \in Q^M, (s, q, L) \in \delta^M(s', q') \} \cup \{ (s, q, \text{right, next}) \mid s' \in \Sigma^M, q' \in Q^M, (s, q, R) \in \delta^M(s', q') \} \cup \{ (s, q, \text{right, prev}) \mid q \in Q^M, (s, q, R) \in \delta^M(s', q) \} \cup \{ (s, q, \text{left, prev}) \mid q \in Q^M, (s, q, L) \in \delta^M(s, q) \} ;

- \( p_{q,2} \in \lambda_2(\varsigma) \), for all \\
  \( \varsigma \in \{ (s, q) \mid s \in \Sigma^M \} \cup \{ (s, q, \text{right, next}) \mid s' \in \Sigma^M, q' \in Q^M, (s, q, R) \in \delta^M(s', q') \} \cup \{ (s, q, \text{left, next}) \mid s' \in \Sigma^M, q' \in Q^M, (s, q, L) \in \delta^M(s', q') \} ;

- \( p_\beta \in \lambda_i(\beta) \cap \lambda_i(\hat{\beta}) \) for both \( i = 1, 2 \).

As above, the extra propositions are interpreted as \( false \) in any other state.

Figures 1, 2 and 3 offer a graphical representation of the encoding of a Turing machine. In Figure 1 we represent the part of the game arena accepting runs which encode the initial configuration. The automaton in Figure 2 accepts runs which encode intermediate configurations and, respectively, in Figure 3, final configurations. Any run \( \rho \in \text{Runs}^I(\Gamma) \cup \text{Runs}^s(\Gamma) \) is uniquely accepted by one of the three parts of the game arena.

\[ \text{Fig. 1. The initial configuration} \]

Next we describe the way configurations of the Turing machine are encoded as runs in the game arena. We denote a configuration of the Turing machine as a word over the alphabet \( \Sigma^M \cup (\Sigma^M \times Q^M) \) where a character of \( \Sigma^M \times Q^M \) appears only once. Hence the set of configurations is \( ((\Sigma^M \setminus \{ \beta \})^* \cdot (\Sigma^M \times Q^M) \cdot (\Sigma^M)^*) \).

In the game arena we have two types of run: runs that encode configurations and the runs that encode transitions.
A configuration \( cnfg = s_1 \cdots s_{i-1} (s_i, q) s_{i+1} \cdots s_n \) is encoded by a run \( \rho = \epsilon_L s_1^* \cdots s_{i-1}^* (s_i, q) s_{i+1}^* \cdots s_n^* \beta^m \epsilon_R \). There are infinitely many runs that encode one configuration, in this case, all runs that start with the state sequence \( \epsilon_L s_1^* \cdots s_{i-1}^* (s_i, q) s_{i+1}^* \cdots s_n^* \beta^m \epsilon_R \). An initial configuration is encoded by all runs that start with \( \epsilon_L (\beta, q^0) \beta^m \epsilon_R \). Respectively, a final configuration \( cnfg_f = s_1 \cdots s_{k-1} /uni27E8.al⟪s_k, q_f /uni27E9.al⟫1 s_{k+1} \cdots s_n \) is encoded by all the runs starting with the overlined state sequence \( \epsilon_L s_1^* \cdots s_{k-1}^* (s_k, q) s_{k+1}^* \cdots s_n^* \beta^m \epsilon_R \), for \( m \in \mathbb{N} \), \( m \neq \infty \).

The other type of run that may occur in \( \Gamma \) encodes transitions. For two configurations

\[
\begin{align*}
\text{cnfg} &= s_1 \cdots s_{k-1} (s_k, q) s_{k+1} s_{k+2} \cdots s_n \quad \text{and} \\
\text{cnfg}' &= s_1 \cdots s_{k-1} s_k' (s_{k+1}, q') s_{k+2} \cdots s_n
\end{align*}
\]

connected by a transition in which the R/W head of the Turing machine goes right, that is, \( (s_k', q', R) \in \delta^M(s_k, q) \) for some \( s_k' \in Q^M, q' \in \Sigma^M \), the transition coding run associated to the
transition \( cnfg \vdash cnfg' \) is the following run in \( \Gamma \):

\[
\epsilon_L s_1^* \cdots s_{k-1} (s_k, q, right, prev) (s_{k+1}, q', right, next) s_{k+2} \cdots s_n \epsilon_R.
\]

Similarly, when \( cnfg' = s_1 \cdots s_{k-2} (s_{k-1}, q') s'_k s_{k+1} \cdots s_n \) and \( \exists s'_k, q' \) for \( (s'_k, q', L) \in \delta^M(s_k, q) \), the transition coding run for the transition \( cnfg \vdash cnfg' \) is defined as follows:

\[
\epsilon_L s_1^* \cdots s_{k-2} (s_{k-1}, q', left, next) (s_k, q, left, prev) s_{k+1} \cdots s_n \epsilon_R.
\]

Before showing the proof for correctness we give some helpful properties of the system presented above.

Considering the valuation function, the following properties hold in our model:

**Proposition 2.** If \( \rho_1 \) and \( \rho_2 \) are two runs in the game arena \( \Gamma \), then \( \rho_1 \sim \rho_2 \) iff \( \rho_1 = \rho_2 \) or \( \rho_1 \) encodes a configuration \( cnfg \) and \( \rho_2 \) encodes a transition coding run \( cnfg \vdash cnfg' \).

**Proof.** The implication from right to left follows directly from the notation of transition coding runs and the definition of the labeling function \( \lambda_1 \). In order to prove the implication from left to right, we shall verify whether we have another run different from the mentioned ones, \( \sim \) equivalent with \( \rho_1 \).

Consider \( \rho_1 = \epsilon_L s_1^* \cdots s_{i-1} (s_i, q_0) s_{i+1} \cdots s_n \epsilon_R \) and \( \rho_2 = x_L x_1 \cdots x_n x_R \). Consider \( (s'_1, q_n, L) \in \delta^M(s_i, q_0) \). The case when the head moves to right is similar. Suppose that \( \rho_1 \sim \rho_2 \), \( \rho_1 \neq \rho_2 \) and \( \rho_1 \) encodes a configuration \( cnfg \) and there is no configuration \( cnfg' \) in the Turing machine \( M \) such that \( \rho_2 = (cnfg \vdash cnfg') \).

We consider only initialized runs, thus the initial state \( x_1 \in \{\epsilon_L, \overline{\epsilon_L}, \epsilon_L\} \). Since a run contains \( \epsilon_L \) followed by \( (q, q_0) \) and \( \overline{\epsilon_L} \) followed by \( (s, q_f) \), we have \( x_L = \epsilon_L \). Following the same idea, we observe that \( x_R = \epsilon_R \) for a finite configuration. Using the definition of \( \lambda_1 \) and \( \lambda_2 \) and the fact that \( (s_j, q', left, next) \) acts \( x \) if and only if \( x = (s_{j+1}, q'', left, prev) \), it follows that for all \( 1 \leq j \leq i-2 \) we have \( x_j = s_j^* \). Hence we have \( x_j = s_j^* \) for \( j, i+1 \leq j \leq n \).

Furthermore, we have that \( \lambda_1(s_{i-1}^*) = \{p_{s_{i-1}}\} = \lambda_1(s) \) only for \( s \in \{s_{i-1}, \overline{s_{i-1}}, s_{i-1}, \overline{s_{i-1}}\} \) \( \cup \{s_{i-1}, q, left, next\} | s', s'' \in \Sigma^M, q', q'' \in Q^M, s.t. (s'', q, L) \in \delta^M(s') \) \( \cup \{s_{i-1}, q, right, next\} | s', s'' \in \Sigma^M, q' \in Q^M, s.t. (s'', q, R) \in \delta^M(s') \}. We can eliminate the overlined states because an overlined state follows only another overlined state or a state containing \( q_f, q \in F^M \). We also eliminate the states of the type \( (s_{i-1}, q, right, next) \) because this states follow only \( (s', q', right, prev) \), for \( s', s'' \in \Sigma^M, q', q'' \in Q^M, (s'', q, R) \in \delta^M(s', q') \). Now, since

\[
\overline{s_{i-1}} \xrightarrow{act} (s_i, q) \text{ and } (s_{i-1}, q', left, next) \xrightarrow{act} (s_i, q, left, prev) \text{ belong to } \delta,
\]

we have either \( x_{i-1} = \overline{s_{i-1}} \) and \( x_i = (s_i, q) \), or \( x_{i-1} = (s_{i-1}, q', left, next) \) and \( x_i = (s_i, q, left, prev) \).

Because none of the observationally equivalent states can form a correct initialized run, we have proved that \( \rho_1 \sim \rho_2 \) implies \( \rho_1 = \rho_2 \) or \( \rho_1 \) encodes a configuration \( cnfg \) and \( \rho_2 \) encodes a transition coding run \( cnfg \vdash cnfg' \).

**Proposition 3.** If \( \rho_1 \) and \( \rho_2 \) are two runs in the game arena \( \Gamma \), then \( \rho_1 \sim \rho_2 \) iff \( \rho_1 = \rho_2 \) or \( \rho_1 \) encodes a configuration \( cnfg \) and \( \rho_2 \) encodes a transition coding run \( cnfg' \vdash cnfg' \).

**Proof.** The proof is similar to the one used for Proposition 2. \( \square \)

Putting together Proposition 2 and Proposition 3 we obtain the following property:
Proposition 4. For each computation in the Turing machine $M$, $cnfg \Rightarrow_M cnfg'$, where $cnfg$ and $cnfg'$ are instantaneous configurations of $M$, iff there exist $\rho, \rho'$ runs in the game arena $\Gamma$ that encode $cnfg$ and, respectively, $cnfg'$, such that $|\rho| = |\rho'|$ and $\rho \sim^C_{\{1,2\}} \rho'$.

Proof. The proof is constructed by following a series of equivalences. In the Turing machine $M$ having the computation $cnfg \Rightarrow_M cnfg'$ means that there exist a natural number $m$, the length of the computation, and $m$ configurations denoted $cnfg_0, cnfg_1, \ldots, cnfg_m$ such that $cnfg = cnfg_0 \Rightarrow_M cnfg_1 \Rightarrow_M \cdots \Rightarrow_M cnfg_m = cnfg'$. We write this $cnfg_0 \Rightarrow_M cnfg_m = (cnfg_i \Rightarrow_M cnfg_{i+1})_{0 \leq i \leq m}$.

From Proposition 2 and Proposition 3 it follows that for two instantaneous configurations of the Turing machine $M$, $cnfg_i$ and $cnfg_{i+1}$, we have $cnfg_i \Rightarrow_M cnfg_{i+1}$ iff there exist $\rho_i, \rho_{i+1}$ two runs that encode $cnfg_i$ and, respectively, $cnfg_{i+1}$ in the game arena $\Gamma$ such that $|\rho_i| = |\rho_{i+1}| \geq |cnfg_i|$ and $\rho_i \sim \rho_{i+1}$, for all $i$ natural numbers. By fixing the length of the runs and configurations to the “guessed” value or the length of $cnfg'$ we can write that $(\rho_i \sim (cnfg_i \sim cnfg_{i+1}) \sim_2 \rho_{i+1}, f_{\forall i \leq m})$. According to our definition of $\lambda_1$ and $\lambda_2$, we have now $\rho_i \sim^C_{\{1,2\}} \rho_{i+1}$.

Note that the reflexive transitive closure of $\Rightarrow_M$ leads to the complete proof. Hence, $cnfg \Rightarrow_M cnfg'$ iff $|\rho| = |\rho'|$ and $\rho \sim^C_{\{1,2\}} \rho'$.

We now turn to the proof of the theorem. We prove in the rest of this section that $(\Gamma, \rho, 0) \models \bigwedge_{C_{\{1,2\}}} (p_1 \land p_3 \land [1,2] \Box p_2)$ iff the Turing machine $M$ halts when starting with the empty tape.

In order to prove the implication from left to right we state that starting with the empty tape, machine $M$ halts. This means that there exists $n \in \mathbb{N}$, there exists $j \in \mathbb{N}, 1 \leq i \leq n$ and there exists a finite sequence of configurations $cnfg_0 \Rightarrow_M cnfg_f$, where $cnfg_0 = (\beta, q_0) \beta^{n-1}$ is an initial configuration, $cnfg_f = s_1 \cdots s_{k-1} (s_k, q_f)s_{k+1} \cdots s_n$ is a final configuration for some $s_1, \ldots, s_{k-1} \in \Sigma^M \setminus \{\beta\}$ and $s_{k+1}, \ldots, s_n \in \Sigma^M$, $0 \leq k \leq n$. Consider the length of the computation to be $f, f \in \mathbb{N}$. By Proposition 4 we can also consider that in the game arena $\Gamma$ there exists a finite sequence of runs $\rho_0, \rho_1, \ldots, \rho_f$ and $\hat{\rho_0}, \hat{\rho_1}, \ldots, \hat{\rho}_{f-1}$ such that

\[
\rho_0 \sim \rho_0 \sim_2 \rho_1 \sim_1 \rho_1 \sim_2 \cdots \rho_1 \sim_1 \rho_f \sim_2 \rho_f
\]

and $\hat{\rho_0} = \epsilon L (\beta, \rho_0) \beta^{n-1} \epsilon R$ encodes the initial configuration $\rho_i = \epsilon L s_i 1 \cdots s_{i-1} (s_j, q) s_{j+1} \cdots s_n \epsilon R$ encodes the $i$-th configuration $cnfg_i$, for all $i, 1 \leq i \leq f$ and for some $j, 1 \leq j \leq n$.

\[
\rho_f = \epsilon L s_1 1 \cdots s_{f-1} (s_l, q) s_{l+1} \cdots s_n \epsilon R,
\]

encodes the final configuration $(\Gamma, \rho_0, i) \equiv_{ck} p_1, (\Gamma, \rho_0, n) \equiv_{ck} p_3, (\Gamma, \rho_f, i) \equiv_{ck} p_2, \ \forall i, 1 \leq i \leq n$.

We consider a game arena with a single action. Therefore there is only one corresponding final run $\rho_f$, $\sim^C_{\{1,2\}}$ reachable from $\rho_0$, and, furthermore, the proposition $p_3$ is visible in the $n$-th state, of $\rho_f$. We therefore conclude that the $ATL^{prov}_C$ formula $[1,2] \Diamond (p_1 \land p_3 \land [1,2] \Box p_2)$ holds in the game arena $\Gamma$.

In order to prove the implication from right to left we show that for any run satisfying the $ATL^{prov}_C$ formula $(\Gamma, \rho, 0) \equiv_{ck} [1,2] \Diamond (p_1 \land p_3 \land [1,2] \Box p_2)$, there exists an initial run $\sim^C_{\{1,2\}}$-equivalent with $\rho$. 

\[
(\Gamma, \rho, 0) \equiv_{ck} [1,2] \Diamond (p_1 \land p_3 \land [1,2] \Box p_2)
\]
If \( (I', \rho, 0) \models_{c_k} [1, 2] \Diamond (p_1 \land p_3 \land [1, 2] \Box p_2) \) then in the game arena there exists \( \rho' \) a run and there exist \( n \in \mathbb{N} \) and \( i \in \mathbb{N}, 1 \leq i \leq n \) such that

\[
\rho = \hat{\epsilon}_L \langle \beta, q_0 \rangle \hat{\beta}^n \epsilon_R \\
\rho' = \hat{\epsilon}_L s_1 \cdots s_{i-1} \langle s_i, q \rangle s_{i+1} \cdots s_n \epsilon_R
\]

This means that there exists \( k \in \mathbb{N} \), a sequence of \( k \) runs \( \rho_0, \ldots, \rho_k \) such that \( \rho = \rho_0 \sim_{a_1} \hat{\rho}_1 \sim_{a_2} \ldots \sim_{a_2} \hat{\rho}_2 \sim_{a_1} \rho_k = \rho' \).

Following the Proposition 3 in the Turing machine \( M \) there exists a sequence of configurations corresponding to the runs \( \rho_1, \rho_2 \ldots \rho_k \) such that

\[
\text{cnfg}_0 \Rightarrow_M \text{cnfg}_1 \Rightarrow_M \cdots \Rightarrow_M \text{cnfg}_k.
\]

where \( \text{cnfg}_0 \) is an initial configuration and

\[
\text{cnfg}_k = s_1 \cdots s_{i-1} \langle s_i, q_f \rangle s_{i+1} \cdots s_n,
\]

for some \( i \), is a final configuration.

Hence, the machine \( M \) halts when starting with an empty tape. \( \square \)

5 Conclusion

We have presented a semantics for the coalition operators of \( ATL \) in which the agents utilize, in their strategies, the same action in states that lie in the same common knowledge observability for the coalition in which they participate. We have shown that the model checking problem for is undecidable for \( ATL^{prs}_{c_k} \), a result inspired from the techniques used in (Fugn et al., 2004; van der Meyden, 1998; van der Meyden and Shilov, 1999; Shilov and Garanina, 2002).

It would be interesting to investigate the possibility to generalize the decidability results on model-checking temporal epistemic logics with common knowledge (van Benthem and Pacuit, 2006; Lomuscio and Penczek, 2007) to the setting presented here.
Bibliography


