Alternating-time Temporal Logic on Finite Traces

Francesco Belardinelli[1], **Alessio Lomuscio**[2], **Aniello Murano**[3] and **Sasha Rubin**[3]
[1] Laboratoire IBISC, UEVE, France
[2] Department of Computing, Imperial College London, UK
[3] DIETI, Università degli Studi di Napoli, Italy
francesco.belardinelli@univ-evry.fr, a.lomuscio@imperial.ac.uk, murano@na.infn.it, rubin@unina.it

Abstract

We develop a logic-based technique to analyse finite interactions in multi-agent systems. We introduce a semantics for Alternating-time Temporal Logic (for both perfect and imperfect recall) and its branching-time fragments in which paths are finite instead of infinite. We study validities of these logics and present optimal algorithms for their model-checking problems in the perfect recall case.

1 Introduction

There is a long tradition in logic-based approaches to Artificial Intelligence (AI) to model the evolution of a system on the basis of an infinite sequence of states [Alur *et al.*, 2002; Mogavero *et al.*, 2014]. This approach is heavily influenced by works in reactive systems and related temporal logic approaches, where a computing system is conceived as an entity that interacts with the environment indefinitely [Harel and Pnueli, 1985].

While this modelling choice has long proven to be valuable in regards to AI as well, many areas of AI are typically characterised by *executions that terminate*. For example, plays in classic ludic games are finite, phases in negotiation and coordination protocols are finite [Jennings *et al.*, 2001], business processes are naturally modelled using finite traces [Pesic *et al.*, 2010], and in automated planning successful executions are often finite (e.g., in classical planning and in strong solutions to fully observable non-deterministic planning) [Geffner and Bonet, 2013].

In this paper we introduce ATL^*_f, Alternating-time Temporal Logic for the modelling and verification of multi-agent systems with finite computations. The syntax is the same as ATL^* [Alur *et al.*, 2002], while the semantics is novel. Precisely, a model comes equipped with a special set of final states F and strategic quantifiers only range on paths that end in a state of F. We automatically get finite-trace semantics for the natural syntactic fragments of ATL^*_f, i.e., ATL_f, CTL_f, and CTL^*_f, the finite-trace variants of the branching-time temporal logics CTL^* and CTL, that we also study. We use ordinary finite-automata, operating on finite words, instead of infinite words or trees, to give optimal algorithms for the model-checking problems of these logics (with perfect recall).

In particular, our algorithms sidestep intrinsic difficulties of model-checking ATL^* that are due to automata operating over infinite words or trees, e.g., determinisation of Büchi automata, which has been resistant to efficient implementation [Tsai *et al.*, 2014], or emptiness of alternating parity tree automata, which is conjectured to be in PTIME and known to be in NP and co-NP [Emerson *et al.*, 2001]. There are algorithms that avoid determinisation or solving emptiness of alternating tree automata, notably Safraless decision procedures [Kupferman *et al.*, 2006; Filiot *et al.*, 2011]. These algorithms, while undeniably elegant, are still complex and tailored to reasoning about infinite traces. In contrast, just like the procedures in [De Giacomo and Vardi, 2015], our algorithms are extremely simple: they only involve automata operating on *finite words*, and simple standard constructions on these such as the classic subset construction for determinisation. We consider this a significant technical improvement.

Related Work. In recent years, several contributions in AI have focused on variants of LTL to reason about finite-computation systems. Along this line, well studied is LTL_f, a variant of the Linear-time Temporal Logic LTL where formulas are interpreted over finite traces (see [De Giacomo and Vardi, 2015] for a survey). This logic has proved particularly useful in planning [Bacchus and Kabanza, 2000; Baier and McIlraith, 2006; Gerevini *et al.*, 2009; De Giacomo and Vardi, 2013; Camacho *et al.*, 2017], and in business process modelling [Pesic *et al.*, 2010; Montali *et al.*, 2010; De Giacomo *et al.*, 2014a]. Moreover, recent work [De Giacomo *et al.*, 2014b] has compared the finite- and infinite-trace semantics for LTL, showing surprising differences both at the modelling and verification levels.

There are two standard semantics for LTL, i.e., over traces and over transition systems [Baier and Katoen, 2008]. In [De Giacomo and Vardi, 2013] model-checking LTL_f is considered for semantics over traces, i.e., models are of the form $\pi_0\pi_1\cdots\pi_n$ where π_i is a set of atomic proposition. In contrast, our models of ATL^*_f are transition systems. This opens up the possibility of modelling complex strategic and branching-time properties of transition systems rather than just properties of traces. Furthermore, thanks to the strategic operators in ATL^*_f, we reduce LTL_f synthesis [De Giacomo and Vardi, 2015] to ATL^*_f model checking (see Section 4).

To the best of our knowledge finite traces have already been considered in the context of branching-time temporal logics, but never for logics of strategies. In [Vardi and Stockmeyer, 1985] a finite-trace semantics for CTL* is introduced. This formal account, which is a special case of our semantics, is analysed briefly in Remark 2. The expressivity of CTL* on parameterised sets of traces, which includes the set of finite traces, has been investigated in [Emerson and Halpern, 1986], but the related model-checking problem has not been tackled as far as we know. Our framework is also related to Game Logic [Pauly and Parikh, 2003], which was designed to reason about games, and has applications to protocol analysis, e.g., cake cutting and secret exchange. However, Game Logic is a generalisation of PDL from programs to games, whereas our framework is in the tradition of branching and alternating-time temporal logics. Moreover, its model checking problem was studied for the two-player case only. Motivated by multi-agent strategic reasoning over finite traces, [Gutierrez *et al.*, 2017] introduced and studied iterated Boolean games with LDL goals over finite traces. Differently from us, they consider the problem of the existence of Nash equilibria.

Lastly, [Kong and Lomuscio, 2018] developed a verification approach for finite traces of multi-agent systems against LDL$_f$K specifications. Their approach focuses on the verification algorithms and the resulting implementation. In contrast, we are here concerned with the theoretical underpinning of an approach supporting the strategic interplay of the agents on finite executions.

2 Concurrent Game Systems with Final States

In this section we introduce concurrent game structures (CGS) endowed with final states. Given a set X of elements, let $u \in X^\omega \cup X^+$ denote an infinite or non-empty finite sequence on X. Then, we write u_i for its $(i+1)$-th element, i.e., $u = u_0 u_1 \ldots$, and $|u|$ for its length, with $|u| = \infty$ for $u \in X^\omega$. The first element of u is denoted by $first(u)$, and its last by $last(u)$. We write $u_{\geq i}$ for its suffix $u_i u_{i+1} \ldots$ starting in u_i, and $u_{\leq i}$ for its prefix $u_0 \ldots u_i$. The empty sequence is denoted by ϵ. For a vector $v \in \prod_i X_i$ we denote its i-th element by $v(i)$.

We now introduce our class of relevant models.

Definition 1 (CGS with final states)**.** *A* concurrent game structure (CGS) with final states *is a tuple* $G = \langle Ag, AP, \{Act_a\}_{a \in Ag}, S, s_0, F, \delta, \lambda \rangle$ *where:*

- *Ag is the finite non-empty set of* agent names*;*
- *AP is the finite non-empty set of* atomic propositions*;*
- *Act_a is the finite non-empty set of* actions *for $a \in Ag$;*
- *S is the finite non-empty set of* states*; with* initial state *$s_0 \in S$, and the non-empty set $F \subseteq S$ of* final states*;*
- *$\delta : S \times Jact \to S$ is the* transition function*, where $Jact = \prod_{a \in Ag} Act_a$ is the set of* joint actions*;*
- *$\lambda : S \to 2^{AP}$ is the* labelling *function that assigns a set of atoms to each state s.*

Def. 1 extends the standard definition of CGS [Alur *et al.*, 2002] by explicitly identifying a set $F \subseteq S$ of final states. Final states can be thought of as end states in games (e.g., all configurations in which a king is in check-mate in chess), or goal states in plans, or remarkable states along a system's execution. Final states allow us to talk about plays that have both a start and an end.

A *run* (resp. *history*) is an infinite (resp. finite) sequence $\pi \in S^+ \cup S^\omega$ of states complying with the transition function, i.e., for every $i < |\pi|$ there exists a joint action $J \in Jact$ such that $\delta(\pi_i, J) = \pi_{i+1}$. We denote with Run (resp. $Hist$) the set of all runs (resp. histories).

A *path* is a history π that ends in a final state, i.e., $last(\pi) \in F$. In the next section, we will define strategic quantifiers to range on paths, i.e., finite sequences ending in final states. We denote the set of all paths by $Path$.

A *(perfect recall or memoryful) strategy* for agent a is a function $\sigma_a : Hist \to Act_a$. A strategy σ is *positional* or *memoryless* if for all $h, h' \in Hist$, $last(h) = last(h')$ implies $\sigma(h) = \sigma(h')$. The set of all memoryful (resp. memoryless) strategies is denoted as $\Sigma_R(G)$ (resp. $\Sigma_r(G)$). For $A \subseteq Ag$ and $y \in \{R, r\}$, let $\sigma_A : A \to \Sigma_y(G)$ denote a *joint strategy* associating a (memoryful or memoryless) strategy σ_a with each agent $a \in A$.

For $s \in S$, a joint strategy σ_A, and $x \in \{\infty, f\}$, we write $\mathsf{out}_x(s, \sigma_A)$, called the *infinite*, resp. *finite, outcomes of σ_A from s*, for the set of runs, resp. paths, $\pi \in Run \cup Path$ such that π is consistent with s and σ_A. That is, $\pi \in \mathsf{out}_x(s, \sigma_A)$ iff (i) $\pi_0 = s$; (ii) for every $i \geq 0$, there exists a joint action $J_i \in Jact$ such that $\pi_{i+1} \in \delta(\pi_i, J_i)$ and for every $a \in A$, $J_i(a) = \sigma_A(a)(\pi_{\leq i})$.

3 ATL* on Finite Traces

In this section we introduce the language ATL* and its fragment ATL [Alur *et al.*, 2002]. Then, we interpret them on finite as well as infinite traces.

Syntax Fix finite sets *AP* of *atomic propositions (atoms)* and Ag of *agents*. The *state (φ) and path (ψ) formulas over AP and Ag* are built using the following grammar:

$$\begin{aligned} \varphi &::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\!\langle A \rangle\!\rangle \psi \\ \psi &::= \varphi \mid \neg\psi \mid \psi \wedge \psi \mid \mathrm{X}\, \psi \mid \psi \,\mathrm{U}\, \psi \end{aligned}$$

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

The class of ATL* *formulas* is the set of state formulas generated by the grammar. The *temporal operators* are X (read "next") and U (read "until"). The *strategy quantifier* is $\langle\!\langle A \rangle\!\rangle$ (read "the agents in A can enforce …"). We introduce the following abbreviations: $[\![A]\!]\psi ::= \neg\langle\!\langle A \rangle\!\rangle\neg\psi$ (read "no matter what the agents in A do …"), $\widetilde{\mathrm{X}}\, \psi ::= \neg\, \mathrm{X}\, \neg\psi$ (read "weak next"), and $\psi \,\mathrm{R}\, \psi' ::= \neg(\neg\psi \,\mathrm{U}\, \neg\psi')$ (read "releases"), $\mathrm{F}\, \psi ::= \mathsf{true} \,\mathrm{U}\, \psi$ (read "eventually") and $\mathrm{G}\, \psi ::= \mathsf{false} \,\mathrm{R}\, \psi$ (read "globally").

Hereafter we consider also the ATL fragment of ATL*. State formulas are built in the same way, whereas path formulas ψ are defined as follows:

$$\psi ::= \mathrm{X}\, \varphi \mid \widetilde{\mathrm{X}}\, \varphi \mid \varphi \,\mathrm{U}\, \varphi \mid \varphi \,\mathrm{R}\, \varphi$$

Notice that operators $\widetilde{\mathrm{X}}$ and R have to be assumed as primitive now. We discuss the reason why in Remark 4.

In the following we also consider the CTL and CTL* fragments of ATL and ATL* respectively. Specifically, in

CTL/CTL* we only allow the empty coalition $\emptyset$ in strategic operators. Further, we use the usual abbreviations $\mathrm{A}\,\psi ::= \langle\langle\emptyset\rangle\rangle\psi$ and $\mathrm{E}\,\psi ::= \neg\,\mathrm{A}\,\neg\psi \equiv [\![\emptyset]\!]\psi$.

Finally, we introduce the LTL fragment of CTL* as the set of path formulas generated by the grammar: $\psi ::= p \mid \neg\psi \mid \psi \wedge \psi \mid \mathrm{X}\,\psi \mid \psi\,\mathrm{U}\,\psi$, where $p \in AP$.

We provide all these different languages with a semantics based on finite traces.

Semantics Fix a CGS G with final states. For $x \in \{\infty, f\}$ and $y \in \{R, r\}$, we simultaneously define, by induction on the structure of formulas, the relations $(G, s) \models_{xy} \varphi$, where $s \in S$ and φ is a state formula, and $(G, \pi) \models_{xy} \psi$ where $\pi \in Run$ if $x = \infty$ and $\pi \in Path$ if $x = f$, and ψ is a path formula:

$$\begin{array}{lll}
(G,s) \models_{xy} p & \text{iff} & p \in \lambda(s) \\
(G,s) \models_{xy} \neg\varphi & \text{iff} & (G,s) \not\models_{xy} \varphi \\
(G,s) \models_{xy} \varphi \wedge \varphi' & \text{iff} & (G,s) \models_{xy} \varphi \text{ and } (G,s) \models_{xy} \varphi' \\
(G,s) \models_{xy} \langle\langle A\rangle\rangle\psi & \text{iff} & \text{for some joint strategy } \sigma_A \in \Sigma_y(G), \\
 & & \text{for all } \pi \in \mathsf{out}_x(s, \sigma_A),\ (G,\pi) \models_{xy} \psi. \\
(G,\pi) \models_{xy} \varphi & \text{iff} & (G,\pi_0) \models_{xy} \varphi \\
(G,\pi) \models_{xy} \neg\psi & \text{iff} & (G,\pi) \not\models_{xy} \psi \\
(G,\pi) \models_{xy} \psi \wedge \psi' & \text{iff} & (G,\pi) \models_{xy} \psi \text{ and } (G,\pi) \models_{xy} \psi' \\
(G,\pi) \models_{xy} \mathrm{X}\,\psi & \text{iff} & \pi_{\geq 1} \neq \epsilon \text{ and } (G,\pi_{\geq 1}) \models_{xy} \psi \\
(G,\pi) \models_{xy} \psi\,\mathrm{U}\,\psi' & \text{iff} & \text{for some } j < |\pi|,\ (G,\pi_{\geq j}) \models_{xy} \psi', \text{ and} \\
 & & \text{for all } k \text{ with } 0 \leq k < j,\ (G,\pi_{\geq k}) \models_{xy} \psi
\end{array}$$

For a state formula φ, we write $G \models_{xy} \varphi$ to mean that $(G, s_0) \models_{xy} \varphi$; whereas φ is a *validity*, or $\models_{xy} \varphi$, iff $G \models_{xy} \varphi$ for every CGS G with final states.

We consider the following sets of validities: For $x \in \{\infty, f\}$ and $y \in \{R, r\}$, let

$$\text{ATL}^*_{xy} = \{\varphi \in \text{ATL}^* \mid \models_{xy} \varphi\}$$
$$\text{ATL}_{xy} = \{\varphi \in \text{ATL} \mid \models_{xy} \varphi\}.$$

Notice that for $x = \infty$ we obtain the standard semantics for ATL* and ATL on infinite traces [Alur *et al.*, 2002]; while $y \in \{R, r\}$ discriminates between the perfect- and imperfect-recall variant. *We will drop the subscripts when they are clear from the context.*

We now unpack the meaning of the strategic operators.

Remark 1. *In standard* ATL* *($x = \infty$, $y = R$) we have validities $\langle\langle Ag\rangle\rangle\psi \leftrightarrow [\![\emptyset]\!]\psi$ and $[\![Ag]\!]\psi \leftrightarrow \langle\langle\emptyset\rangle\rangle\psi$. However, on finite traces we only have $\langle\langle Ag\rangle\rangle\psi \rightarrow [\![\emptyset]\!]\psi$ and $\langle\langle\emptyset\rangle\rangle\psi \rightarrow [\![Ag]\!]\psi$, as the converses fail. Indeed, for $x = f, y = R$:*

1. *$\langle\langle Ag\rangle\rangle\psi$ is true at state s iff there is some infinite path π starting in s such that every prefix of π ending in a final state in F satisfies ψ.*
2. *$[\![Ag]\!]\psi$ is true at state s iff for all infinite paths π starting in s, there is some prefix of π ending in a final state in F, that satisfies ψ.*
3. *$\mathrm{A}\,\psi$ defined as $\langle\langle\emptyset\rangle\rangle\psi$ is true at state s iff all finite paths starting in s and ending in F satisfy ψ.*
4. *$\mathrm{E}\,\psi$ defined as $[\![\emptyset]\!]\psi$ is true at state s iff that there exists a finite path starting in s, ending in F, that satisfies ψ.*

Further, for $x = f, y = r$ only the $\Rightarrow$-implications of (1) and (2) hold, as memoryless strategies may restrict the set of paths produced (whereas (3) and (4) still hold).

We now illustrate our framework with a simple scenario.

Example 1. *Consider any ludic game that has a definite end and/or states in which something "important" happens. E.g., in chess, a "checkmate" is one way to signal the end of the game, and a "check" signals that the opponent's King is under threat. We can model such games as CGS whose final states correspond to the end states or the "important" states.*[1] *Then, e.g., if the final states in the CGS corresponds to "check", a formula of the form $\langle\langle\{Black\}\rangle\rangle\,\mathrm{G}\,\mathrm{F}\,Queen$ says that the Black player has a strategy ensuring that whenever "check" occurs, the black queen is still on the board.*

Discussion We now make some remarks that motivate and discuss the syntax and semantics of ATL* on finite traces, i.e., for $x = f$.

We begin by discussing the relationship between our CTL$^*_{fR}$ and a previous definition of CTL* on finite traces.

Remark 2. *In [Vardi and Stockmeyer, 1985] a semantics for* CTL* *on finite traces is put forward. Specifically, path quantifiers range on all finite paths starting from a given state s, including the trivial path $\pi = \langle s\rangle$. Observe that we can mimic such an interpretation in CGS with final states by assuming that $F = S$. However, such a semantic choice validates peculiar* CTL* *formulas including $\mathrm{A}\,\mathrm{F}\,\varphi \leftrightarrow \varphi$ and $\mathrm{E}\,\mathrm{G}\,\varphi \leftrightarrow \varphi$. As a result, whenever path quantifiers range unrestrictedly over all finite paths, modalities $\mathrm{A}\,\mathrm{F}$ and $\mathrm{E}\,\mathrm{G}$ collapse to truth in the current state. We stress that nothing similar is the case in the semantics we propose, in particular $\mathrm{A}\,\mathrm{F}$ and $\mathrm{E}\,\mathrm{G}$ remain distinct.*

We now discuss, at the level of validities, the difference between perfect and imperfect recall.

Remark 3. *It is well-known that, for infinite traces, recall does not impact validities of* ATL, *i.e.,* $\text{ATL}_{\infty R} = \text{ATL}_{\infty r}$, *whereas it does make a difference for* ATL*, *i.e.,* $\text{ATL}^*_{\infty r} \subset \text{ATL}^*_{\infty R}$ *[Alur* et al.*, 2002]. Here we state without proof that similar results hold also for our semantics:*

$$\text{ATL}_{fr} = \text{ATL}_{fR}$$
$$\text{ATL}^*_{fr} \subset \text{ATL}^*_{fR}$$

Thus, ATL *and* ATL* *have the same distinguishing power as regards perfect/imperfect recall both on infinite and on finite traces.*

Now we discuss the reason we chose the syntax of ATL to include $\widetilde{\mathrm{X}}$, the dual of X.

Remark 4. *It is known that, differently from the case of infinite traces, on finite traces the next operator X is not self-dual. In particular, according to the semantics for $\models_f$ given in [De Giacomo and Vardi, 2015] in* LTL$_f$ *we have that*

$$\models_f \mathrm{X}\,\psi \rightarrow \neg\,\mathrm{X}\,\neg\psi \quad \textit{but} \quad \not\models_f \neg\,\mathrm{X}\,\neg\psi \rightarrow \mathrm{X}\,\psi$$

This remark justifies the introduction of weak next $\widetilde{\mathrm{X}}\,\psi$ *as $\neg\,\mathrm{X}\,\neg\psi$ (thus, e.g., differently from the case of infinite traces, $\langle\langle A\rangle\rangle\,\widetilde{\mathrm{X}}\,\varphi$ is no longer equivalent to $\langle\langle A\rangle\rangle\,\mathrm{X}\,\varphi$).*

[1] Of course such games may have huge state spaces that require additional techniques to analyse. This important dimension, however, is out of the scope of this paper.

Finally, in ATL_f *(for each type of recall* r, R*) we can introduce operators involving* $\llbracket A \rrbracket$ *as follows:*

$$\begin{aligned} \llbracket A \rrbracket \, \mathrm{X} \, \varphi &::= \neg \langle\langle A \rangle\rangle \, \widetilde{\mathrm{X}} \, \neg \varphi \\ \llbracket A \rrbracket \, \widetilde{\mathrm{X}} \, \varphi &::= \neg \langle\langle A \rangle\rangle \, \mathrm{X} \, \neg \varphi \\ \llbracket A \rrbracket (\varphi \, \mathrm{U} \, \varphi') &::= \neg \langle\langle A \rangle\rangle (\neg \varphi \, \mathrm{R} \, \neg \varphi') \\ \llbracket A \rrbracket (\varphi \, \mathrm{R} \, \varphi') &::= \neg \langle\langle A \rangle\rangle (\neg \varphi \, \mathrm{U} \, \neg \varphi') \end{aligned}$$

Finally, we present fixed points for formulas $\langle\langle A \rangle\rangle \psi_1 \, \mathrm{U} \, \psi_2$ and $\langle\langle A \rangle\rangle \psi_1 \, \mathrm{R} \, \psi_2$ in ATL. Such validities are relevant as symbolic model checking algorithms for ATL are based on similar fixed points.

Remark 5. *The following formulas are validities in* ATL_f *(for each type of recall* r, R*):*

$$\langle\langle A \rangle\rangle \psi_1 \, \mathrm{U} \, \psi_2 \quad \leftrightarrow \quad \langle\langle A \rangle\rangle \bot \vee (\psi_2 \vee (\psi_1 \wedge \langle\langle A \rangle\rangle \, \mathrm{X} \langle\langle A \rangle\rangle \psi_1 \, \mathrm{U} \, \psi_2)) \tag{1}$$

$$\langle\langle A \rangle\rangle \psi_1 \, \mathrm{R} \, \psi_2 \quad \leftrightarrow \quad \langle\langle A \rangle\rangle \bot \vee (\psi_1 \vee (\psi_2 \wedge \langle\langle A \rangle\rangle \, \mathrm{X} \langle\langle A \rangle\rangle \psi_1 \, \mathrm{R} \, \psi_2)) \tag{2}$$

We conclude by remarking that on infinite traces $\langle\langle A \rangle\rangle \bot$ is equivalent to $\bot$, which is not the case on finite traces. In fact, on finite traces $\langle\langle A \rangle\rangle \bot$ means that there exists a strategy for coalition A with an empty set of outcomes.

Model Checking We now state the main decision problem tackled in this work.

Definition 2 (Model Checking)**.** *Given a CGS G with final states and a formula* φ*,* model checking G against φ on infinite (resp. finite) traces, with perfect (resp. imperfect) recall *is the following decision problem: decide whether* $G \models_{xy} \varphi$ *for* $x = \infty$ *(resp.* $x = f$*) and* $y = R$ *(resp.* $y = r$*).*

Table 1 recalls the complexity of the model-checking problem for the logic $\mathrm{ATL}^*_{\infty R}$ and some of its fragments. Citations to the original papers are given.

Logic	Complexity	Reference
CTL	PTIME-c	[Clarke *et al.*, 1986]
CTL*	PSPACE-c	[Emerson and Lei, 1985]
ATL	PTIME-c	[Alur *et al.*, 2002]
ATL*	2EXPTIME-c	[Alur *et al.*, 2002]

Table 1: Complexity of model checking for $x = \infty, y = R$.

4 Complexity of Model Checking

In this section we explore the computational complexity of model checking ATL* (and its fragments) over finite traces and under the perfect-recall assumption. See Table 2 for a summary of the results and their corresponding references.

Logic	Complexity	Theorems
$\mathrm{CTL}_{f,R}$	PTIME-c	2, 4
$\mathrm{CTL}^*_{f,R}$	PSPACE-c	3, 5
$\mathrm{ATL}_{f,R}$	PTIME-c	2, 4
$\mathrm{ATL}^*_{f,R}$	2EXPTIME-c	1, 6

Table 2: Complexity of model checking for $x = f, y = R$.

For the rest of this section we assume $y = R$ *and do not write this subscript.*

4.1 Upper Bounds

We solve the model-checking problem for ATL^*_f by using an automata-theoretic approach. Since our paths are finite, we only need to use ordinary automata, i.e., deterministic word automata (DFW), non-deterministic word automata (NFW) [Hopcroft and Ullman, 1979], instead of ω-regular automata. This considerably simplifies the constructions. Overall, we reduce the problem $(G, s) \models \varphi$ to model checking a game with an LTL_f objective ψ; in turn, this can be solved by converting ψ into a DFW accepting the models of ψ (the DFW might be doubly-exponentially larger than the formula), then taking the product of the resulting DFW with the structure G to obtain a safety game that, in turn, can be solved by using the standard fix-point algorithm linearly in the size of the game. The rest of the section explores all the relevant cases in detail.

We begin with a special case that serves as a building block, i.e., $(G, s) \models_f \langle\langle A \rangle\rangle \psi$ where ψ is an LTL_f formula. The algorithm is presented in Figure 1. Intuitively, this corresponds to solving a game with the LTL_f objective ψ: the coalition A plays a game on a graph that simulates G and a DFW for ψ; it is trying to enforce that every play from s stays "safe", i.e., in states such that if the corresponding state of G is in F then the corresponding state of the DFW is accepting (this captures the semantics of $\langle\langle A \rangle\rangle$ which relativises paths to be those ending in F).

Algorithm: GameSolving(G, A, ψ)
Input: CGS G (states S), agents $A \subseteq Ag$, LTL_f formula ψ.
Output: The set $X \subseteq S$ such that $s \in X$ iff $(G, s) \models_f \langle\langle A \rangle\rangle \psi$.

1. Convert ψ into an equivalent NFW.
2. Convert the NFW into an equivalent DFW (D, d_0, Δ, F').
3. Form the product edge-labelled graph on states $S \times D$.
4. Form the set $\mathit{Safe} \subseteq S \times D$ as consisting of (s, d) such that $s \in F$ implies $d \in F'$.
5. Put $s \in X$ iff coalition A can ensure the play in the product, starting in $(s, \Delta(d_0, \lambda(s)))$, always stays in Safe.

Figure 1: Algorithm for solving games with LTL_f objectives in double-exponential time.

Proposition 1. *Model-checking* ATL^*_f *formulas of the form* $\langle\langle A \rangle\rangle \psi$ *where* ψ *is an* LTL_f*-formula is in* 2EXPTIME*. In particular, it is doubly-exponential in* ψ *and polynomial in G.*

Proof. The overall structure of the proposed algorithm is presented in Figure 2. We give some more details.

For step 1, we use the adaptation of the classic Vardi-Wolper construction to finite words, e.g. [De Giacomo and Vardi, 2015], to get an NFW that accepts all the traces (over atoms AP, i.e., alphabet 2^{AP}) that satisfy ψ. The number of states of the NFW is at most exponential in the size of the formula.

For step 2, we apply the standard subset construction for determinising NFW [Rabin and Scott, 1959] to get a DFW with state set D, initial state $d_0 \in D$, transition function $\Delta : D \times$

```
Algorithm: $ModelChecking(G, \varphi)$
Input: CGS $G$, ATL$_f^*$ formula $\varphi$
Output: The set $X \subseteq S$ such that $s \in X$ iff $(G, s) \models \varphi$.
  1. Introduce a new atom $p_\varphi$ and, for each state $s$, label $s$ by
     $p_\varphi$ (i.e., add $p_\varphi$ to $\lambda(s)$) as follows:
     (a) If $\varphi = p \in AP$ then label $s$ by $p_\varphi$ iff $p \in \lambda(s)$.
     (b) If $\varphi = \neg\varphi_1$ then label $s$ by $p_\varphi$ iff $s$ is not in the set
         output by $ModelChecking(G, \varphi_1)$.
     (c) If $\varphi = \varphi_1 \wedge \varphi_2$ then label $s$ by $p_\varphi$ iff $s$ is in the sets
         output by $ModelChecking(G, \varphi_i)$ for $i = 1, 2$.
     (d) If $\varphi = \langle\langle A \rangle\rangle \psi$ (so $\psi$ is a path formula) then
         i. Let $Max(\psi)$ be the set of maximal state-
            subformulas of $\psi$.
        ii. For every $\xi \in Max(\psi)$ label $s$ by $p_\xi$ if $s$ is in
            the set output by $ModelChecking(G, \xi)$.
       iii. Replace every occurrence in $\psi$ of $\xi \in Max(\psi)$
            by $p_\xi$.
        iv. Label $s$ by $p_\varphi$ iff $s$ is in the set output by
            $GameSolving(G, A, \psi)$.
  2. Output the set $X$ of $s \in S$ such that $s$ is labelled by $p_\varphi$.
```

Figure 2: Recursive algorithm for model-checking ATL$_f$ that calls the GameSolving algorithm (Figure 1).

$2^{AP} \to D$, and final states $F' \subseteq D$. The number of states of the DFW is at most exponential in the number of states of the NFW.

For step 3, we form an edge-labelled graph: the vertices are $S \times D$, edge labels are of the form $\alpha : A \to \cup_a Act_a$ (i.e., a tuple of actions of the agents in A), and edges $(s, d) \xrightarrow{\alpha} (s', d')$ if there is some joint action J such that (i) $J(a) = \alpha(a)$ for all $a \in A$, (ii) $\delta(s, J) = s'$, and (iii) $\Delta(d, \lambda(s')) = d'$.

For step 4 we build the set *Safe* of states of the product graph that the coalition A is trying to stay in.

For step 5 we introduce some notation regarding graph-games [Grädel *et al.*, 2002]. A *safety game* is a tuple (V, Σ, E, T) with: vertices V, actions Σ, labelled edges $E \subseteq V \times \Sigma \times V$, and safety set $T \subseteq V$. Note that the graph from step 3 combined with the set from step 4 forms a safety game. Solving a safety game concerns deciding from which vertices v there is a strategy for the agent that ensures every path stays in T (similarly to model-checking the formula $\langle\langle A \rangle\rangle \, \mathrm{G} \, Safe$). This can be solved in linear time using the greatest fix-point of the operation $Y \mapsto T \cap Pre(Y)$ where $Pre(Y) = \{v \in V : \exists\sigma.\forall w.E(v, \sigma, w) \to w \in Y\}$. □

We now provide an algorithm for model-checking ATL$_f^*$, see Figure 2. It follows the standard approach for model-checking ATL* [Alur *et al.*, 2002], i.e., we inductively mark the states of the G by those state subformula that hold. The atomic case is immediate, and the Boolean operations are done inductively. For the strategic operator $\langle\langle A \rangle\rangle \psi$ we may assume, by induction, that ψ is an LTL$_f$ formula, and so we call the GameSolving algorithm in Figure 1. This gives:

Theorem 1. *Model checking* ATL$_f^*$ *is in* 2EXPTIME.

Now, suppose the algorithm is applied to formulas of ATL$_f$. Then, in step $1(d)$ of the algorithm in Figure 2, the path formula ψ is of the form $\mathrm{X}\,p'$, $\widetilde{\mathrm{X}}\,p'$, $p' \,\mathrm{U}\, p''$ or $p' \,\mathrm{R}\, p''$ where p', p'' are atoms. So, the complexity of $GameSolving(G, A, \psi)$ is polynomial (Proposition 1).

Theorem 2. *Model checking* ATL$_f$*, and thus also* CTL$_f$*, is in polynomial time.*

Finally, to solve model-checking of CTL$_f^*$ we make a slight change to the above algorithm. In step $1(d)$ of Figure 2, instead of calling $GameSolving(G, \emptyset, \psi)$ (which costs double-exponential time), we call the LTL$_f$ model-checking procedure described in Figure 3. The complexity analysis is described in the accompanying proof.

```
Algorithm: LTLfModelChecking$(G, \psi)$
Input: CGS $G$, LTL$_f$ formula $\psi$.
Output: The set $X \subseteq S$ such that $s \in X$ iff $(G, s) \models \langle\langle \emptyset \rangle\rangle \psi$.
  1. Convert $\psi$ into an equivalent NFW $(N, I, \Delta, F')$.
  2. Form the product graph on states $S \times N$.
  3. Form the set Safe of states $(s, t)$ such that $s \in F$ implies
     $t \in F'$.
  4. Put $s \in X$ iff every path starting from states of the form
     $(s, n)$ for $n \in I$ stays in Safe.
```

Figure 3: Algorithm for model-checking G against LTL$_f$ formulas.

Theorem 3. *Model checking* CTL$_f^*$ *is in polynomial space.*

Proof. We discuss the steps of the algorithm in Figure 3. In step 1, the NFW can be formed by using an adaptation of the classic Vardi-Wolper construction for finite words, e.g., [Vardi and Wolper, 1994]. Denote its states N, initial states $I \subseteq N$, transition relation $\Delta \subseteq N \times 2^{AP} \times N$, and final states $F' \subseteq N$. This can be built in polynomial space.

In step 2, the graph has states $S \times N$, and edges $(s, n) \to (s', n')$ if there is some joint action J with $s' \in \delta(s, J)$ and $(n, \lambda(s'), n') \in \Delta$.

In step 3, form the set *Safe* as described. In step 4, for $s \in S$ and $n \in I$, we need to check that every path from (s, n) stays in *Safe*. This check can be done in logspace in the size of the graph. Since the graph is exponential, the whole algorithm runs in polynomial space. □

4.2 Lower Bounds

We now supply the lower-bounds for model checking ATL$_f^*$ and its fragments. For CTL$_f$ and ATL$_f$ we reduce from model-checking CTL, known to be PTIME-hard [Clarke *et al.*, 1986]; for CTL$_f^*$ we reduce from model-checking finite traces against LTL$_f$ formulas, known to be PSPACE-hard [De Giacomo and Vardi, 2013]; and for ATL$_f^*$ we reduce from LTL$_f$ synthesis, known to be 2EXPTIME-hard [De Giacomo and Vardi, 2015].

Theorem 4. *Model checking* CTL$_f$ *(and thus* ATL$_f$*) is* PTIME*-hard.*

Proof. The proof is almost identical to the one that shows that CTL model checking is PTIME-hard by reducing from

CIRCUIT-VALUE, a problem that is PTIME-complete when restricted to circuits that are monotone (no negation), synchronized (connections between gates respect layers), and with proper alternation, see Section 3.2.1 of [Schnoebelen, 2003]. Adapt that proof by letting the final states be the sink nodes 0 and 1. □

We now turn to the fragment CTL$^*_f$. In [De Giacomo and Vardi, 2013], a trace-based semantics is given for LTL$_f$. There, the model checking problem asks, given an LTL$_f$ formula φ and a finite trace π, to decide whether φ holds on π. We reduce this trace-based LTL$_f$ model-checking to our structure-based CTL$^*_f$ model-checking.

Theorem 5. *Model-checking* CTL$^*_f$ *is* PSPACE*-hard.*

Proof. Given $\pi = \pi_0\pi_1\pi_2\cdots\pi_{n-1}$, construct a CGS G_π with final states as follows: $Ag = \{1\}$, AP are the atoms appearing in π, $Act_1 = \{a\}$ (i.e., there is only one action), $S = \{i : 0 \leq i \leq n\}$ (the last state is used to make the transition function total), $F = \{n-1\}$ (i.e., the second last state is final), $\delta(i,a) = i+1$ if $i < n$, $\delta(n,a) = n$, and $\lambda(i) = \pi_i$ for $i < n$, and $\lambda(n) = \emptyset$. Note that the complexity of this translation is linear, and for every LTL$_f$ formula and finite trace π, we have that φ holds on π iff $G_\pi \models_f \mathrm{A}\,\varphi$. □

Finally, we turn to the logic ATL$^*_f$. We reduce from LTL$_f$ synthesis, known to be 2EXPTIME-complete [De Giacomo and Vardi, 2015]. Let X, Y be disjoint finite non-empty sets of Boolean variables. The LTL$_f$ *realisability problem (for turn-based players)* asks, given an LTL$_f$ formula φ over atoms $X \cup Y$, to decide if there is a function $c : (2^X)^+ \to 2^Y$ such that for every finite sequence $X_0X_1\ldots X_n$ with $X_i \subseteq X$, the sequence $(X_0 \cup c(X_0))(X_1 \cup c(X_0X_1))\ldots(X_n \cup c(X_0X_1\cdots X_n))$ satisfies φ. In this case we say φ is *realisable*. We now show how to reduce the LTL$_f$ realisability problem to ATL$^*_f$ model-checking.

Theorem 6. *Model-checking* ATL$^*_f$ *is* 2EXPTIME*-hard.*

Proof. The proof is similar to the proof that ATL* is 2EXPTIME-hard which reduces from LTL synthesis [Alur *et al.*, 2002]. The main difference is to deal with the fact that in our semantics paths are finite and end in final states.

Let φ be an LTL$_f$ formula over atoms $X \cup Y$. We define a CGS $G_{X,Y}$ with final states and an LTL$_f$ formula φ_d such that φ is realisable iff $G_{X,Y} \models \langle\!\langle\{2\}\rangle\!\rangle\, \mathrm{X}\,\mathrm{X}\,\varphi_d$.

Define $G_{X,Y}$ with final states $F = \{2\} \times 2^{X\cup Y}$ as follows: $Ag = \{1,2\}$, $AP = F$, $S = (\{1\} \times 2^X) \cup F \cup s_{init}$, $Act_1 = 2^X$, $Act_2 = 2^Y$, $s_0 = s_{init}$; labelling $\lambda(s) = \{s\}$ for $s \in F$, and $\lambda(s) = \emptyset$ otherwise; and transitions: $\delta(s_{init}, (X', \cdot)) = (1, X')$, $\delta((1,X'), (\cdot, Y')) = (2, X' \cup Y')$, and $\delta((2,\cdot), (X',\cdot)) = (1, X')$ (where $X' \subseteq X, Y' \subseteq Y$, and $\cdot$ stands for any set).

In words, the CGS models two players that take turns, with player 1 going first. On his turn player 1 chooses a subset of X, on her turn player 2 chooses a subset of Y, and the resulting sequence of sets of atoms (for traces ending in final states) is in the language $\emptyset\cdot(\emptyset\cdot\{\{z\} : z \in F\})^*$. For example, the finite word $\{x\} \cdot \{x,y\}$ over alphabet $2^{\{x,y\}}$ with $X = \{x\}, Y = \{y\}$ corresponds to the sequence of states of $G_{X,Y}$: $s_{init}, (1,\{x\}), (2,\{x\}), (1,\{x\}), (2,\{x,y\})$ whose labelling is $\emptyset \cdot \emptyset \cdot \{(2,\{x\})\} \cdot \emptyset \cdot \{(2,\{x,y\})\}$.

Define the formula φ_d as follows: $p_d = p$, $(\varphi \wedge \varphi')_d = \varphi_d \wedge \varphi'_d$, $(\neg\varphi)_d = \neg\varphi_d$, $(\mathrm{X}\,\varphi)_d = \mathrm{X}\,\mathrm{X}\,\varphi_d$, $(\varphi\,\mathrm{U}\,\varphi')_d = (F \to \varphi_d)\,\mathrm{U}(F \wedge \varphi_d)$. In words, φ_d simulates φ on the subsequence of the trace consisting of odd numbered positions.

Then, an LTL$_f$ formula φ over $X \cup Y$ is realisable iff $G_{X,Y} \models \langle\!\langle\{2\}\rangle\!\rangle\, \mathrm{X}\,\mathrm{X}\,\varphi_d$. Regarding complexity, the formula φ_d is polynomial in the size of φ. On the other hand, just as in [Alur *et al.*, 2002], LTL$_f$ synthesis is already 2EXPTIME-hard for fixed X and Y, and thus $G_{X,Y}$ has constant size. □

5 Conclusions

Motivated by problems in AI rather than program verification, we defined a logic with the same syntax as ATL* but in which paths are finite instead of infinite. Precisely, a model is equipped with a special set F of final states and strategic quantifiers only account for paths that end in a state of F.

This definition is general enough to capture previous proposals for CTL* and LTL on finite traces, as well as synthesis of LTL on finite traces. Indeed, in Section 4 we showed how to reduce model-checking finite traces against LTL$_f$ formulas to model-checking CGS with final states against CTL$^*_f$ formulas (Theorem 5), and how to reduce LTL$_f$ synthesis to ATL$^*_f$ (with perfect recall) model-checking (Theorem 6). It follows that the formalism ATL$^*_f$ that we introduced is rich enough to express two fundamental decision problems about LTL$_f$, and at no extra cost in terms of computational complexity.

Moreover, in previous proposals to reasoning about finite traces in structures [Vardi and Stockmeyer, 1985], it was assumed (in the language of this paper) that all states are final, which leads to validities such as $\mathrm{A}\,\mathrm{F}\,\varphi \leftrightarrow \varphi$ and $\mathrm{E}\,\mathrm{G}\,\varphi \leftrightarrow \varphi$ that are not intuitive or natural (see Remark 2). Our more refined semantics removes these undesirable validities.

Finally, we emphasise that, unlike algorithms for model checking ATL$^*_{\infty R}$ (e.g., [Alur *et al.*, 2002]), our algorithm for model-checking ATL$^*_{fR}$ does not use complex constructions on automata operating on infinite strings or infinite trees. In fact, we do not translate ATL$^*_{fR}$ (or its fragments) into ATL$^*_{\infty R}$. Instead, our algorithms only involve automata operating on finite words, and simple standard constructions on these, e.g., the classic subset construction for determinisation and a fix-point algorithm synthesising strategies.

In future work we plan to account also for the imperfect information agents might have about the environment as well as the state of other agents.

Acknowledgements

The authors acknowledge support of ANR JCJC Project SVeDaS (ANR-16-CE40-0021); the Royal Academy of Engineering; the Royal Society; and INDAM GNCS 2018 Project ”Metodi Formali per la Verifica e la Sintesi di Sistemi Discreti e Ibridi”.

References

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

[Bacchus and Kabanza, 2000] F. Bacchus and F. Kabanza. Using temporal logics to express search control knowledge for planning. *Artificial Intelligence*, 116(1-2):123–191, 2000.

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

[Baier and McIlraith, 2006] J. A. Baier and S. A. McIlraith. Planning with first-order temporally extended goals using heuristic search. In *AAAI*, pages 788–795, 2006.

[Camacho *et al.*, 2017] A. Camacho, E. Triantafillou, C. J. Muise, J. A. Baier, and S. A. McIlraith. Non-deterministic planning with temporally extended goals: LTL over finite and infinite traces. In *AAAI*, pages 3716–3724, 2017.

[Clarke *et al.*, 1986] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications: A practical approach. *ACM TOPLAS*, 8(2):244–263, 1986.

[De Giacomo and Vardi, 2013] G. De Giacomo and M. Y. Vardi. Linear temporal logic and linear dynamic logic on finite traces. In *IJCAI*, pages 854–860, 2013.

[De Giacomo and Vardi, 2015] G. De Giacomo and M. Y. Vardi. Synthesis for LTL and LDL on finite traces. In *IJCAI*, pages 1558–1564, 2015.

[De Giacomo *et al.*, 2014a] G. De Giacomo, R. De Masellis, M. Grasso, F. M. Maggi, and M. Montali. Monitoring business metaconstraints based on LTL and LDL for finite traces. In *BPM*, LNCS 8659, pages 1–17, 2014.

[De Giacomo *et al.*, 2014b] G. De Giacomo, R. De Masellis, and M. Montali. Reasoning on LTL on finite traces: Insensitivity to infiniteness. In *AAAI*, pages 1027–1033, 2014.

[Emerson and Halpern, 1986] E. A. Emerson and J. Y. Halpern. "sometimes" and "not never" revisited: on branching versus linear time temporal logic. *Journal of the ACM*, 33(1):151–178, 1986.

[Emerson and Lei, 1985] E. A. Emerson and C. L. Lei. Modalities for model checking: Branching time logic strikes back. In *POPL*, pages 84–96, 1985.

[Emerson *et al.*, 2001] E. A. Emerson, C. S. Jutla, and A. P. Sistla. On model checking for the μ-calculus and its fragments. *Theor. Comp. Sci.*, 258(1–2):491–522, 2001.

[Filiot *et al.*, 2011] E. Filiot, N. Jin, and J.-F. Raskin. Antichains and compositional algorithms for LTL synthesis. *Formal Methods in System Design*, 39(3):261–296, 2011.

[Geffner and Bonet, 2013] H. Geffner and B. Bonet. *A Concise Introduction to Models and Methods for Automated Planning*. Morgan & Claypool Publishers, 2013.

[Gerevini *et al.*, 2009] A. Gerevini, P. Haslum, D. Long, A. Saetti, and Y. Dimopoulos. Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. *Artificial Intelligence*, 173(5-6):619–668, 2009.

[Grädel *et al.*, 2002] E. Grädel, W. Thomas, and T. Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *LNCS*. Springer, 2002.

[Gutierrez *et al.*, 2017] J. Gutierrez, G. Perelli, and M. Wooldridge. Iterated games with LDL goals over finite traces. In *AAMAS*, pages 696–704, 2017.

[Harel and Pnueli, 1985] D. Harel and A. Pnueli. On the development of reactive systems. In *Logics and models of concurrent systems*, pages 477–498. Springer, 1985.

[Hopcroft and Ullman, 1979] J. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Adison-Wesley Publishing Company, 1979.

[Jennings *et al.*, 2001] N. R. Jennings, P. Faratin, A. Lomuscio, S. Parsons, M. Wooldridge, and C. Sierra. Automated negotiation: prospects, methods and challenges. *Group Decision and Negotiation*, 10(2):199–215, 2001.

[Kong and Lomuscio, 2018] J. Kong and A. Lomuscio. Model checking multi-agent systems against ldlk specifications on finite traces. In *AAMAS*, 2018. To Appear.

[Kupferman *et al.*, 2006] O. Kupferman, N. Piterman, and M. Y. Vardi. Safraless compositional synthesis. In *CAV*, LNCS 4144, pages 31–44, 2006.

[Mogavero *et al.*, 2014] F. Mogavero, A. Murano, G. Perelli, and M. Y. Vardi. Reasoning about strategies: On the model-checking problem. *ACM Transactions on Computational Logic*, 15(4):34:1–34:47, 2014.

[Montali *et al.*, 2010] M. Montali, M. Pesic, W. M. P. van der Aalst, F. Chesani, P. Mello, and S. Storari. Declarative specification and verification of service choreographiess. *ACM Transactions on the Web (TWEB)*, 4(1):3, 2010.

[Pauly and Parikh, 2003] M. Pauly and R. Parikh. Game logic - an overview. *Studia Logica*, 75(2):165–182, 2003.

[Pesic *et al.*, 2010] M. Pesic, D. Bosnacki, and W. M. P. ravan der Aalst. Enacting declarative languages using LTL: avoiding errors and improving performance. In *SPIN*, LNCS 6349, pages 146–161, 2010.

[Rabin and Scott, 1959] M. O. Rabin and D. S. Scott. Finite Automata and their Decision Problems. *IBM Journal of Research & Development*, 3:115–125, 1959.

[Schnoebelen, 2003] Ph. Schnoebelen. The complexity of temporal logic model checking. In *AiML02*, volume 4 of *Advances in Modal Logic*, pages 437–459. 2003.

[Tsai *et al.*, 2014] M.-H. Tsai, S. Fogarty, M. Y. Vardi, and Y.-K. Tsay. State of Büchi complementation. *Logical Methods in Computer Science*, 10(4), 2014.

[Vardi and Stockmeyer, 1985] M. Y. Vardi and L. Stockmeyer. Improved upper and lower bounds for modal logics of programs. In *STOC*, pages 240–251, 1985.

[Vardi and Wolper, 1994] M. Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, 1994.