Composition of Stochastic Services for LTL_f Goal Specifications

Giuseppe De Giacomo[1,2], Marco Favorito[3], and Luciana Silo[2,4](✉)

[1] University of Oxford, Oxford, UK
[2] Sapienza University of Rome, Rome, Italy
{degiacomo,silo}@diag.uniroma1.it
[3] Banca d'Italia, Rome, Italy
marco.favorito@bancaditalia.it
[4] Camera dei Deputati, Rome, Italy

Abstract. Service composition *à la* Roman model consists of realizing a virtual service by orchestrating suitably a set of already available services. In this paper, we consider a variant where available services are stochastic systems, and the target specification is goal-oriented and specified in Linear Temporal Logic on finite traces (LTL_f). In this setting, we are interested in synthesizing a controller (policy) that maximizes the probability of satisfaction with the goal, while minimizing the expected cost of the utilization of the available services. To do so, we combine techniques from LTL_f synthesis, service composition *à la* Roman Model, reactive synthesis, and bi-objective lexicographic optimization on Markov Decision Processes (MDPs). This framework has several interesting applications, including Smart Manufacturing and Digital Twins.

Keywords: Service Composition · Linear Temporal Logic on finite traces · Markov Decision Process · Lexicographic Multi-Objective Optimization

1 Introduction

The service-oriented computing (SOC) paradigm uses services to support the development of rapid, low-cost, interoperable, evolvable, and massively distributed applications. Services are considered autonomous, platform-independent entities that can be described, published, discovered, and loosely coupled in novel ways [29]. Service composition, i.e., the ability to generate new, more useful services from existing ones, is an active field of research in the SOC area and has been actively investigated for over two decades.

Particularly interesting in this context is the so-called Roman Model [2,3,16] where services are *conversational*, i.e., have an internal state and are procedurally described as finite transition systems (TS), where at each state the service offers a certain set of actions, and each action changes the state of the service in some way. The designer is interested in generating a new service, called *target*, which is described as the other service; however, it is virtual in the sense that no code is associated with its actions. So, for executing the target, one has to delegate each of its actions to some of the available services by suitably orchestrating the services, considering the current state of the target and the current states of the available services. Service composition amounts to synthesizing a controller that can suitably orchestrate the executions of the available services to guarantee that the target actions are always delegated to some service that can actually execute them in its current state. The original paper on the Roman Model [2] was awarded as the most influential paper of the decade at ICSOC 2013 and is, to date, the most cited paper in the ICSOC conference series.

Recently a renewed interest in service composition *à la* Roman Model is stemming out of applications in smart manufacturing, where, through digital twins technology, manufacturing devices can export their behaviour as transition systems and hence being orchestrated very much in the same way as service did back in the early 2000s, see e.g., [15,26,27].

Interestingly, these new applications are also pointing out several variations that are not typically considered in earlier literature on services. First, they advocate for considering a stochastic behaviour of services, such as those studied in [4,37]. Unlike the classical model, in which the target specification can either be satisfied or not with no middle ground, in the stochastic setting, it is possible to define a notion of "approximate solution" in case an exact one does not exist.

Second, the notion of goal-oriented target specification is increasingly championed [25–27]. That is, instead of having the target specified as a transition system, it is specified as a (possibly temporally extended) goal that the composition has to fulfill. Of particular interest are specifications in Linear Temporal Logic on finite traces (LTL$_f$) [18], which are at the base of declarative process specification in Business Process Management (BPM) through the so-called DECLARE paradigm [19,20,30].

Third, apart from satisfying the target, it is of interest to also minimize cost coming from the service utilization [12–14]. This concern, together with the satisfaction of the target, calls for resorting to Multi-Objective Optimization for computing a solution.

In this paper, we address the three above requirements and study goal-oriented stochastic service composition where as goals we adopt arbitrary LTL$_f$ specifications, under the DECLARE assumption of a single action executed at the time (see later). Specifically, we are given a goal specification, and we want to synthesize an orchestrator that, on the one hand, *proactively* chooses actions to form a sequence that satisfies the goal and, on the other hand, delegates each action to an available service in such a way that at the end of the sequence, all services are in their final states. The composition problem consists of maximizing the satisfaction probability of the LTL$_f$ objective, and conditioned on this, minimizing the expected cost of utilization of the services.

A first attempt to do so may resort to Multi-objective MDPs (MOMDPs) [8]. One common solution is to reduce the multi-objective reward/cost optimization to a single reward optimization via a linear weighting of different sources of rewards/costs. However, this means that the two objectives, namely the maximization of target rewards and the minimization of cost uses, are blurred into one scalar value, which hides precious information from the agent. Instead, the maximization of the target objective has the *highest priority*. Among those strategies that maximize the first objective, we aim to find those strategies that achieve the minimum utilization cost. In the literature of multi-objective optimization, the setting in which there is a strict preference order among objectives is called *lexicographic multi-objective optimization* [7,22,34,36]. It is known that, in general, single Markovian rewards cannot capture certain multi-objective tasks, such as ones with lexicographic preferences [33]; hence, such problem cannot be easily reduced to standard techniques on MDPs [31].

Among related works, one of the earliest attempts at combining stochastic planning models with service composition is [21]. There are works based on Markov-HTN Planning [9], multi-objective optimization [10,28], and lexicographic optimization [32], helpful to model the stochastic behaviour as well as complex QoS preferences. However, in all cases, either there are no stateful services, no high-level declarative specification of the desired solution, or no strict preference among objectives.

Our solution technique relies on solving a bi-objective lexicographic optimization [6] over a special MDP, allowing to minimize the services' utilization costs while guaranteeing maximum probability of goal satisfaction. We point out that although this paper has mainly a foundational nature, it also has a significant applicative interest since it gives the foundations and solution techniques of goal-oriented compositions, which are indeed envisioned in the current literature on smart manufacturing where the notion of goal-oriented target specification is increasingly championed [15,25–27].

The rest of the paper is structured as follows. Section 2 explains the theoretical concepts on which the paper is based. Section 3 introduces the service composition framework in stochastic settings that we model, showing the formalization of the problem in terms of bi-objective lexicographic optimization on MDPs and proposes a solution technique able to find an optimal orchestrator. Section 4 shows the application of the formal framework to an industrial case study of an electric motor assembly process, describing in detail the manufacturing goal and the manufacturing actors. Finally, Sect. 5 concludes the paper with final remarks and future works.

2 Preliminaries

LTL$_f$ is a variant of Linear Temporal Logic (LTL) interpreted over finite traces, instead of infinite ones [18]. Given a set $\mathcal{P}$ of atomic propositions, LTL$_f$ formulas φ are defined by $\varphi ::= a \mid \neg\varphi \mid \varphi \wedge \varphi \mid \bigcirc\varphi \mid \varphi\,\mathcal{U}\,\varphi$, where a denotes an atomic proposition in $\mathcal{P}$, $\bigcirc$ is the next operator, and $\mathcal{U}$ is the until operator. We use abbreviations for other Boolean connectives, as well as the following: eventually as $\Diamond\varphi \equiv \mathit{true}\,\mathcal{U}\,\varphi$; always as $\Box\varphi \equiv \neg\Diamond\neg\varphi$; weak next as $\bullet\varphi \equiv \neg\bigcirc\neg\varphi$ (note that, on finite traces, $\neg\bigcirc\varphi$ is not equivalent to $\bigcirc\neg\varphi$); and weak until as $\varphi_1\,\mathcal{W}\,\varphi_2 \equiv (\varphi_1\,\mathcal{U}\,\varphi_2 \vee \Box\varphi_1)$ (φ_1 holds until φ_2 or forever). LTL$_f$ formulas are interpreted on finite traces $a = a_0 \dots a_{l-1}$ where a_i at instant i is a propositional interpretation over the alphabet $2^{\mathcal{P}}$, and l is the length of the trace. An LTL$_f$ formula can be transformed into equivalent *nondeteministic automata* (NFA) in at most EXPTIME and into an equivalent and *deterministic finite automata* (DFA) in at most 2EXPTIME [18]. A DFA is a tuple $\mathcal{A}_\varphi = \langle 2^{\mathcal{P}}, Q, q_0, F, \delta \rangle$ where: (*i*) $2^{\mathcal{P}}$ is the alphabet, (*ii*) Q is a finite set of states, (*iii*) q_0 is the initial state, (*iv*) $F \subseteq Q$ is the set of accepting states and (*v*) $\delta : Q \times 2^{\mathcal{P}} \to Q$ is a total transition function. Note that the DFA alphabet is the same as the set of traces that satisfies the formula φ. An NFA is defined similarly to DFA except that δ is defined as a relation, i.e. $\delta \subseteq Q \times 2^{\mathcal{P}} \times Q$. In particular, LTL$_f$ is used in declarative process specification in BPM, for example in the system DECLARE [30]. In this specific case, it is assumed that only one proposition (corresponding to an action) is true at every time point. We call this the DECLARE *assumption*, and we do adopt it in this paper.

Markov Decision Process. A *Markov Decision Process* (MDP) is a tuple $M = (S, A, P, s_0)$, where: (*i*) S is a finite set of states, (*ii*) s_0 is the initial state, (*iii*) A is a finite set of actions, and (*iv*) $P : (S \times A) \to \varDelta(S)$ is the transition probability function, i.e. a mapping from state-action pairs to probability distributions over S. With $\mathsf{Supp}(d)$, we denote the support of a probability distribution d. An infinite path $\rho \in (S \times A)^\omega$ is a sequence $\rho = s_0 a_1 s_1 a_2 \dots$, where $s_i \in S$, $a_{i+1} \in A$, and $s_{i+1} \in \mathsf{Supp}(P(s_i, a_{i+1}))$, for all $i \in \mathbb{N}$. Similarly, a finite path $\rho \in (S \times A)^* \times S$ is a finite sequence $\rho = s_0 a_1 s_1 a_2 \dots a_m s_m$. For any path ρ of length at least j and any $i \leq j$, we let $\rho[i : j]$ denote the subsequence $s_i a_{i+1} s_{i+1} a_{i+2} \dots a_j s_j$. We use $\mathsf{Paths}^\omega_{\mathcal{M}} = (S \times A)^\omega$ and $\mathsf{Paths}_{\mathcal{M}} = (S \times A)^* \times S$ to denote the set of infinite and finite paths, respectively. A policy $\pi : \mathsf{Paths}_{\mathcal{M}} \to A$ maps a finite path $\rho \in \mathsf{Paths}_{\mathcal{M}}$ to an action $a \in A$. We denote with $\mathsf{Paths}_{\mathcal{M}_\pi}$ the set of finite paths over $\mathcal{M}$ whose actions are compatible with π. Given a finite path $\rho = s_0 a_1 \dots a_m s_m$, the *cylinder* of ρ, denoted by $\mathsf{Paths}^\omega_{\mathcal{M}}(\rho)$, is the set of all infinite paths starting with prefix ρ. The σ-algebra associated with MDP $\mathcal{M}$ and a fixed policy π is the smallest σ-algebra that contains the cylinder sets $\mathsf{Paths}^\omega_{\mathcal{M}_\pi}(\rho)$ for all $\rho \in \mathsf{Paths}_{\mathcal{M}_\pi}$. For a state s in S, a measure is defined for the cylinder sets as $\mathbb{P}_{\mathcal{M}_\pi,s}(\mathsf{Paths}^\omega_{\mathcal{M}_\pi}(s_0 a_1 s_1 \dots a_m s_m)) = \prod_{k=0}^{m-1} P(s_{k+1} | s_k, a_{k+1})$ if $s_0 = s$ and for all k, $a_{k+1} = \pi(\rho[0 : k])$, otherwise 0. We also have $\mathbb{P}_{\mathcal{M}_\pi,s}(\mathsf{Paths}^\omega_{\mathcal{M}_\pi,s}(s)) = 1$ and $\mathbb{P}_{\mathcal{M}_\pi,s}(\mathsf{Paths}^\omega_{\mathcal{M}_\pi,s}(s')) = 0$ for $s' \neq s$. This can be extended to a unique probability measure $\mathbb{P}_{\mathcal{M}_\pi,s}$ on the aforementioned σ-algebra. In particular, if $\mathcal{R} \subseteq \mathsf{Paths}_{\mathcal{M}_\pi}$ is a set of finite paths forming pairwise disjoint cylinder sets, then $\mathbb{P}_{\mathcal{M}_\pi,s}(\bigcup_{\rho \in \mathcal{R}} \mathsf{Paths}^\omega_{\mathcal{M}_\pi}(\rho)) = \sum_{\rho \in \mathcal{R}} \mathbb{P}_{\mathcal{M}_\pi,s}(\mathsf{Paths}^\omega_{\mathcal{M}_\pi}(\rho))$. We denote with $\mathbb{E}_{\mathcal{M}_\pi,s}(X)$ the expected value of a random variable X with respect to the distribution $\mathbb{P}_{\mathcal{M}_\pi,s}$.

3 Composition of Stochastic Services for LTL$_f$ Tasks

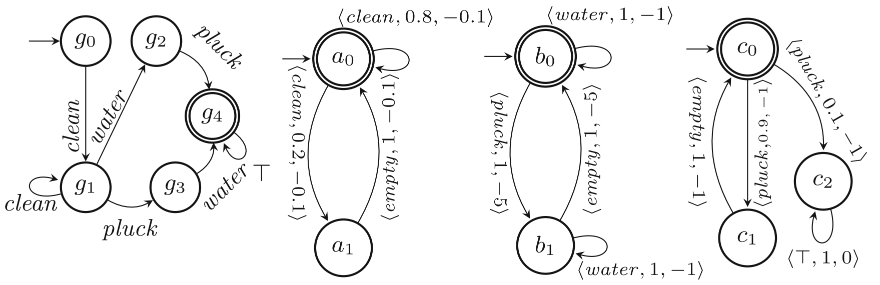
In this section, we present our service composition framework in stochastic settings. We aim to realize an LTL$_f$ goal specification with the available services by modeling nondeterminism using probability distributions over the services' successor states. Moreover, we allow the specification to be approximately satisfied, by considering the objective of maximizing the satisfaction probability of the specification.

3.1 Stochastic Services Framework

A stochastic service is a tuple $\tilde{\mathcal{S}} = \langle \Sigma, A, \sigma_0, F, P, C \rangle$, where: (*i*) Σ is the finite set of service states, (*ii*) A is the finite set of services' actions, (*iii*) $\sigma_0 \in \Sigma$ is the initial state, (*iv*) $F \subseteq \Sigma$ is the set of final states (i.e., states in which the computation may stop but does not necessarily have to), (*v*) $P : \Sigma \times A \to \Delta(\Sigma)$ is the *transition function* that returns for every state σ and action a a distribution over next states, and (*vi*) $C : \Sigma \times A \to \mathbb{R}^+$ is the *cost function* that assigns a (strictly positive) cost to each state-action pair. A *(stochastic) service community* is a collection of stochastic services $\tilde{\mathcal{C}} = \{\tilde{\mathcal{S}}_1, \dots, \tilde{\mathcal{S}}_n\}$. A *(stochastic) trace of* $\tilde{\mathcal{C}}$ is a finite alternating sequence of the form $\tilde{t} = (\sigma_{10} \dots \sigma_{n0}), (a_1, o_1), \dots, (a_m, o_m), (\sigma_{1m} \dots \sigma_{nm})$, where σ_{i0} is the initial state of every service $\mathcal{S}_i$ and, for every $1 \leq k \leq m$, we have (*i*) $\sigma_{ik} \in \Sigma_i$ for all $i \in \{1, \dots, n\}$, (*ii*) $o_k \in \{1, \dots, n\}$ is the service index chosen by the orchestrator at step k, (*iii*) $a_k \in A$, and (*iv*) for all i, $\sigma_{ik} \in \mathsf{Supp}(P_i(\sigma_{i,k-1}, a_{ik}))$ if $o_k = i$, and $\sigma_{ik} = \sigma_{i,k-1}$ otherwise. A *history of* $\mathcal{C}$ is a finite prefix of a trace of $\mathcal{C}$. With $|h| = m$, we denote the length of such history, and with $\mathsf{last}(h)$ we denote the service state configuration at the last step: $(\sigma_{1m} \dots \sigma_{nm})$. Given a trace t, we call $\mathsf{states}(t)$ *sequence of states* of t, i.e. $\mathsf{states}(t) = (\sigma_{10} \dots \sigma_{n0}), (\sigma_{11} \dots \sigma_{n1}), \cdots$. The *choices* of a trace t, denoted with $\mathsf{choices}(t)$, is the sequence of actions in t, i.e. $\mathsf{choices}(t) = (a_1, o_1), (a_m, o_m), \dots$. Note that, due to nondeterminism, there might be many traces of $\mathcal{C}$ associated with the same run. Moreover, we define the *action run* of a trace t, denoted with $\mathsf{actions}(t)$, the projection of $\mathsf{choices}(t)$ only to the components in A. states, $\mathsf{choices}$ and $\mathsf{actions}$ are defined also on history h, in a similar way. Note that both $\mathsf{choices}(h)$ and $\mathsf{actions}(h)$ are empty if $h = (\sigma_{10} \dots \sigma_{n0})$.

An *orchestrator* is a function $\gamma : (\Sigma_1 \times \cdots \times \Sigma_n)^* \to A \times \{1 \dots n\}$ that, given a sequence of states $(\sigma_{10} \dots \sigma_{n0}) \dots (\sigma_{1m} \dots \sigma_{nm})$, returns the action to perform $a \in A$, and the service (actually the service index) that will perform it. Next, we define when an orchestrator is a composition that satisfies φ. Given a trace t, with $\mathsf{histories}(t)$, we denote the set of prefixes of the trace t that ends with a services state configuration. A trace t is an *execution* of an orchestrator γ over $\mathcal{C}$ if for all $k \geq 0$, we have $(a_{k+1}, o_{k+1}) = \gamma((\sigma_{10} \dots \sigma_{n0}) \dots (\sigma_{1k} \dots \sigma_{nk}))$. Let $\mathcal{T}_{\gamma,\tilde{\mathcal{C}}}$ be the set of such executions. Note that due to the nondeterminism of the services, we can have many executions for the same orchestrator, despite the orchestrator being a deterministic function. If $h \in \mathsf{histories}(t)$ for some (infinite) execution $t \in \mathcal{T}_{\gamma,\tilde{\mathcal{C}}}$, we call h a finite execution of γ over $\mathcal{C}$.

Consider a goal specification φ expressed in LTL$_f$ over the set of actions A, and consider a community of n services $\mathcal{C} = \{\mathcal{S}_1, \dots, \mathcal{S}_n\}$, where each set of actions $A_i \subseteq A$. We say that some finite execution h is *successful*, denoted with $\mathsf{successful}(h)$, if the following two conditions hold: (1) $\mathsf{actions}(h) \models \varphi$, and (2)


g_0
g_2
pluck
g_4
clean
water
g_1
g_3
water
$\top$
clean
pluck
$\langle clean, 0.8, -0.1\rangle$
a_0
$\langle clean, 0.2, -0.1\rangle$
$\langle empty, 1, -0.1\rangle$
a_1
$\langle water, 1, -1\rangle$
b_0
$\langle pluck, 1, -5\rangle$
$\langle empty, 1, -5\rangle$
b_1
$\langle water, 1, -1\rangle$
c_0
$\langle empty, 1, -1\rangle$
$\langle pluck, 0.9, -1\rangle$
$\langle pluck, 0.1, -1\rangle$
c_2
c_1
$\langle \top, 1, 0\rangle$

(a) The DFA of φ. (b) Service $\mathcal{B}_1$. (c) Service $\mathcal{B}_2$. (d) Service $\mathcal{B}_3$.

Fig. 1. The garden bot systems and the DFA of the LTL$_f$ goal.

all service states $\sigma_i \in \mathsf{last}(\mathsf{states}(h))$ are such that $\sigma_i \in F_i$. If for execution $t \in \mathcal{T}_{\gamma,\tilde{\mathcal{C}}}$ there exist a finite prefix history $h \in \mathsf{histories}(t)$ such that $\mathsf{successful}(h)$, we say that t is successful. Finally, we say that an orchestrator γ *realizes the* LTL$_f$ *specification* φ *with* $\mathcal{C}$ if, for all traces $t \in \mathcal{T}_{\gamma,\tilde{\mathcal{C}}}$, t is successful.

We are interested in orchestrators that maximize the probability of satisfaction of the goal specification, even when the specification cannot be surely satisfied (e.g. due to a stochastic misbehaviour of some service). Moreover, while guaranteeing the optimal probability of satisfaction, we aim to find those orchestrators that minimize the expected utilization cost of the services, conditioned on the achievement of the task.

Example 1. This example is inspired by the "garden bots system" scenario [37]. The goal is to *clean* the garden by picking fallen leaves and removing dirt, *water* the plants, and *pluck* the ripe fruits and flowers. The action *clean* must be performed at least once, followed by *water* and *pluck* in any order. In DECLARE LTL$_f$, the goal can be expressed as $\varphi = clean \wedge \bigcirc(clean\,\mathcal{U}((water \wedge \bigcirc pluck) \vee (pluck \wedge \bigcirc water)))$. We assume there are three available garden bots, $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$, each with different capabilities and action rewards. In Fig. 1 the three services specifications and the DFA of the LTL$_f$ goal are shown. Transitions labels are of the form $\langle action, prob, reward\rangle$. We are interested in a composition of the bots to maximize the probability of the satisfaction of the goal φ, which also considers rewards/costs. The *clean* action can only be delegated to $\mathcal{B}_1$, and the optimal solution must take into account its stochastic behaviour in order to correctly compute the expected cost. Regarding the *pluck* action, both $\mathcal{B}_2$ and $\mathcal{B}_3$ can perform it; however, the optimal orchestrator will not dispatch it to $\mathcal{B}_3$ because, despite the cost being smaller than the one in $\mathcal{B}_2$, choosing $\mathcal{B}_3$ will lead to a probability of 0.1 of not reaching the final state configuration since the state c_2 is a failure state while choosing $\mathcal{B}_3$ does not compromise the optimal probability of goal satisfaction.

Before proceeding with a formalization of the optimization problem, we introduce additional auxiliary notions. Analogously to what has been done for MDPs, for a finite execution h of γ over $\tilde{\mathcal{C}}$, we use $\mathcal{T}_{\gamma,\tilde{\mathcal{C}}}(h)$ to denote the set of all (infinite) executions $t \in \mathcal{T}_{\gamma,\tilde{\mathcal{C}}}$ such that $h \in \mathsf{histories}(t)$. Moreover, the σ-algebra associated with the stochastic behaviour of the orchestrator γ over the stochastic community $\tilde{\mathcal{C}}$ is the smallest σ-algebra that contains the trace sets $\mathcal{T}_{\gamma,\tilde{\mathcal{C}}}(h)$, for all finite executions h, with the unique probability measure over it defined as:

$$\mathbb{P}_{\gamma,\tilde{\mathcal{C}}}(h) = \prod_{k=1}^{|h|} P_{o_k}(\sigma_{o_k,k} \mid \sigma_{o_k,k-1}, a_k) \tag{1}$$

In particular, note that $\mathbb{P}_{\gamma,\tilde{\mathcal{C}}}(\mathcal{T}_{\gamma,\tilde{\mathcal{C}}}(\langle(\sigma_{10} \dots \sigma_{n0})\rangle)) = 1$. Let $\mathcal{H}^{\varphi}_{\gamma,\tilde{\mathcal{C}}}$ be the set of finite executions h of γ on $\tilde{\mathcal{C}}$ that start from $\sigma_{10} \dots \sigma_{n0}$ such that (i) they are successful, and (ii) there is no prefix history h that is successful. Intuitively, such a set only contains the executions that are successful for the first time. The *satisfaction probability* of φ under orchestrator γ and community $\tilde{\mathcal{C}}$ is given by:

$$\mathcal{P}^{\tilde{\mathcal{C}}}_{\varphi}(\gamma) = \mathbb{P}_{\gamma,\tilde{\mathcal{C}}}\left(\bigcup_{h \in \mathcal{H}^{\varphi}_{\gamma,\tilde{\mathcal{C}}}} \mathcal{T}_{\gamma,\tilde{\mathcal{C}}}(h)\right) \tag{2}$$

It is crucial to observe that since by definition there is no pair $h', h'' \in \mathcal{H}^{\varphi}_{\gamma,\tilde{\mathcal{C}}}(h)$ such that $h' \in \mathsf{prefixes}(h'')$, all trace sets $\mathcal{T}_{\gamma,\tilde{\mathcal{C}}}(h)$ for $h \in \mathcal{H}^{\varphi}_{\gamma,\tilde{\mathcal{C}}}$ are pairwise disjoint sets, which means that $\mathcal{P}^{\tilde{\mathcal{C}}}_{\gamma}$ is a well-defined probability.

Moreover, we define the *(conditioned) expected utilization cost* of services as the expected cost an orchestrator incurs in its successful executions, i.e.:

$$\mathcal{J}^{\tilde{\mathcal{C}}}_{\varphi}(\gamma) = \mathbb{E}_{h \sim \mathbb{P}_{\gamma,\tilde{\mathcal{C}}}}\left[\sum_{k=1}^{|h|} C_{o_k}(\sigma_{o_k,k-1}, a_k) \middle| \, \mathsf{successful}(h)\right] \tag{3}$$

Let $\Gamma(\tilde{\mathcal{C}})$ be the set of orchestrators for the community $\tilde{\mathcal{C}}$. Let $f : \Gamma(\tilde{\mathcal{C}}) \to \mathbb{R}$ be an objective function. We say an orchestrator $\gamma \in \Gamma(\tilde{\mathcal{C}})$ is f*-optimal* if $f(\gamma) = \sup_{\tau \in \Gamma(\tilde{\mathcal{C}})} f(\tau)$, and write Γ_f for the set of f-optimal orchestrators.

Finally, we define our optimization problem. We want to compute an orchestrator γ such that the following holds:

$$\gamma \in \Gamma_{\mathcal{P}^{\tilde{\mathcal{C}}}_{\varphi}} \text{ and } \mathcal{J}^{\tilde{\mathcal{C}}}_{\varphi}(\gamma) = \inf_{\gamma' \in \Gamma_{\mathcal{P}^{\tilde{\mathcal{C}}}_{\varphi}}} \mathcal{J}^{\tilde{\mathcal{C}}}_{\varphi}(\gamma') \tag{4}$$

Intuitively, we fix a lexicographic order on the objective functions $\mathcal{P}^{\tilde{\mathcal{C}}}_{\varphi}$ and $\mathcal{J}^{\tilde{\mathcal{C}}}_{\varphi}$, meaning that we aim to minimize the expected utilization cost to satisfy the specification, conditioned to the satisfaction of the specification, while guaranteeing the optimal probability of satisfying it. Interestingly, in case the specification is *exactly* realizable, the notion of optimal orchestrator according to Eq. (4) coincides with the notion of realizability, as shown in the following results.

Lemma 1. *If γ realizes the specification φ over $\tilde{\mathcal{C}}$, then $\bigcup_{h \in \mathcal{H}^{\varphi}_{\gamma,\tilde{\mathcal{C}}}} \mathcal{T}_{\gamma,\tilde{\mathcal{C}}}(h) = \mathcal{T}_{\gamma,\tilde{\mathcal{C}}}(\langle(\sigma_{10} \dots \sigma_{n0})\rangle)$.*

Proof. We prove (i) $\bigcup_{h\in\mathcal{H}^{\varphi}_{\gamma,\tilde{\mathcal{C}}}} \mathcal{T}_{\gamma,\tilde{\mathcal{C}}}(h) \subseteq \mathcal{T}_{\gamma,\tilde{\mathcal{C}}}(\langle(\sigma_{10}\dots\sigma_{n0})\rangle)$ and (ii) $\bigcup_{h\in\mathcal{H}^{\varphi}_{\gamma,\tilde{\mathcal{C}}}} \mathcal{T}_{\gamma,\tilde{\mathcal{C}}}(h) \supseteq \mathcal{T}_{\gamma,\tilde{\mathcal{C}}}(\langle(\sigma_{10}\dots\sigma_{n0})\rangle)$ separately. Proposition (i) is immediate: every execution belongs to the set of executions starting from $\sigma_{10}\dots\sigma_{n0}$. To prove proposition (ii), we start by observing that for all $t\in\mathcal{T}_{\gamma,\tilde{\mathcal{C}}}(\langle(\sigma_{10}\dots\sigma_{n0})\rangle)$, by definition of realizing orchestrator, they have a prefix $h'\in \mathsf{prefixes}(h)$ that is successful. In particular, if h'' is the shortest prefix of h that is successful, then $h''\in\mathcal{H}^{\varphi}_{\gamma,\tilde{\mathcal{C}}}$ and $t\in\mathcal{T}_{\gamma,\tilde{\mathcal{C}}}(h'')$. This implies that $t\in\bigcup_{h''\in\mathcal{H}^{\varphi}_{\gamma,\tilde{\mathcal{C}}}}\mathcal{T}_{\gamma,\tilde{\mathcal{C}}}(h'')$.

Theorem 1. *Let $\tilde{\mathcal{C}}$ be a community of stochastic services, and φ be a goal specification. The orchestrator γ realizes φ with community $\tilde{\mathcal{C}}$ iff $\mathcal{P}^{\tilde{\mathcal{C}}}_{\varphi}(\gamma)=1$.*

Proof. $(\Rightarrow)$ If an orchestrator γ realizes φ, then all infinite executions $t\in\mathcal{T}_{\gamma,\tilde{\mathcal{C}}}$ have a prefix $h'\in\mathsf{histories}(t)$ that is successful. Let h'' be the shortest of such prefixes. This implies that $t\in\bigcup_{h''\in\mathcal{H}^{\varphi}_{\gamma,\tilde{\mathcal{C}}}}\mathcal{T}_{\gamma,\tilde{\mathcal{C}}}(h'')$. By Lemma 1, this set is equal to $\mathcal{T}_{\gamma,\tilde{\mathcal{C}}}(\langle(\sigma_{10}\dots\sigma_{n0})\rangle)$. Since by definition $\mathsf{Supp}(\mathbb{P}_{\gamma,\tilde{\mathcal{C}}})\subseteq\mathcal{T}_{\gamma,\tilde{\mathcal{C}}}(\langle(\sigma_{10}\dots\sigma_{n0})\rangle)$, we have that $\mathcal{P}^{\tilde{\mathcal{C}}}_{\varphi}(\gamma)=\mathbb{P}_{\gamma,\tilde{\mathcal{C}}}(\bigcup_{h\in\mathcal{H}^{\varphi}_{\gamma,\tilde{\mathcal{C}}}}\mathcal{T}_{\gamma,\tilde{\mathcal{C}}}(h))=\mathbb{P}_{\gamma,\tilde{\mathcal{C}}}(\mathcal{T}_{\gamma,\tilde{\mathcal{C}}}(\langle(\sigma_{10}\dots\sigma_{n0})\rangle))=1$.

$(\Leftarrow)$ Assume an orchestrator γ is such that $\mathcal{P}^{\tilde{\mathcal{C}}}_{\varphi}(\gamma)=1$. This implies that for all orchestrator infinite executions $t\in\mathsf{Supp}(\mathbb{P}_{\gamma,\tilde{\mathcal{C}}})\subseteq\mathcal{T}_{\gamma,\tilde{\mathcal{C}}}(\langle(\sigma_{10}\dots\sigma_{n0})\rangle)$, there is a prefix $h\in\mathsf{histories}(t)$ such that $h\in\mathcal{H}^{\varphi}_{\gamma,\tilde{\mathcal{C}}}$ and $t\in\mathcal{T}_{\gamma,\tilde{\mathcal{C}}}(h)$. This means t is successful, and therefore, all executions are successful, i.e. the definition of realizability.

Theorem 2. *Assume φ is realizable. If an orchestrator γ satisfies Eq. (4), then it realizes the specification φ.*

Proof. Since by assumption φ is realizable, then there exists an orchestrator γ' that realizes it. By Theorem 1, we can deduce that the optimal value of $\mathcal{P}^{\tilde{\mathcal{C}}}_{\varphi}(\gamma')$ is 1. Moreover, by assumption and by Eq. (4)), it follows that $\gamma\in\Gamma_{\mathcal{P}^{\tilde{\mathcal{C}}}_{\varphi}}$, i.e. $\mathcal{P}^{\tilde{\mathcal{C}}}_{\varphi}(\gamma)=1$, by the arguments above. Finally, again by Theorem 1, we get that γ realizes φ.

Finally, we formally state the stochastic version of our problem:

Problem 1 (Stochastic Composition for LTL$_f$ *Specifications).* Given the pair $(\tilde{\mathcal{C}},\varphi)$, where φ is an LTL$_f$ goal specification over the set of actions A, and $\tilde{\mathcal{C}}$ is a community of n stochastic services $\tilde{\mathcal{C}}=\{\tilde{\mathcal{S}}_1,\dots,\tilde{\mathcal{S}}_n\}$, compute, if it exists, an orchestrator that is optimal according to Eq. (4).

Interestingly, Theorem 1 and Theorem 2 show that one can find an orchestrator even in a non-stochastic setting by considering arbitrary services' probability distributions for $P_i(\sigma_i,a)$, for any pair σ_i and a, whose support is compatible with δ_i, and then check whether $\max_{\gamma}\mathcal{P}^{\tilde{\mathcal{C}}}_{\varphi}(\gamma)=1$.

3.2 Stochastic Services Solution Technique

Our solution technique is based on finding an optimal policy for a bi-objective lexicographic optimization on a specifically built MDP. In particular, we consider a variant of the framework introduced in [6]: while as the second objective, they considered the expected number of steps to a target, here we consider the expected cost. Our technique breaks down into the following steps: (1) first, we compute the equivalent NFA of an LTL$_f$ formula, $\mathcal{A}_\varphi$, and (2) we consider the DFA $\mathcal{A}_{\mathsf{act}}$, as in the non-stochastic setting; then (3) we compute a *product of* $\mathcal{A}_{\mathsf{act}}$ *with the stochastic services in* $\tilde{\mathcal{C}}$*, obtaining a new MDP*, $\mathcal{M}'$, that we call the "composition MDP"; (4) we find a policy π for $\mathcal{M}'$ that is optimal w.r.t. the bi-objective lexicographic function, as in [6], and then (5) we derive an orchestrator γ from π that is optimal w.r.t. Eq. 4. We now detail each step.

Step 1. The NFA of an LTL$_f$ formula can be computed by exploiting a well-known correspondence between LTL$_f$ formulas and automata on finite words [17]. In particular, using the LTL$_f$2NFA algorithm [5], we can compute an NFA $\mathcal{A}_\varphi = (A, Q, q_0, F, \delta)$ that is equivalent to the specification φ which can be exponentially larger than the size of the formula. Note that the alphabet of the NFA is A since we assume the specification satisfies the DECLARE assumption: only one action is executed at each time instant.

Step 2. From the NFA of the formula φ, $\mathcal{A}_\varphi$, which is on the alphabet A, we define a *controllable* DFA on the alphabet $A \times Q$, $\mathcal{A}_{\mathsf{act}} = (A \times Q, Q, a_0, F, \delta_{\mathsf{act}})$, where everything is as in $\mathcal{A}_\varphi$ except δ_{act} that is defined as follows: $\delta_{\mathsf{act}}(q, (a, q')) = q'$ iff $(q, a, q') \in \delta$. Notice that if a sequence of actions is accepted by the NFA $\mathcal{A}_\varphi$ as witnessed by the run $r = q_0, a_1, \ldots, q_n$, then the run itself is accepted by $\mathcal{A}_{\mathsf{act}}$. Intuitively, with the DFA $\mathcal{A}_{\mathsf{act}}$, we are giving to the controller not only the choice of actions but also the choice of transitions of the original NFA $\mathcal{A}_\varphi$, so that those chosen transitions lead to the satisfaction of the formula. In other words, for every sequence of actions $a_1, \ldots, a_n$ accepted by the NFA $\mathcal{A}_\varphi$, i.e. satisfying the formula φ, there exists a corresponding alternating sequence $q_0, a_1, \ldots, q_n$ accepted by the DFA $\mathcal{A}_{\mathsf{act}}$, and viceversa. This means that when we project out the Q-component from the accepted sequences of $\mathcal{A}_{\mathsf{act}}$, we get a sequence of actions satisfying φ. It can be shown that:

Proposition 1. *$a_1 \ldots a_m \in \mathcal{L}(\mathcal{A}_\varphi)$ iff $(a_1, q_1) \ldots (a_m, q_m) \in \mathcal{L}(\mathcal{A}_{\mathsf{act}})$, for some $q_1 \cdots q_m$.*

Proof. By definition, $a_1 \ldots a_m \in \mathcal{L}(\mathcal{A}_\varphi)$ iff there exist a run $r = q_1 \ldots q_m$ s.t. for $1 \leq k \leq m$, $\delta(q_{k-1}, a_k) = q_k$ and $q_m \in F$. Consider the word $w' = (a_1, q_1) \ldots (a_m, q_m)$. By construction of $\mathcal{A}_{\mathsf{act}}$, w' induces a run $r^d = r$. Since $q_m \in F$ by assumption, r^d is an accepting run for $\mathcal{A}_{\mathsf{act}}$, and therefore $w' \in \mathcal{L}(\mathcal{A}_{\mathsf{act}})$ is accepted. The other direction follows by construction because, if $(a_1, q_1) \ldots (a_m, q_m) \in \mathcal{L}(\mathcal{A}_{\mathsf{act}})$, then by construction $q_1 \ldots q_m$ is a run of $\mathcal{A}_\varphi$ over word $a_1 \ldots a_m$, and since $q_m \in F$ by assumption $a_1 \ldots a_m \in \mathcal{L}(\mathcal{A}_\varphi)$.

Step 3. Consider a goal specification φ and a community of stochastic services $\tilde{\mathcal{C}}$. Let $\mathcal{A}_{\mathsf{act}} = (A \times Q, Q, q_0, F, \delta_{\mathsf{act}})$ be the controllable DFA associated to the NFA $\mathcal{A}_\varphi$. We define the *Composition MDP* $\mathcal{M} = (S', A', P', s'_0)$ as follows: $S' = Q \times \Sigma_1 \times \cdots \times \Sigma_n$; $A' = A \times Q \times \{1 \ldots n\}$; $s'_0 = (q_0, \sigma_{10} \ldots \sigma_{n0})$; $P'(q', \sigma'_1 \ldots \sigma'_i \ldots \sigma'_n | q, \sigma_1 \ldots \sigma'_i \ldots \sigma_n, (a, q, i)) = P_i(\sigma'_i | \sigma_i, a)$ if $\delta_{\mathsf{act}}(q, (a, q')) = q'$. Moreover, let the composition cost function $C' : S' \times A' \to \mathbb{R}^+$ be defined as $C'((q, \sigma_1 \ldots \sigma_i \ldots \sigma_n), (a, q, i)) = C_i(\sigma_i, a)$.

We are interested in computing optimal policies for $\mathcal{M}$, where the optimality is defined as follows. Consider the target states $T = F \times F_1 \times \cdots \times F_n$. We consider the bi-objective lexicographic optimization over $\mathcal{M}'$, similarly to what has been done in [6]. In particular, we first consider the probability of reaching a set of target states T from $s \in S'$, following a policy π over the MDP $\mathcal{M}'$, denoted with $\mathbb{P}_{\mathcal{M}'_\pi, s}(\Diamond T)$; with $\Pi_{\mathcal{M}', s}(\Diamond T)$, we denote the set of policies with the maximum probability of reaching T, i.e. $\arg\max_\pi \mathbb{P}_{\mathcal{M}'_\pi, s}(\Diamond T)$. Then, we consider the cost of the shortest prefix of ρ that reaches one of the target states in T, i.e. $\mathsf{cost}_T(\rho) = \sum_{k=0}^{i} C'(s'_k, a'_k)$ if $\rho[i] \in T$ and for all $j < i$, $\rho[j] \notin T$. An optimal solution for $\mathcal{M}'$ is a policy π that minimizes the conditional expected cost of reaching a target state $\mathbb{E}_{\mathcal{M}'_\pi, s'_0}[\mathsf{cost}_T(\rho) | \Diamond T]$ among the policies in $\Pi_{\mathcal{M}, s'_0}(\Diamond T)$, that is, the policies which maximize $\mathbb{P}_{\mathcal{M}_\pi, s'_0}(\Diamond T)$, i.e.:

$$\pi \in \Pi_{\mathcal{M}', s_0}(\Diamond T) \text{ and } \pi \in \arg\min_{\pi'} \mathbb{E}_{\rho \sim \mathcal{M}'_\pi, s'_0}\left[\mathsf{cost}_T(\rho) | \Diamond T\right] \tag{5}$$

Step 4. The solution technique we will use is based on the work [6], where the authors propose a two-stage technique to find an optimal policy for a bi-objective lexicographic function in the form of Eq. (5). First, we compute the set of policies (in the form of a set of optimal actions for each state) that maximize the probability of reaching the target states; however, this set of policies also contains the "deferral" policies, i.e. policies that defer the actual reaching of the target states indefinitely, but in such a way that the target can still be reached with maximum probability at any moment. Then, we consider a "pruned MDP" in which (*i*) only optimal action can be taken, and (*ii*) only states from which the target can be reached are kept. The new MDP is used to find policies that minimize the expected cost of reaching the target. By construction, the optimal policy of the pruned MDP guarantees the target is always reached since any deferral policy will incur an infinite cost. The difference between our scenario and [6] is that they consider the length of the path, rather than its cost, as the second objective function. Nevertheless, it is easy to see that their approach works if, instead of considering the expected length of successful paths, we consider their expected total costs (i.e. minimizing path length can be seen as minimizing costs with each transition having unitary cost). Note that the techniques used to find the solutions are standard: the first stage requires solving the maximal reachability probability problem [1] on the composition MDP with the accepting end components as the set of states T. The second stage requires solving a stochastic shortest path problem [31] on the pruned MDP. The two subproblems can be solved efficiently using standard planning algorithms, e.g., Value Iteration or Linear Programming.

Step 5. Once an optimal policy is found, we can obtain its equivalent γ as follows: for any finite prefix of a run $\rho = (q_0, \sigma_{10} \dots \sigma_{n0}), (a_1, q_1, o_1), \dots (a_m, q_m, o_m), (q_m, \sigma_{1m} \dots \sigma_{nm}), \dots$, we set $\gamma((\sigma_{10} \dots \sigma_{n0}) \dots (\sigma_{1m} \dots \sigma_{nm})) = (a_{m+1}, o_{m+1})$, where $\pi(\rho) = (a_{m+1}, q_{m+1}, o_{m+1})$.

Now we aim to establish a relationship between optimal orchestrators according to Eq. (4), and optimal policies for $\mathcal{M}'$ according to Eq. (5). Given an infinite run $\rho = (q_0, \sigma_{10} \dots \sigma_{n0}), (a_1, q_1, o_1) \dots$, we define the trace $t = \tilde{\tau}_{\varphi,\tilde{\mathcal{C}}}(\rho) = (\sigma_{10} \dots \sigma_{n0}), (a_1, o_1), \dots$. The definition easily applies to finite prefixes of ρ but this time mapping into histories of t.

Now, we are going to prove a sequence of lemmata.

Lemma 2 shows that, once fixed a policy π over $\mathcal{M}'$, there is a one-to-one correspondence (modulo choices of $q_0 \dots q_m$ in ρ) between paths ρ in $\mathcal{M}'$ following π and the executions $t \in \mathcal{T}_{\gamma,\mathcal{C}}$ of the equivalent orchestrator of π, γ; Lemma 3 shows that the probabilities of finite paths and histories are the same; Lemma 4 shows that paths that end with states in T correspond to successful histories; and Lemma 5 shows a correspondence between paths in $\mathsf{Paths}_{T,\mathcal{M}'_\pi}$ and $\mathcal{H}^{\varphi}_{\gamma,\tilde{\mathcal{C}}}$.

Lemma 2. *Let π be a policy for $\mathcal{M}'$ and let γ be its equivalent orchestrator. Moreover, let $\rho \in \mathsf{Paths}^{\omega}_{\mathcal{M}'_\pi}$ and t be a trace such that $t = \tilde{\tau}_{\varphi,\tilde{\mathcal{C}}}(\rho)$. Then, $\rho \in \mathsf{Paths}^{\omega}_{\mathcal{M}'_\pi}(\langle s'_0 \rangle)$ iff $t \in \mathcal{T}_{\gamma,\tilde{\mathcal{C}}}(\langle(\sigma_{10} \dots \sigma_{n0})\rangle)$.*

Proof. Let the infinite path $\rho \in \mathsf{Paths}^{\omega}_{\mathcal{M}'_\pi}(\langle s'_0 \rangle)$, and the infinite trace $t = \tilde{\tau}_{\varphi,\tilde{\mathcal{C}}}(\rho))$. We prove the claim by induction on the position of the run/trace.

Base case: we have the claim holds for position 0 because $\rho[0] = (q_0, \sigma_{10} \dots \sigma_{n0})$, and $h[0] = \sigma_{10} \dots \sigma_{n0}$. Therefore, $\langle h[0] \rangle$ satisfies the conditions of the definitions of history and execution of γ iff $s'_0 \in S'$.

Inductive case: assume the claim holds up to position $k \geq 0$. Consider the $(k+1)$-th action according to π, i.e. $\pi(\rho[0{:}k]) = (a_{k+1}, q_{k+1}, o_{k+1})$, and its successor state $\rho[k+1] = (q_{k+1}, \sigma_{1,k+1}, \dots, \sigma_{n,k+1})$. By construction of $\mathcal{M}'$, we have that (*i*) $\sigma_{i,k+1} \in \Sigma_i$ for all services, (*ii*) $o_{k+1} \in \{1 \dots n\}$, (*iii*) $a_{k+1} \in A$, and (*iv*) $\sigma_{k+1} \in \mathsf{Supp}(P_{o_{k+1}}(\sigma_{o_{k+1},k}, a_{k+1}))$. Moreover, by construction of γ, $(a_{k+1}, o_{k+1}) = \gamma(\mathsf{states}(t[0:k]))$, hence $h' = t[0:k], (a_{k+1}, o_{k+1}), (\sigma_{1,k+1} \dots \sigma_{n,k+1})$ is a proper (finite) execution. The same arguments can be applied in the other direction by extending the actions of the trace with the next state $q_{k+1} \in Q$, where q_{k+1} is determined by the policy π (see above). By induction the claim also holds for any arbitrary position, and therefore $\rho \in \mathsf{Paths}^{\omega}_{\mathcal{M}'_\pi}(\langle s'_0 \rangle)$ iff $t \in \mathcal{T}_{\gamma,\tilde{\mathcal{C}}}(\langle(\sigma_{10} \dots \sigma_{n0})\rangle)$.

Lemma 3. *Let π a policy on $\mathcal{M}'$, γ be its equivalent orchestrator, $\rho = s'_0 a_1 \dots s'_m \in \mathsf{Paths}_{\mathcal{M}'_\pi}(s'_0)$ be a finite path on $\mathcal{M}'$, and $\tilde{h} = \tilde{\tau}_{\varphi,\tilde{\mathcal{C}}}(\rho)$ be its associated history. Then, $\mathbb{P}_{\mathcal{M}_\pi, s'_0}(\mathsf{Paths}^{\omega}_{\mathcal{M}_\pi}(\rho)) = \mathbb{P}_{\gamma,\tilde{\mathcal{C}}}(\mathcal{T}_{\gamma,\tilde{\mathcal{C}}}(h))$.*

Proof.

$$\mathbb{P}_{\mathcal{M}'_\pi, s'_0}(\mathsf{Paths}^{\omega}_{\varphi,\tilde{\mathcal{C}}}(\rho)) = \prod_{k=1}^{m} P'(s'_k \mid s'_{k-1}, (a_k, q_k, o_k)) \tag{6}$$

$$= \prod_{k=1}^{m} P_{o_k}(\sigma_{o_k,k} \mid \sigma_{o_k,k-1}, (a_k, o_k)) \tag{7}$$

$$= \mathbb{P}_{\gamma,\tilde{\mathcal{C}}}(\mathcal{T}_{\gamma,\tilde{\mathcal{C}}}(h)) \tag{8}$$

where step 6 is by definition of the probability of a cylinder set, step 7 by definition of P' in $\mathcal{M}'$, and step 8 by Eq. (1).

Lemma 4. *Let $\rho = s_0 a_1 \ldots s_m \in \mathsf{Paths}_{\mathcal{M}'_\pi}$ be a finite path on $\mathcal{M}'$, and let $h = \tilde{\tau}_{\varphi,\tilde{\mathcal{C}}}(\rho)$ be its associated history. Then, $s_m \in T$ iff* $\mathsf{successful}(h)$.

Proof. By induction on the length of the run/history.
Base case: $\rho_0 = \langle s'_0 \rangle = \langle (q_0, \sigma_{10} \ldots \sigma_{n0}) \rangle$. Let $h_0 = \tilde{\tau}_{\varphi,\tilde{\mathcal{C}}}(\rho_0) = \langle (\sigma_{10} \ldots \sigma_{n0}) \rangle$. We have that $\rho_0[0] \in T$ iff $(i.a)$ $q_0 \in F$ and $(i.b)$ $\sigma_{i0} \in F_i$ for all $1 \leq i \leq n$. On the other hand, h is successful iff $(ii.a)$ $\mathsf{actions}(h_0) = \epsilon \models \varphi$ and $(ii.b)$ $\sigma_{i0} \in F_i$. The claim holds because $(i.b)$ is precisely $(ii.b)$, and $(i.a)$ holds iff $(ii.a)$ holds by the correctness of the construction of $\mathcal{A}_{\mathsf{act}}$.
Inductive case: assume the claim holds for $\rho_{k-1} = (q_0, \sigma_{10} \ldots \sigma_{n0}), (a_1, q_1, o_1), \ldots, (a_{k-1}, q_{k-1}, o_{k-1}), (q_{k-1}, \sigma_{1,k-1} \ldots \sigma_{n,k-1})$ and $h_{k-1} = \tilde{\tau}_{\varphi,\tilde{\mathcal{C}}}(\rho)$. Let $a'_k = (a_k, q_k, o_k)$ be any valid next action taken from s_{k-1}, and let $s_k = (q_k, \sigma_{1k} \ldots \sigma_{nk}) \in \mathsf{Supp}(P_{o_k}(s_{k-1}, a_k))$ the next possible state. Consider the sequence $r = q_0 \ldots q_k$. By construction of $\mathcal{M}'$, and correctness of $\mathcal{A}^d$, we have that r is a run over $\mathcal{A}^d$, and that $q_k \in F$ iff $a_1 \ldots a_k \models \varphi$. By definition of $h_k = \tilde{\tau}_{\varphi,\tilde{\mathcal{C}}}(\rho_k)$, we also have that $\mathsf{actions}(h_k) = a_1 \ldots a_k$. Finally, we have that $s_k \in F$ iff (i) $q_k \in F$ and (ii) for all i $\sigma_{ik} \in F_i$ by construction of $\mathcal{M}'$; (i) holds iff (iii) $\mathsf{actions}(h_k) \models \varphi$ by the arguments above; finally, (ii) and (iii) hold iff $\mathsf{successful}(h)$.

Let $\mathsf{Paths}_{T,\mathcal{M}'_\pi}(s'_0)$ be the set of finite paths following π on $\mathcal{M}'$ such that they start with s'_0 and enter in a state in T only at the end of the path and for the first time, i.e. $\mathsf{Paths}_{T,\mathcal{M}'_\pi}(s'_0) = ((S' \setminus T) \times A)^* T \cap \mathsf{Paths}_{\mathcal{M}'_\pi}(s'_0)$.

Lemma 5. $\rho \in \mathsf{Paths}_{T,\mathcal{M}'_\pi}(s'_0)$ *iff* $\tilde{\tau}_{\gamma,\tilde{\mathcal{C}}}(\rho) \in \mathcal{H}^{\varphi}_{\gamma,\tilde{\mathcal{C}}}$

Proof. By Lemma 4, $\rho \in \mathsf{Paths}_{T,\mathcal{M}'_\pi}$ iff $h = \tilde{\tau}_{\gamma,\tilde{\mathcal{C}}}(\rho)$ is successful. Moreover, by Lemma 2, $\rho \in \mathsf{Paths}_{T,\mathcal{M}'_\pi} \subseteq \mathsf{Paths}_{\mathcal{M}'_\pi}$ iff h is an execution of γ. Furthermore, by assumption, any finite prefix ρ', say of length m, of ρ, is such that $\rho'[m] \notin T$. Then, again by Lemma 4, this holds iff $h' = \tilde{\tau}_{\varphi,\tilde{\mathcal{C}}}(\rho')$ is not successful, meaning that does not exist a prefix $h' \in \mathsf{prefixes}(h)$ with $h' \neq h$ such that h' is successful. But this is precisely the membership condition for $\mathcal{H}^{\varphi}_{\gamma,\tilde{\mathcal{C}}}$

This result shows the correctness of our technique:

Theorem 3. *Let $(\tilde{\mathcal{C}}, \varphi)$ be an instance of Problem 2, and let $\mathcal{M}'$ be the composition MDP for $\tilde{\mathcal{C}}$ and φ. We have that π is optimal (w.r.t. Eq. (5)) iff its equivalent orchestrator γ is optimal (w.r.t. Eq. (4)).*

Proof. First, we show that $\pi = \arg\max_{\pi'} \mathbb{P}_{\mathcal{M}'_\pi, s'_0}(\Diamond T)$ iff $\gamma = \arg\max_{\gamma'} \mathcal{P}^{\tilde{\mathcal{C}}}_{\varphi}(\gamma')$. For any pair π and its equivalent γ, we have:

$$\begin{aligned}
\mathbb{P}_{\pi, s'_0}(\Diamond T) &= \sum_{\rho_T \in \mathsf{Paths}_{T,\pi}(s'_0)} \mathbb{P}_{\mathcal{M}'_{\pi'}, s'_0}(\mathsf{Paths}^{\omega}_{\mathcal{M}'_\pi, s'_0}(\rho_T)) &(9)\\
&= \sum_{\tilde{h} \in \mathcal{H}^{\varphi}_{\gamma,\tilde{\mathcal{C}}}} \mathbb{P}_{\gamma,\tilde{\mathcal{C}}}(\mathcal{T}_{\gamma,\tilde{\mathcal{C}}}(\tilde{h})) &(10)\\
&= \mathbb{P}_{\gamma,\tilde{\mathcal{C}}}\Big(\bigcup_{h \in \mathcal{H}^{\varphi}_{\gamma,\tilde{\mathcal{C}}}} \mathcal{T}_{\gamma,\tilde{\mathcal{C}}}(h)\Big) &(11)\\
&= \mathcal{P}^{\tilde{\mathcal{C}}}_{\varphi}(\gamma) &(12)
\end{aligned}$$

where step 9 is by definition of probabilistic reachability, step 10 is by Lemma 3 and Lemma 5, step 11 is by disjointness of all $\mathcal{T}_{\gamma,\tilde{\mathcal{C}}}(h)$ for $h \in \mathcal{H}^{\varphi}_{\gamma,\tilde{\mathcal{C}}}$, and step 12 is by Eq. (2). From this, we obtain that $\pi^* = \arg\max_{\pi'} \mathbb{P}_{\mathcal{M}_{\pi'}, s'_0}(\Diamond T)$ iff $\gamma^* = \arg\max_{\gamma'} \mathcal{P}^{\tilde{\mathcal{C}}}_{\varphi}(\gamma')$.

It remains to prove that π is cost-optimal iff γ is cost-optimal. We have:

$$\begin{aligned}
&\mathbb{E}_{\rho \sim \mathcal{M}'_\pi, s'_0}[\mathsf{cost}_T(\rho) \mid \Diamond T]\\
&\quad= \sum_{\rho_T \in \mathsf{Paths}_{T,\pi}(s'_0)} \mathbb{P}_{\mathcal{M}'_{\pi'}, s'_0}(\mathsf{Paths}^{\omega}_{\mathcal{M}'_\pi, s'_0}(\rho_T)) \cdot \sum_{k=0}^{|\rho_T|} C'(s'_k, a'_{k+1}) &(13)\\
&\quad= \sum_{\tilde{h} \in \mathcal{H}^{\varphi}_{\gamma,\tilde{\mathcal{C}}}} \mathbb{P}_{\gamma,\tilde{\mathcal{C}}}(\mathcal{T}_{\gamma,\tilde{\mathcal{C}}}(\tilde{h})) \cdot \sum_{k=0}^{|h|} C_{o_{k+1}}(\sigma_{o_k,k}, a_{k+1}) &(14)\\
&\quad= \mathbb{E}_{h \sim \mathbb{P}_{\gamma,\tilde{\mathcal{C}}}}\left[\sum_{k=1}^{|h|} C_{o_k}(\sigma_{o_k,k-1}, a_k) \,\middle|\, \mathsf{successful}(h)\right] &(15)\\
&\quad= \mathcal{J}^{\tilde{\mathcal{C}}}_{\varphi}(\gamma) &(16)
\end{aligned}$$

where step 13 by definition of total expected cost conditioned on reaching of target states T, step 14 by construction of $\mathcal{M}'$ and by Lemma 3 and Lemma 5, step 15 by definition of total expected cost on successful executions of γ, and step 16 by Eq. (3). Therefore, if $\pi \in \arg\min_{\pi'} \mathbb{E}_{\rho \sim \mathcal{M}'_\pi, s'_0}[\mathsf{cost}_T(\rho) \mid \Diamond T]$ then $\gamma \in \arg\min_{\gamma'} \mathcal{J}^{\tilde{\mathcal{C}}}_{\varphi}(\gamma')$. Combining both results, we get the thesis.

Computational Cost. Theorem 3 guarantees that we can reduce Problem 1 to the problem of finding an optimal policy for the lexicographic bi-objective optimization problem (Eq. (5)) over a composition MDP $\mathcal{M}'$. As explained above,

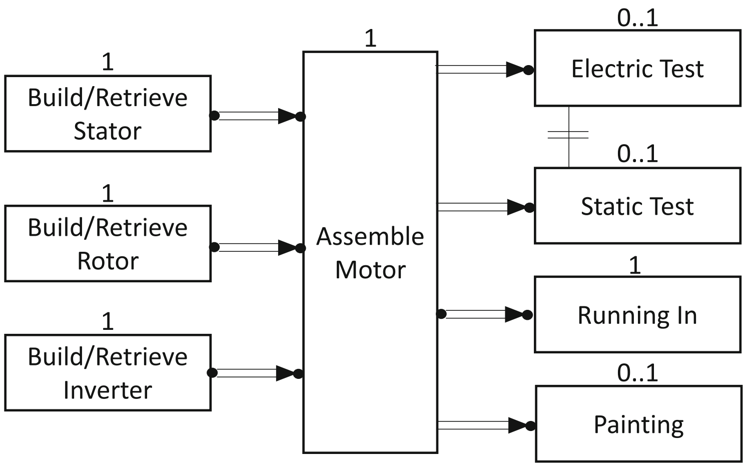


Fig. 2. The electric motor manufacturing process represented using DECLARE.

the two-stage technique requires solving a planning problem over MDPs. Since it is known that both steps require polynomial time complexity in the number of states and actions of the MDP [31] and that our Composition MDP has a state space that is a single-exponential in the size of the goal specification, we get this result:

Theorem 4. *Problem 1 can be solved in at most exponential time in the size of the formula, in at most exponential time in the number of services, and in polynomial time in the size of the services.*

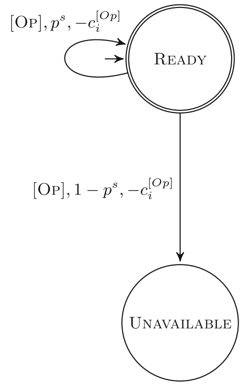
Observe that, differently from the classical setting of LTL/LTL$_f$ synthesis on probabilistic systems [11,35], and analogously to our solution method for the non-stochastic case, we do not have unobservable "adversarial" nondeterminism in the composition MDP; hence, we do not need to determinize the NFA of the goal specification, thereby saving an exponential blow-up in time complexity.

4 Case Study

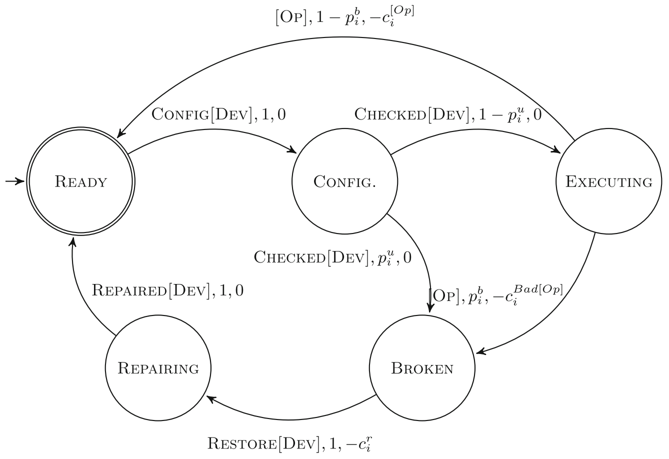
To introduce the proposed approach, its functionalities, and its capacity to address smart manufacturing, we consider the production process of an electric motor widely used in various applications such as industrial machinery, electric vehicles, household appliances, and many others [12]. To function properly, electric motors require certain materials that possess specific electrical and magnetic properties. Therefore, before the manufacturing processes start, the raw materials (i.e., copper, steel, aluminium, magnets, insulation materials, bearings) must be extracted and refined to obtain essential metals and polymers for electric motor parts manufacturing. When the materials are in the manufacturing facility, the effective manufacturing process can start. For the sake of brevity, in the following, we focus on the main aspects of the manufacturing process, skipping the provisioning, but the formalization can be easily extended to cover more details.

Goal. Figure 2 depicts the DECLARE formalization [30] of the electric motor manufacturing process. The main components of an electric motor are the stator, the rotor, and, in the case of alternate current motors with direct current power (e.g., in the case of electric cars). These three components are built or retrieved in any order (*no precedence* DECLARE constraints between these tasks) and then eventually assembled to build a motor (*alternate succession* constraint between Build/Retrieve tasks and the Assemble Motor task). After the motor is assembled, a running-in test must be performed (*alternate succession* constraint between the Assemble Motor task and the Running In task), and at most one (*not coexistence* constraint) between an electric test and a full static test (the latter comprises the former). In addition, the motor can be painted optionally. The Painting, Electric Test and Static Test tasks optionally follow the Assemble Motor task (*alternate precedence* constraints). The process depicts the manufacturing tasks involved in producing a *single motor* as indicated by the *existence constraints*. Machines and/or human operators can perform all these operations.

Services. The behaviour of each process actor can be described as a stochastic service, i.e., a state machine with a probabilistic behaviour used to model two types of actors involved in the manufacturing process, namely *machines* and *human operators*, shown in Fig. 3. Each transition edge has a label which indicates the operation, the probability of transition and the associated cost. Figure 3a depicts a simple stochastic service of human workers. Such services have an initial accepting state in which they are READY and accept operations, and a sink failure state from where no action can be taken. The transition triggered by the [OP] action has a probability of p^s of remaining in the same ready state (success), but a $1 - p^s$ probability of failing. The transition is associated with a certain cost $c_i^{[Op]}$ to perform the action (preceded by a $-$, i.e., the cost is thought of as a negative value, using the reward-based representation). Figure 3b depicts a generic stochastic service of machines. The machine is initially in the READY state, which is also the unique accepting state, where it can receive the CONFIG[DEV] command (the reader, in the following, can imagine the [DEV] trailer to be replaced with the name of the specific manufacturing actor). This action takes the service to the CONFIGURATION state where the actor is set up or warmed up. At the end of this phase, the CHECKED[DEV] action is performed. If the configuration is unsuccessful, with a probability p_i^u representing the possibility of finding the actor unemployable, the actor goes into the BROKEN state. If the configuration is successful, with a probability $1 - p_i^u$, the actor goes to the EXECUTING state, where a family of operations, denoted with [OP] in the picture, can be executed. For the sake of compactness, we only show a single operation, but the service can be easily generalized to the case where a single actor can perform multiple operations. The action [OP] represents one of those operations defined in Fig. 2. Executing an operation implies a certain cost



(a) "Human operator" service.


[Op], $1-p_i^b$, $-c_i^{[Op]}$
Config[Dev], 1, 0
Checked[Dev], $1-p_i^u$, 0
Ready
Config.
Executing
Checked[Dev], p_i^u, 0
[Op], p_i^b, $-c_i^{Bad[Op]}$
Repaired[Dev], 1, 0
Repairing
Broken
Restore[Dev], 1, $-c_i^r$

(b) "Machine" service.

Fig. 3. The two types of service we consider for the Electric Motor case study.

$c_i^{[Op]}$. In some cases, the execution of [Op] may take the actor i to the Broken state with probability p_i^b and also, in this case, the operation implies a high cost $c_i^{Bad[Op]}$. To take the actor back to the Ready state, a Restore[Dev] task must be executed on the actor, which has a repair cost c_i^r depending on the actual conditions of the actor, and that takes the actor to the Repairing state. When the actor is repaired, a Repaired[Dev] event is received, making the actor available again for manufacturing. Noteworthy, the Config[Dev], Checked[Dev], Restore[Dev], Repaired[Dev] operations do not leave any trace on the target process. Noticeably, we can imagine that some of these operations are triggered, in reality, as exogenous events, i.e., they should be reflected in the controller, but the actor will wait for these events instead of autonomously enacting them.

We are interested in the problem of maximising the probability that the smart factory succeeds in producing electric motors at a minimum utilization cost. A two-stage approach can achieve this: in the first stage, we aim to find the maximally permissive strategy that (*i*) determines the equally optimal sequences of actions to satisfy the goal specification and (*ii*) the equally optimal dispatching strategy that decides which services should perform the operation. The optimization must also consider configuration/checking/repairing action to bring the service back to a final configuration. This might require limiting the use of services with certain probability of leading to a failing configuration. In the second stage, we select, among the available strategies, those that also minimize the utilization cost. Crucially, the optimal solution might vary depending on the service available and their capabilities, as well as the probabilities p_i^s, p_i^u, p_i^b and costs $c_i^[$Op], $c_i^{Bad[Op]}$ and c_i^r. Given the high degrees of freedom, it is paramount to use a technique, such as the one proposed in this work, that can automatically handle such a complex scenario.

5 Conclusion

This paper proposes a novel stochastic composition framework in which we aim to maximize the satisfaction probability of a goal specification, expressed as a high-level logic formalism such as LTL_f, and conditioned on this, minimize the utilization costs of the available services. We formalized the problem and proposed a solution based on a reduction to a bi-objective optimization over MDPs, proving the correctness. Finally, we highlighted the relevance of our contribution by providing an industrial case study considered in the literature. In future work, we would like to study the process-oriented variant of our framework, namely, to maximize the probability of realizing *all* traces that are compatible with the specification, and conditioned on this, maximize as much as possible the average expected reward coming from the utilization of the services. This would allow us to consider a hierarchy of target specifications (either goal-oriented or process-oriented), hence delivering a rich framework suitable for several applications. The same kind of generalization can be considered for the reward function of the services, where we can have more than one reward to consider regarding the service utilization. Moreover, we would like to implement our approach using state-of-the-art probabilistic model checkers such as PRISM [24] and Storm [23].

Acknowledgements. This work has been partially supported by the EU H2020 project AIPlan4EU (No. 101016442), the ERC-ADG WhiteMech (No. 834228), the EU ICT-48 2020 project TAILOR (No. 952215), the PRIN project RIPER (No. 20203FFYLK), and the PNRR MUR project FAIR (No. PE0000013). This work has been carried out while Luciana Silo was enrolled in the Italian National Doctorate on Artificial Intelligence run by Sapienza University of Rome.

References

1. de Alfaro, L.: Formal verification of probabilistic systems. Ph.D. thesis, Stanford University, USA (1997). https://searchworks.stanford.edu/view/3910936
2. Berardi, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Mecella, M.: Automatic composition of *E*-services that export their behavior. In: Orlowska, M.E., Weerawarana, S., Papazoglou, M.P., Yang, J. (eds.) ICSOC 2003. LNCS, vol. 2910, pp. 43–58. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-24593-3_4
3. Berardi, D., Calvanese, D., De Giacomo, G., Mecella, M.: Composition of services with nondeterministic observable behavior. In: Benatallah, B., Casati, F., Traverso, P. (eds.) ICSOC 2005. LNCS, vol. 3826, pp. 520–526. Springer, Heidelberg (2005). https://doi.org/10.1007/11596141_43
4. Brafman, R.I., De Giacomo, G., Mecella, M., Sardiña, S.: Service composition in stochastic settings. In: Esposito, F., Basili, R., Ferilli, S., Lisi, F. (eds.) AI*IA 2017. LNCS, vol. 10640, pp. 159–171. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70169-1_12
5. Brafman, R.I., De Giacomo, G., Patrizi, F.: LTLf/LDLf non-Markovian rewards. In: AAAI, pp. 1771–1778. AAAI Press (2018)
6. Busatto-Gaston, D., Chakraborty, D., Majumdar, A., Mukherjee, S., Pérez, G.A., Raskin, J.: Bi-objective lexicographic optimization in Markov decision processes

with related objectives. In: André, É., Sun, J. (eds.) ATVA 2023. LNCS, vol. 14215, pp. 203–223. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-45329-8_10
7. Chatterjee, K., Katoen, J.-P., Weininger, M., Winkler, T.: Stochastic games with lexicographic reachability-safety objectives. In: Lahiri, S.K., Wang, C. (eds.) CAV 2020. LNCS, vol. 12225, pp. 398–420. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-53291-8_21
8. Chatterjee, K., Majumdar, R., Henzinger, T.A.: Markov decision processes with multiple objectives. In: Durand, B., Thomas, W. (eds.) STACS 2006. LNCS, vol. 3884, pp. 325–336. Springer, Heidelberg (2006). https://doi.org/10.1007/11672142_26
9. Chen, K., Xu, J., Reiff-Marganiec, S.: Markov-HTN planning approach to enhance flexibility of automatic web service composition. In: ICWS, pp. 9–16. IEEE Computer Society (2009)
10. Chen, Y., Huang, J., Lin, C., Shen, X.: Multi-objective service composition with QoS dependencies. IEEE Trans. Cloud Comput. (2019)
11. Courcoubetis, C., Yannakakis, M.: The complexity of probabilistic verification. J. ACM **42**(4), 857–907 (1995)
12. De Giacomo, G., Favorito, M., Leotta, F., Mecella, M., Monti, F., Silo, L.: AIDA: a tool for resiliency in smart manufacturing. In: Cabanillas, C., Pérez, F. (eds.) CAiSE 2023. LNBIP, vol. 477, pp. 112–120. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-34674-3_14
13. De Giacomo, G., Favorito, M., Leotta, F., Mecella, M., Silo, L.: Digital twins composition via Markov decision processes. In: ITBPM@BPM. CEUR Workshop Proceedings, vol. 2952, pp. 44–49. CEUR-WS.org (2021)
14. De Giacomo, G., Favorito, M., Leotta, F., Mecella, M., Silo, L.: Modeling resilient cyber-physical processes and their composition from digital twins via Markov decision processes. In: PMAI@IJCAI. CEUR Workshop Proceedings, vol. 3310, pp. 101–104. CEUR-WS.org (2022)
15. De Giacomo, G., Favorito, M., Leotta, F., Mecella, M., Silo, L.: Digital twin composition in smart manufacturing via Markov decision processes. Comput. Ind. **149**, 103916 (2023)
16. De Giacomo, G., Mecella, M., Patrizi, F.: Automated service composition based on behaviors: the roman model. In: Bouguettaya, A., Sheng, Q., Daniel, F. (eds.) Web Services Foundations, pp. 189–214. Springer, New York (2014). https://doi.org/10.1007/978-1-4614-7518-7_8
17. De Giacomo, G., Patrizi, F., Sardina, S.: Automatic behavior composition synthesis. Artif. Intell. **196**, 106–142 (2013)
18. De Giacomo, G., Vardi, M.Y.: Linear temporal logic and linear dynamic logic on finite traces. In: IJCAI, pp. 854–860. IJCAI/AAAI (2013)
19. Di Ciccio, C., Montali, M.: Declarative process specifications: reasoning, discovery, monitoring. In: van der Aalst, W.M.P., Carmona, J. (eds.) Process Mining Handbook. Lecture Notes in Business Information Processing, vol. 448, pp. 108–152. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-08848-3_4
20. Dumas, M., et al.: AI-augmented business process management systems: a research manifesto. ACM Trans. Manag. Inf. Syst. **14**(1), 11:1–11:19 (2023)
21. Gao, A., Yang, D., Tang, S., Zhang, M.: Web service composition using Markov decision processes. In: Fan, W., Wu, Z., Yang, J. (eds.) WAIM 2005. LNCS, vol. 3739, pp. 308–319. Springer, Heidelberg (2005). https://doi.org/10.1007/11563952_28

22. Hahn, E.M., Perez, M., Schewe, S., Somenzi, F., Trivedi, A., Wojtczak, D.: Model-free reinforcement learning for lexicographic omega-regular objectives. In: Huisman, M., Păsăreanu, C., Zhan, N. (eds.) FM 2021. LNCS, vol. 13047, pp. 142–159. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-90870-6_8
23. Hensel, C., Junges, S., Katoen, J., Quatmann, T., Volk, M.: The probabilistic model checker storm. Int. J. Softw. Tools Technol. Transf. **24**(4), 589–610 (2022)
24. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: verification of probabilistic real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 585–591. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22110-1_47
25. Marrella, A., Mecella, M., Sardiña, S.: Supporting adaptiveness of cyber-physical processes through action-based formalisms. AI Commun. **31**(1), 47–74 (2018)
26. Monti, F., Silo, L., Leotta, F., Mecella, M.: On the suitability of AI for service-based adaptive supply chains in smart manufacturing. In: ICWS, pp. 704–706. IEEE (2023)
27. Monti, F., Silo, L., Leotta, F., Mecella, M.: Services in smart manufacturing: comparing automated reasoning techniques for composition and orchestration. In: Aiello, M., Barzen, J., Dustdar, S., Leymann, F. (eds.) SummerSOC 2023. CCIS, vol. 1847, pp. 69–83. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-45728-9_5
28. Moustafa, Ahmed, Zhang, Minjie: Multi-objective service composition using reinforcement learning. In: Basu, Samik, Pautasso, Cesare, Zhang, Liang, Fu, Xiang (eds.) ICSOC 2013. LNCS, vol. 8274, pp. 298–312. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-45005-1_21
29. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F.: Service-oriented computing: state of the art and research challenges. Computer **40**(11), 38–45 (2007)
30. Pesic, M., Schonenberg, H., Van der Aalst, W.M.: Declare: full support for loosely-structured processes. In: EDOC (2007)
31. Puterman, M.L.: Markov decision processes (1994)
32. Sadeghiram, S., Ma, H., Chen, G.: A user-preference driven lexicographic approach for multi-objective distributed web service composition. In: 2020 IEEE Symposium Series on Computational Intelligence (SSCI) (2020)
33. Skalse, J., Abate, A.: On the limitations of Markovian rewards to express multi-objective, risk-sensitive, and modal tasks. In: UAI. Proceedings of Machine Learning Research, vol. 216, pp. 1974–1984. PMLR (2023)
34. Skalse, J., Hammond, L., Griffin, C., Abate, A.: Lexicographic multi-objective reinforcement learning. In: IJCAI, pp. 3430–3436. ijcai.org (2022)
35. Wells, A.M., Lahijanian, M., Kavraki, L.E., Vardi, M.Y.: LTLf synthesis on probabilistic systems. In: GandALF. EPTCS, vol. 326 (2020)
36. Wray, K., Zilberstein, S., Mouaddib, A.I.: Multi-objective MDPs with conditional lexicographic reward preferences. In: AAAI (2015)
37. Yadav, N., Sardiña, S.: Decision theoretic behavior composition. In: AAMAS, pp. 575–582. IFAAMAS (2011)