Strategy Logic

Krishnendu Chatterjee, Thomas A. Henzinger, and Nir Piterman

EECS, University of California, Berkeley, USA
EPFL, Switzerland

Abstract. We introduce *strategy logic*, a logic that treats strategies in two-player games as explicit first-order objects. The explicit treatment of strategies allows us to handle nonzero-sum games in a convenient and simple way. We show that the one-alternation fragment of strategy logic, is strong enough to express Nash-equilibrium, secure-equilibria, as well as other logics that were introduced to reason about games, such as ATL, ATL*, and game-logic. We show that strategy logic is decidable, by constructing tree automata that recognize sets of strategies. While for the general logic, our decision procedure is non-elementary, for the simple fragment that is used above we show that complexity is polynomial in the size of the game graph and optimal in the formula (ranging between 2EXPTIME and polynomial depending on the exact formulas).

1 Introduction

In *graph games*, two players move a token across the edges of a graph in order to form an infinite path. The vertices are partitioned into player-1 and player-2 nodes, depending on which player chooses the successor node. The objective of player 1 is to ensure that the resulting infinite path lies inside a given winning set Ψ_1 of paths. If the game is zero-sum, then the goal of player 2 is to prevent this. More generally, in a nonzero-sum game, player 2 has her own winning set Ψ_2.

Zero-sum graph games have been widely used in the synthesis (or control) of reactive systems [19, 21], as well as for defining and checking the realizability of specifications [1, 7], the compatibility of interfaces [6], simulation relations between transition systems [16, 10], and for generating test cases [3], to name just a few of their applications. The study of nonzero-sum graph games has been more recent, with assume-guarantee synthesis [4] as one of its applications.

The traditional formulation of graph games consists of a two-player graph (the "arena") and winning conditions Ψ_1 and Ψ_2 for the two players (in the zero-sum case, $\Psi_1 = \neg\Psi_2$), and asks for computing the winning sets W_1 and W_2 of vertices for the two players (in the zero-sum case, determinacy [15] ensures that $W_1 = \neg W_2$). To permit the unambiguous, concise, flexible, and structured expression of problems and solutions involving graph games, researchers have introduced *logics* that are interpreted over two-player graphs. An example is the temporal logic ATL [2], which replaces the unconstrained path quantifiers of CTL with constrained path quantifiers: while the CTL formula $\forall\Psi$ asserts that the path property Ψ is inevitable —i.e., Ψ holds on all paths from a given state—

the ATL formula $\langle\langle 1\rangle\rangle\Psi$ asserts that Ψ is enforcible by player 1 —i.e., player 1 has a strategy so that Ψ holds on all paths that can result from playing that strategy. The logic ATL has proved useful for expressing proof obligations in system verification, as well as for expressing subroutines of verification algorithms.

However, because of limitations inherent in the definition of ATL, several extensions have been proposed [2], among them the temporal logic ATL*, the alternating-time μ-calculus, and a so-called *game logic* of [2]: these are motivated by expressing general ω-regular winning conditions, as well as tree properties of computation trees that result from fixing a strategy of one player (module checking [14]). All of these logics treat strategies implicitly through modalities. This is convenient for zero-sum games, but awkward for nonzero-sum games. Indeed, it was not known if Nash equilibria, one of the most fundamental concepts in game theory, can be expressed in these logics.

In order to systematically understand the expressiveness of game logics, and to specify nonzero-sum games, we study in this paper a logic that treats strategies as explicit first-order objects. For example, using explicit strategy quantifiers, the ATL formula $\langle\langle 1\rangle\rangle\Psi$ becomes $(\exists x \in \Sigma)(\forall y \in \Gamma)\Psi(x, y)$ —i.e., "there exists a player-1 strategy x such that for all player-2 strategies y, the unique infinite path that results from the two players following the strategies x and y, satisfies the property Ψ." Strategies are a natural primitive when talking about games and winning, and besides ATL and its extensions, Nash equilibria are naturally expressible in *strategy logic*.

As an example, we define *winning secure equilibria* [5] in strategy logic. A winning secure equilibrium is a special kind of Nash equilibrium, which is important when reasoning about the components of a system, each with its own specification. At such an equilibrium, both players can collaborate to satisfy the combined objective $\Psi_1 \wedge \Psi_2$. Moreover, whenever player 2 decides to abandon the collaboration and enforce $\neg\Psi_1$, then player 1 has the ability to retaliate and enforce $\neg\Psi_2$; that is, player 1 has a winning strategy for the relativized objective $\Psi_2 \implies \Psi_1$ ($\implies$ denotes implication). The symmetric condition holds for player 2; in summary: $(\exists x \in \Sigma)(\exists y \in \Gamma)[(\Psi_1 \wedge \Psi_2)(x, y) \wedge (\forall y' \in \Gamma)(\Psi_2 \implies \Psi_1)(x, y') \wedge (\forall x' \in \Sigma)(\Psi_1 \implies \Psi_2)(x', y)]$. Note that the same player-1 strategy x which is involved in producing the outcome $\Psi_1 \wedge \Psi_2$, must be able to win for $\Psi_2 \implies \Psi_1$; such a condition is difficult to state without explicit quantification over strategies.

Our results are twofold. First, we study the expressive power of strategy logic: we show that the logic is rich enough to express many interesting properties of zero-sum and nonzero-sum games that we know, including ATL*, game logic (and thus module checking), Nash equilibria, and secure equilibria. Indeed, ATL* and the equilibria can be expressed in a simple fragment of strategy logic with no more than one quantifier alternation (note the $\exists\forall$ alternation in the above formula for defining winning secure equilibria). We also show that the simple one-alternation fragment can be translated to ATL* (the translation in general is double exponential in the size of the formula) and thereby the equilibria can be expressed in ATL*.

Second, we analyze the computational complexity of strategy logic. We show that, provided all winning conditions are specified in linear temporal logic (or by word automata), strategy logic is decidable. The proof goes through automata theory, using tree automata to specify the computation trees that result from fixing the strategy of one player. The complexity is nonelementary, with the number of exponentials depending on the quantifier alternation depth in the formula. In the case of the simple one-alternation fragment of strategy logic, which suffices to express ATL^* and equilibria, we obtain much better bounds: for example, for infinitary path formulas (path formulas that are independent of finite prefixes), there is a linear translation of a simple one-alternation fragment formula to an ATL^* formula.

In summary, strategy logic provides a decidable language for talking in a natural and uniform way about all kinds of properties on game graphs, including zero-sum, as well as nonzero-sum, objectives. Of course, for more specific purposes, such as zero-sum reachability games, more restrictive and less expensive logics, such as ATL, are more appropriate; however, the consequences of such restrictions, and their relationships, is best studied within a clean, general framework such as the one provided by strategy logic. In other words, strategy logic can play for reasoning about games the same role that first-order logic with explicit quantification about time has played for temporal reasoning: the latter has been used to categorize and compare temporal logics (i.e., logics with implicit time), leading to a notion of completeness and other results in correspondence theory [13, 9].

2 Graph Games

A game graph $G = ((S, E), (S_1, S_2))$ consists of a directed graph (S, E) with a finite state space S, a set E of edges and a partition (S_1, S_2) of the state space S. The states in S_1 are player 1 states where player 1 chooses the successor and the states in S_2 are player 2 states where player 2 chooses the successor. For a state s we write $E(s)$ to denote the set $\{\, t \mid (s, t) \in E \,\}$ of out-going edges from s. We assume that every state $s \in S$ has at least one out-going edge, i.e., $E(s)$ is non-empty for all states.

Plays. A game is played by two players: player 1 and player 2, who form an infinite path in the game graph by moving a token along edges. They start by placing the token on an initial state and then they take moves indefinitely in the following way. If the token is on a state in S_1, then player 1 moves the token along one of the edges going out of the state. If the token is on a state in S_2, then player 2 does likewise. The result is an infinite path $\pi = \langle s_0, s_1, s_2, \ldots \rangle$ in the game graph; we refer to such infinite paths as *plays*. Hence given a game graph G, a play is an infinite sequence $\langle s_0, s_1, s_2, \ldots \rangle$ of states such that for all $k \geq 0$, $(s_k, s_{k+1}) \in E$. We write π for a play, and Π for the set of all plays.

Strategies. A *strategy* for a player is a recipe that specifies how to extend plays. Formally, a strategy σ for player 1 is a function $\sigma : S^* \cdot S_1 \to S$ that given a finite sequence of states, representing the history of the play so far, that ends

in a player 1 state, chooses the next state. A strategy must choose only available successors, i.e., for all $w \in S^*$ and all $s \in S_1$ we have $\sigma(w \cdot s) \in E(s)$. Strategies for player 2 are defined analogously. We denote by Σ and Γ the set of all strategies for player 1 and player 2, respectively. Given a starting state s, a strategy σ for player 1 and a strategy τ for player 2, there is a unique play, denoted as $\pi(s, \sigma, \tau) = \langle s_0, s_1, s_2, \ldots \rangle$, that is defined as follows: $s = s_0$, and for all $k \geq 0$, (a) if $s_k \in S_1$, then $\sigma(s_0, s_1, \ldots, s_k) = s_{k+1}$ and (b) if $s_k \in S_2$, then $\tau(s_0, s_1, \ldots, s_k) = s_{k+1}$.

3 Strategy Logic

In this section we present *strategy logic*. Let AP be a finite set of atomic propositions. A labeled game graph $(G, AP, \mathcal{L})$ consists of a game graph G along with a labeling function $\mathcal{L} : S \to 2^{AP}$ that maps every state s with the set of atomic propositions $\mathcal{L}(s)$ true in state s. There is a special atomic proposition **true** such that **true** $\in \mathcal{L}(s)$, for all $s \in S$. We now define the *strategy logic* that we consider in this paper. The logic consists of basic path formulas (as in LTL), basic strategy formulas that are path formulas with strategy variables, and with basic strategy formulas as atomic propositions we define a first-order logic of quantified strategy formulas. A state formula is obtained as boolean combination of *closed* quantified strategy formulas (i.e., formula with no free variables). We denote state formulas, basic path formulas, quantified strategy formulas, and closed quantified strategy formuals by Ω, Ψ, Λ and Φ, respectively. We use variables $x, x_1, x_2, \ldots$ to range over strategies for player 1 and denote the set of such variables as X; and use variables $y, y_1, y_2, \ldots$ to range over strategies for player 2 and denote the set of such variables as Y.

Syntax. The state formulas Ω, basic path formulas Ψ, and quantified strategy formulas Λ are defined as follows:

$$\Omega ::= p \mid \Phi \mid \Omega \wedge \Omega \mid \neg\Omega, \qquad \text{where } p \in AP.$$

$$\Psi ::= \Omega \mid \Psi \wedge \Psi \mid \neg\Psi \mid \bigcirc(\Psi) \mid \Psi \,\mathcal{U}\, \Psi.$$

$$\Lambda ::= \Psi(x, y) \mid \Lambda \wedge \Lambda \mid \neg\Lambda \mid Qx.\Lambda \mid Qy.\Lambda, \qquad \text{where } Q \in \{\exists, \forall\}, x \in X, y \in Y.$$

A closed formula Φ is a quantified strategy formula where all strategy variables are quantified. Observe that the closed formulas can be reused as atomic propositions. We formally define the notion of free variables and closed formulas as follows.

1. For a path formula Ψ we have $\mathsf{Free}(\Psi) = \emptyset$.
2. For a quantified strategy formula Λ we inductively define $\mathsf{Free}(\Lambda)$ as follows:
   - If $\Lambda = \Psi(x, y)$, then $\mathsf{Free}(\Lambda) = \{ x, y \}$.
   - If $\Lambda = \Lambda_1 \wedge \Lambda_2$, then $\mathsf{Free}(\Lambda) = \mathsf{Free}(\Lambda_1) \cup \mathsf{Free}(\Lambda_2)$.
   - If $\Lambda = \neg\Lambda_1$, then $\mathsf{Free}(\Lambda) = \mathsf{Free}(\Lambda_1)$.
   - If $\Lambda = Qx.\Lambda_1$, then $\mathsf{Free}(\Lambda) = \mathsf{Free}(\Lambda_1) \setminus \{ x \}$.
   - If $\Lambda = Qy.\Lambda_1$, then $\mathsf{Free}(\Lambda) = \mathsf{Free}(\Lambda_1) \setminus \{ y \}$.

A quantified strategy formula Λ is closed if $\mathsf{Free}(\Lambda) = \emptyset$, i.e., all the strategy variables appearing in the formula are quantified.

Strategy assignments. For a set $Z \subseteq X \cup Y$, an assignment $\mathcal{A}_Z$ assigns: (a) for every variable $x \in Z \cap X$, a strategy $\mathcal{A}_Z(x) \in \Sigma$; and (b) for every variable $y \in Z \cap Y$, a strategy $\mathcal{A}_Z(y) \in \Gamma$. For a set $U \subseteq Z$ we denote by $\mathcal{A}_Z \upharpoonright U$ to be the restriction of the assignment $\mathcal{A}_Z$ to the set U. Given a strategy assignment $\mathcal{A}_Z$ and $\sigma \in \Gamma$, we denote by $\mathcal{A}'_{Z \cup \{ x \}} = \mathcal{A}_Z \cup x \leftarrow \sigma$ the extension of the assignment $\mathcal{A}_Z$ defined as follows: for $w \in Z \cup \{ x \}$ we have $\mathcal{A}_{Z \cup \{ x \}}(w) = \mathcal{A}_Z(w)$ if $w \neq x$ and σ otherwise. The definition for $\mathcal{A}_Z \cup y \leftarrow \tau$ is similar.

Semantics for quantified strategy formulas. The semantics for path formulas Ψ is the usual semantics of LTL. We now describe the satisfaction in state s of a quantified strategy formula Λ with respect to a strategy assignment $\mathcal{A}_Z$ where $\mathsf{Free}(\Lambda) \subseteq Z$.

$$
\begin{array}{lll}
\text{Case 1. } \Lambda = \Psi(x,y). & (s, \mathcal{A}_Z) \models \Psi(x,y) & \text{iff } \pi(s, \mathcal{A}_Z(x), \mathcal{A}_Z(y)) \models \Psi. \\
\text{Case 2. } \Lambda = \Lambda_1 \wedge \Lambda_2. & (s, \mathcal{A}_Z) \models \Lambda_1 \wedge \Lambda_2 & \text{iff } (s, \mathcal{A}_Z \upharpoonright \mathsf{Free}(\Lambda_1)) \models \Lambda_1 \\
 & & \text{and } (s, \mathcal{A}_Z \upharpoonright \mathsf{Free}(\Lambda_2)) \models \Lambda_2. \\
\text{Case 3. } \Lambda = \neg \Lambda_1. & (s, \mathcal{A}_Z) \models \neg \Lambda_1 & \text{iff } (s, \mathcal{A}_Z) \not\models \Lambda_1. \\
\text{Case 4. } \Lambda = \exists x. \Lambda_1. & (s, \mathcal{A}_Z) \models \Lambda & \text{iff } \exists \sigma \in \Sigma.\ (s, \mathcal{A}_Z \cup x \leftarrow \sigma) \models \Lambda_1. \\
\text{Case 5. } \Lambda = \forall x. \Lambda_1. & (s, \mathcal{A}_Z) \models \Lambda & \text{iff } \forall \sigma \in \Sigma.\ (s, \mathcal{A}_Z \cup x \leftarrow \sigma) \models \Lambda_1. \\
\text{Case 6. } \Lambda = \exists y. \Lambda_1. & (s, \mathcal{A}_Z) \models \Lambda & \text{iff } \exists \tau \in \Gamma.\ (s, \mathcal{A}_Z \cup y \leftarrow \tau) \models \Lambda_1. \\
\text{Case 7. } \Lambda = \forall y. \Lambda_1. & (s, \mathcal{A}_Z) \models \Lambda & \text{iff } \forall \tau \in \Gamma.\ (s, \mathcal{A}_Z \cup y \leftarrow \tau) \models \Lambda_1.
\end{array}
$$

The semantics of a closed formula is as follows: (a) $[\![\Phi]\!] = \{ s \in S \mid (s, \mathcal{A}_\emptyset) \models \Phi \}$ and (b) $[\![\Psi]\!] = \{ s \in S \mid \exists \sigma.\ \exists \tau.\ \pi(s, \sigma, \tau) \models \Psi \}$.

We also introduce a simpler fragment of the logic where path formulas do not allow nesting of temporal operators. In a sense, this fragment has a CTL-like flavor and, as we show later, results in decision procedure of lower complexity. Formally, in the definition above, path formulas should be restricted to the following.

$$\Psi ::= \Omega \mid \Psi \wedge \Psi \mid \neg \Psi \mid \bigcirc(\Omega) \mid \Omega\ \mathcal{U}\ \Omega.$$

We call this subset the *unnested path formula subset*.

Examples of strategy logic formulas. We now present some examples of formulas of strategy logic. In the examples for a path formula Ψ we use the following notations: $\Diamond \Psi = \mathbf{true}\ \mathcal{U}\ \Psi$ and $\Box \Psi = \neg(\Diamond \neg \Psi)$, i.e., $\Diamond \Psi$ and $\Box \Psi$ denotes eventually and always conditions, respectively. We now show how to express formulas of the logic ATL and ATL^* [2]. The alternating time temporal-logic ATL^* consists of path formulas along with alternating path operators $\langle\langle 1 \rangle\rangle$, and $\langle\langle 2 \rangle\rangle$; existential path operator $\langle\langle 1, 2 \rangle\rangle$ and universal path operator $\langle\langle \emptyset \rangle\rangle$. The logic ATL can be obtained as a subclass of ATL^* where only unnested path formulas are considered. Some examples of ATL and ATL^* formulas and the

corresponding strategy logic formula is shown below: for a proposition p

$$\langle\langle 1\rangle\rangle(\Diamond p) = \{\ s \in S \mid \exists\sigma.\ \forall\tau.\ \pi(s,\sigma,\tau) \models \Diamond p\ \} = [\exists x.\ \forall y.\ (\Diamond p)(x,y)];$$
$$\langle\langle 2\rangle\rangle(\Box\Diamond p) = \{\ s \in S \mid \exists\tau.\ \forall\sigma.\ \pi(s,\sigma,\tau) \models \Box\Diamond p\ \} = [\exists y.\ \forall x.\ (\Box\Diamond p)(x,y)];$$
$$\langle\langle 1,2\rangle\rangle(\Box p) = \{\ s \in S \mid \exists\sigma.\ \exists\tau.\ \pi(s,\sigma,\tau) \models \Box p\ \}. = [\exists x.\ \exists y.\ (\Box p)(x,y)].$$

Consider the strategy logic formula: $\Phi = \exists x.\ (\exists y_1.\ (\Box p)(x,y_1) \wedge \exists y_2.\ (\Box q)(x,y_2))$. The formula is different from the formula $\langle\langle 1,2\rangle\rangle(\Box p) \wedge \langle\langle 1,2\rangle\rangle(\Box q)$ as the same strategy x must be used. It follows from the results of [2] that the formula Φ cannot be expressed in ATL^*.

One of the nice features of strategy logic is that we can restrict the kind of strategies that interest us. For example, the following formula describes the states from which player 1 can ensure the goal Φ_1 while playing against any strategy that ensures Φ_2 for player 2.

$$\exists x_1.\forall y_1.((\forall x_2.\Phi_2(x_2,y_1)) \implies \Phi_1(x_1,y_1))$$

The mental exercise of "I know that you know that I know that you know ..." can be played in strategy logic up to any constant level. The analog of the above formula, where the level of knowledge is nested up to level k is easy to write. In the full version, we show a game graph on which the knowledge nesting changes the fact whether player 1 can or cannot win. Formally, the formula above ("knowledge nesting 1") is different from the following formula with "knowledge nesting 2":

$$\exists x_1.\forall y_1.((\forall x_2.(\forall y_2.\Phi_1(x_2,y_2)) \implies \Phi_2(x_2,y_1)) \implies \Phi_1(x_1,y_1))$$

As another example, we consider the notion of *dominating* and *dominated* strategies [18]. Given a path formula Ψ, a strategy x_1 for player 1 *dominates* strategy x_2, if forall strategies y for player 2 whenever x_2 and y satisfies Ψ, then x_1 and y satisifies Ψ. A strategy x_1 is dominating if it dominates every strategy x_2. The following formula expresses that x_1 is a dominating strategy

$$\forall x_2.\forall y.\ \Psi(x_2,y) \implies \Psi(x_1,y).$$

Given a path formula Ψ, a strategy x_1 is *dominated* if there is a strategy x_2 such that whenever x_1 satisfies Ψ, then x_2 satisfies Ψ, and in addition, for some strategy of player 2 the strategy x_2 satisfies Ψ, while x_1 fails to satisfy Ψ. The following formula expresses that x_1 is a dominated strategy

$$\exists x_2.((\forall y_1.\ \Psi(x_1,y_1) \implies \Psi(x_2,y_1)) \wedge (\exists y_2.\ \Psi(x_2,y_2) \wedge \neg\Psi(x_1,y_2)))$$

The formulas for dominating and dominated strategies express properties about strategies and are not closed formulas.

4 Simple One-alternation Fragment of Strategy Logic

We define a subset of the logic. Intuitively, the alternation depth of a formula is the number of changes between $\exists$ and $\forall$ quantifiers (a formal definition is given

in Section 6). The subset we consider here is a subset of the formulas that allow only one alternation of quantifiers. We refer to this subset as the *simple one-alternation fragment*. We show later how several nonzero-sum game formulations can be easily captured in this fragment.

Syntax and semantics. We are interested in state formulas that depend on three path formulas: Ψ_1, Ψ_2, and Ψ_3. We would like to characterize the cases where there exist player 1 and player 2 strategies that win Ψ_1 and Ψ_2, respectively, and at the same time cooperate to achieve Ψ_3; or the dual of such cases. We use boolean combination of formulas of the following types and allow them to be used as state formulas as well.

$$\exists x_1.\ \exists y_1.\ \forall x_2.\ \forall y_2.\ \Psi_1(x_1, y_2) \wedge \Psi_2(x_2, y_1) \wedge \Psi_3(x_1, y_1)$$
$$\forall x_1.\ \forall y_1.\ \exists x_2.\ \exists y_2.\ \Psi_1(x_1, y_2) \wedge \Psi_2(x_2, y_1) \wedge \Psi_3(x_1, y_1)$$

Obviously, both formulas have one quantifier alternation. We introduce the following notation for formulas of this type.

$$(\exists\ \Psi_1,\ \exists\ \Psi_2,\ \Psi_3) \qquad | \qquad (\forall\ \Psi_1, \forall\ \Psi_2,\ \Psi_3)$$

Notation. For a path formula Ψ and a state s we define the following: the set $\mathsf{Win}_1(s, \Psi) = \{\ \sigma \in \Sigma \mid \forall \tau \in \Gamma.\ \pi(s, \sigma, \tau) \models \Psi\ \}$ denotes the set of player 1 strategies that satisfy the goal Ψ against all player 2 strategies, we also refer to the strategies in $\mathsf{Win}_1(s, \Psi)$ as the *winning* strategies for player 1 for Ψ from s. Analogously we define $\mathsf{Win}_2(s, \Psi) = \{\ \tau \in \Gamma \mid \forall \sigma \in \Sigma.\ \pi(s, \sigma, \tau) \models \Psi\ \}$. Using the notation of Win_1 and Win_2 the semantics of these state formulas can be also written as follows.

1. Let $\Phi = (\exists\ \Psi_1,\ \exists\ \Psi_2,\ \Psi_3)$, then $[\![\Phi]\!] = \{\ s \mid \exists \sigma \in \mathsf{Win}_1(s, \Psi_1).\ \exists \tau \in \mathsf{Win}_2(s, \Psi_2).\ \pi(s, \sigma, \tau) \models \Psi_3\ \}$.
2. Let $\Phi = (\forall\ \Psi_1,\ \forall\ \Psi_2,\ \Psi_3)$, then $[\![\Phi]\!] = \{\ s \mid \forall \sigma \in \mathsf{Win}_1(s, \Psi_1).\ \forall \tau \in \mathsf{Win}_2(s, \Psi_2).\ \pi(s, \sigma, \tau) \models \Psi_3\ \}$.

5 Expressive Power of Strategy Logic

In this section we show that ATL^*, ATL [2], and several formulations of nonzero-sum games can be expressed in the simple one-alternation fragment. We also show that *game logic*, which was introduced in [2] to express the module checking problem, can be expressed in the one-alternation fragment.

Expressing ATL^* and ATL. The basic semantics of the formulas of ATL^* and how they can be expressed in simple one-alternation fragment is shown below. For a path formula Ψ we have

$$\langle\langle 1 \rangle\rangle(\Psi) = \{\ s \in S \mid \exists \sigma. \forall \tau. \pi(s, \sigma, \tau) \models \Psi\ \} = [\![\exists x. \forall y. \Psi(x, y)]\!] = [\![(\exists \Psi, \exists \mathbf{true}, \mathbf{true})]\!]$$
$$\langle\langle 1, 2 \rangle\rangle(\Psi) = \{\ s \in S \mid \exists \sigma. \exists \tau. \pi(s, \sigma, \tau) \models \Psi\ \}. = [\![\exists x. \exists y. \Psi(x, y)]\!] = [\![(\exists \mathbf{true}, \exists \mathbf{true}, \Psi)]\!].$$

The formulas $\langle\langle 2 \rangle\rangle(\Psi)$ and $\langle\langle \emptyset \rangle\rangle(\Psi)$ can be expressed similarly. Hence the logic ATL^* can be expressed in the simple one-alternation fragment of strategy logic,

and ATL can be expressed in the simple-one alternation fragment with unnested path formulas.

Nash equilibrium in simple one-alternation fragment. In nonzero-sum games the input is a game graph and two path formulas for the players. We start with the definition of payoff profile.

Payoff profile. Given a game graph G, path formulas Ψ_1 and Ψ_2, strategies σ and τ for the players, and a state s, the payoff for the players are defined as follows:

$$p_1(s,\sigma,\tau,\Psi_1) = \begin{cases} 1 & \text{if } \pi(s,\sigma,\tau) \models \Psi_1 \\ 0 & \text{otherwise;} \end{cases} \qquad p_2(s,\sigma,\tau,\Psi_2) = \begin{cases} 1 & \text{if } \pi(s,\sigma,\tau) \models \Psi_2 \\ 0 & \text{otherwise;} \end{cases}$$

A payoff profile consists of payoffs for player 1 and player 2.
We now define Nash equilibrium [11] and show they can be expressed in simple one-alternation fragment.

Nash equilibrium. Given a game graph G and path formulas Ψ_1 and Ψ_2, a strategy profile (σ^*, τ^*) is a *Nash equilibrium* at a state s iff the following conditions hold:

$$(1)\ \forall \sigma \in \Sigma.\ p_1(s,\sigma,\tau^*,\Psi_1) \leq p_1(s,\sigma^*,\tau^*,\Psi_1);$$

$$(2)\ \forall \tau \in \Gamma.\ p_2(s,\sigma^*,\tau,\Psi_2) \leq p_2(s,\sigma^*,\tau^*,\Psi_2).$$

We define the Nash equilibrium profile set of states as follows: for $i, j \in \{ 0, 1 \}$ we define

$$NE(i,j) = \{\ s \in S \mid \text{exists a Nash equilibrium } (\sigma^*,\tau^*) \text{ at } s \text{ such that } p_1(s,\sigma^*,\tau^*,\Psi_1) = i \text{ and } p_2(s,\sigma^*,\tau^*,\Psi_2) = j\ \}$$

Expressing Nash equilibrium. We now present how the Nash equilibrium states can be expressed in simple one-alternation fragment. The formulas are as follows:

$$\begin{aligned} NE(1,1) &= [\![(\exists \mathbf{true},\ \exists \mathbf{true},\ \Psi_1 \wedge \Psi_2)]\!]; \qquad NE(0,0) = [\![(\exists(\neg\Psi_2),\ \exists(\neg\Psi_1),\ \mathbf{true})]\!]; \\ NE(1,0) &= \{\ s \in S \mid \exists\sigma.\ (\exists\tau.\ \pi(s,\sigma,\tau) \models \Psi_1 \wedge \forall\tau'.\ \pi(s,\sigma,\tau') \models \neg\Psi_2)\ \} \\ &= [\![(\exists(\neg\Psi_2),\ \exists \mathbf{true},\ \Psi_1)]\!]; \\ NE(0,1) &= [\![(\exists \mathbf{true},\ \exists(\neg\Psi_1),\ \Psi_2)]\!]. \end{aligned}$$

Secure equilibrium in simple one-alternation fragment. A notion of conditional competitiveness in nonzero-sum games was formalized as the notion of secure equilibrium [5]. We show how secure equilibrium can be expressed in simple one-alternation fragment. We first define a lexico-graphic payoff profile ordering and then the notion of secure equilibrium.

Lexico-graphic ordering. We define two lexico-graphic ordering $\preceq_1$ and $\preceq_2$ of payoff profiles for players 1 and 2, respectively as follows: for payoff profiles (p_1, p_2) and (p_1', p_2') we have

$$(p_1,p_2) \preceq_1 (p_1',p_2') \text{ iff } (p_1 \leq p_1') \vee (p_1 = p_1' \wedge p_2 \geq p_2')$$

$$(p_1,p_2) \preceq_2 (p_1',p_2') \text{ iff } (p_2 \leq p_2') \vee (p_2 = p_2' \wedge p_1 \geq p_1')$$

Secure equilibrium. A *secure equilibrium* is a Nash equilibrium with respect to the ordering $\preceq_1$ and $\preceq_2$. Formally, given a game graph G and path formulas Ψ_1 and Ψ_2, a strategy profile (σ^*, τ^*) is a secure equilibrium at a state s iff the following conditions hold:

$$\forall \sigma \in \Sigma.\ (p_1(s, \sigma, \tau^*, \Psi_1), p_2(s, \sigma, \tau^*, \Psi_2)) \preceq_1 (p_1(s, \sigma^*, \tau^*, \Psi_1), p_2(s, \sigma^*, \tau^*, \Psi_2));$$

$$\forall \tau \in \Gamma.\ (p_1(s, \sigma^*, \tau, \Psi_1), p_2(s, \sigma^*, \tau, \Psi_2)) \preceq_2 (p_1(s, \sigma^*, \tau^*, \Psi_1), p_2(s, \sigma^*, \tau^*, \Psi_2)).$$

We define the secure equilibrium profile set of states as follows: for $i, j \in \{ 0, 1 \}$ we define

$$SE(i,j) = \{ s \in S \mid \text{exists a secure equilibrium } (\sigma^*, \tau^*) \text{ at } s \text{ such that } p_1(s, \sigma^*, \tau^*, \Psi_1) = i \text{ and } p_2(s, \sigma^*, \tau^*, \Psi_2) = j \}$$

The maximal secure equilibrium set of states are defined as follows: for $i, j \in \{ 0, 1 \}$ we define

$$MS(i,j) = \{s \in SE(i,j) \mid \text{if } s \in SE(i',j'), \text{ then } (i',j') \preceq_1 (i,j) \wedge (i',j') \preceq_2 (i,j)\}.$$

The following characterization of the maximal secure equilibrium states was established in [5]:

$$MS(1,0) = \{ s \in S \mid \mathsf{Win}_1(s, \Psi_1 \wedge \neg\Psi_2) \neq \emptyset \};$$
$$MS(0,1) = \{ s \in S \mid \mathsf{Win}_2(s, \Psi_2 \wedge \neg\Psi_1) \neq \emptyset \};$$
$$MS(1,1) = \{ s \in S \mid \exists \sigma \in \mathsf{Win}_1(s, \Psi_2 \rightarrow \Psi_1).\ \exists \tau \in \mathsf{Win}_2(s, \Psi_1 \rightarrow \Psi_2).\ \pi(s, \sigma, \tau) \models \Psi_1 \wedge \Psi_2 \};$$
$$MS(0,0) = S \setminus (MS(1,0) \cup MS(0,1) \cup MS(1,1)).$$

Expressing secure equilibrium. From the characterization of maximal secure equilibrium states, it follows that the maximal secure equilibrium states can be expressed in simple one-alternation fragment:

$$MS(1,0) = [(\exists(\Psi_1 \wedge \neg\Psi_2),\ \exists \mathbf{true},\ \mathbf{true})];$$
$$MS(0,1) = [(\exists \mathbf{true},\ \exists(\Psi_2 \wedge \neg\Psi_1),\ \mathbf{true})];$$
$$MS(1,1) = [(\exists(\Psi_2 \rightarrow \Psi_1),\ \exists(\Psi_1 \rightarrow \Psi_2),\ \Psi_1 \wedge \Psi_2)];$$

The set $MS(0,0)$ can be obtained by complementing the disjunction of the formulas to express $MS(1,1)$, $MS(0,1)$, and $MS(1,1)$.

Game logic and module checking. *Game logic* was introduced in [2] to express the module checking problem. The basic syntax is as follows: Game-logic state formulas are of the following forms: $\exists\{ 1 \}.\theta$ or $\exists\{ 2 \}.\theta$, where θ is a game-logic tree formula. Game-logic tree formulas are obtained as one of the following: (a) game-logic state formulas, (b) Boolean combination of game-logic tree formulas, and (c) $\exists\Psi$ or $\forall\Psi$, where Ψ is a path formula. The informal semantics is as follows: a formula $\exists\{ 1 \}.\theta$ is true if there is a strategy σ for player 1, such that the game-logic tree formula θ is satisfied in the tree generated fixing the strategy σ for player 1 (see [2] for details). Game logic can be expressed in the one-alternation fragment (but not in the simple one-alternation fragment)

of strategy logic. Thus, the module checking problem can be expressed in the one-alternation fragment. The following example illustrates how to translate a game logic formula into a one-alternation strategy logic formula:

$$[\exists\!\!\exists\{1\}.(\exists\Psi_1 \wedge \forall\Psi_2 \vee \forall\Psi_3)] = [\exists x.\ (\exists y_1.\ \Psi_1(x, y_1) \wedge \forall y_2.\Psi_2(x, y_2) \vee \forall y_3.\Psi_3(x, y_3)].$$

The following theorem compares the expressiveness of strategy logic and its fragment with ATL^*, game logic and alternating μ-calculus (proof in Appendix).

Theorem 1. *1. The expressiveness of simple one-alternation fragment and* ATL^* *coincide and one-alternation fragment is more expressive that* ATL^**.*
2. The one-alternation fragment is more expressive than game logic and game logic is more expressive than the simple-one alternation fragment.
3. Alternating μ-calculus is not as expressive as alternation free fragment of strategy logic. Strategy logic is not as expressive as alternating μ-calculus. Monadic Seconder Order (MSO) logic is more expressive than strategy logic.

6 Model Checking Strategy Logic

In this section we solve the model checking problem of strategy logic. We encode strategies by using strategy trees. We reason about strategy trees using tree automata, making our solution similar to Rabin's usage of tree automata for solving Monadic Second Order satisfiability problem [20]. We give the necessary definitions and proceed with the algorithm.

Trees and tree automata. Given a finite set Υ of directions, an *Υ-tree* is a set $T \subseteq \Upsilon^*$ such that if $x \cdot \upsilon \in T$, where $\upsilon \in \Upsilon$ and $x \in \Upsilon^*$, then also $x \in T$. The elements of T are called *nodes*, and the empty word ε is the *root* of T. For every $\upsilon \in \Upsilon$ and $x \in T$, the node x is the *parent* of $x \cdot \upsilon$. Each node $x \neq \varepsilon$ of T has a *direction* in Υ. The direction of the root is the symbol $\perp$ (we assume that $\perp \notin \Upsilon$). The direction of a node $x \cdot \upsilon$ is υ. We denote by $dir(x)$ the direction of node x. An Υ-tree T is a *full infinite tree* if $T = \Upsilon^*$. A *path* π of a tree T is a set $\pi \subseteq T$ such that $\varepsilon \in \pi$ and for every $x \in \pi$ there exists a unique $\upsilon \in \Upsilon$ such that $x \cdot \upsilon \in \pi$.

Given two finite sets Υ and Σ, a *Σ-labeled Υ-tree* is a pair $\langle T, \tau\rangle$ where T is an Υ-tree and $\tau : T \to \Sigma$ maps each node of T to a letter in Σ. When Υ and Σ are not important or clear from the context, we call $\langle T, \tau\rangle$ a labeled tree. We say that an $((\Upsilon \cup \{\perp\}) \times \Sigma)$-labeled Υ-tree $\langle T, \tau\rangle$ is *Υ-exhaustive* if for every node $x \in T$, we have $\tau(x) \in \{dir(x)\} \times \Sigma$.

Consider a game graph $G = ((S, E), (S_1, S_2))$. For $\alpha \in \{1, 2\}$, a strategy $\sigma : S^* \cdot S_\alpha \to S$ can be encoded by an S-labeled S-tree $\langle S^*, \tau\rangle$ by setting $\sigma(v) = \tau(v)$ for every $v \in S^* \cdot S_\alpha$. Notice that σ, may be encoded by many different trees. Indeed, for a node $v = s_0 \cdots s_n$ such that either $s_n \in S_{3-\alpha}$ or there exists some i such that $(s_i, s_{i+1}) \notin E$ the label $\tau(v)$ may be set arbitrarily. We may encode k different strategies by considering an S^k-labeled S-tree. Given a letter $\sigma \in S^k$ we denote by σ_i the projection of σ on its ith coordinate. In this case, the ith strategy is $\sigma_i(v) = \tau(v)_i$ for every $v \in S^* \cdot S_\alpha$. Notice that

the different strategies encoded may belong to different players. We refer to such trees as *strategy trees* and from now on may refer to a strategy as a tree $\langle S^*, \sigma \rangle$. In what follows we encode strategy assignments by strategy trees. We construct tree automata that accept the strategy assignments that satisfy a given strategy logic formula.

We use tree automata to reason about strategy trees. As we only use well known results about such automata we do not give a full formal definition, and refer the reader to [22]. Here, we use alternating parity tree automata (APT). The language of an automaton is the set of trees that it accepts. The size of automata is measured by the number of their states, and their index, which is a measure of the complexity of the acceptance condition. The important qualities of automata that are needed for this paper are summarized in Theorem 2 below.

Theorem 2. – *Given an LTL formula φ we can construct an APT $\mathcal{A}_\varphi$ with $2^{O(|\varphi|)}$ states and index 3 such that $\mathcal{A}_\varphi$ accepts all trees all of whose paths satisfy φ [23].*
– *Given APTs $\mathcal{A}_1$ and $\mathcal{A}_2$ with n_1 and n_2 states and indices k_1 and k_2 respectively, we can construct APTs for the conjunction and disjunction of $\mathcal{A}_1$ and $\mathcal{A}_2$ with $n_1 + n_2$ states and index $max(k_1, k_2)$. We can construct an alternating parity tree automaton for the complement language of $\mathcal{A}_1$ with n_1 states and index k_1 [17].*
– *Given an APT $\mathcal{A}$ with n states and index k, on alphabet $\Sigma \times \Sigma'$, we can construct an APT $\mathcal{A}'$ that accepts trees over alphabet Σ such that for some extension (or all extensions) of the labeling with labels from Σ' is accepted by $\mathcal{A}$. The number of states of $\mathcal{A}'$ is exponential in $n \cdot k$ and its index is linear in $n \cdot k$ [17].*
– *Given an APT $\mathcal{A}$ with n states and index k, we can check whether the language of $\mathcal{A}$ is empty or universal in time exponential in $n \cdot k$ [17, 8].*

Model checking algorithm. We present the model-checking algorithm for the strategy logic. The complexity of the algorithm depends on the number of quantifier alternations of a formula. We now formally define the *alternation depth* of a closed formula.

Alternation depth of variables. The *alternation-depth* of a variable of a closed quantified strategy formula is the number of quantifier switches ($\exists\forall$ or $\forall\exists$) that bind the variable (i.e., the variable is quantified). The alternation-depth of a closed formula is the maximum alternation-depth of a variable occurring in the formula.

Given a strategy logic formula φ, we construct by induction on the structure of the formula an NPT that accepts the set of strategy assignments that satisfy the formula. Wlog, assume that the variables in $X \cup Y$ are not reused. That is, in a closed formula there is a 1-1 and onto relation between the variables and the quantifiers.

Theorem 3. *Given a strategy logic formula φ and a game graph G, we can compute $[\![\varphi]\!]$ in time proportional to $(d+1)$-EXPTIME in the size of φ and d-EXPTIME in the size of G, where d is the alternation depth of φ. In case that*

φ is in the unnested path formula fragment, the complexity in the size of the formula reduces to d-EXPTIME.

Proof. The case where closed quantified strategy formula Λ is used as a state formula in a larger formula Λ', is solved by first computing the set of states satisfying Λ, adding this information to the game graph G, and then computing the set of states satisfying Λ'. In addition, if d is the alternation-depth of φ then φ is a Boolean combination of closed quantified strategy formulas of alternation depth at most d. Thus, it suffices to handle a simple closed quantified strategy logic formula, and reduce the Boolean reasoning to intersection, union, and complementation of the respective sets.

Consider a quantified strategy formula Λ. Let $Z = \{x_1, \ldots, x_n, y_1, \ldots, y_m\}$ be the set of variables used in Λ. Consider the alphabet S^{n+m} and an S^{n+m}-labeled S-tree σ. For a variable $v \in X \cup Y$, we denote by σ_v the strategy that stands in the location of variable v and for a set $Z' \subseteq Z$ we denote by $\sigma_{Z'}$ the set of strategies for the variables in Z'. We now describe how to construct an APT that accepts the set of strategy assignments that satisfy Λ. We build the APT by induction on the structure of the formula. For a subformula Λ'.

Case 1. $\Lambda' = \Psi(x, y)$ – by Theorem 2 we can construct an APT $\mathcal{A}$ that accepts trees all of whose paths satisfy Ψ. According to Theorem 2, $\mathcal{A}$ has $2^{O(|\Psi|)}$ states.

Case 2. $\Lambda' = \Lambda_1 \wedge \Lambda_2$ – given APTs $\mathcal{A}_1$ and $\mathcal{A}_2$ that accept the set of strategy assignments that satisfy Λ_1 and Λ_2, respectively; we construct an APT $\mathcal{A}$ for the conjunction of $\mathcal{A}_1$ and $\mathcal{A}_2$. According to Theorem 2, $|\mathcal{A}| = |\mathcal{A}_1| + |\mathcal{A}_2|$ and the index of $\mathcal{A}$ is the maximum of the indices of $\mathcal{A}_1$ and $\mathcal{A}_2$.

Case 3. $\Lambda' = \neg\Lambda_1$ – given an APT $\mathcal{A}_1$ that accepts the set of strategy assignments that satisfy Λ_1 we construct an APT $\mathcal{A}$ for the complement of $\mathcal{A}_1$. According to Theorem 2, $\mathcal{A}$ has the same number of states and same index as $\mathcal{A}_1$.

Case 4. $\Lambda' = \exists x.\Lambda_1$ – given an APT $\mathcal{A}_1$ that accepts the set of strategy assignments that satisfy Λ_1 we do the following. According to Theorem 2, we can construct an APT $\mathcal{A}'$ that accepts a tree iff there exists a way to extend the labeling of the tree with a labeling for the strategy for x such that the extended tree is accepted by $\mathcal{A}_1$. The number of states of $\mathcal{A}'$ is exponential in $n \cdot k$ and its index is linear in $n \cdot k$. The cases where $\Lambda' = \exists y.\Lambda_1$, $\Lambda' = \forall x.\Lambda_1$, and $\Lambda' = \exists y.\Lambda_1$ are handled similarly.

We note that for a closed Λ, the resulting automaton reads $S^{\emptyset}$-labeled S-trees. Thus, the input alphabet of the automaton has a single input letter and it only reads the structure of the S-tree.

The above construction starts with an automaton that is exponential in the size of a given LTL formula and incurs an additional exponent for every quantifier. In order to pay an exponent 'only' for every quantifier alternation, we have to use nondeterministic and universal automata, and maintain them in this form as long as possible. Nondeterministic automata are good for existential quantification, which comes to them for free, and universal automata are good for universal quantification. By careful analysis of the quantifier alternation hi-

erarchy, we can choose to create automata of the right kind (nondeterministic or universal), and maintain them in this form under disjunctions and conjunctions. Then, the complexity is $d+1$ exponents in the size of the formula and d exponents in the size of the game.

Consider the case where only unnested path formulas are used. Then, given a path formula $\Psi(x,y)$, we construct an APT $\mathcal{A}$ that accepts trees all of whose paths satisfy Ψ. As $\Psi(x,y)$ does not use nesting of temporal operators, we can construct $\mathcal{A}$ with a linear number of states in the size of Ψ.[1] It follows that the total complexity is d exponents in the size of the formula and d exponents in the size of the game. We note that in the case of unnested path formulas one exponent can be removed. The exact details are omitted due to lack of space. ∎

One-alternation fragment. Since ATL^* can be expressed in one-alternation fragment, it follows that model checking simple one-alternation fragment is 2EXPTIME-hard. Also since game logic can be expressed in one-alternation fragment, it follows that one-alternation fragment with unnested path formulas is EXPTIME-hard. These lower bounds along with Theorem 3 yield the following result.

Theorem 4. *Given a game graph G and an one-alternation strategy logic formula Φ, the following assertions hold.*

1. *The computation of $[\![\Phi]\!]$ is 2EXPTIME-complete, and if Φ consists only of unnested path formulas, then the computation of $[\![\Phi]\!]$ is EXPTIME-complete.*
2. *The program complexity (complexity of model checking formulas of bounded length) of one-alternation fragment of strategy logic is EXPTIME.*

Model checking simple one-alternation fragment. We now present a model checking algorithm for the simple one-alternation fragment, with better complexity than the general algorithm. We first present a few notations.

Notations. For a game graph G and a set $U \subseteq S$ of states we denote by $G \restriction U$ the game graph restricted to the set U and require that for all states $u \in U$ we have $E(u) \cap U \neq \emptyset$, i.e., all states in U have an edge in U. A path formula Ψ is *infinitary* if the set of paths that satisfy Ψ is independent of all finite-prefixes. The classical Büchi, coBüchi, parity, Rabin, Streett and Müller conditions are all infinitary conditions. All LTL objectives can be equivalently reduced to infinitary conditions like parity or Müller conditions. We now present the basic model checking result for infinitary path formulas.

Lemma 1. *Let G be a game graph and $\Phi = (\exists\, \Psi_1,\ \exists\, \Psi_2,\ \Psi_3)$ be a state formula with path formulas Ψ_1, Ψ_2 and Ψ_3 such that Ψ_1 and Ψ_2 are infinitary. Let $W_1 = \langle\langle 1 \rangle\rangle(\Psi_1)$ and $W_2 = \langle\langle 2 \rangle\rangle(\Psi_2)$. Then we have $[\![\Phi]\!] = \langle\langle 1,2 \rangle\rangle(\Psi_1 \wedge \Psi_2 \wedge \Psi_3)$ in $G \restriction W_1 \cap W_2$.*

[1] For a single temporal operator the number of states is constant, Boolean combinations between two automata may lead to an automaton whose size is the product of the sizes of the two automata. The number of multiplications is at most logarithmic in the size of the formula leading to a total linear number of states.

Lemma 2. *Let G be a game graph and $\Phi = (\exists\ \Psi_1,\ \exists\ \Psi_2,\ \Psi_3)$ be a state formula with unnested path formulas $\Psi_1 = \Phi_1^1 \mathcal{U} \Phi_1^2$, $\Psi_2 = \Phi_2^1 \mathcal{U} \Phi_2^2$ and Ψ_3. Let $W_1 = \langle\langle 1 \rangle\rangle(\Psi_1)$ and $W_2 = \langle\langle 2 \rangle\rangle(\Psi_2)$. Then we have $[\![\Phi]\!] = \langle\langle 1,2 \rangle\rangle(\Psi_1 \wedge \Psi_2 \wedge \Psi_3) \cap W_1 \cap W_2$.*

Theorem 5. *Let G be a game graph with n states and $\Phi = (\exists\ \Psi_1,\ \exists\ \Psi_2,\ \Psi_3)$ be a state formula with path formulas Ψ_1, Ψ_2 and Ψ_3. Then the following assertions hold.*

- *There is a ATL^* formula Φ' such that $[\![\Phi]\!] = [\![\Phi']\!]$.*
- *If Ψ_1, Ψ_2 and Ψ_3 are unnested path formulas, then there is a ATL^* formula Φ' with unnested path formulas such that $|\Phi'| = O(|\Psi_1| + |\Psi_2| + |\Psi_3|)$ and $[\![\Phi]\!] = [\![\Phi']\!]$.*
- *If Ψ_1, Ψ_2 and Ψ_3 are LTL path formulas, then given a state s deciding whether $s \in [\![\Phi]\!]$ is 2EXPTIME-complete, and $[\![\Phi]\!]$ can be computed in $n^{2^{O(|\Phi|)}} \cdot 2^{2^{O(|\Phi| \cdot \log |\Phi|)}}$ time.*
- *If Ψ_1, Ψ_2 and Ψ_3 are unnested path formulas, then $[\![\Phi]\!]$ can be computed in polynomial time.*
- *The program complexity of simple one-alternation fragment of strategy logic is polynomial time.*

Theorem 5 follows from Lemmas 1 and 2. We present details for part (3): given Ψ_1, Ψ_2 and Ψ_3 are parity conditions, from Lemma 1 it follows that $[\![(\exists \Psi_1,\ \exists \Psi_2, \Psi_3)]\!]$ can be computed by first solving two parity games, and then model checking a graph with conjunction of parity conditions (Streett conditions). Since a LTL formula Ψ can be converted to an equivalent deterministic parity automaton with $2^{2^{O(|\Psi| \cdot \log |\Psi|)}}$ states and $2^{O(|\Psi|)}$ parities (by converting Ψ to a nondeterministic Büchi automata and then determinizing it), applying algorithms for parity games [12] and polynomial time algorithm to model check Streett conditions we obtain the desired result. Observe that the model checking complexity of simple one-alternation fragment with unnested formulas, and the program complexity of simple one-alternation fragment is exponentially better than the one-alternation fragment.

References

1. M. Abadi, L. Lamport, and P. Wolper. Realizable and unrealizable concurrent program specifications. In *Proc. 16th International Colloquium on Automata, Languages and Programming*, volume 372, pages 1–17. Lecture Notes in Computer Science, Springer-Verlag, July 1989.
2. R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713, September 2002.
3. A. Blass, Y. Gurevich, L. Nachmanson, and M. Veanes. Play to test. In *FATES'05*, 2005.
4. K. Chatterjee and T.A. Henzinger. Assume guarantee synthesis. In *Proc. 30th International Conference on Tools and Algorithms for Construction and Analysis of Systems*, Lecture Notes in Computer Science. Springer-Verlag, 2007. to appear.
5. K. Chatterjee, T.A. Henzinger, and M. Jurdziński. Games with secure equilibria. In *Proc. 19th IEEE Symp. on Logic in Computer Science*, pages 160–169. IEEE, 2004.

6. L. de Alfaro and T.A. Henzinger. Interface automata. In *Proceedings of the Ninth Annual Symposium on Foundations of Software Engineering*, pages 109–120. ACM press, 2001.
7. D.L. Dill. *Trace theory for automatic hierarchical verification of speed independent circuits*. MIT Press, 1989.
8. E.A. Emerson, C. Jutla, and A.P. Sistla. On model-checking for fragments of μ-calculus. In *Proc. 5th International Conference on Computer Aided Verification*, volume 697 of *Lecture Notes in Comptuer Science*, pages 385–396, Elounda, Crete, June 1993. Springer-Verlag.
9. D. Gabbay, A. Pnueli, S. Shelah, and J. Stavi. On the temporal analysis of fairness. In *Proc. 7th ACM Symp. on Principles of Programming Languages*, pages 163–173, January 1980.
10. T.A. Henzinger, O. Kupferman, and S. Rajamani. Fair simulation. *Information and Computation*, 173(1):64–81, 2002.
11. J.F. Nash Jr. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences USA*, 36:48–49, 1950.
12. M. Jurdziński. Small progress measures for solving parity games. In *17th Annual Symposium on Theoretical Aspects of Computer Science*, volume 1770 of *Lecture Notes in Computer Science*, pages 290–301. Springer-Verlag, 2000.
13. J.A.W. Kamp. *Tense Logic and the Theory of Order*. PhD thesis, UCLA, 1968.
14. O. Kupferman, M.Y. Vardi, and P. Wolper. Module checking. *Information and Computation*, 164:322–344, 2001.
15. D.A. Martin. Borel determinacy. *Annals of Mathematics*, 65:363–371, 1975.
16. R. Milner. An algebraic definition of simulation between programs. In *Proc. 2nd International Joint Conference on Artificial Intelligence*, pages 481–489. British Computer Society, September 1971.
17. D.E. Muller and P.E. Schupp. Alternating automata on infinite trees. *Theoretical Computer Science*, 54:267–276, 1987.
18. G. Owen. *Game Theory*. Academic Press, 1995.
19. A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. 16th ACM Symp. on Principles of Programming Languages*, pages 179–190, Austin, January 1989.
20. M.O. Rabin. Decidability of second order theories and automata on infinite trees. *Transaction of the AMS*, 141:1–35, 1969.
21. P.J.G. Ramadge and W.M. Wonham. The control of discrete event systems. *IEEE Transactions on Control Theory*, 77:81–98, 1989.
22. W. Thomas. On the synthesis of strategies in infinite games. In E.W. Mayr and C. Puech, editors, *Proc. 12th Symp. on Theoretical Aspects of Computer Science*, volume 900 of *Lecture Notes in Computer Science*, pages 1–13. Springer-Verlag, 1995.
23. M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, November 1994.

7 Appendix

We first present the proof of Theorem 1.

Proof. (**Theorem 1.**) We prove all the cases below.

1. We first show that ATL^* can be expressed in simple one-alternation fragment. For a path formula Ψ we have

$$\langle\langle 1\rangle\rangle(\Psi) = \{\ s \in S \mid \exists\sigma.\forall\tau.\pi(s,\sigma,\tau) \models \Psi\ \} = [\![\exists x.\forall y.\Psi(x,y)]\!] = [\![(\exists\Psi, \exists\mathbf{true}, \mathbf{true})]\!]$$
$$\langle\langle 2\rangle\rangle(\Psi) = \{\ s \in S \mid \exists\tau.\forall\sigma.\pi(s,\sigma,\tau) \models \Psi\ \} = [\![\exists y.\forall x.\Psi(x,y)]\!] = [\![(\exists\mathbf{true}, \exists\Psi, \mathbf{true})]\!];$$
$$\langle\langle 1,2\rangle\rangle(\Psi) = \{\ s \in S \mid \exists\sigma.\exists\tau.\pi(s,\sigma,\tau) \models \Psi\ \}. = [\![\exists x.\exists y.\Psi(x,y)]\!] = [\![(\exists\mathbf{true}, \exists\mathbf{true}, \Psi)]\!];$$
$$\langle\langle\emptyset\rangle\rangle(\Psi) = \{\ s \in S \mid \forall\sigma.\forall\tau.\pi(s,\sigma,\tau) \models \Psi\ \}. = [\![\forall x.\forall y.\Psi(x,y)]\!] = [\![(\forall\mathbf{true}, \forall\mathbf{true}, \Psi)]\!].$$

   Since the simple one-alternation fragment allows closed formulas to be treated as atomic propositions, it follows that the logic ATL^* can be expressed in the simple one-alternation fragment of strategy logic, and ATL can be expressed in the simple-one alternation fragment with unnested path formulas. It follows from Theorem 5 (part 1) that the expressiveness of ATL^* and simple one-alternation fragment coincide. Since game logic is more expressive than ATL^* [2] and game logic can be expressed in the one-alternation fragment it follows that one-alternation fragment is more expressive than ATL^*.
2. The game logic can be expressed in $\exists\forall$ fragment of the strategy logic. By part (1) it follows that the expressive power of simple one-alternation fragment coincides with ATL^*, and the result of [2] shows that game logic is more expressive that ATL^*. It follows that game logic is more expressive than the simple one-alternation fragment.
3. It follows from the results of [2] that the following formula

$$\exists x.\ (\exists y_1.\ (\Box p)(x, y_1) \wedge \exists y_2.\ (\Box q)(x, y_2))$$

   cannot be expressed in alternating μ-calculus. The above formula is an alternation free strategy logic formula. We now present alternating μ-calculus formulas that are not expressible in strategy logic. Consider one-player structures (i.e., $S_2 = \emptyset$). The following formula

$$\nu x.\ [p \wedge AX(AX(x))]$$

   specifies the set of states s such that in all paths from s every even position is labeled by the proposition p. Such counting properties cannot be expressed in strategy logic. Also consider the following formula over one-player structures:

$$\mu x.(q \vee (p \wedge EX(x)) \vee (\neg p \wedge AX(x)))$$

   The formula says that the proposition p turns the one-player game into a two player game such that the p player has a strategy to reach q. This is also not expressible by strategy logic on one-player structures. We now argue that

MSO is more expressive than strategy logic: encoding strategies as trees, a strategy logic formula can be translated to a MSO formula. Hence MSO is as expressive as strategy logic. Since MSO contains alternating μ-calculus and strategy logic is not as expressive as alternating μ-calculus, it follows that MSO is more expressive than strategy logic. ∎

We will now present proofs of Lemmas 1 and 2. We start with a notation.

Notations. For a game graph G and a set $U \subseteq S$ of states we denote by $G \restriction U$ the game graph restricted to the set U and require that for all states in $u \in U$ we have $E(u) \cap U \neq \emptyset$, i.e., all states in U have an edge in U. For a set $U \subseteq S$ of states we denote by $\mathsf{Safe}(U) = \{ \pi = \langle s_1, s_2, \ldots \rangle \mid \forall k \geq 1.\ s_k \in U \}$ the set of paths that always visits states in U.

Proof. (**of Lemma 1.**) We first observe that $[\![\Phi]\!] \subseteq W_1 \cap W_2$ as follows:

$$\begin{aligned}[\![\Phi]\!] &= \{ s \in S \mid \exists \sigma \in \mathsf{Win}_1(s, \Psi_1).\ \exists \tau \in \mathsf{Win}_2(s, \Psi_2).\ \pi(s, \sigma, \tau) \models \Psi_3 \} \\ &\subseteq \{ s \in S \mid \mathsf{Win}_1(s, \Psi_1) \neq \emptyset \} \cap \{ s \in S \mid \mathsf{Win}_2(s, \Psi_2) \neq \emptyset \} \\ &= W_1 \cap W_2.\end{aligned}$$

We now show that $G \restriction W_1 \cap W_2$ is a game graph. Since Ψ_1 is infinitary, for a player 1 state $s \in S_1 \cap W_1$, we have $E(s) \cap W_1 \neq \emptyset$ and for a player 2 state $s \in S_2 \cap W_1$, we have $E(s) \subseteq W_1$. Similarly, for a player 1 state $s \in S_1 \cap W_2$, we have $E(s) \subseteq W_2$ and for a for a player 2 state $s \in S_2 \cap W_2$, we have $E(s) \cap W_2 \neq \emptyset$. It follows that $G \restriction W_1 \cap W_2$ is a game graph. For any winning strategy pair (σ, τ) for all states $s \in W_1 \cap W_2$ we have $\pi(s, \sigma, \tau) \in \mathsf{Safe}(W_1 \cap W_2)$. Let $U = [\![\Phi]\!]$ and we prove that $U = \langle\!\langle 1, 2 \rangle\!\rangle (\Psi_1 \wedge \Psi_2 \wedge \Psi_3)$ in $G \restriction (W_1 \cap W_2)$ by proving inclusion in both directions.

1. We already showed that $U \subseteq W_1 \cap W_2$. We first argue that $U \subseteq \langle\!\langle 1, 2 \rangle\!\rangle (\Psi_1 \wedge \Psi_2 \wedge \Psi_3)$ in $G \restriction (W_1 \cap W_2)$. For a state s in U, fix a witness strategy pair (σ, τ) such that $\sigma \in \mathsf{Win}_1(s, \Psi_1)$, $\tau \in \mathsf{Win}_2(s, \Psi_2)$ and $\pi(s, \sigma, \tau) \models \Psi_3$. We have $\pi(s, \sigma, \tau) \in \mathsf{Safe}(W_1 \cap W_2)$, and since $\sigma \in \mathsf{Win}_1(s, \Psi_1)$ and $\tau \in \mathsf{Win}_2(s, \Psi_2)$ we have $\pi(s, \sigma, \tau) \models \Psi_1 \wedge \Psi_2 \wedge \Psi_3$. Hence (σ, τ) is a witness to show that

$$s \in \{ s_1 \in S \cap (W_1 \cap W_2) \mid \exists \sigma.\ \exists \tau.\ \pi(s_1, \sigma, \tau) \models \Psi_1 \wedge \Psi_2 \wedge \Psi_3 \},$$

   i.e., $s \in \langle\!\langle 1, 2 \rangle\!\rangle (\Psi_1 \wedge \Psi_2 \wedge \Psi_3)$ in $G \restriction (W_1 \cap W_2)$.
2. We now prove the other inclusion to complete the proof. Let $s \in \langle\!\langle 1, 2 \rangle\!\rangle (\Psi_1 \wedge \Psi_2 \wedge \Psi_3)$ in $G \restriction (W_1 \cap W_2)$. Fix a witness strategy pair (σ_1, τ_1) in $G \restriction (W_1 \cap W_2)$ such that $\pi(s, \sigma_1, \tau_1) \models \Psi_1 \wedge \Psi_2 \wedge \Psi_3$. We construct witness strategies to show that $s \in U$ as follows:
   - *Player 1 strategy σ^*.* Player 1 plays the strategy σ_1 as long as player 2 follows τ_1; if player 2 deviates at state s_1, then player 1 switches to a strategy $\widehat{\sigma} \in \mathsf{Win}_1(s_1, \Psi_1)$. Observe that any player 2 deviation still keeps the game in W_1 since for all states in $s_1 \in W_1 \cap S_2$ we have $E(s_1) \subseteq W_1$ and hence the construction is valid.

– *Player 2 strategy τ^*.* Player 2 plays the strategy τ_1 as long as player 1 follows σ_1; if player 1 deviates at state s_1, then player 2 switches to a strategy $\widehat{\tau} \in \mathsf{Win}_1(s_1, \Psi_2)$. Observe that any player 1 deviation still keeps the game in W_2 since for all states in $s_1 \in W_2 \cap S_1$ we have $E(s_1) \subseteq W_2$ and hence the construction is valid.

Since $\pi(s, \sigma_1, \tau_1) \models \Psi_1 \wedge \Psi_2 \wedge \Psi_3$ (hence also $\pi(s, \sigma_1, \tau_1) \models \Psi_1 \wedge \Psi_2$), and the players switch to a respective winning strategy if the other player deviates, it follows that $\sigma^* \in \mathsf{Win}_1(s, \Psi_1)$ and $\tau^* \in \mathsf{Win}_2(s, \Psi_2)$. Moreover, we have $\pi(s, \sigma_1, \tau_1) = \pi(s, \sigma^*, \tau^*) \models \Psi_1 \wedge \Psi_2 \wedge \Psi_3$. Hence we have $s \in U$.

The result follows and it also follows that $[\![\Phi]\!] = \langle\langle 1, 2 \rangle\rangle(\Psi_1 \wedge \Psi_2 \wedge \Psi_3 \wedge \mathsf{Safe}(\langle\langle 1 \rangle\rangle(\Psi_1) \wedge \langle\langle 2 \rangle\rangle(\Psi_2)))$. ∎

Proof. (**of Lemma 2.**) Similar to the proof for Lemma 1 we have $[\![\Phi]\!] \subseteq W_1 \cap W_2$. Let $U = [\![\Phi]\!]$ and we prove that $U = \langle\langle 1, 2 \rangle\rangle(\Psi_1 \wedge \Psi_2 \wedge \Psi_3) \cap W_1 \cap W_2$ by proving inclusion in both directions.

1. We already argued that $U \subseteq W_1 \cap W_2$. We first argue that $U \subseteq \langle\langle 1, 2 \rangle\rangle(\Psi_1 \wedge \Psi_2 \wedge \Psi_3) \cap W_1 \cap W_2$. For a state s in U, fix a witness strategy pair (σ, τ) such that $\sigma \in \mathsf{Win}_1(s, \Psi_1)$, $\tau \in \mathsf{Win}_2(s, \Psi_2)$ and $\pi(s, \sigma, \tau) \models \Psi_3$. Since $\sigma \in \mathsf{Win}_1(s, \Psi_1)$ and $\tau \in \mathsf{Win}_2(s, \Psi_2)$ we have $\pi(s, \sigma, \tau) \models \Psi_1 \wedge \Psi_2 \wedge \Psi_3$. Hence (σ, τ) is a witness to show that
$$s \in \{\, s_1 \in S \cap (W_1 \cap W_2) \mid \exists \sigma.\ \exists \tau.\ \pi(s_1, \sigma, \tau) \models \Psi_1 \wedge \Psi_2 \wedge \Psi_3 \,\},$$
i.e., $s \in \langle\langle 1, 2 \rangle\rangle(\Psi_1 \wedge \Psi_2 \wedge \Psi_3) \cap W_1 \cap W_2$.
2. We now prove the other inclusion to complete the proof. Let $s \in \langle\langle 1, 2 \rangle\rangle(\Psi_1 \wedge \Psi_2 \wedge \Psi_3) \cap W_1 \cap W_2$. Fix a witness strategy pair (σ_1, τ_1) in $G \restriction (W_1 \cap W_2)$ such that $\pi(s, \sigma_1, \tau_1) \models \Psi_1 \wedge \Psi_2 \wedge \Psi_3$. We construct witness strategies to show that $s \in U$ as follows:
   – *Player 1 strategy σ^*.* We first observe that for all $s \in S_2 \cap (W_1 \setminus [\![\Phi_1^2]\!])$ we have $E(s) \subseteq W_1$. Player 1 plays the strategy σ_1 as long as player 2 follows τ_1; if player 2 deviates at state s_1, then either $s_1 \in [\![\Phi_1^2]\!]$ (in which case Ψ_1 is satisfied) or else player 1 switches to a strategy $\widehat{\sigma} \in \mathsf{Win}_1(s_1, \Psi_1)$. Observe that any player 2 deviation from states other than $[\![\Phi_1^2]\!]$ still keeps the game in W_1, and hence the construction is valid.
   – *Player 2 strategy τ^*.* We again observe that for all $s \in S_1 \cap (W_2 \setminus [\![\Phi_2^2]\!])$ we have $E(s) \subseteq W_2$. Player 2 plays the strategy τ_1 as long as player 1 follows σ_1; if player 1 deviates at state s_1, then either $s_1 \in [\![\Phi_2^2]\!]$ (in which case Ψ_2 is satisfied) or else player 1 switches to a strategy $\widehat{\tau} \in \mathsf{Win}_2(s_1, \Psi_2)$. Observe that any player 1 deviation from states other than $[\![\Phi_2^2]\!]$ still keeps the game in W_2, and hence the construction is valid.

   Since $\pi(s, \sigma_1, \tau_1) \models \Psi_1 \wedge \Psi_2 \wedge \Psi_3$ (hence also $\pi(s, \sigma_1, \tau_1) \models \Psi_1 \wedge \Psi_2$), and the players switch to a respective winning strategy if the other player deviates, it follows that $\sigma^* \in \mathsf{Win}_1(s, \Psi_1)$ and $\tau^* \in \mathsf{Win}_2(s, \Psi_2)$. Moreover, we have $\pi(s, \sigma_1, \tau_1) = \pi(s, \sigma^*, \tau^*) \models \Psi_1 \wedge \Psi_2 \wedge \Psi_3$. Hence we have $s \in U$.

The result follows and it also follows that $[\![\Phi]\!] = \langle\langle 1, 2 \rangle\rangle(\Psi_1 \wedge \Psi_2 \wedge \Psi_3) \wedge \langle\langle 1 \rangle\rangle(\Psi_1) \wedge \langle\langle 2 \rangle\rangle(\Psi_2)$. ∎