The Complexity of Synthesis from Probabilistic Components[*]

Krishnendu Chatterjee[†] Laurent Doyen[§] Moshe Y. Vardi[‡]
[†] IST Austria
[§] CNRS, LSV, ENS Cachan
[‡] Rice University, USA

Abstract

The synthesis problem asks for the automatic construction of a system from its specification. In the traditional setting, the system is "constructed from scratch" rather than composed from reusable components. However, this is rare in practice, and almost every non-trivial software system relies heavily on the use of libraries of reusable components. Recently, Lustig and Vardi introduced *dataflow* and *controlflow* synthesis from libraries of reusable components. They proved that dataflow synthesis is undecidable, while controlflow synthesis is decidable. The problem of controlflow synthesis from libraries of *probabilistic components* was considered by Nain, Lustig and Vardi, and was shown to be decidable for qualitative analysis (that asks that the specification be satisfied with probability 1). Our main contributions for controlflow synthesis from probabilistic components are to establish better complexity bounds for the qualitative analysis problem, and to show that the more general quantitative problem is undecidable. For the qualitative analysis, we show that the problem (i) is EXPTIME-complete when the specification is given as a deterministic parity word automaton, improving the previously known 2EXPTIME upper bound; and (ii) belongs to UP $\cap$ coUP and is parity-games hard, when the specification is given directly as a parity condition on the components, improving the previously known EXPTIME upper bound.

1 Introduction

Synthesis from existing components. Reactive systems (hardware or software) are rarely built from scratch, but are mostly developed based on existing components. A component might be used in the design of multiple systems, e.g., function libraries, web APIs, and ASICs. The construction of systems from existing reusable components is an active research direction, with several important works, such as component-based construction [24], "interface-based design" [17], web-service orchestration [5]. The synthesis problem asks for the automated construction of a system given a logical specification. For example, in LTL (linear-time temporal logic) synthesis, the specification is given in LTL and the reactive system to be constructed is a finite-state transducer [23]. In the traditional LTL synthesis setting, the system is "constructed from scratch" rather than "composed" from existing components. Recently, Lustig and Vardi introduced the study of synthesis from reusable or existing components [20].

The model and types of composition. The precise mathematical model for the components and their composition is an important concern (and we refer the reader to [20, 21] for a detailed discussion). As a basic model for a component, following [20], we abstract away the precise details of the component and model a component as a *transducer*, i.e., a finite-state machine with outputs. Transducers constitute a canonical model for reactive components, abstracting away internal architecture and focusing on modeling input/output behavior. In [20], two models of composition were studied, namely, *dataflow* composition, where the output of one component

[*] This research was supported by Austrian Science Fund (FWF) Grant No P23499- N23, FWF NFN Grant No S11407-N23 (RiSE), ERC Start grant (279307: Graph Games), Microsoft Faculty Fellowship Award, 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.

becomes an input to another component, and *controlflow* composition, where at every point of time the control resides within a single component. The synthesis problem for dataflow composition was shown to be undecidable, whereas the controlflow composition was shown to be decidable [20].

Synthesis for probabilistic components. While [20] considered synthesis for non-probabilistic components, the study of synthesis for controlflow composition for probabilistic components was considered in [21]. Probabilistic components are transducers with a probabilistic transition function, that corresponds to modeling systems where there is probabilistic uncertainty about the effect of input actions. Thus the controlflow composition for probabilistic transducers aims at construction of reliable systems from unreliable components. There is a rich literature about verification and analysis of such systems, cf. [25, 15, 16, 26, 4, 19], as well as about synthesis in the presence of probabilistic uncertainty [3].

Qualitative and quantitative analysis. There are two probabilistic notions of correctness, namely, the *qualitative* criterion that requires the satisfaction of the specification with probability 1, and the more general *quantitative* criterion that requires the satisfaction of the specification with probability at least η, given $0 < \eta \leq 1$.

The synthesis questions and previous results. In the synthesis problem for controlflow composition, the input is a library $\mathcal{L}$ of probabilistic components, and we consider specifications given as parity conditions (that allow us to consider all ω-regular properties, which can express all commonly used specifications in verification). The *qualitative (resp., quantitative) realizability* and synthesis problems ask whether there exists a *finite* system S built from the components in $\mathcal{L}$, such that, regardless of the input provided by the external environment, the traces generated by the system S satisfy the specification with probability 1 (resp., probability at least η). Each component in the library can be instantiated an arbitrary number of times in the construction and there is no a-priori bound on the size of the system obtained. The way the specification is provided gives rise to two different problems: (i) *embedded parity realizability*, where the specification is given in the form of a parity index on the states of the components; and (ii) *DPW realizability*, where the specification is given as a separate deterministic parity word automaton (DPW). The results of [21] established the decidability of the qualitative realizability problem, namely, in EXPTIME for the embedded parity realizability problem and 2EXPTIME for the DPW realizability problem. The exact complexity of the qualitative problem and the decidability and complexity of the quantitative problem were left open, which we study in this work.

	Qualitative		Quantitative	
	Our Results	Previous Results	Our Results	Previous Results
Embedded Parity (with exit control)	UP ∩ coUP (Parity-games hard)	EXPTIME	UP ∩ coUP (Parity-games hard)	Open
Embedded Parity (unrestricted exit control)	PTIME	Not considered	PTIME	Not considered
DPW Specifications	EXPTIME-c	2EXPTIME	Undecidable	Open

Table 1: Computational complexity of synthesis from probabilistic components.

Our contributions. Our main contributions are as follows (summarized in Table 1).

1. We show that both the qualitative and quantitative realizability problems for embedded parity lie in UP ∩ coUP, and even the qualitative problem is at least parity-games hard (the parity-games problem also belongs to UP ∩ coUP [18], and the existence of a polynomial-time algorithm is a major and long-standing open problem). Moreover, we show that a special case of the quantitative embedded parity problem (namely, unrestricted exit control) can be solved in polynomial time: a probabilistic component has a set of exits, and in general there is a constraint on the current exit and the next component to transfer the control, and in the unrestricted exit control problem no such constraint is imposed.

2. We show that the qualitative realizability problem for DPW specifications is EXPTIME-complete (an exponential improvement over the previous 2EXPTIME result). Finally, we show that the quantitative realizability problem for DPW specifications is undecidable.

Technical contributions. Our two main technical contributions are as follows. First, for the realizability of embedded parity specifications, while the most natural interpretation of the problem is as a partial-observation stochastic game (as also considered in [21]), we show that the problem can be reduced in polynomial time to a perfect-information stochastic game. Second, for the realizability of DPW specifications, we consider partial-observation stochastic games where the strategies correspond to a correct composition that defines, given an exit state of a component, to which component the control should be transferred. Since we aim at a finite-state system, we need to consider strategies with *finite memory*, and since the control flow is deterministic, we need to consider *pure* (non-randomized) strategies. Moreover, since the composition must be independent of the internal executions of the components, we need to consider strategies with *stuttering* invariance. For example, we consider stutter-invariant strategies that must play the same when observations are repeated, and collapsed stutter-invariant strategies that are stutter-invariant strategies but not allowed to observe the length of the repetitions. We present polynomial-time reductions for both stutter-invariant and collapsed stutter-invariant strategies to games with standard observation-based strategies. Our results establish optimal complexity results for qualitative analysis of partial-observation stochastic games with finite-memory stutter-invariant and collapsed stutter-invariant strategies, which are of independent interest. Finally, we present a polynomial reduction of the qualitative realizability for DPW specifications to partial-observation stochastic games with collapsed stutter-invariant strategies and obtain the EXPTIME-complete result.

2 Definitions

Probability distributions. A probability distribution on a finite set X is a function $f : X \to [0,1]$ such that $\sum_{x \in X} f(x) = 1$. We use $\mathcal{D}(X)$ to denote the set of all probability distributions on set X.

2.1 Transducers

In this section we present the definitions of deterministic and probabilistic transducers, and strategies for them.

Deterministic transducers. A *deterministic transducer* is a tuple $B = \langle \Sigma_I, \Sigma_O, Q, q_0, \delta, L \rangle$, 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, $L : Q \to \Sigma_O$ 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 all $x \in \Sigma_I^*$ and $a \in \Sigma_I$, we have $\delta^*(x \cdot a) = \delta(\delta^*(x), a)$.

Probabilistic transducers. A *probabilistic transducer*, is a tuple $\mathcal{T} = \langle \Sigma_I, \Sigma_O, Q, q_0, \delta, F, L \rangle$, 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, $\delta : (Q \setminus F) \times \Sigma_I \to \mathcal{D}(Q)$ is a probabilistic transition function, $F \subseteq Q$ is a set of exit states, and $L : Q \to \Sigma_O$ is an output function labeling states with output letters. Note that there are no transitions out of an exit state. If F is empty, we say $\mathcal{T}$ is a probabilistic transducer without exits. Note that deterministic transducers can be viewed as a special case of probabilistic transducers.

Strategies for transducers. Given a probabilistic transducer $M = \langle \Sigma_I, \Sigma_O, Q, q_0, \delta, F, L \rangle$, a *strategy* for M is a function $f : Q^+ \to \mathcal{D}(\Sigma_I)$ that probabilistically chooses an input for each finite sequence of states. We denote by $\mathcal{F}$ the set of all strategies. A strategy is memoryless if the choice depends only on the last state in the sequence. A memoryless strategy can be written as a function $g : Q \to \mathcal{D}(\Sigma_I)$. A strategy is *pure* if the choice is deterministic. A pure strategy is a function $h : Q^+ \to \Sigma_I$, and a memoryless and pure strategy is a function $h : Q \to \Sigma_I$.

Probability measure. A strategy f along with a probabilistic transducer M, with set of states Q, induces a probability distribution on Q^ω, denoted μ_f. By standard measure-theoretic arguments, it suffices to define μ_f for the cylinders of Q^ω, which are sets of the form $\beta \cdot Q^\omega$, where $\beta \in Q^*$. First we extend δ to exit states as follows:

for $a \in \Sigma_I$ and $q \in F$, $q' \in Q$, let $\delta(q,a)(q) = 1$ and $\delta(q,a)(q') = 0$ if $q' \neq q$. Then we define $\mu_f(q_0 \cdot Q^\omega) = 1$, and for $\beta \in Q^*$, $q, q' \in Q$, we have $\mu_f(\beta q q' \cdot Q^\omega) = \mu_f(\beta q) \cdot \sum_{a \in \Sigma_I}(f(\beta q)(a) \cdot \delta(q,a)(q'))$. These conditions say that there is a unique start state, and the probability of visiting a state q', after visiting βq, is the same as the probability of the strategy picking a particular letter multiplied by the probability that the transducer transitions from q to q' on that input letter, summed over all input letters.

2.2 Library of Components A *library* is a set of probabilistic transducers that share the same input and output alphabets. Each transducer in the library is called a *component type*. Given a finite set of directions D, we say a library $\mathcal{L}$ has width D, if each component type in the library has exactly $|D|$ exit states. Since we can always add dummy unreachable exit states to any component, we assume, w.l.o.g., that all libraries have an associated width, usually denoted D. In the context of a particular component type, we often refer to elements of D as exits, and subsets of D as sets of exits.

2.3 Controlflow Composition from Libraries We first informally describe the notion of controlflow composition of components from a library as defined in [21]. The components in the composition take turns interacting with the environment, and at each point in time, exactly one component is active. When the active component reaches an exit state, control is transferred to some other component. Thus, to define a controlflow composition, it suffices to name the components used and describe how control should be transferred between them. We use a deterministic transducer to define the transfer of control. Each library component can be used multiple times in a composition, and we treat these occurrences as distinct *component instances*. We emphasize that the composition can contain potentially arbitrarily many instances of each component type inside it. Thus, the size of the composition, a priori, is not bounded. Note that our notion of composition is *static*, where the components called are determined before run time, rather than *dynamic*, where the components called are determined during run time.

Let $\mathcal{L}$ be a library of width D. A *composer* over $\mathcal{L}$ is a deterministic transducer $C = \langle D, \mathcal{L}, \mathcal{M}, \mathsf{M}_0, \Delta, \lambda\rangle$. Here $\mathcal{M}$ is an arbitrary finite set of states. There is no bound on the size of $\mathcal{M}$. Each $\mathsf{M}_i \in \mathcal{M}$ is a component from $\mathcal{L}$ and $\lambda(\mathsf{M}_i) \in \mathcal{L}$ is the type of M_i. We use the following notational convention for component instances and names: the upright letter M always denotes component names (i.e., states of a composer) and the italicized letter M always denotes the corresponding component instances (i.e., elements of $\mathcal{L}$). Further, for notational convenience we often write M_i directly instead of $\lambda(\mathsf{M}_i)$. Note that while each M_i is distinct, the corresponding components M_i need not be distinct. Each composer defines a unique composition over components from $\mathcal{L}$. The current state of the composer corresponds to the component that is in control. The transition function Δ describes how to transfer control between components: $\Delta(\mathsf{M}, i) = \mathsf{M}'$ denotes that when the composition is in the ith final state of component M it moves to the start state of component M'. A composer can be viewed as an implicit representation of a composition. We give an explicit definition of composition below.

DEFINITION 1. (CONTROLFLOW COMPOSITION) *Let $C = \langle D, \mathcal{L}, \mathcal{M}, \mathsf{M}_0, \Delta, \lambda\rangle$ be a composer over library $\mathcal{L}$ of width D, where $\mathcal{M} = \{\mathsf{M}_0, \ldots, \mathsf{M}_n\}$, $\lambda(\mathsf{M}_i) = \langle \Sigma_I, \Sigma_O, Q_i, q_0^i, \delta_i, F_i, L_i\rangle$ and $F_i = \{q_x^i : x \in D\}$. The composition defined by C, denoted $\mathcal{T}_C$, is a probabilistic transducer $\langle \Sigma_I, \Sigma_O, Q, q_0, \delta, \emptyset, L\rangle$, where $Q = \bigcup_{i=0}^n (Q_i \times \{i\})$, $q_0 = \langle q_0^0, 0\rangle$, $L(\langle q, i\rangle) = L_i(q)$, and the transition function δ is defined as follows: For $\sigma \in \Sigma_I$, $\langle q, i\rangle \in Q$ and $\langle q', j\rangle \in Q$,*

1. *If $q \in Q_i \setminus F_i$, then*

$$\delta(\langle q, i\rangle, \sigma)(\langle q', j\rangle) = \begin{cases} \delta_i(q,\sigma)(q') & \textit{if } i = j \\ 0 & \textit{otherwise} \end{cases}$$

2. *If $q = q^i_x \in F_i$, where $\Delta(\mathsf{M}_i, x) = \mathsf{M}_k$, then*

$$\delta(\langle q, i\rangle, \sigma)(\langle q', j\rangle) = \begin{cases} 1 & \textit{if } j = k \textit{ and } q' = q^k_0 \\ 0 & \textit{otherwise} \end{cases}$$

Note that the composition is a probabilistic transducer without exits. When the composition is in a state $\langle q, i\rangle$ corresponding to a non-exit state q of component M_i, it behaves like M_i. When the composition is in a state $\langle q_f, i\rangle$ corresponding to an exit state q_f of component M_i, the control is transferred to the start state of another component as determined by the transition function of the composer. Thus, at each point in time, only one component is active and interacting with the environment.

2.4 Parity objectives and values for probabilistic transducer An *index function* for a transducer is a function that assigns a natural number, called a priority index, to each state of the transducer. An index function α defines a parity objective Φ_α that is the subset of Q^ω that consists of the set of infinite sequence of states such that the minimum priority that is visited infinitely often is even. Given a probabilistic transducer $\mathcal{T}$ and a parity objective Φ, the value of the probabilistic transducer for the objective, denoted as $\mathsf{val}(\mathcal{T}, \Phi)$, is $\inf_{f \in \mathcal{F}} \mu_f(\Phi)$. In other words, it is the minimal probability with which the parity objective is satisfied over all strategies in the transducer.

2.5 The synthesis questions In this work we consider two types of synthesis questions for controlflow composition. In the first problem (namely, synthesis for embedded parity) the parity objective is specified directly on the state space of the library components, and in the second problem (namely, synthesis from DPW specifications) the parity objective is specified by a separate deterministic parity automaton.

2.5.1 Synthesis for Embedded Parity We first consider an index function that associates to each state of the components in the library a priority, and a specification defined as a parity condition over the sequence of visited states.

Exit control relation. Given a library $\mathcal{L}$ of width D, an *exit control relation* is a set $R \subseteq D \times \mathcal{L}$. We say that a composer $C = \langle D, \mathcal{L}, \mathcal{M}, \mathsf{M}_0, \Delta, \lambda\rangle$ is *compatible* with R, if the following holds: for all $\mathsf{M}, \mathsf{M}' \in \mathcal{M}$ and $i \in D$, if $\Delta(\mathsf{M}, i) = \mathsf{M}'$ then $\langle i, M'\rangle \in R$. Thus, each element of R can be viewed as a constraint on how the composer is allowed to connect components. An exit control relation is *non-blocking* if for every $i \in D$ there exists a component $M \in \mathcal{L}$ such that $\langle i, M\rangle \in R$ (i.e., every exit has at least one possible component for the next choice). For technical convenience we only consider non-blocking exit control relations. If $R = D \times \mathcal{L}$ (i.e., there is no constraint on the composer to connect components), then we refer to the relation as *unrestricted* exit control relation.

DEFINITION 2. (EMBEDDED PARITY REALIZABILITY AND SYNTHESIS.) *Consider a library $\mathcal{L}$ of width D, an exit control relation R for $\mathcal{L}$, and an index function α for the components in $\mathcal{L}$ that defines the parity objective Φ_α. The* qualitative *(resp.,* quantitative*)* embedded parity realizability problem *is to decide whether there exists a composer C over $\mathcal{L}$, such that C is compatible with R, and* $\mathsf{val}(\mathcal{T}_C, \Phi_\alpha) = 1$ *(resp.,* $\mathsf{val}(\mathcal{T}_C, \Phi_\alpha) \geq \eta$*, for a given rational threshold $\eta \in (0, 1)$). A witness composer for the qualitative problem is called an almost-sure composer, and for the quantitative problem is called an η-optimal composer. The corresponding* embedded parity synthesis problems *are to find such a composer C if it exists.*

2.5.2 Synthesis for DPW Specifications A *deterministic parity automaton (DPW)* is a deterministic transducer where the labeling function is an index function that defines a parity objective. Given a DPW A, every word (infinite sequence of input letters) induces a run of the automaton, which is an infinite sequence of states, and the word is accepted if the run satisfies the parity objective. The language L_A of a DPW A is the set of words accepted by A. Let A be a deterministic parity automaton (DPW), M be a probabilistic transducer and $\mathcal{L}$ be a library of components. We say A is a *monitor* for M (resp. $\mathcal{L}$) if the input alphabet of A is the same as the output alphabet of M (resp. $\mathcal{L}$). Let A be a monitor for M and let L_A be the language accepted by A. The value of M for A, denoted as $\mathsf{val}(M, A)$, is $\inf_{f \in \mathcal{F}} \mu_f(\lambda^{-1}(L_A))$. Note that the compatibility of the composer with an exit control relation can be encoded in the DPW (without loss of generality, we do not allow two distinct exit states to have the same output).

DEFINITION 3. (DPW REALIZABILITY AND SYNTHESIS.) *Consider a library $\mathcal{L}$ and a DPW A that is a monitor for $\mathcal{L}$. The* qualitative *(resp.,* quantitative*)* DPW probabilistic realizability problem *is to decide whether there exists a composer C over $\mathcal{L}$, such that $\mathsf{val}(\mathcal{T}_C, A) = 1$ (resp., $\mathsf{val}(\mathcal{T}_C, A) \geq \eta$, for a given rational threshold $\eta \in (0, 1)$). A witness composer for the qualitative problem is called an almost-sure composer, and for the quantitative problem is called an η-optimal composer. The corresponding* DPW probabilistic synthesis problems *are to find such a composer C if it exists.*

REMARK 1. *We remark that the realizability problem for libraries with components can be viewed as a 2-player partial-observation stochastic parity game. Informally, the game can be described as follows: the two players are the composer C and the environment E. The C player chooses components and the E player chooses sequence of inputs in the components chosen by C. However, C cannot see the inputs of E or even the length of the time inside a component. At the start C chooses a component M from the library $\mathcal{L}$. The turn passes to E, who chooses a sequence of inputs, inducing a probability distribution over paths in M from its start state to some exit x in D. The turn then passes to C, which must choose some component M' in $\mathcal{L}$ and pass the turn to E and so on. As C cannot see the moves made by E inside M, the choice of C cannot be based on the run in M, but only on the exit induced by the inputs selected by E and previous moves made by C. So C must choose the same next component M' for different runs that reach exit x of M. In general, different runs will visit different priorities inside M. This is a two-player stochastic parity game where one of the players (namely the composer C) does not have full information. However, there is also a crucial difference from traditional partial-observation games, as the composer does not even see the number of steps executed inside a component. If C has a winning strategy that requires finite memory, then such a strategy would yield a suitable finite composer to satisfy the parity objective defined by the index function α, thus solving the synthesis problem.*

3 The Complexity of Realizability for Embedded Parity

In this section we will establish the results for the complexity of realizability for embedded parity. In [21] the problem was interpreted as a partial-observation game (see Remark 1). While the natural interpretation of the embedded parity problem is a partial-observation game, we show how the problem can be interpreted as a perfect-information stochastic game.

3.1 Perfect-information Stochastic Parity Games In this section we present the basic definitions and results for perfect-information stochastic games.

Perfect-information stochastic games. A perfect-information stochastic game consists of a tuple $G = \langle S, S_1, S_2, A_1, A_2, \delta^G \rangle$, where S is a finite set of states partitioned into player-1 states (namely, S_1) and player-2 states (namely S_2), A_1 (resp., A_2) is the set of actions for player 1 (resp., player 2), and $\delta^G : (S_1 \times A_1) \cup (S_2 \times A_2) \to \mathcal{D}(S)$ is a probabilistic transition function that given a player-1 state and player-1 action, or a player-2 state and a player-2 action gives a probability distribution over the successor states. If the transition function is *deterministic* (that is the codomain of δ^G is S instead of $\mathcal{D}(S)$), then the game is a perfect-information deterministic game.

Plays and strategies. A *play* is an infinite sequence of state-action pairs $\langle s_0 a_0 s_1 a_1 \ldots \rangle$ such that for all $j \geq 0$ we have that if $s_j \in S_i$ for $i \in \{1, 2\}$, then $a_j \in A_i$ and $\delta^G(s_j, a_j)(s_{j+i}) > 0$. A strategy is a recipe for a player to choose actions to extend finite prefixes of plays. Formally, a strategy π for player 1 is a function $\pi : S^* \cdot S_1 \to \mathcal{D}(A_1)$ that given a finite sequence of visited states gives a probability distribution over the actions (to be chosen next). A *pure* strategy chooses a deterministic action, i.e., is a function $\pi : S^* \cdot S_1 \to A_1$. A pure memoryless strategy is a pure strategy that does not depend on the finite prefix of the play but only on the current state, i.e., is a function $\pi : S_1 \to A_1$. The definitions for player-2 strategies τ are analogous. We denote by Π (resp., Π^{PM}) the set of all (resp., all pure memoryless) strategies for player 1, and analogously Γ (resp., Γ^{PM} for player 2). Given strategies $\pi \in \Pi$ and $\tau \in \Gamma$, and a starting state s, there is a unique probability measure over events (i.e., measurable subsets of S^ω), which is denoted as $\mathbb{P}_s^{\pi,\tau}(\cdot)$.

Finite-memory strategies. A pure player-1 strategy uses *finite-memory* if it can be encoded by a transducer $\langle \mathfrak{M}, m_0, \pi_u, \pi_n \rangle$ where $\mathfrak{M}$ is a finite set (the memory of the strategy), $m_0 \in \mathfrak{M}$ is the initial memory value, $\pi_u : \mathfrak{M} \times S \to \mathfrak{M}$ is the memory-update function, and $\pi_n : \mathfrak{M} \to A_1$ is the next-action function. Note that a finite-memory strategy is a deterministic transducer with input alphabet S, output alphabet A_1, where π_u is the deterministic transition function, and π_n is the output labeling function. However, for finite-memory strategies, since the input and output is always the set of states and actions for player 1, for simplicity, we will represent them as a tuple $\langle \mathfrak{M}, m_0, \pi_u, \pi_n \rangle$. The *size* of the strategy is the number $|\mathfrak{M}|$ of memory values. If the current state is s, and the current memory value is m, then the memory is updated to $m' = \pi_u(m, s)$, and the strategy chooses the next action $\pi_n(m')$. Formally, $\langle \mathfrak{M}, m_0, \pi_u, \pi_n \rangle$ defines the strategy π such that $\pi(\rho) = \pi_n(\widehat{\pi}_u(m_0, \rho))$ for all $\rho \in S^+$, where $\widehat{\pi}_u$ extends π_u to sequences of states as expected.

Parity objectives, almost-sure, and value problem. Given a perfect-information stochastic game, a parity objective is defined by an index function α on the state space. Given a strategy π, the value of the strategy in a state s of the game G with parity objective Φ_α, denoted by $\mathsf{val}^G(\pi, \Phi_\alpha)(s)$, is the infimum of the probabilities among all player-2 strategies, i.e., $\mathsf{val}^G(\pi, \Phi_\alpha)(s) = \inf_{\tau \in \Gamma} \mathbb{P}_s^{\pi,\tau}(\Phi_\alpha)$. The value of the game is $\mathsf{val}^G(\Phi_\alpha)(s) = \sup_{\pi \in \Pi} \mathsf{val}^G(\pi, \Phi_\alpha)(s)$. A strategy π is almost-sure winning from s if $\mathsf{val}^G(\pi, \Phi_\alpha)(s) = 1$. The following theorem summarizes the basic results about perfect-information games.

THEOREM 3.1. *The following assertions hold* [14, 6, 9, 12, 1]*:*

1. (Complexity). *The quantitative decision problem (of whether* $\mathsf{val}^G(\Phi_\alpha) \geq \eta$*, given rational* $\eta \in (0, 1]$*) for perfect-information stochastic parity games lies in NP* $\cap$ *coNP (also UP* $\cap$ *coUP).*
2. (Memoryless determinacy). *We have*

$$\mathsf{val}^G(\Phi_\alpha)(s) = \sup_{\pi \in \Pi^{PM}} \inf_{\tau \in \Gamma} \mathbb{P}_s^{\pi,\tau}(\Phi_\alpha) = \inf_{\tau \in \Gamma^{PM}} \sup_{\pi \in \Pi} \mathbb{P}_s^{\pi,\tau}(\Phi_\alpha),$$

i.e., the quantification over the strategies can be restricted to $\pi \in \Pi^{PM}$ *and* $\tau \in \Gamma^{PM}$.

3.2 Complexity Results We first present the reduction for upper bounds.

The upper-bound reduction. Consider a library $\mathcal{L}$ of width D, a non-blocking exit control relation R for $\mathcal{L}$, and an index function α for $\mathcal{L}$ that defines the parity objective Φ_α. Let the number of components be $k + 1$, and let $M_i = \langle \Sigma_I, \Sigma_O, Q_i, q_0^i, \delta_i, F_i, L_i \rangle$ for $0 \leq i \leq k$. Let us denote by $[k] = \{0, 1, 2, \ldots, k\}$. We define a perfect-information stochastic game $G_\mathcal{L} = \langle S, S_1, S_2, A_1, A_2, \delta_\mathcal{L}^G \rangle$ with an index function α_G as follows: $S = \bigcup_{i=0}^k (Q_i \times \{i\}) \cup \{\bot\}$, $S_1 = \bigcup_{i=0}^k (F_i \times \{i\})$, $S_2 = S \setminus S_1$, $A_1 = [k]$, and $A_2 = \Sigma_I$. The state $\bot$ is a losing absorbing state (i.e., a state with self-loop as the only outgoing transition and assigned odd priority by the index function α_G), and the other transitions defined by the function $\delta_\mathcal{L}^G$ are as follows: (i) for $s = \langle q, i \rangle \in S_2$, and $\sigma \in A_2$

$$\delta_\mathcal{L}^G(\langle q, i \rangle, \sigma)(\langle q', j \rangle) = \begin{cases} \delta_i(q, \sigma)(q') & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

(ii) for $s = \langle q_x^i, i \rangle \in S_1$ and $j \in [k]$, we have that if $\langle x, M_j \rangle \in R$, then $\delta_\mathcal{L}^G(\langle q_x^i, i \rangle, j)(\langle q_0^j, j \rangle) = 1$, else $\delta_\mathcal{L}^G(\langle q_x^i, i \rangle, j)(\bot) = 1$. The intuitive description of the transitions is as follows: (1) Given a player-2 state that is a non-exit state q in a component M_i, and an action for player 2 that is an input letter, the transition function $\delta_{\mathcal{L}}^G$ mimics the transition δ_i of M_i; and (2) given a player-1 state that is an exit state q_x^i in component i, and an action for player 1 that is the choice of a component j, if $\langle x, M_j \rangle$ is allowed by R, then the next state is the starting state of component j, and if the choice $\langle x, M_j \rangle$ is invalid (not allowed by R), then the next state is the losing absorbing state $\perp$. For all $\langle q, i \rangle \in S \setminus \{\perp\}$ we have $\alpha_G(\langle q, i \rangle) = \alpha(q)$, and we denote by Φ_{α_G} the parity objective in $G_{\mathcal{L}}$. Note the similarity of the state space description in comparison with Definition 1 for controlflow composition.

Correctness of reduction. There are two steps to establish correctness of the reduction. The first step is given a composer for $\mathcal{L}$ to construct a finite-memory strategy for player 1 in $G_{\mathcal{L}}$. Intuitively, this is simple as a composer represents a strategy for a partial-observation game (Remark 1), whereas in $G_{\mathcal{L}}$ we have perfect information. However, not every strategy in $G_{\mathcal{L}}$ can be converted to a composer (i.e., a perfect-information game strategy cannot be converted to a partial-observation strategy). But we show that a pure memoryless strategy in $G_{\mathcal{L}}$ can be converted to a composer. While [21] treats the problem as a partial-observation game, our insight to convert pure memoryless strategies of the perfect-information game $G_{\mathcal{L}}$ to composers allows us to establish the correctness with respect to $G_{\mathcal{L}}$. We present both the steps below.

LEMMA 3.1. *Consider a library $\mathcal{L}$ of width D, a non-blocking exit control relation R for $\mathcal{L}$, and an index function α for $\mathcal{L}$ that defines the parity objective Φ_α. Let $G_{\mathcal{L}}$ be the corresponding perfect-information stochastic game with parity objective Φ_{α_G}. For all composers C, and the corresponding strategy π_C in $G_{\mathcal{L}}$ we have* $\mathsf{val}(\mathcal{T}_C, \Phi_\alpha) = \mathsf{val}^{G_{\mathcal{L}}}(\pi_C, \Phi_{\alpha_G})(\langle q_0^0, 0 \rangle)$.

Proof. Composer to finite-memory strategies. Given a composer $C = \langle D, \mathcal{L}, \mathcal{M}, \mathsf{M}_0, \Delta, \lambda \rangle$ for the library we define a finite-memory strategy $\pi_C = \langle \mathfrak{M}, m_0, \pi_u, \pi_n \rangle$ for the perfect-information stochastic game $G_{\mathcal{L}}$ as follows: (a) $\mathfrak{M} = \mathcal{M}$ and $m_0 = \mathsf{M}_0$; and (b) $\pi_u(\mathsf{M}_i, s) = \mathsf{M}_i$ for $s \in S_2$, and $\pi_u(\mathsf{M}_i, \langle q_x^j, j \rangle) = \mathsf{M}_\ell$ for $s \in S_1$ if $\Delta(\mathsf{M}_i, x) = \mathsf{M}_\ell$, where $\lambda(\mathsf{M}_i) = M_j$; and (c) $\pi_n(\mathsf{M}_i) = j$ where $\lambda(\mathsf{M}_i) = M_j$.

In other words, the finite-memory strategy has the same state space as the composer, and if the current state is a player-2 state, then it does not update the memory state, and given the current state is a player-1 state it updates it memory state according to the transition function of the composer, and the action played is according to the labeling function of the transducer. In other words, the strategy π_C mimics the composer, and there is a one-to-one correspondence between strategies for player 2 in the perfect-information stochastic game, and the strategies of the environment in $\mathcal{T}_C$. Without loss of generality we consider that the composer must always start with the first component (i.e., M_0) and hence the starting state is $\langle q_0^0, 0 \rangle$. This gives us the desired result. ∎

For the second step we first consider valid pure memoryless strategies.

Valid pure memoryless strategies in $G_{\mathcal{L}}$. A pure memoryless strategy π in $G_{\mathcal{L}}$ is *valid* if the following condition holds: for all states $\langle q_x^i, i \rangle \in S_1$ if $\pi(\langle q_x^i, i \rangle) = j$, then $\langle x, M_j \rangle \in R$, i.e., the choices of the pure memoryless strategies respect the exit control relation.

LEMMA 3.2. *Consider a library $\mathcal{L}$ of width D, a non-blocking exit control relation R for $\mathcal{L}$, and an index function α for $\mathcal{L}$ that defines the parity objective Φ_α. Let $G_{\mathcal{L}}$ be the corresponding perfect-information stochastic game with parity objective Φ_{α_G}. For all valid pure memoryless strategies π in $G_{\mathcal{L}}$, and the corresponding composer C_π, we have* $\mathsf{val}(\mathcal{T}_{C_\pi}, \Phi_\alpha) = \mathsf{val}^{G_{\mathcal{L}}}(\pi, \Phi_{\alpha_G})(\langle q_0^0, 0 \rangle)$.

Proof. Valid pure memoryless strategies to composers. Given a valid pure memoryless strategy π in $G_{\mathcal{L}}$ we define a composer $C_\pi = \langle D, \mathcal{L}, \mathcal{M}, \mathsf{M}_0, \Delta, \lambda \rangle$ as follows: $\mathcal{M} = [k]$, $\mathsf{M}_0 = 0$, $\lambda(i) = M_i$, and for $0 \leq i \leq k$ and $x \in D$ we have that $\Delta(i, x) = j$ where $\pi(\langle q_x^i, i \rangle) = j$ for $q_x^i \in F_i$. In other words, for the composer there is a state for every component, and given a component and an exit state, the composer plays as the pure memoryless strategy. Note that since π is valid, the composer obtained from π is compatible with the relation R. Note that the composer mimics the pure memoryless strategy, and there is a one-to-one correspondence between strategies of player 2 in $G_{\mathcal{L}}$ and strategies of the environment in $\mathcal{T}_{C_\pi}$, which establishes the result. ∎

LEMMA 3.3. *Consider a library $\mathcal{L}$ of width D, a non-blocking exit control relation R for $\mathcal{L}$, and an index function α for $\mathcal{L}$ that defines the parity objective Φ_α. Let $G_{\mathcal{L}}$ be the corresponding perfect-information stochastic game with parity objective Φ_{α_G}. There exists an almost-sure composer iff there exists an almost-sure winning strategy in $G_{\mathcal{L}}$ from $\langle q_0^0, 0\rangle$, and there exists an η-optimal composer iff the value in $G_{\mathcal{L}}$ at $\langle q_0^0, 0\rangle$ is at least η.*

Proof. Sufficiency of valid pure memoryless strategies. Note that in the game $G_{\mathcal{L}}$ we can restrict to only valid pure memoryless strategies because for a state $s \in S_1$ if an action is chosen that is not allowed by a valid strategy, then it leads to the losing absorbing state $\bot$. Thus in $G_{\mathcal{L}}$ if there is an almost-sure winning strategy (resp., a strategy to ensure that the value is at least η), then there is a valid pure memoryless strategy to ensure the same (Theorem 3.1). ∎

Lemma 3.3 along with Theorem 3.1 imply that the qualitative and quantitative problems for the realizability for embedded parity lie in NP $\cap$ coNP (also UP $\cap$ coUP). We now present two related results. We first show that with unrestricted exit control relation the problem can be solved in polynomial time, and then show that in general (i.e., with exit control relation) the problem is at least as hard as solving perfect-information deterministic parity games (for which no polynomial-time algorithm is known).

The unrestricted exit control relation problem. For the unrestricted exit control relation problem, we modify the construction of the game $G_{\mathcal{L}}$ as follows: we add another state $\top$ that belongs to player 1, and for every state $s \in S_1$ and every action $a \in A_1$ the next state is $\top$, and from $\top$ player 1 can choose any component (i.e., state $\langle q_0^j, j\rangle$ in $G_{\mathcal{L}}$). We refer to the modified game as $\overline{G}_{\mathcal{L}}$. If the exit control relation is unrestricted then the result corresponding to Lemma 3.3 holds for $\overline{G}_{\mathcal{L}}$. However, in $\overline{G}_{\mathcal{L}}$ the number of memoryless player-1 strategies is only $k+1$ (one each for the choice of each component at $\top$), and once a memoryless strategy for player 1 is fixed we obtain an MDP which can be solved in polynomial time (and the qualitative problem in strongly polynomial time by discrete graph theoretic algorithms) [16, 13, 10, 11]. Hence it follows that both the qualitative and quantitative realizability and synthesis problems for embedded parity with unrestricted exit control relation can be solved in polynomial time.

The hardness reduction. We now present a reduction form perfect-information deterministic parity games to the realizability problem for embedded parity. For simplicity we consider *alternating* games where the players make move in alternate turns, i.e., we consider a perfect-information game $G = \langle S, S_1, S_2, A_1, A_2, \delta^G\rangle$ where δ^G is deterministic and is decomposed into two functions $\delta_1^G : S_1 \times A_1 \to S_2$ and $\delta_2^G : S_2 \times A_2 \to S_1$ (player-1 move leads to player-2 state and vice versa). A perfect-information deterministic game with an index function α can be converted to an equivalent alternating game with a linear blow-up by adding dummy states. Given an alternating perfect-information game, let $S_2 = \{s_0^2, s_1^2, s_2^2, \ldots, s_k^2\}$ and $S_1 = \{s_1^1, s_2^1, \ldots, s_d^1\}$. We construct $\mathcal{L}$ of width S_1, an exit control relation R, and an index function α' as follows: (a) there are $k+1$ components $M_0, M_1, \ldots, M_k$ one for each state in S_2; (b) each $M_i = \langle \Sigma_I, \Sigma_O, Q_i, q_0^i, \delta_i, F_i, L_i\rangle$ is defined as follows: (i) $\Sigma_I = A_2$, (ii) Σ_O and L_i are not relevant for the reduction, (iii) $Q_i = \{q_0^i, q_1^i, \ldots, q_d^i\}$ with $F_i = Q_i \setminus \{q_0^i\}$, and (iv) $\delta_i(q_0^i, \sigma) = q_j^i$ where $s_j^1 = \delta_2^G(s_i^2, \sigma)$ for all $\sigma \in \Sigma_I$. The exit control relation R is as follows: $R = \{\langle i, M_j\rangle \mid \exists a_1 \in A_1.\ \delta_1^G(s_i^1, a_1) = s_j^2\}$. Intuitively, there exists a component for each state in S_2, and the exit states for each component corresponds to states of player 1. In the start state q_0^i for component M_i the transition function δ_i represents the transition function δ_2^G of the perfect-information game, i.e., given an input letter σ which is an action for player 2, if the transition given σ is from s_i^2 to s_j^1, then in M_i there is a corresponding transition to q_j^i. The exit control relation represents the transitions for player 1. Since the exit states in each component is reached in one step, the composer strategies for the library represents perfect-information strategies. The index function α' is as follows: $\alpha'(q_0^i) = \alpha(s_i^2)$ and $\alpha'(q_j^i) = \alpha(s_j^1)$ for $j \geq 1$. There exists an almost-sure winning strategy in G from s_0^2 iff there exists an almost-sure composer for $\mathcal{L}$ with R. Also note that for the reduction every component transducer is deterministic.

THEOREM 3.2. (COMPLEXITY OF EMBEDDED PARITY REALIZABILITY) *Consider a library $\mathcal{L}$ of width D, a non-blocking exit control relation R for $\mathcal{L}$, and an index function α for $\mathcal{L}$ that defines the parity objective Φ_α. The following assertions hold:*

1. *The realizability problem belongs to NP $\cap$ coNP (also UP $\cap$ coUP), and is at least as hard as the (almost-sure) decision problem for perfect-information deterministic parity games.*
2. *If R is an unrestricted exit control relation, then the realizability and synthesis problems can be solved in polynomial time*

4 The Complexity of Realizability for DPW Specifications

In this section we present three results. First, we present a new result for partial-observation stochastic parity games. Second, we show that the qualitative realizability problem for DPW specifications can be reduced to our solution for partial-observation stochastic games yielding an EXPTIME-complete result for the problem. Finally, we show that the quantitative realizability problem for DPW specifications is undecidable.

4.1 Partial-observation Stochastic Parity Games In this section we consider partial-observation games with various restrictions on strategies and present a new result for a class of strategies that correspond to the analysis of the qualitative realizability problem.

Partial-observation stochastic games. In a stochastic game with partial observation, some states are not distinguishable for player 1. We say that they have the same observation for player 1. Formally, a partial-observation stochastic game consists of a stochastic game $G = \langle S, S_1, S_2, A_1, A_2, \delta^G \rangle$, a finite set $\mathcal{O}$ of observations, and a mapping $\mathsf{obs} : S \to \mathcal{O}$ that assigns to each state s of the game an observation $\mathsf{obs}(s)$ for player 1.

Observational equivalence and strategies. The observation mapping induces indistinguishability of play prefixes for player 1, and therefore we need to consider only the player-1 strategies that play in the same way after two indistinguishable play prefixes. We consider three different classes of strategies depending on the indistinguishability of play prefixes for player 1 and they are as follows: (i) the play prefixes have the same observation sequence; (ii) the play prefixes have the same observation sequence, except that the last observation may be repeated arbitrarily many times; and (iii) the play prefixes have the same sequence of distinct observations, that is they have the same observation sequence up to repetition (stuttering). We now formally define the classes of strategies.

Classes of strategies. The *observation sequence* of a sequence $\rho = s_0 s_1 \dots s_n$ is the sequence $\mathsf{obs}(\rho) = \mathsf{obs}(s_0) \dots \mathsf{obs}(s_n)$ of state observations; the *collapsed stuttering* of ρ is the sequence $\overline{\mathsf{obs}}(\rho) = o_0 o_1 o_2 \dots$ of distinct observations defined inductively as follows: $o_0 = \mathsf{obs}(s_0)$ and for all $i \geq 1$ we have $o_i = \mathsf{obs}(s_i)$ if $\mathsf{obs}(s_i) \neq \mathsf{obs}(s_{i-1})$, and $o_i = \epsilon$ otherwise (where ϵ is the empty sequence). We consider three types of strategies. A strategy π for player 1 is

- *observation-based* if for all sequences $\rho, \rho' \in S^+$ such that $\mathsf{last}(\rho) \in S_1$ and $\mathsf{last}(\rho') \in S_1$, if $\mathsf{obs}(\rho) = \mathsf{obs}(\rho')$ then $\pi(\rho) = \pi(\rho')$;
- *observation-based stutter-invariant* if it is observation-based and for sequences $\rho \in S^+$, for all states $s \in S$, if $\mathsf{obs}(s) = \mathsf{obs}(\mathsf{last}(\rho))$, then $\pi(\rho) = \pi(\rho s)$;
- *observation-based collapsed-stutter-invariant* if for all sequences $\rho, \rho' \in S^+$ such that $\mathsf{last}(\rho) \in S_1$ and $\mathsf{last}(\rho') \in S_1$, if $\overline{\mathsf{obs}}(\rho) = \overline{\mathsf{obs}}(\rho')$, then $\pi(\rho) = \pi(\rho')$.

The key difference between observation-based stutter-invariant and observation-based collapsed-stutter-invariant strategies is as follows: they both must play the same action while an observation is repeated, however,

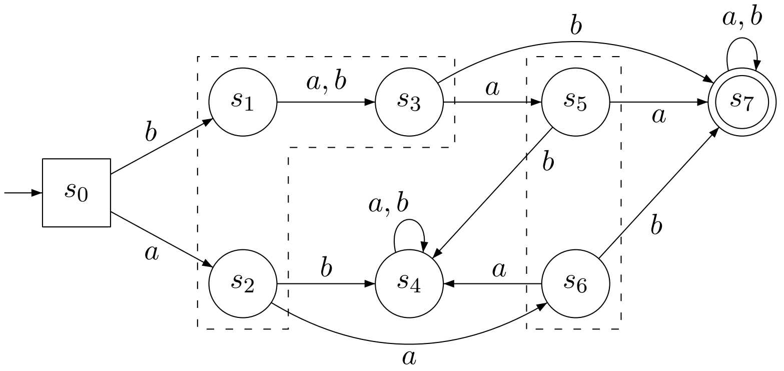

Figure 1: Reachability game where there is no collapsed-stuttering invariant winning strategy, but there is a observation-based stutter-invariant winning strategy. States $\{s_1, s_2, s_3\}$ are indistinguishable for player 1, as well as $\{s_5, s_6\}$. The objective for player 1 is to reach s_7.

collapsed-stutter-invariant strategies cannot observe the length of the repetitions of an observation, whereas a stutter-invariant strategy can. In the following example we illustrate this difference.

Example. Figure 1 shows a (non-stochastic) reachability game where the objective of player 1 is to reach s_7. The states s_1, s_2, s_3 have the same observation (for player 1), and the states s_5, s_6 as well. The game starts in a player-2 state s_0 with successors s_1 and s_2. Thus after one step, player 1 does not know whether the game is in s_1 or in s_2. If the game is in s_2, then the action b leads to s_4 from where the target state s_7 is not reachable. Hence playing b is not a good choice. Playing a gives either a new observation (s_6 is reached), or the same observation as in the previous step (s_3 is reached). At this point, a simple observation-based strategy can play b and reach s_7 for sure. However, an observation-based stutter-invariant (or collapsed-stutter-invariant) strategy must keep playing a in s_3 (since the observation did not change) and reaches s_5 with same observation as in s_6. Now, an observation-based *stutter-invariant* strategy wins by playing a in s_5 and b in s_6. Indeed with observation-based stutter-invariant strategies, player 1 can distinguish whether the game is in s_5 or in s_6 (simply looking at the length of the play prefix in this case). However, with an observation-based *collapsed-stutter-invariant* strategy, player 1 cannot win because the play prefixes $\rho_1 = s_0 \dots s_5$ and $\rho_2 = s_0 \dots s_6$ have the same collapsed stuttering sequence of observations, thus forcing player 1 to choose the same action a or b, and action a is losing for ρ_2 and action b for ρ_1. Hence in this game there is no observation-based collapsed-stutter-invariant winning strategy, but there is an observation-based stutter-invariant winning strategy.

The previous example shows that collapsed-stutter-invariant strategies are different from stutter-invariant, as well as from standard observation-based strategies. Our goal is to decide the existence of finite-memory almost-sure winning strategies in partial-observation stochastic parity games. This problem has been studied for observation-based strategies and optimal complexity result (EXPTIME-completeness) has been established in [8]. We now present a polynomial-time reduction for deciding the existence of finite-memory almost-sure winning collapsed-stutter-invariant strategies to observation-based strategies.

Reduction of collapsed-stutter-invariant problem to observation-based problem. There are two main ideas of the reduction. (1) The first is that whenever player 1 plays an action a, the action a is stored in the state space as long as the observation of the state remains the same. This allows to check that player 1 plays always the same action along a sequence of identical observations. This only captures the stutter-invariant restriction, but not the collapsed-stutter-invariant restriction. (2) Second, whenever a transition is executed, player 2 is allowed to loop arbitrarily many times through the new state. This ensures that player 1 cannot rely on the number of times he

<table>
<tr><td></td><td colspan="2">$s \xrightarrow{a} s'$</td></tr>
<tr><td></td><td>$\mathsf{obs}(s) = \mathsf{obs}(s')$
(storing last action of player 1)</td><td>$\mathsf{obs}(s) \neq \mathsf{obs}(s')$
(forgetting last action of player 1)</td></tr>
<tr><td rowspan="2">$s \in S_1$</td><td>$\langle s, x\rangle \xrightarrow{a} \bot$ for $x \neq a$ and $x \neq 0$
$\langle s, a\rangle \xrightarrow{a} \langle s', \overline{a}\rangle$
$\langle s, 0\rangle \xrightarrow{a} \langle s', \overline{a}\rangle$</td><td>$\langle s, x\rangle \xrightarrow{a} \bot$ for $x \neq a$ and $x \neq 0$
$\langle s, a\rangle \xrightarrow{a} \langle s', \overline{0}\rangle$
$\langle s, 0\rangle \xrightarrow{a} \langle s', \overline{0}\rangle$</td></tr>
<tr><td colspan="2">in $\langle s, x\rangle$, if $x \in A_1$, then player 1 should play the stored action x;
if $x = 0$, no action is stored and player 1 can choose any action.</td></tr>
<tr><td>$s \in S_2$</td><td>$\langle s, x\rangle \xrightarrow{a} \langle s', \overline{x}\rangle$</td><td>$\langle s, x\rangle \xrightarrow{a} \langle s', \overline{0}\rangle$</td></tr>
<tr><td>$s \in S_1 \cup S_2$</td><td colspan="2">$$\begin{tikzcd}\langle s, \overline{x}\rangle \arrow[loop above, "\sharp"] \arrow[r, "a"] & \langle s, x\rangle\end{tikzcd}$$</td></tr>
<tr><td></td><td colspan="2">player 2 can play all actions available in the original game, and
repeat arbitrarily many times the current observation.</td></tr>
</table>

Figure 2: Game transformation for the reduction of collapsed-stutter-invariant problem to observation-based problem.

sees an observation, thus that player 1 is collapsed-stutter-invariant. However, it should be forbidden for player 2 to loop forever in a state, which can be ensured by assigning priority 0 to the loop (hence player 1 would win the parity objective if the loop is taken forever by player 2). We now formally present the reduction.

The formal reduction. The reduction is illustrated in Figure 2 and formally presented below. Given a partial-observation stochastic game $G = \langle S, S_1, S_2, A_1, A_2, \delta^G\rangle$ with observation mapping $\mathsf{obs} : S \to \mathcal{O}$, we construct a game $G' = \langle S', S'_1, S'_2, A_1, A'_2, \delta^{G'}\rangle$ as follows:

- $S' = S \times (A_1 \cup \overline{A}_1 \cup \{0, \overline{0}\}) \cup \{\bot\}$ where $\overline{A}_1 = \{\overline{a} \mid a \in A_1\}$, assuming that $0 \not\in A_1$. The states $\langle s, 0\rangle$ are a copy of the state space of the original game, and in the states $\langle s, a\rangle$ with $s \in S_1$ and $a \in A_1$, player 1 is required to play action a; in the states $\langle s, \overline{0}\rangle$ and $\langle s, \overline{a}\rangle$, player 2 can stay for arbitrarily many steps. The state $\bot$ is absorbing and losing for player 1.
- $S'_1 = S_1 \times (A_1 \cup \{0\}) \cup \{\bot\}$.
- $S'_2 = S' \setminus S'_1 = (S_2 \times (A_1 \cup \{0\})) \cup (S \times (\overline{A}_1 \cup \{\overline{0}\}))$.
- $A'_2 = A_2 \cup \{\sharp\}$, assuming $\sharp \not\in A_2$.
- The probabilistic transition function $\delta^{G'}$ is defined as follows: for all player-1 states $\langle s, x\rangle \in S'_1$ and actions $a \in A_1$:
  - if $x \in A_1 \setminus \{a\}$, then let $\delta^{G'}(\langle s, x\rangle, a))(\bot) = 1$, that is player 1 loses the game if he does not play the stored action;
  - if $x = a$ or $x = 0$, then for all $s' \in S'$ let
    $\delta^{G'}(\langle s, x\rangle, a))(\langle s', \overline{a}\rangle) = \delta^G(s, a)(s')$ if $\mathsf{obs}(s') = \mathsf{obs}(s)$, and let
    $\delta^{G'}(\langle s, x\rangle, a))(\langle s', \overline{0}\rangle) = \delta^G(s, a)(s')$ if $\mathsf{obs}(s') \neq \mathsf{obs}(s)$; thus we store the action a as long as the state observation does not change;

- All other probabilities $\delta^{G'}(\langle s, x\rangle, a))(\cdot)$ are set to 0, for example $\delta^{G'}(\langle s, 0\rangle, a))(\langle s', y\rangle) = 0$ for all $y \neq \overline{a}$;

and for all player-2 states $\langle s, x\rangle \in S'_2$, and actions $a \in A_2$:

- if $x \in A_1 \cup \{0\}$, then for all $s' \in S'$ let $\delta^{G'}(\langle s, x\rangle, a))(\langle s', \overline{x}\rangle) = \delta^G(s, a)(s')$ if $\mathsf{obs}(s') = \mathsf{obs}(s)$, and let $\delta^{G'}(\langle s, x\rangle, a))(\langle s', \overline{0}\rangle) = \delta^G(s, a)(s')$ if $\mathsf{obs}(s') \neq \mathsf{obs}(s)$; thus all actions are available to player 2 as in the original game, and the stored action x of player 1 is maintained if the state observation does not change;
- if $x = \overline{b}$ for some $b \in A_1 \cup \{0\}$, then let $\delta^{G'}(\langle s, \overline{b}\rangle, \sharp))(\langle s, \overline{b}\rangle) = 1$, and $\delta^{G'}(\langle s, \overline{b}\rangle, a))(\langle s, b\rangle) = 1$ if $a \neq \sharp$; thus player 2 can decide to stay arbitrarily long in $\langle s, \overline{b}\rangle$ before going back to $\langle s, b\rangle$;
- All other probabilities $\delta^{G'}(\langle s, x\rangle, a))(\cdot)$ and $\delta^{G'}(\langle s, x\rangle, \sharp))(\cdot)$ are set to 0.

The observation mapping obs' is defined according to the first component of the state: $\mathsf{obs}'(\langle s, x\rangle) = \mathsf{obs}(s)$. Given an index function α for G, define the index function α' for G' as follows: $\alpha'(\langle s, x\rangle) = \alpha(s)$ and $\alpha'(\langle s, \overline{x}\rangle) = 0$ for all $s \in S$ and $x \in A_1 \cup \{0\}$, and $\alpha'(\perp) = 1$. Hence, the state $\perp$ is losing for player 1, and the player-2 states $\langle s, \overline{x}\rangle$ are winning for player 1 if player 2 stays there forever.

LEMMA 4.1. *Given a partial-observation stochastic game G with observation mapping* obs *and parity objective Φ_α defined by the index function α, a game G' with observation mapping* obs' *and parity objective $\Phi_{\alpha'}$ defined by the index function α' can be constructed in polynomial time such that the following statements are equivalent:*

- *there exists a finite-memory almost-sure winning observation-based collapsed-stutter-invariant strategy π for player 1 in G from s_0 for the parity objective Φ_α;*
- *there exists a finite-memory almost-sure winning observation-based strategy π' for player 1 in G' from $\langle s_0, \overline{0}\rangle$ for the parity objective $\Phi_{\alpha'}$.*

Correctness argument. The correctness proof has two directions: first, given a finite-memory almost-sure winning collapsed-stutter-invariant strategy in G to construct a similar witness of observation-based strategy in G', and vice versa (the second direction). We present both directions below.

First direction. For the first direction, given a finite-memory almost-sure winning collapsed-stutter-invariant strategy π for player 1 in G, we construct a finite-memory observation-based strategy π' for player 1 in G' such that π' is almost-sure winning in G'.

To define $\pi'(\rho')$ for a play prefix ρ' in G' such that $\mathsf{last}(\rho') \in S'_1$, we construct a play prefix ρ in the original game G and define $\pi'(\rho') = \pi(\rho)$. We construct ρ as $\mu(\rho')$ where μ is a mapping that first removes from ρ' all states of the form $\langle s, \overline{x}\rangle$ for $s \in S$ and $x \in A_1 \cup \{0\}$, and then projects all the other states $\langle t, \cdot\rangle$ to their first component t. It follows that $\rho = \mu(\rho')$ is a play prefix in G and if the observation sequences of two prefixes ρ'_1, ρ'_2 in G' are the same (i.e., $\mathsf{obs}'(\rho'_1) = \mathsf{obs}'(\rho'_2)$), then the collapsed stuttering of $\mu(\rho'_1)$ and $\mu(\rho'_2)$ is also the same. It follows that the constructed strategy π' is well defined, and since π is collapsed-stutter-invariant, π' is observation-based.

We show that π' is almost-sure winning for the parity objective $\Phi_{\alpha'}$ in G'. Consider an arbitrary strategy τ' for player 2 in G'. We can assume without loss of generality that:

- τ' is pure since when strategy π' is fixed in G', we get a 1-player stochastic games, and pure strategies are sufficient in 1-player stochastic games [7];
- no play compatible with π' and τ' gets stuck in a state of the form $\langle s, \overline{x} \rangle$ for $s \in S$ and $x \in A_1 \cup \{0\}$ (i.e., no play loops forever through a self-loop on some state $\langle s, \overline{x} \rangle$ after some prefix ρ'); this assumption is also without loss of generality because if τ' is a spoiling strategy (i.e., it ensures against π' that the parity objective is satisfied with probability less than 1) and a play gets stuck after some prefix ρ', then we can define a strategy τ'' that plays arbitrarily after ρ' but does not get stuck, i.e., never plays $\sharp$, since getting stuck forever implies winning for player 1. It follows that τ'' is also a spoiling strategy and never gets stuck.

From the strategy τ' we define a strategy τ for player 2 in G (that basically mimics τ' ignoring the $\sharp$ actions). It follows by induction that (up to the mapping μ) the probability measure over plays in G under strategies π and τ coincides with the probability measure over plays in G' under strategies π' and τ'. Since μ preserves the satisfaction of the parity objective, it follows that with probability 1 the parity objective is satisfied in G' under π' and τ', and thus π' is an almost-sure winning observation-based strategy in G'.

Second direction. For the second direction of the lemma, consider that there exists a finite-memory almost-sure winning observation-based strategy π' for player 1 in G' from state $\langle s_0, \overline{0} \rangle$ for the parity objective $\Phi_{\alpha'}$, and we show that there exists an observation-based collapsed-stutter-invariant almost-sure winning observation-based strategy for player 1 in G from s_0.

Let $\langle \mathfrak{M}', m'_0, \pi'_u, \pi'_n \rangle$ be a transducer that encodes π' (thus $\mathfrak{M}'$ is finite). We construct a transducer $\langle \mathfrak{M}, m_0, \pi_u, \pi_n \rangle$ and show that it encodes an observation-based collapsed-stutter-invariant strategy π that is almost-sure winning in G from s_0 for the parity objective Φ_α. Intuitively, given a memory value $m' \in \mathfrak{M}'$ and a sequence of states with identical observation o visited in the game G', the memory will be updated (according to π'_u) and the actions played (according to π'_n) may be different depending on the number of repetitions of the observation o. To construct a collapsed-stutter-invariant strategy, we update the memory to a value that occurs infinitely often in the sequence of memory updates obtained when observing o repeatedly. This ensures that, as long as the observation does not change, the constructed strategy plays always the same action. Moreover, this action could indeed be played by the original strategy (even after an arbitrarily long sequence of identical observations). The transducer for π is defined as follows:

- $\mathfrak{M} = \mathfrak{M}' \times \mathcal{O}$. A memory value $m = \langle m', o \rangle$ corresponds to memory value m' in the transducer for π', and the current observation is o. As long as the next observation is o, the memory value m does not change (there is a self-loop on m for all inputs s such that $\mathsf{obs}(s) = o$).
- $m_0 = \langle m', \mathsf{obs}(s_0) \rangle$ where m' is such that $m' = \pi'_u(m'_0, s_0^n)$ for infinitely many n, and $s_0^n = \underbrace{s_0 s_0 \dots s_0}_{n \text{ times}}$ is the n-fold repetition of s_0 (such m' always exists since $\mathfrak{M}'$ is finite); note that in the definition of m' we can equivalently replace s_0^n by s^n for any s such that $\mathsf{obs}(s) = \mathsf{obs}(s_0)$ since the strategy π' is observation-based.
- For all $\langle m'_1, o_1 \rangle \in \mathfrak{M}$ and $s \in S$, if $\mathsf{obs}(s) = o_1$, then $\pi_u(\langle m'_1, o_1 \rangle, s) = \langle m'_1, o_1 \rangle$ (self-loop), and if $\mathsf{obs}(s) \neq o_1$, then $\pi_u(\langle m'_1, o_1 \rangle, s) = \langle m'_2, \mathsf{obs}(s) \rangle$ where $m'_2 = \pi'_u(m'_1, s^n)$ for infinitely many n.
- $\pi_n(\langle m', o \rangle) = \pi'_n(m')$.

First, we show that the strategy π is almost-sure winning in G from s_0. The proof of this claim is by contradiction: assume that π is not almost-sure winning. Then there exists a spoiling strategy τ for player 2 such that the parity objective Φ_α is satisfied with probability less than 1. From τ, we define a strategy τ' for player 2 in G' intuitively as follows: the strategy τ' mimics the strategy τ when the current state is of the form $\langle s, x\rangle$ where $s \in S_2$ and $x \in A_1 \cup \{0\}$, and ensures the following invariant: given any prefix ρ' in G', the memory value of π' after ρ' is m' if and only if the memory value of π after $\mu(\rho')$ is of the form $\langle m', \cdot\rangle$, where μ is the mapping defined in the first direction of the proof. Player 2 can always ensure this invariant as follows: given the memory value of π is updated to $\langle m', o\rangle$, repeat the self-loop on action $\sharp$ sufficiently many times to let the memory value of π' be updated to m', which is always possible since by definition of the transducer for π, the value m' is "hit" infinitely often in the transducer for π' when a state with observation o is visited forever. For example, the strategy τ' stays in the initial state $\langle s_0, \overline{0}\rangle$, which is a player 2 state, for n steps where n is such that $m' = \pi'_u(m'_0, s_0^n)$ where m' is such that $\langle m', \mathsf{obs}(s_0)\rangle$ is the initial memory value of π. It follows from this definition of τ' that for all play prefixes ρ' in G' that are compatible with π' and τ' in G', the play prefix $\mu(\rho')$ is compatible with π and τ and has the same probability as ρ'. Therefore the respective probability measures in G and in G' coincide (up to the mapping μ) and since for all infinite plays ρ in G and ρ' in G' such that $\rho = \mu(\rho')$, we have $\rho \in \Phi_\alpha$ if and only if $\rho \in \Phi_{\alpha'}$, it follows that τ' is a spoiling strategy in G (the parity objective $\Phi_{\alpha'}$ is satisfied with probability less than 1 under strategies π' and τ'). This contradicts that π' is almost-sure winning. Hence the original claim that the strategy π is almost-sure winning in G holds.

Second, we show that the strategy π is observation-based collapsed-stutter-invariant. We have by induction that $\pi_u(m_0, \rho) = \pi_u(m_0, \rho \cdot s)$ for all play prefixes ρ and states s such that $\mathsf{obs}(s) = \mathsf{obs}(\mathsf{last}(\rho))$, and it follows that $\pi(\rho) = \pi(\rho')$ if $\overline{\mathsf{obs}}(\rho) = \overline{\mathsf{obs}}(\rho')$ which concludes the argument.

Since solving partial-observation stochastic parity games with finite-memory observation-based strategies is EXPTIME-complete for almost-sure winning [8], we get an EXPTIME upper bound for games with observation-based collapsed-stutter-invariant strategy by the reduction in Lemma 4.1. The same complexity results hold for the class of observation stutter-invariant strategies, by removing all $\sharp$-labelled self-loops for player 2 in the reduction for collapsed-stutter-invariant strategies. We note that an EXPTIME lower bound can be established for those problems by a converse reduction that introduces in every transition an intermediate dummy state with a different observation, thus two consecutive observations are always different, and the observation-based strategies are also collapsed-stutter-invariant.

THEOREM 4.1. *The qualitative problem of deciding whether there exists a finite-memory almost-sure winning observation-based collapsed-stutter-invariant (or stutter-invariant) strategy in partial-observation stochastic games with parity objectives is EXPTIME-complete.*

4.2 The Complexity of Qualitative Realizability We now present the complexity result for qualitative realizability for DPW specifications via a reduction to the problem of deciding the existence of finite-memory almost-sure winning collapsed-stutter-invariant strategies in partial-observation games. The reduction formalizes the intuition described in Remark 1.

Reduction of synthesis for DPW specifications to collapsed-stutter-invariant problem. The reduction is analogous to the upper-bound reduction presented in Section 3.2. Given a library $\mathcal{L}$ of width D and a DPW, the game we construct is the product of the game $G_{\mathcal{L}}$ with the DPW. The states are of the form $\langle q, i, p\rangle$ where $\langle q, i\rangle$ is a state of $G_{\mathcal{L}}$ and p is a state of the DPW. The third component p is updated according to the deterministic transition function δ_P of the DPW. Thus the successors of $\langle q, i, p\rangle$ are of the form $\langle \cdot, \cdot, p'\rangle$ where $p' = \delta_P(p, L_i(q))$. For example, the transitions for player-2 states $s = \langle q, i, p\rangle$, and actions $\sigma \in A_2$ are defined by

$$\delta^G_{\mathcal{L}}(\langle q, i, p\rangle, \sigma)(\langle q', j, p'\rangle) = \begin{cases} \delta_i(q, \sigma)(q') & \text{if } i = j \text{ and } p' = \delta_P(p, L_i(q)) \\ 0 & \text{otherwise} \end{cases}$$

The index function in the game is defined according to the third component of the states and according to the index function α_P of the DPW, thus $\alpha_G(\langle q, i, p\rangle) = \alpha_P(p)$. The observation mapping is defined by

$\mathsf{obs}(\langle q, i, p\rangle) = i$ if $q \notin F_i$, and $\mathsf{obs}(\langle q, i, p\rangle) = \langle q, i\rangle$ if $q \in F_i$. Thus the state of the DPW is not observable, and only the components name and exit states are observable. Note that $\mathcal{O} = [k] \cup \bigcup_{i=0}^{k}(F_i \times \{i\})$ where $[k] = \{0, 1, 2, \ldots, k\}$ and the number of components is $k + 1$. The correctness argument is established using similar arguments as in Section 3.2, by showing that a composer for $\mathcal{L}$ can be mapped to a collapsed-stutter-invariant strategy in $G_{\mathcal{L}}$ that is almost-sure winning, and vice versa we can construct a composer for $\mathcal{L}$ from an almost-sure winning collapsed-stutter-invariant strategy in $G_{\mathcal{L}}$ by the inverse mapping.

This reduction and Theorem 4.1 show that the realizability problem with DPW specifications can be solved in EXPTIME, and an EXPTIME lower bound is known for this problem [2].

THEOREM 4.2. *The qualitative realizability problem for controlflow composition with DPW specifications is EXPTIME-complete.*

4.3 Undecidability of the Quantitative Realizability In this section we establish undecidability of the quantitative realizability problem by a reduction from the quantitative decision problem for probabilistic automata (which is undecidable).

Probabilistic automata. A probabilistic automaton $\mathcal{A} = \langle \Sigma, Q, q_0, \delta\rangle$ is a probabilistic transducer without outputs and exit states, i.e., Σ consists of the input letters, Q is the finite state space with initial state q_0, and $\delta : Q \times \Sigma \to \mathcal{D}(Q)$ is the probabilistic transition function. Consider a probabilistic automaton $\mathcal{A}$ with an index function α on Q. A word is an infinite sequence of letters from Σ, and a *lasso-shaped* word $w = w_1(w_2)^\omega$ consists of a finite word w_1 followed by an infinite repetition of a non-empty finite word w_2. Given a probabilistic automaton $\mathcal{A}$ with index function α, the quantitative decision problem of whether there exists a lasso-shaped word that is accepted with probability at least η is undecidable, for rational $\eta \in (0, 1)$ given as input [22], and the undecidability proof holds even for index function for reachability, Büchi, or coBüchi objectives.

The key ideas of reduction. The key ideas of the reduction are as follows. We consider a component for each letter of the input alphabet, and each component has a unique exit (i.e., $|D| = 1$). Hence a composer represents a choice of word, and since the state space of a composer is finite, a composer represents a lasso-shaped word. Conversely, for every lasso-shaped word there is a composer. In each component, in the starting state, there is a choice by the environment among the states of the probabilistic automaton, and given the choice of a state, the probabilistic transition is executed according to the current state and choice of letter (represented by the choice component), and finally there is a transition to the unique exit state. To ensure that the choice in each component really chooses the correct state of the probabilistic automaton we use the DPW. The DPW keeps track of the current state of the probabilistic automaton, and requires that the choice from the starting state of the component matches the current state (otherwise it accepts immediately).

The reduction. Consider a probabilistic automaton $\mathcal{A} = \langle \Sigma, Q, q_0, \delta\rangle$ with an index function α on Q. Let $\Sigma = \{0, 1, 2, \ldots, k\}$. We construct a library $\mathcal{L}$ with $k + 1$ components $M_0, M_1, \ldots, M_k$ each with a unique exit as follows. We have $M_i = \langle \Sigma_I, \Sigma_O, Q_i, q_0^i, \delta_i, F_i, L_i\rangle$ where the components are as follows:

1. $\Sigma_I = Q$; and $\Sigma_O = Q \cup \{\$\}$;
2. $Q_i = \{q_0^i\} \cup (Q \times \{1, 2\}) \cup \{\mathsf{ex}_i\}$;
3. $F_i = \{\mathsf{ex}_i\}$;
4. $L_i(q_0^i) = L_i(\mathsf{ex}_i) = \$$ and $L_i(\langle q, j\rangle) = q$ for $q \in Q$ and $j \in \{1, 2\}$; and
5. (a) $\delta_i(q_0^i, \sigma)(\langle \sigma, 1\rangle) = 1$ (i.e., given the start state and an input letter $\sigma = q$ corresponding to a state of Q, the next state is $\langle q, 1\rangle$); (b) $\delta_i(\langle q, 1\rangle, \sigma)(\langle q', 2\rangle) = \delta(q, i)(q')$ (i.e., irrespective of the input choice σ, the second component changes from 1 to 2, and the first component changes according to the transition function δ of $\mathcal{A}$ for the choice of input letter i); and (c) $\delta_i(\langle q, 2\rangle, \sigma)(\mathsf{ex}_i) = 1$ (i.e., irrespective of choice of σ the next state is the unique exit state).

The DPW. We now describe the DPW along with the library of components. The DPW has alphabet $Q \cup \{\$\}$ and state space $(Q \times \{0, 1, 2\}) \cup \{\top\}$. The transition function δ_P is as follows:

1. $\delta_P(\langle q,0\rangle,\sigma) = \langle q,1\rangle$ if $\sigma = q$, else if $\sigma \neq q$, then $\delta((q,0),\sigma) = \top$ (i.e., in a state where the second component is 0 the automaton expects to read the same input as the first component, and if it reads so it changes the second component from 0 to 1, otherwise it goes to the $\top$ state). This step corresponds to reading the choice from the start state of a component.
2. $\delta_P(\langle q,1\rangle,\sigma) = \langle \sigma,2\rangle$ (i.e., when the second component is 1 it updates the first component according to the transition read, and the second component changes from 1 to 2). This step corresponds to reading the transition in a component that mimics the transition of the probabilistic automaton.
3. $\delta_P(\langle q,2\rangle,\sigma) = \langle q,0\rangle$ (i.e., irrespective of the input, the first component remains the same, and the second component changes from 2 to 0). This step corresponds to reading the transition to the exit state in a component.

To be very precise, one also needs to add more states in the DPW for transition from the exit state of a component to the start state of the next component (which is omitted for simplicity). The state $\top$ is an absorbing state (irrespective of the input the next state is $\top$ itself). The index function α_P for the DPW maps according to the index function of α and the first component, i.e., for $\langle q,i\rangle$ where $q \in Q$ and $i \in \{0,1,2\}$ we have $\alpha_P(\langle q,i\rangle) = \alpha(q)$; and $\alpha_P(\top) = 0$.

Correctness argument. Given the probabilistic automaton $\mathcal{A}$ with index function α, let Φ_α be the corresponding parity objective. For a lasso-shaped word w, let $\mathbb{P}^w(\Phi_\alpha)$ denote the probability that the word satisfies the parity objective. Given a composer C_w that corresponds to a lasso-shaped word w, consider a strategy of the environment that given the starting state of a component always chooses the state according to the first component of the current state of the DPW. Then it follows that the probability distribution of $\mathcal{A}$ is executed, and hence we have $\mathbb{P}^w(\Phi_\alpha) \leq \mathsf{val}(\mathcal{T}_{C_w}, \Phi_{\alpha_P})$, where Φ_{α_P} is the parity objective induced by the DPW. Conversely, if the environment does not choose according to the current state of the DPW, then the DPW immediately accepts. It follows that $\mathbb{P}^w(\Phi_\alpha) \geq \mathsf{val}(\mathcal{T}_{C_w}, \Phi_{\alpha_P})$, and thus we have $\mathbb{P}^w(\Phi_\alpha) = \mathsf{val}(\mathcal{T}_{C_w}, \Phi_{\alpha_P})$. Hence the answer to the quantitative realizability problem is YES iff the answer to the quantitative decision problem for $\mathcal{A}$ is YES. We have the following result.

THEOREM 4.3. *The quantitative realizability problem for controlflow composition with DPW specifications is undecidable.*

References

[1] D. Andersson and P. B. Miltersen. The complexity of solving stochastic games on graphs. In *ISAAC'09*, pages 112–121, 2009.

[2] G. Avni and O. Kupferman. Synthesis from component libraries with costs. In *Proc. of CONCUR: Concurrency Theory*, LNCS 8704, pages 156–172. Springer, 2014.

[3] C. Baier, M. Größer, M. Leucker, B. Bollig, and F. Ciesinski. Controller synthesis for probabilistic systems. In *IFIP TCS'04*, pages 493–506, 2004.

[4] C. Baier and J.-P. Katoen. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008.

[5] D. Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Mecella. Automatic composition of e-services that export their behavior. In *ICSOC'03*, pages 43–58, 2003.

[6] K. Chatterjee. *Stochastic ω-regular Games*. PhD thesis, University of California, Berkeley, 2007.

[7] K. Chatterjee, L. Doyen, H. Gimbert, and T. A. Henzinger. Randomness for free. In *CoRR abs/1006.0673 (Full version)*, 2010. Conference version Proc. of MFCS, Springer, LNCS 6281, pages 246-257.

[8] K. Chatterjee, L. Doyen, S. Nain, and M. Y. Vardi. The complexity of partial-observation stochastic parity games with finite-memory strategies. In *Proc. of FOSSACS: Foundations of Software Science and Computation Structures*, LNCS 8412, pages 242–257. Springer, 2014.

[9] K. Chatterjee and N. Fijalkow. A reduction from parity games to simple stochastic games. In *Proc. of GANDALF: Games, Automata, Logics and Formal Verification*, volume 54 of *EPTCS*, pages 74–86, 2011.

[10] K. Chatterjee and M. Henzinger. Faster and dynamic algorithms for maximal end-component decomposition and related graph problems in probabilistic verification. In *Proc. of SODA: Symposium on Discrete Algorithms*, pages 1318–1336. SIAM, 2011.

[11] K. Chatterjee and M. Henzinger. Efficient and dynamic algorithms for alternating büchi games and maximal end-component decomposition. *J. ACM*, 61(3):15, 2014.

[12] K. Chatterjee and T. A. Henzinger. Reduction of stochastic parity to stochastic mean-payoff games. *Inf. Process. Lett.*, 106(1):1–7, 2008.

[13] K. Chatterjee, M. Jurdziński, and T. A. Henzinger. Simple stochastic parity games. In *Proc. of CSL: Computer Science Logic*, LNCS 2803, pages 100–113. Springer, 2003.

[14] K. Chatterjee, M. Jurdziński, and T. A. Henzinger. Quantitative stochastic parity games. In *Proc. of SODA: Symposium on Discrete Algorithms*, pages 114–123, 2004. Technical Report: UCB/CSD-3-1280 (October 2003).

[15] C. Courcoubetis and M. Yannakakis. Markov decision processes and regular events. In *ICALP 90: Automata, Languages, and Programming*, volume 443 of *Lecture Notes in Computer Science*, pages 336–349. Springer, 1990.

[16] C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *J. ACM*, 42(4):857–907, 1995.

[17] L. de Alfaro and T. A. Henzinger. Interface theories for component-based design. In *EMSOFT'01*, pages 148–165, 2001.

[18] M. Jurdzinski. Deciding the winner in parity games is in UP $\cap$ co-UP. *Information Processing Letters*, 68(3):119–124, 1998.

[19] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *CAV'11*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

[20] Y. Lustig and M. Y. Vardi. Synthesis from component libraries. In *FOSSACS'09*, pages 395–409, 2009.

[21] S. Nain, Y. Lustig, and M. Y. Vardi. Synthesis from probabilistic components. *Logical Methods in Computer Science*, 10(2), 2014.

[22] A. Paz. *Introduction to probabilistic automata*. Academic Press, Inc. Orlando, FL, USA, 1971.

[23] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. of POPL*, pages 179–190. ACM Press, 1989.

[24] J. Sifakis. A framework for component-based construction extended abstract. In *SFEM'05*, pages 293–300, 2005.

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

[26] M. Y. Vardi. Probabilistic linear-time model checking: An overview of the automata-theoretic approach. In *ARTS'99*, pages 265–276, 1999.