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 contribution for controlflow synthesis from probabilistic components is 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 [17], interface-based design [11]. The synthesis problem asks for the automated construction of a system given a logical specification. For example, in LTL (linear-time temporal logic) synthesis,

* This research was supported by Austrian Science Fund (FWF) Grant No P23499-N23, FWF NFN Grant No S11407-N23 (SHiNE), ERC Start grant (279307: Graph Games), EU FP7 Project Cassting, 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.

the specification is given in LTL and the reactive system to be constructed is a finite-state transducer [16]. 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 [13].

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 [13, 14] for a detailed discussion). As a basic model for a component, following [13], 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 [13], two models of composition were studied, namely, *dataflow* composition, where the output of one component 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, and the controlflow composition to be decidable [13].

Synthesis for probabilistic components. While [13] considered synthesis for non-probabilistic components which was extended to non-probabilistic recursive state components in [10], the study of synthesis for controlflow composition for probabilistic components was considered in [14]. 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 the construction of reliable systems from unreliable components. There is a rich literature about verification and analysis of such systems, cf. [18, 9, 19, 3, 12].

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

	Qualitative		Quantitative	
	Our Results	Previous Results	Our Results	Previous Results
Embedded Parity	UP ∩ coUP (Parity-games hard)	EXPTIME	UP ∩ coUP (Parity-games hard)	Open
DPW Specifications	EXPTIME-c	2EXPTIME	Undecidable	Open

Table 1. Computational complexity of synthesis from probabilistic components.

parity word automaton (DPW). The results of [14] 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 solve.

Our contributions. Our main contributions are (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.
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 [14]), 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. We present polynomial-time reductions for 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 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 stutter-invariant strategies and obtain the EXPTIME-complete result. Detailed proofs are available in [6].

2 Definitions

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.

Probabilistic transducers. Let $\mathcal{D}(X)$ denote the set of all probability distributions on set X. 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, and probability measure. 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 $h : Q \to \Sigma_I$. A strategy f along with a probabilistic transducer M, with set of states Q, induces a probability distribution on Q^ω, denoted μ_f (see [6] for detailed definition).

Library of components. A *library* is a finite 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.

Controlflow composition from libraries. We first informally describe the notion of controlflow composition of components from a library as defined in [14]. 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 calls 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. An explicit definition is presented in the full version of the paper. Note that the composition, denoted $\mathcal{T}_C$, is a probabilistic transducer without exits. When the composition $\mathcal{T}_C$ 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.

Parity objectives and values. 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 consisting of the set of infinite sequence of states such that the largest 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 $\inf_{f \in \mathcal{F}} \mu_f(\Phi)$, i.e., it is the minimal probability with which the parity objective is satisfied over all strategies in the transducer.

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

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.

Definition 1 (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*)* realizability problem for controlflow composition with embedded parity *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., $\mathsf{val}(\mathcal{T}_C, \Phi_\alpha) \geq \eta$, given rational $\eta \in (0,1)$). A witness composer for the qualitative (resp., quantitative) problem is called an almost-sure (resp., η-optimal) composer. The corresponding* embedded parity synthesis problems *are to find such a composer C if it exists.*

Synthesis for DPW specifications. A *deterministic parity word 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 $\inf_{f \in \mathcal{F}} \mu_f(\lambda^{-1}(L_A))$. The compatibility of the composer with an exit control relation can be encoded in the DPW (w.l.o.g., two distinct exit states do not have the same output).

Definition 2 (DPW realizability and synthesis). *Consider a library $\mathcal{L}$ and a DPW A that is a monitor for $\mathcal{L}$. The* qualitative *(resp.,* quantitative*)* realizability problem for controlflow composition with DPW specifications *is to decide whether there exists a composer C over $\mathcal{L}$, such that $\mathsf{val}(\mathcal{T}_C, A) = 1$ (resp., $\mathsf{val}(\mathcal{T}_C, A) \geq \eta$, given rational $\eta \in (0,1)$). A witness composer for the qualitative (resp., quantitative) problem is called an almost-sure (resp., η-optimal) composer. The corresponding* DPW probabilistic synthesis problems *are to find such a composer C if it exists.*

Remark 1. The realizability problem for libraries with components can be viewed as a 2-player partial-observation stochastic parity game [14]. 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 the same exit of M.

3 Realizability with Embedded Parity

We establish the results for the complexity of realizability with embedded parity. 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

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 $\delta^G(s_j, a_j)(s_{j+1}) > 0$. 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), denoted by $\mathbb{P}_s^{\pi,\tau}(\cdot)$.

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. 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 $\mathsf{val}^G(\pi, \Phi_\alpha)(s) = 1$. Theorem 1 summarizes results about perfect-information games.

Theorem 1. *The following assertions hold* [8, 4, 7, 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 UP ∩ 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

The upper-bound reduction. Consider a library $\mathcal{L}$ of width D, an 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 $0 \leq i \leq k$, where $F_i = \{q_x^i : x \in D\}$. Let $[k] = \{0, 1, 2, \ldots, k\}$. 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 $\sigma \in A_2$, we have $\delta_{\mathcal{L}}^G(\langle q, i \rangle, \sigma)(\langle q', j \rangle) = \delta_i(q, \sigma)(q')$ if $i = j$, and 0 otherwise; and (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 $\bot$. For all $\langle q, i \rangle \in S \setminus \{\bot\}$ define $\alpha_G(\langle q, i \rangle) = \alpha(q)$, and let Φ_{α_G} be the parity objective in $G_{\mathcal{L}}$.

Lemma 1. *Consider a library* $\mathcal{L}$ *of width* D*, an 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 if and only if 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 if and only if the value in $G_{\mathcal{L}}$ at $\langle q_0^0, 0\rangle$ is at least η.

Proof sketch. There are two steps to establish correctness of the reduction. The first step is given a composer over $\mathcal{L}$ 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. But we show that a pure memoryless strategy in $G_{\mathcal{L}}$ can be converted to a composer.

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.

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. 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 $\mathcal{T}_{C_\pi}$.

Theorem 2 (Complexity of embedded parity realizability). *The qualitative and quantitative realizability problems for controlflow composition with embedded parity belong to UP $\cap$ coUP, and are at least as hard as the (almost-sure) decision problem for perfect-information deterministic parity games.*

4 Realizability with 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

We consider partial-observation games with restrictions on strategies that correspond to the qualitative realizability problem, and present a new result to solve such games.

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 two 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; and (ii) the play prefixes have the same sequence of distinct observations, that is they have the same observation sequence up to repetition (stuttering).

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 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 two 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 $\pi(\rho) = \pi(\rho')$;
- *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 $\pi(\rho) = \pi(\rho')$.

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, which is EXPTIME-complete [5].

Reduction of collapsed-stutter-invariant problem to observation-based problem. There are two main ideas of the reduction. (1) First, 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. (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 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 wins the parity objective if the loop is taken forever by player 2, and otherwise, visiting priority 0 infinitely often does not change the winner of the game.

The formal reduction. 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 $0 \notin A_1$. 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.

- $S'_1 = S_1 \times (A_1 \cup \{0\}) \cup \{\bot\}$; $S'_2 = S' \setminus S'_1$; and $A'_2 = A_2 \cup \{\sharp\}$, assuming $\sharp \notin A_2$.
- 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 $\alpha'(\bot) = 1$. Hence, the state $\bot$ 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 2. *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 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 $\Phi_{\alpha'}$.*

Theorem 3. *The qualitative problem of deciding the existence of a finite-memory almost-sure winning collapsed-stutter-invariant strategy in partial-observation stochastic games with parity objectives is EXPTIME-complete.*

4.2 Qualitative and Quantitative Realizability

We present a polynomial reduction for the qualitative realizability problem with DPW specifications to the existence of finite-memory collapsed-stutter-invariant almost-sure winning strategies, and thus show that the problem can be solved in EXPTIME. An EXPTIME lower bound is known for this problem [2].

Theorem 4. *The qualitative realizability problem for controlflow composition with DPW specifications is EXPTIME-complete.*

Finally, we establish undecidability of the quantitative realizability problem by a reduction from the quantitative decision problem for probabilistic automata (which is undecidable [15]).

Theorem 5. *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 *CONCUR'14*, LNCS 8704, pages 156–172. Springer, 2014.
3. C. Baier and J.-P. Katoen. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008.
4. K. Chatterjee. *Stochastic ω-regular Games*. PhD thesis, UC Berkeley, 2007.
5. K. Chatterjee, L. Doyen, S. Nain, and M. Y. Vardi. The complexity of partial-observation stochastic parity games with finite-memory strategies. In *FOSSACS'14*, LNCS 8412, pages 242–257. Springer, 2014.
6. K. Chatterjee, L. Doyen, and M. Y. Vardi. The complexity of synthesis from probabilistic components. *CoRR*, abs/1502.04844, 2015.
7. K. Chatterjee and T. A. Henzinger. Reduction of stochastic parity to stochastic mean-payoff games. *IPL*, 106(1):1–7, 2008.
8. K. Chatterjee, M. Jurdziński, and T. A. Henzinger. Quantitative stochastic parity games. In *SODA'04*, pages 114–123, 2004.
9. C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *J. ACM*, 42(4):857–907, 1995.
10. I. De Crescenzo and S. La Torre. Modular synthesis with open components. In *RP*, pages 96–108, 2013.
11. L. de Alfaro and T. A. Henzinger. Interface theories for component-based design. In *EMSOFT'01*, pages 148–165, 2001.
12. M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *CAV'11*, *LNCS 6806*, pages 585–591. Springer, 2011.
13. Y. Lustig and M. Y. Vardi. Synthesis from component libraries. In *FOSSACS'09*, pages 395–409, 2009.
14. S. Nain, Y. Lustig, and M. Y. Vardi. Synthesis from probabilistic components. *LMCS*, 10(2), 2014.
15. A. Paz. *Introduction to probabilistic automata*. Academic Press, Inc., 1971.
16. A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. of POPL*, pages 179–190. ACM Press, 1989.
17. J. Sifakis. A framework for component-based construction extended abstract. In *SFEM'05*, pages 293–300, 2005.
18. M. Y. Vardi. Automatic verification of probabilistic concurrent finite-state systems. In *FOCS'85*, pages 327–338, 1985.
19. M. Y. Vardi. Probabilistic linear-time model checking: An overview of the automata-theoretic approach. In *ARTS'99*, pages 265–276, 1999.