2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science

Solving Partial-Information Stochastic Parity Games

Sumit Nain and Moshe Y. Vardi
Department of Computer Science
Rice University
Houston, Texas, 77005
Email: {nain,vardi}@cs.rice.edu

***Abstract*—We study one-sided partial-information 2-player concurrent stochastic games with parity objectives. In such a game, one of the players has only partial visibility of the state of the game, while the other player has complete knowledge. In general, such games are known to be undecidable, even for the case of a single player (POMDP). These undecidability results depend crucially on player strategies that exploit an infinite amount of memory. However, in many applications of games, one is usually more interested in finding a finite-memory strategy. We consider the problem of whether the player with partial information has a finite-memory winning strategy when the player with complete information is allowed to use an arbitrary amount of memory. We show that this problem is decidable.**

***Keywords*-Stochastic games, Partial-observation games, Alternating tree automata, Finite-memory strategy.**

I. Introduction

Two player infinite stochastic games play an important role in many areas of computer science as they provide a natural setting to model nondeterminism and reactivity in the presence of uncertainty or randomness. In particular, infinite games with omega-regular objectives are a fundamental tool in the analysis of many aspects of reactive systems such as modeling, verification, refinement, and synthesis [1], [7], [10]. For example, the standard approach to the synthesis problem for reactive systems reduces the problem to finding the winning strategy of a suitable game [13].

The formalism of infinite games can vary depending on a number of factors such as the number of players, the order in which the players make moves (concurrent or turn-based), the degree of randomness allowed (stochastic or deterministic), the type of objective (reachability, Büchi, or parity), the type of winning condition (deterministic, qualitative probabilistic, or quantitative probabilistic), and the presence or absence of information asymmetry (perfect information, partial information, or one-sided partial-information) [2], [5]. The expressiveness of the formalism varies with these parameters. In this framework, the most general class of games is the class of partial-information 2-player concurrent stochastic games with parity objectives. In this work we focus on an important subclass of partial-information concurrent games: the class of one-sided partial-information 2-player concurrent stochastic parity games.

The most common approach to games assumes a setting with perfect information where both players have complete knowledge of the state of the game. However, in many settings, the assumption of perfect information is not valid and it is natural to allow an information asymmetry between the players. Examples include controllers with noisy sensors and software modules that expose partial interfaces [14].

Since such partial-information games are more general than perfect-information games and also quite useful in their own right, it would be desirable to develop algorithms to solve them. Unfortunately, the addition of partial information substantially increases the difficulty of solving games. While perfect-information stochastic parity games can be solved in NP $\cap$ co-NP [6], allowing partial information makes the problem undecidable even for single player stochastic games (i.e., Partially Observable Markov Decision Process) [12]. Despite the obstacle of undecidability, some positive results have been proved for partial-information games. Typically these results have been obtained for restricted classes of partial-information games such as requiring objectives to be completely visible [4] or restricting objectives to weaker conditions than parity [2], [3].

In contrast, our approach here is to focus on the general case of parity objectives, which suffice to represent all omega-regular objectives. Instead of weakening the expressiveness of the objective, we restrict the partial-information player to using a finite-memory strategy, while allowing the player with complete information to play an unrestricted strategy. In practice, many applications of games depend upon obtaining a winning strategy for the system against an adversarial environment. Often the goal is to obtain a finite-memory strategy and use it some constructive fashion. Thus, if the existence of an unrestricted winning strategy is undecidable, it is natural to ask whether a finite-memory winning strategy exists. Note that in the world of perfect-information games, it is almost always the case that memoryless determinacy holds (that is, there is a winning strategy iff there is a winning strategy that uses no memory).

There are two standard approaches to solving games: the automata-theoretic approach which has strong connections to deterministic games [16], and techniques based on Markov chains that are used to solve stochastic games [6]. We

1043-6871/13 $26.00 © 2013 IEEE
DOI 10.1109/LICS.2013.40

cannot use either directly, because the presence of partial information complicates any potential Markov-chain-based ergodic analysis, while the automata-theoretic method is complicated by the presence of probabilistic transitions. Instead, our approach is a novel application of techniques combined from the two methods.

An outline of our approach: We represent the game as a labeled regular tree (this is simply the unfolding of the finite game graph). We first show that while there is no a priori restriction on the amount of memory used by the player with complete-information, we can safely assume that it uses only a finite amount of memory. Then a pair of opposing player strategies can be embedded into the game tree by labeling the nodes with the player actions generated when the play proceeds according to those strategies. The resulting labeling is regular if both the strategies are finite-memory. This embedding can be viewed as an abstraction in that it discards all the quantitative information about probabilities contained in the strategies.

In the conventional approach to stochastic games, one uses ergodic sets (strongly connected components) of the resulting Markov chain to obtain a combinatorial characterization of the qualitative probabilistic behaviour. One of our key contributions is a method to perform an analogous ergodic analysis directly on labeled trees. This has the dual benefits of avoiding the blowup caused by generating the Markov chain and allowing the use of powerful techniques based on tree automata. Using this result, we characterize winning strategies in terms of reachability of the nodes of the strategy labeled game tree and show that this condition can be checked by a tree automaton. Finally, we show how to ensure that the winning strategies obtained are observation-based by applying techniques from synthesis with incomplete information [8].

The main result of this paper is that one-sided partial-information stochastic parity games are decidable when the partial-information player is restricted to playing finite-memory strategies with no restrictions on the player with complete information. This stands in contrast to the undecidability of the problem for general strategies. One-sided partial-information games are important because they are a natural model for the behavior of modular systems that interact with an environment, where the interactions between the modular components are globally visible but the computation within each component is not visible to other components. For example, synthesis from components can be viewed as such a game [9].

The structure of the rest of the paper is as follows: Section 2 contains the required mathematical background on trees, automata and Markov chains, while in Section 3 we formally describe our model of one-sided partial-information games and state the problem of finding a observation-based pure finite-memory winning strategy in such a game, and in Section 4 we show that the problem is decidable and discuss the complexity of our solution. Finally, in Section 5 we briefly discuss how our solution can easily be extended to the case of randomized finite-memory strategies.

II. Preliminaries

Labeled Trees: Given a set D of directions, a *D-tree* is a set $T \subseteq D^*$ such that (a) there is an element $x_0 \in T$, called the *root* of T, such that, for all $x \in T$ there exists $y \in D^*$ with $x = x_0 \cdot y$, and (b) if $x \cdot c$ is a non-root element of T, where $x \in D^*$ and $c \in D$, then x is also an element of T. The elements of T are called its *nodes*. For every node $x \in T$, the set of *successors* of x is given by $\{x \cdot c \in T : c \in D\}$. A node with no successors is called a *leaf*. A *path* π of a tree T is a set $\pi \subseteq T$ such for every pair of nodes x, y in π, there exists $z \in D^*$ such that $x = y \cdot z$ or $y = x \cdot z$. A path is infinite if it has no leaf nodes, otherwise it is finite. A *subtree* of T is a tree $T' \subseteq T$. For a node $x \in T$, the *subtree rooted at* x, denoted $T(x)$, is the tree $\{x \cdot y \in T : y \in D^*\}$. The *full* D-tree is D^*. The *full subtree* at x is the tree whose set of nodes is $x \cdot D^*$.

Given an alphabet Σ, a *Σ-labeled D-tree* is a pair $\langle T, \tau \rangle$, where T is a tree and $\tau : T \to \Sigma$ maps each node of T to a letter in Σ. A *subtree* of $\langle T, \tau \rangle$, is a Σ-labeled D-tree $\langle T', \tau' \rangle$, where T' is a subtree of T and $\tau'(x) = \tau(x)$, for all $x \in T'$. Given $\Gamma \subseteq \Sigma$, we say a Σ-labeled D-tree $\langle T, \tau \rangle$ is *Γ-recurrent* if T contains no leaf nodes and for each $x \in T$ there exists $y \in D^*$ such that $\tau(x \cdot y) \in \Gamma$ and $x \cdot y \in T$. In a Γ-recurrent tree, from every node there is a path to some Γ-labeled node. As we show below, the size of Γ can be controlled by passing to a suitable subtree.

Lemma 2.1: Let $\langle T, \tau \rangle$ be a Γ-recurrent tree. Then there exists $\gamma \in \Gamma$ and $x \in T$ such that $\langle T(x), \tau \rangle$ is γ-recurrent.

Proof: We first note that if $\langle T, \tau \rangle$ is Γ-recurrent, then for every node $x \in T$, the subtree $\langle T(x), \tau \rangle$ is also Γ-recurrent. We now prove the statement of the lemma by induction on the size of Γ. In the base case, when Γ is a singleton, $\langle T, \tau \rangle$ itself is the required subtree. Let $|\Gamma| > 1$, and let $\gamma' \in \Gamma$ be such that $\langle T, \tau \rangle$ is not γ'-recurrent. Then there exists $y \in T$ such that there is no path from y to a γ'-labeled node and therefore $\langle T(y), \tau \rangle$ only contains labels from $\Gamma' = \Gamma - \{\gamma'\}$. Since $\langle T(y), \tau \rangle$ is Γ-recurrent, it must actually be Γ'-recurrent. Thus, by the induction hypothesis, there must exist a node $z \in T(y)$ and $\gamma \in \Gamma'$ such that $\langle T(z), \tau \rangle$ is γ-recurrent. ■

Transducers: A *transducer* is a deterministic finite automaton with outputs. Formally, a transducer is a tuple $B = \langle \Sigma_I, \Sigma_O, Q, q_0, \delta, \lambda \rangle$, where: Σ_I is a finite input alphabet, Σ_O is a finite output alphabet, Q is a finite set of states, $q_0 \in Q$ is an initial state, $\lambda : Q \to \Sigma_O$ is an output function labeling states with output letters, and $\delta : Q \times \Sigma_I \to Q$ is a transition function. We define $\delta^* : \Sigma_I^* \to Q$ as follows: $\delta^*(\epsilon) = q_0$ and for $x \in \Sigma_I^*$ and $a \in \Sigma_I$, $\delta^*(x \cdot a) = \delta(\delta^*(x), a)$. We denote by $tree(B)$, the Σ_O-labeled Σ_I-tree $\langle \Sigma_I^*, \tau \rangle$, where for all $x \in \Sigma_I^*$, we

have $\tau(x) = \lambda(\delta^*(x))$. We say $tree(B)$ is the *unwinding* of B. A Σ-labeled D-tree T is called *regular*, if there exists a deterministic transducer C such that $T = tree(C)$.

Tree Automata: For a set X, let $\mathcal{B}^+(X)$ be the set of positive Boolean formulas over X (i.e., Boolean formulas built from elements in X using $\wedge$ and $\vee$), including the formulas **True** (an empty conjunction) and **False** (an empty disjunction). For a set $Y \subseteq X$ and a formula $\theta \in \mathcal{B}^+(X)$, we say that Y *satisfies* θ iff assigning **True** to elements in Y and assigning **False** to elements in $X - Y$ makes θ true. An *alternating tree automaton* is tuple $\mathcal{A} = \langle \Sigma, D, Q, q_0, \delta, \beta \rangle$, where Σ is the input alphabet, D is a set of directions, Q is a finite set of states, $q_0 \in Q$ is an initial state, $\delta : Q \times \Sigma \to \mathcal{B}^+(D \times Q)$ is a transition function, and β specifies the acceptance condition that defines a subset of Q^ω. Each element of $\mathcal{B}^+(D \times Q)$ is called an *atom*. The alternating automaton $\mathcal{A}$ runs on Σ-labeled full D-trees. A run of $\mathcal{A}$ over a Σ-labeled D-tree $\langle T, \tau \rangle$ is a $(T \times Q)$-labeled $\mathbb{N}$-tree $\langle T_r, r \rangle$. Each node of T_r corresponds to a node of T. A node in T_r, labeled by (x, q), describes a copy of the automaton that reads the node x of T and visits the state q. Note that multiple nodes of T_r can correspond to the same node of T. The labels of a node and its successors have to satisfy the transition function. Formally, $\langle T_r, r \rangle$ satisfies the following conditions:

1) $\epsilon \in T_r$ and $r(\epsilon) = (\epsilon, q_0)$.
2) Let $y \in T_r$ with $r(y) = (x, q)$ and $\delta(q, \tau(x)) = \theta$. Then there is a set $S = \{(c_0, q_0), (c_1, q_1), \ldots, (c_n, q_n)\} \subseteq D \times Q$ such that S satisfies θ, and for all $0 \leq i \leq n$, we have $y \cdot i \in T_r$ and $r(y \cdot i) = (x \cdot c_i, q_i)$. S is allowed to be empty.

An infinite path π of a run $\langle T_r, r \rangle$ is labeled by a word in Q^ω. Let $inf(\pi)$ be the set of states in Q that occur infinitely often in $r(\pi)$. The ***Büchi*** acceptance condition is given as $\beta \subseteq Q$, and π satisfies β if $inf(\pi) \cap \beta \neq \emptyset$. The *parity* acceptance condition is given as a function $\beta : Q \to \{1, \ldots, k\}$, and π satisfies β if $\min(\{\beta(q) : q \in inf(\pi)\})$ is even. A run $\langle T_r, r \rangle$ is accepting if all its infinite paths satisfy the acceptance condition. An automaton accepts a tree iff there exists a run that accepts it. The language of $\mathcal{A}$, denoted $\mathcal{L}(\mathcal{A})$, is the set of all Σ-labeled D-trees accepted by $\mathcal{A}$.

Theorem 2.2: [11] Let $\mathcal{A}$ be an alternating parity tree automaton. Then $\mathcal{L}(\mathcal{A})$ is non-empty if and only if $\mathcal{A}$ accepts some regular tree. ■

The transition function δ of an alternating tree automaton is *nondeterministic* if every formula produced by δ can be written in disjunctive normal form such that if two atoms (c_1, q_1) and (c_2, q_2) occur in the same conjunction then c_1 and c_2 must be different. A *nondeterministic tree automaton* $\mathcal{A}$ is an alternating tree automaton with a nondeterministic transition function. In this case the transition function returns a set of $|D|$-ary tuples of states and can be represented as a function $\delta : Q \times \Sigma \to 2^{Q^{|D|}}$.

Markov Chains and Ergodic Sets: Given a directed graph $G = (V, E)$, a *strongly connected component* (SCC) of G is a subset U of V, such that for all $u, v \in U$, u is reachable from v. We can define a natural partial order on the set of maximal strongly connected components of G as follows: $U_1 \leq U_2$ if there exists $u_1 \in U_1$ and $u_2 \in U_2$ such that u_1 is reachable from u_2. Then $U \subseteq V$ is an *ergodic set* of G if it is a minimal element of the partial order.

A *Markov chain* is a tuple $M = \langle X, x_0, \mu \rangle$, where X is a finite set of states, x_0 is the initial state and $\mu : X^2 \to [0, 1]$ is a function that assigns transition probabilities to each pair of states such that $\sum_{x' \in X} \mu(x, x') = 1$ for all $x \in X$. A subset Y of X is called an *ergodic set* of M if Y is an ergodic set of the directed graph (X, Z) where $Z = \{(x, x') \in X^2 : \mu(x, x') > 0\}$. A path in M means a path in the graph (X, Z), and a state x of M is *reachable* if there is a path in M from x_0 to x.

III. One-Sided Partial-Information Games

A *one-sided partial-information 2-player concurrent stochastic game* is a tuple $G = \langle S, s_0, A_1, A_2, \delta, O \rangle$, where

- S is a finite set of states.
- s_0 is the starting state of the game.
- A_i is a finite set of actions for player i.
- $\delta : S_i \times A_1 \times A_2 \to Dist(S)$, is the transition function which returns a probability distribution on the set of all states.
- $O \subseteq 2^S$ is a finite set of observations for Player 1 that partition the state space S. This partition defines a function $obs : S \to O$ that maps each state to its observation such that $s \in obs(s)$ for all $s \in S$. If $O = \{\{s\} : s \in S\}$, then all states are completely visible and we omit O from G.

Plays and Observation Sequences: A *play* in G is an infinite sequence of states $\rho = s_0 s_1 \ldots$ such that for all $j \geq 0$, there exists $a_j^1 \in A_1$ and $a_j^2 \in A_2$ with $\delta(s_j, a_j^1, a_j^2)(s_{j+1}) > 0$. The set of plays in G is denoted $Plays(G)$. The *observation sequence* of a play ρ is the unique infinite sequence $obs(\rho) = o_0 o_1 \ldots \in O^\omega$ such that $obs(s_j) = o_j$ for all $j \geq 0$.

Finite-memory Strategy: A randomized strategy in G for Player i is a function $f : S^* \to Dist(A_i)$. A pure strategy in G for Player i is a function $f : S^* \to A_i$. A (pure or randomized) strategy f for Player 1 is *observation-based* if for all $\rho, \rho' \in S^*$, if $obs(\rho) = obs(\rho')$, then $f(\rho) = f(\rho')$. A *pure finite-memory strategy* for Player i, is a strategy f that can be represented as a finite transducer $B_f = (S, A_i, Q, q_0, \Delta, \lambda)$, such that for $\rho \in S^*$, $f(\rho) = \lambda(\Delta^*(\rho))$. f can also be represented by a regular A_i-labeled S-tree $\langle S^*, f \rangle$. Similarly, a pure finite-memory observation-based strategy f can be represented as a finite transducer with input alphabet O and output alphabet A_1, or as a regular A_1-labeled O-tree $\langle O^*, f \rangle$.

Parity Objective: An objective for Player 1 in G is a set $\phi \subseteq S^\omega$ of infinite sequences of states. A play $\rho \in Plays(G)$ satisfies the objective ϕ if $\rho \in \phi$. For $k \in N$, let $\alpha : S \to \{0, 1, \ldots, k\}$ be a priority index function, which maps each state to a nonnegative integer priority. The parity objective ϕ_α requires that the minimum priority that occurs infinitely often in a play should be even. Formally, $\phi_\alpha = \{\rho \in S^\omega : \min\{\alpha(s) | s \in Inf(\rho)\}$ is even$\}$. For priority $p \leq k$, we define the sub-objective $\phi_p = \{\rho \in S^\omega | \min(\{\alpha(s) : s \in inf(\rho)\}) = p\}$. Then the parity objective can be partitioned into simpler sub-objectives as $\phi_\alpha = \phi_0 \cup \phi_2 \cup \cdots \cup \phi_m$ where $m \leq k < m + 2$.

Almost-sure winning: Given strategies f and g for the two players, the probabilities of measurable subsets of S^ω are well defined, and ϕ_α is measurable for a priority index α [15]. For a measurable objective ϕ, we denote by $Pr_G^{f,g}(\phi)$ the probability that ϕ is satisfied by the play obtained from the starting state s_0 when the strategies f and g are used. Given a game G, an observation-based strategy f for Player 1 is *almost-sure winning* for the objective ϕ if for all randomized strategies g for Player 2, we have $Pr_G^{f,g}(\phi) = 1$.

In this paper we focus on 2-player concurrent stochastic games with parity objectives where only one of the players has complete information about the state of the game at all times. We use the convention that Player 1 is the player with partial information and Player 2 has complete information. Our winning condition is almost-sure winning, that is, the play must satisfy the objective with probability 1. Our goal is to decide whether the player with partial information has an observation-based pure finite-memory winning strategy and to find such a strategy if it exists.

Our problem*: Given a one-sided partial-information concurrent stochastic game $G = \langle S, s_0 A_1, A_2, \delta, O\rangle$, with parity objective ϕ_α, is there an observation-based pure finite-memory almost-sure winning strategy for player 1?*

Note that we put no restriction on Player 2, who is allowed to play arbitrary strategies. Since 2-player games are frequently used for the verification of system behaviour in the presence of an adversarial environment, it is quite natural here to not restrict the power of the second player. However, as we show in the next section, it turns out that it is sufficient to consider only pure finite-memory strategies for Player 2.

For the rest of the paper, we use the following convention: *winning* means almost-sure winning; *game* means a one-sided partial-information stochastic game, in particular we use the letter G to represent the game $\langle S, s_0, A_1, A_2, \delta, O\rangle$; α is a priority index function and ϕ_α is the corresponding parity objective.

IV. Solving One-Sided Games

A. Restricting Memory for Player 2

We first show that we can safely restrict Player 2 to a finite amount of memory without losing any expressiveness for Player 1. The key idea is that once Player 1 decides on a finite-memory strategy, the game reduces to a one-player stochastic game (i.e. a Markov decision process) with Player 2 the only player remaining. It is well-known that only memoryless strategies are needed to win in a one-player game [6]. But a memoryless strategy in the one player game can be converted into a finite-memory strategy in the original game. Thus if Player 2 can defeat the particular strategy chosen by Player 1 in the original game, then he can defeat it using finite-memory. Conversely, if a finite-memory Player 1 strategy can defeat every finite-memory Player 2 strategy, then it is a winning strategy.

Proposition 4.1: Player 1 has a pure finite-memory winning strategy f iff for every pure finite-memory player 2 strategy g we have $Pr_G^{f,g}(\phi_\alpha) = 1$.

Proof: Let f be a pure finite-memory player 1 strategy for G. Let $B = (S, A_1, Q, q_0, \Delta, \lambda)$ be the transducer representation of f. Suppose f is not winning for the objective ϕ_α. Then there exists a player 2 strategy g for G such that $Pr_G^{f,g}(\phi_\alpha) < 1$. We consider the complete-information 1-player stochastic game G' obtained by incorporating the choices of f into G. Then $G' = (S \times Q, \emptyset, A_2, \delta')$, where $\delta'((s, q), a)(s', q') = \delta(s, \lambda(q), a)(s')$ if $\Delta(q, s) = q'$ and 0 otherwise. Let $\alpha' : S \times Q \to \{1, \ldots, k\}$ be defined as $\alpha'(s, q) = \alpha(s)$, and let $g' : (S \times Q)^* \to A_2$ be defined as $g'((s_1, q_1) \ldots (s_n, q_n)) = g'(s_1 \ldots s_n)$. Then we have $Pr_{G'}^{g'}(\phi'_\alpha) < 1$. Further, since it is known that pure memoryless strategies are sufficient for complete information 1-player stochastic games [6], therefore there must exist a pure memoryless player 2 strategy $h : S \times Q \to A_2$ for G' such that $Pr_{G'}^{h}(\phi'_\alpha) < 1$. Now consider the transducer $B' = (S, A_2, S \times Q, q_0, \Delta', \lambda')$, where $\Delta'((s, q), s') = (s', \Delta(q, s))$ and $\lambda'((s, q)) = h(s, q)$. Let h' be the pure finite-memory player 2 strategy for G that is represented by B'. Then we have $Pr_G^{f,h'}(\phi_\alpha) < 1$. Therefore if an arbitrary strategy g defeats f in G, then a pure finite-memory strategy h' can also defeat f in G. This implies that to ensure that a given finite-memory player 1 strategy f is winning in G, it is sufficient to check that f wins against all pure finite-memory strategies of player 2. ■

B. Markov Chains and Labeled Trees

For the rest of Section 4, we only consider pure finite-memory strategies for both players. Given a game G, a pair of finite-memory strategies can resolve all the strategic choices in the game, leaving behind a finite-state probabilistic structure. Here we consider two such structures: Markov chains and regular labeled trees. The analysis of stochastic games using Markov chains is well understood. The standard

approach involves taking the product of the game with the finite-state strategies to obtain a larger composite memoryless system that is a Markov chain. By analyzing the ergodic structure of the Markov chain, qualitative questions about probabilities are reduced to problems involving simple graph reachability. While this approach is quite elegant, it suffers from two drawbacks: it involves a potentially large increase in the state space, and more importantly for our purpose, it is not very amenable to solving problems involving partial or incomplete information. In contrast, the automata-theoretic approach deals well with incomplete information [8], but tree automata are rarely used to analyze probability directly. Our aim here is to use the standard Markov-chain-based analysis to develop a similar ergodic treatment for regular labeled trees. We first give the definition for the standard Markov chain construction.

Definition 4.2: Given pure finite-memory strategies f and g for Player 1 and Player 2 respectively, where $B_f = (S, A_1, Q_f, q_0^f, \Delta_f, \lambda_f)$ and $B_g = (S, A_2, Q_g, q_0^g, \Delta_g, \lambda_g)$ are the transducer representations of f and g respectively, the product of G with B_f and B_g is a Markov chain $G \times B_f \times B_g$, defined as follows:

- The set of states is $S \times Q_f \times Q_g$.
- The initial state is (s_0, q_0^f, q_0^g).
- For $s, s' \in S$, $q_1, q_1' \in Q_f$ and $q_2, q_2 \in Q_g$, the probability to transition from (s, q_1, q_2) to (s', q_1', q_2'), is given by $\delta(s, \lambda_f(q_1), \lambda_g(q_2))(s')$ if $\Delta_f(q_1, s) = q_1'$ and $\Delta_g(q_2, s) = q_2'$, and 0 otherwise.

States of this Markov chain inherit priority indices from states of the game. Formally, we extend the priority index function α to the states of the Markov chain as follows: For $(s, q_1, q_2) \in S \times Q_f \times Q_g$, $\alpha(s, q_1, q_2) = \alpha(s)$.

A strategy $f : S^* \to A_i$ for player i can be viewed as a labeling function for the full S-tree S^*. So a pair of strategies can be simply represented as a labeled S-tree, where each node $\rho \in S^*$ is labeled with the actions chosen by the strategies at that node. In addition we find it convenient to also label each node with its corresponding game state. As we will see, this labeled tree is sufficient for qualitative analysis of the game.

Definition 4.3: For a word γ, let $last(\gamma)$ denote the last symbol of γ. We denote by $tree(G, f, g)$, the $(A_1 \times A_2 \times S)$-labeled S-tree $\langle s_0 S^*, \nu \rangle$ with $\nu(\rho) = (f(\rho), g(\rho), last(\rho))$. We also extend the priority index function α to nodes in S^* as follows: For $\rho \in S^*$, $\alpha(\rho) = \alpha(last(\rho))$.

Let x and $y = x \cdot z$ be nodes in $\langle T, \tau \rangle = tree(G, f, g)$. We inductively define a notion of probabilistic reachability. A node is always probabilistically reachable from itself. We say that y is *probabilistically reachable* from x if there exists $s \in S$ such that $\delta(last(x), f(x), g(x))(s) > 0$ and y is probabilistically reachable from $x \cdot s$. We say a node in a tree is *probabilistically reachable* if it is probabilistically reachable from the root.

We point out that the labeled tree representation is more abstract than the product Markov chain representation. It lets us avoid computing the product and the associated blowup. The Markov chain representation and the label tree representation are closely related. We first state without proof some simple properties of this relationship.

Definition 4.4: Let $tree(G, f, g) = \langle T, \tau \rangle$ and $M = G \times B_f \times B_g$. Given a state $x = (s, q_1, q_2) \in M$, we define $x|_S = s$ to be the game-state component of x. Given a finite path $\pi = (s_1, q_1^f, q_1^g) \ldots (s_n, q_n^f, q_n^g)$ in M, we define $play(\pi) = s_1 \ldots s_n \in S^*$.

Lemma 4.5: If π is a finite path in $G \times B_f \times B_g$ starting from the initial state, then $play(\pi)$ is probabilistically reachable in $tree(G, f, g)$. Conversely, if $\rho \in S^*$ is probabilistically reachable in $tree(G, f, g)$, then there is a unique path π from the initial state of $G \times B_f \times B_g$ such that $play(\pi) = \rho$. ■

We define a notion of a *p-ergodic subtree* that allows us to reduce the probabilistic analysis of the game structure to the reachability properties of labeled subtrees. Intuitively, these subtrees can be viewed as the analogue of ergodic sets. They play the same role for labeled trees as ergodic sets do for Markov chains.

Definition 4.6 (p-ergodic subtree): Given a $(A_1 \times A_2 \times S)$-labeled S-tree $\langle T, \tau \rangle$, a node $x \in T$, and $p \in \mathbb{N}$, we say the subtree $\langle T(x), \tau \rangle$ is p-ergodic, if x is probabilistically reachable in T and for all $y \in T(x)$ we have:

- If y is probabilistically reachable from x then $\alpha(y) \geq p$.
- There exists $z \in T(x)$ such that z is probabilistically reachable from y and $\alpha(z) = p$.

We show that p-ergodic subtrees are similar to ergodic sets in terms of their probabilistic behavior. This is one of the main technical results of this paper.

Theorem 4.7: Let f and g be pure finite-memory strategies for Players 1 and 2, respectively. Then the following are equivalent:

1) $Pr_G^{f,g}(\phi_p) > 0$.
2) The Markov chain $G \times B_f \times B_g$ has a reachable ergodic set whose lowest priority is p.
3) $tree(G, f, g)$ has a p-ergodic subtree.

Proof: The equivalence of (1) and (2) is well known and follows directly from the fact that every infinite play of the game must end up in an ergodic set of the Markov chain with probability 1 [15]. We show that (2) and (3) are equivalent. Let $tree(G, f, g) = \langle T, \tau \rangle$ and $M = G \times B_f \times B_g$. Let X be the set of states of M and x_0 be its start state.

(2) ⇒ (3): Suppose M has a reachable ergodic set $Z \subseteq X$ whose lowest priority is p. Then there exists $z \in Z$ such that $\alpha(z) = p$ and z is reachable from x_0. Let π be the shortest path from x_0 to z in M. Let $\rho = play(\pi) \in S^*$. We show that the subtree $\langle T(\rho), \tau \rangle$ is p-ergodic. Since z is reachable in M along π, therefore, by Lemma 4.5, the node

ρ is probabilistically reachable in $tree(G, f, g)$. Let $\rho' \in S^*$ be such that $\rho \cdot \rho'$ is probabilistically reachable from ρ. Then, by Lemma 4.5, there exists a path π' in M from z to some z' such that $play(\pi') = \rho'$ and $z'|_S = last(\rho')$. Since there is a path from z to z', z' is also in the ergodic set Z and $\alpha(z') \geq p$, which implies $\alpha(\rho \cdot \rho') = \alpha(last(\rho')) \geq p$. Thus, every node in $T(\rho)$ that is probabilistically reachable from ρ has priority at least p. Finally, since an ergodic set is connected, there is a path π'' from z' to z in M. Let $\rho'' = play(\pi'')$. Then $\rho \cdot \rho' \cdot \rho''$ is probabilistically reachable from $\rho \cdot \rho'$ and $\alpha(\rho \cdot \rho' \cdot \rho'') = \alpha(last(\rho'')) = \alpha(z) = p$. Therefore all the conditions in Definition 4.6 are satisfied and so $\langle T(\rho), \tau \rangle$ is a p-ergodic subtree of $tree(G, f, g)$.

(3) $\Rightarrow$ (2): Suppose the subtree $\langle T(\rho), \tau \rangle$ of $tree(G, f, g)$ is a p-ergodic subtree. Let $S_p = \{s \in S : \alpha(s) = p\}$. Then $\langle T(\rho), \tau \rangle$ is also S_p-recurrent. By Lemma 2.1, there exists $s \in S_p$ and $\rho \in T(\rho)$, such that ρ' is probabilistically reachable from ρ and $\langle T(\rho'), \tau \rangle$ is s-recurrent, that is, there is a path from every node in $\langle T(\rho'), \tau \rangle$ to a node with label s. Since every subtree of a s-recurrent tree is also s-recurrent, we can assume without loss of generality that $last(\rho') = s$ and $\alpha(\rho') = p$. Consider the path π in M that corresponds to ρ'. Let π end in state z of M. Then $\alpha(z) = p$. Let $Z \subseteq X$ be the set of all the states of M that are reachable from z. If $z' \in Z$ and π' is a path from z to z', then $\rho'' = play(\pi')$ is such that $\rho' \cdot \rho''$ is probabilistically reachable from ρ'. Since $T(\rho)$ is p-ergodic, and $\rho'' \in T(\rho') \subseteq T(\rho)$, we have $p \leq \alpha(\rho' \cdot \rho'') = \alpha(last(\rho'')) = \alpha(z')$. Thus, all states in Z have priority at least p. Also, since $\langle T(\rho'), \tau \rangle$ is s-recurrent, there must exist $\rho''' \in S^*$ such that $\rho' \cdot \rho'' \cdot \rho'''$ is probabilistically reachable from $\rho' \cdot \rho''$ and $\rho' \cdot \rho'' \cdot \rho'''$ is labeled with s. Let π'' be the path in M that corresponds to ρ'''. Then π'' is a path from z' to some state z'' with $\alpha(z'') = p$. Since reachability is closed under transitivity, therefore from every state in Z, there is a path to a state with priority p. Note that it is not necessary for Z to itself be an ergodic set. We show however, that Z contains an ergodic set with the desired property. In a directed graph, by the very definition of ergodic sets, from every vertex there is a path to some ergodic set. So Z contains at least one ergodic set Y of M. Since Y is a subset of Z, every state in Y has priority at least p. Suppose that no state in Y has priority exactly p. But since there is no path that leaves an ergodic set, this would then contradict the fact that from every state in Z we can reach a state with priority p. Therefore at least one state in Y must have priority p. This proves that M has a reachable ergodic set with lowest priority p. ∎

C. Solving for Finite-memory Strategies

The most important feature of the definition of p-ergodic subtree is that it is defined in terms of local reachability properties that can be checked by a nondeterministic Büchi tree automaton. We next build such a tree automaton.

Lemma 4.8: There exists a nondeterministic Büchi tree automaton (NBT) $\mathcal{A}_p$ that accepts a $(A_1 \times A_2 \times S)$-labeled S-tree T iff T has a p-ergodic subtree.

Proof: For clarity of exposition, we assume that $S = \{0, 1\}$ and define the automaton over binary trees, but the definition can be easily extended to n-ary trees.

We define $\mathcal{A}_p = (A_1 \times A_2 \times S, Q, q_0, \delta, \beta)$, where $Q = \{\mathsf{search}, \mathsf{cut}, \mathsf{wait}, \mathsf{look}, \mathsf{visit}, \mathsf{err}\}$, $q_0 = \mathsf{search}$, and $\beta = \{\mathsf{visit}, \mathsf{wait}, \mathsf{cut}\}$. The states of the automaton can then be described as follows:

- search: In this state the automaton is searching for the root of the special subtree.
- cut: This represents a branch not taken.
- wait and look: In these states the automaton has entered the subtree and is looking for nodes with priority p.
- visit: In this state the automaton has just visited a node with priority p in the subtree.
- err: This is an error state that is entered if there is a priority lower than p in the subtree.

Given a label $\nu = (a, b, s) \in A_1 \times A_2 \times S$, we define $X_\nu = \{s' \in S : \delta(s, a, b)(s') > 0\}$. Then the transition function δ is defined as follows:

1) For $q \in \{\mathsf{cut}, \mathsf{err}\}$, $\delta(q, \nu) = \{q, q\}$.
2) For $q = \mathsf{search}$

$$\delta(q, \nu) = \begin{cases} \{(\mathsf{search}, \mathsf{cut}), (\mathsf{wait}, \mathsf{cut})\} & \text{if } X_\nu = \{0\} \\ \{(\mathsf{cut}, \mathsf{search}), (\mathsf{cut}, \mathsf{wait})\} & \text{if } X_\nu = \{1\} \\ \{(\mathsf{search}, \mathsf{cut}), (\mathsf{cut}, \mathsf{search}), \\ \quad (\mathsf{wait}, \mathsf{cut}), (\mathsf{cut}, \mathsf{wait})\} & \text{if } X_\nu = \{0, 1\} \end{cases}$$

3) For $q \in \{\mathsf{wait}, \mathsf{look}, \mathsf{visit}\}$, if $\alpha(s) < p$ then $\delta(q, \nu) = \{\mathsf{err}, \mathsf{err}\}$, if $\alpha(s) = p$ then

$$\delta(q, \nu) = \begin{cases} \{(\mathsf{visit}, \mathsf{cut})\} & \text{if } X_\nu = \{0\} \\ \{(\mathsf{cut}, \mathsf{visit})\} & \text{if } X_\nu = \{1\} \\ \{(\mathsf{visit}, \mathsf{visit})\} & \text{if } X_\nu = \{0, 1\} \end{cases}$$

and if $\alpha(s) > p$ then

$$\delta(q, \nu) = \begin{cases} \{(\mathsf{look}, \mathsf{cut})\} & \text{if } X_\nu = \{0\} \\ \{(\mathsf{cut}, \mathsf{look})\} & \text{if } X_\nu = \{1\} \\ \{(\mathsf{look}, \mathsf{wait}), (\mathsf{wait}, \mathsf{look})\} & \text{if } X_\nu = \{0, 1\} \end{cases}$$

In the first stage, $\mathcal{A}_p$ guesses the location of the root of the special subtree T. While searching for this root, $\mathcal{A}_p$ remains in the state search. When it encounters the root of T, it enters the state wait for the first time. This starts the second stage, where $\mathcal{A}_p$ considers only probabilistically reachable nodes in T. In directions that correspond to a node that is not probabilistically reachable in T, $\mathcal{A}_p$ moves to the state cut and remains there perpetually. From every probabilistically reachable node in T, $\mathcal{A}_p$ guesses a path to another probabilistically reachable node with label p, using the states wait and look. It starts this search in state

wait, moves to state look immediately, remains there until it encounters a probabilistically reachable node with label p, and then moves to state visit. If there is no path from some node to another node with label p, all runs corresponding to the choice of T as subtree will eventually get stuck in look. Thus, some run corresponding to T as the required subtree is accepting iff T satisfies the required conditions. ■

We now transform $\mathcal{A}_p$ to obtain an automaton $\mathcal{A}$ that accepts trees corresponding to winning strategies for Player 1. While $\mathcal{A}_p$ accepts $(A_1 \times A_2 \times S)$-labeled S-trees, $\mathcal{A}$ accepts A_1-labeled S-trees. The transformation proceeds in steps: first we build $\mathcal{A}'_p$ which nondeterministically guesses the moves of Player 2, then get rid of the S labels by enlarging the state space to obtain $\mathcal{A}''_p$, and next we unify multiple $\mathcal{A}''_p$ automata corresponding to different sub-objectives to obtain an automaton $\mathcal{A}'$ for the parity objective. Note that $\mathcal{A}_p$ accepts trees corresponding to strategy pairs where Player 2 beats Player 1, so $\mathcal{A}'$ accepts all losing Player 1 strategies. So the required automaton $\mathcal{A}$ is obtained by complementing $\mathcal{A}'$. Formally:

Theorem 4.9: There exists a nondeterministic parity tree automaton (NPT) $\mathcal{A}$ such that $\mathcal{A}$ accepts the labeled tree $\langle S^*, f\rangle$ iff f is a winning strategy for Player 1.

Proof: Given two labelings $u : S^* \to X$, $v : S^* \to Y$, we define the product labeling $u \times v : S^* \to X \times Y$ as $u \times v(x) = (u(x), v(x))$ for all $x \in S^*$.

Let $\mathcal{A}_p = (A_1 \times A_2 \times S, Q, q_0, \delta, \beta)$ be the NBT defined in Lemma 4.8. We define $\mathcal{A}'_p = (A_1 \times S, Q, q_0, \delta', \beta)$, where

$$\delta'(q, (a, s)) = \bigvee_{a' \in A_2} \delta(q, (a, a', s))$$

Thus $\mathcal{A}'_p$ takes player 1 strategy as input, guesses the strategy of player 2 and checks the resulting labeled tree for the presence of a p-ergodic subtree by simulating $\mathcal{A}_p$.

Note that $\mathcal{A}'_p$ expects $(A_1 \times S)$-labeled S-trees as input. In order to get rid of the S portion of the labeling, we move it into the state space as follows: we define $\mathcal{A}''_p = (A_1, Q \times S, (q_0, s_0), \delta'', \beta \times S)$, where for every $q \in Q$, $s \in S$ and $a \in A_1$, the transition $\delta''((q, s), a)$ is obtained from $\delta(q, (a, s))$ by replacing each atom (s', q') by the atom $(s', (q', s'))$. Then $\mathcal{A}''_p$ accepts A_1-labeled S-trees. Note that if $\mathcal{A}_p$ is an NBT then so are $\mathcal{A}'_p$ and $\mathcal{A}''_p$.

Let f be a (not necessarily finite-memory) strategy for Player 1. Let $\nu : S^* \to S$ be the labeling $\nu(x) = last(x)$. If there exists a strategy g for Player 2 such that $Pr_G^{f,g}(\phi_p) > 0$, then, by Theorem 4.7 and Lemma 4.8, $\mathcal{A}_p$ accepts $\langle S^*, f \times g \times \nu\rangle$. Then, by construction, $\mathcal{A}'_p$ accepts $\langle S^*, f \times \nu\rangle$ and A''_p accepts $\langle S^*, f\rangle$. Alternately, if for all player 2 strategies g, we have $Pr_G^{f,g}(\phi_p) = 0$, then, by Lemma 4.8, $\langle S^*, f \times g \times \nu\rangle$ is not accepted by $\mathcal{A}_p$ for all possible labelings g. Consequently, $\langle S^*, f \times \nu\rangle$ is not accepted by $\mathcal{A}'_p$, so A''_p will not accept $\langle S^*, f\rangle$. Thus, $\mathcal{A}''_p$ accepts an A_1-labeled S-tree $\langle S^*, f\rangle$ iff there exists some player 2 strategy g, such that $Pr_G^{f,g}(\phi_p) > 0$.

Consider the automaton $\mathcal{A}'$ whose language is the union of the language of each $\mathcal{A}''_p$, for all odd $p \leq \max(\alpha)$. Let f be a strategy for Player 1. Then $\mathcal{A}'$ accepts an A_1-labeled S-tree $\langle S^*, f\rangle$ iff there exists a player 2 strategy g such that $Pr_G^{f,g}(\phi_k) > 0$ for some odd $k \leq \max(\alpha)$. The latter condition is equivalent to $Pr_G^{f,g}(\phi_\alpha) < 1$. Therefore, $\mathcal{A}'$ accepts $\langle S^*, f\rangle$ iff f is not winning.

Finally, consider the NPT $\mathcal{A} = \overline{\mathcal{A}'}$, which is the complement of $\mathcal{A}'$. Then $\mathcal{A}$ accepts $\langle S^*, f\rangle$ iff f is a winning strategy for Player 1. ■

D. Solving for Observation-based Strategies

Note that the finite-memory strategy found by the method of Theorem reftheorem:non-observation is not required to be observation-based. In fact, we have not used the concept of partial information in the technical development so far. Nevertheless, the automata-based method we have developed is robust in the presence of partial information, and with one further transformation we can obtain an automaton that accepts only observation-based winning strategies.

While a general strategy is a A_1-labeled S-tree, an observation-based strategy is best viewed as a A_1-labeled O-tree. We already have an automaton $\mathcal{A}$ that accepts A_1-labeled S-trees corresponding to winning finite-memory strategies. We now develop an automaton $\mathcal{B}$ that accepts A_1-labeled O-trees corresponding to the trees accepted by $\mathcal{A}$. We use a variant of a technique first developed for synthesis from incomplete information [8].

Theorem 4.10: There exists an alternating parity tree automaton (APT) $\mathcal{B}$ such that $\mathcal{B}$ is non-empty iff Player 1 has an observation-based finite-memory winning strategy for ϕ_α.

Proof: Let $\mathcal{A} = (A_1, Q, q_0, \delta, \beta)$ be the NPT defined in Theorem 4.9. Then $\mathcal{A}$ accepts A_1-labeled S-trees. Let $\mathcal{B} = (A_1, Q, q_0, \delta', \beta)$ where for every $q \in Q$ and $a \in A_1$, the transition $\delta'(q, a)$ is obtained from $\delta(q, a)$ by replacing each atom (s, q') by the atom $(obs(s), q')$. Then $\mathcal{B}$ accepts A_1-labeled O-trees. Let $f : O^* \to A_1$ be an observation-based strategy for Player 1. We define a general Player 1 strategy $g : S^* \to A_1$ as follows: for $\rho \in S^*$, $g(\rho) = f(obs(\rho))$ (recall that obs maps each game state in a play to its corresponding observation). Then $\mathcal{B}$ accepts $\langle O^*, f\rangle$ iff $\mathcal{A}$ accepts $\langle S^*, g\rangle$. Also, from the perspective of the opponent, f and g are exactly the same strategy, so f is a winning strategy iff g is a winning strategy. Assume first f be an observation-based finite-memory winning strategy for Player 1. Then g is also wining strategy for Player 1. By Theorem 4.9, $\mathcal{A}$ accepts $\langle S^*, g\rangle$, so it follows that $\mathcal{B}$ accepts $\langle O^*, f\rangle$. Assume now that $\mathcal{B}$ accepts $\langle O^*, f\rangle$. By Theorem , we can assume that f is a finite-memory strategy. Since we also have that $\mathcal{A}$ accepts $\langle S^*, g\rangle$, by Theorem 4.9, g is a winning strategy, which implies that f is also a winning strategy. So $\mathcal{B}$ is the required automaton. ■

E. Computational Complexity

The NBT $\mathcal{A}'_p$ accepts $|S|$-ary trees and has $O(1)$ states with an alphabet of size $O(|A_1| \cdot |S|)$, so the NBT $\mathcal{A}''_p$ accepts $|S|$-ary trees and has $O(|S|)$ states with an alphabet of size $O(|A_1|)$. Then $\mathcal{A}'$ is an NBT with $O(k|S|)$ where $k = \max(\alpha)$. Therefore $\mathcal{A}$, which is obtained by complementing $\mathcal{A}'$, is an NPT with $2^{Poly(|S|)}$ states and parity index $O(k|S|)$. The translation from $\mathcal{A}$ to $\mathcal{B}$ adds no blowup, but $\mathcal{B}$ is an APT, while $\mathcal{A}$ is an NPT. Since emptiness for an alternating parity tree automaton can be checked in time exponential in the size of the automaton [11], therefore $\mathcal{B}$ can be checked for emptiness in time doubly exponential in the size of the game G.

Theorem 4.11: The complexity of finding if the partial-information player has a finite-memory winning strategy in a one-sided partial-information stochastic parity game is at most 2EXPTIME. ■

V. Discussion and Future Work

Randomized Finite-memory Strategies: While our focus in this work is on pure finite-memory strategies for the partial-information player, it is straightforward to extend our methods and results to the case of mixed or randomized finite-memory strategies. If Player 1 uses a randomized finite-memory strategy, this would not increase the state space of the resulting Markov chain but instead would increase the number of transitions with positive probability as a randomized strategy would activate multiple actions for Player 1 in each game state. This could be represented on a labeled tree by letting the labels for Player 1 range over sets of actions instead of single actions, that is, we would have to consider a $(2^{A_1} \times A_2 \times S)$-labeled tree. Thus the situation is essentially equivalent to restricting Player 1 to a pure finite-memory strategy but allowing an exponentially larger set of actions instead. Solving the equivalent game using our method would still take time doubly exponential in the state space, so allowing randomized finite-memory strategies does not change the upper bound on time complexity.

Future Work: We have shown that the problem of finding a finite-memory winning strategy for the partial-information player in a one-sided partial-information concurrent game is in 2EXPTIME. The question of tighter lower and upper bounds we leave for the future. It would also be interesting to consider the problem from the perspective of the stronger player; that is, whether we can find a finite-memory winning strategy for the player with complete information (Player 2), if it exists, is an open question.

Acknowledgment

This work was supported in part by NSF grants CNS 1049862 and CCF-1139011, by NSF Expeditions in Computing project "ExCAPE: Expeditions in Computer Augmented Program Engineering", by BSF grant 9800096, and by gift from Intel.

References

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

[2] K. Chatterjee and L. Doyen. The complexity of partial-observation parity games. In *Proc. LPAR'10*, LNCS 6397. Springer, 2010.

[3] K. Chatterjee, L. Doyen, T.A. Henzinger, and J.F. Raskin. Algorithms for omega-regular games of incomplete information. *Logical Methods in Computer Science*, 3(3:4), 2007.

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

[5] K. Chatterjee and T.A. Henzinger. A survey of stochastic ω-regular games. *J. Computer and System Sciences*, 78(2): 394-413, 2012.

[6] K. Chatterjee, M. Jurdzinski, and T. A. Henzinger. Simple stochastic parity games. In *Proc. CSL'03*, LNCS 2803, pages 100–113. Springer, 2003.

[7] L. de Alfaro and T.A. Henzinger. Interface-based design. In *Engineering Theories of Software-intensive Systems*, NATO Science Series: Mathematics, Physics, and Chemistry 195, pages 83–104. Springer, 2005.

[8] O. Kupferman and M.Y. Vardi. Synthesis with incomplete informatio. In *2nd Int. Conf. on Temporal Logic*, pages 91–106. Kluwer, 1997.

[9] Y. Lustig, S. Nain, and M.Y. Vardi. Synthesis from Probabilistic Components. In *Proc. CSL'11*, LIPIcs 12, 2011.

[10] R. McNaughton. Infinite games played on finite graphs. *Annals of Pure and Applied Logic*, 65:149184, 1993.

[11] D.E. Muller and P.E. Schupp. Simulating alternating tree automata by nondeterministic automata: New results and new proofs of theorems of Rabin, McNaughton and Safra. *Theoretical Computer Science*, 141:69–107, 1995.

[12] A. Paz. Introduction to probabilistic automata. Academic Press, 1971.

[13] 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, 1989.

[14] J.H. Reif. The complexity of two-player games of incomplete information. *J. Computer and System Sciences*, 29:274301, 1984.

[15] M.Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *Proc. FOCS'85*, pages 327–338. IEEE, 1985.

[16] T. Wilke. Alternating Tree Automata, Parity Games, and Modal μ-Calculus. *Bulletin of the Belgian Mathematical Society*, Vol 8(2), 2001.