Model Checking $ATL^*$ on vCGS

F. Belardinelli, C. Dima, I. Boureanu, and V. Malvone

March 12, 2019

We prove that the model checking $ATL^*$ on vCGS is undecidable. To do so, we reduce this problem to model checking $ATL^*$ on iCGS.

Consider an $ATL^*$ formula $\varphi$ to be model-checked on a given iCGS $M = \langle Ag,\{Act_a\}_{a \in Ag}, S, S_0, P, \tau, \{\neg a\}_{a \in Ag}, \pi \rangle$. Out of $M$ and $\varphi$, we define a vCGS $\Delta_M$ (which we sometimes refer to simply as $\Delta$) and an $ATL^*$ formula $\psi_{\varphi, M}$, as follows.

Agents & atoms. The set of agents in $\Delta_M$ is $Ag' = Ag \cup \{e\}$, where $e$ denotes the Environment agent. For each agent $a \in Ag$, the set of atoms controlled by $a$ includes an atom for each action in $Act_a$, i.e., $V_a = \{p_{act} | act \in Act_a\}$. The environment controls atoms corresponding to each state of the iCGS and each indistinguishability class for each agent, i.e., $V_e = \{p_s | s \in S\} \cup \{p[s]_a | s \in S, a \in Ag\}$, with $[s]_a = \{s' \in S | s' \sim_a s\}$.

An agent specification is $spec_a = \langle V_a, GC_a \rangle$ with the guarded commands to be defined hereafter. First, for each agent specification $spec_a$ and subset $W \subseteq V_a$, by $invis^a(W)$ we denote the boolean formula specifying that all of $a$'s atoms in $W$ are left invisible to any agent other than $a$, i.e., $a$ does not share any of her controlled atoms from $W$ with anyone:

$$invis^a(W) = \bigwedge_{b \in Ag \setminus \{a\}} \bigwedge_{v \in W} vis(v,b) := \text{ff}$$

For the special case of the environment (i.e., where $a$ in the above is replaced with the Environment and $W \subseteq V_e$, etc), we simply write $invis(W)$ instead of $invis^a(W)$ (i.e., $invis(W) = \bigwedge_{b \in Ag} \bigwedge_{v \in W} vis(v,b) := \text{ff}$.)

Second, for an agent $b \in Ag \setminus \{a\}$, by $vis(W,b)$ we denote the boolean formula specifying that all atoms in $W$ are visible to $b$, i.e., $a$ does share all her controlled atoms in $W$ with $b$:

$$vis(W,b) = \bigwedge_{v \in W} vis(v,b) := \text{tt}$$

For each agent $a \in Ag$ and for the Environment $e$, we also use boolean variables $turn_a$ and $turn_e$ to simulate their turns and mechanise the (synchronisation over) actions.

Guarded commands of init-type.

1. For each $s_0 \in S_0$, a guarded command $\gamma_{[s_0]_a}$ of init-type to agents $a$ is defined in $\Delta_M$ as follows:
\[\gamma_{[s_0]_a} := p_{[s_0]_a} \leadsto \text{turn}_a = \text{ff}, \text{invis}(V_a)\]

2. For each \(s_0 \in S_0\), a guarded command \(\gamma_{[s_0]_e}\) of \textit{init}-type for the Environment \(e\) is defined in \(\Delta_M\) as follows:

\[\gamma_{0_e} := \text{tt} \leadsto \text{invis}(\{p_{s_0} \mid s_0 \in S_0\}) \land b \in A_g \text{vis}(\{p_{[s_0]_b}\}, b)\]

Guarded commands of \textit{update}-type. Let \(\text{Act}_a\) in \(M\) be given as the set \(\{\alpha_1, \ldots, \alpha_{k_a}\}\). Then, agent \(a\)'s guarded commands of \textit{update}-type are added in \(\Delta_M\) as follows:

3. For every \(1 \leq i \leq k_a\) and each equivalence class \([s]_a \in S\), guarded command \(\gamma_{i,[s]_a}\) is defined as

\[\gamma_{i,[s]_a} := \text{turn}_a \land p_{[s]_a} \leadsto p_{\alpha_i} := \text{tt}, \bigwedge_{j \neq i} p_{\alpha_j} := \text{ff}, \text{turn} := \text{ff}\]

4. To simulate \(a\)'s turn (or a “moving forward” in the system—hence the name below), we add the following guarded command:

\[\gamma_{\text{fwd}_a} := \neg \text{turn} \leadsto \text{turn} := \text{tt}\]

Then, the environment agent has the following guarded commands of \textit{update}-type:

5. One guarded command of \textit{update}-type for \(e\), to make him “move forward” (i.e., take turns):

\[\gamma_{\text{fwd}_e} := \neg \text{turn}_e \leadsto \text{turn}_e := \text{tt}\]

6. For each \(s \in S\), \(a \in A_g\), and \(i_a \leq k_a\), let \(t\) denoted \(\tau(s,(\alpha_{i_a})_{a \in A_g})\); for this, another guarded command for \(e\) is as follows:

\[\gamma_{s, (i_a)_{a \in A_g}} := \text{turn}_e \land p_s \land p_{\alpha_{i_a}} \leadsto \text{turn}_e := \text{ff}, p_t := \text{tt}, \bigwedge_{b \in A_g} P_{[t]_b} := \text{ff}, \bigwedge_{p \in \pi(s)} P := \text{tt}, \bigwedge_{p \notin \pi(s)} P := \text{ff}, p_{s'} := \text{ff}, \bigwedge_{b \in A_g} P_{[s']_b} = \text{ff}\]

Now, using the reduction above (which is in \text{PTIME}), we can formally state the following result:

**Theorem 1 (3.4)** The model checking problem for ATL* (resp. ATL) on iCGS is \text{PTIME}-reducible to the same on vCGS.

**Proof.** Given an ATL* formula \(\varphi\), we construct the formula \(\varphi'\) in which each next operator is duplicated. For example, for the formula \(\varphi = Xp\), we set \(\varphi' = XXp\). For ATL we duplicate each coalition-next operator instead: for \(\varphi = \langle\langle A\rangle\rangle Xp\), we set \(\varphi' = \langle\langle A\rangle\rangle X wins(A) Xp\). Then we can show that, for any iCGS \(M\), the vCGS \(\Delta_M\) constructed as above is such that \(M \models \varphi\) iff \(\Delta_M \models \varphi'\). We do so by structural induction on the formula \(\varphi\).
Using Theorem 1 above and knowing from [1] that model checking $ATL^*$ and $ATL$ on iCGSs is undecidable, we can state the following result:

**Corollary 1 (3.5)** The model checking problem for $ATL^*$ (resp. $ATL$) on vCGS are undecidable.

We conclude by recalling that if we assume positional strategies, the model checking problem for $ATL^*$ (resp. $ATL$) on iCGS are PSPACE- (resp. $\Delta^2_P$-) complete [3, 2]. Hence, the same complexities apply to vCGS.

**References**

[1] C. Dima and F. Tiplea. Model-checking ATL under imperfect information and perfect recall semantics is undecidable. *CoRR*, abs/1102.4225, 2011.

[2] W. Jamroga and J. Dix. Model checking abilities under incomplete information is indeed $\Delta^2_P$-complete. In *EUMAS’06*, pages 14–15, 2006.

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