Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning (KR 2020)
Main Track

A Three-valued Approach to Strategic Abilities under Imperfect Information

Francesco Belardinelli[1,2] , **Vadim Malvone**[2]
[1]Imperial College London, United Kingdom
[2]Laboratoire IBISC, Université d'Evry, France
francesco.belardinelli@imperial.ac.uk, vadim.malvone@univ-evry.fr

Abstract

A major challenge for logics for strategies is represented by their verification in contexts of imperfect information. In this contribution we advance the state of the art by approximating the verification of Alternating-time Temporal Logic (ATL) under imperfect information by using perfect information and a three-valued semantics. In particular, we develop novel automata-theoretic techniques for the linear-time logic LTL, then apply these to finding "failure" states, where the ATL specification to be model checked is undefined. Such failure states can then be fed into a refinement procedure, thus providing a sound, albeit partial, verification procedure.

1 Introduction

Logic-based languages to reason about the strategic abilities of agents are a thriving area of research in the applications of formal methods to knowledge reasoning and representation (Jamroga 2018; Fagin et al. 1995). Over the years, several logics for strategies have been introduced, including Alternating-time Temporal Logic (Alur et al. 2002), Coalition Logic (Pauly 2002), Strategy Logic (Chatterjee et al. 2007; Mogavero et al. 2014), which has also led to the development of model checking tools (Alur et al. 1998; Lomuscio et al. 2017; Kurpiewski et al. 2019).

A key challenge for these logics for strategies is represented by their verification in contexts of imperfect information. Indeed, the model checking problem for the Alternating-time Temporal Logic ATL under the assumption of *perfect information* is known to be PTIME-complete (Alur et al. 2002). However, under *imperfect information* it ranges between Δ_2^P-complete to undecidable, depending on the underlying assumption on memory (Jamroga and Dix 2006; Dima and Tiplea 2011). Unfortunately, when reasoning about knowledge, the assumption of perfect information is either unrealistic or computationally costly. Thus, if logics for strategies are to be deployed in concrete multi-agent scenarios, it is crucial to develop even partial verification methods capable of tackling contexts of imperfect information. To this end, several proposals have been put forward, focusing on how the information is shared amongst agents (Berthon et al. 2017; Belardinelli et al. 2017a), or developing notions of constructive knowledge (Ågotnes et al. 2015) and bounded recall (Belardinelli et al. 2018), or again approximating strategy operators by using the μ-calculus (Bulling and Jamroga 2011) (see Section 6 for an in-depth comparison with related work).

In this contribution we advance the state of the art in reasoning about strategic abilities under imperfect information. More precisely, we develop further the line initiated in (Belardinelli et al. 2019), whereby imperfect information is approximated (or *abstracted*) by using perfect information and a three-valued semantics; thus leading to a sound, albeit partial, verification procedure for the logic ATL^* under imperfect information and perfect recall. The verification procedure there outlined is partial, as it can return the undefined truth value uu for some specifications, in some states in the system. In those cases, we would like to use such "failure" states to refine the abstract model. However, a key question left open in (Belardinelli et al. 2019) concerns how to find such failure states. In (Ball and Kupferman 2006) such a procedure was provided but only for the Alternating μ-calculus (AMC) under perfect information. Here we consider the arguably more complex case of full ATL^* under imperfect information, whose model checking problem is undecidable in general, differently from AMC. Moreover, we prove novel results on automata-theoretic techniques for linear-time temporal logic (LTL) interpreted on a three-valued semantics, that we deem of independent interest.

The contribution is structured as follows. In Sec. 2 we present the syntax of ATL^*, as well as its semantics given on concurrent game structures with imperfect information (iCGS). In Sec. 3 we recall the knowledge-based abstraction in (Belardinelli et al. 2019) and the related three-valued semantics. Then, in Sec. 4 we develop novel automata-theoretic techniques for three-valued LTL. Specifically, we show how to construct Büchi automata accepting all traces making an LTL formula undefined and then consider the related non-emptiness problem. These results are used in Sec. 5 to find failure states, which can then be fed into the refinement algorithm in (Belardinelli et al. 2019). We conclude in Sec. 6 by discussing related literature and pointing to future work.

2 Classic Imperfect Information

In this section we introduce the classic two-valued semantics for the Alternating-time Temporal Logic ATL^* under imperfect information and perfect recall. We assume sets

$Ag = \{1, \ldots, m\}$ of agents and AP of atoms. Given a set U, $\overline{U}$ denotes its complement. We denote the length of a tuple v as $|v|$, and its i-th element as v_i. Then, $last(v) = v_{|v|}$ is the last element in v. For $i \leq |v|$, let $v_{\geq i}$ be the suffix $v_i, \ldots, v_{|v|}$ of v starting at v_i and $v_{\leq i}$ its prefix $v_1, \ldots, v_i$.

We start by introducing concurrent game structures with imperfect information as models for multi-agent systems (Alur et al. 2002; Jamroga and van der Hoek 2004).

Definition 1 (iCGS). *Given sets Ag of agents and AP of atoms, a* concurrent game structure with imperfect information *is a tuple $M = \langle S, s_0, \{Act_i\}_{i \in Ag}, \{\sim_i\}_{i \in Ag}, d, \delta, V\rangle$ such that:*

- *$S \neq \emptyset$ is a finite set of* states, *with* initial state $s_0 \in S$.
- *For every agent $i \in Ag$, Act_i is a nonempty, finite set of* actions. *Then, let $Act = \bigcup_{i \in Ag} Act_i$ be the set of all actions, and $ACT = \prod_{i \in Ag} Act_i$ the set of all* joint actions.
- *For every agent $i \in Ag$, $\sim_i$ is the* indistinguishability relation *between states: for every $s, s' \in S$, $s \sim_i s'$ iff states s and s' are* observationally indistinguishable *for agent i.*
- *The* protocol function $d : Ag \times S \to (2^{Act} \setminus \emptyset)$ *defines the availability of actions so that for $i \in Ag$, $s \in S$, (i) $d(i, s) \subseteq Act_i$ and (ii) $s \sim_i s'$ implies $d(i, s) = d(i, s')$.*
- *The* transition function $\delta : S \times ACT \to S$ *returns a successor $s' = \delta(s, \vec{a})$ to every state $s \in S$ and joint action $\vec{a} \in ACT$ such that $a_i \in d(i, s)$ for every $i \in Ag$.*
- $V : S \times AP \to \{\mathrm{tt}, \mathrm{ff}\}$ *is the* two-valued labelling function.

By Def. 1 an iCGS represents the interactions of a group Ag of agents, from the initial state $s_0 \in S$, according to the transition function δ, as constrained by protocol d. Moreover, every agent i has imperfect information as regards the state of the system: in any state s, i considers possible all states s' that are i-indistinguishable from s (Fagin et al. 1995). When every $\sim_i$ is the identity relation, we obtain a standard CGS with perfect information (Alur et al. 2002).

Given a coalition $\Gamma \subseteq Ag$ and a joint action $\vec{a} \in ACT$, let $\vec{a}_\Gamma$ (resp. $\vec{a}_{\overline{\Gamma}}$) be the restricted tuple of actions for the agents in Γ (resp. $\overline{\Gamma}$) only. Finally, for $\vec{a}$ and $\vec{b}$ in ACT, $(\vec{a}_\Gamma, \vec{b}_{\overline{\Gamma}})$ denotes the joint action where the actions for the agents in Γ (resp. $\overline{\Gamma}$) are taken from $\vec{a}$ (resp. $\vec{b}$).

A history $h \in S^+$ is a finite (non-empty) sequence of states. The indistinguishability relations are extended to histories in a synchronous, pointwise manner, i.e., histories $h, h' \in S^+$ are *indistinguishable* for agent $i \in Ag$, or $h \sim_i h'$, iff (i) $|h| = |h'|$ and (ii) for all $j \leq |h|$, $h_j \sim_i h'_j$.

We now introduce the Alternating-time Temporal Logic ATL^* (Alur et al. 2002) to reason about strategic abilities

Definition 2 (ATL^*). *State (φ) and path (ψ) formulas in ATL^* are defined as follows, where $q \in AP$ and $\Gamma \subseteq Ag$:*

$$\begin{array}{rcl} \varphi & ::= & q \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\langle\Gamma\rangle\rangle\psi \\ \psi & ::= & \varphi \mid \neg\psi \mid \psi \wedge \psi \mid X\psi \mid (\psi U \psi) \end{array}$$

Formulas in ATL^* are all and only the state formulas.

A formula $\langle\langle\Gamma\rangle\rangle\psi$ is read as "coalition Γ has a strategy to achieve goal ψ". The meaning of linear-time operators *next* X and *until* U is standard; whereas operators $[\![\Gamma]\!]$, *release* R, *finally* F, and *globally* G can be introduced as usual (Baier and Katoen 2008). In particular, the language of the linear-time logic LTL corresponds to the path formulas in ATL^* built from atoms only. Hereafter we also consider the fragment of Γ*-formulas*, i.e., formulas in which the strategic operator $\langle\langle\Gamma\rangle\rangle$ ranges only over a given coalition $\Gamma \subseteq Ag$.

We interpret formulas in ATL^* by using *uniform strategies* (Jamroga and van der Hoek 2004).

Definition 3 (Strategy). *A* uniform strategy with perfect recall *for agent $i \in Ag$ is a function $f_i : S^+ \to Act_i$ such that for all histories $h, h' \in S^+$, (i) $f_i(h) \in d(i, last(h))$; and (ii) if $h \sim_i h'$ then $f_i(h) = f_i(h')$.*

By Def. 3, item (i), any strategy for agent i returns actions available to i; and by (ii), the same action is returned, whenever histories are indistinguishable for i.

Given an iCGS M, a *path* $p \in S^\omega$ is an infinite sequence $s_1 s_2 \ldots$ of states such that, for every $j \geq 1$, $s_{j+1} = \delta(s_j, \vec{a})$ for some joint action $\vec{a} \in ACT$. Given a joint strategy $F_\Gamma = \{f_i \mid i \in \Gamma\}$, a path p is F_Γ*-compatible* iff for every $j \geq 1$, $p_{j+1} = \delta(p_j, \vec{a})$ for some joint action $\vec{a} \in ACT$ such that for every $i \in \Gamma$, $a_i = f_i(p_{\leq j})$. Let $out(s, F_\Gamma)$ be the set of all F_Γ-compatible paths from s.

We now interpret ATL^* formulas on iCGS according to a semantics with two truth values: ff and tt.

Definition 4 (Satisfaction). *The two-valued satisfaction relation $\models^2$ for an iCGS M, state $s \in S$, path $p \in S^\omega$, atom $q \in AP$, and ATL^* formula ϕ is defined as follows (clauses for Boolean connectives are immediate and thus omitted):*

$$\begin{array}{lll} (M, s) \models^2 q & \textit{iff} & V(s, q) = \mathrm{tt} \\ (M, s) \models^2 \langle\langle\Gamma\rangle\rangle\psi & \textit{iff} & \textit{for some joint strategy } F_\Gamma, \\ & & \textit{for all paths } p \in out(s, F_\Gamma),\ (M, p) \models^2 \psi \\ (M, p) \models^2 \varphi & \textit{iff} & (M, p_1) \models^2 \varphi \\ (M, p) \models^2 X\psi & \textit{iff} & (M, p_{\geq 2}) \models^2 \psi \\ (M, p) \models^2 \psi U \psi' & \textit{iff} & \textit{for some } k \geq 1,\ (M, p_{\geq k}) \models^2 \psi', \textit{ and} \\ & & \textit{for all } j,\ 1 \leq j < k \Rightarrow (M, p_{\geq j}) \models^2 \psi \end{array}$$

A formula φ is true *in M, or $M \models^2 \varphi$, iff $(M, s_0) \models^2 \varphi$.*

We now state the model checking problem for the classic, two-valued semantics.

Definition 5 (Model Checking Problem). *Given an iCGS M and a formula ϕ, determine whether $M \models^2 \phi$.*

It is well-known that model checking formulas in ATL^* on iCGS with imperfect information and perfect recall is undecidable in general (Dima and Tiplea 2011). In the rest of the paper we describe a partial decision procedure; but first we illustrate the formal machine with a toy example.

Example 1. *In Fig. 1 we present a coordination game played by two trains t_1 and t_2, and a controller c at a junction. Train t_1 (resp. t_2) and c need to coordinate and select the same direction, left (L) or right (R), to move from the initial state s_I. After this first step, the controller can still change her mind. Specifically, she can either change arbitrarily the selection (E), request a new selection to the trains (A), or execute it (O). Further, train t_1 cannot observe the*

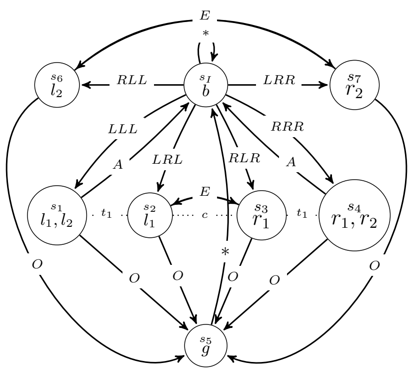

Figure 1: The iCGS M for Example 1. Notice that the transitions are generated by triples of actions. To improve readability, occurrences of the action *idle* (I) are omitted. Moreover, $*$ denotes any joint action for which a transition is not given explicitly. Further, dotted lines are used for indistinguishable states, each one labelled with the relevant agent.

choice of t_2, i.e., if t_1 chooses L (resp. R), then she cannot distinguish whether t_2 selects R or L, and c has partial observability on the choices of t_1 and t_2: she cannot distinguish the identities of t_1 and t_2, that is, she does not distinguish between the result of joint actions RLR and LRL[1]. *Finally, we use six atoms: b for the initial state s_I, l_1 (resp. l_2) for coordination on the left between t_1 (resp. t_2) and c, r_1 (resp. r_2) for coordination on the right between t_1 (resp. t_2) and c, and g to mark that the players have coordinated.*

As an example of specification in ATL^*, consider formula $\varphi = \langle\langle\Gamma\rangle\rangle F(l_1 \wedge \neg bUg)$, for $\Gamma = \{t_1, c\}$, which can be read as: t_1 and c have a joint strategy such that eventually they tentatively coordinate on the left, but then an agreement has to be reached before visiting the initial state again. Notice that φ is true in M by the joint strategy whereby t_1 chooses L in s_I and I in all other states, and c chooses L in s_I and O in all other states. However, we want to be able to model check such specifications in general.

3 Knowledge-based Abstraction

Given the undecidability of model checking ATL^* under imperfect information and perfect recall, in this section we present perfect information, three-valued abstractions of iCGS. While we keep the section self-contained, we refer to (Belardinelli et al. 2019) for full proofs. In particular, we remind that defined truth values of ATL^* formulas transfer from such abstractions to the original iCGS (Theorem 1).

Given a coalition $\Gamma \subseteq Ag$ of agents, the *common knowledge relation* $\sim^C_\Gamma$ is the reflexive and transitive closure $(\bigcup_{i\in\Gamma} \sim_i)^*$ of the union of indistinguishability relations $\sim_i$ for all $i \in \Gamma$ (Fagin et al. 1995). Clearly, $\sim^C_\Gamma$ is an equivalence relation. Then, let $[s]_\Gamma = \{s' \in S \mid s' \sim^C_\Gamma s\}$ be the equivalence class of state s according to $\sim^C_\Gamma$. The relation $\sim^C_\Gamma$ is extended to histories in a synchronous, pointwise manner, i.e., given $h, h' \in S^+$, $h \sim^C_\Gamma h'$ iff (i) $|h| = |h'|$ and (ii) for all $j \le |h|$, $h_j \sim^C_\Gamma h'_j$. So, we introduce the notation $[h]_\Gamma = \{h' \in S^+ \mid h' \sim^C_\Gamma h\}$.

[1]Note that, for simplicity, we considered imperfect information only on the side of the coordination between t_1 and c. To make the iCGS in Fig. 1 symmetric we only need to add $s_1 \sim_{t_2} s_6$, $s_4 \sim_{t_2} s_7$, and $s_6 \sim_c s_7$.

Definition 6 (Abstract CGS)**.** *Given an iCGS $M = \langle S, s_0, \{Act_i\}_{i\in Ag}, \{\sim_i\}_{i\in Ag}, d, \delta, V\rangle$ and coalition $\Gamma \subseteq Ag$, the* abstraction $M_\Gamma = \langle S_\Gamma, [s_0]_\Gamma, \{Act_i\}_{i\in Ag}, d^{may}_\Gamma, d^{must}_\Gamma, \delta^{may}_\Gamma, \delta^{must}_\Gamma, V_\Gamma\rangle$ *is such that:*

- *$S_\Gamma = \{[s]_\Gamma \mid s \in S\}$ is the set of equivalence classes for all states $s \in S$, with initial state $[s_0]_\Gamma$;*
- *for every $t, t' \in S_\Gamma$ and joint action $\vec{a}$, (i) $t' \in \delta^{may}_\Gamma(t, \vec{a})$ iff $\delta(s, \vec{a}) = s'$ for some $s \in t$, $s' \in t'$; (ii) $t' \in \delta^{must}_\Gamma(t, \vec{a})$ iff for all $s \in t$, there is $s' \in t'$ such that $\delta(s, \vec{a}) = s'$;*
- *for $t \in S_\Gamma$ and $i \in Ag$, $d^{must}_\Gamma(i, t) = \{a_i \in Act_i \mid$ for all $s \in t, a_i \in d(i, s)\}$; and $d^{may}_\Gamma(i, t) = \{a_i \in Act_i \mid \delta^{may}_\Gamma(t, (a_i, \vec{a}_{\bar{i}}))$ is defined for some $\vec{a}_{\bar{i}}\}$;*
- *for $v \in \{\mathrm{tt}, \mathrm{ff}\}$, $q \in AP$, and $t \in S_\Gamma$, $V_\Gamma(t, q) = v$ iff $V(s, q) = v$ for all $s \in t$; otherwise, $V_\Gamma(t, q) = \mathrm{uu}$.*

Intuitively, $must$-transitions in the abstract CGS are under-approximations of the transitions in the original iCGS; whereas may-transitions can be interpreted as over-approximations. The undefined value uu can be thought of as unknown, unspecified, or inconsistent. This is standard in multi-valued abstraction-based methods (Shoham and Grumberg 2004; Ball and Kupferman 2006) and we do not discuss this further. A truth value τ is *defined* if $\tau \neq \mathrm{uu}$.

To interpret formulas in ATL^* on three-valued abstractions, we introduce $must$- and may-strategies. In what follows, for $x = may$ (resp. $must$), $\overline{x} = must$ (resp. may).

Definition 7. *For $x \in \{may, must\}$, a x-*strategy *with* perfect recall *for agent $i \in Ag$ is a function $f^x_i : S^+ \to Act_i$ such that for every history $h \in S^+$, $f^x_i(h) \in d^x_\Gamma(i, last(h))$.*

For $x \in \{may, must\}$ and joint strategy $F^x_\Gamma = \{f^x_i \mid i \in \Gamma\}$, a path $p \in S^\omega$ is *F^x_Γ-compatible* iff for every $j \ge 1$, $p_{j+1} \in \delta^{\overline{x}}(p_j, \vec{a})$ for some joint action $\vec{a}$ such that for every $i \in \Gamma$, $a_i = f^x_i(p_{\le j})$. As in (Belardinelli et al. 2019), when we consider a may (resp. $must$) strategy for coalition Γ, we need to consider $must$ (resp. may) transitions in the model. Then, let $out(s, F^x_\Gamma)$ be the set of all F^x_Γ-compatible paths starting from state s.

Finally, we define the three-valued, perfect information semantics for ATL^* on abstractions as follows.

Definition 8 (Satisfaction)**.** *The three-valued satisfaction relation $\models^3$ for abstraction M_Γ, state $s \in S$, path $p \in S^\omega$, atom $q \in AP$, $v \in \{\mathrm{tt}, \mathrm{ff}\}$, and Γ-formula ϕ is defined as in Table 1. In all other cases the value of ϕ is* uu.

Then, $(M_\Gamma \models^3 \varphi) = \mathrm{tt}$ (resp. ff) iff $((M_\Gamma, s_0) \models^3 \varphi) = \mathrm{tt}$ (resp. ff). Otherwise, $(M_\Gamma \models^3 \varphi) = \mathrm{uu}$.

In the clauses for strategy operators $\langle\langle\Gamma\rangle\rangle$, $must$-strategies are used to check for truth, while may-strategies appear in the clauses for falsehood.

We now recall the preservation result from abstraction M_Γ to the original iCGS M.

$$
\begin{array}{ll}
((M_\Gamma, s) \models^3 q) = v & \text{iff } V_\Gamma(s, q) = v \\
((M_\Gamma, s) \models^3 \langle\langle\Gamma\rangle\rangle\psi) = \text{tt} & \text{iff for some } F_\Gamma^{must}, \text{ for all } p \in out(s, F_\Gamma^{must}),\ ((M_\Gamma, p) \models^3 \psi) = \text{tt} \\
((M_\Gamma, s) \models^3 \langle\langle\Gamma\rangle\rangle\psi) = \text{ff} & \text{iff for every } F_\Gamma^{may}, \text{ for some } p \in out(s, F_\Gamma^{may}),\ ((M_\Gamma, p) \models^3 \psi) = \text{ff} \\
((M_\Gamma, p) \models^3 \varphi) = v & \text{iff } ((M_\Gamma, p_1) \models^3 \varphi) = v \\
((M_\Gamma, p) \models^3 X\psi) = v & \text{iff } ((M_\Gamma, p_{\geq 2}) \models^3 \psi) = v \\
((M_\Gamma, p) \models^3 \psi U \psi') = \text{tt} & \text{iff for some } k \geq 1,\ ((M_\Gamma, p_{\geq k}) \models^3 \psi') = \text{tt, and for all } j,\ 1 \leq j < k \Rightarrow ((M_\Gamma, p_{\geq j}) \models^3 \psi) = \text{tt} \\
((M_\Gamma, p) \models^3 \psi U \psi') = \text{ff} & \text{iff for all } k \geq 1, \text{ either } ((M_\Gamma, p_{\geq k}) \models^3 \psi') = \text{ff or for some } j < k,\ ((M_\Gamma, p_{\geq j}) \models^3 \psi) = \text{ff.}
\end{array}
$$

Table 1: The three-valued, perfect information satisfaction relation for ATL^*. Boolean operators are interpreted as in Kleene's three-valued logic and therefore the corresponding clauses are omitted.

Theorem 1 ((Belardinelli et al. 2019)). *Given an iCGS M, state s, and coalition $\Gamma \subseteq Ag$, for every Γ-formula ϕ,*

$$((M_\Gamma, [s]_\Gamma) \models^3 \phi) = \text{tt} \quad \Rightarrow \quad (M, s) \models^2 \phi$$
$$((M_\Gamma, [s]_\Gamma) \models^3 \phi) = \text{ff} \quad \Rightarrow \quad (M, s) \not\models^2 \phi$$

Further, for abstract CGS we recall the decidability of the corresponding model checking problem.

Theorem 2 ((Belardinelli et al. 2019)). *The model checking problem for ATL^* on abstract CGS (with perfect information) is 2EXPTIME-complete.*

By combining Theorem 1 and 2 we can outline a method to verify the strategic abilities of agents under imperfect information and perfect recall. Given an iCGS M and a Γ-formula ϕ in ATL^*, we first build the abstract, three-valued CGS M_Γ according to Def. 6. We can model check ϕ on M_Γ, as the corresponding decision problem is decidable by Theorem 2, and then transfer any defined answer to the original iCGS M in virtue of Theorem 1. Unfortunately, if undefined (uu) is returned, then no conclusive answer can be drawn. In (Belardinelli et al. 2019) a procedure is provided to refine the abstraction in a conservative way. However, this refinement procedure assumes the existence of a "failure" state in which the truth value of the relevant formula is undefined, but no algorithm is given for finding such failure states.

In Sec. 5 we describe such an algorithm, but first we prove some general results on automata for three-valued LTL in Sec. 4. To conclude, we illustrate the abstraction procedure with our coordination game in Example 1.

Example 2. *In Fig. 2 we show the abstract CGS obtained from the iCGS in Example 1 by considering formula $\varphi = \langle\langle\Gamma\rangle\rangle F(l_1 \wedge \neg bUg)$ for $\Gamma = \{t_1, c\}$. Specifically, abstraction M_Γ includes five abstract states according to the equivalence relation $\sim^C_{\{t_1,c\}}$. Notice that formula φ is undefined in M_Γ due to the undefined value of atom l_1 in state a_2.*

4 Automata for Three-valued LTL

In this section we introduce an automata-theoretic approach to the verification of the three-valued linear-time logic LTL. We refer to (Baier and Katoen 2008) for a detailed presentation of LTL; here we observe that the syntax of LTL can be obtained from Def. 2 by considering as state formulas atoms only (i.e., $\varphi ::= q$). Then, the three-valued semantics for LTL follows from Table 1 by considering only the conditions concerning the operators in the syntax of LTL.

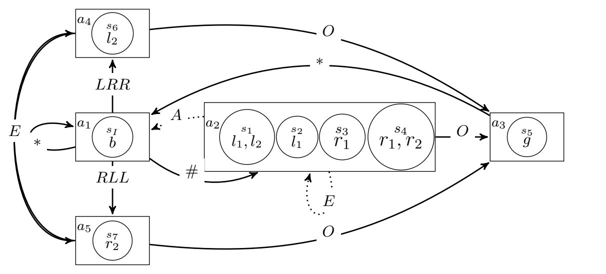

Figure 2: The abstract CGS for the iCGS in Example 1, where $\# \in \{LLL, LRL, RLR, RRR\}$, $must$-transitions are depicted with continuous lines, and may-transitions are the continuous and dashed lines.

These results will be used in Sec. 5, in procedure $FailureState()$ to find failure states. To this end, we build upon the standard, two-valued, automata-theoretic approach to the verification of LTL (Vardi 1995; Baier and Katoen 2008). We start by recalling that the syntax of LTL can be obtained by considering the path formulas in Def. 2, where state formulas φ are atoms q only. Then, the three-valued semantics of LTL is as in Def. 8, where again state formulas are atoms only.

Now, we recall the definition of generalized nondeterministic Büchi automata (GNBA).

Definition 9. *A GNBA is a tuple $A = \langle Q, Q_0, \Sigma, \delta, \mathcal{F}\rangle$ where (i) Q is a finite set of* states *with $Q_0 \subseteq Q$ as the set of* initial states*; (ii) Σ is an* alphabet*; (iii) $\delta : Q \times \Sigma \to 2^Q$ is the* transition relation*; (iv) $\mathcal{F}$ is a (possibly empty) subset of 2^Q. The elements in $\mathcal{F}$ are called* acceptance sets.

The accepted language *$L(A)$ consists of all infinite words $w \in \Sigma^\omega$ for which there exists at least one infinite run $q_0q_1q_2 \ldots \in Q^\omega$ such that for each acceptance set $F \in \mathcal{F}$ there are infinitely many indices i with $q_i \in F$.*

We now show that for every LTL formula ψ, there exists an automaton $A_{\psi,\text{uu}}$ that accepts exactly the infinite paths that evaluate ψ to undefined (uu). We first provide some definitions necessary for the construction.

Definition 10 (Closure and Elementarity). *The* closure *$cl(\psi)$ of an LTL formula ψ is the set consisting of all subformulas ϕ of ψ as well as their negation $\neg\phi$.*

Let $B \subseteq cl(\psi)$. Set B is consistent *w.r.t. propositional logic iff for all $\psi_1 \wedge \psi_2, \neg\neg\phi \in cl(\psi)$: (i) $\psi_1 \wedge \psi_2 \in B$ iff $\psi_1 \in B$ and $\psi_2 \in B$; (ii) if $\phi \in B$ then $\neg\phi \notin B$; (iii) $\neg\neg\phi \in B$ iff $\phi \in B$; (iv) if* tt $\in cl(\psi)$ *then* tt $\in B$.

Further, B is locally consistent *w.r.t. the until operator iff for all $\psi_1 U \psi_2 \in cl(\psi)$: (i) if $\psi_2 \in B$ then $\psi_1 U \psi_2 \in B$; (ii) if $\neg(\psi_1 U \psi_2) \in B$ then $\neg\psi_2 \in B$; (iii) if $\psi_1 U \psi_2 \in B$ and $\psi_2 \notin B$ then $\psi_1 \in B$; (iv) if $\neg\psi_1 \in B$ then $\neg(\psi_1 U \psi_2) \in B$ or $\neg\psi_2 \notin B$.*

Finally, B is elementary *iff it is both consistent and locally consistent.*

Notice that, differently from the standard construction for two-valued LTL (Baier and Katoen 2008), here we do not require elementary sets to be maximal, but we do require extra conditions (iii) on consistency, and (ii) and (iv) on local consistency. Hereafter $Lit = AP \cup \{\neg q \mid q \in AP\}$ is the set of *litterals*.

Definition 11. *Let ψ be a formula in LTL. We define the automaton $A_{\psi,\mathrm{uu}} = \langle Q, Q_0, 2^{Lit}, \delta, \mathcal{F}\rangle$ as follows:*

- *Q is the set of all elementary sets $B \subseteq cl(\psi)$ with $Q_0 = \{B \in Q \mid \psi \notin B$ and $\neg\psi \notin B\}$.*
- *The transition relation δ is given by: if $A \neq B \cap Lit$, then $\delta(B, A) = \emptyset$; otherwise $\delta(B, A)$ is the set of all elementary sets B' of formulas such that for every $X\phi, \psi_1 U \psi_2 \in cl(\psi)$: (i) $X\phi \in B$ iff $\phi \in B'$; (ii) $\neg X\phi \in B$ iff $\neg\phi \in B'$; (iii) $\psi_1 U \psi_2 \in B$ iff $\psi_2 \in B$ or, $\psi_1 \in B$ and $\psi_1 U \psi_2 \in B'$; (iv) $\neg(\psi_1 U \psi_2) \in B$ iff $\neg\psi_2 \in B$ and, $\neg\psi_1 \in B$ or $\neg(\psi_1 U \psi_2) \in B'$.*
- *$\mathcal{F} = \{F_{\psi_1 U \psi_2}, F_{\neg(\psi_1 U \psi_2)} \mid \psi_1 U \psi_2 \in cl(\psi)\}$, where $F_{\psi_1 U \psi_2} = \{B \in Q \mid$ if $\psi_1 U \psi_2 \in B$ then $\psi_2 \in B\}$ and $F_{\neg(\psi_1 U \psi_2)} = \{B \in Q \mid$ if $\neg(\psi_1 U \psi_2) \in B$ then $\neg\psi_1, \neg\psi_2 \in B\}$.*

By Def. 11 the transition relation works as follows: if the automaton reads a set A of literals that do not appear in the state, then the transition is not defined. Otherwise, the automaton checks the transitions enabled w.r.t. the semantics of the LTL operators. Notice that in Def. 11 we need to provide conditions on negated formulas as well, as elementary sets are not necessarily maximal. Moreover, we can define automata $A_{\psi,\mathrm{tt}}$ and $A_{\psi,\mathrm{ff}}$ by setting $Q_0^{\mathrm{tt}} = \{B \in Q \mid \psi \in B\}$ and $Q_0^{\mathrm{ff}} = \{B \in Q \mid \neg\psi \in B\}$ respectively. For both automata we can prove results similar to Theorem 3 below.

We now prove that the paths that evaluate ψ as undefined are exactly those included in the language of $A_{\psi,\mathrm{uu}}$. To prove this result, we make use of the following lemma.

Lemma 1. *Let run $B = B_1 B_2 \ldots$ in $A_{\psi,\mathrm{uu}}$ and path $p = p_1 p_2 \ldots$ in $(2^{Lit})^\omega$ satisfy (i) $B_{i+1} \in \delta(B_i, p_i)$, for all $i \geq 0$; and (ii) for all $F \in \mathcal{F}$, there exist infinitely many $j \geq 0$ such that $B_j \in F$. Then, for all $\phi, \neg\phi' \in cl(\psi)$, (a) $\phi \in B_1$ iff $(p \models \phi) = \mathrm{tt}$; and (b) $\neg\phi' \in B_1$ iff $(p \models \phi') = \mathrm{ff}$.*

Proof. The proof is by mutual induction on the structure of ϕ, $\neg\phi'$, where the induction hypothesis is that for all $i \geq 0$, $\phi \in B_i$ iff $(p_i p_{i+1} \ldots \models \phi) = \mathrm{tt}$ and $\neg\phi' \in B_i$ iff $(p_i p_{i+1} \ldots \models \phi') = \mathrm{ff}$. Due to limited space, we prove only (a), as (b) is proved by induction similarly. Notice that by construction, $\delta(B_i, p_i)$ is defined iff $p_i = B_i \cap Lit$.

Base case: The statement for $\phi = q \in AP$ follows directly from the fact that $(p_i p_{i+1} \ldots \models \phi) = \mathrm{tt}$ iff $q \in p_i = B_i \cap Lit$, iff $q \in B_i$.

Inductive steps: based on the induction hypothesis that the claim holds for formulas $\phi', \psi_1, \psi_2 \in cl(\psi)$, we need to prove that it also holds for $\phi = X\phi'$, $\phi = \neg\phi'$, $\phi = \psi_1 \wedge \psi_2$, and $\phi = \psi_1 U \psi_2$ in $cl(\psi)$. For reasons of space we provide details only for $\phi = \psi_1 U \psi_2$. Let $p = p_i p_{i+1} \ldots \in (2^{Lit})^\omega$ and $B_i B_{i+1} \ldots \in Q^\omega$ satisfy conditions (i) and (ii), we then show that $\phi \in B_i$ iff $(p_i p_{i+1} \ldots \models \phi) = \mathrm{tt}$.

$(\Leftarrow)$ Suppose that $(p_i p_{i+1} \ldots \models \psi_1 U \psi_2) = \mathrm{tt}$. Then, for some $j \geq i$, $(p_j p_{j+1} \ldots \models \psi_2) = \mathrm{tt}$ and $(p_k p_{k+1} \ldots \models \psi_1) = \mathrm{tt}$ for all $i \leq k < j$. By the induction hypothesis applied to ψ_1 and ψ_2, it follows that $\psi_2 \in B_j$ and $\psi_1 \in B_k$ for all $i \leq k < j$. Since B_j is elementary, $\psi_1 U \psi_2 \in B_j$ as well. Further, by definition of δ, we obtain that $\psi_1 U \psi_2 \in B_k$ for all $i \leq k < j$. In particular, $\psi_1 U \psi_2 \in B_i$.

$(\Rightarrow)$ Suppose that $\psi_1 U \psi_2 \in B_i$. Since B_i is elementary, then either $\psi_1 \in B_i$ or $\psi_2 \in B_i$. If $\psi_2 \in B_i$, it follows from the induction hypothesis that $(p_i p_{i+1} \ldots \models \psi_2) = \mathrm{tt}$, and thus $(p_i p_{i+1} \ldots \models \psi_1 U \psi_2) = \mathrm{tt}$. On the other hand, if $\psi_2 \notin B_i$, then both $\psi_1 \in B_i$ and $\psi_1 U \psi_2 \in B_i$. To obtain a contradiction suppose that $\psi_2 \notin B_j$ for all $j \geq i$. By the definition of δ, by using an inductive argument we have that $\psi_1 \in B_j$ and $\psi_1 U \psi_2 \in B_j$ for all $j \geq i$. Further, since $B_i B_{i+1} \ldots$ satisfies constraint (ii), we have $B_j \in F_{\psi_1 U \psi_2}$ for infinitely many $j \geq i$. On the other hand, we have $\psi_2 \notin B_j$ and $\psi_1 U \psi_2 \in B_j$ iff $B_j \notin F_{\psi_1 U \psi_2}$, which is a contradiction. Thus, $\psi_2 \in B_j$ for some $j > 0$. Assume that k is the smallest index such that $\psi_2 \in B_k$. By the induction hypothesis applied to ψ_1 and ψ_2 it follows $(p_k p_{k+1} \ldots \models \psi_2) = \mathrm{tt}$ and $(p_j p_{j+1} \ldots \models \psi_1) = \mathrm{tt}$ for all $i \leq j < k$. Hence $(p_i p_{i+1} \ldots \models \psi_1 U \psi_2) = \mathrm{tt}$. $\square$

Finally, we prove the main theoretical result in this section. Hereafter, $Paths(\psi, \mathrm{uu})$ is the set of paths $p \in (2^{Lit})^\omega$ such that $(p \models \psi) = \mathrm{uu}$.

Theorem 3. *For every LTL formula ψ there exists a GNBA $A_{\psi,\mathrm{uu}}$ (given as in Def. 11) s.t. $L(A_{\psi,\mathrm{uu}}) = Paths(\psi, \mathrm{uu})$.*

Moreover, the size of $A_{\psi,\mathrm{uu}}$ is exponential in the size of ψ.

Proof. Clearly, by Def. 11 the size of $A_{\psi,\mathrm{uu}}$ in terms of number of states is exponential in the size of ψ.

Then, we prove the set inclusions in both directions.

(1) Let $p = p_1 p_2 \ldots \in Paths(\psi, \mathrm{uu})$. For $i \geq 0$, define sets B_i of formulas as $\{\phi \in cl(\psi) \mid (p_i p_{i+1} \ldots \models \phi) = \mathrm{tt}\} \cup \{\neg\phi \in cl(\psi) \mid (p_i p_{i+1} \ldots \models \phi) = \mathrm{ff}\}$. Notice that every B_i is elementary, i.e., $B_i \in Q$. Now, we prove that $B_1 B_2 \ldots$ is an accepting run for p. Observe that $B_{i+1} \in \delta(B_i, p_i)$, since for all $i > 0$:

- $p_i = B_i \cap Lit$.
- For $X\phi \in cl(\psi)$, we have $X\phi \in B_i$ iff $(p_i p_{i+1} \ldots \models X\phi) = \mathrm{tt}$, iff $(p_{i+1} p_{i+2} \ldots \models \phi) = \mathrm{tt}$, iff $\phi \in B_{i+1}$.
- Similarly, $\neg X\phi \in B_i$ iff $(p_i p_{i+1} \ldots \models X\phi) = \mathrm{ff}$ iff $(p_{i+1} p_{i+2} \ldots \models \phi) = \mathrm{ff}$ iff $\neg\phi \in B_{i+1}$.
- For $\psi_1 U \psi_2 \in cl(\psi)$, we have $\psi_1 U \psi_2 \in B_i$ iff $(p_i p_{i+1} \ldots \models \psi_1 U \psi_2) = \mathrm{tt}$ iff $(p_i p_{i+1} \ldots \models \psi_2) = \mathrm{tt}$ or, $(p_i p_{i+1} \ldots \models \psi_1) = \mathrm{tt}$ and $(p_{i+1} p_{i+2} \ldots \models \psi_1 U \psi_2) = \mathrm{tt}$, iff $\psi_2 \in B_i$ or, $\psi_1 \in B_i$ and $\psi_1 U \psi_2 \in B_{i+1}$.

- Similarly, $\neg(\psi_1 U \psi_2) \in B_i$ iff $(p_i p_{i+1} \ldots \models \psi_1 U \psi_2) = \text{ff}$ iff $(p_i p_{i+1} \ldots \models \psi_2) = \text{ff}$ and, $(p_i p_{i+1} \ldots \models \psi_1) = \text{ff}$ or $(p_{i+1} p_{i+2} \ldots \models \psi_1 U \psi_2) = \text{ff}$ iff $\neg\psi_2 \in B_i$ and, $\neg\psi_1 \in B_i$ or $\neg(\psi_1 U \psi_2) \in B_{i+1}$.

The above shows that $B_1 B_2 \ldots$ is a run in $A_{\psi,\text{uu}}$. Now, we need to prove that it is accepting, i.e., for each subformula $\psi_{1,j} U \psi_{2,j} \in cl(\psi)$, $B_i \in F_j$ for infinitely many i and for each subformula $\neg(\psi_{1,x} U \psi_{2,x}) \in cl(\psi)$, $B_y \in F_x$ for infinitely many y. We prove this point by contradiction. Consider there are finitely many i such that $B_i \in F_j$, then $B_i \notin F_j = F_{\psi_{1,j} U \psi_{2,j}}$ implies that $\psi_{1,j} U \psi_{2,j} \in B_i$ and $\psi_{2,j} \notin B_i$. By considering how B_i is constructed, we have that $(p_i p_{i+1} \ldots \models \psi_{1,j} U \psi_{2,j}) = \text{tt}$ and $(p_i p_{i+1} \ldots \models \psi_{2,j}) \neq \text{tt}$. In particular, for some $k > i$ we have $(p_k p_{k+1} \ldots \models \psi_{2,j}) = \text{tt}$. By the definition of B_i, it follows that $\psi_{2,j} \in B_k$, and by definition of F_j, $B_k \in F_j$. So, if $B_i \in F_j$ for finitely many i, then $B_k \in F_j$ for infinitely many k, which is a contradiction. The case for subformulas $\neg(\psi_{1,x} U \psi_{2,x}) \in cl(\psi)$ is proved similarly.

(2) Let $p = p_1 p_2 \ldots \in L(A_{\psi,\text{uu}})$, i.e., there is an accepting run $B_1 B_2 \ldots$ for p in $A_{\psi,\text{uu}}$. By the definition of $A_{\psi,\text{uu}}$, we have that $\delta(B, A) = \emptyset$ for all pairs (B, A) with $A \neq B \cap Lit$. Then, it follows that $p_i = B_i \cap Lit$ for all $i \geq 0$. Thus, $p = (B_1 \cap Lit)(B_2 \cap Lit) \ldots$ and we need to prove that $((B_1 \cap Lit)(B_2 \cap Lit) \ldots \models \psi) = \text{uu}$. But, this follows by Lemma 1 and the fact that neither ψ nor $\neg\psi$ belong to B_1. ☐

We conclude by recalling that to obtain a GNBA accepting all paths that make true (resp. false) a given LTL formula, it suffices to modify the set of initial states to $Q_0^{\text{tt}} = \{B \in Q \mid \psi \in B\}$ (resp. $Q_0^{\text{ff}} = \{B \in Q \mid \neg\psi \in B\}$).

5 Finding Failure States

In Sec. 3 we mentioned that the refinement procedure in (Belardinelli et al. 2019) takes as input a "failure" state s_f in which some subformula of the specification to be checked is undefined. However, no hint is given as to how to find such state s_f. Hereafter we tackle this problem, but first we recall the notion of failure state from (Ball and Kupferman 2006).

Definition 12 (Failure State). *A state s is a* failure state with respect to formula φ *iff $((M, s) \models^3 \varphi) = \text{uu}$ and, either $\varphi = q \in AP$, or $\varphi = \langle\langle\Gamma\rangle\rangle\psi$ and $((M, p) \models^3 \psi) \in \{\text{tt}, \text{ff}\}$ for every path p starting from s.*

Intuitively, s is a failure state with respect to φ iff $((M, s) \models^3 \varphi) = \text{uu}$ even though M has definite truth values for all subformulas of φ in the relevant states.

To introduce the procedure to find failure states, we first define the product between abstract CGS and GNBA.

Definition 13 (Product). *Given an abstract CGS $M = \langle S, s_0, \{Act_i\}_{i \in Ag}, d^{may}, d^{must}, \delta^{may}, \delta^{must}, V\rangle$ and a GNBA $A = \langle Q, \Sigma, \delta, Q_0, \mathcal{F}\rangle$, their product $M \otimes A = \langle S \times Q, \overline{S}_0, \{Act_i\}_{i \in Ag}, \overline{d^{may}}, \overline{d^{must}}, \overline{\delta^{may}}, \overline{\delta^{must}}, \overline{V}\rangle$ is s.t. for $s, t \in S$, $q, q' \in Q$, $q_0 \in Q_0$, and $x \in \{may, must\}$.*

- $\overline{S}_0 = \{(s_0, q) \mid q \in \delta(q_0, V(s_0))\}$;
- $\overline{d^x}((s, q)) = d^x(s)$;
- $\overline{\delta^x}((s, q), \alpha) = (t, q')$ *iff* $\delta^x(s, \alpha) = t$ *&* $q' \in \delta(q, V(t))$;
- $\overline{V}(s, q) = q$;

Algorithm 1 $FailureState(s, \varphi)$

```
1:  if $\varphi = q$ then
2:      return $(s, \varphi)$
3:  if $\varphi = \neg\varphi'$ then
4:      return $FailureState(s, \varphi')$
5:  if $\varphi = \varphi_1 \wedge \varphi_2$ then
6:      Let $i = min\{1, 2\}$ such that $((M, s) \models^3 \varphi_i) = \text{uu}$
        return $FailureState(s, \varphi_i)$
7:  if $\varphi = \langle\langle\Gamma\rangle\rangle\psi$ then
8:      if $Paths(M \otimes A_{\psi,\text{uu}}) = \emptyset$ then
9:          return $(s, \varphi)$
10:     else
11:         return $FailurePath(p \in Paths(M \otimes A_{\psi,\text{uu}})_{|S}, \psi)$
```

Algorithm 2 $FailurePath(p, \psi)$

```
1:  if $\psi = \varphi$ then
2:      return $FailureState(p_1, \varphi)$
3:  if $\psi = \neg\psi'$ then
4:      return $FailurePath(p, \psi')$
5:  if $\psi = \psi_1 \wedge \psi_2$ then
6:      Let $i = min\{1, 2\}$ such that $((M, p) \models^3 \psi_i) = \text{uu}$
        return $FailurePath(p, \psi_i)$
7:  if $\psi = X\psi'$ then
8:      return $FailurePath(p_{\geq 2}, \psi')$
9:  if $\psi = \psi_1 U \psi_2$ then
10:     $check_{\psi_1} = check_{\psi_2} = true$; $i = 0$
11:     while $check_{\psi_1} = true \wedge check_{\psi_2} = true$ do
12:         $i = i + 1$
13:         if $((M, p_{\geq i}) \models^3 \psi_2) = \text{uu}$ then
14:             $check_{\psi_2} = false$
15:         else if $((M, p_{\geq i}) \models^3 \psi_1) = \text{uu}$ then
16:             $check_{\psi_1} = false$
17:     if $check_{\psi_2} = false$ then
18:         return $FailurePath(p_{\geq i}, \psi_2)$
19:     else
20:         return $FailurePath(p_{\geq i}, \psi_1)$
```

The procedure $FailureState()$ to find failure states and relevant subformulas is depicted in Algorithm 1: $FailureState(s, \varphi)$ takes as input a state s and a formula φ (with at most one strategic operator) such that $((M, s) \models^3 \varphi) = \text{uu}$ and returns state s' and subformula φ' of φ.

We can check that the procedure $FailureState()$ is sound.

Proposition 1. *Suppose that $((M, s) \models^3 \varphi) = \text{uu}$. If $FailureState(s, \varphi) = (s', \varphi')$ then s' is a failure state.*

Proof. We prove the soundness of $FailureState()$ by induction. Given a model M, state s, and formula φ with no nested strategy operators, the algorithm $FailureState(s, \varphi)$ starts by considering the base case in which φ is an atom (lines 1-2). Here, φ is a failure state since the atom q is undefined on it. For the inductive step, we have the following cases. In the case of Boolean operators (lines 3-6), the procedure propagates over subformulas. To deal with the strategic operator (lines 7-11), the algorithm checks whether there is a path in the product $M \otimes A_{\psi,\mathrm{uu}}$ (Def. 13). The product between model M and automaton $A_{\psi,\mathrm{uu}}$ accepts all paths in M that make the subformula ψ undefined. If there is no such path, then the procedure returns the current state and formula. Otherwise, procedure $\mathit{FailurePath}(p, \varphi)$ in Algorithm 2 is called, where p is a path consistent with the product of M and $A_{\psi,\mathrm{uu}}$, i.e., $p \in \mathit{Paths}(M \otimes A_{\psi,\mathrm{uu}})_{|S}$[2]. In procedure $\mathit{FailurePath}(p, \varphi)$ the base case for state formulas (lines 1-2) returns to $\mathit{FailureState}()$ by taking as input the first state of path p. In lines 3-6 $\mathit{FailurePath}()$ handles the Boolean operators, and in lines 7-8 solves the next operator in accordance with its semantics. The main point of interest is the until operator U (lines 9-20). To prove that the **while** loop on line 11 terminates, we make use of Lemma 2 below, whereby we can show that the case of the until operator U in procedure $\mathit{FailurePath}()$ terminates after a finite number of step. □

Lemma 2. *Consider an abstract CGS M, path p, and formula* $\varphi = \psi U \psi'$. *If* $((M, p) \models^3 \varphi) = \mathrm{uu}$ *then for some* $i \geq 0$, *either* $((M, p_{\geq i}) \models^3 \psi) = \mathrm{uu}$ *or* $((M, p_{\geq i}) \models^3 \psi') = \mathrm{uu}$.

Proof. We prove the lemma by contradiction. Suppose that $((M, p) \models^3 \varphi) = \mathrm{uu}$ and for all $i \geq 0$, for some $v, v' \in \{\mathrm{tt}, \mathrm{ff}\}$, $((M, p_{\geq i}) \models^3 \psi) = v$ and $((M, p_{\geq i}) \models^3 \psi') = v'$. We then consider the following cases:

1. If $((M, p_{\geq i}) \models^3 \psi') = \mathrm{ff}$ (resp. tt) for all $i \geq 0$, then by the three-valued semantics we have $((M, p) \models^3 \varphi) = \mathrm{ff}$ (resp. tt), which is a contradiction.
2. If (1) is not the case, then formula ψ' is sometimes true and sometimes false, but always defined by assumption. Consider the smallest $i \geq 0$ such that $((M, p_{\geq i}) \models^3 \psi') = \mathrm{tt}$. Then, we can only have one of the following:
   (a) If for all $1 \leq j < i$, $((M, p_{\geq j}) \models^3 \psi) = \mathrm{tt}$, then by the three-valued semantics we have $((M, p) \models^3 \varphi) = \mathrm{tt}$, which is a contradiction.
   (b) Otherwise, there exists $1 \leq j < i$ such that $((M, p_{\geq j}) \models^3 \psi) = \mathrm{ff}$. Since we assumed that i is the smallest natural number for which ψ' is true, then for all $1 \leq k < j$ we have $((M, p_{\geq k}) \models^3 \psi') = \mathrm{ff}$. Hence, by the three-valued semantics it follows that $((M, p) \models^3 \varphi) = \mathrm{ff}$, which is again a contradiction. □

In Algorithm 3 we report the high-level iterative model checking procedure. Given an iCGS M, state s, and Γ-formula φ to check, we first construct the abstract CGS M_Γ based on M and Γ. Then, we model check formula φ in the abstract state $s_\Gamma \supseteq s$, which is decidable by Theorem 2. If a defined truth value is returned, by Theorem 1 we transfer this result to the original model checking problem. On the other hand, if $((M_\Gamma, s_\Gamma) \models^3 \varphi) = \mathrm{uu}$ then we use a bottom-up procedure (lines 4-13). We start by checking for each state if the innermost formula having a strategic operator is undefined (line 7). If this is the case, we call procedure $\mathit{FailureState}(s_\Gamma, \varphi)$ to find failure state s_f that "makes" the formula φ_i undefined (line 7). Then, in line 9 we call the function $\mathit{Refinement}(M_\Gamma, M, s_f)$ that is a slight variant of the refinement procedure in (Belardinelli et al. 2019) with s_f as input. Intuitively, we look at incoming transitions into s_f. For concrete states s and s' in s_f, if the Γ-component of actions ending respectively in s and s' are different, any uniform strategy for Γ will visit either s or s'. As a result, the abstract state s_f can be split "safely" into an s- and an s'-component. More precisely, the procedure $\mathit{Refinement}()$, shown in Algorithm 4, begins by initializing as true the values of a matrix m that stores the relation outlined above between the concrete states in s_f (lines 1-2). Then, the algorithm calls the subroutine $\mathit{Check}_1(M_\Gamma, M, s_f, m)$, shown in Algorithm 5, which updates the values in m by considering the concrete transition function δ in M. In particular, at each iteration $\mathit{Check}_1()$ considers one predecessor t_f of s_f (line 1). Then, two other loops (lines 2-3) consider pairs of states

Algorithm 3 $\mathit{ModelCheckingProcedure}(M, s, \varphi, \Gamma)$

```
 1: $(M_\Gamma, s_\Gamma) = \mathit{Abstraction}(M, \Gamma)$
 2: if $((M_\Gamma, s_\Gamma) \models^3 \varphi) \neq \mathrm{uu}$ then
 3:     return $((M_\Gamma, s_\Gamma) \models^3 \varphi)$
 4: $i = 0$; $\varphi_i = \mathit{InnermostFormula}(\varphi)$
 5: while $\varphi_i \neq \varphi$ do
 6:     for $s'_\Gamma \in S_\Gamma$ do
 7:         while $((M_\Gamma, s'_\Gamma) \models_3 \varphi_i) = \mathrm{uu} \wedge \mathit{split} = \mathit{true}$ do
 8:             $(s_f, \psi) = \mathit{FailureState}(s'_\Gamma, \varphi_i)$
 9:             $(M_\Gamma, \mathit{split}) = \mathit{Refinement}(M_\Gamma, M, s_f)$
10:     $\mathit{Add}(\mathit{atom}_{\varphi_i}, L(s))$
11:     $M_\Gamma = \mathit{UpdateModel}(M_\Gamma, \mathit{atom}_{\varphi_i})$
12:     $\varphi = \mathit{UpdateFormula}(\varphi, \mathit{atom}_{\varphi_i})$
13:     $i = i + 1$; $\varphi_i = \mathit{InnermostFormula}(\varphi)$
14: while $((M_\Gamma, s_\Gamma) \models^3 \varphi) = \mathrm{uu} \wedge \mathit{split} = \mathit{true}$ do
15:     $(s_f, \psi) = \mathit{FailureState}(s_\Gamma, \varphi_i)$
16:     $(M_\Gamma, \mathit{split}) = \mathit{Refinement}(M_\Gamma, M, s_f)$
17: if $\mathit{split} = \mathit{true}$ then
18:     return $(M_\Gamma, s_\Gamma) \models^3 \varphi$
19: else
20:     return "No refinement is available, $\varphi$ remains uu"
```

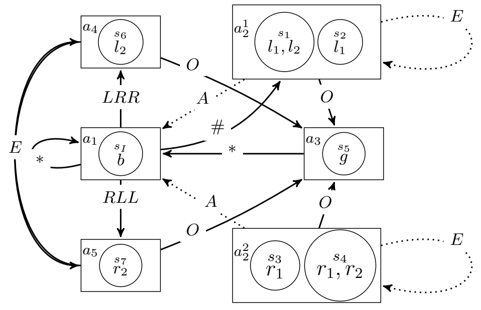

Figure 3: The refinement for the CGS in Example 2.

[2]The restriction $|_S$ means that path p is a sequence of states in which each state is only the first component of the product $M \otimes A_{\psi,\mathrm{uu}}$, i.e., a state in M.

Algorithm 4 *Refinement*(M_Γ, M, s_f)

```
1: for $s, s' \in s_f$ do
2:    $m[s,s'] = true$
3: $Check_1(M_\Gamma, M, s_f, m)$; $update = true$
4: while $update = true$ do
5:    $Check_2(M_\Gamma, s_f, m, update)$
6: $split = false$
7: while $s, s' \in s_f \land split = false$ do
8:    if $m[s,s'] = true$ then
9:       $split = true$; $remove(s_f, S_\Gamma)$
10:      $add(v, S_\Gamma)$; $add(w, S_\Gamma)$; $add(s, v)$; $add(s', w)$
11:      for $t \in s_f$ do
12:         if $m[s,t] = true$ then
13:            $add(t, w)$
14:         else
15:            $add(t, v)$
16: return $(M_\Gamma, split)$
```

Algorithm 5 $Check_1(M_\Gamma, M, s_f, m)$

```
1: for $t_f \in Pre(s_f)$ do
2:    for $s, s' \in s_f$ do
3:       for $t, t' \in t_f$ do
4:          if $\delta(t, \vec{a}) = s \land \delta(t', \vec{b}) = s'$ then
5:             for $i \in \Gamma$ do
6:                if $s \sim_i s' \land \vec{a}_i = \vec{b}_i$ then
7:                   $m[s,s'] = false$
```

Algorithm 6 $Check_2(M_\Gamma, s_f, m, update)$

```
1: $update = false$
2: for $s, s' \in s_f$ do
3:    if $m[s,s'] = true$ then
4:       for $t \in s_f$ do
5:          if $m[s,t] = false \land m[s',t] = false$ then
6:             $m[s,s'] = false$; $update = true$
```

s and s' in the abstract state s_f and pairs of states t and t' in the predecessor t_f. If s and s' are indistinguishable for some agent $i \in \Gamma$ and i performs the same action in the transitions from t and t' to s and s' respectively, then we update the value of the corresponding cell in m to false (lines 4-7). The subroutine $Check_1()$ carries out the first round of updates on m. Further updates in the *Refinement*() algorithm are performed by the subroutine $Check_2(M_\Gamma, s_f, m, update)$, shown in Algorithm 6, which considers the "indirect" binding that some concrete states may have in an abstract state. Specifically, given the states s and s' in the abstract state s_f that have $true$ as value in m (lines 2-3), we need to consider the relation that s and s' have with the other states in s_f: if the values in m for both states related with some other state t are $false$, then we update the value of cell $m[s,s']$ to $false$ as well (lines 4-6). Subroutine $Check_2()$ is called repeatedly in algorithm *Refinement*() as long as guard $update$ remains $true$, i.e., until we have at least an update in each call of the procedure. When $update$ becomes $false$, we proceed to check whether there is at least an element $true$ in m (line 8). If this is the case, we can split s_f. So, we assign the related concrete states s and s' to two different, new abstract states v and w (line 10). Finally, we populate the new abstract states v and w with the other concrete states in the old abstract state s_f (which is removed) according to matrix m (lines 11-15). When the loop in lines 7-9 of the $ModelCheckingProcedure()$ is concluded, we update the structure (lines 10-11) and the formula (line 12) and continue with the new innermost formula (line 13). This part of the procedure (lines 5-13) terminates when we have φ with a single, outermost strategic operator. So, we can check formula φ on s_Γ. If this formula is undefined, we use a loop (lines 14-16) that calls procedure *FailureState*(s_Γ, φ) to find failure state s_f making formula φ undefined. Then, in line 16 we call the refinement procedure with s_f as input. When the **while** loop in lines 14-16 is terminated, we check the truth value of the boolean variable *split* that is returned by the refinement procedure (line 17). We recall that *split* is true if and only if the model has been refined. If this is the case, the **while** loop is exited, as the last refinement step made the formula φ defined and then by Theorem 1 and 2 we transfer the defined truth value to the original model checking problem (line 18). On the other hand, if *split* is false, it was not possible to refine the model in a way to make the formula defined with our procedure (lines 19-20).

Example 3. *As an example of the application of procedure FailureState() we consider as input formula* $\varphi = \langle\langle\Gamma\rangle\rangle F(l_1 \land \neg bUg)$ *and state* a_1 *in Fig. 2. In Example 2 we observed that* $((M_\Gamma, a_1) \models^3 \varphi) = \text{uu}$. *Since the main operator in* φ *is the strategic modality* $\langle\langle\Gamma\rangle\rangle$, *procedure FailureState() goes to line 7. In particular, it constructs the automaton* $A_{\psi,\text{uu}}$ *that accepts all paths where* $\psi = true\, U(l_1 \land \neg bUg)$ *is undefined. Now, the language of the product between model* M_Γ *and* $A_{\psi,\text{uu}}$ *is not empty. Hence, the procedure calls FailurePath() with input, for instance,* $p = a_1a_2a_3\ldots$, *i.e., one of the paths in the product. Since formula* ψ *has until U as the main operator, the procedure goes to line 9. Now,* $((M_\Gamma, p_{\geq 2}) \models^3 \psi) = \text{uu}$ *and we call FailurePath*$(p_{\geq 2}, \psi')$ *with* $\psi' = l_1 \land \neg bUg$. *Observe that the main operator in* ψ' *is* $\land$ *and therefore we go to lines 5-6. Here,* $((M_\Gamma, p_{\geq 2}) \models^3 l_1) = ((M_\Gamma, p_2) \models^3 l_1) = \text{uu}$, *then we call FailurePath*$(p_{\geq 2}, l_1)$, *and by lines 1-2, FailureState*(p_2, l_1) *finally returns failure state* a_2 *and atom* l_1. *This ends the* $FailureState()$ *procedure. So, the* $ModelCheckingProcedure()$ *calls the* $Refinement()$ *procedure. Here, given* a_2 *as failure state, the* $Refinement()$ *procedure splits state* a_2 *in new states* a_2^1 *with concrete states* s_1 *and* s_2, *and* a_2^2 *with concrete states* s_3 *and* s_4 *as in Fig. 3. In the new model formula* φ *is defined (specifically, true) and this ends the whole procedure.*

5.1 Complexity Results

We conclude this section by discussing the complexity of our model checking procedure. First, notice that $ModelCheckingProcedure()$ does not necessarily terminate *with a defined truth value*. Indeed, the ATL^* model checking problem in case of imperfect information and perfect recall is undecidable in general. This is meant to be a sound, albeit partial, verification algorithm.

Theorem 4. *$ModelCheckingProcedure()$ terminates in 2EXPTIME.*

Proof. We analyze in detail Algorithm 3. The procedure of abstraction has to explore a polynomial number of states to generate the abstract model. Since the abstraction procedure returns a CGS with perfect information, the verification of ATL^* formulas can be performed by using the automata-theoretic techniques in (Alur et al. 2002) for instance. This leads to a model checking procedure in 2EXPTIME. The loops in lines 5-13 explore a polynomial number of formulas (**while** in line 5), a polynomial number states (**for** in line 6), and a polynomial number of operation on the model (**while** in line 7). The last loop is shown to be polynomial by variable $split$ that guarantees termination. Further, procedure $FailurePath()$ explores a polynomial number of states and formulas and procedure $FailureState()$ builds a formula automaton by using polynomial space. So, it appears that the refinement procedure (lines 8-9) can be performed in PSPACE. By considering the fact that in lines 14-16 we have again the refinement procedure, we can conclude that the whole complexity of our procedure is in 2EXPTIME. □

By Theorem 4 the complexity of our partial model checking procedure is high. However, we claim that it is still better than the general undecidability result.

6 Related Work and Conclusions

Recently several approaches to the verification of ATL^* under imperfect information and perfect recall have been put forward. Typically, these contributions assume restrictions either on the syntax or the semantics of the specification language, or develop abstraction and approximation methods. In the first line, decidability results have been proved for hierachical (Aminof et al. 2012; Berthon et al. 2017) and broadcast systems (Belardinelli et al. 2017b),(2017a). In the second line, techniques to construct syntactic (Bulling and Jamroga 2011) and semantic (Belardinelli et al. 2018) approximations have been investigated. Our contribution falls in the second line, specifically semantic approximations, even though it differs from (Belardinelli et al. 2018), where memory is abstracted to achieve decidability, as here we approximate information instead.

More closely related is a series of works on three-valued abstractions for temporal and strategy logics. An abstraction-refinement framework for CTL on a three-valued semantics was studied in (Shoham and Grumberg 2004),(2007), then extended to the μ-calculus in (Grumberg et al. 2007). As regards ATL, three-valued abstractions have also been put forward in (Ball and Kupferman 2006; Lomuscio and Michaliszyn 2014),(2015; 2016). However, there are considerable differences between these approaches and the one here pursued. In fact, their methods focus on settings of perfect information, and (Lomuscio and Michaliszyn 2014)(2015; 2016) considers *non-uniform* strategies (Raimondi and Lomuscio 2005), whereby the corresponding model checking problem is decidable. Their aim, therefore, is to speed-up the verification task and not, as we do here, to provide a sound, albeit partial, procedure for an undecidable problem. Moreover, we consider the full language ATL^*, while the references above only deal with its fragment ATL (Alur et al. 2002).

Regarding the multi-valued automata technique for LTL used in this work, we now discuss the differences w.r.t. (Kupferman and Lustig 2007; Chechik et al. 2001; Bruns and Godefroid 2003). In particular, (Bruns and Godefroid 2003) consider a reduction from multi-valued to two-valued LTL, but they do not provide automata-theoretic techniques. On the other hand, (Chechik et al. 2001) present an automata-theoretic approach to general multi-valued LTL following the tableau-based construction in (Gerth et al. 1995). Also (Kupferman and Lustig 2007) is devoted to general multi-valued automata. Specifically, the authors define lattices, deterministic and non-deterministic automata, as well as their extensions to Büchi acceptance conditions. As an application of their theoretical results, they provide an automata construction for multi-valued LTL, but only in passing, without a clear explanation of states and transitions. To sum up, differently from (Kupferman and Lustig 2007; Chechik et al. 2001; Bruns and Godefroid 2003), the approach we proposed here modifies minimally the automata-theoretic construction for two-valued LTL in (Baier and Katoen 2008) and extends it to a three-valued interpretation. In this sense we claim that our contribution is novel w.r.t. the current literature. Furthermore, it is not clear how the techniques in (Kupferman and Lustig 2007; Chechik et al. 2001; Bruns and Godefroid 2003) could be used in our construction. As mentioned above, (Bruns and Godefroid 2003) does not really deal with LTL. The approach in (Chechik et al. 2001) is more suitable for on-the-fly verification. Finally, in (Kupferman and Lustig 2007) the authors only briefly discuss model checking, and their approach is tailored more generally for multi-valued logics.

Finally, we mentioned that the present work builds upon (Belardinelli et al. 2019), where a three-valued abstraction and refinement procedure for ATL^* is presented. We observed that the refinement procedure takes a failure state as input, but in (Belardinelli et al. 2019) no method was provided to find such failure states. Here we presented such an algorithm that, differently from the state of the art (Ball and Kupferman 2006), operates on the whole ATL^*, under imperfect information. To this end, we developed automata-theoretic techniques for LTL in a three-valued sematics, that we deem of independent interest.

As future work we intend to build a toolkit to generate abstractions and refinements automatically. Any such toolkit will require the novel implementation of the three valued semantics here described and will therefore constitute a substantial undertaking. Another interesting question that we would like to explore as future work is to find the "most promising" failure states. It might be possible to find robust heuristics to find good candidates for refinement. Finally, we plan to extend the verification techniques here developed to more expressive languages including Strategy Logic (Chatterjee et al. 2007; Mogavero et al. 2014) in the light of the recent comparison results in (Belardinelli et al. 2019).

Acknowledgments

F. Belardinelli acknowledges the support of ANR JCJC Project SVeDaS (ANR-16-CE40-0021).

References

Ågotnes, T.; Goranko, V.; Jamroga, W.; and Wooldridge, M. 2015. Knowledge and ability. In *Handbook of Logics for Knowledge and Belief*. 543–589.

Alur, R.; Henzinger, T.; Mang, F.; Qadeer, S.; Rajamani, S.; and Tasiran, S. 1998. MOCHA: Modularity in model checking. In *CAV98*, 521–525.

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

Aminof, B.; Kupferman, O.; and Murano, A. 2012. Improved model checking of hierarchical systems. *Inf. Comput.* 210:68–86.

Baier, C., and Katoen, J. P. 2008. *Principles of Model Checking*. MIT Press.

Ball, T., and Kupferman, O. 2006. An abstraction-refinement framework for multi-agent systems. In *LICS06*, 379–388.

Belardinelli, F.; Jamroga, W.; Kurpiewski, D.; Malvone, V.; and Murano, A. 2019. Strategy Logic with Simple Goals: Tractable Reasoning about Strategies. In *IJCAI19*, 88–94.

Belardinelli, F.; Lomuscio, A.; Murano, A.; and Rubin, S. 2017a. Verification of broadcasting multi-agent systems against an epistemic strategy logic. In *IJCAI17*, 91–97.

Belardinelli, F.; Lomuscio, A.; Murano, A.; and Rubin, S. 2017b. Verification of multi-agent systems with imperfect information and public actions. In *AAMAS17*, 1268–1276.

Belardinelli, F.; Lomuscio, A.; and Malvone, V. 2018. Approximating perfect recall when model checking strategic abilities. In *KR2018*, 435–444.

Belardinelli, F.; Lomuscio, A.; and Malvone, V. 2019. An abstraction-based method for verifying strategic properties in multi-agent systems with imperfect information. In *AAAI19*, 6030–6037.

Berthon, R.; Maubert, B.; Murano, A.; Rubin, S.; and Vardi, M. Y. 2017. Strategy logic with imperfect information. In *LICS*, 1–12.

Berthon, R.; Maubert, B.; and Murano, A. 2017. Decidability results for ATL* with imperfect information and perfect recall. In *AAMAS17*, 1250–1258.

Bruns, G., and Godefroid, P. 2003. Model checking with multi-valued logics. Technical Report ITD-03-44535H, Bell Labs.

Bulling, N., and Jamroga, W. 2011. Alternating epistemic mu-calculus. In *IJCAI11*, 109–114.

Chatterjee, K.; Henzinger, T.; and Piterman, N. 2007. Strategy logic. In *CONCUR07*, 59–73.

Chechik, M.; Devereux, B.; and Gurfinkel, A. 2001. Model-checking in finite state-space systems with fine-grained abstractions using spin. In *SPIN*, 16–36.

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

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

Gerth, R.; Peled, D.; Vardi, M.; and Wolper, P. 1995. Simple on-the-fly automatic verification of linear temporal logic. In *PSTV95*, 3–18.

Grumberg, O.; Lange, M.; Leucker, M.; and Shoham, S. 2007. When not losing is better than winning: Abstraction and refinement for the full mu-calculus. *Inf. Comput.* 205(8):1130–1148.

Jamroga, W., and Dix, J. 2006. Model checking abilities under incomplete information is indeed Δ_p^2-complete. In *EUMAS06*, 14–15.

Jamroga, W., and van der Hoek, W. 2004. Agents that know how to play. *Fund. Inf.* 62:1–35.

Jamroga, W. 2018. Logical methods for specification and verification of multi-agent systems.

Kupferman, O., and Lustig, Y. 2007. Lattice automata. In *VMCAI07*, 199–213.

Kurpiewski, D.; Jamroga, W.; and Knapik, M. 2019. STV: model checking for strategies under imperfect information. In *AAMAS19*, 2372–2374.

Lomuscio, A., and Michaliszyn, J. 2014. An abstraction technique for the verification of multi-agent systems against ATL specifications. In *KR14*, 428–437.

Lomuscio, A., and Michaliszyn, J. 2015. Verifying multi-agent systems by model checking three-valued abstractions. In *AAMAS15*, 189–198.

Lomuscio, A., and Michaliszyn, J. 2016. Verification of multi-agent systems via predicate abstraction against ATLK specifications. In *AAMAS16*, 662–670.

Lomuscio, A.; Qu, H.; and Raimondi, F. 2017. MCMAS: an open-source model checker for the verification of multi-agent systems. In *STTT* 19(1):9–30.

Mogavero, F.; Murano, A.; Perelli, G.; and Vardi, M. 2014. Reasoning about strategies: On the model-checking problem. *ACM Trans. Comp. Log.* 15(4):34:1–34:47.

Pauly, M. 2002. A modal logic for coalitional power in games. *J. Log. Comput.* 12(1):149–166.

Raimondi, F., and Lomuscio, A. 2005. The complexity of symbolic model checking temporal-epistemic logics. In *CS&P*, 421–432.

Shoham, S., and Grumberg, O. 2004. Monotonic abstraction-refinement for CTL. In *TACAS04*, 546–560.

Shoham, S., and Grumberg, O. 2007. A game-based framework for CTL counterexamples and 3-valued abstraction-refinement. *ACM Trans. Comput. Log.* 9(1):1.

Vardi, M. Y. 1995. An automata-theoretic approach to linear temporal logic. In *Banff Higher Order Workshop*, 238–266.