Reasoning about Memoryless Strategies under Partial Observability and Unconditional Fairness Constraints

Simon Busard[a,1], Charles Pecheur[a], Hongyang Qu[b,2], Franco Raimondi[c]

[a] *ICTEAM Institute, Université catholique de Louvain, Louvain-la-Neuve, Belgium*
[b] *Dept. of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, United Kingdom*
[c] *Dept. of Computer Science, Middlesex University, London, United Kingdom*

Abstract

Alternating-time Temporal Logic (ATL) is a logic to reason about strategies that a set of agents can adopt to achieve a specified collective goal. ATL can also be used to specify what agents can do in open systems, where they can interact with their environment in many different ways.

A number of extensions for this logic exist; some of them combine strategies and partial observability, some others include fairness constraints, but to the best of our knowledge no work provides a unified framework for strategies, partial observability and fairness constraints. Integration of these three concepts is of particular importance when reasoning about the capabilities of agents that do not have full knowledge of a system, for instance when the agents can assume that the environment behaves in a fair way.

In this work we present $ATLK_{irF}$, a logic combining strategies under partial observability in a system with fairness constraints on states. We introduce a model-checking algorithm for $ATLK_{irF}$ by extending the algorithm for a full-observability variant of the logic and we investigate its complexity. We validate our proposal with a detailed experimental evaluation, using an implementation that we make publicly available.

Keywords: alternating-time temporal logic, partial observability, fairness constraints, model checking

Email addresses: simon.busard@uclouvain.be (Simon Busard), charles.pecheur@uclouvain.be (Charles Pecheur), h.qu@sheffield.ac.uk (Hongyang Qu), f.raimondi@mdx.ac.uk (Franco Raimondi)

[1] This work is supported by the European Fund for Regional Development and by the Walloon Region.

[2] This author is supported by the EPSRC project EP/J011894/2.

1. Introduction

Alternating-time Temporal Logic (ATL) is a logic for reasoning about the strategies of a subset of agents in a system, and the goals they can achieve using these strategies [1]. ATL can be used to reason about the capabilities of the agents in an open system. ATL considers that the agents have full knowledge of the system and its execution—that is, in every particular state of the system, the agents know the full information about it. This hypothesis about the agents is called *full observability* or *perfect information*. On the other hand, some logics extending ATL allow to reason about agents that do not have the full knowledge of the system; in any particular state of the system, the agents only see a part of it. This situation is called *partial observability* or *imperfect information*.

Starting from [2], partial observability has been investigated by many authors, see for instance [3] and references therein. For example, Jamroga and van der Hoek proposed, among other logics, ATOL, combining partial observability with strategies of agents [4]. Along the same lines, Schobbens studied ATL_{ir} [5], seen as the minimal ATL-based logic for strategies under partial observability [6].

Furthermore, some efforts have been made on bringing fairness to ATL. For instance, the work of Alur et al. [1], or the work of Klüppelholz and Baier [7] introduce the notion of fairness constraints in ATL. But, to the best of our knowledge, no work provides a unified framework for strategies, partial observability and fairness constraints.

This paper proposes $ATLK_{irF}$, a logic for reasoning about strategies under partial observability and epistemic properties of agents in a system with unconditional fairness constraints *on states*; these fairness constraints are *unconditional* because they ask states to appear infinitely often, without condition (compared to strong and weak fairness constraints, that ask for states to appear infinitely often if other states appear). This paper also presents $ATLK_{IrF}$, a variant of $ATLK_{irF}$ with full observability and fairness constraints. Furthermore, this paper provides model-checking algorithms for both logics.

$ATLK_{IrF}$ and $ATLK_{irF}$ have been named by following the notations of ATL_{ir}. Indeed, the subscripts ir of ATL_{ir} means *imperfect information* for the i, compared to I standing for *perfect information* or full observability, and *imperfect recall* for the r, compared to R for *full recall* or memory-full strategies. $ATLK_{IrF}$ has the subscripts for perfect information (or full observability), imperfect recall, and F has been added for *fairness*. Similarly, $ATLK_{irF}$ has the subscripts for imperfect information, imperfect recall, and fairness.

Furthermore, $ATLK_{irF}$ is restricted to memoryless strategies. Other cases could be considered; nevertheless, it has been shown that checking the existence of memory-full strategies with partial observability is undecidable [9]. Also, reasoning about strategies with bounded memory can be reduced to reasoning about memoryless strategies in a derived model in which the memory of the agents is encoded in the states of the model (see [5, 4] for more information). So, restricting to memoryless strategies keeps the model-checking problem decidable, while still allowing more general strategies—i.e. bounded-memory strategies—to be handled.

While some other extensions of ATL—such as $ATOL$ or CSL [4, 3]—are rich enough to speak about all kinds of observability capabilities of the agents, $ATLK_{irF}$ focuses only on one kind of observability. Nevertheless, the general principles presented in this paper also apply to other variants and the logic could be easily modified to reason about other kinds of observability capabilities. Some variations of $ATLK_{irF}$ are discussed in Section 6.

To motivate the need for fairness constraints in ATL under partial observability, consider the simple card game example in [4]. The game is played between a player and a dealer. It uses three cards, A, K and Q; A wins over K, K wins over Q and Q wins over A. First, the dealer gives one card to the player, keeps one and leaves the last one on table, face down. Then the player can keep his card or swap it with the one on the table. The player wins if his card wins over the dealer's card. Figure 1 presents the graph of the game, where the dealer first chooses both cards, then the player can keep his card or swap it with the third one.

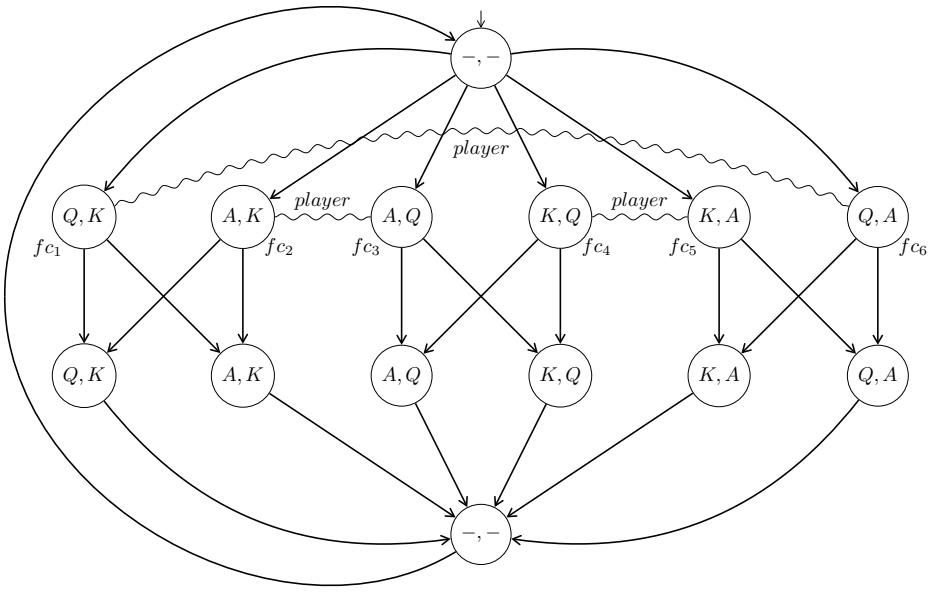


Figure 1: The graph of the card game. Circles are states, with pairs of cards (K, A means player has K, dealer has A). Arrows are transitions (actions of both agents are easily infered). The waved edges link states that are indistinguishable for the player. Fairness constraints fc_i label the intermediate states.

First of all, if the player has full observability, he knows the card of the dealer and thus knows what to do (keeping his card or swapping with the one on table) to win the game. On the other hand, ATL_{ir} considers uniform strategies. Such strategies choose the same action in all states that are indistinguishable for the agents. Under ATL_{ir} semantics, we say that a group of agents can enforce a goal if they have a uniform strategy such that all executions following this strategy satisfy the goal. For example, in the card game, a uniform strategy

for the player will choose to keep K and A and to change when he has Q. It cannot, however, choose a different action when the player has K but the dealer has different cards, because the player does not see the dealer's card until the end of the game. Under ATL_{ir}, the player has no winning uniform strategy: he cannot distinguish between, for example, $\langle A, K\rangle$ and $\langle A, Q\rangle$ (where $\langle a, b\rangle$ means "player has card a, dealer has card b") and thus has to make the same action in both states, with a different result in each case.

Consider now a variation of this game: the game does not terminate after the first round. Instead, cards are redistributed at the end of the game. In this case, too, the player has no eventually winning strategy, even after an infinite number of plays: for instance, he will have to choose between keeping or swapping cards in $\langle A, K\rangle$ and $\langle A, Q\rangle$, so he will not be able to enforce a win because the dealer (that chooses the given cards) can be unfair and always give the losing pair. But if we add one fairness constraint per intermediate state—that is, the states where the dealer has just dealt the cards—the player has a strategy to eventually win the game. In this case, we only consider paths along which all fairness constraints are met infinitely often: this situation corresponds to a fair dealer, giving all the possible pairs infinitely often. The player can thus eventually win because $\langle A, K\rangle$ will eventually happen—even if he cannot distinguish it from $\langle A, Q\rangle$—, so he knows a strategy to eventually win a round: keeping his card. In fact, in this particular system, any strategy of the player wins since all the situations have to eventually happen.

Another example of application of fairness constraints in ATL under partial observability is Multi-Agent Programs [8]. These programs are composed of concurrent agent programs and fairness constraints are used to avoid unfair schedules. Dastani and Jamroga express fairness as formulas of the logic $EATL_p^+$ [8], a variant of ATL^* for multi-agent programs that restricts the objectives of coalitions of programs to Boolean combinations of temporal operators. The present paper, instead, deals only with ATL and therefore fairness constraints cannot be expressed as formulas of the logic. By adding fairness constraints to only consider executions along which the scheduler allows all programs to run infinitely often, $ATLK_{irF}$ allows the user to reason about the strategies of these programs under fair schedules. The situation is similar to the case of LTL versus CTL model checking: in the first case, model checking fairness is reduced to model checking a more complex formula using the same verification algorithms; in the second case fairness is incorporated into bespoke verification algorithms. The advantage of restricting the logic to ATL, that is, to objectives restricted to single temporal operators, lies in the complexity of model checking the logic. Indeed, $EATL_p^+$ has been shown to be Δ_3^P-complete, while the logic this paper proposes is only Δ_2^P-complete[3].

Other works such as the original ATL paper [1] and the work of Klüppelholz and Baier [7] introduce fairness constraints on actions, asking for an infinitely

[3] A Δ_2^P-complete problem needs a polynomial number of calls to a NP oracle ($\Delta_2^P = P^{NP}$); a Δ_3^P-complete problem needs a polynomial number of calls to a $\Sigma_2 P$ oracle ($\Delta_3^P = P^{NP^{NP}}$).

often enabled action to be taken infinitely often. On the other hand, for CTL and derived branching logics such as epistemic logics, fairness constraints are commonly provided on states [10]; this kind of fairness constraints allows the user to model, for example, a lossy channel that would not lose all the messages and would actually send the messages infinitely often. Furthermore, it has been shown that (weak, strong or unconditional) fairness constraints on actions can be reduced to (weak, strong or unconditional, respectively) fairness constraints on states getting a penalty on the number of states of the system (see [11, Chapter 5], for instance). $ATLK_{irF}$ combines CTL operators with ATL ones and it is thus natural to speak about fairness constraints on states in this framework.

The rest of the paper is structured as follows: Section 2 presents the syntax and semantics of our logics. Section 3 and Section 4 present algorithms for model checking these logics. Section 5 studies the complexity of the two model-checking problems. Section 6 discusses some issues and limitations of both logics and Section 7 describes implementation details and presents some experiments. Section 8 discusses the recent related work. Finally, Section 9 summarizes the contribution and draws some future work.

2. Syntax and Semantics

This section presents the syntax and semantics of $ATLK_{IrF}$ and $ATLK_{irF}$, two extensions of ATL with full and partial observability, respectively, under fairness constraints on states.

2.1. Syntax

Both logics share the same syntax, following this grammar:

$$\phi ::= true \mid p \mid \neg\phi \mid \phi \vee \phi \mid E\psi \mid \langle\Gamma\rangle\psi \mid K_i\phi \mid E_\Gamma\phi \mid D_\Gamma\phi \mid C_\Gamma\phi$$
$$\psi ::= X\phi \mid \phi\ U\ \phi \mid \phi\ W\ \phi$$

where p is an atomic proposition of a set AP, Γ is a subset of a set of agents Ag, i is an agent of Ag.

Other operators can be defined in terms of these ones:

$$\begin{aligned} false &\equiv \neg true \\ \phi_1 \wedge \phi_2 &\equiv \neg(\neg\phi_1 \vee \neg\phi_2) \\ \phi_1 \implies \phi_2 &\equiv \neg\phi_1 \vee \phi_2 \\ \phi_1 \iff \phi_2 &\equiv (\phi_1 \implies \phi_2) \wedge (\phi_2 \implies \phi_1) \\ A\psi &\equiv \neg E\neg\psi \\ [\Gamma]\psi &\equiv \neg\langle\Gamma\rangle\neg\psi \\ F\phi &\equiv true\ U\ \phi \\ G\phi &\equiv \phi\ W\ false \end{aligned}$$

In particular,

$$\neg\langle\Gamma\rangle\phi_1 U\phi_2 \equiv [\Gamma](\neg\phi_2)W(\neg\phi_1 \wedge \neg\phi_2)$$
$$\neg\langle\Gamma\rangle\phi_1 W\phi_2 \equiv [\Gamma](\neg\phi_2)U(\neg\phi_1 \wedge \neg\phi_2).$$

The logics combine temporal operators (EX, AF, EG, etc.) from CTL [12, Chapter 3], standard knowledge operators (K_i, E_Γ, D_Γ, C_Γ) [13] and strategic operators ($\langle\Gamma\rangle X$, $[\Gamma]G$, etc.) from ATL [1].

Intuitively, the $E\psi$ CTL path quantifier says that there exists a path satisfying the path formula ψ. A path satisfies $X\phi$ if its second state satisfies ϕ. A path satisfies ϕ_1 U ϕ_2 if ϕ_1 is true along the path until ϕ_2 is true, and ϕ_2 is eventually true. Similarly, a path satisfies ϕ_1 W ϕ_2 if ϕ_1 is true until ϕ_2 is true, but ϕ_2 does not have to be eventually true.

Furthermore, a state s satisfies $K_i\phi$ if the agent i knows that ϕ is true in s. s satisfies $D_\Gamma\phi$ if, by sharing their knowledge of s, the agents of Γ know that ϕ is true. s satisfies $E_\Gamma\phi$ if all the agents of Γ know that ϕ is true, independently. s satisfies $C_\Gamma\phi$ if ϕ is common knowledge among agents of Γ [13].

Finally, a state s satisfies $\langle\Gamma\rangle\psi$ if Γ have a strategy in s such that all paths enforced by the strategy satisfy ψ.

Note that, unlike for CTL, the path operator W (*Weak until*) is needed for expressing the dual operator for $\langle\Gamma\rangle\phi_1 U\phi_2$ [14]. Furthermore, the standard $G\phi$ path operator can be expressed as ϕ W $false$. We can thus limit ourselves to the minimal set of path operators composed of X, U and W.

2.2. Models and notation

$ATLK_{IrF}$ and $ATLK_{irF}$ formulas are interpreted over *models*

$$M = \langle Ag, S, Act, T, I, \{\sim_i\}_{i\in Ag}, V, FC\rangle \tag{1}$$

where

- Ag is a set of n *agents*;
- $S \subseteq S_1 \times ... \times S_n$ is a set of *global states*. Each global state s is composed of n local states $s_1, \ldots, s_n$, one for each agent;
- $Act \subseteq Act_1 \times ... \times Act_n$ is a set of *joint actions*, each of which is composed of n actions, one for each agent. We call a *partially joint action* an element a_Γ of $Act_\Gamma = \prod_{i\in\Gamma} Act_i$, for $\Gamma \subseteq Ag$;
- $T \subseteq S \times Act \times S$ is a transition relation between states in S labelled with joint actions (we write $s \xrightarrow{a} s'$ for $(s, a, s') \in T$). We define the function $img : S \times Act \to 2^S$ as $img(s, a) = \{s' \in S | s \xrightarrow{a} s'\}$, that is, $img(s, a)$ is the set of states reached in one step from s through a;
- $I \subseteq S$ is the a set of *initial states*;

- $\{\sim_i\}_{i\in Ag}$ is a set of equivalence relations between states, one for each agent $i \in Ag$. $\sim_i$ partitions the set of states in terms of knowledge of agent i. $s \sim_i s'$ iff $s_i = s'_i$, that is, two states are indistinguishable for agent i if they share the same local state for i;
- $V : S \to 2^{AP}$ labels states with atomic propositions of AP;
- $FC \subseteq 2^S$ is a set of *fairness constraints*, each of which is a set of states.

Given a set of agents $\Gamma \subseteq Ag$, $\sim_\Gamma = \bigcap_{i\in\Gamma} \sim_i$ is called the *distributed knowledge* relation;$\sim^E_\Gamma = \bigcup_{i\in\Gamma} \sim_i$ is called the *everyone knows* or *group knowledge* relation; $\sim^C_\Gamma$ is the transitive closure of the relation $\sim^E_\Gamma$ and is called the *common knowledge* relation. Intuitively, $s \sim_\Gamma s'$ if no one in Γ can distinguish s and s' and $s \sim^E_\Gamma s'$ if someone in Γ cannot distinguish s and s'.

A *joint action* $a = (a_1, ..., a_n)$ *completes* a partially joint action $a_\Gamma = (a'_i, ..., a'_j)$ composed of actions of agents in $\Gamma \subseteq Ag$—written $a_\Gamma \sqsubseteq a$—if the actions in a for agents in Γ correspond to the actions in a_Γ. Given two disjoint subsets of agents Γ and Γ', and two partially joint actions $a_\Gamma = (a_i, ..., a_j)$ and $a'_{\Gamma'} = (a'_k, ..., a'_l)$ for Γ and Γ' respectively, $a_\Gamma \sqcup a'_{\Gamma'}$ is the (partially) joint action composed of the actions of both joint actions a_Γ and $a'_{\Gamma'}$.

We define the function

$$enabled(s, \Gamma) = \{a_\Gamma \in Act_\Gamma | \exists s' \in S, a \in Act \text{ s.t. } a_\Gamma \sqsubseteq a \wedge s \xrightarrow{a} s'\}, \quad (2)$$

returning the set of partially joint actions for the group of agents Γ that can be played by them in s. We also write $enabled(s, i)$, where i is a single agent, for $enabled(s, \{i\})$.

A model M represents a non-deterministic system where each agent has imperfect information about the current global state. A first restriction is made on T:

$$\forall s \in S, enabled(s, Ag) = \prod_{i\in Ag} enabled(s, i), \quad (3)$$

that is, in every state, the choices of actions for each agent is not constrained by the choices of other agents. Another restriction made on T is

$$\forall s, s' \in S, s \sim_i s' \implies enabled(s, i) = enabled(s', i). \quad (4)$$

This means that the actions an agent can perform in two epistemically equivalent states are the same. These two restrictions ensure that an agent needs only his own knowledge of the current state to choose his action, and he will be able to choose his action, whatever the others choose.

A *path* in a model M is a sequence $\pi = s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} ...$ such that $(s_i, a_{i+1}, s_{i+1}) \in T$ for all $i \leq 0$. We use $\pi(d)$ for s_d. A state s is *reachable* in M if there exist a path π and $d \geq 0$ such that $\pi(0) \in I$ and $\pi(d) = s$. A path π is *fair* according to a set of fairness conditions $FC = \{fc_1, ..., fc_k\}$ if for each fairness condition fc, there exist infinitely many positions $d \geq 0$ such

that $\pi(d) \in fc$. A state s is *fair* if there exists a fair path starting at s. A fair reachable state is thus a state belonging to a fair path starting at an initial state.

A *memory-full strategy* for a group of agents Γ is a function $f_\Gamma : S^+ \to Act_\Gamma$ where, for any path prefix $\pi = s_0...s_m$, $f_\Gamma(\pi) \in enabled(s_m, \Gamma)$; a memory-full strategy maps the sequence of visited states to an enabled action for Γ. A *memoryless strategy* for a group of agents Γ is a memory-full strategy f_Γ such that, for any two path prefixes $\pi = s_0...s_m$ and $\pi' = s'_0...s'_{m'}$, $f_\Gamma(\pi) = f_\Gamma(\pi')$ if $s_m = s'_{m'}$; that is, a memoryless strategy only needs to know the current state to produce the next action, and can be viewed as a function $f_\Gamma : S \to Act_\Gamma$. A *memoryless uniform strategy* for the group Γ—abbreviated as *uniform* strategy for Γ in the sequel—is a memoryless strategy f_Γ where $\forall s, s' \in S, s' \sim_\Gamma s \implies f_\Gamma(s) = f_\Gamma(s')$, that is, the group cannot choose two different actions for two indistinguishable states. Intuitively, a strategy is uniform for Γ if Γ can effectively run it, that is, in every state, Γ only need to share their knowledge of the current global state to know which action to choose. In the sequel, we mainly speak about memoryless uniform strategies.

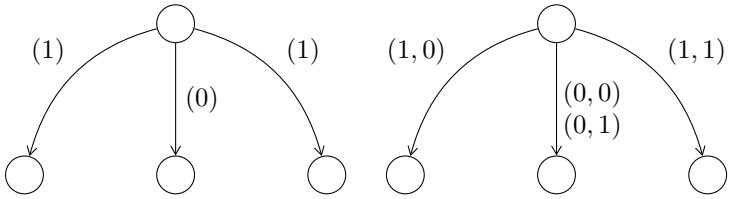
The *strategy outcomes* from a state s for a strategy f_Γ, denoted with $out(s, f_\Gamma)$, is the set of paths a strategy can reach, that is, $out(s, f_\Gamma) = \{\pi = s_0 \xrightarrow{a_1} s_1...| s_0 = s \wedge \forall d \geq 0, s_{d+1} \in img(s_d, a_{d+1}) \wedge f_\Gamma(s_0...s_d) \sqsubseteq a_{d+1}\}$.

Note that the models described above are a generalization of standard ATL and ATL_{ir} models, namely (imperfect information) concurrent game structures ($(i)CGS$ in short) [1, 5]. First, fairness constraints are added to the model to specify which paths are fair. Second, our models describe a non-deterministic transition relation: for a state and a joint action, there can be several successors. On the contrary, $(i)CGS$ are described such that the transition relation is deterministic, that is, for any given state and joint action, there is at most one successor.

This generalization to non-deterministic transition relations does not add expressivity to the models. It is possible to transform any non-deterministic model into a deterministic one by adding a new agent to the model responsible for breaking the non-determinism when needed. More precisely, we can add a new agent ND and update the transition relation such that, for any given state s and partially joint action $a_{Ag \setminus ND}$ (for all agents except ND), ND can choose a different action for each possible successor of s through $a_{Ag \setminus ND}$. In this way, for any given state and joint action, there is only one successor; the model thus becomes deterministic and is equivalent to an $(i)CGS$.

The determinization of models is illustrated in Figure 2. The model—shown in Figure 2(a)—is composed of one agent and four states. The transition relation is non-deterministic because the top state has two successors through the joint action (1). Figure 2(b) shows the determinized model: a new agent is added (showed by new actions on the transitions), and for every previously non-deterministic transition, the action of the new agent breaks the non-determinism. In this determinized model, the transition from the top state through action 1 of the first agent is not non-deterministic anymore. Note that we need to extend action (0) with two new actions as well to keep the constraint on the model

given in Equation 3, that is, the set of enabled joint actions in any state (in this case, the top state) is the cartesian product of the sets of each agent's enabled actions.



(a) Non-deterministic model (b) Determinized model

Figure 2: Determinizing a non-deterministic model with one agent. Vertices are states; edges are transitions, labelled with joint actions.

On the other hand, while this generalization to non-deterministic transition relations adds no expressivity to the model, it can be used to abstract away agents of a system. For example, the scheduler of a multi-agent program can be modeled as part of the non-deterministic environment. Another application is the case of lossy communication channels: using non-deterministic transitions, the lossy channels do not have to be defined as different agents, and the fact that a message sent on a channel is lost or not is non-deterministically handled by the environment.

2.3. Semantics

The semantics of both logics are defined over states of a model M by the relations $M, s \models_{IrF} \phi$ and $M, s \models_{irF} \phi$, for $ATLK_{IrF}$ and $ATLK_{irF}$, respectively. M is omitted when clear from the context.

Both relations share a part of their definition; we write $s \models_{rF} \phi$ when $s \models_{IrF} \phi$ and $s \models_{irF} \phi$ are defined in the same way. The relations $s \models_{IrF} \phi$ and $s \models_{irF} \phi$ are recursively defined over the structure of ϕ and follow the standard interpretation for most of the operators:

$s \models_{rF} \mathit{true}$

$s \models_{rF} p \Leftrightarrow p \in V(s)$

$s \models_{rF} \neg\phi \Leftrightarrow s \not\models_{rF} \phi$

$s \models_{rF} \phi_1 \vee \phi_2 \Leftrightarrow s \models_{rF} \phi_1$ or $s \models_{rF} \phi_2$

$s \models_{rF} E\psi \Leftrightarrow$ there exists a *fair path* π such that $\pi(0) = s$ and $\pi \models_{rF} \psi$

$s \models_{rF} K_i\phi \Leftrightarrow s' \models_{rF} \phi$ for all s' s.t. $s \sim_i s'$ and s' is a *fair reachable* state

$s \models_{rF} D_\Gamma\phi \Leftrightarrow s' \models_{rF} \phi$ for all s' s.t. $s \sim_\Gamma s'$ and s' is a *fair reachable* state

$s \models_{rF} E_\Gamma\phi \Leftrightarrow s' \models_{rF} \phi$ for all s' s.t. $s \sim_\Gamma^E s'$ and s' is a *fair reachable* state

$s \models_{rF} C_\Gamma\phi \Leftrightarrow s' \models_{rF} \phi$ for all s' s.t. $s \sim_\Gamma^C s'$ and s' is a *fair reachable* state.

The relation $\pi \models_{rF} \psi$ over paths π of the model is defined as:

$$\begin{aligned}
\pi \models_{rF} X\phi \Leftrightarrow\ & \pi(1) \models_{rF} \phi \\
\pi \models_{rF} \phi_1 U \phi_2 \Leftrightarrow\ & \text{there exists } d \geq 0 \text{ such that } \pi(d) \models_{rF} \phi_2 \\
& \text{and for all } e < d, \pi(e) \models_{rF} \phi_1 \\
\pi \models_{rF} \phi_1 W \phi_2 \Leftrightarrow\ & \text{there exists } d \geq 0 \text{ such that } \pi(d) \models_{rF} \phi_2 \\
& \text{and for all } e < d, \pi(e) \models_{rF} \phi_1, \\
& \text{or for all } d \geq 0, \pi(d) \models_{rF} \phi_1.
\end{aligned}$$

The meaning of the $\langle\Gamma\rangle$ operator is different in the two semantics:

$$\begin{aligned}
s \models_{IrF} \langle\Gamma\rangle\psi \Leftrightarrow\ & \text{there exists a } \textit{memoryless strategy } f_\Gamma \text{ for } \Gamma, \\
& \text{such that for all } \textit{fair paths } \pi \in \mathit{out}(s, f_\Gamma), \pi \models_{IrF} \psi; \\
s \models_{irF} \langle\Gamma\rangle\psi \Leftrightarrow\ & \text{there exists a } \textit{uniform strategy } f_\Gamma \text{ for } \Gamma, \\
& \text{such that for all } s' \sim_\Gamma s, \\
& \text{for all } \textit{fair paths } \pi \in \mathit{out}(s', f_\Gamma), \pi \models_{irF} \psi.
\end{aligned}$$

The first semantics can be viewed, in the case of strategic operators, as a particular case of the second one where the equivalence relation used for uniform strategies distinguishes all states. The first semantics allows Γ to know everything about the system when choosing and playing their strategies, while the second one restricts them to knowledge they effectively have.

Intuitively, both semantics say that s satisfies $\langle\Gamma\rangle\psi$ if agents in Γ have a collective strategy—that is, a choice of a Γ partially joint action in every state—such that, whatever the action of the other agents is, the objective ψ will be satisfied by all the resulting fair paths. The main difference with the standard ATL semantics is the fact that only fair paths are considered. We can understand it the following way: s satisfies $\langle\Gamma\rangle\psi$ if Γ have a strategy to enforce ψ, assuming all other agents will act fairly, that is, the other agents only follow fair paths. For example, in the case of multi-agent programs, s satisfies $\langle\Gamma\rangle\psi$ if the programs in Γ have a strategy to enforce ψ, assuming that the scheduler—modeled as another agent of the system, or as the environment—is fair, that is, each program will run infinitely often.

Furthermore, $ATLK_{irF}$ compares to $ATLK_{IrF}$ like ATL_{ir} compares to vanilla ATL: $ATLK_{irF}$ restricts Γ to only use strategies that they can effectively apply, according to their (shared) knowledge of the system [5]. This notion of applicable strategies is out of the scope of this paper and is further discussed in [4].

The proposed semantics for $ATLK_{irF}$ considers the group of agents Γ as a single agent (by using the *distributed knowledge* relation), regarding the uniformity of strategies, as well as the equivalence of starting points of these strategies. Intuitively, this semantics captures the cases where all the agents of Γ gather to decide about the possible strategies to win (thus, sharing their knowledge of the current state), and then stay together to actually play the chosen strategy (that is, sharing their knowledge of the visited states along the reached paths). We can also understand this semantics as the agents of Γ being

controlled by a unique controller, able to use their distributed knowledge to gain more information about the current state of the system. The implicit universal quantification over all Γ-indistinguishable starting states captures the fact that the agents of Γ know that they have a uniform strategy to win; together with the restriction to uniform strategies, this amounts to reasoning about strategies that Γ can effectively play, as exposed by Jamroga and van der Hoek in [4]. The fact that Γ is considered as a single agent is further discussed in Section 6.2.

Note that the restriction to memoryless strategies in both semantics is strong. Nevertheless, it is necessary in the case of partial observability: it has been shown that the problem of the existence of a memory-full uniform strategy under partial observability is undecidable [9]. On the other hand, the restriction to memoryless strategies in the case of full observability is not an actual restriction: Section 3.3 will show that there exists a winning memoryless strategy if and only if there exists a winning memory-full one. We express the semantics in terms of memoryless strategies for the sake of homogeneity with the partial observability case.

There are redundancies between the temporal and the strategic operators. The formula $E\psi$ can be expressed in vanilla ATL as $[\emptyset]\psi$: there is a path satisfying ψ if the empty set of agents cannot avoid ψ, that is, all the agents can cooperate to lead to at least one path satisfying ψ [1]. Similarly, $A\psi$ can be expressed as $\langle\emptyset\rangle\psi$, that is, all paths satisfy ψ if, whatever all agents do, ψ is enforced. The same equivalences apply under $ATLK_{IrF}$ and $ATLK_{irF}$. Nevertheless, both kinds of operators are kept in the logics to clearly separate operators about the pure execution of the model (the temporal ones) and operators speaking about strategies (the strategic ones). Furthermore the model-checking algorithms for $ATLK_{irF}$ presented in this paper are clearly more efficient when dealing with temporal operators than with strategic ones.

Because we deal with non-deterministic models with fairness constraints (see Section 2.2), some equivalences that exist in ATL are not kept: $A\psi \not\equiv [Ag]\psi$ and $E\psi \not\equiv \langle Ag\rangle\psi$. These equivalences are broken because of non-determinism, but also because of fairness constraints. Let us illustrate this with two examples.

The first example is given in Figure 3. This model is composed of one agent and the transition relation is non-deterministic because, in the top state, the joint action (1) leads to two successors. In this top state, the property $EX\ p$ is true, because there is a successor where p is true. But the property $\langle Ag\rangle X\ p$ is not true because the (single) agent has no strategy to enforce that the next states satisfy p: playing action 1 may lead to either successor, and p is false in the right one. Dually, the top state does not satisfy $AX\ p$, but it satisfies $[Ag]X\ p$.

The second example is given in Figure 4. In this case, the model contains no fair path since the right state can be visited at most once. In the top state of this model, the property $EX\ p$ is trivially false because there is no fair path (at all) satisfying $X\ p$; on the other hand, the property $\langle Ag\rangle X\ p$ is vacuously true because Ag has a strategy such that all fair paths (there are none) satisfy $X\ p$. Dually, the top state satisfies $AX\ p$, but it does not satisfy $[Ag]X\ p$. Note that this case is due to an ill-defined model containing no fair path, and this

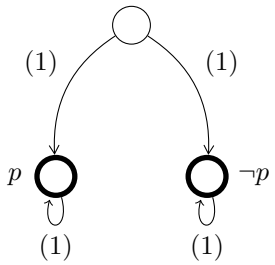


Figure 3: A model with one agent where the equivalences $A\psi \not\equiv [Ag]\psi$ and $E\psi \not\equiv \langle Ag\rangle\psi$ are broken due to non-determinism. Vertices are states, labelled with propositions true in this state; edges are transitions, labelled with joint actions. The bold states belong to the (single) fairness constraint of the model (all paths are fair).

should be avoided in meaningful models. The problem of vacuous strategies will be further discussed in Section 6.3.

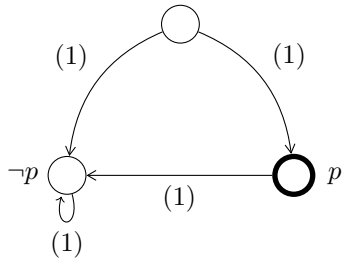


Figure 4: A model with one agent where the equivalences $A\psi \not\equiv [Ag]\psi$ and $E\psi \not\equiv \langle Ag\rangle\psi$ are broken due to presence of fairness constraints. Vertices are states, labelled with propositions true in this state; edges are transitions, labelled with joint actions. The bold state belongs to the (single) fairness constraint of the model.

3. Model Checking Strategies under Full Observability

This section presents a model-checking algorithm for $ATLK_{IrF}$ and a proof of correctness for this algorithm. We first present the algorithm for the variant with full observability because the model-checking algorithm for the variant with partial observability is based on this one.

3.1. A model-checking algorithm using fixpoint computations

The proposed model-checking algorithm for $ATLK_{IrF}$ is defined by the function $eval_{IrF} : ATLK_{IrF} \to 2^S$ returning the set of states of a given (implicit) model M satisfying a given $ATLK_{IrF}$ property. This function is defined in the standard way for Boolean connectors, CTL [12, Chapter 6] and knowledge operators [15].

The evaluation for strategic operators relies on functions $Pre_{[.]} : 2^{Ag} \times 2^S \to 2^S$ and $Reach_{[.]} : 2^{Ag} \times 2^S \times 2^S \to 2^S$ defined as

$$Pre_{[\Gamma]}(Z) = \{s | \forall a_\Gamma \in enabled(s, \Gamma), \exists a \text{ s.t. } a_\Gamma \sqsubseteq a \wedge img(s, a) \cap Z \neq \emptyset\}, \quad (5)$$

$$Reach_{[\Gamma]}(P_1, P_2) = \mu Y.P_2 \cup (P_1 \cap Pre_{[\Gamma]}(Y)). \quad (6)$$

Intuitively, $Pre_{[\Gamma]}(Z)$ returns the set of states in which Γ cannot avoid to reach a state of Z in one step; that is, the set of states in which, whatever Γ do, the other agents have a way to potentially reach Z by choosing the right action. Furthermore, $Reach_{[\Gamma]}(P_1, P_2)$ returns the set of states from which Γ cannot avoid a finite path to a state of P_2 through states of P_1.

The $[\Gamma]$ operators are evaluated as follows:

$$eval_{IrF}([\Gamma]X\phi) = Pre_{[\Gamma]}\big(eval_{IrF}(\phi) \cap Fair_{[\Gamma]}\big) \quad (7)$$

$$eval_{IrF}([\Gamma]\phi_1 U \phi_2) = Reach_{[\Gamma]}\big(\Phi_1, \Phi_{2,F}\big) \quad (8)$$

$$eval_{IrF}([\Gamma]\phi_1 W \phi_2) = \nu Z.\Phi_{2,F} \cup \Big(\Phi_1 \cap \bigcap_{fc \in FC} Pre_{[\Gamma]}\Big(Reach_{[\Gamma]}\big(\Phi_1, \Phi_{2,F} \cup (Z \cap fc)\big)\Big)\Big) \quad (9)$$

where

$$\Phi_1 = eval_{IrF}(\phi_1), \quad (10)$$

$$\Phi_{2,F} = eval_{IrF}(\phi_2) \cap Fair_{[\Gamma]}, \quad (11)$$

$$Fair_{[\Gamma]} = \nu Z. \bigcap_{fc \in FC} Pre_{[\Gamma]}\big(Reach_{[\Gamma]}(S, Z \cap fc)\big). \quad (12)$$

$\mu Z.\tau(Z)$ and $\nu Z.\tau(Z)$ are the least and greatest fixpoints of function $\tau(Z)$. Note that all functions τ involved in the fixpoint computations are monotonic; since S is finite, then

$$\mu Y.\tau(Y) = \bigcup_i \tau^i(\emptyset) = \tau^\infty(\emptyset) \quad (13)$$

$$\nu Y.\tau(Y) = \bigcap_i \tau^i(S) = \tau^\infty(S), \quad (14)$$

where $\tau^{i+1}(Z) = \tau(\tau^i(Z))$ and $\tau^0(Z) = Z$ [12, Chapter 6].

The other $[\Gamma]$ and $\langle\Gamma\rangle$ operators are expressible thanks to the ones above and the $\neg$ operator (see Section 2.1).

Intuitively, $Fair_{[\Gamma]}$ returns the set of states in which Γ cannot avoid a path going through all fairness constraints infinitely often, that is, a fair path. We say that s is Γ-*fair* if $s \in Fair_{[\Gamma]}$.

$eval_{IrF}([\Gamma]X\phi)$ is the set of states from which Γ cannot avoid a Γ-fair successor satisfying ϕ. $eval_{IrF}([\Gamma]\phi_1 U \phi_2)$ is the set of states from which Γ cannot avoid to reach a Γ-fair state satisfying ϕ_2 through states satisfying ϕ_1.

$eval_{IrF}([\Gamma]\phi_1 W \phi_2)$ is the set of states from which Γ cannot avoid a path of states satisfying ϕ_1 and going through all $fc \in FC$ infinitely often, or a path of states satisfying ϕ_1 until a Γ-fair state satisfying ϕ_2 is reached.

Note that the $\langle\Gamma\rangle$ operators can be evaluated using the $[\Gamma]$ and $\neg$ operators, but can also be computed directly using the dual forms of the ones above. More precisely, let

$$\begin{aligned} Pre_{\langle\Gamma\rangle}(Z) &= \overline{Pre_{[\Gamma]}(\overline{Z})} \\ &= \{s | \exists a_\Gamma \in enabled(s,\Gamma) \text{ s.t. } \forall a, a_\Gamma \sqsubseteq a \implies img(s,a) \subseteq Z\} \end{aligned} \tag{15}$$

be the dual of the $Pre_{[\Gamma]}$ function returning the set of states in which Γ can surely reach Z in one step, and

$$NFair_{\langle\Gamma\rangle} = \overline{Fair_{[\Gamma]}} = \mu Z. \bigcup_{fc \in FC} Pre_{\langle\Gamma\rangle}\big(\nu Y.(Z \cup \overline{fc}) \cap Pre_{\langle\Gamma\rangle}(Y)\big) \tag{16}$$

be the set of states in which Γ can avoid fair paths. Using these two new functions, the direct computation of $\langle\Gamma\rangle$ operators can be written as

$$eval_{IrF}(\langle\Gamma\rangle X\phi) = Pre_{\langle\Gamma\rangle}\big(eval_{IrF}(\phi) \cup NFair_{\langle\Gamma\rangle}\big) \tag{17}$$

$$\begin{aligned} &eval_{IrF}(\langle\Gamma\rangle \phi_1 U \phi_2) = \\ &\mu Z.\Phi_{1,2,N} \cap \left(\Phi_2 \cup \bigcup_{fc \in FC} Pre_{\langle\Gamma\rangle}\Big(\nu Y.\Phi_{1,2,N} \cap (Z \cup \overline{fc}) \cap \big(\Phi_2 \cup Pre_{\langle\Gamma\rangle}(Y)\big)\Big)\right) \end{aligned} \tag{18}$$

$$eval_{IrF}(\langle\Gamma\rangle \phi_1 W \phi_2) = \nu Z.\Phi_{1,2,N} \cap \big(\Phi_2 \cup Pre_{\langle\Gamma\rangle}(Z)\big) \tag{19}$$

where

$$\Phi_2 = eval_{IrF}(\phi_2), \tag{20}$$

$$\Phi_{1,2,N} = eval_{IrF}(\phi_1) \cup eval_{IrF}(\phi_2) \cup NFair_{\langle\Gamma\rangle}. \tag{21}$$

3.2. Correctness of the algorithm

This section proves the correctness of the model-checking algorithm presented in the previous section, that is, it proves that for any state s of a given (implicit) model M and any $ATLK_{IrF}$ formula ϕ,

$$s \in eval_{IrF}(\phi) \text{ iff } s \models_{IrF} \phi. \tag{22}$$

The proof is composed of two parts: this section proves that the algorithm effectively computes the set of states in which Γ have a memory-full strategy to win. The next section proves that Γ have a memory-full strategy to win if and only if they have a memoryless one. For this latter part, we use game theory results to show that our objectives do not need memory to be won.

In the sequel, we say that Γ *cannot avoid* x if for any strategy f_Γ, x can happen. For example, Γ *cannot avoid a fair path satisfying* $F\phi$ *in* s if *for any strategy* f_Γ*, there exists a fair path* $\pi \in out(s, f_\Gamma)$ *satisfying* $F\phi$.

Before proving the correctness of the algorithm for the different strategic operators, we need to show some intermediate results.

Lemma 1. *$Reach_{[\Gamma]}(P_1, P_2)$ computes the set of states in which Γ cannot avoid a finite path of states of P_1 to a state of P_2.*

Proof. We can prove this lemma by induction over the computation of the least fixpoint of function $\tau(Y) = P_2 \cup (P_1 \cap Pre_{[\Gamma]}(Y))$. Indeed, it is easy to show that $\tau^i(\emptyset)$ computes the set of states in which Γ cannot avoid a finite path of length at most $i-1$ of states of P_1 to a state of P_2. Thus, in $s \in \tau^\infty(\emptyset)$, Γ cannot avoid to reach P_2 through P_1 in a finite number of steps. □

Lemma 2. *$Fair_{[\Gamma]}$ computes the set of states in which Γ cannot avoid a fair path.*

Proof. We can prove this lemma by induction over the computation of the greatest fixpoint of function $\tau(Z) = \bigcap_{fc \in FC} Pre_{[\Gamma]}(Reach_{[\Gamma]}(S, Z \cap fc))$. Indeed, it is easy to show that $\tau^i(S)$ computes the set of states in which Γ cannot avoid to reach any fairness condition $fc \in FC$ at least i times, in at least one step. Thus, in $s \in \tau^\infty(S)$, Γ cannot avoid to visit any $fc \in FC$ infinitely many times, that is, Γ cannot avoid a fair path. □

We can now prove the correctness of the algorithm for the three strategic operators $[\Gamma]X\phi$, $[\Gamma]\phi_1 U \phi_2$ and $[\Gamma]\phi_1 W \phi_2$.

Lemma 3. *$eval_{IrF}([\Gamma]X\phi)$ returns the set of states s of M in which for any memory-full strategy f_Γ for Γ, there exists a fair path $\pi \in out(s, f_\Gamma)$ such that $\pi \models X\phi$.*

Proof. The proof is trivial by definition of $Pre_{[.]}$ and Lemma 2. □

Lemma 4. *$eval_{IrF}([\Gamma]\phi_1 U \phi_2)$ returns the set of states s of M in which for any memory-full strategy f_Γ for Γ, there exists a fair path $\pi \in out(s, f_\Gamma)$ such that $\pi \models \phi_1 U \phi_2$.*

Proof. $eval_{IrF}([\Gamma]\phi_1 U \phi_2)$ computes the set of states in which Γ cannot avoid a finite path through states satisfying ϕ_1 to a Γ-fair state satisfying ϕ_2, by Lemma 1. □

Lemma 5. *$eval_{IrF}([\Gamma]\phi_1 W \phi_2)$ returns the set of states s of M in which for any memory-full strategy f_Γ for Γ, there exists a fair path $\pi \in out(s, f_\Gamma)$ such that $\pi \models \phi_1 W \phi_2$.*

Proof. We can prove this lemma by induction over the computation of the greatest fixpoint of function

$$\tau(Z) = \begin{array}{l} (eval_{IrF}(\phi_2) \cap Fair_{[\Gamma]}) \cup (eval_{IrF}(\phi_1) \cap \\ \bigcap_{fc \in FC} Pre_{[\Gamma]}(Reach(\begin{array}{l} eval_{IrF}(\phi_1), \\ (eval_{IrF}(\phi_2) \cap Fair_{[\Gamma]}) \cup (Z \cap fc) \end{array}))). \end{array} \quad (23)$$

Indeed, it is easy to show that $\tau^i(S)$ computes the set of states in which Γ cannot avoid to reach a Γ-fair state satisfying ϕ_2 through states satisfying ϕ_1, or in which Γ cannot avoid to visit any fairness condition $fc \in FC$ at least i times through states satisfying ϕ_1, in at least one step. Thus, if $s \in \tau^\infty(S)$, Γ cannot avoid to reach either a Γ-fair state satisfying ϕ_2, or to visit any $fc \in FC$ infinitely many times, through states satisfying ϕ_1, thus Γ cannot avoid a fair path satisfying $\phi_1 U \phi_2$ or a fair path satisfying $G\phi_1$, that is, they cannot avoid a fair path satisfying $\phi_1 W \phi_2$. □

Lemmas 3, 4 and 5 can be now combined into Theorem 6 saying that the algorithm is correct regarding the existence of memory-full strategies winning the objectives.

Theorem 6. *Let s be the state of a model M and ψ be an $ATLK_{IrF}$ path formula, $s \in eval_{IrF}(\langle\Gamma\rangle\psi)$ iff, in s, Γ have a **memory-full** strategy f_Γ such that all fair paths $\pi \in out(s, f_\Gamma)$ satisfy ψ.*

Proof. The proof directly follows from Lemmas 3, 4 and 5 and the equivalence $[\Gamma]\psi \equiv \neg\langle\Gamma\rangle\neg\psi$. □

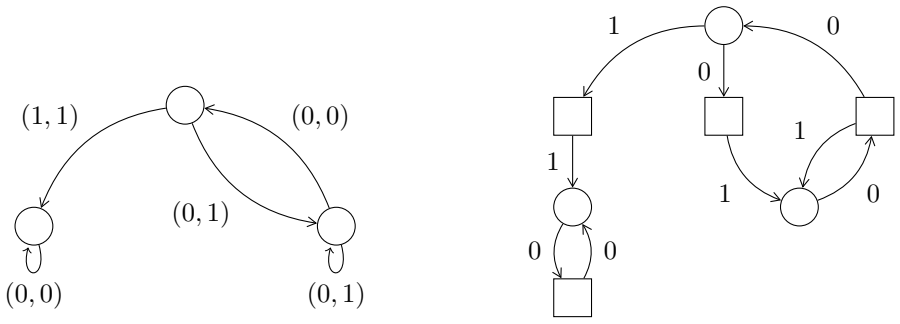
Theorem 6 says that the proposed model-checking algorithm effectively computes, for strategic operators, the set of states in which Γ have a **memory-full** strategy to win. Nothing in the computation or in the proof restricts Γ to always play the same action in any given state. But $ATLK_{IrF}$ is about *memoryless* strategies. The next section shows that there is a memory-full strategy to win iff there is a memoryless one, ending the proof of correctness.

3.3. $ATLK_{IrF}$ and memoryless strategies

This section gives the outline of the proof that the model-checking algorithm presented in Section 3.1 is correct according to the existence of winning memoryless strategies, that is, it effectively computes the set of states in which Γ have a memoryless strategy to win. To show that, we use results from game theory about memory of strategies for different objectives. The full proof is available in Appendix A. This full proof uses results from the work of E. Grädel [16] and W. Thomas [17], and gets inspiration from results presented in [18].

First, given a model and a strategic $ATLK_{IrF}$ property $\langle\Gamma\rangle\psi$, we can build a two-player, turn-based game such that a state of the model satisfies the property if and only if the corresponding state of the game is winning for the player corresponding to Γ. Intuitively, Γ is mapped to the first player 0 and $\overline{\Gamma}$ is mapped to the second player 1. Furthermore, for each action a_Γ of Γ enabled

in a state s, a new state s_{a_Γ} for 1 is created, representing the fact that Γ chose the corresponding action; in these states for 1, $\overline{\Gamma}$ can then choose the next state. This mapping is illustrated in Figure 5.



(a) A model

(b) The corresponding game graph

Figure 5: A model and its corresponding game graph. Γ is the first agent, $\overline{\Gamma}$ is the second one. Vertices are states; circles in the game graph are player 0 states, squares are player 1 states; edges are transitions, labelled with actions (for clarity, in the game graph).

On the other hand, the $ATLK_{IrF}$ property is translated into an objective for the corresponding two-player turn-based game, and we can show that, to a strategy for 0 winning the objective on the game, there corresponds a strategy for Γ in the original model. Indeed, whenever the player 0 chooses a successor in the game, there exists a corresponding action in the original model, and every path enforced in the game by the winning strategy can be mapped to a path in the original model, enforced by the corresponding strategy.

Furthermore, using standard results in game theory about memoryless strategies of particular objectives, we can show that our objectives do not need memory to be won in such games. For this, we propose custom algorithms to find all the winning states of the game; by the way the algorithms compute these states, we can show that there exists a winning memoryless strategy in these states, and so the player does not need memory to win the game. Thus, since there exists a correspondance between winning strategies in the game and winning strategies in the original model, agents do not need memory either in the cases of $ATLK_{IrF}$ objectives on our models.

Finally, the correctness of the model-checking algorithm for $ATLK_{IrF}$ follows from Theorem 7.

Theorem 7. *$eval_{IrF}(\langle\Gamma\rangle\psi)$ returns the set of states of M satisfying $\langle\Gamma\rangle\psi$ under $ATLK_{IrF}$, that is*

$$\forall s \in S, s \models_{IrF} \langle\Gamma\rangle\psi \text{ if and only if } s \in eval_{IrF}(\langle\Gamma\rangle\psi). \tag{24}$$

Proof. From Theorem 6, we know that $eval_{IrF}(\langle\Gamma\rangle\psi)$ computes the set of states of the model in which Γ have a memory-full strategy to win ψ. Furthermore, we know that Γ have a memory-full strategy to win if and only if they have

a memoryless one. Thus, we can conclude that in the states computed by $eval_{IrF}(\langle\Gamma\rangle\psi)$, Γ have a memoryless strategy to win, and that the algorithm is correct for strategic operators. □

Note that for other operators, the algorithm follows the standard algorithms and the proof is the same (see [12, Chapter 6], for example).

4. Model Checking Strategies under Partial Observability

This section presents three model-checking algorithms for $ATLK_{irF}$. The first one splits the model into uniform strategies and uses the model-checking algorithm for $ATLK_{IrF}$ to get states satisfying the property. The second one improves the approach by alternating between splitting the model and filtering out losing actions. Finally, Section 4.3 discusses the ideas behind the two algorithms, generalizes them and proposes a third algorithm combining the advantages of the two first approaches.

4.1. Splitting then filtering

A first algorithm for model checking $ATLK_{irF}$ strategic properties is presented in Algorithm 1. It relies on the model-checking algorithm for $ATLK_{IrF}$ and uses two sub-algorithms: *Split* and a modified version of $eval_{IrF}$. *Split* divides the transition relation of the model into uniform strategies *strat* and $eval_{IrF}$ performs the model-checking algorithm for $ATLK_{IrF}$ on the fraction of the model restricted to *strat*. Note that for the other operators (CTL operators, Boolean connectors, etc.) model checking $ATLK_{irF}$ is identical to model checking $ATLK_{IrF}$, and $eval_{irF}$ works as $eval_{IrF}$.

Algorithm 1: $eval_{irF}(\langle\Gamma\rangle\psi)$

Data: M a given (implicit) model, Γ a subset of agents of M, ψ an $ATLK_{irF}$ path formula.
Result: The set of states of M satisfying $\langle\Gamma\rangle\psi$.

$sat = \{\}$
for $strat \in Split(S \times Act_\Gamma)$ **do**
  $winning = eval_{IrF}(\langle\Gamma\rangle\psi, strat)$
  $sat = sat \cup \{s \in winning | \forall s' \sim_\Gamma s, s' \in winning\}$
return *sat*

Intuitively, Algorithm 1 gets all the possible uniform strategies for Γ thanks to *Split*. Then, for each of these strategies, it calls the model-checking algorithm for $ATLK_{IrF}$, restricted to the strategy. Finally, it only keeps the states for which all equivalent states satisfy the property in the given strategy.

The $eval_{IrF}$ algorithm is modified by adding an argument *strat* to $Pre_{\langle\Gamma\rangle}$, where, given $strat \subseteq S \times Act_\Gamma$ and $Z \subseteq S$,

$$Pre_{\langle\Gamma\rangle}(Z, strat) = \left\{ s \;\middle|\; \begin{array}{l} \exists a_\Gamma \in enabled(s,\Gamma) \text{ s.t. } \langle s, a_\Gamma\rangle \in strat \\ \wedge\ \forall a, a_\Gamma \sqsubseteq a \implies img(s,a) \subseteq Z \end{array} \right\}. \quad (25)$$

$Pre_{\langle\Gamma\rangle}(Z, strat)$ is $Pre_{\langle\Gamma\rangle}(Z)$ restricted to states and actions allowed by $strat$. Furthermore, $eval_{IrF}$ recursively calls $eval_{irF}$ on sub-formulas, instead of $eval_{IrF}$. More precisely,

$$eval_{IrF}(\langle\Gamma\rangle X\phi, strat) = Pre_{\langle\Gamma\rangle}\big(eval_{irF}(\phi) \cup NFair_{\langle\Gamma\rangle}(strat), strat\big) \quad (26)$$

$$\begin{array}{l} eval_{IrF}(\langle\Gamma\rangle\phi_1 U\phi_2, strat) = \\ \mu Z.\Phi_{1,2,N} \cap \Big(\Phi_2 \cup \bigcup\limits_{fc\in FC} Pre_{\langle\Gamma\rangle}\Big(\nu Y.\ \begin{array}{l}\Phi_{1,2,N} \cap (Z \cup \overline{fc}) \cap \\ \big(\Phi_2 \cup Pre_{\langle\Gamma\rangle}(Y, strat)\big)\end{array}\ , strat\Big)\Big) \end{array} \quad (27)$$

$$eval_{IrF}(\langle\Gamma\rangle\phi_1 W\phi_2, strat) = \nu Z.\Phi_{1,2,N} \cap \big(\Phi_2 \cup Pre_{\langle\Gamma\rangle}(Z, strat)\big) \quad (28)$$

where

$$\Phi_2 = eval_{irF}(\phi_2), \quad (29)$$

$$\Phi_{1,2,N} = eval_{irF}(\phi_1) \cup eval_{irF}(\phi_2) \cup NFair_{\langle\Gamma\rangle}(strat), \quad (30)$$

$$NFair_{\langle\Gamma\rangle}(strat) = \mu Z.\bigcup_{fc\in FC} Pre_{\langle\Gamma\rangle}(\nu Y.(Z \cup \overline{fc}) \cap Pre_{\langle\Gamma\rangle}(Y, strat), strat). \quad (31)$$

The *Split* algorithm is given in Algorithm 2. $Split(S \times Act_\Gamma)$ returns the set of uniform strategies for Γ, where each strategy f_Γ is represented by the action $f_\Gamma(s)$ for group Γ allowed in each state s. This action needs to be the same for each state in the same equivalence class. We call a *move* an element of $S \times Act_\Gamma$; such an element $\langle s, a_\Gamma\rangle$ corresponds to Γ choosing the action a_Γ in state s.

We say that two moves $\langle s, a_\Gamma\rangle$ and $\langle s', a'_\Gamma\rangle$ are conflicting, and we write $\langle s, a_\Gamma\rangle \perp \langle s', a'_\Gamma\rangle$, if both states are equivalent ($s \sim_\Gamma s'$) and the actions are different ($a_\Gamma \neq a'_\Gamma$). Given a set of moves $SA \subseteq S \times Act_\Gamma$,

$$Conflicts(SA) = \{m \in SA | \exists m' \in SA \text{ s.t. } m \perp m'\} \quad (32)$$

returns the set of conflicting moves of SA.

Intuitively, *Split* gets all the conflicting equivalence classes. If there are no such conflicts, SA represents a uniform strategy. Otherwise, *Split* picks one conflicting equivalence class *equivalent*, gets all the possible actions *actions* in this class and, for each of these actions a_Γ, creates a new strategy by computing the cross product of the strategy playing a_Γ in the equivalence class, and the strategies *substrats* for other equivalence classes recursively computed by *Split*. In other words, *Split* iteratively splits conflicting equivalence classes and computes the cross product of all the splittings to build all possible uniform strategies of the model.

Before proving the correctness of the model-checking algorithm, let us prove the correctness of the *Split* algorithm.

Algorithm 2: $Split(SA)$

Data: Γ a given (implicit) subset of agents, $SA \subseteq S \times Act_\Gamma$.
Result: The set of all the largest subsets *strat* of SA such that no conflict appears in *strat*.

$conflicting = Conflicts(SA)$
if $conflicting = \emptyset$ **then return** $\{SA\}$
else
  $\langle s, a_\Gamma \rangle =$ **pick** one element in $conflicting$
  $equivalent = \{\langle s', a'_\Gamma \rangle \in SA | s' \sim_\Gamma s\}$
  $actions = \{a'_\Gamma \in Act_\Gamma | \exists \langle s, a'_\Gamma \rangle \in equivalent\}$
  $substrats = Split(SA \backslash equivalent)$
  $strats = \{\}$
  for $a_\Gamma \in actions$ **do**
    $equivStrat = \{\langle s', a_\Gamma \rangle \in equivalent\}$
    $strats = strats \cup \{equivStrat \cup substrat | substrat \in substrats\}$
  return *strats*

Lemma 8. *$Split(SA)$ computes the set of all the largest subsets strat of $SA \subseteq S \times Act_\Gamma$ such that no conflicts appear in strat.*

Proof. We can prove the correctness of *Split* by induction over the number of conflicting equivalence classes of SA. If SA does not contain any conflicting equivalence classes, SA is its own single largest subset in which no conflicts appear. Otherwise, let us assume that $Split(SA \backslash equivalent)$, with *equivalent* a conflicting equivalence class of SA, returns the set of all the largest non-conflicting subsets of $SA \backslash equivalent$. Then, *Split* returns the cartesian product between all the largest non-conflicting subsets of *equivalent* and all the largest non-conflicting subsets of $SA \backslash equivalent$. Because these cannot be conflicting as they belong to different equivalence classes, we can conclude that *Split* returns the set of the largest non-conflicting subsets of SA. □

The correctness of Algorithm 1 is then given by the following theorem.

Theorem 9. *Given an (implicit) model M, a subset Γ of agents of M and an $ATLK_{irF}$ formula $\langle\Gamma\rangle\psi$,*

$$\forall s \in S, s \models_{irF} \langle\Gamma\rangle\psi \text{ if and only if } s \in eval_{irF}(\langle\Gamma\rangle\psi). \tag{33}$$

Proof. First, $Split(S \times Act_\Gamma)$ returns all the possible uniform strategies of the system, where a uniform strategy is represented by the only action allowed in each equivalence class of states (states equivalent in terms of the knowledge of Γ), this action being the same for every state of the class. Indeed, the set of the largest non-conflicting subsets of $S \times Act_\Gamma$ is the set of possible uniform strategies.

Furthermore, $winning = eval_{IrF}(\langle\Gamma\rangle\psi, strat)$ returns the set of states for which the strategy *strat* is winning. Indeed, it uses $ATLK_{IrF}$ model-checking

algorithm, restricted to actions in *strat*. It thus returns the set of states for which there is a (not necessarily uniform) winning strategy in *strat*. As *strat* is, by construction, a uniform strategy for Γ, *winning* is the set of states for which there exists a uniform winning strategy—in fact, it is *strat* itself.

Finally, the set $\{s \in winning | \forall s' \sim_\Gamma s, s' \in winning\}$ is the set of states s for which *strat* is a winning strategy for all $s' \sim_\Gamma s$. *sat* accumulates all the states s for which there is a winning strategy for all states indistinguishable from s. As this is exactly the semantics of the property, that is, *sat* is exactly the set of states of the system satisfying the property, the proof is done. □

Note that for other operators, the model-checking algorithm for $ATLK_{irF}$ follows the standard algorithms, like the one for $ATLK_{IrF}$, and the proof is the same.

4.2. Alternating between splitting and filtering

One problem of the algorithm presented in the previous section is the number of uniform strategies it considers. This number can be very huge (see Section 5) while the number of winning strategies can be small. This comes from the fact that $eval_{irF}$ blindly splits equivalence classes in which no winning strategy exists, increasing drastically the number uniform strategies to consider.

A solution presented in this section is to filter out states and actions that surely cannot be winning. For this, we use the $ATLK_{IrF}$ model-checking algorithm to filter out states and actions that are not part of a (not necessarily uniform) strategy; because the filtered states (and actions) cannot be part of a winning strategy, they cannot be part of a winning uniform strategy. More precisely, the proposed improvement consists in alternating between filtering out states and actions that cannot be part of a strategy, and splitting remaining conflicting equivalence classes. The filtering is performed using a modified version of the $ATLK_{IrF}$ model-checking algorithm presented in Section 3. Note that we can filter states that do not satisfy the property under $ATLK_{IrF}$ semantics since they cannot satisfy the property under $ATLK_{irF}$ semantics neither.

The improved algorithm is presented in Algorithm 3. Using this algorithm, we can compute the set of states satisfying $\langle\Gamma\rangle\psi$ as $eval_{irF}^{improved}(\langle\Gamma\rangle\psi, S \times Act_\Gamma)$. This algorithm uses the function $eval_{IrF}^{moves}$. This function is a modification of the $eval_{IrF}$ function restricted to a set of moves and where moves are returned instead of states. More precisely, let

$$Moves_\Gamma(Z, SA) = \{\langle s, a_\Gamma\rangle \in SA | s \in Z \wedge a_\Gamma \in enabled(s, \Gamma)\} \tag{34}$$

returning the moves composed of states of Z and the enabled actions that are allowed by SA. Let

$$Pre_{\langle\Gamma\rangle}^{moves}(Z, SA) = \left\{ \langle s, a_\Gamma\rangle \in SA \;\middle|\; \begin{array}{l} a_\Gamma \in enabled(s, \Gamma) \\ \wedge \forall a, a_\Gamma \sqsubseteq a \implies img(s, a) \subseteq Z \end{array} \right\} \tag{35}$$

returning the states from which Γ can enforce to reach Z in one step, associated with the actions that allow them to do so, but restricted to actions allowed by

SA. That is, the set of moves of SA allowing Γ to surely reach Z in one step. Let

$$NFair^{moves}_{\langle\Gamma\rangle}(SA) = \mu Z.\ \bigcup_{fc\in FC} Pre^{moves}_{\langle\Gamma\rangle}\left(\nu Y.\ \begin{array}{l} Moves_\Gamma(Z \cup \overline{fc}, SA)) \\ \cap\ Pre^{moves}_{\langle\Gamma\rangle}(Y, SA) \end{array}, SA\right) \tag{36}$$

be the set of states in which Γ can avoid fair paths thanks to moves of SA, with the actions involved in the possible strategies.

Given a set of moves SA, $eval^{moves}_{IrF}$ is defined as

$$\begin{array}{l} eval^{moves}_{IrF}(\langle\Gamma\rangle X\phi, SA) = \\ Pre^{moves}_{\langle\Gamma\rangle}\Big(Moves_\Gamma(eval_{irF}(\phi), S \times Act_\Gamma) \cup NFair^{moves}_{\langle\Gamma\rangle}(SA), SA\Big) \end{array} \tag{37}$$

$$\begin{array}{l} eval^{moves}_{IrF}(\langle\Gamma\rangle \phi_1 U \phi_2, SA) = \\ \mu Z.\Phi_{1,2,N} \cap \left(\Phi_2 \cup \bigcup_{fc\in FC} Pre^{moves}_{\langle\Gamma\rangle}\Big(\nu Y.\ \begin{array}{l} \Phi_{1,2,N} \cap Moves_\Gamma(Z \cup \overline{fc}, SA) \\ \cap\ \big(\Phi_2 \cup Pre^{moves}_{\langle\Gamma\rangle}(Y, SA)\big) \end{array}, SA\Big)\right) \end{array} \tag{38}$$

$$eval^{moves}_{IrF}(\langle\Gamma\rangle \phi_1 W \phi_2, SA) = \nu Z.\Phi_{1,2,N} \cap \big(\Phi_2 \cup Pre^{moves}_{\langle\Gamma\rangle}(Z, SA)\big) \tag{39}$$

where

$$\begin{array}{l} \Phi_{1,2,N} = \\ Moves_\Gamma(eval_{irF}(\phi_1), SA) \cup Moves_\Gamma(eval_{irF}(\phi_2), SA) \cup NFair^{moves}_{\langle\Gamma\rangle}(SA), \end{array} \tag{40}$$

$$\Phi_2 = Moves_\Gamma(eval_{irF}(\phi_2), SA). \tag{41}$$

Intuitively, $eval^{moves}_{IrF}(\langle\Gamma\rangle\psi, SA)$ returns the states satisfying $\langle\Gamma\rangle\psi$ associated to the actions of SA that allow them to do so, restricted to the ones of SA.

Note that in the expression for $\langle\Gamma\rangle X\phi$, the set of moves extracted from $eval_{irF}(\phi)$ is not restricted to SA. This is necessary because these moves do not belong to a strategy to win $X\phi$. As the idea of the algorithm is to filter out moves until the result is a uniform strategy, the set of moves extracted from $eval_{irF}(\phi)$ do not belong to such strategy; on the other hand, the corresponding states are necessary for the computation of $eval^{moves}_{IrF}(\langle\Gamma\rangle X\phi)$.

The intuition behind Algorithm 3 is to start by computing the set of states and the associated actions from which a (not necessarily uniform) strategy exists (line 1), then get all conflicting equivalence classes (line 2) and, if there are conflicts, choose one conflicting equivalence class of states and possible actions (lines 6 to 8) and for each possible action a_Γ, recursively call the algorithm with the strategies following a_Γ (lines 11 and 12)—that is, split the class into uniform strategies for this class and recursively call the algorithm on each strategy.

Algorithm 3: $eval^{improved}_{irF}(\langle\Gamma\rangle\psi, SA)$

Data: M a given (implicit) model, Γ a subset of agents of M, ψ an $ATLK_{irF}$ path formula, $SA \subseteq S \times Act_\Gamma$.
Result: The set of states of M satisfying $\langle\Gamma\rangle\psi$ in SA.

```
1  SA' = eval^moves_IrF(⟨Γ⟩ψ, SA)
2  conflicting = Conflicts(SA')
   if conflicting = ∅ then
4      return {s ∈ S | ∀s' ∼_Γ s, ∃a_Γ ∈ Act_Γ s.t. ⟨s', a_Γ⟩ ∈ SA'}
   else
6      ⟨s, a_Γ⟩ = pick one element in conflicting
7      equivalent = {⟨s', a'_Γ⟩ ∈ SA' | s ∼_Γ s'}
8      actions = {a_Γ ∈ Act_Γ | ∃⟨s, a_Γ⟩ ∈ equivalent}
       sat = {}
       for a_Γ ∈ actions do
11         strat = {⟨s', a_Γ⟩ ∈ equivalent} ∪ (SA' \ equivalent)
12         sat = sat ∪ eval^improved_irF(⟨Γ⟩ψ, strat)
       return sat
```

Algorithm 3 returns the set of states satisfying the property in SA, that is, states s in which Γ has a uniform strategy using only actions in SA that is winning for all states equivalent to s. So, to get the states of the model satisfying the property, we have to take all the states for which there is a winning uniform strategy in $S \times Act_\Gamma$.

The correctness of Algorithm 3 is given by the following theorem.

Theorem 10. *$eval^{improved}_{irF}(\langle\Gamma\rangle\psi, S \times Act_\Gamma)$ computes the set of states of M satisfying $\langle\Gamma\rangle\psi$, i.e.*

$$\forall s \in S, s \in eval^{improved}_{irF}(\langle\Gamma\rangle\psi, S \times Act_\Gamma) \text{ iff } s \models_{irF} \langle\Gamma\rangle\psi. \tag{42}$$

Proof. First, $eval^{moves}_{IrF}(\langle\Gamma\rangle, SA)$ computes the moves of SA composed of the states for which there exists a winning strategy in SA, and actions that compose these winning strategies. In other words, $eval^{moves}_{IrF}(\langle\Gamma\rangle, SA)$ filters out from SA the states for which there is no winning strategy in SA, and the actions that are not part of a winning strategy. Indeed, the fixpoint computations of $eval^{moves}_{IrF}$ correspond to the fixpoint computations of $eval_{IrF}$ where the $Pre^{moves}_{\langle\Gamma\rangle}$ operator only consider moves available in SA. Thus, the states returned by these fixpoint computations are the states for which there exists a winning strategy in SA instead of in the whole system.

Furthermore, we can show, with a proof similar to the ones of Lemmas 1 to 5, that a move of SA is returned by $eval^{moves}_{IrF}$ iff this move belongs to a winning strategy of SA. Indeed, $Pre_{\langle\Gamma\rangle}$ computes, through fixpoint computations, the states for which there exists a winning strategy; $Pre^{moves}_{\langle\Gamma\rangle}$ computes, in addition, the actions that allow Γ to win.

Second, $eval_{irF}^{improved}(\langle\Gamma\rangle, SA)$ computes the states for which there exists a winning uniform strategy in SA. Indeed, if SA' contains no conflict, this means that SA' represents a (unique) winning uniform strategy. The states satisfying $\langle\Gamma\rangle\psi$ in SA' are the ones for which all equivalent states are in SA', and this is what is computed and returned at Line 4. On the other hand, if SA' contains some conflicts, SA' represents several uniform strategies. In this case, the algorithm splits one conflicting equivalence class and tries all the possible splittings for this class, thus missing no potential uniform strategy.

Finally, $eval_{irF}^{improved}(\langle\Gamma\rangle, S \times Act_\Gamma)$ is the set of states satisfying $\langle\Gamma\rangle\psi$ since the whole model is considered. □

4.3. Mixing splitting and filtering

The two algorithms presented above can be viewed as particular cases of algorithms mixing filtering and splitting. Indeed, Algorithm 1 first splits all conflicting equivalence classes of the model—by using the *Split* sub-algorithm—and then filters out states that are losing for each strategy to get all states satisfying the property. On the other hand, Algorithm 3 alternates between filtering out states that are losing for the current subset of strategies, and splitting the current subset of strategies into uniform ones, one conflicting equivalence class at a time, until the strategy is uniform and the remaining states are winning.

We can imagine other variants of mixing filtering and splitting. In particular, $eval_{irF}^{FSF}(\langle\Gamma\rangle\psi)$—given as Algorithm 4—first filters out states that have no winning strategy, as well as losing moves, then splits all equivalence classes and ends by, for each resulting uniform strategy, filtering out the states for which the strategy is losing[4]. Compared to the first solution, this algorithm gains much from the first filtering, giving fewer equivalence classes to split, while gaining from the second solution by avoiding to filter between each equivalence class splitting. This algorithm really combines the advantages of both approaches in practice, as shown by experimental results presented in Section 7.

Algorithm 4: $eval_{irF}^{FSF}(\langle\Gamma\rangle\psi)$

Data: M a given (implicit) model, Γ a subset of agents of M, ψ an $ATLK_{irF}$ path formula.
Result: The set of states of M satisfying $\langle\Gamma\rangle\psi$.

$sat = \{\}$
$SA = eval_{IrF}^{moves}(\langle\Gamma\rangle\psi, S \times Act_\Gamma)$
for $strat \in Split(SA)$ **do**
  $winning = eval_{IrF}(\langle\Gamma\rangle\psi, strat)$
  $sat = sat \cup \{s \in winning | \forall s' \sim_\Gamma s, s' \in winning\}$
return sat

[4]The FSF superscript stands for "filter, split, filter".

The correctness of Algorithm 4 is given by the following theorem. The proof is similar to the ones for the two previous algorithms.

Theorem 11. *$eval^{FSF}_{irF}(\langle\Gamma\rangle\psi)$ computes the set of states of M satisfying $\langle\Gamma\rangle\psi$, i.e.*

$$\forall s \in S, s \in eval^{FSF}_{irF}(\langle\Gamma\rangle\psi) \text{ iff } s \models_{irF} \langle\Gamma\rangle\psi. \tag{43}$$

Proof. First, the proof of Theorem 10 showed that $eval^{moves}_{IrF}$ do not lose winning moves. So, the moves of $(S \times Act_\Gamma) \backslash eval^{moves}_{IrF}(\langle\Gamma\rangle\psi, S \times Act_\Gamma)$ cannot be part of a winning uniform strategy and there exists a winning uniform strategy from a given state s in $S \times Act_\Gamma$ if and only if there exists a winning uniform strategy from s in $eval^{moves}_{IrF}(\langle\Gamma\rangle\psi, S \times Act_\Gamma)$. Furthermore, the proof of Theorem 9 showed that enumerating all uniform strategies in $S \times Act_\Gamma$ and accumulating the states for which these strategies are winning correctly computes the set of states satisfying $\langle\Gamma\rangle\psi$. Thus, since we showed that filtering out losing moves does not ignore winning uniform strategies, Algorithm 4 correctly computes the set of states satisfying $\langle\Gamma\rangle\psi$. □

5. Complexity Considerations

This section discusses complexities of $ATLK_{IrF}$ and $ATLK_{irF}$ model-checking problems. While the first one is in **P**, the second is one is Δ_2^P-complete. A problem is in $\Delta_2^P = \mathbf{P^{NP}}$ if it can be solved in deterministic polynomial time with subcalls to an **NP**-oracle.

5.1. $ATLK_{IrF}$ model-checking complexity

This section shows that the model-checking problem for $ATLK_{IrF}$ is in **P**, that is, there exists an algorithm to solve the problem that is polynomial in terms of the size of the model and of the formula. Let us show that the algorithm proposed in Section 3 is effectively polynomial in terms of the size of M and ϕ. This section only focuses on the evaluation of strategic operators; for the other operators, standard literature already discusses the problem.

First, the computation of the $Pre_{[\Gamma]}(Z)$ operator is polynomial: given any set of states Z, it is necessary to check each state and transition of the system at most once to get all the states belonging to $Pre_{[\Gamma]}(Z)$. Second, $Reach_{[\Gamma]}(P, Q)$ can be computed in polynomial time: $\tau(Z) = Q \cup (P \cap Pre_{[\Gamma]}(Z))$ is monotonic, thus the least fixpoint of $\tau(Z)$ is reached in at most $|S|$ steps. Each step computes the $Pre_{[\Gamma]}$ operator, thus the $Pre_{[\Gamma]}$ operator is evaluated at most $|S|$ times, hence the polynomial time complexity of $Reach_{[\Gamma]}$. Third, $Fair_{[\Gamma]}$ can also be computed in polynomial time: it is composed of a greatest fixpoint of a τ function evaluating $Reach_{[\Gamma]}$ for each fairness constraint $fc \in FC$. The τ function is monotonic, thus only a polynomial number of calls to $Reach_{[\Gamma]}$ are needed, hence a polynomial time complexity.

Furthermore, all the strategic operators are compositions of the above three functions, thus they have a polynomial time complexity. Finally, one evaluation is needed for each subformula of the checked formula ϕ to check it. Thus, the model-checking problem for $ATLK_{IrF}$ is in **P**.

5.2. $ATLK_{irF}$ model-checking complexity

This section shows that the model-checking problem for $ATLK_{irF}$ is Δ_2^P-complete.

Model checking ATL with perfect recall and partial observability is an undecidable problem [9], while model checking ATL_{ir} is a Δ_2^P-complete problem [6]. Let us show that $ATLK_{irF}$ subsumes ATL_{ir} in the case of two agents; its model-checking problem is therefore Δ_2^P-hard, since the model-checking problem for ATL_{ir} is already Δ_2^P-complete for only two agents [6].

Intuitively, any $iCGS$—a model over which ATL_{ir} properties are interpreted—can be transformed into a model with only one fairness constraint composed of the full state-space; the corresponding model shares the same labeled graph as the $iCGS$ and fairness constraints say that any path of the model is a fair path.

It is obvious that there exists a uniform strategy for an agent i in an $iCGS$ such that all enforced paths satisfy ψ if and only if there exists a uniform strategy for the same agent in the corresponding model such that all enforced fair paths satisfy ψ, since every path of the model is fair.

Note that there is a difference in the definition of strategies for a group of agents in ATL_{ir} and $ATLK_{irF}$: collective strategies in ATL_{ir} are tuples of individual strategies for the agents of the coalition, while strategies in $ATLK_{irF}$ consider Γ as a single agent. This difference has no impact on the complexity of $ATLK_{irF}$ model checking since the proof of Δ_2^P-hardness of ATL_{ir} is based on a two-agents game [6], and ATL_{ir} and $ATLK_{irF}$ coincide when Γ is composed of only one agent.

Let us now show that there is a Δ_2^P algorithm for model checking $ATLK_{irF}$ property, thus that the problem is Δ_2^P-complete. The proposed algorithm uses a non-deterministic variant of Algorithm 1 where the uniform strategies are non-deterministically chosen among the possible ones. More precisely, the proposed algorithm uses Algorithm 5 to compute the states of the model satisfying a given strategic operator.

Algorithm 5: $eval_{irF}^{ND}(\langle\Gamma\rangle\psi)$

Data: M a given (implicit) model, Γ a subset of agents of M, ψ an $ATLK_{irF}$ path formula.
Result: The set of states of M satisfying $\langle\Gamma\rangle\psi$.

$states = S$
$strat = \{\}$
while $states \neq \emptyset$ **do**
  $s =$ **pick** one element in $states$
  $states = states \backslash \{s' | s' \sim_\Gamma s\}$
6 $a_\Gamma =$ **choose** one action in $enabled(s, \Gamma)$
  $strat = strat \cup \{\langle s', a_\Gamma\rangle | s' \sim_\Gamma s\}$
$win_{IrF} = eval_{IrF}|_{strat}(\langle\Gamma\rangle\psi)$
return $\{s \in S | \forall s' \sim_\Gamma s, s' \in win_{IrF}\}$

Algorithm 5 is effectively non-deterministic: the **choose** operation at Line 6 is a non-deterministic choice among the enabled actions of Γ in s. Furthermore, this algorithm is polynomial: the **while** loop is repeated at most $|S|$ times since the size of *states* is decreased by at least one at each step. Furthermore, as shown in the previous section, the $eval_{IrF}|_{strat}(\langle\Gamma\rangle\psi)$ algorithm runs in polynomial time, thus Algorithm 5 is in **NP**.

Finally, the model-checking problem for $ATLK_{irF}$ is in Δ_2^P since we need to call an **NP** algorithm for each subformula of the formula to check, thus **P** calls to an **NP** procedure. We can then conclude that model checking $ATLK_{irF}$ is a Δ_2^P-complete problem.

6. Discussion

This section discusses two choices that have been made when designing $ATLK_{IrF}$ and $ATLK_{irF}$ semantics: the type of fairness constraints and the knowledge relations used for uniform strategies. Furthermore, it discusses the issue of vacuous strategies and presents possible solutions.

6.1. Fairness constraints

This section presents the fairness constraints chosen in the semantics of $ATLK_{IrF}$ and discusses alternatives. It also discusses the link between fairness and strategies.

Let us remind the semantics of $ATLK_{IrF}$ strategic operators. Given a state s of a model $M = \langle Ag, S, Act, T, I, \{\sim_i\}_{i\in Ag}, V, FC\rangle$ and a path property ψ,

$$s \models_{IrF} \langle\Gamma\rangle\psi \Leftrightarrow \begin{array}{l}\text{there exists a } \textit{memoryless strategy } f_\Gamma \text{ for } \Gamma,\\ \text{such that for all } \textit{fair paths } \pi \in out(s, f_\Gamma), \pi \models_{IrF} \psi;\end{array} \quad (44)$$

where a path is *fair* iff it meets all fairness constraints $fc \in FC$ infinitely often. This kind of fairness definition is called *unconditional* fairness and is standard in the framework of $Fair\ CTL$ and is used in the family of SMV model checkers (e.g. NuSMV) [12]. There exist other kinds of fairness such as *strong* and *weak* fairness, expressed on actions or on states. For example, a strong fairness constraint over actions says that a path π is fair iff *an action that is enabled along states of* π *infinitely many times must be taken infinitely many times*; a weak fairness constraint says that a path is fair iff *an action that is eventually enabled permanently must be taken infinitely many times.* While fairness constraints can be expressed on actions or on states, it has been shown that fairness constraints on actions can be reduced to fairness constraints on states, thus it is sufficient to limit ourselves to fairness constraints on states [11, Chapter 3].

The three kinds of fairness constraints can be useful to reason about the strategies of agents. For example, unconditional fairness constraints can be used to reason about the strategies of multi-agent programs under a fair scheduler; in this case, unconditional fairness constraints ensure that only executions along which the scheduler allows all programs to run infinitely often are considered, by tracking the last run program and constraining each program to run infinitely

often [8]. In the same vein, weak and strong fairness constraints can be used to reason about strategies of agents under particular circumstances. For example, suppose a multi-agent program where one particular program controls a mutex in the system; if an agent needs to lock this mutex to achieve its task, it cannot win its objective without the cooperation of the controlling program. On the other hand, by adding strong fairness constraints to ensure that if the agent asks infinitely often for a mutex, the controlling program will grant it, the agent has a strategy to win its objective.

The advantage of unconditional fairness, compared to strong and weak fairness, is that it does not differentiate between memoryless and memory-full strategies, that is, we can only focus on memoryless ones without losing any expressive power. On the other hand, it has been shown that in the case of strong and weak fairness, memory is necessary to win the objectives: two variants on ATL have been proposed, both needing memory [1]. Nevertheless, the amount of memory needed to win the objectives under weak and strong fairness is finite. Furthermore, we know that strategies that are allowed to use a bounded amount of memory for choosing the next action can be reduced to memoryless strategies in a derived model where the memory is encoded in the states [4, 5], allowing the reasoning to be restricted to memoryless strategies.

Another concern about fairness constraints is about the set of agents that must enforce fair paths. In $ATLK_{IrF}$ semantics, Γ win if they have a strategy such that **all fair paths satisfy the objective**; in this case, $\overline{\Gamma}$ have to produce fair paths that violate the objective to prevent Γ to win. Let us call this semantics the *weak-strategy semantics*. Another choice is possible: Γ win if they have a strategy such that **all enforced paths are fair and satisfy the objective**; in this case, Γ have to enforce only fair paths to win, and $\overline{\Gamma}$ can prevent them to win by avoiding fair paths (regardless of the objective). Let us call this new semantics the *strong-strategy semantics*.

While weak strategy objectives do not need memory to be won (see Section 3.3 and Appendix A), strong-strategy ones need memory to be won. For example, let us consider the model presented in Figure 6 and the property $\langle Ag \rangle Fp$. The model contains only one agent that can play two different actions in the top state. For a path to be fair, both the left and right states must be visited infinitely often. Under the weak-strategy semantics, the property is true in the top state: by playing 0 in the top state, the agent will enforce no fair path (the only enforced path never meets the right state), and the property is vacuously true. Note that if the agent can use memory, the result is the same, the agent still has a strategy to win. On the other hand, under the strong-strategy semantics, the agent has a memory-full strategy to win the objective in the top state—for example, play 0 and 1 alternatively—but no memoryless strategy to win. He has to stick to the same action in the top state, and will never meet the other state. Thus, in the case of the strong-strategy semantics, memory makes a difference.

When designing $ATLK_{IrF}$ semantics, we kept the weak-strategy semantics because it corresponds to the usual situation in which fairness is an assumption about the environment. Furthermore, this semantics is useful, for example, to reason about multi-agent programs [8]; in this case, the weak-strategy semantics

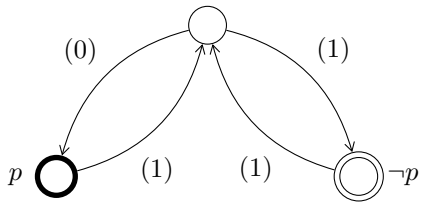


Figure 6: A model with one agent where the agent needs memory to win under the strong-strategy semantics. Vertices are states, labelled with propositions true in this state; edges are transitions, labelled with joint actions; the bold state belongs to one fairness constraint, the double-lined one to another.

allows the user to reason about the strategies of the programs while assuming a fair scheduler.

6.2. Knowledge relations

This section presents the different knowledge relations used in the semantics of $ATLK_{irF}$, discusses alternatives and motivates the choice made for $ATLK_{irF}$.

Let us remind the semantics of $ATLK_{irF}$ strategic operators. Given a state s of a model $M = \langle Ag, S, Act, T, I, \{\sim_i\}_{i\in Ag}, V, FC\rangle$ and a path property ψ,

$$s \models_{irF} \langle\Gamma\rangle\psi \Leftrightarrow \begin{array}{l}\text{there exists a } \textit{uniform strategy } f_\Gamma \text{ for } \Gamma,\\ \text{such that for all } s' \sim_\Gamma s,\\ \text{for all } \textit{fair paths } \pi \in \textit{out}(s', f_\Gamma), \pi \models_{irF} \psi.\end{array} \tag{45}$$

There are two different knowledge relations used in this semantics. The first one is about the states in which the strategy must start: the property is satisfied in s if there is a strategy that wins in all states s' equivalent to s. In the sequel, this equivalence relation is called the *starting-point equivalence relation*. In the case of $ATLK_{irF}$, this relation is the distributed knowledge relation $\sim_\Gamma$.

The second knowledge relation used in $ATLK_{irF}$ semantics is the one used in the definition of uniform strategies. $ATLK_{irF}$ only considers uniform strategies, that is, strategies that play the same action in two different equivalent states. In the sequel, this equivalence relation is called the *strategy-points equivalence relation*. In the case of $ATLK_{irF}$, this relation is also the distributed knowledge relation $\sim_\Gamma$.

Several equivalence relations could be used in place of the starting point and the strategy-points equivalence relations. For example, ATL_{ir} diverges from $ATLK_{irF}$ by defining a collective strategy for Γ as a tuple of strategies for agents of Γ; this can be seen as using the individual knowledge relation of each agent to define the strategy-points equivalence relation. The intuition behind $ATLK_{irF}$ semantics (already explained in Section 2.3) is that the group of agents is considered as controlled by a single controller: they gather to choose the strategy (they can use their distributed knowledge to know where they start) but also to play the strategy (and they can also use their distributed knowledge to play the strategy). The intuition behind ATL_{ir} is slightly different: the

agents are completely separated and cannot share their knowledge for choosing the strategy (using only their own knowledge to know where they start), as well as to play the chosen strategy (they can only rely on their individual knowledge of the encountered states to know which action to play).

Other combinations for both equivalence relations can be used. For example, Jamroga and Bulling use the *everyone knows* relation for the starting-point equivalence relation and the individual knowledge relation for the strategy-points equivalence relation [19]. Technically, any standard knowledge relation (group knowledge, common knowledge, distributed knowledge) can be used for either equivalence. We chose to use the distributed knowledge for both equivalences because the intuition is natural and uniform.

Finally, the algorithms presented in this paper can be easily adapted to use other knowledge relations for both equivalences. Most of them (such as group, common and distributed knowledge) only need a change of knowledge relation used in the algorithms. On the other hand, the case of individual knowledge, as used in ATL_{ir}, needs to split conflicting equivalence classes one agent at a time, leading to an additional blow-up of the number of uniform strategies to consider. Nevertheless, this is a minor modification of the presented algorithms.

6.3. Vacuous strategies

This section discusses the problem of vacuous strategies and presents possible solutions. Let us illustrate this problem with the example of the card game presented in the Introduction. We already discussed the case of the player, who can eventually win the game, relying on the fair behavior of the dealer: because he knows the dealer will eventually deal a winning hand for him, he can always keep his card and finally win the game. On the other hand, the dealer can also always win the game; he has a strategy such that the player never wins. This strategy is to avoid any fair path; if he enforces no fair path, all enforced fair paths vacuously satisfy any objective and, in particular, the objective consisting in never letting the player win.

More generally, the problem of vacuous strategies is that if a group of agents can avoid fair paths, that is, they have a strategy such that no fair paths are enforced, then they can win over any objective, even an unsatisfiable one such as $F\ false$. Since their strategy allows to avoid fair paths, all the fair paths (there are none) trivially satisfy any temporal property. Note that this problem is already present in CTL, where a state s satisfies any $A\psi$ property if s is not fair, that is, no fair path starts in s. In the case of CTL, the usual approach is to inform the user whenever the model accepts no fair paths. This solution can be used in the present case, and we can inform the user whenever Γ can avoid fair paths. This can be performed by checking whether $s \in NFair_{\langle\Gamma\rangle}$.

Nevertheless, there are other ways to address the problem by changing the semantics. For example, we can change the semantics (and the algorithms accordingly) such that only strategies enforcing at least one fair path are considered. This is different from the strong-strategy semantics discussed above in the sense that, in the present case, the agents still do not need to enforce only fair paths. The only requirement is that their strategy enforce at least one fair path. This

can be achieved, for example, by checking whether there exists a fair path in a generated uniform strategy before checking whether it is winning or not. In the case of the card game, this would solve the concern above: if we only consider strategies that contain at least one fair path, the dealer has no strategy to prevent the player to win, since he will need to deal each pair of cards infinitely often.

7. Experiments

The algorithms for model checking $ATLK_{IrF}$ and $ATLK_{irF}$ have been implemented in a BDD-based framework. This section discusses the implementation of these algorithms with BDDs, focusing on parts that have been implemented slightly differently to fit the BDD framework.

Second, this section presents some early experiments showing performances of the algorithms. The implementation is a prototype showing the applicability of the presented approaches, that would not compete with industrial tools performing the same kind of tasks. These experiments are thus not meant to show the absolute performances of the implementation but the relative gain of the proposed algorithms.

7.1. Implementing model-checking algorithms with BDDs

The algorithms presented in this paper have been implemented with BDDs, thanks to PyNuSMV, a Python framework based on NuSMV [20]. While most of the parts can be directly implemented through BDD operations and simple loops, such as unions, intersections and fixpoint computations, some operations and algorithms need more work to be implemented. This section focuses on the $Pre_{\langle\Gamma\rangle}$ operator and its variations, and on the *Split* algorithm and the computation of conflicting classes.

In the BDD-based model-checking framework, sets of states and sets of actions are represented with BDDs. Furthermore, the whole transition relation of the system, as well as the different knowledge relations for each agent, are also represented with BDDs (see for example [12, Chapter 6]). In this framework, any operation over sets of states or actions is performed as an operation on BDDs.

Given a BDD representing the transition relation of the system, we can easily define the operator $Pre'(Z)$ returning the BDD of the moves leading to at least one state of the BDD of the set Z, that is,

$$Pre'(Z) = \{\langle s, a\rangle | \exists s' \in Z \text{ s.t. } s' \in img(s, a)\}. \tag{46}$$

This operator is present in BDD-based model-checking tools like NuSMV because it is the basis operator for BDD-based CTL model checking. The implementation of this operator relies on the BDD representation of the transition relation and the existential quantification on BDDs.

Given the Pre' operator and a BDD representing a set of states Z, we can implement the $Pre_{\langle\Gamma\rangle}$ with BDD operations:

$$Pre_{\langle\Gamma\rangle}(Z) = \exists a.(\overline{\exists a_{\overline{\Gamma}}.Pre'(\overline{Z})} \cap Pre'(Z)), \tag{47}$$

where $\exists$ is the existential quantification over BDDs, $\cap$ is the conjunct of BDDs, and $\overline{\ }$ is the negation of BDDs. Intuitively, $\exists a_{\overline{\Gamma}}.Pre'(\overline{Z})$ is the set of pairs of states and actions of Γ such that there exists a completing action leading to $\overline{Z}$. Thus, $\overline{\exists a_{\overline{\Gamma}}.Pre'(\overline{Z})}$ is the set of pairs of states and actions of Γ such that all completing actions surely lead to Z. Then, $\exists a.(\overline{\exists a_{\overline{\Gamma}}.Pre'(\overline{Z})} \cap Pre'(Z))$ is the set of states such that there is an action for Γ surely leading to Z. Note that $\overline{\exists a_{\overline{\Gamma}}.Pre'(\overline{Z})}$ must be restricted to $Pre'(Z)$ to be sure to only keep actions that are actually enabled.

The *Split* algorithm is, on the other hand, not easily implemented as described in Algorithm 2. More precisely, the computation of

$$Conflicts(SA) \tag{48}$$

needs some modifications to be implemented with BDDs. In fact, to directly compute this set of conflicting state/action pairs, we need to quantify over a relation between two state/action pairs; that is, we need to quantify over a BDD ranging over two copies of state and action variables. Such a BDD can become really huge and is not necessarily available in a BDD-based model-checking framework like NuSMV.

One way to implement the *Split* algorithm is to pick one state/action pair in SA, get all equivalent states—by using a BDD ranging over two copies of state variables, instead of two copies of state and action variables, as discussed above—then all state/action pairs of SA restricted to these states, and finally check that there is only one available action for these states in SA. If this is not the case, this particular equivalence class is conflicting and needs to be split. Otherwise, we can ignore this equivalence class and pick another state/action pair from SA.

This implementation is less efficient than Algorithm 2 because it has to enumerate all equivalence classes (through picking state/action pairs) before concluding that SA is non-conflicting. On the other hand, it does not need to build a huge BDD representing the relation between two state/action pairs.

7.2. Experimental evaluation

To test the implementation of the algorithms presented in this paper, two sets of models have been designed. The first set of models is based on a modification of the problem of the card game presented in the Introduction; the second set is based on the ancient Chinese story of Tian Ji. The two sets are designed such that it is easy to build larger models, to show how the approach scales up. This section first describes the two sets of models, discusses their modelling, and performs experimentations with the implementation of the algorithms with PyNuSMV.

First, the card game is modified as follows. The game is still composed of a dealer and player; there are N cards $c_1, ..., c_N$ such that c_i wins over c_j iff $i > j$, but c_1 wins over c_N. The game is divided into four phases. In the first phase, the dealer takes one card for him; in the second phase, he gives one card to the

player. In the third phase, the player has two choices: either to keep his card, or to discard it and get one card from the stack of remaining cards; he can change his card as many times as he wants, but can never get back a discarded card. The fourth phase is entered when the player chooses to keep his card or when the stack is empty. The winner is known during the last phase; the winner is the one with the winning card. The game is replayed infinitely.

The formal model of this problem is composed of two agents, the player and the dealer. The states are composed of the card of the player, the card of the dealer, the stack of remaining cards and a counter to keep track of the current phase. The dealer can choose the card to give at different phases and the player can choose to keep or change his card. The dealer knows both cards while the player only knows his card. Finally, the fairness conditions are set such that a path is fair if the two first cards (the dealer's card and the player's first card) follow a fair distribution.

Second, the Tian Ji's model is inspired from the ancient Chinese story of Tian Ji. The system is composed of two agents: a king and his general Tian Ji. They play a horse racing game. They both own N horses $h_1, ..., h_N$ with different speeds; if the king races with horse h_i and Tian Ji with h_j, the winner is the one with the highest index; if $i = j$, the winner is chosen non-deterministically. N races are played, and the final winner is the one with the most races won. The game is replayed infinitely.

The formal model of this problem is composed of two agents, Tian Ji and the king. The states are composed of Tian Ji's remaining horses, the king's remaining horses, and the score of each of them. At each step of the game, both can choose between their respective remaining horses the one to play at the next race. Tian Ji only knows which horses he chooses but not the king's. Finally, the fairness conditions are set such that a path is fair if the king chooses his horses in all possible orders, infinitely many times each; these fairness constraints have been implemented by asking the king to choose the full order of his horses *a priori*, before starting the first race, and asking that a fair path goes through all the possible choices infinitely often.

The first property checked over the card game is $\Phi_1 = \langle player \rangle F\ win$. The property is satisfied under both $ATLK_{IrF}$ and $ATLK_{irF}$ semantics; in the first case, the player knows the card of the dealer—because he has full observability—and thus knows if he has to change or keep his card. In the second case, the player does not know the card of the dealer, but knowing that the dealer is fair is enough for him to win: by keeping his card in every game, he knows that the dealer will eventually give him a winning card. In fact, all uniform memoryless strategies are winning because, for each of them, the player knows that the fair dealer will eventually give the winning pair of cards.

A similar property has been checked over the problem of Tian Ji: $\Phi_2 = \langle TianJi \rangle F\ win$. This property is satisfied by the model under both semantics: under $ATLK_{IrF}$ Tian Ji knows *a priori* the order of the horses of the king and can thus choose the right horses to race. Under $ATLK_{irF}$ semantics, he can choose a unique order and play it repeatedly; he knows that the king will eventually run an order that is losing for him. Again, all memoryless uniform

strategies are winning because, for each of them, the corresponding losing horses order for the king will eventually be played.

Tables 1 and 2 show the running time of the implemented algorithms for Φ_1 and Φ_2, respectively[5]. First, we can see that the time needed to check the properties under $ATLK_{IrF}$ slowly increases. This is expected since the algorithm is polynomial in terms of the size of the model. On the other hand, the time needed by the basic algorithm $eval_{irF}$ grows exponentially, showing the bad complexity of the algorithm: the number of uniform strategies to check grows exponentially and, for each of them, a polynomial check has to be performed. Note also that the improved algorithm $eval_{irF}^{improved}$ performs **worse** than the basic algorithm: because almost all states and actions of the system can be part of a winning strategy, filtering the losing ones does not remove many of them. Furthermore, as the algorithm alternates between filtering and splitting conflicting equivalence classes, the number of filtering steps is higher than for the basic algorithm. Finally, the third proposed approach, $eval_{irF}^{FSF}$ performs as well as the basic algorithm: because there are only a few losing moves, filtering them out does not give some advantage against the basic approach; nevertheless, since there is only one application of filtering, the useless extra effort has no significant impact on performances.

Table 1: Running time for model checking Φ_1 over the card game models.

Cards	**States**	$ATLK_{IrF}$	$eval_{irF}$	$eval_{irF}^{improved}$	$eval_{irF}^{FSF}$
3	28	0m0.351s	0m0.610s	0m0.998s	0m0.664s
5	326	0m0.764s	0m10.991s	0m21.410s	0m12.575s
7	2696	0m2.517s	7m2.823s	13m33.928s	7m15.172s
9	18442	0m31.405s	> 30m	> 30m	> 30m
10	46091	8m52.097s			

Table 2: Running time for model checking Φ_2 over Tian Ji's models.

Horses	**States**	$ATLK_{IrF}$	$eval_{irF}$	$eval_{irF}^{improved}$	$eval_{irF}^{FSF}$
3	61	0m0.214s	0m1.401s	0m2.582s	0m1.494s
4	409	0m1.685s	> 30m	> 30m	> 30m
5	3271	3m33.885s			

The improved algorithm $eval_{irF}^{improved}$ performs worse than the basic one on the first two checked properties. To show that this improved algorithm can perform better than the basic one on some formulas and models, one more property has been checked on both models. These formulas do not highlight relevant properties of the application but, by their nature, they allow the

[5]All the presented results have been obtained by running the PyNuSMV implementation on a Intel® Xeon® CPU E5-2630 at 2.30GHz, with Ubuntu 13.04. The time limit has been set to 2000 seconds and the memory limit to 30GB.

improved algorithm to perform better.

The second property checked over the card game is

$$\Phi_3 = AG\left(\begin{array}{l} (phase = 3 \wedge player.card = c_1) \\ \implies \langle player \rangle X\ player.card = c_1 \end{array} \right). \tag{49}$$

This property says that when the player can choose to keep his card or change it—that is, the system is in the third phase—and he has the c_1 card, then he can keep it. Again, the property is satisfied under both semantics. It includes a strategic subformula that is satisfied by only a few states of the model. Indeed, the player needs to already own c_1 to be able to enforce next states in which he owns the card. Thus, by filtering out losing states before splitting the remaining moves, $eval_{irF}^{improved}$ should be able to drastically reduce the number of uniform strategies to consider, compared to the basic algorithm.

A similar property has been checked over the problem of Tian Ji:

$$\Phi_4 = AG\left(\begin{array}{l} TJ.horses = \text{“all horses”} \\ \implies \langle TianJi \rangle X\ TJ.horses = \text{“all horses but the lowest one”} \end{array} \right). \tag{50}$$

This property says that when Tian Ji can still choose any horse, then he can choose the lowest one. Again, this property is satisfied under both semantics. Similarly to Φ_3, this property includes a strategic subformula that is satisfied by only a few states of the model. $eval_{irF}^{improved}$ should also drastically reduce the number of strategies to consider in this case.

Tables 3 and 4 show the running time of the implemented algorithms for Φ_3 and Φ_4, respectively. First of all, note that the properties are composed of one CTL operator AG and one strategic operator $\langle \Gamma \rangle X$. While the former is easily checked in polynomial time, the latter needs more time to be checked under $ATLK_{irF}$ semantics. Again, we can see the slow growing of the time needed to check the properties under $ATLK_{IrF}$. Furthermore, checking them under $ATLK_{irF}$ semantics with the basic algorithm still needs an exponentially growing time: the algorithm still needs to enumerate all uniform strategies. On the other hand, $eval_{irF}^{improved}$ performs really better. As expected, pre-filtering out the losing actions gets only a few states left; the strategic subformulas in both properties are satisfied by only a few states, leading to a few uniform strategies to check. This greatly decreases the time needed to check the property. Finally, $eval_{irF}^{FSF}$ even outperforms $eval_{irF}^{improved}$. Indeed, while $eval_{irF}^{improved}$ filters out losing moves several times, $eval_{irF}^{FSF}$ only performs filtering once. In fact, this first filtering is really important because it removes most of the losing moves; on the other hand, the subsequent filters of $eval_{irF}^{improved}$ do not remove a lot of moves, performing extra useless work.

These results show that $eval_{irF}$ and $eval_{irF}^{improved}$ can be interesting, depending on the checked property and the model itself. Note that the $eval_{irF}^{improved}$ performs worse in the case of Φ_1 and Φ_2 because it repeatedly tries to filter out losing actions while these actions are not numerous. These results also show that the third approach $eval_{irF}^{FSF}$ outperforms the two first approaches on the tested

Table 3: Running time for model checking Φ_3 over the card game models.

Cards	**States**	$ATLK_{IrF}$	$eval_{irF}$	$eval_{irF}^{improved}$	$eval_{irF}^{FSF}$
3	28	0m0.165s	0m0.296s	0m0.184s	0m0.187s
5	326	0m0.759s	0m3.398s	0m0.437s	0m0.632s
7	2696	0m1.583s	2m9.233s	0m2.108s	0m1.820s
9	18442	0m24.610s	> 30m	0m47.673s	0m26.190s
10	46091	1m56.126s		9m10.488s	2m6.998s

Table 4: Running time for model checking Φ_4 over Tian Ji's models.

Horses	**States**	$ATLK_{IrF}$	$eval_{irF}$	$eval_{irF}^{improved}$	$eval_{irF}^{FSF}$
3	61	0m0.186s	0m0.546s	0m0.183s	0m0.202s
4	409	0m0.961s	29m38.394s	0m0.890s	0m1.097s
5	3271	1m9.873s	> 30m	0m27.088s	0m51.896s

models. These good results come from the fact that, for the tested models and properties, the subsequent filters of $eval_{irF}^{improved}$ are not useful, and only the first filtering is important.

8. Related Work

While model checking ATL under full observability has received a great deal of attention, to the best of our knowledge results about implementations for model checking strategies under partial observability are more limited. In this section we present some recent work in this direction.

First, the work presented in this paper has been extended by the authors [21]. The idea of this extension is to improve the model-checking process by reducing the number of strategies to check. Indeed, if we are interested in whether there exists a winning uniform strategy in the initial states for a given set of agents, it is not necessary to check all uniform strategies, but only the ones that are reachable from the initial states; these reachable strategies are called partial strategies because they do not specify a move in every state, but only in the states that matter. Furthermore, this extension proposes three practical optimizations to further improve the process.

Second, Pilecki, Bednarczyk and Jamroga proposed a similar approach based on the notion of partial strategies [22]. To generate these partial strategies, a forward traversal of the model is performed, starting from the initial state and splitting equivalence classes that are encountered in the traversal. The work of Pilecki et al. interleaves this forward traversal with CTL computations to stop the process as soon as a winning partial strategy has been found. This approach is closely related to the idea of alternating between splitting and filtering presented in Section 4.2. Indeed, the idea of alternating between splitting and filtering is to stop the process of generating a uniform strategy as soon as we know that there cannot exist a winning uniform strategy. While the approach of Pilecki

et al. still has strong limitations—it is limited to one initial state and one top strategic operator—and is based on an explicit model-checking approach, it can be extended to handle the full logic and to fit the symbolic model-checking framework of this paper.

Third, Huang and van der Meyden proposed a fully symbolic approach to check the existence of winning uniform strategies [23]. More precisely, their approach is more general since the logic they present subsumes several strategic logics, and $ATLK_{irF}$ can be defined in their framework. The present paper proposes to represent each uniform strategy as a unique BDD in order to keep the number of variables of these BDDs small. On the other hand, the work of Huang and van der Meyden proposes to encode the strategies of the agents in the model itself, resulting to a much larger set of BDD variables. The model-checking problem is then reduced to fixpoint computations on the extended model. Finally, they showed that their approach is more efficient than the one presented in this paper for the case of ATL_{ir}. Nevertheless, the efficiency of their approach is similar to the efficiency of the two ones presented earlier in this section.

Finally, model checking uniform strategies has also been discussed by Calta et al. [24]. Their approach is also based on splitting possible actions into uniform strategies, but their algorithm is tightly linked to the nature of the objectives, namely CTL-like objectives. So, their approach cannot be easily adapted to deal with fairness constraints.

9. Conclusion

A number of studies in the past have investigated the problem of model checking strategies under partial observability and, separately, some work has provided algorithms for including fairness constraints on *actions* in the case of full observability. To the best of our knowledge, the issue of fairness constraints and partial observability have never been addressed together.

In this paper we presented $ATLK_{irF}$, a logic combining partial observability and fairness constraints on *states* (which is the standard approach for temporal and epistemic logics), and we have provided a model-checking algorithm.

Furthermore, the structure of our algorithm is compatible with symbolic model checking using BDDs, and we worked on its implementation in the model checker MCMAS [25], where fairness constraints are only supported for temporal and epistemic operators, and with PyNuSMV [20], a Python framework for prototyping BDD-based model-checking algorithms. This paper presented some early experiments illustrating the high complexity of $ATLK_{irF}$ model checking.

Further work. Several discussions about the particular choices made in this paper have been presented. In particular, $ATLK_{irF}$ uses unconditional fairness constraints to define fair paths in the system. While this kind of fairness is useful, for example, to reason about the strategies of multi-agent programs under a fair scheduler, other kinds of fairness such as strong fairness would also lead to interesting properties. Nevertheless, this paper explained that using strong fairness constraints instead of unconditional fairness constraints leads to other

difficulties related to the memory of the agents. It is not obvious for the authors how to adapt the current algorithms to reason about strategies under strong fairness constraints, and this is left as future work.

Second, the strategies discussed in this paper are completely deterministic. One extension of this work could consider randomized strategies, allowing the agents to randomly choose played actions with different probabilities. Another extension could consider probabilistic notions of fairness, asking for unfair paths to occur with probability zero. Nevertheless, these approaches use model checking techniques that are dramatically different from the ones developed in this paper. This is so left as future work as well.

Finally, the approach considers only memoryless strategies. Some more general kinds of strategies such as bounded-memory strategies can be reduced to memoryless strategies; on the other hand, the case of memory-full uniform strategies has been shown to be undecidable [9]. Nevertheless, it is not obvious for the authors whether unbounded-memory strategies—that is, strategies that can use a finite but unbounded amount of memory—can be reduced to one case or the other, and this question is also left as future work.

Acknowledgments

Thanks are due to Pierre-Yves Schobbens and Jean-François Raskin for inspiring discussions about ATL and its variants, and games and memory of strategies. Thanks are also due to the reviewers for their useful comments.

References

[1] R. Alur, T. A. Henzinger, O. Kupferman, Alternating-time temporal logic, J. ACM 49 (5) (2002) 672–713. doi:10.1145/585265.585270.

[2] W. van der Hoek, M. Wooldridge, Cooperation, knowledge, and time: Alternating-time temporal epistemic logic and its applications, Studia Logica 75 (2003) 125–157. doi:10.1023/A:1026185103185.

[3] W. Jamroga, T. Ågotnes, Constructive knowledge: what agents can achieve under imperfect information, Journal of Applied Non-Classical Logics 17 (4) (2007) 423–475. doi:10.3166/jancl.17.423-475.

[4] W. Jamroga, W. van der Hoek, Agents that know how to play, Fundamenta Informaticae Volume 63 (2) (2004) 185–219.

[5] P.-Y. Schobbens, Alternating-time logic with imperfect recall, Electronic Notes in Theoretical Computer Science 85 (2) (2004) 82 – 93. doi:10.1016/S1571-0661(05)82604-0.

[6] W. Jamroga, J. Dix, Model checking abilities under incomplete information is indeed Δ_2^P-complete, in: EUMAS'06, 2006.

[7] S. Klüppelholz, C. Baier, Alternating-time stream logic for multi-agent systems, in: Coordination Models and Languages, LNCS 5052, Springer, 2008, pp. 184–198. doi:10.1007/978-3-540-68265-3_12.

[8] M. Dastani, W. Jamroga, Reasoning about strategies of multi-agent programs, in: Proceedings of AAMAS 10, 2010, pp. 997–1004.

[9] C. Dima, F. L. Tiplea, Model-checking atl under imperfect information and perfect recall semantics is undecidable, CoRR abs/1102.4225.

[10] E. M. Clarke, E. A. Emerson, A. P. Sistla, Automatic verification of finite-state concurrent systems using temporal logic specifications, ACM Trans. Program. Lang. Syst. 8 (2) (1986) 244–263. doi:10.1145/5397.5399. URL http://doi.acm.org/10.1145/5397.5399

[11] C. Baier, J.-P. Katoen, Principles of Model Checking, The MIT Press, 2008.

[12] E. M. Clarke, O. Grumberg, D. Peled, Model Checking, MIT Press, 1999.

[13] R. Fagin, J. Y. Halpern, Y. Moses, M. Y. Vardi, Reasoning about Knowledge, MIT Press, Cambridge, 1995.

[14] F. Laroussinie, N. Markey, G. Oreiby, On the expressiveness and complexity of atl, CoRR abs/0804.2435.

[15] A. Lomuscio, W. Penczek, Symbolic model checking for temporal-epistemic logics, SIGACT News 38 (3) (2007) 77–99. doi:10.1145/1324215.1324231.

[16] E. Grädel, Positional determinacy of infinite games, in: V. Diekert, M. Habib (Eds.), STACS 2004, Vol. 2996 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2004, pp. 4–18. doi:10.1007/978-3-540-24749-4_2.

[17] W. Thomas, On the synthesis of strategies in infinite games, in: E. Mayr, C. Puech (Eds.), STACS 95, Vol. 900 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 1995, pp. 1–13. doi:10.1007/3-540-59042-0_57.

[18] K. R. Apt, E. Grädel, Lectures in Game Theory for Computer Scientists, 1st Edition, Cambridge University Press, New York, NY, USA, 2011.

[19] W. Jamroga, N. Bulling, Comparing variants of strategic ability, in: Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Volume One, IJCAI'11, AAAI Press, 2011, pp. 252–257.

[20] S. Busard, C. Pecheur, Pynusmv: Nusmv as a python library, in: G. Brat, N. Rungta, A. Venet (Eds.), Nasa Formal Methods 2013, Vol. 7871 of LNCS, Springer-Verlag, 2013, pp. 453–458.

[21] S. Busard, C. Pecheur, H. Qu, F. Raimondi, Improving the model checking of strategies under partial observability and fairness constraints, in: S. Merz, J. Pang (Eds.), Formal Methods and Software Engineering, Vol. 8829 of Lecture Notes in Computer Science, Springer International Publishing, 2014, pp. 27–42. doi:10.1007/978-3-319-11737-9_3.
URL http://dx.doi.org/10.1007/978-3-319-11737-9_3

[22] J. Pilecki, M. Bednarczyk, W. Jamroga, Synthesis and verification of uniform strategies for multi-agent systems, in: N. Bulling, L. van der Torre, S. Villata, W. Jamroga, W. Vasconcelos (Eds.), Computational Logic in Multi-Agent Systems, Vol. 8624 of Lecture Notes in Computer Science, Springer International Publishing, 2014, pp. 166–182. doi:10.1007/978-3-319-09764-0_11.
URL http://dx.doi.org/10.1007/978-3-319-09764-0_11

[23] X. Huang, R. van der Meyden, Symbolic model checking epistemic strategy logic (2014).
URL http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8654/8590

[24] J. Calta, D. Shkatov, H. Schlingloff, Finding uniform strategies for multi-agent systems, in: J. Dix, J. Leite, G. Governatori, W. Jamroga (Eds.), Computational Logic in Multi-Agent Systems, Vol. 6245 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2010, pp. 135–152. doi:10.1007/978-3-642-14977-1_12.

[25] A. Lomuscio, H. Qu, F. Raimondi, MCMAS: A model checker for the verification of multi-agent systems, in: Proceedings of CAV 2009, Vol. 5643 of LNCS, Springer, 2009, pp. 682–688. doi:10.1007/978-3-642-02658-4_55.

Appendix A. $ATLK_{IrF}$ and memoryless strategies

This section proves that the objectives of games involved in $ATLK_{IrF}$ do not need memory to be won, filling the gap between the semantics presented in Section 2 and the proof of correctness of the algorithm of Section 3. To show that, we use game theory results. We first have to transform the models and $ATLK_{IrF}$ properties into games. Thanks to this transformation, we can use several existing results about other objectives to prove that our objectives do not need memory to be won.

A *game graph* [16] is a structure $G = (V, V_0, V_1, E, \Omega)$ where

- V is a finite set of states, partitioned into V_0 and V_1. V_i are states where player i chooses the next state;
- $E : V \times V$ is a transition function;
- $\Omega : V \to C$ assigns a color of C to each state.

A *play* of G is an infinite sequence of states $v_0v_1v_2...$ of V such that $(v_i, v_{i+1}) \in E, \forall i \geq 0$.

To a game graph G we can associate a *winning condition* Win representing the set of plays that are winning for each player. The couple (G, Win) represents a *game*.

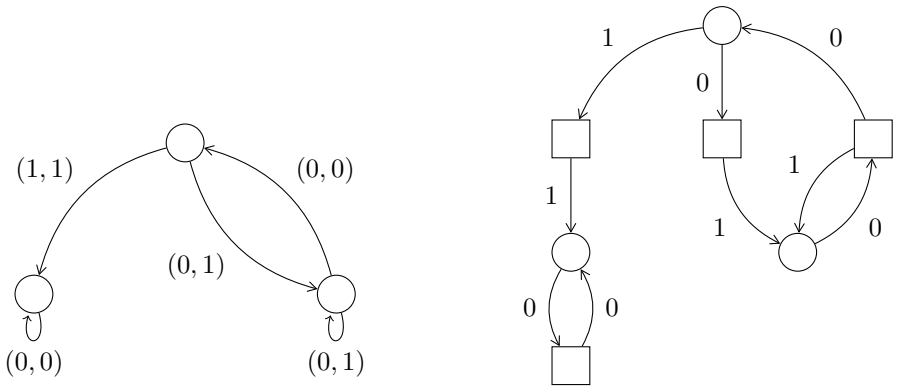
A *strategy* for player i in a game graph is a function $f_i : V^*V_i \to V$ associating to a finite prefix ending in a state of V_i the successor to choose. We say that a strategy f_i for player i is *memoryless* if the same successor is given for any two prefixes ending in the same state, that is, if $f_i(\pi v) = f_i(\pi' v)$ for any $\pi, \pi' \in V^*$, $v \in V_i$.

Given a model $M = \langle Ag, S, Act, T, I, \{\sim_i\}_{i\in Ag}, V, FC\rangle$ and an $ATLK_{IrF}$ property $\langle\Gamma\rangle\psi$, we can build the corresponding game $(G_{M,\Gamma,\psi}, Win_{M,\Gamma,\psi})$.
$G_{M,\Gamma,\psi} = (V, V_0, V_1, E, \Omega)$ where

- to each state $s \in S$ corresponds a state $v_{0,s}$ in V_0;
- to each state $s \in S$ and action $a_\Gamma \in \mathit{enabled}(s, \Gamma)$ corresponds a state v_{1,s,a_Γ} in V_1;
- $V = V_0 \cup V_1$;
- $(v_{0,s}, v_{1,s,a_\Gamma}) \in E$ for any $s \in S$ and $a_\Gamma \in \mathit{enabled}(s, \Gamma)$, and $(v_{1,s,a_\Gamma}, v_{0,T(s,a)}) \in E$, for any $s \in S$ and $a = a_\Gamma \sqcup a_{\overline{\Gamma}}$, where $a_\Gamma \in \mathit{enabled}(s, \Gamma)$ and $a_{\overline{\Gamma}} \in \mathit{enabled}(s, \overline{\Gamma})$;
- the set of colors C is composed of all fairness constraints $fc \in FC$ and all first-depth state subformulas of ψ;
- the states $v_{0,s}$ and v_{1,s,a_Γ} $(\forall a_\Gamma \in \mathit{enabled}(s, \Gamma))$ are colored by Ω with all the first-depth state subformulas of ψ and the fairness conditions $fc \in FC$ that hold in s.

Intuitively, Γ is mapped to player 0 and $\overline{\Gamma}$ to player 1. We decompose each state s of M into a single V_0 state, from which Γ can choose a V_1 state, where player 1 can choose to go to one V_0 state. The states are labelled with the first-depth state subformulas of ψ that are satisfied by their corresponding state. Note that, by construction, the game graph $G_{M,\Gamma,\psi}$ is always *bipartite*, that is, V_i states only lead to V_{1-i} states. An example of a model and the corresponding game graph is given in Figure A.7 (omitting coloring). Note that we only consider deterministic models here, but it is not a limitation (see Section 2.2).

We define the acceptance condition $Win_{M,\Gamma,\psi}$ of the game as an LTL formula with colors of $G_{M,\Gamma,\psi}$ as atomic propositions. The set of plays that are winning for player 0 are the ones satisfying the LTL formula; the other plays, satisfying the negation of the formula, are winning for player 1. The acceptance condition $Win_{M,\Gamma,\psi}$ depends on the set of fairness conditions FC of M and on the property



(a) A model (b) The corresponding game graph

Figure A.7: A model and its corresponding game graph. Γ is the first agent, $\overline{\Gamma}$ is the second one. Vertices are states; circles in the game graph are player 0 states, squares are player 1 states; edges are transitions, labelled with actions (for clarity, in the game graph).

ψ:

$$\text{if } \psi = X\phi, \text{ then } Win_{M,\Gamma,\psi} = \bigwedge_{fc \in FC} GFfc \implies XX\phi; \tag{A.1}$$

$$\text{if } \psi = \phi_1 U \phi_2, \text{ then } Win_{M,\Gamma,\psi} = \bigwedge_{fc \in FC} GFfc \implies \phi_1 U \phi_2; \tag{A.2}$$

$$\text{if } \psi = \phi_1 W \phi_2, \text{ then } Win_{M,\Gamma,\psi} = \bigwedge_{fc \in FC} GFfc \implies \phi_1 W \phi_2. \tag{A.3}$$

The objective of Equation A.1 says that a strategy is winning in state v for player 0 if all paths enforced by this strategy starting at v meet at least one $fc \in FC$ only finitely many times or their third state—that is the next state belonging to the same V_i as v—is labelled by ϕ. The objective of Equation A.2 says that winning strategies only enforce paths meeting at least one $fc \in FC$ only finitely many times or reaching a state labelled by ϕ_2 through states labelled by ϕ_1. Similarly, the objective of Equation A.3 says that winning strategies only enforce paths meeting at least one $fc \in FC$ only finitely many times or composed of states labelled by ϕ_1 until a state labelled by ϕ_2, if any.

By construction, given an objective $\langle\Gamma\rangle\psi$, there is a winning strategy in s of M iff there is a winning strategy in the corresponding state $v_{0,s}$ of the game $(G_{M,\Gamma,\psi}, Win_{M,\Gamma,\psi})$. This is expressed by the following lemma.

Lemma 12. *Let ψ be an $ATLK_{IrF}$ path formula, s be a state of a model M. $s \models \langle\Gamma\rangle\psi$ in M iff there exists a winning strategy for player 0 in the state $v_{0,s}$ of the corresponding game $(G_{M,\Gamma,\psi}, Win_{M,\Gamma,\psi})$.*

Proof. Let f_Γ be a strategy for Γ in M. The corresponding strategy f_0 in $(G_{M,\Gamma,\psi}, Win_{M,\Gamma,\psi})$ is defined as follows. For any prefix path $\pi = s_0 \xrightarrow{a_{\Gamma_1} \sqcup a_{\overline{\Gamma}_1}}$

$s_1...s_n$ in M, $f_\Gamma(\pi)$ returns an action a_Γ; to π corresponds, in $G_{M,\Gamma,\psi}$, the prefix path $\pi' = v_{0,s_0}v_{1,s_0,a_{\Gamma_1}}v_{0,s_1}...v_{0,s_n}$. $f_0(\pi')$ simply returns the successor of v_{0,s_n} given by $v_{1,s_n,f_\Gamma(\pi)}$, that is, the state corresponding to Γ playing $f_\Gamma(\pi)$ in s_n.

Note that while the labelling of an infinite path $\pi = s_0s_1s_2...$ with atomic propositions in M leads to the infinite sequence $V(s_0)V(s_1)V(s_2)...$, the corresponding path $\pi' = v_{0,s_0}v_{1,s_0,a_{\Gamma_1}}v_{0,s_1}v_{1,s_1,a_{\Gamma_2}}v_{0,s_2}v_{1,s_2,a_{\Gamma_2}}...$ in $G_{M,\Gamma,\psi}$ is labelled with the same labels, but repeated twice in sequence: the labelling of π' is $V(s_0)V(s_0)V(s_1)V(s_1)V(s_2)V(s_2)...$

Let us show that if f_Γ is winning for ψ in a state s of M, then f_0 is winning for the corresponding objective in state $v_{0,s}$ of $(G_{M,\Gamma,\psi}, Win_{M,\Gamma,\psi})$. We can prove it by contradiction. If f_0 is not winning, there is a path π' starting at $v_{0,s}$ enforced by f_0 that does not satisfy the objective, that is, π' meets each $fc \in FC$ infinitely often and violates ψ. The corresponding path π in M starting at s is enforced by f_Γ since the corresponding actions are used, and is labelled with the same atomic propositions as shown above. π is fair—since it meets each $fc \in FC$ infinitely often—and violates ψ. Thus π does not satisfy the objective, the strategy enforces a fair path violating ψ, and thus is not winning, leading to the contradiction.

To show that if f_0 is winning, then f_Γ is winning for the corresponding objective, we can simply use the symmetric argument. □

Note that the result is true for Γ but not for $\overline{\Gamma}$. This means that $\overline{\Gamma}$ do not necessarily win in states of M corresponding to states of the game $(G_{M,\Gamma,\psi}, Win_{M,\Gamma,\psi})$ where player 1 wins. This is due to the fact that, in $(G_{M,\Gamma,\psi}, Win_{M,\Gamma,\psi})$, player 1 can choose a different successor depending on the previous choice of player 0; on the other hand, in M, for $\overline{\Gamma}$ to have a winning strategy, they must be able to choose one winning action, regardless of the one chosen by Γ. In fact, if we want to speak about strategies of $\overline{\Gamma}$, it is necessary to build another game graph $(G_{M,\overline{\Gamma},\psi}, Win_{M,\overline{\Gamma},\psi})$, in which the results are different.

We can now prove that in the game theory framework, the objectives above do not need memory to be won by player 0, that is, player 0 has a strategy to win his objective iff he has a memoryless strategy to win it. Before proving this, we need to show some intermediate results.

First, let $Safe_i(\Phi)$ be defined as the winning region for player i for the game $(G, G\phi)$, where ϕ labels all states of Φ (and only them). In $v \in Safe_i(\Phi)$, player i has a memoryless strategy to stay in Φ [17]. Furthermore, we define the new game graph $G' = G \backslash Z$ as the game graph G where all states of Z are removed, as well as their adjacent edges.

Lemma 13. *Let $G = (V, V_0, V_1, E, \Omega)$ and $Win = \bigvee_{fc \in FC} FG\neg fc$, player 0 has a winning memoryless strategy in his winning region of the game (G, Win).*

Proof. A Streett-Rabin condition on colors of C is composed of two sets $(\mathcal{F}_0, \mathcal{F}_1)$ such that $\mathcal{F}_0 \subseteq \mathcal{P}(C)$, $\mathcal{F}_1 = \mathcal{P}(C) \backslash \mathcal{F}_0$ and $\mathcal{F}_0$ is closed under union. Player i wins a play iff the set of colors met infinitely often on the play belongs to $\mathcal{F}_i$. Given a Streett-Rabin condition, player 1 has a winning memoryless strategy in his winning region [16].

We can express the winning condition $Win = \bigvee_{fc \in FC} FG\neg fc$ has a Streett-Rabin condition by swapping the roles of players (player 0 becomes player 1, and vice versa) and setting

$$\mathcal{F}_0 = \{FC\} \text{ and } \mathcal{F}_1 = \mathcal{P}(FC) \backslash \mathcal{F}_0. \tag{A.4}$$

Because player 0 wins this objective iff all fairness constraints $fc \in FC$ are met infinitely often, player 1 wins if the original $Win = \bigvee_{fc \in FC} FG\neg fc$ objective is met. Finally, Theorem 4 of [16] tells us that player 1—player 0 in our original game—has a winning memoryless strategy to win in his winning region. Thus the $Win = \bigvee_{fc \in FC} FG\neg fc$ objective can be won with a memoryless strategy in the winning region of player 0. □

Lemma 14. *Let $G = (V, V_0, V_1, E, \Omega)$ and $Reach_i(A, B) = W_k$ such that $W_k = W_{k+1}$, where*

$$\begin{aligned} W_0 &= B \\ W_{j+1} &= \begin{array}{l} W_j \cup \{v \in V_i \cap A | \exists v' \in W_j \text{ s.t. } (v, v') \in E\} \\ \cup \{v \in V_{1-i} \cap A | \nexists v' \in V \backslash W_j \text{ s.t. } (v, v') \in E\}. \end{array} \end{aligned}$$

Let $\Phi_1 = \Omega^{-1}(\phi_1)$ and $\Phi_2 = \Omega^{-1}(\phi_2)$, the two sets of states labeled with ϕ_1 and ϕ_2, respectively. $Reach_i(\Phi_1, \Phi_2)$, computes the winning region of player i in the game graph G with winning condition $\phi_1 U \phi_2$. Furthermore, both players have a memoryless strategy to win in their winning region.

Proof. Let $v \in Reach_i(\Phi_1, \Phi_2)$, we can prove by induction over j that player i has a strategy to win $\phi_1 U \phi_2$ in v. If $v \in W_0 = \Phi_2$, v is labeled by ϕ_2 and player i already wins $\phi_1 U \phi_2$. Otherwise, $v \in W_{j+1}$. In this case, v is labeled by ϕ_1. Furthermore, player i can lead, from v, to states v' of W_j where he has, by induction hypothesis, a strategy to win $\phi_1 U \phi_2$. He thus has a strategy to win $\phi_1 U \phi_2$ in v: going to any such v', and playing the winning strategy there.

On the other hand, player $1 - i$ has a strategy in $v \in V \backslash Reach_i(\Phi_1, \Phi_2)$ to win $\neg\phi_2 W(\neg\phi_1 \wedge \neg\phi_2)$. Indeed, either v is in $V \backslash (\Phi_1 \cup \Phi_2)$, satisfies $\neg\phi_1 \wedge \neg\phi_2$, and player $1 - i$ already wins his objective. Or v is in $(V \cap \Phi_1) \backslash Reach_i(\Phi_1, \Phi_2)$ and satisfies ϕ_1 and $\neg\phi_2$. In this latter case, in v, player $1 - i$ can lead to states v' of $V \backslash Reach_i(\Phi_1, \Phi_2)$, otherwise it would belong to $Reach_i(\Phi_1, \Phi_2)$ because it belongs to Φ_1 and could not avoid to reach $Reach_i(\Phi_1, \Phi_2)$. That is, from v, player $1 - i$ has a strategy to stay in $V \backslash Reach_i(\Phi_1, \Phi_2)$ or to reach $V \backslash (\Phi_1 \cup \Phi_2)$, i.e. a strategy to win $G\neg\phi_2 \vee (\neg\phi_2 U(\neg\phi_1 \wedge \neg\phi_2)) = \neg\phi_2 W(\neg\phi_1 \wedge \neg\phi_2)$.

Finally, we can show that in their winning regions, both players have a memoryless winning strategy. Indeed, in $v \in W_{j+1}$, player i can choose any successor belonging to W_j. This way, it will not come back to v before winning his objective, thus the strategy is memoryless in v. Furthermore, in $v \in W_0$, he can choose any successor he wants. On the other hand, in $v \in V \backslash (\Phi_1 \cup \Phi_2)$, player $1-i$ already wins and can choose any successor and, in $v \in (V \cap \Phi_1) \backslash Reach_i(\Phi_1, \Phi_2)$, player $1 - i$ can always choose the same successor in $V \backslash Reach_i(\Phi_1, \Phi_2)$, thus the strategy is memoryless in v. □

Note that Lemma 14 implies that $\phi_1 W \phi_2$ can also be won without memory: if player i wins $\neg\phi_2 U(\neg\phi_1 \wedge \neg\phi_2)$ in $Reach_i(\overline{\Phi_2}, \overline{\Phi_1} \cap \overline{\Phi_2})$, then player $1-i$ wins $\phi_1 W \phi_2$ without memory in $V \backslash Reach_i(\overline{\Phi_2}, \overline{\Phi_1} \cap \overline{\Phi_2})$.

Now that we showed that the objectives $Win = \bigvee_{fc \in FC} FG\neg fc$, $Win = \phi_1 U \phi_2$ and $Win = \phi_1 W \phi_2$ do not need memory to be won, we can show that the objectives of interest (Equations A.1, A.2 and A.3) neither need memory to be won. For this, we show how to compute the winning region and then show that, from this computation, we can extract a memoryless strategy.

Lemma 15. *Let $G = (V, V_0, V_1, E, \Omega)$ and $Win = \bigvee_{fc \in FC} FG\neg f \vee XX\phi$. If the game graph G is bipartite, player 0 has a winning memoryless strategy in his winning region of the game (G, Win).*

Proof. First, let $Pre_i(Z)$ be defined as

$$Pre_i(Z) = \begin{array}{l} \{v \in V_i | \exists v' \in Z \text{ s.t. } (v, v') \in E\} \\ \cup \{v \in V_{1-i} | \nexists v' \in V \backslash Z \text{ s.t. } (v, v') \in E\}. \end{array} \tag{A.5}$$

Intuitively, $Pre_i(Z)$ returns the set of states in which player i can force to reach Z in one step.

Algorithm 6 returns the winning regions for both players in the game (G, Win). It relies on the Pre_i operator and on the sub-algorithm $Solve(G, \bigvee_{fc \in FC} FG\neg fc)$ returning the winning regions for both players in the game $(G, \bigvee_{fc \in FC} FG\neg fc)$.

Algorithm 6: $Solve(G, \bigvee_{fc \in FC} FG\neg fc \vee XX\phi)$

Data: $G = (V, V_0, V_1, E, \Omega)$ a game graph with $\{\phi\} \cup FC$ the set of colors coloring the states of G.
Result: (W_0, W_1), the winning regions of players 0 and 1, respectively, in the game $(G, \bigvee_{fc \in FC} FG\neg fc \vee XX\phi)$.

$(W_{01}, W_{11}) = Solve(G, \bigvee_{fc \in FC} FG\neg fc)$
$W_{02} = Pre_0(W_{01} \cup Pre_0(W_{01} \cup \Omega^{-1}(\phi)))$
return $(W_{02}, V \backslash W_{02})$

We can show that Algorithm 6 effectively computes the winning regions in the game of interest. First, note that in W_{01}, player 0 has a strategy to win $\bigvee_{fc \in FC} FG\neg fc$ and thus wins $\bigvee_{fc \in FC} FG\neg fc \vee XX\phi$. In $v \in Pre_0(W_{01} \cup \Omega^{-1}(\phi))$, player 0 has a strategy to win $\bigvee_{fc \in FC} FG\neg fc \vee X\phi$ because either the successors are in W_{01}, and the player wins $\bigvee_{fc \in FC} FG\neg fc$, or in $\Omega^{-1}(\phi)$, and he wins $X\phi$. Finally, in $v \in Pre_0(W_{01} \cup Pre_0(W_{01} \cup \Omega^{-1}(\phi)))$, player 0 can force a successor in which he has a strategy to win $X(\bigvee_{fc \in FC} FG\neg fc \vee X\phi) = \bigvee_{fc \in FC} XFG\neg fc \vee XX\phi = \bigvee_{fc \in FC} FG\neg fc \vee XX\phi$.

On the other hand, player 1 has a strategy to win $\bigwedge_{fc \in FC} GFfc \wedge XX\neg\phi$ in $v \in V \backslash W_{02}$. Indeed, in $v \in V \backslash Pre_0(W_{01} \cup Pre_0(W_{01} \cup \Omega^{-1}(\phi)))$, player 1 has a strategy to avoid W_{01} and $Pre_0(W_{01} \cup \Omega^{-1}(\phi))$ in one step and in $v \in V \backslash Pre_0(W_{01} \cup \Omega^{-1}(\phi))$ he has a strategy to avoid W_{01} and $\Omega^{-1}(\phi)$ in one

step. Thus he can always avoid W_{01} and avoid $\Omega^{-1}(\phi)$ in two steps, thus wins $\bigwedge_{fc\in FC} GFfc \wedge XX\neg\phi$.

Finally, we can show that there exists a memoryless strategy for player 0 in his winning region W_{02}. Indeed, if $v \in W_{01}$, player 0 has a memoryless strategy to win $\bigvee_{fc\in FC} FG\neg fc$ (see Lemma 13). In $v \in W_{02}\backslash W_{01}$, player 0 can choose a successor in $W_{01} \cup Pre_0(W_{01} \cup \Omega^{-1}(\phi))$. Then, he can either play to win $\bigvee_{fc\in FC} FG\neg fc$ (if possible), otherwise he can choose a successor of $W_{01} \cup \Omega^{-1}(\phi)$. For the strategy to be memoryless, the choices made in both steps must be either identical, or made in different states. Because the game graphs under interest are bipartite, in $v \in V$ player 0 cannot choose v as the successor, and the two choices are always made in different states. Thus, the strategy described above is memoryless, and player 0 has a memoryless strategy to win $\bigvee_{fc\in FC} FG\neg fc \vee XX\phi$. □

Lemma 16. *Let $G = (V, V_0, V_1, E, \Omega)$ and $Win = \bigvee_{fc\in FC} FG\neg fc \vee \phi_1 U \phi_2$. Player 0 has a winning memoryless strategy in his winning region of the game (G, Win).*

Proof. Algorithm 7 returns the winning regions for both players in the game (G, Win). This algorithm relies on the sub-algorithm $Solve(G, \bigvee_{fc\in FC} FG\neg fc)$, $Reach_i(\Phi_1, \Phi_2)$ and $Safe_i(\Phi)$ and on Algorithm 8 returning the winning region in a game $(G, \phi_1 U \bigvee_{fc\in FC}(G(\phi_1 \wedge \neg fc))$.

Algorithm 7: $Solve(G, \bigvee_{fc\in FC} FG\neg fc \vee \phi_1 U \phi_2)$

Data: $G = (V, V_0, V_1, E, \Omega)$ a game graph with $\{\phi_1, \phi_2\} \cup FC$ the set of colors coloring the states of G.

Result: (W_0, W_1), the winning regions of players 0 and 1, respectively, in the game $(G, \bigvee_{fc\in FC} FG\neg fc \vee \phi_1 U \phi_2)$.

$P = \Omega^{-1}(\phi_1)$
$Q = \Omega^{-1}(\phi_2)$
$(W_{01}, W_{11}) = Solve(G, \bigvee_{fc\in FC} FG\neg fc)$
$W_{02} = Reach_0(P, Q \cup W_{01})$
$(W_{03}, W_{13}) = Solve(G\backslash W_{02}, \phi_1 U \bigvee_{fc\in FC} G(\phi_1 \wedge \neg fc))$
return $(W_{03} \cup W_{02}, W_{13})$

First, let us show that Algorithm 8 effectively returns the winning regions of both players in game $(G, \phi_1 U \bigvee_{fc\in FC} G(\phi_1 \wedge \neg fc))$ and that there is a winning memoryless strategy for player 0 in his winning region. We can prove it by induction over the size of the game. The base case of the induction arises when $W'_{01} = \emptyset$. In this case, $\bigcup_{fc\in FC} Safe_0(P \cap V\backslash\Omega^{-1}(fc)) = \emptyset$. Thus for all $v \in V$, player 1 has a strategy to win $F(\neg\phi_1 \vee fc)$ for any $fc \in FC$. Player 1 has thus a strategy to win $\bigwedge_{fc\in FC} GF(\neg\phi_1 \vee fc)$: play the strategy to win $F(\neg\phi_1 \vee fc)$ for $fc \in FC$, this ends up in any state from which the player can still play the strategy to win $F(\neg\phi_1 \vee fc')$ for another $fc' \in FC$,

Algorithm 8: $Solve(G, \phi_1 U \bigvee_{fc \in FC} G(\phi_1 \wedge \neg fc))$

Data: $G = (V, V_0, V_1, E, \Omega)$ a game graph with $\{\phi_1\} \cup FC$ the set of colors coloring the states of G.

Result: (W_0, W_1), the winning regions of players 0 and 1, respectively, in the game $(G, \phi_1 U \bigvee_{fc \in FC} G(\phi_1 \wedge \neg fc))$.

$P = \Omega^{-1}(\phi_1)$
$W'_{01} = Reach_0(P, \bigcup_{fc \in FC} Safe_0(P \cap V \backslash \Omega^{-1}(fc)))$
if $W'_{01} = \emptyset$ **then**
  return $(\emptyset, V)$
else
  $(W'_{02}, W'_{12}) = Solve(G \backslash W'_{01}, \phi_1 U \bigvee_{fc \in FC} G(\phi_1 \wedge \neg fc))$
  return $(W'_{02} \cup W'_{01}, W'_{12})$

and so on. Finally, since player 1 can win $\bigwedge_{fc \in FC} GF(\neg\phi_1 \vee fc)$, he can win $\bigwedge_{fc \in FC} F(\neg\phi_1 \vee fc) W(\neg\phi_1 \wedge F(\neg\phi_1 \vee fc))$, his objective of interest, in any state $v \in V$.

The inductive case supposes that $Solve(G', \phi_1 U \bigvee_{fc \in FC} G(\phi_1 \wedge \neg fc))$ returns the correct winning regions for any game graph G' smaller than G. In this case, we can show that (1) W'_{01} is winning for player 0 in G; (2) if W'_{02} is winning for player 0 in $G \backslash W'_{01}$, then it is winning in G; (3) if W'_{12} is winning for player 1 in $G \backslash W'_{01}$, then it is winning in G too.

Let us show the first point. In $v \in W'_{01}$, player 0 can reach in a finite number of steps through states of P a state v' of $\bigcup_{fc \in FC} Safe_0(P \cap V \backslash \Omega^{-1}(fc))$. In v', player 0 has a strategy to win $G(\phi_1 \wedge \neg fc)$ for a $fc \in FC$. Thus, in v, player 0 has a strategy to win $\phi_1 U \bigvee_{fc \in FC} G(\phi_1 \wedge \neg fc)$: use the strategy to reach v' and then use a the strategy to win $G(\phi_1 \wedge \neg fc)$ for the specified $fc \in FC$.

Let us show now the second point. If v belongs to W'_{02}, player 0 has a strategy in $G \backslash W'_{01}$ to win $\phi_1 U \bigvee_{fc \in FC} G(\phi_1 \wedge \neg fc)$. We can show that whenever such a strategy meets a state v' with a successor in W'_{01}, player 1 has no interest to play this transition: from v to v', all states satisfy ϕ_1, by definition of the objective of the sub-game; thus, if player 1 chooses the successor in W'_{01}, player 0 can still play the strategy to win $\phi_1 U \bigvee_{fc \in FC} G(\phi_1 \wedge \neg fc)$, and thus all the resulting paths will be winning for player 0. The winning region W'_{02} is thus also winning in G.

Furthermore, let us show the third point. If $v \in W'_{12}$, player 1 has also a wining strategy in G: whenever such a strategy meets a state v' with a successor in W'_{01}, player 1 can still win. Indeed, either $v' \notin P$, thus player 0 already loses; or, $v' \in P$, $v' \in V_1$ and v' has another successor outside W'_{01}, otherwise v' would belong to W'_{01}. Thus, player 1 can avoid this successor in W'_{01} and follow the original strategy.

Finally, let us show that there is a memoryless strategy for player 0 in his winning region. In W'_{01}, he has a memoryless strategy: in $W'_{01} \backslash \bigcup_{fc \in FC} Safe_0(P \cap$

$V \backslash \Omega^{-1}(fc)$) use the strategy to reach $\bigcup_{fc \in FC} Safe_0(P \cap V \backslash \Omega^{-1}(fc))$, and then use the memoryless strategy to win $G(\phi_1 \wedge \neg fc)$ for one possible $fc \in FC$. Furthermore, let us assume that in W'_{02}, there exists a memoryless strategy f_0; the corresponding strategy in G is to follow f_0 whenever W'_{01} is not reached, and the strategy described above when W'_{01} is reached.

Let us now show that Algorithm 7 effectively returns the winning regions for both players in game $(G, \bigvee_{fc \in FC} FG\neg fc \vee \phi_1 U \phi_2)$ and that there is a winning memoryless strategy for player 0 in his winning region. For this, we have to show that (1) W_{01} is winning for player 0, (2) W_{02} is winning for player 0, (3) if W_{03} is winning for player 0 in $G \backslash W_{02}$, then it is winning for him in G and (4) if W_{13} is winning for player 1 in $G \backslash W_{02}$, then it is winning for him in G.

First, W_{01} is effectively winning for player 0: he has a strategy to win $\bigvee_{fc \in FC} FG\neg fc$, thus a strategy to win the objective of interest. Second, in W_{02}, player 0 can reach $Q \cup W_{01}$ through P, thus can win $\bigvee_{fc \in FC} FG\neg fc \vee (\phi_1 U \phi_2)$: play the strategy to reach $Q \cup W_{01}$ through P, if Q is reached, player 0 is done, otherwise, play the strategy to win $\bigvee_{fc \in FC} FG\neg fc$, and thus win the objective. The third point can be proved by considering two cases. In the first case, the winning strategy for player 0 in $G \backslash W_{02}$ never reaches a state with a successor in W_{02}: the strategy is also winning for $\phi_1 U \bigvee_{fc \in FC} G(\phi_1 \wedge \neg fc)$ in G, thus winning for $\bigvee_{fc \in FC} FG\neg fc$, thus winning for the objective of interest. In the second case, the strategy meets a state v' with a successor in W_{02}. Choosing this successor will not allow player 1 to win: all states already met satisfy ϕ_1, thus choosing the successor in W_{02} still allow player 0 to win the objective of interest.

Let us now show the fourth point. If $v \in W_{13}$, then player 1 can win

$$Win' = \bigwedge_{fc \in FC} F(\neg\phi_1 \vee fc) W(\neg\phi_1 \wedge \bigwedge_{fc \in FC} F(\neg\phi_1 \vee fc)) \tag{A.6}$$

with strategy f_1. Note that $W_{13} \cap Q = \emptyset$, thus v satisfies $\neg\phi_2$. If f_1 never meets a state v' with a successor in W_{02}, then f_1 wins the objective Win' in G, thus all enforced paths either satisfy $G \bigwedge_{fc \in FC} F(\neg\phi_1 \vee fc)$ and never meet $\neg\phi_1$, satisfy $\bigwedge_{fc \in FC} GFfc \wedge G(\neg\phi_1 \wedge \neg\phi_2)$, thus the objective of interest, or meet $\neg\phi_1 \wedge \neg\phi_2$, wins $\neg\phi_2 U(\neg\phi_1 \wedge \neg\phi_2)$, and from there can win $\bigwedge_{fc \in FC} GFfc$ (because out of W_{01}).

Otherwise, f_1 meets a state v' with a successor in W_{02}. Either v' satisfies $\neg\phi_1$ and player 1 can win $\bigwedge_{fc \in FC} GFfc$, thus the objective of interest, or v' satisfies ϕ_1, $v' \in V_1$, and v' has another successor in $V \backslash W_{02}$. Player 1 can thus choose this successor and follow the same strategy as above.

Finally, let us show that there is a memoryless strategy in the winning region of player 0: in W_{01}, he can play the memoryless strategy to win $\bigvee_{fc \in FC} FG\neg fc$; in $W_{02} \backslash W_{01}$, he can play the memoryless strategy to win $Q \cup W_{01}$, and if W_{01} is reached, he can play the corresponding strategy; finally, player 0 also has a memoryless strategy in W_{03}, ending the proof. □

Lemma 17. *Let* $G = (V, V_0, V_1, E, \Omega)$ *and* $Win = \bigvee_{fc \in FC} FG\neg fc \vee \phi_1 W \phi_2$. *Player* 0 *has a winning memoryless strategy in his winning region of the game* (G, Win).

Proof. Algorithm 9 returns the winning regions for both players in the game (G, Win). It relies on the sub-algorithm $Solve(G, \bigvee_{fc \in FC} FG\neg fc)$ and $Reach_i$.

Algorithm 9: $Solve(G, \bigvee_{fc \in FC} FG\neg fc \vee \phi_1 W \phi_2)$

Data: $G = (V, V_0, V_1, E, \Omega)$ a game graph with $\{\phi_1, \phi_2\} \cup FC$ the set of colors coloring the states of G.

Result: (W_0, W_1), the winning regions of players 0 and 1, respectively, in the game $(G, \bigvee_{fc \in FC} FG\neg fc \vee \phi_1 W \phi_2)$.

$P = \Omega^{-1}(\phi_1)$
$Q = \Omega^{-1}(\phi_2)$
$(W_{01}, W_{11}) = Solve(G, \bigvee_{fc \in FC} FG\neg fc)$
$G' = G \backslash W_{01}$
$W_{12} = Reach_1(G', V' \backslash Q, V' \backslash (P \cup Q))$
return $(V \backslash W_{12}, W_{12})$

We can show that Algorithm 9 computes the winning regions for both players in the game of interest. First, in $v \in W_{01}$, player 0 has a strategy to win $\bigvee_{fc \in FC} FG\neg fc$, thus a strategy to win the objective of interest $\bigvee_{fc \in FC} FG\neg fc \vee \phi_1 W \phi_2$. Second, in $v \in W_{12}$, player 1 has a strategy to win $\neg\phi_2 U(\neg\phi_1 \wedge \neg\phi_2)$ by definition of *Reach*. Thus, in $v \in V' \backslash W_{12}$, player 0 has a strategy to win $(\phi_1 \vee \phi_2) W ((\phi_1 \vee \phi_2) \wedge \phi_2) = \phi_1 W \phi_2$. Let us show that player 0 also has a strategy to win the objective of interest in $v \in V' \backslash W_{12}$ in G. Consider that the strategy never reaches a state with a successor in W_{01}. In this case, player 0 also wins $\phi_1 W \phi_2$ in G, thus the objective of interest. Consider now that the strategy reaches a states v' with a successor in W_{01}. Note that v' satisfies ϕ_1 or the game is already won, otherwise it would not be part of the strategy, and in that case $v' \in V_1$, otherwise it would belong to W_{01}. By choosing the successor in W_{01}, player 1 would also lose because then all paths would satisfy $\bigvee_{fc \in FC} FG\neg fc$, thus player 0 would also win.

In $v \in W_{12}$, player 1 has a strategy to win $\neg\phi_2 U(\neg\phi_1 \wedge \neg\phi_2)$ in G'. Let us now show that player 1 also has a strategy in v to win in G the objective of interest. First, consider that the strategy in v never reaches a state with a successor in W_{01}; in this case, the strategy also wins $\neg\phi_2 U(\neg\phi_1 \wedge \neg\phi_2)$ in G. Furthermore, when reaching a state satisfying $\neg\phi_1 \wedge \neg\phi_2$ (and this will eventually happen), player 1 is still in $V \backslash W_{01}$, and thus can win $\bigwedge_{fc \in FC} GFfc$ in this state. Player 1 can thus win $\neg\phi_2 U(\neg\phi_1 \wedge \neg\phi_2) \wedge \bigwedge_{fc \in FC} GFfc$, the objective of interest. Consider now that the strategy reaches a state v' with a successor in W_{01}. v' belongs to V_1 otherwise it would already belong to W_{01}. Either v' satisfies $\neg\phi_1 \wedge \neg\phi_2$, and belongs to $V \backslash W_{01}$, and can win the objective as above, or it satisfies $\phi_1 \wedge \neg\phi_2$, has successor in the next iteration of $Reach_1$, and thus can avoid to go into W_{01}, and also win the objective as above.

Let us show now that player 0 has a memoryless strategy in his winning region. Let $v \in W_{01}$, we know that player 0 has a memoryless strategy to win $\bigvee_{fc \in FC} FG\neg fc$, thus to win the objective of interest. Let $v \in V \backslash (W_{12} \cup W_{01})$, player 0 has a memoryless strategy to win $\phi_1 W \phi_2$. We can modify this strategy such that whenever a state in W_{01} is reached, the memoryless strategy to win $\bigvee_{fc \in FC} FG\neg fc$ is played, resulting into a memoryless strategy to win the objective of interest. □

Lemma 12 and Lemmas 15, 16 and 17 can now be combined into Theorem 18 saying that no memory is needed to win the objectives of interest.

Theorem 18. *Let s be a state of a given (implicit) model and ψ an $ATLK_{IrF}$ path formula. Γ have a memoryless strategy to enforce ψ in s if and only if they have a memory-full strategy to enforce ψ in s.*

Proof. By Lemma 12, Γ have a strategy to win ψ in s iff the state corresponding to s in the game $(G_{M,\Gamma,\psi}, Win_{M,\Gamma,\psi})$ is winning for player 0. Furthermore, by Lemmas 15, 16 and 17, for any objective of interest, there is a memoryless strategy in the winning region of player 0. Thus, Γ have a memoryless strategy to win ψ in s iff there is a memory-full strategy to win ψ in s. □