Strategy Logic

Krishnendu Chatterjee[a] Thomas A. Henzinger[a,b] Nir Piterman[c,*]

[a] *EECS, University of California, Berkeley, USA*

[b] *Computer and Communication Sciences, EPFL, Switzerland*

[c] *Imperial College London, UK*

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 specify properties of nonzero-sum games in a simple and natural way. We show that the one-alternation fragment of strategy logic is strong enough to express the existence of Nash equilibria and secure equilibria, and subsumes 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 nonelementary, for the simple fragment that is used above we show that the complexity is polynomial in the size of the game graph and optimal in the size of the formula (ranging from polynomial to 2EXPTIME depending on the form of the formula).

Key words: Game theory, Logic, ATL and ATL*, Non-zero-sum games, Automata theory.

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 [PR89,RW89], as well as for defining and checking the realizability of specifications [ALW89,Dil89], the compatibility of interfaces [dAH01], simulation relations between transition systems [HKR02,Mil71], and for generating test cases [BGNV05], to name just a few of their applications. The study of nonzero-sum graph games has been more recent, with assume-guarantee synthesis [CH07] 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 [Mar75] 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 [AHK02], which replaces the unconstrained path quantifiers

* Corresponding Author

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 [AHK02], among them the temporal logic ATL^*, the alternating-time μ-calculus, and a so-called *game logic* of [AHK02]: these are motivated by expressing general ω-regular winning conditions, as well as tree properties of computation trees that result from fixing the strategy of one player (module checking [KVW01]). 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. It would follow from our results that Nash equilibria can be expressed in ATL^*.

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* [CHJ04] 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 $\Psi_2 \Rightarrow \Psi_1$ (where $\Rightarrow$ denotes implication). The symmetric condition holds for player 2; in summary: $(\exists x \in \Sigma)(\exists y \in \Gamma)[(\Psi_1 \wedge \Psi_2)(x, y) \wedge (\forall y' \in \Gamma)(\Psi_2 \Rightarrow \Psi_1)(x, y') \wedge (\forall x' \in \Sigma)(\Psi_1 \Rightarrow \Psi_2)(x', y)]$. 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 $\Psi_2 \Rightarrow \Psi_1$; 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 of 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 [GPSS80,Kam68].

In this work we consider perfect-information games and, consequently, only pure strategies (no probabilistic choice). An extension of this work to the setting of partial-information games is an interesting research direction (cf. [Kai06]). Other possible extensions include reasoning about concurrent games and about perfect-information games with probabilistic transitions, as well as increasing the expressive power of the logic by allowing more ways to bound strategies (e.g., comparing strategies).

2. Graph Games

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

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 $k \geq 0$, we have $(s_k, s_{k+1}) \in E$. We write 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 $\sigma\colon S^* \cdot S_1 \to S$ that given a finite sequence of states, which represents the history of the play so far, and which 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 $\sigma(w \cdot s) \in E(s)$. The strategies for player 2 are defined symmetrically. We denote by Σ and Γ the sets of all strategies for player 1 and player 2, respectively. Given a starting state $s \in S$, a strategy σ for player 1, and a strategy τ for player 2, there is a unique play, denoted as $\pi(s, \sigma, \tau) = \langle s_0, s_1, s_2, \ldots\rangle$, which is defined as follows: $s = s_0$, and for all $k \geq 0$, we have (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

Strategy logic is interpreted over labeled game graphs. Let P be a finite set of atomic propositions. A *labeled game graph* $\mathcal{G} = (G, P, L)$ consists of a game graph G together with a labeling function $L\colon S \to 2^P$ that maps every state s to the set $L(s)$ of atomic propositions that are true at s. We assume that there is a special atomic proposition $\mathbf{tt} \in P$ such that $\mathbf{tt} \in L(s)$ for all $s \in S$.

Syntax. The formulas of strategy logic consist of the following kinds of subformulas. Path formulas Ψ are LTL formulas, which are interpreted over infinite paths of states. Atomic strategy formulas are path formulas $\Psi(x, y)$ with two arguments —a variable x that denotes a player-1 strategy, and a variable y that denotes a player-2 strategy. From atomic strategy formulas, we define a first-order logic of quantified strategy formulas. The formulas of strategy logic are the closed strategy formulas (i.e., strategy formulas without free strategy variables); they are interpreted over states. We denote path and strategy formulas by Ψ and Φ, respectively. We use the variables $x, x_1, x_2, \ldots$ to range over strategies for player 1, and denote the set of such variables by X; similarly, the variables $y, y_1, y_2, \ldots \in Y$ range over strategies for player 2. Formally, the path and strategy formulas are defined by the following grammar:

$$\Psi ::= p \mid \Phi \mid \Psi \wedge \Psi \mid \neg\Psi \mid \bigcirc\Psi \mid \Psi \;\mathcal{U}\; \Psi, \text{ where } p \in P \text{ and } \Phi \text{ is closed;}$$

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

Observe that the closed strategy formulas can be reused as atomic propositions. We formally define the free variables of strategy formulas as follows:
$\mathsf{Free}(\Psi(x, y)) = \{x, y\}$;
$\mathsf{Free}(\Phi_1 \wedge \Phi_2) = \mathsf{Free}(\Phi_1) \cup \mathsf{Free}(\Phi_2)$;

$\mathsf{Free}(\Phi_1 \vee \Phi_2) = \mathsf{Free}(\Phi_1) \cup \mathsf{Free}(\Phi_2)$;
$\mathsf{Free}(Qx.\Phi') = \mathsf{Free}(\Phi') \setminus \{x\}$, for $Q \in \{\exists, \forall\}$;
$\mathsf{Free}(Qy.\Phi') = \mathsf{Free}(\Phi') \setminus \{y\}$, for $Q \in \{\exists, \forall\}$.
A strategy formula Φ is *closed* if $\mathsf{Free}(\Phi) = \emptyset$. We define additional boolean connectives such as $\Rightarrow$, and additional temporal operators such as $\Box$ and $\Diamond$, as usual. Note that in order to use the connective $\Rightarrow$ we may have to use the duality $\neg\exists z.\Psi$ is equivalent to $\forall z.\neg\Psi$. We do not introduce formally the negation operator for strategy formulas, its treatment is standard.

Semantics. For a set $Z \subseteq X \cup Y$ of variables, a *strategy assignment* A_Z assigns to every variable $x \in Z \cap X$, a player-1 strategy $A_Z(x) \in \Sigma$, and to every variable $y \in Z \cap Y$, a player-2 strategy $A_Z(y) \in \Gamma$. Given a strategy assignment A_Z and player-1 strategy $\sigma \in \Sigma$, we denote by $A_Z[x \leftarrow \sigma]$ the extension of the assignment A_Z to the set $Z \cup \{x\}$, defined as follows: for $w \in Z \cup \{x\}$, we have $A_Z[x \leftarrow \sigma](w) = A_Z(w)$ if $w \neq x$, and $A_Z[x \leftarrow \sigma](x) = \sigma$. The definition of $A_Z[y \leftarrow \tau]$ for player-2 strategies $\tau \in \Gamma$ is analogous.

The semantics of path formulas Ψ is the usual semantics of LTL. We now describe the satisfaction of a strategy formula Φ at a state $s \in S$ with respect to a strategy assignment A_Z, where $\mathsf{Free}(\Phi) \subseteq Z$:

$$\begin{aligned}
(s, A_Z) &\models \Psi(x,y) && \text{iff} \quad \pi(s, A_Z(x), A_Z(y)) \models \Psi;\\
(s, A_Z) &\models \Phi_1 \wedge \Phi_2 && \text{iff} \quad (s, A_Z) \models \Phi_1 \text{ and } (s, A_Z) \models \Phi_2;\\
(s, A_Z) &\models \Phi_1 \vee \Phi_2 && \text{iff} \quad (s, A_Z) \models \Phi_1 \text{ or } (s, A_Z) \models \Phi_2;\\
(s, A_Z) &\models \exists x.\Phi' && \text{iff} \quad \exists \sigma \in \Sigma.\ (s, A_Z[x \leftarrow \sigma]) \models \Phi';\\
(s, A_Z) &\models \forall x.\Phi' && \text{iff} \quad \forall \sigma \in \Sigma.\ (s, A_Z[x \leftarrow \sigma]) \models \Phi';\\
(s, A_Z) &\models \exists y.\Phi' && \text{iff} \quad \exists \tau \in \Gamma.\ (s, A_Z[y \leftarrow \tau]) \models \Phi';\\
(s, A_Z) &\models \forall y.\Phi' && \text{iff} \quad \forall \tau \in \Gamma.\ (s, A_Z[y \leftarrow \tau]) \models \Phi'.
\end{aligned}$$

The semantics of a closed strategy formula Φ is the set $[\![\Phi]\!] = \{s \in S \mid (s, A_\emptyset) \models \Phi\}$ of states.

Unnested path formulas. Of special interest is the fragment of strategy logic where path formulas do not allow any nesting of temporal operators. This fragment has a CTL-like flavor, and as we show later, results in a decision procedure with a lower computational complexity. Formally, the *unnested* path formulas are restricted as follows:

$$\Psi ::= p \mid \Phi \mid \Psi \wedge \Psi \mid \neg\Psi \mid \bigcirc\Phi \mid \Phi\ \mathcal{U}\ \Phi, \text{ where } p \in P \text{ and } \Phi \text{ is closed.}$$

The resulting closed strategy formulas are called the *unnested-path-formula fragment* of strategy logic.

Examples. We now present some examples of formulas of strategy logic. We first show how to express formulas of the logics ATL and ATL^* [AHK02] in strategy logic. The alternating-time temporal logic ATL^* consists of path formulas quantified by the alternating path operators $\langle\!\langle 1 \rangle\!\rangle$ and $\langle\!\langle 2 \rangle\!\rangle$, the existential path operator $\langle\!\langle 1,2 \rangle\!\rangle$ (or $\exists$), and the universal path operator $\langle\!\langle \emptyset \rangle\!\rangle$ (or $\forall$). The logic ATL is the subclass of ATL^* where only unnested path formulas are considered. Some examples of ATL and ATL^* formulas and the equivalent strategy formulas are as follows: for a proposition $p \in P$,

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

Consider the strategy formula $\Phi = \exists x.\ (\exists y_1.\ (\Box p)(x, y_1)\ \wedge\ \exists y_2.\ (\Box q)(x, y_2))$. This formula is different from the following two formulas:

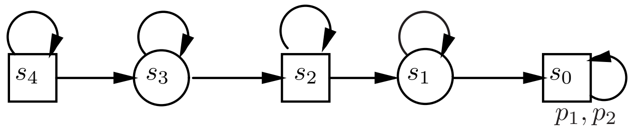

Fig. 1. A two-player game graph.

- the formula $\langle\langle 1,2\rangle\rangle(\Box p) \wedge \langle\langle 1,2\rangle\rangle(\Box q)$ is too weak, i.e., for all game graphs we have $[\![\Phi]\!] \subseteq [\![\langle\langle 1,2\rangle\rangle(\Box p) \wedge \langle\langle 1,2\rangle\rangle(\Box q)]\!]$ and there exists a game graph such that the inclusion is strict; and
- the formula $\langle\langle 1,2\rangle\rangle(\Box(p \wedge q))$ is too strong, i.e., for all game graphs we have $[\![\langle\langle 1,2\rangle\rangle(\Box p \wedge \Box q)]\!] \subseteq [\![\Phi]\!]$ and there exists game graph such that the inclusion is strict.

It was shown in [AHK02] (section 6.2 of [AHK02]) that the formula Φ cannot be expressed in ATL^*.

One of the features of strategy logic is that we can restrict the kinds of strategies that interest us. For example, the following strategy 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:

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

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 analogue of the above formula, where the level of knowledge is nested up to level k, can be expressed in strategy logic. For example, the formula above ("knowledge nesting 1") is different from the following formula with "knowledge nesting 2":

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

In Example 1 we show that the formulas with different knowledge nesting identify different parts of game graphs. We do not know whether the corresponding fixpoint of 'full knowledge nesting' can be expressed in strategy logic.

Example 1 (Knowledge nesting) *Consider the game graph shown in Fig 1. The $\Box$ states are player-1 states, and $\bigcirc$ states are player-2 states. The state s_0 is labeled with propositions p_1 and p_2. The objectives for the players are to reach the proposition p_1 and p_2, respectively, (i.e., objective for player 1 is $\Diamond p_1$ and objective for player 2 is $\Diamond p_2$). With no knowledge nesting player 1 cannot satisfy her objective at state s_2, since player 2 can always choose the edge $s_1 \to s_1$. But given player 2 plays a winning strategy for $\Diamond p_2$, player 1 can satisfy $\Diamond p_1$ choosing the strategy $s_2 \to s_1$. Similarly, without knowing that player 1 is playing the strategy $s_2 \to s_1$, player 2 cannot satisfy $\Diamond p_2$ at s_3. However, with "knowledge nesting 2" player 2 can satisfy $\Diamond p_2$ at s_3 and with "knowledge nesting 3" player 1 can satisfy $\Diamond p_1$ at s_4.* ∎

As another example, we consider the notion of dominating and dominated strategies [Owe95]. Given a path formula Ψ and a state $s \in S$, a strategy x_1 for player 1 *dominates* another player-1 strategy x_2 if for all player-2 strategies y, whenever $\pi(s, x_2, y) \models \Psi$, then $\pi(s, x_1, y) \models \Psi$. The strategy x_1 is *dominating* if it dominates every player-1 strategy x_2. The following strategy formula expresses that x_1 is a dominating strategy:

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

Given a path formula Ψ and a state $s \in S$, a strategy x_1 for player 1 is *dominated* if there is a player-1 strategy x_2 such that (a) for all player-2 strategies y_1, if $\pi(s, x_1, y_1) \models \Psi$, then $\pi(s, x_2, y_1) \models \Psi$, and (b) for some player-2 strategy y_2, we have both $\pi(s, x_2, y_2) \models \Psi$ and $\pi(s, x_1, y_2) \not\models \Psi$. The following strategy formula expresses that x_1 is a dominated strategy:

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

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

4. Simple One-Alternation Fragment of Strategy Logic

In this section we define a subset of strategy 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 strategy quantifiers. We refer to this subset as the simple one-alternation fragment. We show later how several important concepts in nonzero-sum games can be captured in this fragment.

Syntax. We are interested in strategy formulas that depend on three path formulas: Ψ_1, Ψ_2, and Ψ_3. The strategy formulas in the simple one-alternation fragment assert that there exist player-1 and player-2 strategies that ensure Ψ_1 and Ψ_2, respectively, and at the same time cooperate to satisfy Ψ_3. Formally, the *simple one-alternation* strategy formulas are restricted as follows:

$$\Phi ::= \Phi \wedge \Phi \mid \neg\Phi \mid \exists x_1.\ \exists y_1.\ \forall x_2.\ \forall y_2.\ (\Psi_1(x_1, y_2) \wedge \Psi_2(x_2, y_1) \wedge \Psi_3(x_1, y_1)),$$

where $x_1, x_2 \in X$, and $y_1, y_2 \in Y$. The resulting closed strategy formulas are called the *simple one-alternation fragment* of strategy logic. Obviously, the formulas have a single quantifier alternation. We use the abbreviation $(\exists\ \Psi_1,\ \exists\ \Psi_2,\ \Psi_3)$ for simple one-alternation strategy formulas of the form $\exists x_1.\exists y_1.\forall x_2.\forall y_2.\ (\Psi_1(x_1, y_2) \wedge \Psi_2(x_2, y_1) \wedge \Psi_3(x_1, y_1))$. We will show that ATL^* as well as important concepts like Nash equilibria and secure equilibria of nonzero-sum games can be expressed in the simple one-alternation fragment. The simple one-alternation fragment is a proper sub-class of the one alternation fragment and we will present more efficient model checking algorithms for this fragment as compared to the general one-alternation fragment.

Notation. For a path formula Ψ and a state s we define the set $\mathsf{Win}_1(s, \Psi) = \{\sigma \in \Sigma \mid \forall \tau \in \Gamma.\ \pi(s, \sigma, \tau) \models \Psi\}$ to denote the set of player-1 strategies that enforce Ψ against all player-2 strategies. We refer to the strategies in $\mathsf{Win}_1(s, \Psi)$ as the *winning* player-1 strategies for Ψ from s. Analogously, we define $\mathsf{Win}_2(s, \Psi) = \{\tau \in \Gamma \mid \forall \sigma \in \Sigma.\ \pi(s, \sigma, \tau) \models \Psi\}$ as the set of winning player-2 strategies for Ψ from s. Using the notation Win_1 and Win_2, the semantics of simple one-alternation strategy formulas can be written as follows: if $\Phi = (\exists\ \Psi_1,\ \exists\ \Psi_2,\ \Psi_3)$, then $[\![\Phi]\!] = \{s \in S \mid \exists \sigma \in \mathsf{Win}_1(s, \Psi_1).\ \exists \tau \in \mathsf{Win}_2(s, \Psi_2).\ \pi(s, \sigma, \tau) \models \Psi_3\}$. Thus the intuitive interpretation of the formula $\Phi = (\exists\ \Psi_1,\ \exists\ \Psi_2,\ \Psi_3)$ is as follows: there is a player-1 winning strategy σ for Ψ_1 and there is a player-2 winning strategy τ for Ψ_2 such that the path formed by fixing the strategies σ and τ satisfies Ψ_3.

5. Expressive Power of Strategy Logic

In this section we show that ATL^* and several concepts in nonzero-sum games can be expressed in the simple one-alternation fragment of strategy logic. We also show that *game logic*, which was introduced in [AHK02] to express the module-checking problem [KVW01], can be expressed in the one-alternation fragment of strategy logic (but not in the simple one-alternation fragment).

Expressing ATL^* and ATL. For every path formula Ψ, we have

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

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

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 defined in the simple one-alternation fragment of strategy logic, and ATL can be defined in the simple one-alternation fragment with unnested path formulas. Observe that ATL^* formulas are expressed as closed formulas, and hence nesting of $\langle\!\langle\cdot\rangle\!\rangle$ operators of ATL^* formulas does not increase the alternation depth when expressed as formulas in the strategy logic. Hence ATL^* can be expressed in the simple one-alternation fragment of strategy logic.

Expressing Nash equilibria. In nonzero-sum games the input is a labeled game graph and two path formulas, which express the objectives of the two players. We define Nash equilibria [Nas50] and show that their existence can be expressed in the simple one-alternation fragment of strategy logic.

Payoff profiles. Given a labeled game graph (G, P, L), two path formulas Ψ_1 and Ψ_2, strategies σ and τ for the two players, and a state $s \in S$, the *payoff* for player ℓ, where $\ell \in \{1, 2\}$, is defined as follows:

$$p_\ell(s, \sigma, \tau, \Psi_\ell) = \begin{cases} 1 & \text{if } \pi(s, \sigma, \tau) \models \Psi_\ell; \\ 0 & \text{otherwise.} \end{cases}$$

The *payoff profile* (p_1, p_2) consists of the payoffs $p_1 = p_1(s, \sigma, \tau, \Psi_1)$ and $p_2 = p_2(s, \sigma, \tau, \Psi_2)$ for player 1 and player 2.

Nash equilibria. A *strategy profile* (σ, τ) consists of strategies $\sigma \in \Sigma$ and $\tau \in \Gamma$ for the two players. Given a labeled game graph (G, P, L) and two path formulas Ψ_1 and Ψ_2, the strategy profile (σ^*, τ^*) is a *Nash equilibrium* at a state $s \in S$ if the following two conditions hold:

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

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

The state sets of the corresponding payoff profiles are defined as follows: for $i, j \in \{0, 1\}$, we have

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

Existence of Nash equilibria. We now define the state sets of the payoff profiles for Nash equilibria by simple one-alternation strategy formulas. The formulas are as follows:

$$NE(1,1) = [\![(\exists \mathbf{tt},\ \exists \mathbf{tt},\ \Psi_1 \wedge \Psi_2)]\!];$$

$$NE(0,0) = [\![(\exists \neg\Psi_2,\ \exists \neg\Psi_1,\ \mathbf{tt})]\!];$$

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

$$NE(0,1) = [\![(\exists \mathbf{tt},\ \exists \neg\Psi_1,\ \Psi_2)]\!].$$

Intuitively, every pair of strategies σ and τ, such that $\pi(s, \sigma, \tau) \models \Psi_1 \wedge \Psi_2$ is a $NE(1,1)$ because it achieves the maximal payoff for each player, and hence neither player has an incentive to deviate. Conversely, if a strategy pair σ and τ ensures a $NE(1,1)$, then $\pi(s, \sigma, \tau) \models \Psi_1 \wedge \Psi_2$. For a Nash equilibrium strategy pair (σ, τ), if the payoff for player 1 is 0, then it means that for all strategies σ' we have $\pi(s, \sigma', \tau) \not\models \Psi_1$, i.e., the strategy τ ensures $\neg\Psi_1$ (or in other words, the strategy τ is winning for $\neg\Psi_1$). Similar reasoning holds when the payoff for player 2 is 0 for a Nash equilibrium. Every pair of strategies σ and τ, such that σ ensures $\neg\Psi_2$ and τ ensures $\neg\Psi_1$ is in $NE(0,0)$ because for every strategy σ' player 1's payoff is going to remain 0, and analogously for player 2. Conversely, if σ and τ is Nash equilibrium strategy such that the payoff profile is $(0,0)$, then σ ensures $\neg\Psi_1$ and τ ensures $\neg\Psi_2$. Finally, if σ ensures $\neg\Psi_2$ and $\pi(s, \sigma, \tau) \models \Psi_1$, then we have a $NE(1,0)$. Indeed, player 1's payoff can only decrease and player 2's payoff remains 0 regardless of the chosen strategy. Thus we obtain the above characterization of Nash equilibrium payoff profile by strategy logic formulas.

Expressing secure equilibria. A notion of conditional competitiveness in nonzero-sum games was formalized by introducing secure equilibria [CHJ04]. The notion of secure has been shown to be relevant in the context of verification and assume-guarantee synthesis [CH07]. We show that the existence of secure equilibria can be expressed in the simple one-alternation fragment of strategy logic.

Lexicographic ordering of payoff profiles. We define two lexicographic orderings $\preceq_1$ and $\preceq_2$ on payoff profiles. For two 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 equilibria. A secure equilibrium is a Nash equilibrium with respect to the lexicographic preference orderings $\preceq_1$ and $\preceq_2$ on payoff profiles for the two players. Formally, given a labeled game graph (G, P, L) and two path formulas Ψ_1 and Ψ_2, a strategy profile (σ^*, τ^*) is a *secure equilibrium* at a state $s \in S$ if the following two conditions hold:

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

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

The state sets of the corresponding payoff profiles are defined as follows: for $i, j \in \{0,1\}$, we have

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

It follows from the definitions that the sets $SE(i,j)$, for $i, j \in \{0,1\}$, can be expressed in the one-alternation fragment (in the $\exists\forall$ fragment). The state sets of maximal payoff profiles for secure equilibria are defined as follows: for $i, j \in \{0,1\}$, we have

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

The following alternative characterizations of these sets are established in [CHJ04]:

$$MS(1,0) = \{s \in S \mid \mathsf{Win}_1(s, \Psi_1 \wedge \neg\Psi_2) \neq \emptyset\};$$

$$MS(0,1) = \{s \in S \mid \mathsf{Win}_2(s, \Psi_2 \wedge \neg\Psi_1) \neq \emptyset\};$$

$$MS(1,1) = \{s \in S \mid \exists \sigma \in \mathsf{Win}_1(s, \Psi_2 \Rightarrow \Psi_1).\ \exists \tau \in \mathsf{Win}_2(s, \Psi_1 \Rightarrow \Psi_2).\ \pi(s,\sigma,\tau) \models \Psi_1 \wedge \Psi_2\};$$

$$MS(0,0) = S \setminus (MS(1,0) \cup MS(0,1) \cup MS(1,1)).$$

Existence of secure equilibria. From the alternative characterizations of the state sets of the maximal payoff profiles for secure equilibria, it follows that these sets can be defined by simple one-alternation strategy formulas. The formulas are as follows:

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

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

$$MS(1,1) = [\![(\exists(\Psi_2 \Rightarrow \Psi_1),\ \exists(\Psi_1 \Rightarrow \Psi_2),\ \Psi_1 \wedge \Psi_2)]\!].$$

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

Game logic and module checking. The syntax of *game logic* [AHK02] is as follows. State formulas have the form $\exists\{1\}.\ \theta$ or $\exists\{2\}.\ \theta$, where θ is a tree formula. Tree formulas are (a) state formulas, (b) boolean combinations of tree formulas, and (c) either $\exists\Psi$ or $\forall\Psi$, where Ψ is a path formula. Informally, the formula $\exists\{1\}.\ \theta$ is true at a state if there is a strategy σ for player 1 such that the tree formula θ is satisfied in the tree that is generated by fixing the strategy σ for player 1 (see [AHK02] for details). Game logic can be defined in the one-alternation fragment of strategy logic (but not in the simple one-alternation fragment). The following example illustrates how to translate a state formula of game logic into a one-alternation strategy formula:

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

Consequently, the module-checking problem [KVW01] can be expressed by one-alternation strategy formulas.

The following theorem compares the expressive power of strategy logic and its fragments with ATL^*, game logic, the alternating-time μ-calculus [AHK02,Koz83], and monadic second-order logic [Rab69,Tho97]. We remark that our formulas are interpreted over 2-player turn-based game structures, and the followint theorem compares the expressive powers of different logics when interpreted over 2-player turn-based game structures.

Theorem 1 (Expressiveness). *The following assertions hold.*

(i) The expressiveness of the simple one-alternation fragment of strategy logic coincides with ATL^**, and the one-alternation fragment of strategy logic is more expressive than* ATL^**.*

(ii) The one-alternation fragment of strategy logic is more expressive than game logic, and game logic is more expressive than the simple one-alternation fragment of strategy logic.

(iii) The alternating-time μ-calculus is not as expressive as the alternation-free fragment of strategy logic, and strategy logic is not as expressive as the alternating-time μ-calculus.

(iv) Monadic second order (MSO) logic is more expressive than strategy logic.

Proof. We prove all the cases below.

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

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

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

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

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

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) (see the paragraph following Theorem 5) that the expressiveness of ATL^* and simple one-alternation fragment coincide. Since game logic is more expressive than ATL^* [AHK02] and game logic can be expressed in the one-alternation fragment it follows that one-alternation fragment is more expressive than ATL^*.

(ii) The game logic can be expressed in $\exists\forall$ fragment of the strategy logic. The translation a state formula of game logic into a closed one-alternation strategy formula is as follows:

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

The translation of any formula of game logic to a closed one-alternation strategy logic formula follows by induction. Game logic is not as expressive as one-alternation fragment: for example, in one-alternation fragment of strategy logic starting with a existential quantification over several strategy variables (such as $\exists x_1.\exists x_2.\exists y_1.\exists y_2.$) several strategy trees are created that can be analyzed simultaneously by the inner formula, and such multiple instantiations of trees simultaneously is not allowed in game logic. By part (1) it follows that the expressive power of simple one-alternation fragment coincides with ATL^*, and the result of [AHK02] shows that game logic is more expressive that ATL^*. It follows that game logic is more expressive than the simple one-alternation fragment.

(iii) It follows from the results of [AHK02] 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: states with proposition p acts as player 1 states (i.e., $[\![p]\!] = S_1$) and states with proposition $\neg p$ acts as player 2 states (i.e., $[\![\neg p]\!] = S_2$), and the formula specifies that there is a p player strategy to reach q against all $\neg p$ player stategy. Thus alternating μ-calculus can transform a one-player structure to a 2-player game structure. Strategy logic formulas on one-player structure can only quantify over strategies of one player and cannot convert it to a 2-player game structure. Hence the above property is not expressible by strategy logic on one-player structures.

(iv) We now argue that MSO is more expressive than strategy logic: encoding strategies as trees, a strategy logic formula can be translated to an 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. ∎

6. Model Checking Strategy Logic

In this section we solve the model-checking problem for 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 the satisfiability problem of monadic second-order logic [Rab69]. We give the necessary definitions and proceed with the algorithm.

Strategy 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 Λ, a *Λ-labeled Υ-tree* is a pair $\langle T, \rho \rangle$, where T is an Υ-tree, and ρ: $T \to \Lambda$ maps each node of T to a letter in Λ. When Υ and Λ are not important or clear from the context, we call $\langle T, \rho \rangle$ a labeled tree. We say that an $((\Upsilon \cup \{\perp\}) \times \Lambda)$-labeled Υ-tree $\langle T, \rho \rangle$ is *Υ-exhaustive* if for every node $z \in T$, we have $\rho(z) \in \{dir(z)\} \times \Lambda$.

Consider a game graph $G = ((S, E), (S_1, S_2))$. For $\alpha \in \{1, 2\}$, a strategy σ: $S^* \cdot S_\alpha \to S$ can be encoded by an S-labeled S-tree $\langle S^*, \rho \rangle$ by setting $\sigma(v) = \rho(v)$ 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 $\rho(v)$ may be set arbitrarily. We may encode k different strategies by considering an S^k-labeled S-tree. Given a letter $\lambda \in S^k$, we denote by λ_i the projection of λ on its i-th coordinate. In this case, the i-th strategy is $\sigma_i(v) = \rho(v)_i$ for every $v \in S^* \cdot S_\alpha$. Notice that the different encoded strategies may belong to different players. We refer to such trees as *strategy trees*, and from now on, we may refer to a strategy as a tree $\langle S^*, \sigma \rangle$. In what follows we encode strategies by strategy trees. We construct tree automata that accept the strategy assignments that satisfy a given formula of strategy logic.

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 [Tho97]. Here, we use *alternating parity tree automata* (APTs). The language of an automaton is the set of labeled trees that it accepts. The size of an automaton is measured by the number of states, and the index, which is a measure of the complexity of the acceptance (parity) condition. The important qualities of automata that are needed for this paper are summarized in Theorem 2 below.

Theorem 2 *(i) Given an* LTL *formula Ψ, we can construct an APT $\mathcal{A}_\Psi$ with $2^{O(|\Psi|)}$ states and index* 3 *such that $\mathcal{A}_\Psi$ accepts all labeled trees all of whose paths satisfy Ψ* [VW94].

(ii) Given two 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 the languages of $\mathcal{A}_1$ and $\mathcal{A}_2$ with $n_1 + n_2$ states and index $\max(k_1, k_2)$*. We can also construct an APT for the complementary language of $\mathcal{A}_1$ with n_1 states and index k_1* [MS87].

(iii) Given an APT $\mathcal{A}$ with n states and index k over the alphabet $\Lambda \times \Lambda'$, we can construct an APT $\mathcal{A}'$ that accepts a labeled tree over the alphabet Λ if some extension (or all extensions) of the labeling with labels from Λ' is accepted by $\mathcal{A}$. The number of states of $\mathcal{A}'$ is exponential in $n \cdot k$, and its index is linear in $n \cdot k$ [MS87].

(iv) 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 $n \cdot k$ [EJS93,MS87].

Model-checking algorithm. The complexity of the model-checking algorithm for strategy formulas depends on the number of quantifier alternations of a formula. We now formally define the alternation depth of a closed strategy formula. The *alternation depth* of a strategy formula is defined as follows. For path formulas Ψ, we set $QA(\Psi) = 0$. For a strategy formula Φ we set $QA(\Phi)$ as follows.

– If $\Phi = \Psi(x, y)$, then $QA(\Phi) = 0$.

– If $\Phi = \Phi_1 \wedge \Phi_2$ or $\Phi = \Phi_1 \vee \Phi_2$ then $QA(\Phi) = max(QA(\Phi_1), QA(\Phi_2))$.
– If $\Phi = \exists x.\Phi$ or $\Phi = \exists y.\Phi$ then $QA(\Phi)$ is $max(QA(\Phi') + 1)$, where Φ' ranges over all universally quantified subformulas of Φ. If Φ has no universally quantified subformulas, then $QA(\Phi) = 0$.
– If $\Phi = \forall x.\Phi$ or $\Phi = \forall y.\Phi$ then $QA(\Phi)$ is $max(QA(\Phi') + 1)$, where Φ' ranges over all existentially quantified subformulas of Φ. If Φ has no existentially quantified subformulas, then $QA(\Phi) = 0$.

For example, consider the formula $\Phi = \exists x_1.\ \forall y_1.((\forall x_2.\Psi_2(x_2, y_1)) \Rightarrow \Psi_1(x_1, y_1))$. The quantifier alternation of $\exists x_2 \neg\Psi_2(x_2, y_1)$, $\Psi_1(x_1, y_1)$, and their conjunction is zero. The quantifier alternation of $\forall y_1.((\forall x_2.\Psi_2(x_2, y_1)) \Rightarrow \Psi_1(x_1, y_1))$ is one. Finally, $QA(\Phi) = 2$.

Given a strategy formula Φ, we construct by induction on the structure of the formula a nondeterministic parity tree (NPT) automaton that accepts the set of strategy assignments that satisfy the formula. Without loss of generality, we assume that the variables in $X \cup Y$ are not reused; that is, in a closed strategy formula, there is a one-to-one and onto relation between the variables and the quantifiers.

Theorem 3 *Given a labeled game graph $\mathcal{G}$ and a closed strategy formula Φ of alternation depth d, we can compute the set $[\![\Phi]\!]$ of states in time proportional to d-EXPTIME in the size of $\mathcal{G}$, and $(d+1)$-EXPTIME in the size of Φ. If Φ contains only unnested path formulas, then the complexity in the size of the formula reduces to d-EXPTIME.*

Proof. The case where a closed strategy logic formula Φ is used as a state formula in a larger formula Φ', is solved by first computing the set of states satisfying Φ, adding this information to the labeled game graph $\mathcal{G}$, and then computing the set of states satisfying Φ'. In addition, if d is the alternation-depth of Φ then Φ is a boolean combination of closed strategy formulas of alternation depth at most d. Thus, it suffices to handle a closed strategy formula, and reduce the boolean reasoning to intersection, union, and complementation of the respective sets. Consider a strategy formula Φ. Let $Z = \{x_1, \ldots, x_n, y_1, \ldots, y_m\}$ be the set of variables used in Φ. 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 Φ. We build the APT by induction on the structure of the formula. For a subformula Φ' we consider the following cases.

Case 1. $\Phi' = \Psi(x, y)$ —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. $\Phi' = \Phi_1 \wedge \Phi_2$ —given APTs $\mathcal{A}_1$ and $\mathcal{A}_2$ that accept the set of strategy assignments that satisfy Φ_1 and Φ_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. $\Phi' = \exists x.\Phi_1$ —given an APT $\mathcal{A}_1$ that accepts the set of strategy assignments that satisfy Φ_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 $n \cdot k$. Furthermore, we have to check that the extra labeling corresponds to a valid strategy. That is, check that the transition enabled from a given state in the tree correspond to the transition enabled from the same state in the game. This requires multiplying the number of states of A' by the size of the game. The cases where $\Phi' = \exists y.\Phi_1$, $\Phi' = \forall x.\Phi_1$, and $\Phi' = \forall y.\Phi_1$ are handled similarly.

We note that for a closed strategy formula Φ, 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 addition, the first quantification multiplies the number of states of the automaton in the size of the game. Further quantifiers then incur an exponent in the size of the game as well. In Appendix A we specialize these results by using nondeterministic and universal tree automata to get the exact complexity of $d+1$ 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

[1] For a single temporal operator the number of states is constant, and 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

total complexity is d exponents in the size of the formula and d exponents in the size of the game. Thus in the case of unnested path formulas one exponent can be removed. The inductive construction is just like the construction for the unrestricted logic that is presented in the appendix. ∎

One-alternation fragment. Since ATL^* can be expressed in the simple one-alternation fragment of strategy logic, it follows that model checking simple one-alternation strategy formulas is 2EXPTIME-hard [AHK02]. Also, since module checking can be expressed in the one-alternation fragment, it follows that model checking one-alternation strategy formulas with unnested path formulas is EXPTIME-hard [KVW01]. These lower bounds together with Theorem 3 yield the following results.

Theorem 4 *Given a labeled game graph $\mathcal{G}$ and a closed one-alternation strategy formula Φ, the computation of $[\![\Phi]\!]$ is EXPTIME-complete in the size of $\mathcal{G}$, and 2EXPTIME-complete in the size of Φ. If Φ contains only unnested path formulas, then the complexity in the size of the formula is EXPTIME-complete.*

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

Notation. For a labeled game graph $\mathcal{G}$ and a set $U \subseteq S$ of states, we denote by $\mathcal{G} \upharpoonright U$ the restriction of the labeled game graph to the set U, and we use the notation only when for all states $u \in U$, we have $E(u) \cap U \neq \emptyset$; i.e., all states in U have a successor in U. A path formula Ψ is *infinitary* if the set of paths that satisfy Ψ is independent of all finite prefixes. Formally, for a path π and finite prefix w of π, let $\pi|w$ be the path obtained by removing w from π. A path formula Ψ is infinitary if for all paths π and all finite prefixes w of π we have $\pi \models \Psi$ iff $\pi|w \models \Psi$. The classical Büchi, coBüchi, parity, Rabin, Streett, and Müller conditions are all infinitary conditions. Every LTL objective on a labeled game graph can be reduced to an infinitary condition, such as a parity or Müller condition, on a modified game graph.

Lemma 1 *Let $\mathcal{G}$ be a labeled game graph, and let $\Phi = (\exists\ \Psi_1,\ \exists\ \Psi_2,\ \Psi_3)$ be a simple one-alternation strategy 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 $[\![\Phi]\!] = [\![\langle\!\langle 1,2 \rangle\!\rangle(\Psi_1 \wedge \Psi_2 \wedge \Psi_3)]\!]$ in the restricted graph $\mathcal{G} \upharpoonright (W_1 \cap W_2)$.*

Proof. 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 $\mathcal{G} \upharpoonright W_1 \cap W_2$ 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 $\mathcal{G} \upharpoonright W_1 \cap W_2$ is a game graph. For any winning strategy pair (σ,τ) for all states $s \in W_1 \cap W_2$ we have $\pi(s,\sigma,\tau) \models \Box(W_1 \cap W_2)$ (i.e., $\pi(s,\sigma,\tau)$ only visits states in $W_1 \cap W_2$). Let $U = [\![\Phi]\!]$ and we prove that $U = [\![\langle\!\langle 1,2 \rangle\!\rangle(\Psi_1 \wedge \Psi_2 \wedge \Psi_3)]\!]$ in $\mathcal{G} \upharpoonright (W_1 \cap W_2)$ by proving inclusion in both directions.

(i) 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 $\mathcal{G} \upharpoonright (W_1 \cap W_2)$. 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 $\pi(s,\sigma,\tau) \models \Box(W_1 \cap W_2)$, 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 $\mathcal{G} \upharpoonright (W_1 \cap W_2)$.

(ii) 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 $\mathcal{G} \upharpoonright (W_1 \cap W_2)$. Fix a witness strategy pair (σ_1,τ_1) in $\mathcal{G} \upharpoonright (W_1 \cap W_2)$ 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

in the size of the formula, resulting in a linear total number of states.

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 (a) $\pi(s, \sigma_1, \tau_1) \models \Psi_1 \wedge \Psi_2 \wedge \Psi_3$ (hence also $\pi(s, \sigma_1, \tau_1) \models \Psi_1 \wedge \Psi_2$), (b) the players switch to a respective winning strategy if the other player deviates, and (c) Ψ_1 and Ψ_2 are infinitary, 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 desired result follows and it also follows that

$$[\![\Phi]\!] = [\![\langle\!\langle 1,2 \rangle\!\rangle \Big(\Psi_1 \wedge \Psi_2 \wedge \Psi_3 \wedge \Box\big(\langle\!\langle 1 \rangle\!\rangle(\Psi_1) \wedge \langle\!\langle 2 \rangle\!\rangle(\Psi_2)\big)\Big)]\!].$$

∎

Lemma 2 *Let $\mathcal{G}$ be a labeled game graph, and let $\Phi = (\exists\ \Psi_1,\ \exists\ \Psi_2,\ \Psi_3)$ be a simple one-alternation strategy formula with unnested path formulas Ψ_1, Ψ_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 $[\![\Phi]\!] = [\![\langle\!\langle 1,2 \rangle\!\rangle(\Psi_1 \wedge \Psi_2 \wedge \Psi_3)]\!] \cap W_1 \cap W_2$.*

Proof. 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.

(i) 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., $s \in [\![\langle\!\langle 1,2 \rangle\!\rangle(\Psi_1 \wedge \Psi_2 \wedge \Psi_3)]\!] \cap W_1 \cap W_2$.

(ii) 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 $\mathcal{G} \upharpoonright (W_1 \cap W_2)$ 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: let $\Psi_1 = \Phi_1^1 \mathcal{U} \Phi_1^2$ and and $\Psi_2 = \Phi_2^1 \mathcal{U} \Phi_2^2$. The witness strategy construction follows from the following two parts.
  - *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 desired result follows and it also follows that

$$[\![\Phi]\!] = [\![\langle\!\langle 1,2 \rangle\!\rangle(\Psi_1 \wedge \Psi_2 \wedge \Psi_3) \wedge \langle\!\langle 1 \rangle\!\rangle(\Psi_1) \wedge \langle\!\langle 2 \rangle\!\rangle(\Psi_2)]\!].$$

∎

Theorem 5 *Let $\mathcal{G}$ be a labeled game graph with n states, and let $\Phi = (\exists\ \Psi_1,\ \exists\ \Psi_2,\ \Psi_3)$ be a simple one-alternation strategy formula.*

(i) We can compute the set $[\![\Phi]\!]$ of states in $n^{2^{O(|\Phi|)}} \cdot 2^{2^{O(|\Phi| \cdot \log|\Phi|)}}$ time; hence for formulas Φ of constant length the computation of $[\![\Phi]\!]$ is polynomial in the size of $\mathcal{G}$. The computation of $[\![\Phi]\!]$ is 2EXPTIME-complete in the size of Φ.

(ii) If Ψ_1, Ψ_2, and Ψ_3 are unnested path formulas, then there is an ATL^* *formula Φ' with unnested path formulas such that $|\Phi'| = O(|\Psi_1| + |\Psi_2| + |\Psi_3|)$ and $[\![\Phi]\!] = [\![\Phi']\!]$. Therefore $[\![\Phi]\!]$ can be computed in polynomial time.*

Theorem 5 follows from Lemmas 1 and 2. We present details only for part (1): given Ψ_1, Ψ_2, and Ψ_3 as 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 a conjunction of parity conditions (i.e., a Streett condition). An 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 automaton, and then determinizing). From the conversion of an LTL formula to a deterministic parity automaton (that is an infinitary condition) and Lemma 1 it follows that expressive power of simple one-alternation fragment and ATL^* coincide. By applying an algorithm for solving parity games [Jur00] and a polynomial-time algorithm for model checking Streett conditions, we obtain the desired upper bound. Observe that the model-checking complexity of the simple one-alternation fragment of strategy logic with unnested path formulas, as well as the program complexity of the simple one-alternation fragment (i.e., the complexity in terms of the game graph, for formulas of bounded size), are exponentially better than the corresponding complexities of the full one-alternation fragment.

7. Conclusions

We introduced Strategy Logic, a logic for reasoning about two-player games. The logic treats strategies as explicit first-order objects. We showed that the logic is more expressive than ATL, ATL^*, and game logic. Nash equilibria and secure equilibria can be directly expressed in the logic. We gave automata-theoretic algorithms for model checking Strategy Logic, and more efficient algorithms for the one-alternation fragment of the logic. In the case of the one-alternation fragment, our algorithms are tight.

One important direction of extending Strategy Logic would be to concurrent games [dAHK98] and stochastic games [Con92]. In these games, winning strategies may need randomization [dAHK98], and qualitative as well as quantitative modes of winning are of interest. In the quantitative case, given strategies for both players, the value of an LTL formula would no longer be Boolean, but a probability value. Existential and universal quantification over strategies may then be replaced by the supremum and infimum. Current techniques for stochastic games [CJH04,CdAH05] may enable us to model check the one-alternation fragment of the logic, but we have no theory of automata that would allow a simple solution to the general model-checking problem. We note that the work of Baier et al. [BBGK07] extends ATL and ATL^* with probabilities but lacks the explicit treatment of strategies as in Strategy Logic.

We mentioned that it may be worthwhile to extend Strategy Logic also in the direction of partial-information games. One would then have to introduce a hierarchy of knowledge within the logic to preserve decidability [Kai06]. In addition, pure (i.e., non-randomized) strategies are no longer sufficient for partial-information games [CH05,CDHR06], and as for concurrent and stochastic games, the lack of automata-theoretic methods for probabilities makes model checking difficult. If we restrict our attention to handle pure strategies, work on handling partial-information games in the context of automata [KV00] may be extended to our setting.

Finally, the algorithms presented in this article for model checking Strategy Logic are exponential in both the size of the formula and the size of the game structure. We showed the tightness of the complexity bounds in the case of the one-alternation fragment of the logic. For higher levels of the quantifier alternation hierarchy, the complexity keeps increasing, but no matching lower bounds are known. It would be interesting to find either matching lower bounds or improve the upper bounds. This is especially important for the complexity in terms of size of the game structure.

Acknowledgments. We thank Lukasz Kaiser for discussions on possible extension of Strategy Logic to partial-information games. This research was supported in part by the Swiss National Science Foundation and by the NSF grants CCR-0225610 and CCR-0234690.

References

[AHK02] R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713, September 2002.

[ALW89] 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. Springer-Verlag, July 1989.

[BBGK07] C. Baier, T. Brázdil, M. Grösser, and A. Kucera. Stochastic game logic. In *QEST'07*, pages 227–236. IEEE, 2007.

[BGNV05] A. Blass, Y. Gurevich, L. Nachmanson, and M. Veanes. Play to test. In *5th Formal Approaches to Software Testing*, volume 3997 of *Lecture Notes in Computer Science*, pages 32–46. Springer-Verlag, 2005.

[CdAH05] K. Chatterjee, L. de Alfaro, and T.A. Henzinger. The complexity of stochastic Rabin and Streett games. In *ICALP'05*, pages 878–890. LNCS 3580, Springer, 2005.

[CDHR06] K. Chatterjee, L. Doyen, T.A. Henzinger, and J.F. Raskin. Algorithms for omega-regular games with imperfect information. In *CSL'06*, pages 287–302. LNCS 4207, Springer, 2006.

[CH05] K. Chatterjee and T.A. Henzinger. Semiperfect-information games. In *FSTTCS'05*. LNCS 3821, Springer, 2005.

[CH07] K. Chatterjee and T.A. Henzinger. Assume guarantee synthesis. In *13th Tools and Algorithms for the Construction and Analysis of Systems*, volume 4424 of *Lecture Notes in Computer Science*, pages 261–275. Springer-Verlag, 2007.

[CHJ04] 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.

[CJH04] K. Chatterjee, M. Jurdziński, and T.A. Henzinger. Quantitative stochastic parity games. In *SODA'04*, ACM-SIAM, pages 114–123, 2004.

[Con92] A. Condon. The complexity of stochastic games. *Information and Computation*, 96(2):203–224, 1992.

[dAH01] L. de Alfaro and T.A. Henzinger. Interface automata. In *8th Foundations of Software Engineering*, pages 109–120. ACM press, 2001.

[dAHK98] L. de Alfaro, T.A. Henzinger, and O. Kupferman. Concurrent reachability games. In *FOCS,98*, pages 564–575. IEEE, 1998.

[Dil89] D.L. Dill. *Trace theory for automatic hierarchical verification of speed independent circuits*. MIT Press, 1989.

[EJS93] 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.

[GPSS80] 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.

[HKR02] T.A. Henzinger, O. Kupferman, and S. Rajamani. Fair simulation. *Information and Computation*, 173(1):64–81, 2002.

[Nas50] J.F. Nash Jr. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences USA*, 36:48–49, 1950.

[Jur00] 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.

[Kai06] L. Kaiser. Game quantification on automatic structures and hierarchical model checking games. In *15th Computer Science Logic*, volume 4207 of *Lecture Notes in Computer Science*, pages 411–425. Springer-Verlag, 2006.

[Kam68] J.A.W. Kamp. *Tense Logic and the Theory of Order*. PhD thesis, UCLA, 1968.

[Koz83] D. Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27:333–354, 1983.

[KV00] O. Kupferman and M.Y. Vardi. Synthesis with incomplete information. In *Advances in Temporal Logic*, pages 109–127. Kluwer Academic Publishers, January 2000.

[KV05] O. Kupferman and M.Y. Vardi. Safraless decision procedures. In *Proc. 46th IEEE Symp. on Foundations of Computer Science*, pages 531–540, Pittsburgh, October 2005.

[KVW01] O. Kupferman, M.Y. Vardi, and P. Wolper. Module checking. *Information and Computation*, 164:322–344, 2001.

[Mar75] D.A. Martin. Borel determinacy. *Annals of Mathematics*, 65:363–371, 1975.

[Mil71] 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.

[MS87] D.E. Muller and P.E. Schupp. Alternating automata on infinite trees. *Theoretical Computer Science*, 54:267–276, 1987.

[Owe95] G. Owen. *Game Theory*. Academic Press, 1995.

[PR89] 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.

[Rab69] M.O. Rabin. Decidability of second order theories and automata on infinite trees. *Transaction of the AMS*, 141:1–35, 1969.

[RW89] P.J.G. Ramadge and W.M. Wonham. The control of discrete event systems. *IEEE Transactions on Control Theory*, 77:81–98, 1989.

[Tho97] W. Thomas. Languages, automata, and logic. *Handbook of Formal Language Theory*, III:389–455, 1997.

[VW94] M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, November 1994.

Appendix

Appendix A. Model Checking

We specialize the automata construction presented in the proof of Theorem 3 to get the exact stated complexity bound. We start by stating a few more known results and then incorporate them into our construction.

We start by formalizing the concept of d-EXPTIME. Let $exp(0, n) = n$ and let $exp(d + 1, n) = 2^{exp(d,n)}$. We say that a function f is d-EXPTIME in n if for some polynomial p and for every n we have $f(n) \leq exp(d, p(n))$. The following properties of the function $exp(d, n)$ are simple to show.

Lemma 3 *For every positive d_1, d_2, n_1, and n_2, if $d = max(d_1, d_2)$ and $n = max(n_1, n_2)$ all the following are true.*

- $2 \cdot exp(d_1, n_1) \leq exp(d_1, n_1 + 1)$.
- $exp(1, n_1) \cdot exp(1, n_2) = exp(1, n_1 + n_2)$ *and for* $d > 1$ *we have* $exp(d_1, n_1) \cdot exp(d_2, n_2) \leq exp(d, n+1)$.
- $exp(d_1, n_1) + exp(d_2, n_2) \leq exp(d, n + 1)$.
- $exp(1, n)! \leq exp(2, n\ log\ n)$ *and for* $d_1 > 1$ *we have* $exp(d_1, n_1)! \leq exp(d_1 + 1, n_1 + 1)$.

Proof.

Case 1. If $d_1 = 1$, then $2 \cdot exp(d_1, n_1) = 2 \cdot 2^{n_1} = 2^{n_1+1} = exp(d_1, n_1 + 1)$. If $d_1 > 1$, then $2 \cdot exp(d_1, n_1) = 2^{1+exp(d_1-1,n_1)} \leq 2^{exp(d_1-1,n_1+1)} = exp(d_1, n_1 + 1)$.

Case 2.

$exp(1, n_1) \cdot exp(1, n_2) = 2^{n_1+n_2} = exp(1, n_1 + n_2)$

Suppose that $d > 1$.

$$\begin{array}{lll} exp(d_1, n_1) \cdot exp(d_2, n_2) & \leq & exp(d, n) \cdot exp(d, n) \quad = \\ 2^{exp(d-1,n)+exp(d-1,n)} & \leq & 2^{exp(d-1,n+1)} \quad = \\ exp(d, n + 1) & & \end{array}$$

Case 3. Follows from Case (1) above.

Case 4.

Suppose that $d_1 > 1$.

$$\begin{array}{lllll} exp(d_1, n_1)! & \leq & exp(d_1, n_1)^{exp(d_1,n_1)} & = & 2^{exp(d_1-1,n_1)\cdot exp(d_1,n_1)} \quad \leq \\ 2^{exp(d_1,n_1+1)} & = & exp(d_1 + 1, n_1 + 1) & & \end{array}$$

∎

We use the acronyms NBW for nondeterministic Büchi word automaton and NPT for nondeterministic parity tree automaton. Given an APT $\mathcal{A} = \langle \Sigma, S, \delta, s_0, \alpha \rangle$ we define the dual $\overline{\mathcal{A}} = \langle \Sigma, S, \overline{\delta}, s_0, \alpha + 1 \rangle$, where $\overline{\delta}(s, a)$ is obtained from $\delta(s, a)$ by replacing $\vee$ by $\wedge$ and $\wedge$ by $\vee$ and $\alpha + 1$ is the acceptance condition obtained from α by replacing F_i by F_{i+1}. It is well known that $\mathcal{A}$ and $\overline{\mathcal{A}}$ accept complementary languages. We use the acronym UPT for the dual of an NPT. The transition function of an NPT can be represented as $\delta : S \times \Sigma \rightarrow 2^{S^k}$ where k is the branching degree of the tree, or equivalently for every $s \in S$ and $a \in \Sigma$ we have $\delta(s, a) = \bigvee_{i \in I} \bigwedge_{j=1}^{k} (j, s_{i,j})$. For a UPT, the transition function is $\overline{\delta}$ such that for every $s \in S$ and $a \in \Sigma$ we have $\overline{\delta}(s, a) = \bigwedge_{i \in I} \bigvee_{j=1}^{k} (j, s_{i,j})$. Notice that this is different from the usual definition of universal tree automaton where the transition is purely conjunctive (cf. [KV05]).

We now state a few known results about automata and logic.

Theorem 6 *(i) Given a path formula $\Psi(x, y)$, we can construct an NBW $\mathcal{N}_\Psi$ with $2^{O(|\Psi|)}$ states that accepts all infinite words that satisfy Ψ [VW94].*

(ii) Given two NPTs $\mathcal{N}_1$ and $\mathcal{N}_2$ with n_1 and n_2 states and indices k_1 and k_2, respectively, we can construct an NPT for the disjunction of $\mathcal{N}_1$ and $\mathcal{N}_2$ with $n_1 + n_2$ states and index $max(k_1, k_2)$ and an NPT for the conjunction of $\mathcal{N}_1$ and $\mathcal{N}_2$ with $n_1 \cdot n_2 \cdot \frac{(k_1+k_2)!}{k_1!k_2!}$ states and index $k_1 + k_2$.

(iii) Given an NPT $\mathcal{N}$ over alphabet $\Lambda \times \Lambda'$, we can construct an NPT $\mathcal{N}'$ that accepts a labeled tree over the alphabet Λ if some extension of the labeling with labels from Λ' is accepted by $\mathcal{N}$. The number of states of $\mathcal{N}'$ and its index are equal to those of $\mathcal{N}$.

(iv) Given an NPT $\mathcal{N}$ with n states and index k, we can check whether the language of $\mathcal{N}$ is empty in time proportional to $n^{O(k)}$ [Jur00].

(v) Given an APT $\mathcal{A}$ with n states and index k, we can construct an equivalent NPT $\mathcal{N}$ with $2^{O(nk\ log\ nk)}$ states and index $O(nk)$ [MS87].

The following are additional facts about NPT and UPT that mostly follow from the duality of NPT and UPT.

Corollary 1 *(i) Given a path formula $\Psi(x, y)$, we can construct an NBT $\mathcal{N}_\Psi$ with $|S| \cdot 2^{O(|\Psi|)}$ states that accepts all strategy trees such that the path described by the combination of strategies x and y satisfies Ψ.*

(ii) Given a path formula $\Psi(x, y)$, we can construct a UPT $\mathcal{U}_\Psi$ with $|S| \cdot 2^{O(|\Psi|)}$ states that accepts all strategy trees such that the path described by the combination of strategies x and y satisfies Ψ.

(iii) Given two UPTs $\mathcal{U}_1$ and $\mathcal{U}_2$ with n_1 and n_2 states and indices k_1 and k_2, respectively, we can construct a UPT for the conjunction of $\mathcal{U}_1$ and $\mathcal{U}_2$ with $n_1 + n_2$ states and index $max(k_1, k_2)$ and a UPT for the disjunction of $\mathcal{U}_1$ and $\mathcal{U}_2$ with $n_1 \cdot n_2 \cdot \frac{(k_1+k_2)!}{k_1!k_2!}$ states and index $k_1 + k_2$.

(iv) Given a UPT $\mathcal{U}$ over alphabet $\Lambda \times \Lambda'$, we can construct a UPT $\mathcal{U}'$ that accepts a labeled tree over alphabet Λ if all extensions of the labeling with labels from Λ' are accepted by $\mathcal{U}'$. The number of states of $\mathcal{U}'$ and its index are equal to those of $\mathcal{U}$.

(v) Given a UPT $\mathcal{U}$ with n states and index k, we can check whether the language of $\mathcal{N}$ is universal in time proportional to $n^{O(k)}$.

(vi) Given an NPT $\mathcal{N}$ with n states and index k, we can construct an equivalent UPT $\mathcal{U}$ with $2^{O(nk\ log\ nk)}$ states and index $O(nk)$.

(vii) Given a UPT $\mathcal{U}$ with n states and index k, we can construct an equivalent NPT $\mathcal{N}$ with $2^{O(nk\ log\ nk)}$ states and index $O(nk)$.

Proof.

(i) Consider the path formula Ψ. By Theorem 6 there exists an NBW $\mathcal{N}_\Psi$ that accepts all infinite words that satisfy Ψ. Let $\mathcal{N}_\Psi = \langle 2^P, Q, \delta, q_0, F\rangle$ and let S be the set of states of the game G. Consider some Λ-labeled S-tree $\langle T, \rho\rangle$ where $\Lambda = S^k$ for some k. For every $\lambda \in \Lambda$ denote by λ_x the projection of λ on the strategy x and by λ_y the projection of λ on the strategy y. We construct an NBT $\mathcal{N}$ that reads Λ-labeled S-trees and runs $\mathcal{N}_\Psi$ in the directions prescribed by the strategies x and y. Technically, we define the NBT $\mathcal{N} = \langle \Lambda, (S \times Q) \cup \{\top\}, \eta, (s_0, q_0), (S \times F) \cup \{\top\}\rangle$, where for every $(s, q) \in S \times Q$ and $\lambda \in \Lambda$ the transition η is defined as follows.

$$\eta((s,q),\lambda) = \begin{cases} \displaystyle\bigvee_{q'\in\delta(q,L(s))} (\rho_x(\lambda),(\rho_x(\lambda),q')) \wedge \bigwedge_{s'\neq\rho_x(\lambda)} (s',\top) & \text{If } s \in S_1 \\ \displaystyle\bigvee_{q'\in\delta(q,L(s))} (\rho_y(\lambda),(\rho_y(\lambda),q')) \wedge \bigwedge_{s'\neq\rho_x(\lambda)} (s',\top) & \text{If } s \in S_2 \end{cases}$$

Finally, for every $\lambda \in \Lambda$ the transition $\eta(\top, \lambda) = \bigwedge_{s\in S}(s, \top)$.

It is simple to see that $\mathcal{N}'$ accepts a strategy tree if the path that is the combination of x and y satisfies Ψ.

(ii) Consider the path formula $(\neg\Psi)(x, y)$. By the construction above there exists an NBT $\mathcal{N}_{\neg\Psi}$ that accepts all trees in which the path described by the combination of strategies x and y satisfies $\neg\Psi$. Let $\mathcal{U}_\Psi = \overline{\mathcal{N}_{\neg\Psi}}$. The UPT $\mathcal{U}_\Psi$ accepts a legal strategy tree if the path described by the combination of x and y does not satisfy Ψ.

(iii) Consider the NPTs $\mathcal{N}_1 = \overline{\mathcal{U}_1}$ and $\mathcal{N}_2 = \overline{\mathcal{U}_2}$. By Theorem 6 there exist NPTs $\mathcal{N}_\cup$ and $\mathcal{N}_\cap$ for the disjunction and conjunction of $\mathcal{N}_1$ and $\mathcal{N}_2$. The UPTs $\mathcal{U}_\cap = \overline{\mathcal{N}_\cup}$ and $\mathcal{U}_\cup = \overline{\mathcal{N}_\cap}$ are the required automata.

(iv) Let $\mathcal{N} = \overline{\mathcal{U}}$. By Theorem 6 there exists an NPT $\mathcal{N}_\exists$ that accepts a tree over alphabet Λ if some extension of the labeling with labels from Λ' is accepted by $\mathcal{N}$. It follows that $\mathcal{U}_\forall = \overline{\mathcal{N}_\exists}$ accepts a tree over alphabet Λ if all extensions of the labeling with labels from Λ' are accepted by $\mathcal{U}$.

(v) By duality of UPT and NPT.

(vi) Consider an NPT $\mathcal{N}$. It is simple to construct an APT $\mathcal{A}$ that complements $\mathcal{N}$ with same index and same number of states. By Theorem 6 there is an NPT $\mathcal{N}'$ that accepts the same language as $\mathcal{A}$. The UPT $\mathcal{U} = \overline{\mathcal{N}'}$ accepts the same language as $\mathcal{N}$.

(vii) This is dual to the construction described above for the conversion of NPT to UPT. ∎

We now turn to the proof of Theorem 3.

Proof. The idea behind this tighter construction is to use NPTs and UPTs and maintain them in this form as long as possible. Nondeterministic automata are good for existential quantification, which comes to them for free, and UPTs are good for universal quantification, which comes to them for free. Every quantifier alternation requires to go from NPT to UPT or vice versa incurring an exponential blow-up. As we start with an automaton whose size may be exponential in the size of a path subformula, we get $d+1$-exponentials in the size of the formula and d-exponentials in the size of the game, where d is the number of quantifier alternations.

Consider a formula Φ. As explained above, we assume that Boolean combinations of formulas are handled separately and thus assume Φ is of the form $Qz.\Phi'$ for $Q \in \{\exists, \forall\}$. If $Q = \forall$ we say that Φ is *universal* and if $Q = \exists$ we say that Φ is *existential*. Consider Φ' a strict subformula of Φ. We say that Φ' is *universal* if the minimal subformula $Qz.\Phi''$ such that Φ' is a subformula of Φ'' is universal, i.e., $Q = \forall$. We say that Φ' is *existential* if it is not universal. For every existential subformula we are going to construct an NPT and for every universal subformula we are going to construct a UPT.

Recall, that the set of strategy formulas is defined as follows:

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

Consider a strategy formula Φ. Let $Z = \{x_1, \ldots, x_n, y_1, \ldots, y_m\}$ be the set of variables used in Φ. 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 automaton that accepts the set of strategy assignments that satisfy Φ. Our automata respect the following invariants:

(i) The automaton is NPT whenever the subformula is existential

(ii) The automaton in UPT whenever the subformula is universal.

(iii) The size of the automaton is at most $d+1$-exponential in the size of the formula d-exponential in the size of the game and its index is at most d-exponential in the size of the formula and $d-1$-exponential in the size of the game, where d is the number of quantifier alternations of the subformula.

We build the automaton by induction on the structure of the formula. For a subformula Φ' we consider the following cases.

Case 1. $\Phi' = \Psi(x,y)$ where Φ' is existential—we construct an NPT $\mathcal{N}$ that accepts all trees in which the path prescribed by the combination of the strategies x and y satisfies Ψ. By Theorem 6 the automaton $\mathcal{N}$ has $|S| \cdot 2^{O(|\Psi|)}$ states and has index 3. It follows that the invariants are satisfied.

Case 2. $\Phi' = \Psi(x,y)$ where Φ' is universal—we construct a UPT $\mathcal{U}$ that accepts all trees in which the path prescribed by the combination of the strategies x and y satisfies Ψ. By Corollary 1 the automaton $\mathcal{U}$ has $|S| \cdot 2^{O(|\Psi|)}$ states and has index 3. It follows that the invariants are satisfied.

Case 3. $\Phi' = \Phi_1 \wedge \Phi_2$ where Φ' is existential—by definition Φ_1 and Φ_2 are also existential and hence we have NPTs $\mathcal{N}_1$ and $\mathcal{N}_2$ for the sets of strategy assignments that satisfy Φ_1 and Φ_2, respectively; we construct an NPT $\mathcal{N}$ for the conjunction of $\mathcal{N}_1$ and $\mathcal{N}_2$. Let n_1 and n_2 denote the number of states of $\mathcal{N}_1$ and $\mathcal{N}_2$ and let k_1 and k_2 denote their indices. Let n denote the number of states of $\mathcal{N}$ and k its index. According to Theorem 6, $n = n_1 n_2 \frac{(k_1+k_2)!}{k_1! k_2!}$ and $k = k_1 + k_2$. By Lemma 3 if for some d, m we have $n_1, n_2 \leq exp(d,m)$ and $k_1, k_2 \leq exp(d-1,m)$ and $d > 1$, then $n \leq exp(d, m+1)$ and $k \leq exp(d, m+1)$.

Case 4. $\Phi' = \Phi_1 \wedge \Phi_2$ where Φ' is universal—by definition Φ_1 and Φ_2 are also universal and hence we have UPTs $\mathcal{U}_1$ and $\mathcal{U}_2$ for the sets of strategy assignments that satisfy Φ_1 and Φ_2, respectively; we construct a UPT $\mathcal{U}$ for the conjunction of $\mathcal{U}_1$ and $\mathcal{U}_2$. The number of states of $\mathcal{U}$ is the sum of the number of states of $\mathcal{U}_1$ and $\mathcal{U}_2$ and its index is the maximal index of the two.

Case 5. $\Phi' = \Phi_1 \vee \Phi_2$ where Φ' is existential—by definition Φ_1 and Φ_2 are also existential and hence we have NPTs $\mathcal{N}_1$ and $\mathcal{N}_2$ for the sets of strategy assignments that satisfy Φ_1 and Φ_2, respectively; we construct an NPT $\mathcal{N}$ for the disjunction of $\mathcal{N}_1$ and $\mathcal{N}_2$. The number of states of $\mathcal{N}$ is the sum of the number of states of $\mathcal{N}_1$ and $\mathcal{N}_2$ and its index is the maximal index of the two.

Case 6. $\Phi' = \Phi_1 \vee \Phi_2$ where Φ' is universal—by definition Φ_1 and Φ_2 are also universal and hence we have UPTs $\mathcal{U}_1$ and $\mathcal{U}_2$ for the sets of strategy assignments that satisfy Φ_1 and Φ_2, respectively; we construct a UPT $\mathcal{U}$ for the disjunction of $\mathcal{U}_1$ and $\mathcal{U}_2$. Let n_1 and n_2 denote the number of states of $\mathcal{U}_1$ and $\mathcal{U}_2$ and let k_1 and k_2 denote their indices. Let n denote the number of states of $\mathcal{U}$ and k its index. According to

Theorem 6, $n = n_1 n_2 \frac{(k_1+k_2)!}{k_1!k_2!}$ and $k = k_1 + k_2$. By Lemma 3 if for some d, m we have $n_1, n_2 \leq exp(d, m)$ and $k_1, k_2 \leq exp(d-1, m)$ and $d > 1$, then $n \leq exp(d, m+1)$ and $k \leq exp(d, m+1)$.

Case 7. $\Phi' = \exists z.\Phi_1$ —by definition Φ_1 is existential and there exists an NPT $\mathcal{N}_1$ that accepts the set of strategy assignments that satisfy Φ_1. We construct a DPT $\mathcal{D}$ that checks that the strategy assigned to z is legal (i.e., the choice depicted by the strategy is available). The automaton $\mathcal{D}$ is linear in S and has index 1. Let $\mathcal{N}_2$ denote the conjunction of $\mathcal{N}_1$ and $\mathcal{D}$. Its number of states is the product of the number of states of $\mathcal{N}_1$ and $\mathcal{D}$ and its index is the index of $\mathcal{N}_1$. According to Theorem 2, we can construct an NPT $\mathcal{N}'$ that accepts a tree iff there exists a way to extend the labeling of the tree with a labeling for the strategy for z such that the extended tree is accepted by $\mathcal{N}_2$, i.e., the strategy for z is legal and satisfies Φ_1. The number of states of $\mathcal{N}'$ and its index are equal to those of $\mathcal{N}_2$. If Φ' is existential then $\mathcal{N}'$ is the NPT for Φ'. If Φ' is universal then according to Corollary 1 there exists a UPT $\mathcal{U}$ that accepts the language of $\mathcal{N}'$. Let n and k denote the number of states of $\mathcal{N}'$, then the number of states of $\mathcal{U}$ is $n' = 2^{O(nk\ log\ nk)}$ and its index is $k' = O(nk)$. By Lemma 3 if for some d, m we have $n \leq exp(d, m)$ and $k \leq exp(d-1, m)$ and $d > 1$, then $n' \leq exp(d+1, m+3)$ and $k \leq exp(d, m+1)$.

Case 8. $\Phi' = \forall z.\Phi_1$ —by definition Φ_1 is universal and there exists a UPT $\mathcal{U}_1$ that accepts the set of strategy assignments that satisfy Φ_1. We construct a DPT $\mathcal{D}$ that checks that the strategy assigned to z is legal (i.e., the choice depicted by the strategy is available). Let $\mathcal{D}_1$ denote the DPT that complements $\mathcal{D}$. As before $\mathcal{D}_1$ is linear in S and has index 1. Let $\mathcal{U}_2$ denote the disjunction of $\mathcal{U}_1$ and $\mathcal{D}_1$. Its number of states is the product of the number of states of $\mathcal{U}_1$ and $\mathcal{D}_1$ and its index is the index of $\mathcal{U}_1$. According to Corollary 1, we can construct a UPT $\mathcal{U}'$ that accepts a tree iff all ways to extend the labeling of the tree with a labeling for the strategy for z are accepted by $\mathcal{U}_2$, i.e., either the strategy for z is illegal or it satisfies Φ_1. The number of states of $\mathcal{U}'$ and its index are equal to those of $\mathcal{U}_1$. If Φ' is universal then $\mathcal{U}'$ is the UPT for Φ'. If Φ' is existential then according to Corollary 1 there exists an NPT $\mathcal{N}$ that accepts the language of $\mathcal{U}'$. Let n and k denote the number of states of $\mathcal{U}'$, then the number of states of $\mathcal{N}$ is $n' = 2^{O(nk\ log\ nk)}$ and its index is $k' = O(nk)$. By Lemma 3 if for some d, m we have $n \leq exp(d, m)$ and $k \leq exp(d-1, m)$ and $d > 1$, then $n' \leq exp(d+1, m+3)$ and $k \leq exp(d, m+1)$.

We analyze the size of the resulting automaton. We start with automata created from path formulas that are exponential in the length of the formula and linear in the size of the graph. Quantifier alternations incur an increase in the number of exponents. All other operations maintain the same number of exponents. We note that in cases 3,6,7, and 8 we assume that the automata are large enough in order to use Lemma 3. It follows that many conjunctions of NPTs or many disjunctions of UPTs may cause further explosion in the number of states and index of the resulting automata. However, this is adjusted by the relation between the length of the formula and its quantifier alternation. In order to handle cases 7 and 8 we note that in low levels of the quantifier alternation hierarchy the polynomials in the exponent may have a large degree.

Hence we get either a UPT or an NPT that accepts the set of strategy trees that satisfy the formula. If the formula is closed, it follows that the automaton accepts the tree without actually reading the labeling. Thus, the automaton is non-empty iff it is universal. If the automaton is an NPT we check whether it is nonempty and if the automaton is a UPT we check whether it is universal. If the automaton has $n \leq exp(d+1, m)$ states and index $k \leq exp(d, m)$ then its emptiness / universality can be decided in time $exp(d+1, m)^{exp(d,m)} \leq exp(d+1, m+1)$ for large enough d.

Finally, we note that for unnested path formulas, we start with an automaton of constant size instead of exponential. Thus, the number of exponentials in the size of the formula reduces by one and the required bound follows. The Theorem follows. ∎