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

Raluca Diaconu[1,2] and Cătălin Dima[2]

[1] Department of Computer Science, "Al.I.Cuza" University of Iaşi, Iaşi 700506, Romania
[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 coaltion. 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 $\langle\!\langle A \rangle\!\rangle \phi$ in ATL expresses that a *coalition* A of *agents* can cooperate to ensure that the formula ϕ 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 coaltion 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_{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 upto 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, $\langle\!\langle A\rangle\!\rangle \Box \phi$ 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 interpretion 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 groupe 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.*
- *$Q = \bigtimes_{a\in Ag} Q_a \times Q_e$ 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, $C_A = \bigtimes_{a\in A} C_a$, 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.*

– *Π is the set of atomic propositions and $\lambda : Q \longrightarrow 2^{\Pi}$ is the state-labeling function, associating with each state $q \in Q$ the set of atomic propositions which hold in q.*

We write $q \xrightarrow{c} q'$ for a transition $(q, c, q') \in \delta$. Also, given a global state q, we denote $q\big|_a$ its a-component and call it the *a-projection* of q.

A run ρ is a (finite or infinite) sequence of transitions agreeing on intermediate states, i.e. $\rho = \left(q_{i-1} \xrightarrow{c_i} q_i\right)_{1 \le i \le \eta}$ with $\eta \in \mathbb{N}$ or $\rho = \left(q_{i-1} \xrightarrow{c_i} q_i\right)_{1 \le i < \infty}$. The *length* of a run ρ, denoted $|\rho|$, is the number of its transitions, η, in the case of finite runs, or ∞ for infinite runs. A run ρ is *initialized* if $q_0 \in Q_0$.

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

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,

$$q \sim_a q' \text{ iff } q\big|_a = q'\big|_a \tag{1}$$

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

$$\rho \sim_a \rho' \text{ iff } \rho[i] \sim_a \rho'[i] \text{ for all } 1 \le i \le |\rho|. \tag{2}$$

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:

$$q \sim_A q' \text{ iff } q \sim_a q' \text{ for all } a \in A.$$

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

$$\rho \sim_A \rho' \text{ iff } \rho[i] \sim_A \rho'[i] \text{ for all } 1 \le i \le |\rho|.$$

The *common knowledge equivalence relation* on states is a different extension of the observability reltions $\sim_a$ 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 $\sim_A^C$ and is formally defined as follows:

$$\sim_A^C = \Big(\bigcup_{a \in A} \sim_a \Big)^*. \tag{3}$$

Note that when $card(Q/\sim_A^C) = card(Q)$ for all coalitions $A \subseteq Ag$, the system is equivalent to a system with complete information.

$\sim_A^C$ can be extended on runs in the following way:

$$\rho \sim_A^C \rho' \text{ iff } \exists n \in \mathbb{N} \text{ s.t. } \exists \rho_0, \rho_1, \cdots, \rho_n \in Runs^\omega \text{ and } \exists a_1, \cdots, a_n \in A \text{ s.t. } \rho = \rho_0, \rho' = \rho_n, \rho_i \sim_{a_{i+1}} \rho_{i+1}, \forall 0 \le i < n. \tag{4}$$

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 $\sigma : \big(Q/\sim_A\big)^* \longrightarrow C_A$.*

*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 Γ. Given a dk-strategy $\sigma \in \Sigma_{dk}(A,\Gamma)$, we define the *projection* on agent $a \in A$ of σ as the mapping $\sigma\big\lfloor_a : \big(Q/\sim_A\big)^* \to C_a$ with $\sigma\big\lfloor_a(\alpha) = \sigma(\alpha)\big\lfloor_a$ for all $\alpha \in \big(Q/\sim_A\big)^*$.

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

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

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 system 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 $\sigma : \big(Q/\sim^C_A\big)^* \to C_A$.*

We write $\Sigma_{ck}(A,\Gamma)$ for the set of all strategies of coalition A in game arena Γ.

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 $\sigma\big\lfloor_a : \big(Q/\sim^C_A\big)^* \to C_a$ with $\sigma\big\lfloor_a(\alpha) = \sigma(\alpha)\big\lfloor_a$ for all $\alpha \in \big(Q/\sim^C_A\big)^*$.

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

$$\sigma(\rho[0\cdots i]_{\sim_A}) = c_{i+1}|_A.$$

Similarly, a ck-strategy σ is compatible with ρ if for all $i \leq |\rho|$,

$$\sigma(\rho[0\cdots i]_{\sim^C_A}) = c_{i+1}|_A.$$

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. $q\big\lfloor_a = q'\big\lfloor_a$, 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. $q\big\lfloor_a$) 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 $\lambda_A : Q \longrightarrow 2^{\Pi_A}$ the function defined by $\lambda_A(q) = \lambda(q) \cap \Pi_A$, and, by abuse of notation, we write λ_a for $\lambda_{\{a\}}$. Note also that, within this framework, a dk-strategy is a mapping $\sigma : (2^{\Pi_A})^* \longrightarrow C_A$.

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

The following resuls follows directly from the definitions:

Proposition 1. *The following strict inclusions hold:*

$$\Sigma_{ck}(A, \Gamma) \subsetneq \Sigma_{ind}(A, \Gamma) \subsetneq \Sigma_{dk}(A, \Gamma)$$

3 Syntax and Semantics of ATL and ATL_C^{prs}

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:

$$\phi ::= p \mid \phi \wedge \phi \mid \neg\phi \mid \langle\!\langle A \rangle\!\rangle \bigcirc \phi \mid \langle\!\langle A \rangle\!\rangle \phi \,\mathcal{U}\, \phi \mid \langle\!\langle A \rangle\!\rangle \phi \,\mathcal{W}\, \phi$$

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

Formulas of the type $\langle\!\langle A \rangle\!\rangle \phi$ read as "coalition A can enforce ϕ". The operator $\langle\!\langle \cdot \rangle\!\rangle$ is a *path quantifier* and $\bigcirc$("next"), $\mathcal{U}$("until"), $\mathcal{W}$("weak until") are *temporal operators*.

The usual derived operators can be obtained as follows:

$$\langle\!\langle A \rangle\!\rangle \Box \phi \equiv \langle\!\langle A \rangle\!\rangle \phi \,\mathcal{W}\, false \qquad \langle\!\langle A \rangle\!\rangle \Diamond \phi \equiv \langle\!\langle A \rangle\!\rangle true \,\mathcal{U}\, \phi$$
$$[\![A]\!] \Box \phi \equiv \neg \langle\!\langle A \rangle\!\rangle \Diamond \neg\phi \qquad [\![A]\!] \Diamond \phi \equiv \neg \langle\!\langle A \rangle\!\rangle \Box \neg\phi$$

Formulas of the type $[\![A]\!]\phi$ reads as "coalition A cannot avoid" ϕ.

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

- $(\Gamma, \rho, i) \models_\alpha p$ if $p \in \lambda(\rho[i])$ for any $\alpha \in \{ind, ck, dk\}$.
- $(\Gamma, \rho, i) \models_\alpha \phi_1 \wedge \phi_2$ if $(\Gamma, \rho, i) \models_\alpha \phi_1$ and $(\Gamma, \rho, i) \models_\alpha \phi_2$, again for any $\alpha \in \{ind, ck, dk\}$.
- $(\Gamma, \rho, i) \models_\alpha \neg\phi$ if $(\Gamma, \rho, i) \not\models_\alpha \phi$, for any $\alpha \in \{ind, ck, dk\}$.
- $(\Gamma, \rho, i) \models_{ind} \langle\!\langle A \rangle\!\rangle \bigcirc \phi$ if there exists $\sigma \in \Sigma_{ind}(A, \Gamma)$ such that $(\Gamma, \rho', i+1) \models_\alpha \phi$ for all runs ρ' which are compatible with σ and satisfy $\rho'[0 \cdots i] = \rho[0 \cdots i]$;
- $(\Gamma, \rho, i) \models_{ind} \langle\!\langle A \rangle\!\rangle \phi_1 \,\mathcal{U}\, \phi_2$ if there exists $\sigma \in \Sigma_{ind}(A, \Gamma)$ such that for all runs ρ' which are compatible with σ and satisfy $\rho'[0 \cdots i] = \rho[0 \cdots i]$ there exists $j \geq i$ such that $(\Gamma, \rho', j) \models_{ind} \phi_2$ and $(\Gamma, \rho', k) \models_{ind} \phi_1$, for all $k, i \leq k \leq j-1$;
- $(\Gamma, \rho, i) \models_{ind} \langle\!\langle A \rangle\!\rangle \phi_1 \,\mathcal{W}\, \phi_2$ if there exists $\sigma \in \Sigma_{ind}(A, \Gamma)$ such that for all runs ρ' which are compatible with σ and satisfy $\rho'[0 \cdots i] = \rho[0 \cdots i]$ there exists $j \geq i$ such that $(\Gamma, \rho', j) \models_{ind} \phi_2$ and $(\Gamma, \rho', k) \models_{ind} \phi_1$, for all $k, i \leq k \leq j-1$, or $(\Gamma, \rho', j) \models_{ind} \phi_1$, 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:

- $(\Gamma, \rho, i) \models_{ck} \langle\!\langle A \rangle\!\rangle \bigcirc \phi$ if there exists $\sigma \in \Sigma_{ck}(A, \Gamma)$ such that $(\Gamma, \rho', i+1) \models_{ck} \phi$ for all runs ρ' which are compatible with σ and satisfy $\rho'[0 \cdots i] \sim_A \rho[0 \cdots i]$;

- $(\Gamma, \rho, i) \vDash_{ck} \langle\!\langle A \rangle\!\rangle \phi_1 \mathcal{U} \phi_2$ if there exists $\sigma \in \Sigma_{ck}(A, \Gamma)$ such that for all runs ρ' which are compatible with σ and satisfy $\rho'[0 \cdots i] \sim_A \rho[0 \cdots i]$ there exists $j \geq i$ such that $(\Gamma, \rho', j) \vDash_{ck} \phi_2$ and $(\Gamma, \rho', k) \vDash_{ck} \phi_1$, for all $k, i \leq k \leq j-1$;
- $(\Gamma, \rho, i) \vDash_{ck} \langle\!\langle A \rangle\!\rangle \phi_1 \mathcal{W} \phi_2$ if there exists $\sigma \in \Sigma_{ck}(A, \Gamma)$ such that for all runs ρ' which are compatible with σ and satisfy $\rho'[0 \cdots i] \sim_A^C \rho[0 \cdots i]$ there exists $j \geq i$ such that $(\Gamma, \rho', j) \vDash_{ck} \phi_2$ and $(\Gamma, \rho', k) \vDash_{ck} \phi_1$, for all $k, i \leq k \leq j-1$, or $(\Gamma, \rho', j) \vDash_{ck} \phi_1$, for all $j \in \mathbb{N}, j \geq i$.

Finally, the $\vDash_{dk}$ semantics is defined similarly with $\vDash_{ck}$, with the difference that all references to $\sim_A^C$ 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 ϕ is satisfied in the game arena Γ, written $\Gamma \vDash_\alpha \phi$, if $(\Gamma, \rho, 0) \vDash_\alpha \phi$ for all $\rho \in Runs^\omega(\Gamma)$.

We note that the usual notation ATL denotes the logic in which the semantics is given by $\vDash_{ind}$. Also, in (Dima et al., 2010), ATL_{prs}^D denotes the logic in which the semantics is given by $\vDash_{dk}$. This is a modification the ATL logic with operators of the type $\langle\!\langle A \rangle\!\rangle_{D(A)}$ 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 $\vDash_{ck}$ and the operators of the type $\langle\!\langle A \rangle\!\rangle_{C(A)}$ from (Jamroga and van der Hoek, 2004).

In the balance of the paper, ATL_C^{prs} denotes the logic whose semantics is given by $\vDash_{ck}$.

4 Undecidability of the model-checking problem for ATL_C^{prs}

In this section we present our undecidability result for the model checking problem in ATL_C^{prs}. The problem statement is the following:

Problem 1. (The Model Checking Problem for ATL_C^{prs}) Given a game arena Γ and an ATL_C^{prs} formula ϕ, decide whether $\Gamma \vDash_{ck} \phi$.

Theorem 1. *The model checking problem for ATL_C^{prs} is undecidable.*

Proof. We reduce the Halting problem for Turing machines that start with an empty tape to the model checking problem for ATL_C^{prs}. 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 transision.

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 game reaches some state which signals that the Turing machine halts.

Formally, for a Turing machine $M = \langle Q^M, \Gamma^M, \beta, \Sigma^M, \delta^M, q_0, F^M \rangle$ 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:

$$\begin{aligned} Q = {} & \Sigma^{M\bullet} \cup {}^\bullet\Sigma^M \cup \overline{\Sigma^{M\bullet}} \cup {}^\bullet\overline{\Sigma^M} \cup \{\tilde{\beta}\} \cup (\Sigma^M \times Q^M) \\ & \cup (\Sigma^M \times Q^M \times \Sigma^M \times \{left, right\} \times \{prev, next\}) \\ & \cup \{\epsilon_L, \epsilon_R, \overline{\epsilon_L}, \overline{\epsilon_R}, \tilde{\epsilon_L}, \tilde{\epsilon_R}\}. \end{aligned} \tag{5}$$

(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 $\Sigma^{M\bullet}$, ${}^\bullet\Sigma^M$, $\overline{\Sigma^{M\bullet}}$ and ${}^\bullet\overline{\Sigma^M}$ are four copies of the set of states of M. The states in $\Sigma^{M\bullet} = \{s^\bullet \mid s \in \Sigma^M, s \neq \beta\}$ correspond to the symbols preceding the head. The states in ${}^\bullet\Sigma^M = \{{}^\bullet s \mid s \in \Sigma^M\}$ 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 $\overline{\Sigma^{M\bullet}} = \{\overline{s^\bullet} \mid s \in \Sigma^M, s \neq \beta\}$ and ${}^\bullet\overline{\Sigma^M} = \{\overline{s^\bullet} \mid s \in \Sigma^M\}$ *overlined states*. We shall use them to encode final configurations. The state $\tilde{\beta}$ 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 $(\Sigma^M \times Q^M \times \Sigma^M \times \{left, right\} \times \{prev, next\})$ 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 ϵ_L will encode the left margin of the tape and the state ϵ_R, the right margin. Likewise, $\tilde{\epsilon_L}, \tilde{\epsilon_R}$ 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 $Q_0 = \{\epsilon_L, \overline{\epsilon_L}, \tilde{\epsilon_L}\}$ and the set of actions consists of a singleton set $C_1 = C_2 = C = \{act\}$.

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 $s, s' \in \Sigma^M$, $s^\bullet \xrightarrow{act} s'^\bullet \in \delta$, ${}^\bullet s \xrightarrow{act} {}^\bullet s' \in \delta$, $\beta \xrightarrow{act} \beta \in \delta$, $\overline{s^\bullet} \xrightarrow{act} \overline{s'^\bullet} \in \delta$, $\overline{{}^\bullet s} \xrightarrow{act} \overline{{}^\bullet s'} \in \delta$, $\overline{\beta} \xrightarrow{act} \overline{\beta} \in \delta$.
– A state in the set $\Sigma^M \times Q^M$ is preceded by the states in $\Sigma^{M\bullet}$ and followed by states in ${}^\bullet\Sigma^M$; this is modeled by the following transitions: for all $s, s', s'' \in \Sigma^M, s \neq \beta$, for all $q \in Q^M \setminus (F^M \cup \{q_0\})$, $s^\bullet \xrightarrow{act} \langle s', q\rangle \in \delta$, and $\langle s', q\rangle \xrightarrow{act} {}^\bullet s'' \in \delta$.
– The same rule applies for the final configuration: for all $s, s' \in \Sigma^M, s \neq \beta$, for all $q_f \in F^M$, $\overline{s^\bullet} \xrightarrow{act} \langle s', q_f\rangle \in \delta$, $\langle s', q_f\rangle \xrightarrow{act} \overline{{}^\bullet s} \in \delta$.
– 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, $s^\bullet \xrightarrow{act} \langle s_n, q_n, left, next\rangle \in \delta$, $\langle s_n, q_n, left, next\rangle \xrightarrow{act} \langle s_p, q_p, left, prev\rangle \in \delta$, $\langle s_p, q_p, left, prev\rangle \xrightarrow{act} {}^\bullet s' \in \delta$,
– When the head moves to right we have $s^\bullet \xrightarrow{act} \langle s_p, q_p, right, prev\rangle \in \delta$, $\langle s_p, q_p, right, prev\rangle \xrightarrow{act} \langle s_n, q_n, right, next\rangle \in \delta$, $\langle s_n, q_n, right, next\rangle \xrightarrow{act} {}^\bullet s' \in \delta$ for all symbols $s, s', s_p, s_n \in \Sigma^M$ and for all states $q_p, q_n \in Q^M \setminus (F^M \cup \{q_0\})$ for which $(s_n, q_n, R) \in \delta^M(s_p, q_p)$ holds.
– No transition leading to the initial state is permitted. Instead, from the initial state we can pass into any other symbol: $\epsilon_L \xrightarrow{act} s^\bullet \in \delta$, $\epsilon_L \xrightarrow{act} \langle s, q\rangle \in \delta$. In a run encoding a transition the next state will be the leftmost encoding the transition. $\epsilon_L \xrightarrow{act} \langle s, q, right, prev\rangle \in \delta$, $\epsilon_L \xrightarrow{act} \langle s, q, left, next\rangle \in \delta$.
– A dual situation holds for the final state, ϵ_R: for all symbols $s \in \Sigma^M$, and for all states $q \in Q^M$, ${}^\bullet s \xrightarrow{act} \epsilon_R \in \delta$, $\langle s, q\rangle \xrightarrow{act} \epsilon_R \in \delta$, $\langle s, q, right, next\rangle \xrightarrow{act} \epsilon_R \in \delta$, $\langle s, q, left, prev\rangle \xrightarrow{act} \epsilon_R \in \delta$.
– The same rules apply for the runs encoding the final configuration. In this situation the head must be on a final state: $\overline{\epsilon_L} \xrightarrow{act} \overline{s^\bullet} \in \delta$, $\overline{\epsilon_L} \xrightarrow{act} \langle s, q_f\rangle \in \delta$, for all symbols $s \in \Sigma^M, s \neq \beta$ and for all final states $q_f \in F^M$;
– $\overline{{}^\bullet s} \xrightarrow{act} \overline{\epsilon_R} \in \delta$, $\langle s, q_f\rangle \xrightarrow{act} \epsilon_R \in \delta$, for all $s \in \Sigma^M$ and for all $q_f \in F^M$;
– In the initial run the head is on the first position and encodes the initial configuration, the empty tape. $\tilde{\epsilon_L} \xrightarrow{act} \langle \beta, q_0\rangle \in \delta$. In order to force the game arena to encode the initial configuration we distinguish between β and $\tilde{\beta}$. $\tilde{\beta}$ will only follow an initial state. $\langle \beta, q_0\rangle \xrightarrow{act} \tilde{\beta} \in \delta$, $\tilde{\beta} \xrightarrow{act} \tilde{\beta} \in \delta$, $\tilde{\beta} \xrightarrow{act} \tilde{\epsilon_R} \in \delta$;
– Once the final state is reached it is never left. $\epsilon_R \xrightarrow{act} \epsilon_R$, $\tilde{\epsilon_R} \xrightarrow{act} \tilde{\epsilon_R}$, $\overline{\epsilon_R} \xrightarrow{act} \overline{\epsilon_R}$;
– Finally, nothing else belongs to δ.

The set of atomic propositions is

$$\Pi = \{p_1, p_2, p_3\} \cup \{p_\beta\} \cup \{p_{s,1}, p_{q,1}, p_{s,2}, p_{q,2} \mid s \in \Sigma^M, q \in Q\}.$$

The valuation function is defined by the following rules:

$$p_1 \in \lambda_1(x), p_1 \in \lambda_2(x) \text{ for all } x \in \left\{\tilde{\beta}, \langle \beta, q_0^M\rangle, \tilde{\epsilon_L}, \tilde{\epsilon_R}\right\} \quad (6)$$

$$p_2 \in \lambda_1(x), p_2 \in \lambda_2(x) \text{ for all } x \in \left\{\overline{s^\bullet}, \overline{{}^\bullet s}, \langle s, q\rangle, \overline{\epsilon_L}, \overline{\epsilon_R} \mid s \in \Sigma^M, q \in Q^M\right\} \quad (7)$$

$$p_3 \in \lambda_1(\tilde{\epsilon_R}), p_1 \in \lambda_2(\tilde{\epsilon_R}) \quad (8)$$

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
  $\varsigma \in \{s^\bullet, \overline{s^\bullet}, {}^\bullet s, \overline{{}^\bullet s}, \langle s,q\rangle \mid q \in Q^M\}$
  $\cup \{\langle s,q,left,next\rangle \mid s',s'' \in \Sigma^M, q' \in Q^M, (s'',q,L) \in \delta^M(s',q')\}$
  $\cup \{\langle s,q,right,next\rangle \mid s',s'' \in \Sigma^M, q' \in Q^M, (s'',q,R) \in \delta^M(s',q')\}$
  $\cup \{\langle s,q,right,prev\rangle \mid s' \in \Sigma^M, q' \in Q^M, (s',q',R) \in \delta^M(s,q)\}$
  $\cup \{\langle s,q,left,prev\rangle \mid s' \in \Sigma^M, q' \in Q^M, (s',q',L) \in \delta^M(s,q)\}$;
- $p_{q,1} \in \lambda_1(\varsigma)$, for all
  $\varsigma \in \{\langle s,q\rangle \mid s \in \Sigma^M\}$
  $\cup \{\langle s,q,right,prev\rangle \mid s' \in \Sigma^M, q' \in Q^M, (s',q',R) \in \delta^M(s,q)\}$
  $\cup \{\langle s,q,left,prev\rangle \mid s' \in \Sigma^M, q' \in Q^M, (s',q',L) \in \delta^M(s,q)\}$;
- $p_{s,2} \in \lambda_2(\varsigma)$, for all
  $\varsigma \in \{s^\bullet, \overline{s^\bullet}, {}^\bullet s, \overline{{}^\bullet s}, \langle s,q\rangle \mid q \in Q^M\}$
  $\cup \{\langle s',q',left,prev\rangle \mid q \in Q^M, (s,q,L) \in \delta^M(s',q')\}$
  $\cup \{\langle s',q',right,prev\rangle \mid q \in Q^M, (s,q,R) \in \delta^M(s',q')\}$
  $\cup \{\langle s,q,right,next\rangle \mid s' \in \Sigma^M, q' \in Q^M, (s,q,R) \in \delta^M(s',q')\}$
  $\cup \{\langle s,q,left,next\rangle \mid s' \in \Sigma^M, q' \in Q^M, (s,q,L) \in \delta^M(s',q')\}$;
- $p_{q,2} \in \lambda_2(\varsigma)$, for all
  $\varsigma \in \{\langle s,q\rangle \mid s \in \Sigma^M\}$
  $\cup \{\langle s,q,right,next\rangle \mid s' \in \Sigma^M, q' \in Q^M, (s,q,R) \in \delta^M(s',q')\}$
  $\cup \{\langle s,q,left,next\rangle \mid s' \in \Sigma^M, q' \in Q^M, (s,q,L) \in \delta^M(s',q')\}$;
- $p_\beta \in \lambda_i(\beta) \cap \lambda_i(\tilde{\beta})$ 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 $\rho \in Runs^f(\Gamma) \cup Runs^\omega(\Gamma)$ is uniquely accepted by one of the three parts of the game arena.

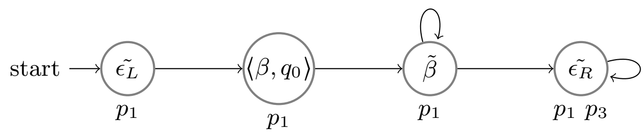

Fig. 1. The initial configuration

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.

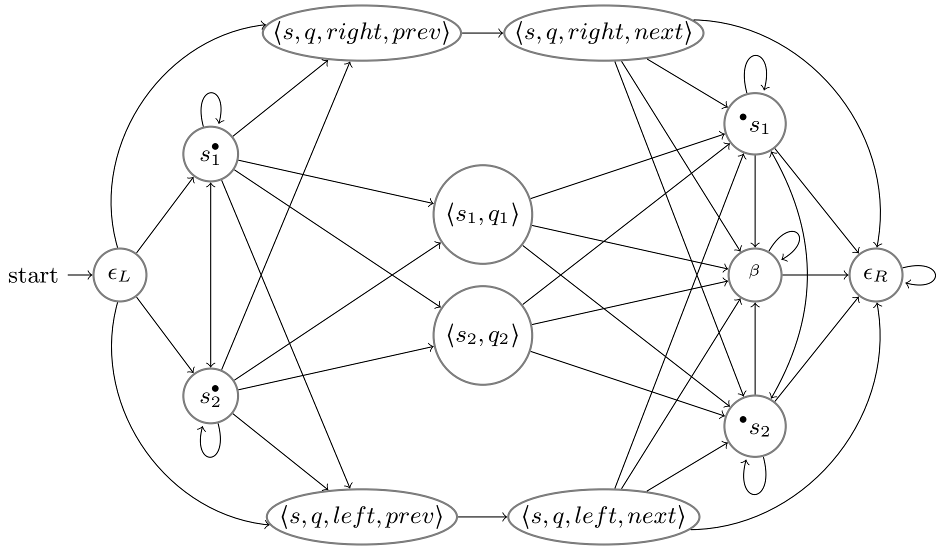

Fig. 2. An intermediate configuration (only the states corresponding with two symbols from Σ^M are represented here)

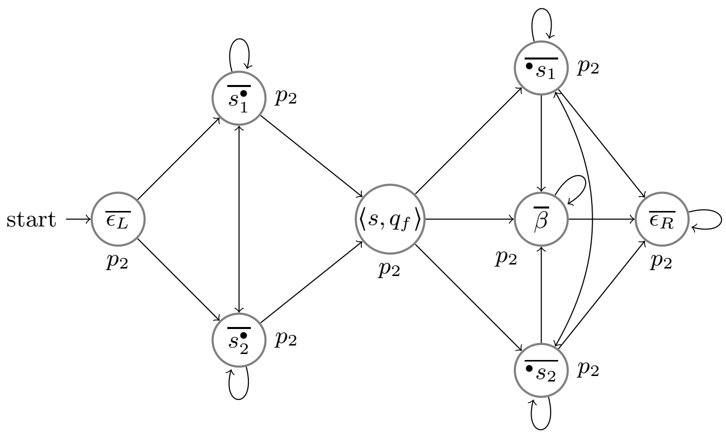

Fig. 3. The final configuration

A configuration $cnfg = s_1 \cdots s_{i-1} \langle s_i, q\rangle s_{i+1} \cdots s_n$ is encoded by a run $\rho = \epsilon_L s^{\bullet}_1 \cdots s^{\bullet}_{i-1} \langle s_i, q\rangle {}^{\bullet}s_{i+1} \cdots {}^{\bullet}s_n \beta^m \epsilon_R$. There are infinitely many runs that encode one configuration, in this case, all runs that start with the state sequence $\epsilon_L s^{\bullet}_1 \cdots s^{\bullet}_{i-1} \langle s_i, q\rangle {}^{\bullet}s_{i+1} {}^{\bullet}s_{i+2} \cdots {}^{\bullet}s_n \beta^m \epsilon_R$. An initial configuration is encoded by all runs that start with $\tilde{\epsilon_L} \langle \beta, q_0\rangle \tilde{\beta}^m \tilde{\epsilon_R}$. Respectively, a final configuration $cnfg_f = s_1 \cdots s_{k-1} \langle s_k, q_f\rangle s_{k+1} \cdots s_n$ is encoded by all the runs starting with the overlined state sequence $\overline{\epsilon_L} \overline{s^{\bullet}_1} \cdots \overline{s^{\bullet}_{k-1}} \langle s_k, q\rangle \overline{{}^{\bullet}s_{k+1}} \cdots \overline{{}^{\bullet}s_n} \beta^m \overline{\epsilon_R}$, for $m \in \mathbb{N}$, $m \neq \infty$.

The other type of run that may occur in Γ encodes transitions. For two configurations

$$cnfg = s_1 \cdots s_{k-1} \langle s_k, q\rangle s_{k+1} s_{k+2} \cdots s_n \text{ and}$$
$$cnfg' = s_1 \cdots s_{k-1} s'_k \langle s_{k+1}, q'\rangle s_{k+2} \cdots s_n$$

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 Γ:

$$\epsilon_L s_1^{\bullet} \cdots s_{k-1}^{\bullet} \langle s_k, q, right, prev \rangle \langle s_{k+1}, q', right, next \rangle^{\bullet} s_{k+2} \cdots {}^{\bullet} s_n \epsilon_R.$$

Similarly, when $cnfg' = s_1 \cdots s_{k-2} \langle s_{k-1}, q' \rangle s'_k s_{k+1} \cdots s_n$ 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:

$$\epsilon_L s_1^{\bullet} \cdots s_{k-2}^{\bullet} \langle s_{k-1}, q', left, next \rangle \langle s_k, q, left, prev \rangle^{\bullet} s_{k+1} \cdots {}^{\bullet} s_n \epsilon_R$$

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 ρ_1 and ρ_2 are two runs in the game arena Γ, then $\rho_1 \sim_1 \rho_2$ iff $\rho_1 = \rho_2$ or ρ_1 encodes a configuration $cnfg$ and ρ_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 λ_1. In order to prove the implication from left to right, we shall verify whether we have another run different from the mentioned ones, $\sim_1$ equivalent with ρ_1.

Consider $\rho_1 = \epsilon_L s_1^{\bullet} \cdots s_{i-1}^{\bullet} \langle s_i, q_p \rangle^{\bullet} s_{i+1} \cdots {}^{\bullet} s_n \epsilon_R$ and $\rho_2 = x_L x_1 \cdots x_n x_R$. Consider $(s'_i, q_n, L) \in \delta^M(s_i, q_p)$. The case when the head moves to right is similar. Suppose that $\rho_1 \sim_1 \rho_2$, $\rho_1 \neq \rho_2$ and ρ_1 encodes a configuration $cnfg_1$ and there is no configuration $cnfg_2$ in the Turing machine M such that $\rho_2 = (cnfg_1 \vdash cnfg_2)$.

We consider only initialized runs, thus the initial state $x_l \in \{\epsilon_L, \overline{\epsilon_L}, \tilde{\epsilon_L}\}$. Since a run contains $\tilde{\epsilon_L}$ followed by $\langle \beta, q_0 \rangle$ and $\overline{\epsilon_L}$ followed by $\langle s, q_f \rangle$, 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 λ_1 and λ_2 and the fact that $\langle s_j, q', left, next \rangle \xrightarrow{act} x$ if and only if $x = \langle s_{j+1}, q'', left, prev \rangle$, it follows that for all $j, 1 \leq j \leq i-2$ we have $x_j = s_j^{\bullet}$. Hence we have $x_j = s_j^{\bullet}$ for $j, i+1 \leq j \leq n$.

Furthermore, we have that $\lambda_1(s_{i-1}^{\bullet}) = \{p_{s_{i-1}}\} = \lambda_1(s)$ only for $s \in \{s_{i-1}^{\bullet}, \overline{s_{i-1}^{\bullet}}, {}^{\bullet}s_{i-1}, \overline{{}^{\bullet}s_{i-1}}\} \cup \{\langle s_{i-1}, q, left, next \rangle \mid s', s'' \in \Sigma^M, q' \in Q^M, \text{ s.t. } (s'', q, L) \in \delta^M(s', q')\} \cup \{\langle s_{i-1}, q, right, next \rangle \mid s', s'' \in \Sigma^M, q' \in Q^M, \text{ s.t. } (s'', q, R) \in \delta^M(s', q')\}$. We can eliminate the overlined states because an overlined state follows only another overlined state or a state containing q_f, $q_f \in F^M$. We also eliminate the states of the type $\langle s_{i-1}, q, right, next \rangle$ because this states follow only $\langle s', q', right, prev \rangle$, for $s', s'' \in \Sigma^M, q' \in Q^M, (s'', q, R) \in \delta^M(s', q')$. Now, since only $s_{i-1}^{\bullet} \xrightarrow{act} \langle s_i, q \rangle$ and $\langle s_{i-1}, q', left, next \rangle \xrightarrow{act} \langle s_i, q, left, prev \rangle$ belong to δ, we have either $x_{i-1} = s_{i-1}^{\bullet}$ and $x_i = \langle s_i, q \rangle$, or $x_{i-1} = \langle s_{i-1}, q', left, next \rangle$ and $x_i = \langle s_i, q, left, prev \rangle$.

Because none of the observationally equivalent states can form a correct initialized run, we have proved that $\rho_1 \sim_1 \rho_2$ implies $\rho_1 = \rho_2$ or ρ_1 encodes a configuration $cnfg$ and ρ_2 encodes a transition coding run $cnfg \vdash cnfg'$. □

Proposition 3. *If ρ_1 and ρ_2 are two runs in the game arena Γ, then $\rho_1 \sim_2 \rho_2$ iff $\rho_1 = \rho_2$ or ρ_1 encodes a configuration $cnfg$ and ρ_2 encodes a transition coding run $cnfg' \vdash cnfg$.*

Proof. The proof is similar to the one used for Proposition 2. □

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 ρ, ρ' runs in the game arena Γ that encode $cnfg$ and, respectively, $cnfg'$, such that $|\rho| = |\rho'|$ and $\rho \sim_{\{1,2\}}^C \rho'$.*

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 $cnfg_0, cnfg_1, \cdots, cnfg_m$ such that $cnfg = cnfg_0 \Rightarrow_M cnfg_1 \Rightarrow_M \cdots \Rightarrow_M cnfg_m = cnfg'$. We write this $cnfg_0 \Rightarrow_M^* cnfg_m = (cnfg_i \Rightarrow_M cnfg_{i+1})_{0 \le i \le m-1}$.

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 ρ_i, ρ_{i+1} two runs that encode $cnfg_i$ and, respectively, $cnfg_{i+1}$ in the game arena Γ such that $|\rho_i| = |\rho_{i+1}| \ge |cnfg_i|$ and $\rho_i \sim_1 (cnfg_i \vdash cnfg_{i+1}) \sim_2 \rho_{i+1}$, 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 $(\rho_i \sim_1 (cnfg_i \vdash cnfg_{i+1}) \sim_2 \rho_{i+1})_{0 \le i \le m-1}$. According to our definition of λ_1 and λ_2, we have now $\rho_1 \sim_{\{1,2\}}^C \rho_m$.

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

We now turn to the proof of the theorem. We prove in the rest of this section that $(\Gamma, \rho, 0) \vDash_{ck} [\![1,2]\!] \diamond (p_1 \wedge p_3 \wedge [\![1,2]\!] \square p_2)$ iff the 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 $i \in \mathbb{N}, 1 \le i \le n$ and there exists a finite sequence of configurations $cnfg_0 \Rightarrow_M^* cnfg_f$, where $cnfg_0 = \langle \beta, q_0 \rangle \beta^{n-1}$ is an initial configuration, $cnfg_f = s_1 \cdots s_{k-1} \langle s_k, q_f \rangle s_{k+1} \cdots s_n$ is a final configuration for some $s_1, \cdots, s_{k-1} \in \Sigma^M \setminus \{\beta\}$ and $s_{k+1}, \cdots, s_n \in \Sigma^M$, $0 \le k \le n$. 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 Γ there exists a finite sequence of runs $\rho_0, \rho_1, \cdots, \rho_f$ and $\hat{\rho}_0, \hat{\rho}_1, \cdots, \hat{\rho}_{f-1}$ such that

$$\begin{array}{l} \rho_0 \sim_1 \hat{\rho}_0 \sim_2 \rho_1 \sim_1 \hat{\rho}_1 \sim_2 \cdots \sim_1 \hat{\rho}_{f-1} \sim_2 \rho_f \text{ and} \\ \rho_0 = \tilde{\epsilon_L} \langle \beta, \rho_0 \rangle \tilde{\beta}^{n-1} \tilde{\epsilon_R} \text{ encodes the initial configuration} \\ \rho_i = \epsilon_L s_1^\bullet \cdots s_{j-1}^\bullet \langle s_j, q \rangle^\bullet s_{j+1} \cdots,^\bullet s_n \epsilon_R \text{ encodes the } i\text{-th configuration} \\ cnfg_i, \text{ for } \underline{\text{all } i, 1 \le i \le f} \text{ and for some } j, 1 \le j \le n. \\ \rho_f = \overline{\epsilon_L} \overline{s_1^\bullet} \cdots \overline{s_{l-1}^\bullet} \langle s_l, q \rangle \overline{{}^\bullet s_{l+1}} \cdots \overline{{}^\bullet s_n} \overline{\epsilon_R}, \text{ encodes the final configuration} \\ (\Gamma, \rho_0, i) \vDash_{ck} p_1, (\Gamma, \rho_0, n) \vDash_{ck} p_3, (\Gamma, \rho_f, i) \vDash_{ck} p_2, \ \forall i, 1 \le i \le n. \end{array} \tag{9}$$

We consider a game arena with a single action. Therefore there is only one corresponding final run ρ_f, $\sim_{\{1,2\}}^C$ reachable from ρ_0, and, furthermore, the proposition p_3 is visible in the n-th state, of ρ_f. We therefore conclude that the ATL_C^{prs} formula $[\![1,2]\!] \diamond (p_1 \wedge p_3 \wedge [\![1,2]\!] \square p_2)$ holds in the game arena Γ.

In order to prove the implication from right to left we show that for any run satisfying the ATL_C^{prs} formula $(\Gamma, \rho, 0) \vDash_{ck} [\![1,2]\!] \diamond (p_1 \wedge p_3 \wedge [\![1,2]\!] \square p_2)$, there exists an initial run $\sim_{1,2}^C$-equivalent with ρ.

If $(\Gamma, \rho, 0) \vDash_{ck} [\![1,2]\!] \Diamond (p_1 \wedge p_3 \wedge [\![1,2]\!] \Box p_2)$ then in the game arena there exists ρ' a run and there exist $n \in \mathbb{N}$ and $i \in \mathbb{N}, 1 \leq i \leq n$ such that

$$\rho = \tilde{\epsilon_L} \langle \beta, q_0 \rangle \tilde{\beta}^n \tilde{\epsilon_R}$$

$$\rho' = \overline{\epsilon_L} \overline{s_1^{\bullet}} \cdots \overline{s_{i-1}^{\bullet}} \langle s_i, q \rangle \overline{{}^{\bullet}s_{i+1}} \cdots \overline{{}^{\bullet}s_n} \overline{\epsilon_R}$$

This means that there exists $k \in \mathbb{N}$, a sequence of k runs $\rho_0, \cdots, \rho_k$ such that $\rho = \rho_0 \sim_{a_1} \hat{\rho}_1 \sim_{a_2} \rho_2 \sim_{a_1} \cdots \sim_{a_2} \rho_k = \rho'$.

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

$$cnfg_0 \Rightarrow_M cnfg_1 \Rightarrow_M \cdots \Rightarrow_M cnfg_k.$$

where $cnfg_0$ is an initial configuration and

$$cnfg_k = s_1 \cdots s_{i-1} \left\langle s_i, q_f \right\rangle s_{i+1} \cdots s_n,$$

for some i, is a final configuration.

Hence, the machine M halts when starting with an empty tape. □

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_C^{prs}, a result inspired from the techniques used in (Fagin 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

Alur, R., Henzinger, T., and Kupferman, O. (1998). Alternating-time temporal logic. In *Proceedings of COMPOS'97*, volume 1536 of *LNCS*, pages 23–60. Springer Verlag.

Alur, R., Henzinger, T., and Kupferman, O. (2002). Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713.

Bulling, N., Dix, J., and Jamroga, W. (2010). Model checking logics of strategic ability: Complexity. In Dastani, M., Hindriks, K. V., and Meyer, J.-J. C., editors, *Specification and Verification of Multi-Agent Systems*, pages 125–160. Springer.

Dima, C., Enea, C., and Guelev, D. (2010). Model-checking an alternating-time temporal logic with knowledge, imperfect information, perfect recall and communicating coalitions. *Electronic Proceedings in Theoretical Computer Science*, 25:103–117.

Fagin, R., Halpern, J., Moses, Y., and Vardi, M. (2004). *Reasoning about knowledge*. The MIT Press.

Jamroga, W. and Agotnes, T. (2007). Constructive knowledge: What agents can achieve under imperfect information. *Journal of Applied Non-Classical Logics*, 17(4):423–475.

Jamroga, W. and van der Hoek, W. (2004). Agents that know how to play. *Fundamenta Informaticae*, 63(2-3):185–219.

Lomuscio, A. and Penczek, W. (2007). Logic column 19: Symbolic model checking for temporal-epistemic logics. *CoRR*, abs/0709.0446.

Schobbens, P.-Y. (2004). Alternating-time logic with imperfect recall. *Electronic Notes in Theoretical Computer Science*, 85(2):82–93.

Shilov, N. V. and Garanina, N. O. (2002). Model checking knowledge and fixpoints. In *Proceedings of FICS'02*, pages 25–39, Extended version available as Preprint 98, Ershov Institute of Informatics, Novosibirsk.

van Benthem, J. and Pacuit, E. (2006). The tree of knowledge in action: Towards a common perspective. In *Proceedings of AiML'06*, pages 87–106. College Publications.

van der Meyden, R. (1998). Common knowledge and update in finite environments. *Information and Computation*, 140(2):115–157.

van der Meyden, R. and Shilov, N. V. (1999). Model checking knowledge and time in systems with perfect recall (extended abstract). In *Proceedings of FSTTCS'99*, volume 1738 of *LNCS*, pages 432–445.