The Knowledge Engineering Review

RESEARCH ARTICLE

Open Access

https://doi.org/10.48130/ker-0026-0003

The Knowledge Engineering Review **2026**, 41: e003

Stochastic service composition for LTL$_f$ reachability and safety tasks

Giuseppe De Giacomo[1,2], Marco Favorito[3] and Luciana Silo[2*]

[1] *Department of Computer Science, University of Oxford, Oxford OX1 3QD, United Kingdom*
[2] *Department of Computer, Control and Management Engineering (DIAG), Sapienza University of Rome, Via Ariosto 25, 00185 Rome, Italy*
[3] *Bank of Italy, Via Nazionale 91, 00184 Rome, Italy*
* Correspondence: silo@diag.uniroma1.it (Silo L)

Abstract

Service composition *à la* Roman model consists of realizing a virtual service by suitably orchestrating 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 satisfying the temporal logic objective (either reachability or safety) while minimizing the expected cost of using the available services. To do so, we combine techniques from LTL$_f$ synthesis, service composition *à la* Roman Model, and bi-objective lexicographic optimization on Markov Decision Processes (MDPs). This framework has several interesting applications, including Smart Manufacturing and Digital Twins.

Citation: De Giacomo G, Favorito M, Silo L. 2026. Stochastic service composition for LTL$_f$ reachability and safety tasks. *The Knowledge Engineering Review* 41: e003 https://doi.org/10.48130/ker-0026-0003

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[1]. 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[2–4] 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.

Recently, a renewed interest in service composition *à la* Roman Model has emerged from its potential applications in smart manufacturing. In particular, through digital twins technology, manufacturing devices can export their behaviour as transition systems, and hence be orchestrated in very much the same way as services did back in the early 2000s[5–7].

These technological trends align with the vision of smart manufacturing, which advocates the need for flexible, modular, and interoperable system architectures. Modern manufacturing systems often consist of heterogeneous components, ranging from robotic workstations to cyber-physical subsystems, that communicate through standardized APIs. In such settings, each component can be naturally abstracted as a service. Consequently, adopting the Service-Oriented Architectures (SOAs) approach enables the dynamic composition and coordination of such services, leveraging their exposed behaviors while maintaining loose coupling and scalability.

In this context, service composition *à la* Roman Model provides a useful abstraction for orchestrating device behavior toward achieving temporally extended goals. By treating each device or software module as a service with internal dynamics, and expressing high-level objectives via Linear Temporal Logic on finite traces (LTL$_f$), our framework bridges declarative specification with executable orchestration.

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 Yadav et al.[8] and Brafman et al.[9]. 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 [5,6,10]. 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 LTL$_f$ [11], which are at the base of declarative process specification in Business Process Management (BPM) through the so-called DECLARE paradigm[12–14]. Third, apart from satisfying the target, it is of interest to also minimize the cost coming from the service utilization[15–17]. This concern, together with the satisfaction of the target, calls for resorting to Multi-Objective Optimization for computing a solution.

A first attempt to do so may resort to Multi-Objective MDPs (MOMDPs)[18]. 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[19–22]. It is known that, in general, single Markovian rewards cannot capture certain multi-objective tasks, such as ones with lexicographic preferences[23]; hence, such problems cannot be easily reduced to standard techniques on MDPs[24].

Among related works, one of the earliest attempts at combining stochastic planning models with service composition is a study by Gao et al.[25]. There are works based on Markov-HTN Planning[26], multi-objective optimization[27,28], and lexicographic optimization[29], helpful to model the stochastic behavior 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.

1.1 Contributions

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 process-traces semantics, where only a single action is executed at a time, as commonly done in reasoning about business processes in the DECLARE framework[12]. 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 key contribution of this work extends the findings of De Giacomo et al.[30], including both reachability and safety tasks, introducing a novel reduction technique for safety specification. 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.

We consider two variants of the LTL$_f$ stochastic service composition problem. In the first variant of the problem, as the first objective, we aim to maximize the probability of satisfying at least once an LTL$_f$ formula φ. This is a reachability property: there exists a finite prefix $t_{\leq k}$ of an infinite trace t such that $t_{\leq k} \models \varphi$. This is the classical use of LTL$_f$ to specify synthesis tasks[31]. As the second objective, we consider the minimization of the expected cost of utilization of services, conditioned on the optimal satisfaction of the first objective. In the second variant, as the first objective, we aim to maximize the probability of not violating an LTL$_f$ formula φ. This is a safety property: every finite prefix $t_{\leq k}$ of an infinite trace t is such that $t_{\leq k} \models \varphi$. This is the classical use of LTL$_f$ to specify environment behaviors[32]. As the second objective, we consider minimizing the expected mean cost of utilization of services, conditioned on the optimal satisfaction of the first objective.

Observe that our reachability and safety LTL$_f$ properties are conceptually similar to those formalized in studies by Aminof et al.[33,34], where they use $\exists\varphi$ to denote reachability and $\forall\varphi$ to denote safety, where φ is an arbitrary LTL$_f$ formula. However, the difference in our reachability and safety tasks within our composition framework lies in the fact that, in the satisfaction condition of these tasks, we also impose constraints on whether the services are in an accepting state; thus, we consider both the satisfaction of the LTL$_f$ task and the configurations of the service community.

The solution technique for both variants of the composition problem relies on solving a bi-objective lexicographic optimization[35] over a special MDP, allowing for minimizing 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[5–7,10].

This paper presents an approach for stochastic service composition to find orchestrators that maximize the probability of satisfying the reachability and safety LTL$_f$ tasks and, among such orchestrators, find those who also minimize a cost utilization function. We extended our previous work[30] by (i) generalizing the framework so as to encompass both variants of stochastic service composition, one for reachability and the other for safety LTL$_f$ tasks; (ii) including a new section regarding the composition under LTL$_f$ safety tasks; (iii) discussed in more detail a case study as a conceptual validation of our framework.

1.2 Outline

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. Section 4 studies the stochastic composition problem in the case of the reachability task and provides a solution by reducing the problem to a bi-objective lexicographic optimization on MDPs[35]. Section 5 studies the stochastic composition problem in the case of a safety task and provides reduction to bi-objective lexicographic optimization on MDPs[35]; the reduction is analogous to the reachability case, but with several technical differences that require a deeper analysis. Section 6 shows the conceptual validation 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, Section 7 concludes the paper with final remarks and future work.

2 Preliminaries

2.1 Linear Temporal Logic on finite traces (LTL$_f$)

LTL$_f$ is a variant of Linear Temporal Logic (LTL) interpreted over finite traces, instead of infinite ones[11]. Given a set $\mathcal{P}$ of atomic propositions, LTL$_f$ formulas φ are defined by

$$\varphi ::= tt \mid p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \bigcirc\varphi \mid \varphi\, \mathcal{U}\, \varphi$$

where tt represents the true formula, p 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 $\Diamond\varphi \equiv true\, \mathcal{U}\, \varphi$; 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). In the original proposal, LTL$_f$ formulas were interpreted on finite traces. For simplicity, here we consider the semantics on simple (possibly empty) finite traces, also known as process traces[36] (while the choice between finite-trace and process-trace semantics does not affect the generality of our framework, we prefer process-trace semantics as it aligns more naturally with our context. In this case, the traces result from the services' actions, which are more effectively represented as atomic propositions rather than in a factorized form). A process trace over propositions $\mathcal{P}$ is a sequence $a = a_0 \ldots a_{l-1}$ where $a_i \in \mathcal{P}$, and l is the length of the trace. The empty trace is denoted with ϵ.

Given a finite (possibly empty) trace a, we define when an LTL$_f$ formula φ is true in a at the instant $i \in \{0, \ldots, l-1\}$, denoted by $a, i \models \varphi$, by inductively considering subformulas as follows[37]:

- $a, i \models tt$;

- $a,i \models p$ iff $a_i = p$;
- $a,i \models \neg\varphi$ iff $a,i \not\models \varphi$;
- $a,i \models \varphi_1 \wedge \varphi_2$ iff $a,i \models \varphi_1$ and $a,i \models \varphi_2$;
- $a,i \models \bigcirc\varphi$ iff $i < l-1$ and $a,i+1 \models \varphi$;
- $a,i \models \varphi_1 \,\mathcal{U}\, \varphi_2$ iff there exists j with $i \geq j < l$ such that $a,j \models \varphi_2$, and for all k with $i \leq k < j$ we have that $a,k \models \varphi_1$.

Whenever $a,0 \models \varphi$ holds, we say that a is a *model* of φ, or that a *satisfies* φ. We also write $a \models \varphi$ as an abbreviation for $a,0 \models \varphi$.

A deterministic finite automaton (DFA) is a tuple $\mathcal{A} = \langle \mathcal{P}, Q, q_0, F, \delta \rangle$ where: (i) $\mathcal{P}$ 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) $\delta : Q \times \mathcal{P} \to Q$ is a total transition function. Note that the DFA alphabet is the same as the set of traces that satisfy the formula φ. A trace a is *accepted* by a DFA $\mathcal{A}$ iff a sequence of states $q_0,\ldots,q_l$ exists in Q such that: (i) $q_{i+1} \in \delta(q_i, a_i)$ for $i = 0,\ldots,l-1$; and (ii) $q_l \in F$. We denote with $\mathcal{L}(\mathcal{A})$ the set of traces accepted by $\mathcal{A}$. An LTL$_f$ formula can be transformed into an equivalent DFA $\mathcal{A}_\varphi$ in at most 2EXPTIME[11].

2.2 Markov Decision Process (MDP)

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) $P : (S \times A) \to \Delta(S)$ is the transition probability function, i.e. a mapping from state-action pairs to probability distributions over S. With Supp(d) , we denote the support of a probability distribution d. An infinite path $\rho \in (S \times A)^\omega$ is a sequence $\rho = s_0 a_1 s_1 a_2 \ldots$, where $s_i \in S$, $a_{i+1} \in A$, and $s_{i+1} \in \text{Supp}(P(s_i, a_{i+1}))$, for all $i \in \mathbb{N}$. Similarly, a finite path $\rho \in (S \times A)^* \times S$ is a finite sequence $\rho = s_0 a_1 s_1 a_2 \ldots a_m s_m$. For any path ρ of length at least j and any $i \leq j$, we let $\rho[i:j]$ denote the subsequence $s_i a_{i+1} s_{i+1} a_{i+2} \ldots a_j s_j$. Moreover, let $\text{last}(\rho) = s_m$ to denote the last state visited by the finite path ρ. We use $\text{Paths}^\omega_{\mathcal{M}} = (S \times A)^\omega$ and $\text{Paths}_{\mathcal{M}} = (S \times A)^* \times S$ to denote the set of infinite and finite paths, respectively. A policy $\pi : \text{Paths}_{\mathcal{M}} \to A$ maps a finite path $\rho \in \text{Paths}_{\mathcal{M}}$ to an action $a \in A$. We denote with $\text{Paths}_{\mathcal{M}_\pi}$ the set of finite paths over $\mathcal{M}$ whose actions are compatible with π. Given a finite path $\rho = s_0 a_1 \ldots a_m s_m$, the *cylinder* of ρ, denoted by $\text{Paths}^\omega_{\mathcal{M}}(\rho)$, 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 $\text{Paths}^\omega_{\mathcal{M}_\pi}(\rho)$ for all $\rho \in \text{Paths}_{\mathcal{M}_\pi}$. For a state s in S, a measure is defined for the cylinder sets as $\mathbb{P}_{\mathcal{M}_\pi,s}(\text{Paths}^\omega_{\mathcal{M}_\pi}(s_0 a_1 s_1 \ldots a_m s_m)) = \prod_{k=0}^{m-1} P(s_{k+1} | s_k, a_{k+1})$ if $s_0 = s$ and for all k, $a_{k+1} = \pi(\rho[0:k])$, otherwise 0. We also have $\mathbb{P}_{\mathcal{M}_\pi,s}(\text{Paths}^\omega_{\mathcal{M}_\pi}(s)) = 1$ and $\mathbb{P}_{\mathcal{M}_\pi,s}(\text{Paths}^\omega_{\mathcal{M}_\pi}(s')) = 0$ 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 $\mathcal{R} \subseteq \text{Paths}_{\mathcal{M}_\pi}$ is a set of finite paths forming pairwise disjoint cylinder sets, then $\mathbb{P}_{\mathcal{M}_\pi,s}(\bigcup_{\rho\in\mathcal{R}} \text{Paths}^\omega_{\mathcal{M}_\pi}(\rho)) = \sum_{\rho\in\mathcal{R}} \mathbb{P}_{\mathcal{M}_\pi,s}(\text{Paths}^\omega_{\mathcal{M}_\pi}(\rho))$. 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 Stochastic service composition framework

In this section, we present our service composition framework in stochastic settings.

A *(stochastic) service* is a tuple $\mathcal{S} = \langle \Sigma, A, \sigma_0, F, P, C \rangle$, 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. Without loss of generality, we assume that for any $\sigma \in \Sigma$, there exists at least one $a \in A$ such that $|\text{Supp}(\sigma, a)| > 0$. A *(*stochastic*)* service community is a collection of stochastic services $C = \{\mathcal{S}_1, \ldots, \mathcal{S}_n\}$. A trace *of* C is an infinite alternating sequence of the form $t = (\sigma_{10} \ldots \sigma_{n0}), (a_1, o_1), \ldots$, where σ_{i0} is the initial state of every service $\mathcal{S}_i$ and, for every $k \geq 1$, we have (i) $\sigma_{ik} \in \Sigma_i$ for all $i \in \{1,\ldots,n\}$, (ii) $o_k \in \{1,\ldots,n\}$ is the service index chosen by the orchestrator at step k, (iii) $a_k \in A$, and (iv) for all i, $\sigma_{ik} \in \text{Supp}(P_i(\sigma_{i,k-1}, a_{ik}))$ if $o_k = i$, and $\sigma_{ik} = \sigma_{i,k-1}$ otherwise. A history of C is a finite prefix of a trace of C. We denote the length of h with $|h| = m$, and we denote the service state configuration at the last step of h, i.e., $(\sigma_{1m} \ldots \sigma_{nm})$, with last($h$). Given a trace t, we call states(t) sequence of states of t, i.e. $\text{states}(t) = (\sigma_{10} \ldots \sigma_{n0}), (\sigma_{11} \ldots \sigma_{n1}), \cdots$. The choices of a trace t, denoted with choices(t), is the sequence of actions in t, i.e. $\text{choices}(t) = (a_1, o_1), (a_m, o_m), \ldots$. Note that, due to nondeterminism, there might be many traces of C associated with the same run. Moreover, we define the *action run* of a trace t, denoted with actions(t), the projection of choices(t) only to the components in A. The predicates states, choices and actions are defined also on history h, in a similar way. Note that both choices(h) and actions(h) are empty sequences if $h = (\sigma_{10} \ldots \sigma_{n0})$.

An orchestrator is a function $\gamma : (\Sigma_1 \times \cdots \times \Sigma_n)^* \to A \times \{1 \ldots n\}$ that, given a sequence of states $(\sigma_{10} \ldots \sigma_{n0}) \ldots (\sigma_{1m} \ldots \sigma_{nm})$, returns the action to perform $a \in A$, and the service (actually the service index) that will perform it. Given a trace t, with prefixes(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 C if for all $k \geq 0$, we have $(a_{k+1}, o_{k+1}) = \gamma((\sigma_{10} \ldots \sigma_{n0}) \ldots (\sigma_{1k} \ldots \sigma_{nk}))$. Let $\mathcal{T}_{\gamma,C}$ be the set of such executions. Note that due to the services' nondeterminism, we can have many executions for the same orchestrator despite the orchestrator being a deterministic function. If $h \in \text{prefixes}(t)$ for some (infinite) execution $t \in \mathcal{T}_{\gamma,C}$, we call h a finite execution, or a history, of γ over C, and we define $\mathcal{H}_{\gamma,C}$ to be the set of finite executions of γ over C. Analogously to what has been done for MDPs, for a finite execution h of γ over C, we use $\mathcal{T}_{\gamma,C}(h)$ to denote the set of all (infinite) executions $t \in \mathcal{T}_{\gamma,C}$ such that $h \in \text{prefixes}(t)$. Moreover, the σ-algebra associated with the stochastic behaviour of the orchestrator γ over the stochastic community C is the smallest σ-algebra that contains the trace sets $\mathcal{T}_{\gamma,C}(h)$, for all finite executions h, with the unique probability measure over it defined as:

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

where $a_k = \gamma((\sigma_{10} \ldots \sigma_{n0}), \ldots (\sigma_{1k} \ldots \sigma_{nk}))$. In particular, note that $\mathbb{P}_{\gamma,C}(\mathcal{T}_{\gamma,C}(\langle(\sigma_{10} \ldots \sigma_{n0})\rangle)) = 1$.

Example 1 This example is inspired by the 'garden bots system' scenario[8]. In this environment, the garden bots can: *clean* the garden by picking fallen leaves and removing dirt, *water* the plants, and *pluck* the ripe fruits and flowers. 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 service specifications are shown. Transitions labels are of the form $\langle action, prob, reward \rangle$.

We consider an orchestrator γ such that (i) always picks $\mathcal{B}_1$, and in state a_0 chooses the unique available action *clean* while in state a_1 chooses the unique available action *empty*. The probability of the finite execution $h_1 = (a_0, b_0, c_0), (clean, \mathcal{B}_1), (a_0, b_0, c_0)$ to occur under γ is $\mathbb{P}_{\gamma,C}(h_1) = 0.8$, meaning that the transition $a_0 \xrightarrow{clean} a_0$ is taken. Instead, the probability of the finite execution $h_1 = (a_0, b_0, c_0), (clean, \mathcal{B}_1), (a_1, b_0, c_0)$ to occur is $\mathbb{P}_{\gamma,C}(h_1) = 0.2$, meaning that the transition $a_0 \xrightarrow{clean} a_1$ is taken.

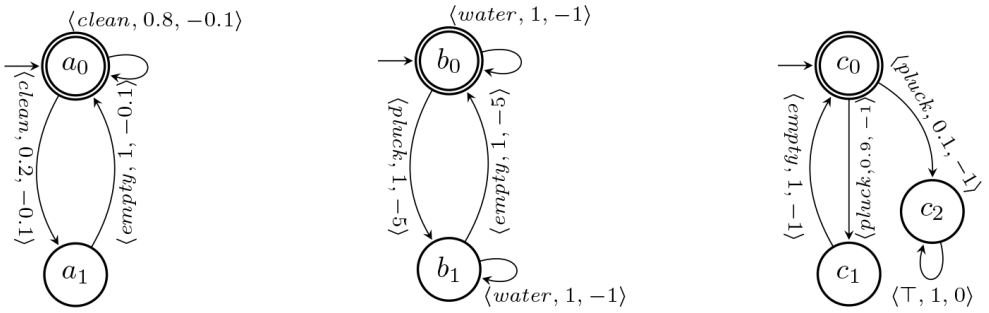

Fig. 1 The garden bot systems and the DFA of the LTL$_f$ reachability task.

4 Composition of stochastic services for reachability LTL$_f$ tasks

In this section, we present our service composition framework in stochastic settings where the goal is to maximize the probability of satisfying the LTL$_f$ reachability task of φ, where φ is an LTL$_f$ formula. Intuitively, we are interested in orchestrators that maximize the probability of satisfaction of the LTL$_f$ specification, even when the specification cannot be surely satisfied (e.g., due to stochastic misbehavior 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.

Formally, we consider a LTL$_f$ formula φ expressed over the set of actions A, and consider a community of n services $\mathcal{C} = \{\mathcal{S}_1, \ldots, \mathcal{S}_n\}$, where each set of actions $A_i \subseteq A$. We say that some finite execution h is *successful (for $\mathcal{C}$ and φ)*, denoted with successful$_{\mathcal{C},\varphi}(h)$, if there exist some execution $h' \in$ prefixes(h) such that the following two conditions hold: (1) actions$(h') \models \varphi$ and (2) all service states $\sigma_i \in$ last(states(h')) are such that $\sigma_i \in F_i$. If for execution $t \in \mathcal{T}_{\gamma,\mathcal{C}}$ there exist a finite prefix history $h \in$ prefixes(t) such that successful$_{\mathcal{C},\varphi}(h)$ for φ, we say that t is *successful for φ*. Finally, we say that an orchestrator γ *realizes the* LTL$_f$ *reachability task φ with $\mathcal{C}$* if, for all traces $t \in \mathcal{T}_{\gamma,\mathcal{C}}$, t is successful with respect to φ. When $\mathcal{C}$ and φ are clear from the context, we omit the suffix in successful$_{\mathcal{C},\varphi}$ and only write successful.

Example 2 Continuing example 1, we consider the following sequential task: *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 LTL$_f$, the task can be expressed as

$$\varphi = clean \wedge \bigcirc(clean\, \mathcal{U}\, ((water \wedge \bigcirc pluck) \vee (pluck \wedge \bigcirc water)))$$

A successful execution is

$$h_1 = \langle (a_0,b_0,c_0),(clean,\mathcal{B}_1),(a_0,b_0,c_0),(water,\mathcal{B}_2),(a_0,b_0,c_0),\\ (pluck,\mathcal{B}_2),(a_0,b_1,c_0),(empty,\mathcal{B}_2),(a_0,b_0,c_0)\rangle$$

Note that, before the action *"empty"* is taken, the LTL$_f$ task φ is already satisfied; however, the garden bot $\mathcal{B}_2$ is not in an accepting state. The last action in the execution, *"empty"*, is necessary to move the bot $\mathcal{B}_2$ to its accepting state. Another successful execution is

$$h_2 = \langle (a_0,b_0,c_0),(clean,\mathcal{B}_1),(a_0,b_0,c_0),(pluck,\mathcal{B}_3),(a_0,b_0,c_1),\\ (empty,\mathcal{B}_3),(a_0,b_0,c_0),(water,\mathcal{B}_2),(a_0,b_0,c_0)\rangle$$

On the other hand, any extension of the execution

$$h_3 = \langle (a_0,b_0,c_0),(clean,\mathcal{B}_1),(a_0,b_0,c_0),(pluck,\mathcal{B}_3),(a_0,b_0,c_2)\rangle$$

will not be successful since service $\mathcal{B}_3$ can never reach one of its accepting states from c_2.

The satisfaction probability of the reachability task φ under orchestrator γ and community $\mathcal{C}$ is given by:

$$\mathcal{P}^{\text{reach}}_{\mathcal{C},\varphi}(\gamma) = \mathbb{P}_{\gamma,\mathcal{C}}\Big(\{t \in \mathcal{T}_{\gamma,\mathcal{C}} \,\big|\, \text{successful}(t)\}\Big) \tag{2}$$

Intuitively, $\mathcal{P}^{\text{reach}}_{\mathcal{C},\varphi}(\gamma)$ is the probability measure (under $\mathbb{P}_{\gamma,\mathcal{C}}$) of the set of **successful** executions generated by executing the orchestrator γ over the service community $\mathcal{C}$.

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

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

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

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

Intuitively, we fix a lexicographic order on the objective functions $\mathcal{P}^{\text{reach}}_{\mathcal{C},\varphi}$ and $\mathcal{J}^{\text{cost}}_{\mathcal{C},\varphi}$, 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.

First, let $\mathcal{H}^{\varphi}_{\gamma,\mathcal{C}}$ be the set of finite executions h of orchestrator γ on service community $\mathcal{C}$ that start from $\sigma_{10}\ldots\sigma_{n0}$ such that (i) successful(h) holds, and (ii) there is no prefix history $h' \in$ prefixes(h) such that successful(h') holds:

$$\mathcal{H}^{\varphi}_{\gamma,\mathcal{C}} = \{h \in \mathcal{H}_{\gamma,\mathcal{C}} : \text{successful}(h) \wedge \nexists h' \in \text{prefixes}(h) \text{ s.t. } h' \\ \neq h \wedge \text{successful}(h)\} \tag{5}$$

Intuitively, such a set only contains the finite executions h such that they are successful for φ for the first time. In fact, by definition of $\mathcal{H}^{\varphi}_{\gamma,\mathcal{C}}$, if $h \in \mathcal{H}^{\varphi}_{\gamma,\mathcal{C}}$ then successful(h) holds.

Lemma 1 The following equality holds:

$$\{t \in \mathcal{T}_{\gamma,\mathcal{C}} \,\big|\, \text{successful}(t)\} = \bigcup_{h\in\mathcal{H}^{\varphi}_{\gamma,\mathcal{C}}} \mathcal{T}_{\gamma,\mathcal{C}}(h)$$

Proof We prove the two set inclusions (i) $\subseteq$ and (ii) $\supseteq$ separately.

(i) if $t \in \mathcal{T}_{\gamma,\mathcal{C}}$ and successful(t), then there exist a prefix $h \in$ prefixes(t) such that h is successful. Let h' be the shortest of such prefixes: this means that no prefix h" of h' is successful. From this, by definition of $\mathcal{H}^{\varphi}_{\gamma,\mathcal{C}}$, it follows that $h' \in \mathcal{H}^{\varphi}_{\gamma,\mathcal{C}}$. Then, given that t is an

execution of γ over C with prefix h', by definition of $\mathcal{T}_{\gamma,C}$, we have that $t \in \mathcal{T}_{\gamma,C}(h')$, and therefore the thesis holds.

(ii) let $t \in \bigcup_{h \in \mathcal{H}^{\varphi}_{\gamma,C}} \mathcal{T}_{\gamma,C}(h)$. Let $h \in \mathcal{H}^{\varphi}_{\gamma,C}$ be the history such that $t \in \mathcal{H}^{\varphi}_{\gamma,C}$, which must exist by assumption. By definition of $\mathcal{H}^{\varphi}_{\gamma,C}$, we have successful($h$), and by definition of successful, since $h \in$ prefixes(t), we also have successful(t). Moreover, $\mathcal{T}_{\gamma,C}(h) \subseteq \mathcal{T}_{\gamma,C}$, which concludes the proof. □

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

Lemma 2 *If γ realizes the reachability task φ over C, then* $\bigcup_{h \in \mathcal{H}^{\varphi}_{\gamma,C}} \mathcal{T}_{\gamma,C}(h) = \mathcal{T}_{\gamma,C}(\langle\langle(\sigma_{10} \dots \sigma_{n0})\rangle\rangle)$.

Proof We prove (i) $\bigcup_{h \in \mathcal{H}^{\varphi}_{\gamma,C}} \mathcal{T}_{\gamma,C}(h) \subseteq \mathcal{T}_{\gamma,C}(\langle\langle(\sigma_{10} \dots \sigma_{n0})\rangle\rangle)$ and (ii) $\bigcup_{h \in \mathcal{H}^{\varphi}_{\gamma,C}} \mathcal{T}_{\gamma,C}(h) \supseteq \mathcal{T}_{\gamma,C}(\langle\langle(\sigma_{10} \dots \sigma_{n0})\rangle\rangle)$ 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 $t \in \mathcal{T}_{\gamma,C}(\langle\langle(\sigma_{10} \dots \sigma_{n0})\rangle\rangle)$, by definition of realizing orchestrator, they have a prefix $h' \in$ prefixes(h) that is successful. In particular, if $h"$ is the shortest prefix of h that is successful, then $h" \in \mathcal{H}^{\varphi}_{\gamma,C}$ and $t \in \mathcal{T}_{\gamma,C}(h")$. This implies that $t \in \bigcup_{h" \in \mathcal{H}^{\varphi}_{\gamma,C}} \mathcal{T}_{\gamma,C}(h")$. □

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

Proof ($\Rightarrow$) If an orchestrator γ realizes φ, then all infinite executions $t \in \mathcal{T}_{\gamma,C}$ are successful. By Lemma 1, this implies that $t \in \bigcup_{h \in \mathcal{H}^{\varphi}_{\gamma,C}} \mathcal{T}_{\gamma,C}(h)$. By Lemma 2, this set is equal to $\mathcal{T}_{\gamma,C}(\langle\langle(\sigma_{10} \dots \sigma_{n0})\rangle\rangle)$. Since by definition $\text{Supp}(\mathbb{P}_{\gamma,C}) \subseteq \mathcal{T}_{\gamma,C}(\langle\langle(\sigma_{10} \dots \sigma_{n0})\rangle\rangle)$, we have that $\mathcal{P}^{\text{reach}}_{C,\varphi}(\gamma) = \mathbb{P}_{\gamma,C}(\bigcup_{h \in \mathcal{H}^{\varphi}_{\gamma,C}} \mathcal{T}_{\gamma,C}(h)) = \mathbb{P}_{\gamma,C}(\mathcal{T}_{\gamma,C}(\langle\langle(\sigma_{10} \dots \sigma_{n0})\rangle\rangle)) = 1$.

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

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

Proof Since by assumption the reachability task φ is realizable, then there exists an orchestrator γ' that realizes it. By Theorem 1, we can deduce that the optimal value of $\mathcal{P}^{\text{reach}}_{C,\varphi}(\gamma')$ is 1. Moreover, by assumption and by Eq. (4), it follows that $\gamma \in \Gamma_{\mathcal{P}^{\text{reach}}_{C,\varphi}}$, i.e. $\mathcal{P}^{\text{reach}}_{C,\varphi}(\gamma) = 1$, 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$_f$ Reachability Tasks). Given the pair (C, φ), where φ is an LTL$_f$ reachability task over the set of actions A, and C is a community of n stochastic services $C = \{\mathcal{S}_1, \dots, \mathcal{S}_n\}$, compute an orchestrator that is optimal according to Eq. (4).

Interestingly, Theorems 1 and 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 $\max_{\gamma} \mathcal{P}^{\text{reach}}_{C,\varphi}(\gamma) = 1$.

Example 3 Continuing example 2, we aim to find a composition of the available bots $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$ so as to maximize the probability of the satisfaction of the reachability task 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 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.

4.1 Solution technique

The 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 study by Busatto-Gaston et al.[35]: while as the second objective they considered the expected number of steps to a target, here we consider the expected cost. Given an instance of the reachability composition problem (C, φ), our technique breaks down into the following steps: (1) first, we translate the LTL$_f$ formula into the equivalent DFA $\mathcal{A}_{\varphi}$; then (2) we combine $\mathcal{A}_{\varphi}$ with the services, computing a *product of* $\mathcal{A}_{\varphi}$ with the stochastic services in C, obtaining a new MDP, $\mathcal{M}'$, that we call the 'composition MDP'; (3) we solve the resulting MDP, finding a policy π for $\mathcal{M}'$ that is optimal w.r.t. the bi-objective lexicographic function, as in study by Busatto-Gaston et al.[35], and then (4) we derive an orchestrator γ from π that is optimal w.r.t. Eq. (4). We now detail each step.

Step 1. The DFA of an LTL$_f$ formula can be computed by exploiting a well-known correspondence between LTL$_f$ formulas and automata on finite words[11]. That is, we can compute a DFA $\mathcal{A}_{\varphi} = (A, Q, q_0, F, \delta)$ that is equivalent to the specification φ, which can be double-exponentially larger than the size of the formula.

Step 2. Next, we define the Composition MDP $\mathcal{M}' = (S', A', P', s'_0)$ as follows:

- $S' = Q \times \Sigma_1 \times \dots \times \Sigma_n$;
- $A' = A \times \{1 \dots n\}$;
- $s'_0 = (q_0, \sigma_{10} \dots \sigma_{n0})$;
- $P'(q', \sigma'_1 \dots \sigma'_i \dots \sigma'_n | q, \sigma_1 \dots \sigma_i \dots \sigma_n, (a,i)) = P_i(\sigma'_i | \sigma_i, a)$ if $\delta(q,a) = q'$ and $\sigma'_j = \sigma_j$ for all $j \neq i$, otherwise 0.

Moreover, let the composition cost function $C' : S' \times A' \to \mathbb{R}^+$ be defined as $C'((q, \sigma_1 \dots \sigma_i \dots \sigma_n), (a,i)) = C_i(\sigma_i, a)$.

We are interested in computing optimal policies for $\mathcal{M}'$, where the optimality is defined as follows. As the set of target states T, we consider:

$$T = \{(q, \sigma_1, \dots, \sigma_n) \mid (q, \sigma_1, \dots, \sigma_n) \in S' \wedge q \in F \wedge \forall i = 1, \dots, n : \sigma_i \in F_i\}$$

Intuitively, $T \subseteq S'$ is the set of states of $\mathcal{M}'$ where the state component from $\mathcal{A}$, q, is accepting, and the state components from the services $\mathcal{S}_i$, σ_i, are accepting in $\mathcal{S}_i$. We consider the bi-objective lexicographic optimization over $\mathcal{M}'$, similarly to what has been done in study by Busatto-Gaston et al.[35]. 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. $\text{cost}_T(\rho) = \sum_{k=0}^{i} C'(s'_k, a'_k)$ 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 $\mathbb{E}_{\rho \sim \mathcal{M}'_{\pi}, s'_0}[\text{cost}_T(\rho) | \Diamond T]$ 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.:

$$\pi \in \Pi_{\mathcal{M}',s_0}(\Diamond T) \text{ and } \pi \in \arg\min_{\pi'} \mathbb{E}_{\rho \sim \mathcal{M}'_{\pi}, s'_0}[\text{cost}_T(\rho) | \Diamond T] \qquad (6)$$

Step 3. The solution technique we will use is based on the work[35], where the authors propose a two-stage technique to find an optimal policy for a bi-objective lexicographic function in the form of Eq. (6). 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 Busatto-Gaston et al.[35] 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[38] 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[24] on the pruned MDP. The two subproblems can be solved efficiently using standard planning algorithms, e.g., Value Iteration or Linear Programming.

Step 4. Once an optimal policy is found, we can obtain its equivalent γ as follows: for any finite prefix of a run $\rho = (q_0,\sigma_{10}\dots\sigma_{n0}),(a_1,o_1),\dots(a_m,o_m),(q_m,\sigma_{1m}\dots\sigma_{nm}),\dots$, we set $\gamma((\sigma_{10}\dots\sigma_{n0})\dots(\sigma_{1m}\dots\sigma_{nm})) = (a_{m+1},o_{m+1})$, where $\pi(\rho) = (a_{m+1},o_{m+1})$.

Now we aim to establish a relationship between optimal orchestrators according to Eq. (4), and optimal policies for $\mathcal{M}'$ according to Eq. (6). Given an infinite run $\rho = (q_0,\sigma_{10}\dots\sigma_{n0}),(a_1,o_1)\dots$, we define the trace $t = \tilde{\tau}_{\varphi,\mathcal{C}}(\rho) = (\sigma_{10}\dots\sigma_{n0}),(a_1,o_1),\dots$. The definition easily applies to finite prefixes of ρ but this time mapping into histories of t.

In the following, we are going to prove a sequence of lemmata. Lemma 3 shows that, once fixed a policy π over $\mathcal{M}'$, there is a one-to-one correspondence 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 4 shows that the probabilities of finite paths and histories are the same; Lemma 5 shows that paths that end with states in T correspond to successful histories; and Lemma 6 shows a correspondence between paths in $\text{Paths}_{T,\mathcal{M}'_\pi}$ and $\mathcal{H}^{\varphi}_{\gamma,\mathcal{C}}$.

Lemma 3 Let π be a policy for $\mathcal{M}'$ and let γ be its equivalent orchestrator. Moreover, let $\rho \in \text{Paths}^{\omega}_{\mathcal{M}'_\pi}$ and t be a trace such that $t = \tilde{\tau}_{\varphi,\mathcal{C}}(\rho)$. Then, $\rho \in \text{Paths}^{\omega}_{\mathcal{M}'_\pi}(\langle s'_0\rangle)$ iff $t \in \mathcal{T}_{\gamma,\mathcal{C}}(\langle(\sigma_{10}\dots\sigma_{n0})\rangle)$.

Proof Let the infinite path $\rho \in \text{Paths}^{\omega}_{\mathcal{M}'_\pi}(\langle s'_0\rangle)$, and the infinite trace $t = \tilde{\tau}_{\varphi,\mathcal{C}}(\rho))$. 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 $k \geq 0$.

($\Rightarrow$) Consider the $(k+1)$-th action according to π, i.e. $\pi(\rho[0:k]) = (a_{k+1},o_{k+1})$, and its successor state $\rho[k+1] = (q_{k+1},\sigma_{1,k+1},\dots,\sigma_{n,k+1})$. By inductive hypothesis, we have that $h = t[0:k]$ is a valid execution under orchestrator γ, where $t = \tilde{\tau}_{\varphi,\mathcal{C}}(\rho)$. We need to show that $h' = t[0:k],(a_{k+1},o_{k+1}),(\sigma_{1,k+1},\dots,\sigma_{n,k+1})$ is a valid extension of h under orchestrator γ. 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) $a_{k+1} \in A$. We also aim to show that (iv) for all $j = 1,\dots,n$, $\sigma_{j,k+1} \in \text{Supp}(P_j(\sigma_{j,k},a_{k+1}))$ if $j = o_{k+1}$; otherwise $\sigma_{j,k+1} = \sigma_{j,k}$. Proposition (iv) holds because of the definition of P' and by the hypothesis of ρ being a path of $\mathcal{M}'$ (the support of the transition probability function of all the states along a path must be non-empty). The propositions (i), (ii), (iii), (iv) are precisely the conditions that each step of a valid execution must satisfy. Moreover, by construction of γ, $(a_{k+1},o_{k+1}) = \gamma(\text{states}(t[0:k]))$, 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 under orchestrator γ.

($\Leftarrow$) Now, consider a finite execution under orchestrator γ, $h = t[0:k]$, and its extension $h' = t[0:k],(a_{k+1},o_{k+1}),(\sigma_{1,k+1}\dots\sigma_{n,k+1})$. We need to show that the finite path prefix $\rho[0:k+1]$ is a valid finite path in $\mathcal{M}'$ under policy π. Since by inductive hypothesis the claim holds up to the k-th step, we only need to show that the $(k+1)$-th step defined as $(a_{k+1},o_{k+1}),(q_{k+1},\sigma_{1,k+1}\dots\sigma_{n,k+1})$, for some q_{k+1}, is a valid extension of $\rho[0:k]$ under policy π. By construction, we have that $\pi(\rho[0:k]) = \gamma(\text{states}(h')) = (a_{k+1},o_{k+1})$. Regarding the successor state, we have to show that $(q_{k+1},\sigma_{1,k+1}\dots\sigma_{n,k+1}) \in \text{Supp}(P')$. But this follows from our choice of $q_{k+1} = \delta(q_k,a_{k+1})$ and by the condition (iv) of a valid execution step.

By induction, the claim also holds for any arbitrary position, and therefore $\rho \in \text{Paths}^{\omega}_{\mathcal{M}'_\pi}(\langle s'_0\rangle)$ iff $t \in \mathcal{T}_{\gamma,\mathcal{C}}(\langle(\sigma_{10}\dots\sigma_{n0})\rangle)$. □

Lemma 4 Let π a policy on $\mathcal{M}'$, γ be its equivalent orchestrator, $\rho = s'_0a_1\dots s'_m \in \text{Paths}_{\mathcal{M}'_\pi}(s'_0)$ be a finite path on $\mathcal{M}'$, and $h = \tilde{\tau}_{\varphi,\mathcal{C}}(\rho)$ be its associated history. Then, $\mathbb{P}_{\mathcal{M}'_\pi,s'_0}(\text{Paths}^{\omega}_{\mathcal{M}'_\pi}(\rho)) = \mathbb{P}_{\gamma,\mathcal{C}}(\mathcal{T}_{\gamma,\mathcal{C}}(h))$.

Proof

$$\mathbb{P}_{\mathcal{M}'_\pi,s'_0}(\text{Paths}^{\omega}_{\mathcal{M}'_\pi}(\rho)) = \prod_{k=1}^{m} P'(s'_k \mid s'_{k-1},(a_k,o_k)) \tag{7}$$

$$= \prod_{k=1}^{m} P_{o_k}(\sigma_{o_k,k} \mid \sigma_{o_k,k-1},(a_k,o_k)) \tag{8}$$

$$= \mathbb{P}_{\gamma,\mathcal{C}}(\mathcal{T}_{\gamma,\mathcal{C}}(h)) \tag{9}$$

where step 7 is by definition of the probability of a cylinder set, step 8 by definition of P' in $\mathcal{M}'$, and step 9 by Eq. (1). □

Lemma 5 Let $\rho = s_0a_1\dots s_m \in \text{Paths}_{\mathcal{M}'_\pi}$ be a finite path on $\mathcal{M}'_\pi$ for some policy π, and let $h = \tilde{\tau}_{\varphi,\mathcal{C}}(\rho)$ be its associated history. Then, successful(h) iff there exists $k \in [0,m]$ such that $s_k \in T$.

Proof Let k be an integer between 0 and m. By construction of $\mathcal{M}'$, we have that $s_k = (q_k,\sigma_{1k}\dots\sigma_{nk}) \in T$ iff (a) $q_k \in F$ and (b) for all $i = 1,\dots,n$ we have $\sigma_{ik} \in F_i$. The proposition (a) means that the sequence of states $r = q_0,\dots,q_k$ is an accepting run over $\mathcal{A}_\varphi$ for trace actions(h') $= a_1,\dots a_k$, where $h' = h[0:k]$. By definition of acceptance, the above holds iff actions(h') $\in \mathcal{L}(\mathcal{A}_\varphi)$. By the correctness of the construction of $\mathcal{A}_\varphi$, this is true iff actions(h') $\models \varphi$. Finally, since $h' \in$ prefixes(h), we have actions(h') $\models \varphi$ and (b) both imply that successful(h) holds.

Conversely, assume successful(h). Then by definition there exists a prefix $h' = h[0:k]$ such that actions(h') $\models \varphi$ and such that (c) $\sigma_{ik} \in F_i$ for all $i = 1,\dots,n$. By the correctness of the construction of $\mathcal{A}_\varphi$, actions(h') $\models \varphi$ iff actions(h') $\in \mathcal{L}(\mathcal{A}_\varphi)$, meaning that there exists an accepting run $r = q_0,\dots,q_k$ over $\mathcal{A}_\varphi$, where (d) $q_k \in F$. Conditions (c) and (d) imply that $\rho[k] = s_k = (q_k,\sigma_{1k}\dots\sigma_{nk}) \in T$, by definition of the set T. □

Let $\text{Paths}_{T,\mathcal{M}'_\pi}(s'_0)$ 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. $\text{Paths}_{T,\mathcal{M}'_\pi}(s'_0) = ((S' \setminus T) \times A)^* T \cap \text{Paths}_{\mathcal{M}'_\pi}(s'_0)$.

Lemma 6 $\rho \in \text{Paths}_{T,\mathcal{M}'_\pi}(s'_0)$ *iff* $\tilde{\tau}_{\varphi,C}(\rho) \in \mathcal{H}^{\varphi}_{\gamma,C}$

Proof By Lemma 5, $\rho \in \text{Paths}_{T,\mathcal{M}'_\pi}$ iff $h = \tilde{\tau}_{\varphi,C}(\rho)$ is successful, since ρ ends in a state in T. Moreover, since $\text{Paths}_{T,\mathcal{M}'_\pi} \subseteq \text{Paths}_{\mathcal{M}'_\pi}$, by Lemma 3, we have that $\rho \in \text{Paths}_{T,\mathcal{M}'_\pi}$ 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 5, this holds iff $h' = \tilde{\tau}_{\varphi,C}(\rho')$ is not successful, meaning that does not exist a prefix $h' \in \text{prefixes}(h)$ with $h' \neq h$ such that h' is successful. But this is precisely the membership condition for $\mathcal{H}^{\varphi}_{\gamma,C}$. □

This result shows the correctness of our technique:

Theorem 3 Let (C,φ) be an instance of Problem 1, and let $\mathcal{M}'$ be the composition MDP for C and φ. We have that π is optimal (w.r.t. Eq. [6]) if 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 $\gamma = \arg\max_{\gamma'} \mathcal{P}^{\text{reach}}_{C,\varphi}(\gamma')$. For any pair π and its equivalent γ, we have:

$$\mathbb{P}_{\pi,s'_0}(\Diamond T) = \sum_{\rho_T \in \text{Paths}_{T,\pi}(s'_0)} \mathbb{P}_{\mathcal{M}'_\pi,s'_0}(\text{Paths}^{\omega}_{\mathcal{M}'_\pi}(\rho_T)) \tag{10}$$

$$= \sum_{h \in \mathcal{H}^{\varphi}_{\gamma,C}} \mathbb{P}_{\gamma,C}(\mathcal{T}_{\gamma,C}(h)) \tag{11}$$

$$= \mathbb{P}_{\gamma,C}\Big(\bigcup_{h \in \mathcal{H}^{\varphi}_{\gamma,C}} \mathcal{T}_{\gamma,C}(h) \Big) \tag{12}$$

$$= \mathcal{P}^{\text{reach}}_{C,\varphi}(\gamma) \tag{13}$$

where step 10 is by definition of probabilistic reachability, step 11 is by Lemma 4 and 6, step 12 is by disjointness of all $\mathcal{T}_{\gamma,C}(h)$ for $h \in \mathcal{H}^{\varphi}_{\gamma,C}$, and step 13 is by Lemma 1 and Eq. (2). From this, we obtain that $\pi^* = \arg\max_{\pi'} \mathbb{P}_{\mathcal{M}_{\pi'},s'_0}(\Diamond T)$ iff $\gamma^* = \arg\max_{\gamma'} \mathcal{P}^{\text{reach}}_{C,\varphi}(\gamma')$.

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

$$\mathbb{E}_{\rho \sim \mathcal{M}'_\pi,s'_0}[\text{cost}_T(\rho) \mid \Diamond T] = \sum_{\rho_T \in \text{Paths}_{T,\pi}(s'_0)} \mathbb{P}_{\mathcal{M}'_{\pi'},s'_0}(\text{Paths}^{\omega}_{\mathcal{M}'_\pi}(\rho_T)) \cdot \sum_{k=0}^{|\rho_T|} C'(s'_k, a'_{k+1}) \tag{14}$$

$$= \sum_{h \in \mathcal{H}^{\varphi}_{\gamma,C}} \mathbb{P}_{\gamma,C}(\mathcal{T}_{\gamma,C}(h)) \cdot \sum_{k=0}^{|h|} C_{o_{k+1}}(\sigma_{o_k,k}, a_{k+1}) \tag{15}$$

$$= \mathbb{E}_{h \sim \mathbb{P}_{\gamma,C}}\Big[\sum_{k=1}^{|h|} C_{o_k}(\sigma_{o_k,k-1}, a_k) \Big| \text{successful}(h) \Big] \tag{16}$$

$$= \mathcal{J}^{\text{cost}}_{C,\varphi}(\gamma) \tag{17}$$

where step 14 by definition of total expected cost conditioned on reaching of target states T, step 15 by construction of $\mathcal{M}'$ and by Lemma 4 and 6, step 16 by definition of total expected cost on successful executions of γ, and step 17 by Eq. (3). Therefore, if $\pi \in \arg\min_{\pi'} \mathbb{E}_{\rho \sim \mathcal{M}'_{\pi'},s'_0}[\text{cost}_T(\rho) \mid \Diamond T]$ then $\gamma \in \arg\min_{\gamma'} \mathcal{J}^{\text{cost}}_{C,\varphi}(\gamma')$. 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. [6]) over a composition MDP $\mathcal{M}'$. As explained above, 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[24] and that our composition MDP has a state space that is double-exponentially larger in the size of the goal specification, we get this result:

Theorem 4 Problem 1 can be solved in at most double-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.

This is in line with the classical setting of LTL/LTL$_f$ synthesis on probabilistic systems[39,40], and analogous to our solution method for the non-stochastic case[41].

5 Stochastic service composition for safety LTL$_f$ tasks

In this section, we consider the composition problem where the first objective is to maximize the probability of not violating a safety LTL$_f$ specification, and as second objective, we maximize the expected conditional cost. This means that the notion of successful execution must be revised in order to consider a safety goal rather than a reachability goal.

We say that some infinite execution $t \in \mathcal{T}_{\gamma,C}$ is *legal (for C and φ)*, denoted with $\text{legal}_{C,\varphi}(h)$, if for all finite prefix histories $h \in \text{prefixes}(t)$ we have that (1) $\text{actions}(h) \models \varphi$, i.e. the safety condition is not violated at any time on the trace, and (2) for all positions k and all services i, it holds that $\sigma_{i,k} \in F_i$, i.e. every service is in an accepting state. We use $\text{legal}_{C,\varphi}(h)$ also for finite executions. We denote with $\mathcal{H}^{\varphi,\text{safe}}_{\gamma,C}$ the set of finite and legal execution of orchestrator γ on service community C. Finally, we say that an orchestrator γ *realizes the* LTL$_f$ *specification* φ *with* C if, for all traces $t \in \mathcal{T}_{\gamma,C}$, t is legal. When C and φ are clear from the context, we omit the suffix in $\text{legal}_{C,\varphi}$ and only write legal.

Example 4 Continuing with example 1, we consider the following LTL$_f$ safety task: $\Box\neg pluck$, i.e. never *pluck* the ripe fruits and flowers. For this safety task, differently from Fig. 1, we consider states a_1, b_1, and c_2 to be 'accepting', meaning that they are states of their respective service in which the execution of the composition safety task can enter into. An orchestrator that maximizes the probability of satisfying this goal will only use $\mathcal{B}_1$ and $\mathcal{B}_2$, since $\mathcal{B}_3$ can only perform *pluck* from state c_0, whose execution would make the safety task to fail.

The satisfaction probability of the safety task φ under orchestrator γ and community C is given by:

$$\mathcal{P}^{\text{safe}}_{C,\varphi}(\gamma) = \mathbb{P}_{\gamma,C}\Big(\{t \in \mathcal{T}_{\gamma,C} \,\big|\, \text{legal}(t)\}\Big) \tag{18}$$

Intuitively, $\mathcal{P}^{\text{safe}}_{C,\varphi}(\gamma)$ is the probability measure (under $\mathbb{P}_{\gamma,C}$) of the set of legal executions generated by executing the orchestrator γ over the service community C.

To formally define the second objective, we first define the total cost of horizon m for an execution $t = (\sigma_{10}\ldots\sigma_{n0}),(a_1,o_1),\ldots$ as $\text{Cost}^{\text{C}}_m(t) = \sum_{i=0}^{m-1} C_{o_i}(\sigma_{o_i},a_{i+1})$. Then, for an orchestrator γ and a state $\sigma = (\sigma_1,\ldots,\sigma_n)$, the expected mean-cost is defined as:

$$\mathbb{E}_{\gamma,\sigma}[MC] = \limsup_{m\to\infty} \frac{1}{m} \mathbb{E}_{h\sim\gamma,\sigma}[\text{Cost}^{\text{C}}_m] \tag{19}$$

The optimal expected average cost starting from a state σ is defined over all orchestrators γ for C as $\inf_\gamma \mathbb{E}_{\gamma,\sigma}[MC]$. Moreover, we use $\mathbb{E}_{h\sim\mathbb{P}_{\gamma,C}}[\text{Cost}^{\text{C}}_m \mid \text{legal}(h)]$ to denote the expected conditional finite horizon cost, where the condition is on whether h is legal. Finally, we define our second objective, the expected conditional mean-cost, as:

$$\mathcal{J}^{\text{avg-cost}}_{C,\varphi}(\gamma) = \limsup_{m\to\infty} \frac{1}{m} \mathbb{E}_{h\sim\mathbb{P}_{\gamma,C}}\Big[\text{Cost}^{\text{C}}_m \mid \text{legal}(h)\Big] \tag{20}$$

Intuitively, $\mathcal{J}^{\text{avg-cost}}_{C,\varphi}(\gamma)$ takes into account the costs that the orchestrator incurs in executing its policy by picking the chosen services, but averaged across the entire length of the execution.

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

$$\gamma \in \Gamma_{\mathcal{P}^{\text{safe}}_{C,\varphi}} \text{ and } \mathcal{J}^{\text{avg-cost}}_{C,\varphi}(\gamma) = \inf_{\gamma' \in \Gamma_{\mathcal{P}^{\text{safe}}_{C,\varphi}}} \mathcal{J}^{\text{avg-cost}}_{C,\varphi}(\gamma') \tag{21}$$

Intuitively, we fix a lexicographic order on the objective functions $\mathcal{P}^{\text{safe}}_{C,\varphi}$ and $\mathcal{J}^{\text{avg-cost}}_{C,\varphi}$, meaning that we aim to minimize the expected utilization mean-cost to satisfy the specification, conditioned to the satisfaction of the specification, while guaranteeing the optimal probability of satisfying it.

Now we aim to find a connection between the satisfaction probability and the notion of realizability.

Let us consider the set of finite executions of γ with the community C that becomes illegal for the first time, denote with $\overline{\mathcal{H}^{\varphi}_{\gamma,C}}$. Formally:

$$\overline{\mathcal{H}^{\varphi}_{\gamma,C}} = \{h \in \mathcal{H}_{\gamma,C} \mid \neg\text{legal}(h) \wedge \forall h' \in (\text{prefixes}(h) \setminus \{h\}) : \text{legal}(h')\} \tag{22}$$

Lemma 7 Let t be an infinite execution of γ over C. t is legal iff $t \notin \left(\bigcup_{h \in \overline{\mathcal{H}^{\varphi}_{\gamma,C}}} \mathcal{T}_{\gamma,C}(h)\right)$

Proof ($\Rightarrow$) by contrapositive, assume $t \in (\bigcup_{h \in \overline{\mathcal{H}^{\varphi}_{\gamma,C}}} \mathcal{T}_{\gamma,C}(h))$. Then, there exist some $h \in \overline{\mathcal{H}^{\varphi}_{\gamma,C}}$ such that $t \in \mathcal{T}_{\gamma,C}(h)$. By definition of $\overline{\mathcal{H}^{\varphi}_{\gamma,C}}$, we have that $\neg\text{legal}(h)$. Since $h \in \text{prefixes}(t)$ by definition of $\mathcal{T}_{\gamma,C}(h)$, we have that, by definition of legal, and from the above arguments, it follows that $\neg\text{legal}(t)$, i.e., there exist a finite prefix of t that is not legal, and therefore its infinite extension t is also not legal.

($\Leftarrow$) by contrapositive, assume t is not legal. This means that there exists a prefix $h \in \text{prefixes}(t)$ such that $\neg\text{legal}(h)$. Let h' be the shortest of such prefixes, i.e., the finite execution that violates the safety conditions for the first time. By definition of $\overline{\mathcal{H}^{\varphi}_{\gamma,C}}$, it holds that $h \in \overline{\mathcal{H}^{\varphi}_{\gamma,C}}$. Finally, since $h' \in \text{prefixes}(t)$, we also have $t \in \mathcal{T}_{\gamma,C}(h')$. Therefore, the claim holds. □

Observe that, from Lemma 7, the following equality holds:

$$\{t \in \mathcal{T}_{\gamma,C} \,|\, \text{legal}(t)\} = \mathcal{T}_{\gamma,C} \setminus \left(\bigcup_{h \in \overline{\mathcal{H}^{\varphi}_{\gamma,C}}} \mathcal{T}_{\gamma,C}(h) \right) \tag{23}$$

And in turn, from Eq. (23), we have:

$$\begin{aligned}
\mathcal{P}^{\text{safe}}_{C,\varphi}(\gamma) &= \mathbb{P}_{\gamma,C}\left(\{t \in \mathcal{T}_{\gamma,C}) \,|\, \text{legal}(t)\}\right) \\
&= \mathbb{P}_{\gamma,C}\left(\mathcal{T}_{\gamma,C} \setminus \left(\bigcup_{h \in \overline{\mathcal{H}^{\varphi}_{\gamma,C}}} \mathcal{T}_{\gamma,C}(h) \right)\right) \\
&= \mathbb{P}_{\gamma,C}(\mathcal{T}_{\gamma,C}) - \mathbb{P}_{\gamma,C}\left(\mathcal{T}_{\gamma,C} \cap \left(\bigcup_{h \in \overline{\mathcal{H}^{\varphi}_{\gamma,C}}} \mathcal{T}_{\gamma,C}(h) \right)\right) \\
&= 1 - \mathbb{P}_{\gamma,C}\left(\bigcup_{h \in \overline{\mathcal{H}^{\varphi}_{\gamma,C}}} \mathcal{T}_{\gamma,C}(h) \right)
\end{aligned} \tag{24}$$

The intuitive meaning of the expression in Eq. (24) for $\mathcal{P}^{\text{safe}}_{C,\varphi}$ is that we can define the probability of satisfaction of the safety specification as the probability of avoiding finite executions h such that $\neg\text{legal}(h)$.

In case the specification is *exactly* realizable, the notion of optimal orchestrator according to Eq. (21) coincides with the notion of 'safe realizability', as shown in the following results.

Lemma 8 *O*rchestrator γ realizes the safety specification φ over C *iff* $\overline{\mathcal{H}^{\varphi}_{\gamma,C}} = \emptyset$.

Proof ($\Rightarrow$) Assume $\overline{\mathcal{H}^{\varphi}_{\gamma,C}} \neq \emptyset$. By assumption, there exist one finite execution h in the set $\overline{\mathcal{H}^{\varphi}_{\gamma,C}}$. By Lemma 8, this implies that all infinite executions t that are extensions of h, i.e., $t \in \mathcal{T}_{\gamma,C}(h)$, are not legal. But this means that there exists at least one execution of γ that does satisfy the safety specification φ; hence, γ does not realize the safety specification φ.

($\Leftarrow$) Assume that γ does not realize the safety specification φ. This means that there exist a trace $t \in \mathcal{T}_{\gamma,C}$ such that $\neg\text{legal}(t)$. By Lemma 7, this implies that $t \in \mathcal{T}_{\gamma,C}(h)$, for some $h \in \overline{\mathcal{H}^{\varphi}_{\gamma,C}}$. Therefore, $\overline{\mathcal{H}^{\varphi}_{\gamma,C}} \neq \emptyset$. □

Theorem 5 Let C be a community of stochastic services, and φ be a LTL$_f$ safety specification. The orchestrator γ realizes φ with community C iff $\mathcal{P}^{\text{safe}}_{C,\varphi}(\gamma) = 1$.

Proof ($\Rightarrow$) if γ realizes φ, then by Lemma 8, $\overline{\mathcal{H}^{\varphi}_{\gamma,C}} = \emptyset$, and therefore $\mathbb{P}_{\gamma,C}(\bigcup_{h \in \overline{\mathcal{H}^{\varphi}_{\gamma,C}}} \mathcal{T}_{\gamma,C}(h)) = 0$. Finally, by the definition of $\mathcal{P}^{\text{safe}}_{C,\varphi}$ (Eq. [18]) and by Eq. (24), we have that $\mathcal{P}^{\text{safe}}_{C,\varphi}(\gamma) = 1 - \mathbb{P}_{\gamma,C}(\bigcup_{h \in \overline{\mathcal{H}^{\varphi}_{\gamma,C}}} \mathcal{T}_{\gamma,C}(h)) = 1$

($\Leftarrow$) If $\mathcal{P}^{\text{safe}}_{C,\varphi}(\gamma) = 1$, then by definition of $\mathcal{P}^{\text{safe}}_{C,\varphi}$ (Eq. [18]) and by Eq. (24) we have that $\mathbb{P}_{\gamma,C}(\bigcup_{h \in \overline{\mathcal{H}^{\varphi}_{\gamma,C}}} \mathcal{T}_{\gamma,C}(h)) = 0$. Since the left-hand side is a probability, the equation holds only if the argument of $\mathbb{P}_{\gamma,C}$ is empty, which in turn implies that $\overline{\mathcal{H}^{\varphi}_{\gamma,C}} = \emptyset$. Finally, the claim follows by applying Lemma 8. □

Theorem 6 Assume LTL$_f$ safety specification φ is realizable. If an orchestrator γ satisfies Eq. (21), then it realizes the specification φ.

Proof Since by assumption φ is realizable, then there exists an orchestrator γ' that realizes it. By Theorem 5, we can deduce that the optimal value of $\mathcal{P}^{\text{safe}}_{C,\varphi}(\gamma')$ is 1. Moreover, by assumption and by Eq. (21), it follows that $\gamma \in \Gamma_{\mathcal{P}^{\text{safe}}_{C,\varphi}}$, i.e. $\mathcal{P}^{\text{safe}}_{C,\varphi}(\gamma) = 1$, by the arguments above. Finally, again by Theorem 5, we get that γ realizes φ.

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

Problem 2 (Stochastic Composition for LTL$_f$ Safety Tasks). Given the pair (C,φ), where φ is an LTL$_f$ safety specification over the set of actions A, and C is a community of n stochastic services $C = \{\mathcal{S}_1,\ldots,\mathcal{S}_n\}$, compute an orchestrator that is optimal according to Eq. (21).

As in the reachability case of Section 4, Theorems 5 and 6 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 the service's transition function (even non-deterministic, as in study by De Giacomo et al.[41]), and then check whether $\max_\gamma \mathcal{P}^{\text{safe}}_{C,\varphi}(\gamma) = 1$.

5.1 Solution technique

Similarly to the previous case, the solution technique for the safety variant of the composition problem relies on finding an optimal policy for a bi-objective lexicographic optimization on a specifically built MDP. In particular, also for this problem, we consider a variant of the framework introduced in study by Busatto-Gaston et al.[35]; while the second objective they considered was the maximization of the expected conditional mean payoff, here we consider the minimization of the expected conditional mean costs. In general, the technique has a significant overlap with the reachability variant. However, the main difference lies in how we build the automata-based construction: in the reachability case, we are interested in an orchestrator such that, for any (infinite) execution t, a finite prefix of t satisfies the task φ; in the safety case, we are interested in an orchestrator such that, for all prefixes, the safety task φ is satisfied. The definition of the composition MDP must take into account this different requirement.

Our technique breaks down into the following steps: (1) first, we translate the LTL$_f$ formula into the equivalent DFA $\mathcal{A}_\varphi$; then (2) we combine $\mathcal{A}_\varphi$ with the services, computing a product of $\mathcal{A}_\varphi$ with the stochastic services in C, obtaining a new MDP, $\mathcal{M}'$, that we call the 'composition MDP'; (3) we solve the resulting MDP, finding a policy π for $\mathcal{M}'$ that is optimal w.r.t. the (safety) bi-objective lexicographic

function, as in study by Busatto-Gaston et al.[35], and then (4) we derive an orchestrator γ from π that is optimal w.r.t. Eq. (4).

Step 1. This step is the same as the previous case: starting from the LTL$_f$ specification φ, we compute the DFA $\mathcal{A}_\varphi = (A, Q, q_0, F, \delta)$ that is equivalent to the specification φ[11].

Step 2. Then, we define the composition MDP $\mathcal{M}' = (S', A', P', s'_0)$ with the associated cost function C', as in step 2 for the reachability case. We are interested in computing optimal policies for $\mathcal{M}'$, where the optimality is defined as follows. Consider the set of *bad states*:

$$\text{Bad} = \{(q, \sigma_1, \ldots, \sigma_n) \mid (q, \sigma_1, \ldots, \sigma_n) \in S' \wedge (q \notin F \vee \exists i \in [1, n] : \sigma_i \notin F_i)\}$$

In other words, Bad is the set of states such that their Q-component is not an accepting state or the current state of some service is not accepting. We consider the safety bi-objective lexicographic optimization over $\mathcal{M}'$, similarly to what has been done in study by Busatto-Gaston et al.[35]. In particular, we first consider the probability of avoiding a set of bad states Bad from $s \in S'$, following a policy π over the MDP $\mathcal{M}'$, denoted with $\mathbb{P}_{\mathcal{M}'_\pi, s}(\Box\neg\text{Bad})$; with $\Pi_{\mathcal{M}', s}(\Box\neg\text{Bad})$, we denote the set of policies with the maximum probability of avoiding states in Bad from state s, i.e. $\arg\max_\pi \mathbb{P}_{\mathcal{M}'_\pi, s}(\Box\neg\text{Bad})$. Then, as the second objective, we consider the expected conditional mean-cost $\mathbb{E}_{\mathcal{M}'_\pi, s'_0}[MC \mid \Box\neg\text{Bad}]$. An optimal solution for $\mathcal{M}'$ is a policy π that minimizes the expected conditional mean-cost $\limsup_{m\to\infty} \frac{1}{n}\mathbb{E}_{\mathcal{M}'_\pi, s'_0}[\text{Cost}_m \mid \Box\neg\text{Bad}]$, while avoiding the set of bad states, but *only* among the policies in $\Pi_{\mathcal{M}', s'_0}(\Box\neg\text{Bad})$, i.e., the policies that maximize $\mathbb{P}_{\mathcal{M}'_\pi, s'_0}(\Box\neg\text{Bad})$:

$$\pi \in \Pi_{\mathcal{M}', s'_0}(\Box\neg\text{Bad}) \text{ and } \pi \in \arg_{\pi'} \inf \mathbb{E}_{\rho\sim\mathcal{M}'_{\pi'}, s_0}[MC \mid \Box\neg\text{Bad}] \qquad (25)$$

Step 3. The solution technique we will use is, once again, based on the work[35], where the authors propose a two-stage technique to find an optimal policy for a bi-objective lexicographic function in the form of Eq. (25). First, we compute the set of policies (in the form of a set of optimal actions for each state) that maximize the probability of staying far from the bad set of states Bad; however, this set of policies also contains policies that do not take into account the utilization costs of the services. Then, we consider a 'pruned MDP' in which (i) only optimal actions can be taken, and (ii) only 'good' states are kept. The new MDP is used to find policies that minimize the expected conditional mean-cost while maximizing the probability of staying in good states. Crucially, observe that, by construction, avoiding states in Bad is equivalent to not violating the safety specification φ, since bad states have their Q-component to be states of $\mathcal{A}_\varphi$. Moreover, by construction, the optimal policy of the pruned MDP guarantees that no better probability of staying safe can be achieved, since sub-optimal actions for the first objective are pruned. In this case, the difference between our scenario and Busatto-Gaston et al.[35] is that they consider reward maximization, rather than cost minimization, as the second objective function. Nevertheless, it is easy to see that we can minimize costs by maximizing negative rewards. Note that the two subproblems can be solved efficiently using standard planning algorithms, e.g., value iteration or linear programming.

Step 4. Once an optimal policy is found, we can obtain its equivalent γ as in step 4 of the reachability case.

Now we are going to establish a relationship between optimal orchestrators according to Eq. (21), and optimal policies for $\mathcal{M}'$ according to Eq. (25). Note that Lemma 3 and Lemma 4 also hold in this context since the composition MDP construction $\mathcal{M}'$ is the same as for the reachability case.

Let $\text{Paths}^\omega_{\neg\text{Bad}, \mathcal{M}'_\pi}(s'_0)$ be the set of infinite paths following π on $\mathcal{M}'$ such that they start with s'_0 and never enter in bad state in Bad, i.e. $\text{Paths}^\omega_{\neg\text{Bad}, \mathcal{M}'_\pi}(s'_0) = ((S' \setminus \text{Bad}) \times A)^\omega \cap \text{Paths}^\omega_{\mathcal{M}'_\pi}(s'_0)$. Let $\text{Paths}_{\text{Bad}, \mathcal{M}'_\pi}(s'_0)$ be the 'dual' set, i.e. the set of finite paths following π on $\mathcal{M}'$ such that they start with s'_0 and enter in bad state in Bad only at the end of the path and for the first time, i.e. $\text{Paths}_{\text{Bad}, \mathcal{M}'_\pi}(s'_0) = ((S' \setminus \text{Bad}) \times A)^* \text{Bad} \cap \text{Paths}_{\mathcal{M}'_\pi}(s'_0)$. Let also $\text{Paths}^\omega_{\text{Bad}, \mathcal{M}'_\pi}(s'_0) = \bigcup_{\rho' \in \text{Paths}_{\text{Bad}, \mathcal{M}'_\pi}(s'_0)} \text{Paths}^\omega_{\mathcal{M}'_\pi}(\rho')$, i.e. the set of infinite paths such that at least once they visited a bad state, and $\text{Paths}_{\neg\text{Bad}, \mathcal{M}'_\pi}(s'_0)$ the set of finite paths that never visit a Bad state. It is easy to see that the two sets $\text{Paths}^\omega_{\neg\text{Bad}, \mathcal{M}'_\pi}(s'_0)$ and $\text{Paths}^\omega_{\text{Bad}, \mathcal{M}'_\pi}(s'_0)$ are disjoint and complete. Intuitively, regarding disjointness, a finite path $\rho' \in \text{Paths}_{\text{Bad}, \mathcal{M}'_\pi}(s'_0)$ cannot be the prefix of any infinite path $\rho \in \text{Paths}^\omega_{\neg\text{Bad}, \mathcal{M}'_\pi}(s'_0)$, since $\text{last}(\rho') \in \text{Bad}$ and ρ does not visit any bad state. Regarding completeness, there are no other cases in between, i.e. a infinite path either belongs to the first set or to the second set:

Lemma 9 *The following two propositions hold:*

1. $\text{Paths}^\omega_{\neg\text{Bad}, \mathcal{M}'_\pi}(s'_0) \cap \text{Paths}^\omega_{\text{Bad}, \mathcal{M}'_\pi}(s'_0) = \emptyset$;

2. $\text{Paths}^\omega_{\neg\text{Bad}, \mathcal{M}'_\pi}(s'_0) \cup \text{Paths}^\omega_{\text{Bad}, \mathcal{M}'_\pi}(s'_0) = \text{Paths}^\omega_{\mathcal{M}'_\pi}(s'_0)$.

Another crucial observation is that $\text{Paths}_{\text{Bad}, \mathcal{M}'_\pi}(s'_0)$ is a set of finite paths forming pairwise disjoint cylinder sets. To see this, it is enough to observe that each of these paths has only the last state in Bad, while the previous states are all ¬Bad; this means that they cannot share a common prefix.

Lemma 10 *Let* $\rho = s_0 a_1 \ldots s_m \in \text{Paths}_{\mathcal{M}'_\pi}$ be a finite path on $\mathcal{M}'_\pi$ for some policy π, and let $h = \tilde{\tau}_{\varphi, C}(\rho)$ be its associated history. Then, $\text{legal}(h)$ iff for all $k \in [0, m]$ we have $s_k \notin \text{Bad}$.

Proof We prove that $\neg\text{legal}(h)$ iff $\exists k. s_k \in \text{Bad}$. By definition of Bad, if a state $s = (q, \sigma_1, \ldots, \sigma_n)$ is in Bad, it means that (a) $q \notin F$ or (b) there exists i such that $\sigma_i \notin F_i$. Let us consider a generic time step k and the associated state s_k in ρ. For s_k, Proposition (a) holds iff the sequence of Q-components of the states occurring in $\rho[0:k]$, i.e. $r = q_0, \ldots, q_k$, is a non-accepting run over $\mathcal{A}_\varphi$ for trace $a_1 \ldots a_k$. By definition of acceptance of a trace, the above holds iff $a_1 \ldots a_k \in \mathcal{L}(\mathcal{A}_\varphi)$. By the correctness of the construction of $\mathcal{A}_\varphi$, this is true iff $a_1 \ldots a_k \not\models \varphi$. By definition of $\tilde{\tau}$, we can rewrite the above expression as (c) $\text{actions}(h') \not\models \varphi$, where $h' = h[0:k]$. Finally, by definition of legal, we have that for some k (c) or (b) holds iff $\neg\text{legal}(h)$ □

Lemma 11 *Let* π *be a policy for* $\mathcal{M}'$ *and let* γ *be its equivalent orchestrator. Moreover, let* $\rho = s_0 a_1 s_1 \cdots s_m \in \text{Paths}_{\mathcal{M}'_\pi}$ *be a finite path on* $\mathcal{M}'_\pi$ *and* $h = \tilde{\tau}_{\varphi, C}(\rho)$. *Then,* $\rho \in \text{Paths}_{\text{Bad}, \mathcal{M}'_\pi}$ *iff* $h \in \overline{\mathcal{H}^\varphi_{\gamma, C}}$.

Proof First, note that by Lemma 3, $\rho \in \text{Paths}_{\mathcal{M}'_\pi}$ iff $h = \tilde{\tau}_{\varphi, C}(\rho)$ is an execution of γ.

($\Rightarrow$) By Lemma 10, since $\rho[m] \in \text{Bad}$, we have $\neg\text{legal}(h)$. Moreover, for all $j = 0, \ldots, m-1$, all states in $\rho[0:j]$ are not in Bad, and therefore again by Lemma 10, for each prefix $h[0:j] \in \text{prefixes}(h)$, $\text{legal}(h[0:j])$ holds. By definition of $\overline{\mathcal{H}^\varphi_{\gamma, C}}$, this implies that $h \in \overline{\mathcal{H}^\varphi_{\gamma, C}}$.

($\Leftarrow$) If $h \in \overline{\mathcal{H}^\varphi_{\gamma, C}}$, then by definition of $\overline{\mathcal{H}^\varphi_{\gamma, C}}$ we have that (1) $\neg\text{legal}(h)$ and that (2) for all $h' \in \text{prefixes}(h)$ such that $h' \neq h$, we have $\text{legal}(h')$. By Lemma 10, proposition (2) implies that for all $j = 0, \ldots, m-1$, all prefixes $\rho[0:j]$ are such that their states are not in Bad. Moreover, from (1), and again by Lemma 10, it follows that there exists $k \in [0, m]$ such that $\rho[k] \in \text{Bad}$, but from the previous argument, k must be equal to m since states at positions smaller than m in ρ are not in Bad. In other words, the last state in ρ is in Bad, while all the other states along ρ are not. Therefore, by definition of $\text{Paths}_{\text{Bad}, \mathcal{M}'_\pi}$, we have $\rho \in \text{Paths}_{\text{Bad}, \mathcal{M}'_\pi}$. □

Lemma 12 Let π be a policy for $\mathcal{M}'$ and let γ be its equivalent orchestrator. Moreover, let $\rho = s_0 a_1 s_1 \cdots s_m \in \text{Paths}_{\mathcal{M}'_\pi}$ be a finite path on $\mathcal{M}'_\pi$ and $h = \tilde{\tau}_{\varphi, C}(\rho)$. Then, $\rho \in \text{Paths}_{\neg\text{Bad}, \mathcal{M}'_\pi}$ if $h \in \mathcal{H}^{\varphi, \text{safe}}{}_{\gamma, C}$.

Proof By definition of $\mathcal{H}^{\varphi, \text{safe}}{}_{\gamma, C}$, $h \in \mathcal{H}^{\varphi, \text{safe}}{}_{\gamma, C}$ iff $\text{legal}(h)$. By Lemma 10, $\text{legal}(h)$ iff for all $k = 0 \ldots m$, $\rho[k] \notin \text{Bad}$. By definition of $\text{Paths}_{\neg\text{Bad}, \mathcal{M}'_\pi}$, this holds iff $\rho \in \text{Paths}_{\neg\text{Bad}, \mathcal{M}'_\pi}$. □

Theorem 7 Let (C,φ) be an instance of Problem 2, and let $\mathcal{M}'$ be the composition MDP for C and φ. We have that π is optimal (w.r.t. Eq. [25]) iff its equivalent orchestrator γ is optimal (w.r.t. Eq. [21]).

Proof First, we show that $\pi = \arg\max_{\pi'} \mathbb{P}_{\mathcal{M}'_{\pi},s'_0}(\Box\neg\text{Bad})$ iff $\gamma = \arg\max_{\gamma'} \mathcal{P}^{\text{safe}}_{C,\varphi}(\gamma')$. For any pair π and its equivalent γ, we have:

$$\mathbb{P}_{\mathcal{M}'_{\pi},s'_0}(\Box\neg\text{Bad}) = \mathbb{P}_{\mathcal{M}'_{\pi},s'_0}(\text{Paths}^{\omega}_{\neg\text{Bad},\mathcal{M}'_{\pi}}(s'_0)) \tag{26}$$

$$= 1 - \mathbb{P}_{\mathcal{M}'_{\pi},s'_0}(\text{Paths}^{\omega}_{\text{Bad},\mathcal{M}'_{\pi}}(s'_0)) \tag{27}$$

$$= 1 - \sum_{\rho'\in\text{Paths}_{\text{Bad},\mathcal{M}'_{\pi}}(s'_0)} \mathbb{P}_{\mathcal{M}'_{\pi},s'_0}(\text{Paths}^{\omega}_{\mathcal{M}'_{\pi}}(\rho')) \tag{28}$$

$$= 1 - \sum_{h\in\overline{\mathcal{H}^{\varphi}_{\gamma,C}}} \mathbb{P}_{\gamma,C}(\mathcal{T}_{\gamma,C}(h)) \tag{29}$$

$$= 1 - \mathbb{P}_{\gamma,C}\Big(\bigcup_{h\in\overline{\mathcal{H}^{\varphi}_{\gamma,C}}} \mathcal{T}_{\gamma,C}(h)\Big) \tag{30}$$

$$= \mathcal{P}^{\text{safe}}_{C,\varphi}(\gamma) \tag{31}$$

where step 26 is by definition of probabilistic safety, step 27 is by Lemma 9, step 28 is by disjointness of all finite paths $\rho' \in \text{Paths}_{\text{Bad},\mathcal{M}'_{\pi}}$ that end in a Bad state, step 29 is by Lemma 4 and 11, step 30 is by disjointness of all $\mathcal{T}_{\gamma,C}(h)$ for $h \in \overline{\mathcal{H}^{\varphi}_{\gamma,C}}$, and step 31 is by definition of $\mathcal{P}^{\text{safe}}_{C,\varphi}$ (Eq. [18]) and by Eq. (24). From this, we obtain that $\pi^* = \arg\max_{\pi'} \mathbb{P}_{\mathcal{M}'_{\pi},s'_0}(\Box\neg\text{Bad})$ iff $\gamma^* = \arg\max_{\gamma'} \mathcal{P}^{\text{safe}}_{C,\varphi}(\gamma')$.

It remains to prove that π is cost-optimal iff γ is cost-optimal. In the following, we use $\text{Paths}^{(m)}$ to denote paths over $\mathcal{M}'$ of length m. By definition of the expected conditional finite horizon cost $\mathbb{E}_{\mathcal{M}'_{\pi},s'_0}[\text{Cost}_m \mid \Box\neg\text{Bad}]$, using the law of total expectation, we get:

$$\mathbb{E}_{\mathcal{M}'_{\pi},s'_0}(\text{Cost}_m \mid \Box\neg\text{Bad}) = \sum_{\rho\in\text{Paths}^{(m)}_{\mathcal{M}'_{\pi}}} \mathbb{E}\Big[\text{Cost}_m \mid \text{Paths}^{\omega}_{\mathcal{M}'_{\pi}}(\rho), \Box\neg\text{Bad}\Big] \\ \mathbb{P}_{\mathcal{M}'_{\pi}}\Big(\text{Paths}^{\omega}_{\mathcal{M}'_{\pi}}(\rho) \mid \Box\neg\text{Bad}\Big)$$

Intuitively, we are averaging the total costs of all paths of length m that never visit Bad by the probability of those paths being visited under policy π in $\mathcal{M}$. Since Cost_m depends only on the first m steps, it is constant on each cylinder; hence, $\mathbb{E}[\text{Cost}_m \mid \text{Paths}^{\omega}_{\mathcal{M}'_{\pi}}(\rho), \Box\neg\text{Bad}] = \text{Cost}_m = \sum_{k=0}^{m-1} C'(s'_k, a_{k+1})$. Moreover, by applying the definition of conditional probability ($P(A \mid B) = \frac{P(A\cap B)}{P(B)}$), and by observing that paths that do not start with s'_0 have probability 0, we get:

$$\sum_{\rho\in\text{Paths}^{(m)}_{\mathcal{M}'_{\pi}}(s'_0)} \Big(\sum_{k=0}^{m-1} C'(s'_k, a_{k+1})\Big) \frac{\mathbb{P}_{\mathcal{M}'_{\pi}}\big(\text{Paths}^{\omega}_{\mathcal{M}'_{\pi}}(\rho) \cap \Box\neg\text{Bad}\big)}{\mathbb{P}_{\mathcal{M}'_{\pi}}(\Box\neg\text{Bad})}$$

Note that the denominator $\mathbb{P}_{\mathcal{M}'_{\pi},s'_0}(\Box\neg\text{Bad})$ is a normalization factor due to the condition $\Box\neg\text{Bad}$ in the conditional probability. Moreover, note that $\mathbb{P}_{\mathcal{M}'_{\pi},s'_0}(\Box\neg\text{Bad}) = \mathbb{P}(\text{Paths}^{\omega}_{\neg\text{Bad},\mathcal{M}'_{\pi}}(s'_0))$; hence, the above expression can be rewritten as:

$$\sum_{\rho\in\text{Paths}^{(m)}_{\mathcal{M}'_{\pi}}(s'_0)} \Big(\sum_{k=0}^{m-1} C'(s'_k, a_{k+1})\Big) \frac{\mathbb{P}_{\mathcal{M}'_{\pi},s'_0}\big(\text{Paths}^{\omega}_{\mathcal{M}'_{\pi}}(\rho) \cap \text{Paths}^{\omega}_{\neg\text{Bad},\mathcal{M}'_{\pi}}(s'_0)\big)}{\mathbb{P}_{\mathcal{M}'_{\pi},s'_0}(\text{Paths}^{\omega}_{\neg\text{Bad},\mathcal{M}'_{\pi}}(s'_0))}$$

Note that whenever $\rho \in (\text{Paths}^{(m)}_{\mathcal{M}'_{\pi}}(s'_0) \setminus \text{Paths}_{\neg\text{Bad},\mathcal{M}'_{\pi}})$, the corresponding term in the sum is zero. Therefore, we can rewrite the summation condition as:

$$\sum_{\rho\in\text{Paths}^{(m)}_{\neg\text{Bad},\mathcal{M}'_{\pi}(s'_0)}} \Big(\sum_{k=0}^{m-1} C'(s'_k, a_{k+1})\Big) \frac{\mathbb{P}_{\mathcal{M}'_{\pi},s'_0}(\text{Paths}^{\omega}_{\mathcal{M}'_{\pi}}(\rho))}{\mathbb{P}_{\mathcal{M}'_{\pi},s'_0}(\text{Paths}^{\omega}_{\neg\text{Bad},\mathcal{M}'_{\pi}}(s'_0))}$$

By Lemma 12, there is a bijection between paths in $\rho \in \text{Paths}^{(m)}_{\mathcal{M}'_{\pi}}(s'_0)$ and histories in $h \in \mathcal{H}^{\varphi,\text{safe}}{}_{\gamma,C}$ with $|h| = m$. By Lemma 4, the probability of the cylinder sets of ρ and h is the same. Moreover, by construction of $\mathcal{M}'$, for all $k = 0,\ldots,m-1$, the terms of the form $C'(s'_k, a_{k+1})$ in the sum of the costs can be rewritten as $C_{o_{k+1}}(\sigma_{o_{k+1},k}, a_{k+1})$, i.e., the cost of executing action a_{k+1} with the o_{k+1}-th service in its current state. Finally, by the first part of the proof (steps 26–31), we have that $\mathbb{P}_{\mathcal{M}'_{\pi},s'_0}(\Box\neg\text{Bad}) = \mathcal{P}^{\text{safe}}_{C,\varphi}(\gamma)$ and that the probability of a path with no bad states in $\mathcal{M}$ is the same as the probability of the corresponding trace t being legal in C. Therefore, we can rewrite the expression as follows:

$$\sum_{h\in\mathcal{H}^{\varphi,\text{safe}}{}_{\gamma,C}:|h|=m} \Big(\sum_{k=0}^{m-1} C_{o_{k+1}}(\sigma_{o_{k+1},k}, a_{k+1})\Big) \frac{\mathbb{P}_{\mathcal{M}'_{\pi},s'_0}(\mathcal{T}_{\gamma,C}(h))}{\mathcal{P}^{\text{safe}}_{C,\varphi}(\gamma)}$$

$$= \mathbb{E}_{h\sim\mathbb{P}_{\gamma,C}}\Big[\text{Cost}^{\text{C}}_m \mid \text{legal}(t)\Big]$$

From the above, it follows that

$$\mathcal{J}^{\text{avg-cost}}_{C,\varphi}(\gamma) = \limsup_{m\to\infty} \frac{1}{m} \mathbb{E}_{h\sim\mathbb{P}_{\gamma,C}}\Big[\text{Cost}^{\text{C}}_m \Big| \text{legal}(h)\Big] \\ = \limsup_{m\to\infty} \frac{1}{m} \mathbb{E}_{\rho\sim\mathcal{M}'_{\pi},s'_0}(MC \mid \Box\neg\text{Bad}),$$

since we are taking the limit of equal terms. Therefore, $\pi \in \arg\inf_{\pi'} \limsup \frac{1}{m}\mathbb{E}_{\rho\sim\mathcal{M}'_{\pi'},s'_0}(MC \mid \Box\neg\text{Bad})$ if $\gamma \in \arg\inf_{\gamma'} \mathcal{J}^{\text{avg-cost}}_{C,\varphi}(\gamma')$. Combining both results, we get the thesis. □

Computational cost. Theorem 7 guarantees that we can reduce Problem 2 to the problem of finding an optimal policy for the lexicographic bi-objective optimization problem (Eq. [6]) over the composition MDP $\mathcal{M}'$, where this time the first objective is to maximize the probability of avoiding a set of states in an MDP, and as a second objective, we minimize the expected conditional mean-cost.

As explained above, 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[24] 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 8: Problem 2 can be solved in at most double-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.

6 Case study

To demonstrate how the proposed approach can be instantiated and applied in a smart manufacturing scenario, we consider a representative production process: the assembly of an electric motor a widely used component in various applications such as industrial machinery, electric vehicles, household appliances, and many others[17]. 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: Fig. 2 depicts the DECLARE formalization[12] 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. Each DECLARE pattern can be transformed into an LTL$_f$ formula[42]. Therefore, the entire process can be encoded into an LTL$_f$ formula φ, which we omit. Then, we will consider the reachability task φ as the goal of the service composition.

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

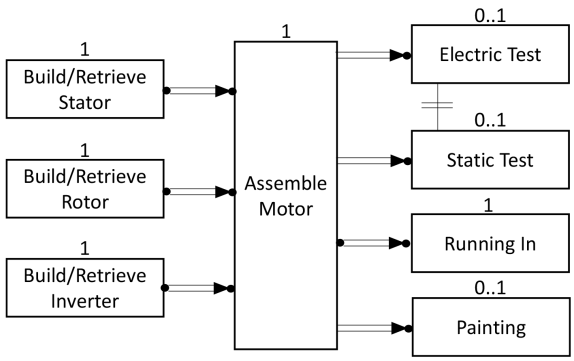

Fig. 2 The electric motor manufacturing process represented using DECLARE.

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 $c_i^{[Op]}$. 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)

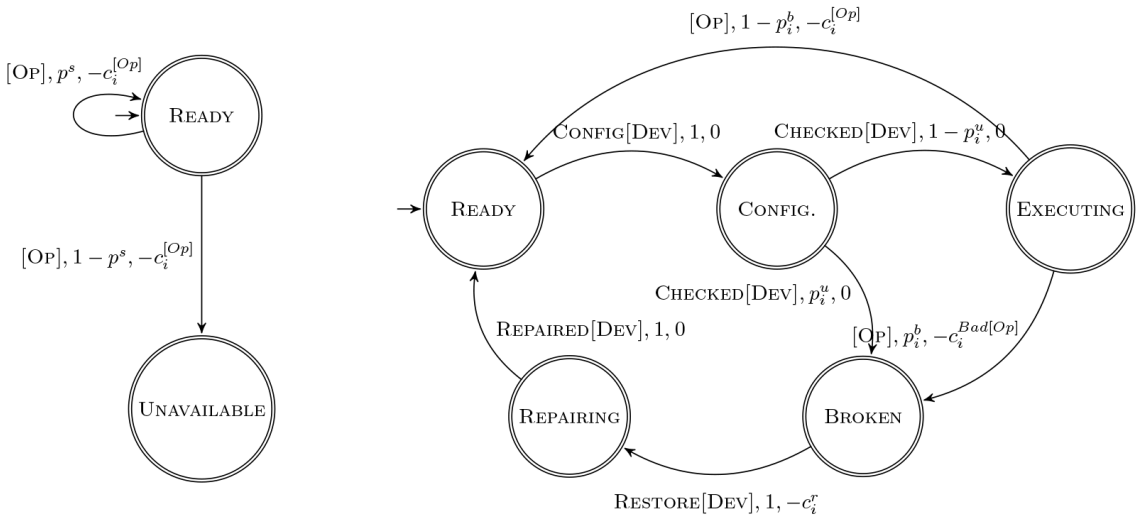

Fig. 3 The two types of service we consider for the electric motor case study. (a) 'Human operator' service. (b) 'Machine' service.

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.

7 Conclusions

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 two variants of the problem, one for reachability tasks and the other for safety tasks, 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 service reward/cost function, where we can consider more than one reward regarding service utilization. Moreover, we would like to implement our approach using state-of-the-art probabilistic model checkers such as PRISM[43] and Storm[44]. Another interesting direction for future work is incorporating internal-action hiding and parallel composition, which are standard in compositional modeling frameworks.

Author contributions

The authors confirm contributions to the paper as follows: study conception and design: De Giacomo G, Favorito M, Silo L; analysis and interpretation of results: De Giacomo G, Favorito M, Silo L; draft manuscript preparation: De Giacomo G, Favorito M, Silo L. All authors reviewed the results and approved the final version of the manuscript.

Data availability

Data sharing is not applicable to this article as no datasets were generated or analyzed during the current study.

Acknowledgments

This work is supported in part by the ERC Advanced Grant WhiteMech (Grant No. 834228), the PRIN project RIPER (Grant No. 20203FFYLK), the PNRR MUR project FAIR (Grant No. PE0000013), and the UKRI Erlangen AI Hub on Mathematical and Computational Foundations of AI (Grant No. EP/Y028872/1). 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.

Conflict of interest

The authors declare that they have no conflict of interest.

Dates

Received 6 January 2025; Revised 8 November 2025; Accepted 4 February 2026; Published online 30 April 2026

References

[1] Papazoglou MP, Traverso P, Dustdar S, Leymann F. 2007. Service-oriented computing: state of the art and research challenges. *Computer* 40(11):38–45

[2] Berardi D, Calvanese D, De Giacomo G, Lenzerini M, Mecella M. 2003. Automatic composition of *E*-services that export their behavior. In *Service-Oriented Computing - ICSOC 2003. Lecture Notes in Computer Science*. Vol. 2910. Berlin, Heidelberg: Springer. pp. 43–58 doi: 10.1007/978-3-540-24593-3_4

[3] Berardi D, Calvanese D, De Giacomo G, Mecella M. 2005. Composition of services with nondeterministic observable behavior. In *Service-Oriented Computing ICSOC2005. Lecture Notes in Computer Science*. Vol. 3826. Berlin, Heidelberg: Springer. pp. 520–526 doi: 10.1007/11596141_43

[4] De Giacomo G, Mecella M, Patrizi F. 2013. Automated service composition based on behaviors: the Roman model. In *Web Services Foundations*. New York, NY: Springer. pp. 189–214 doi: 10.1007/978-1-4614-7518-7_8

[5] Monti F, Silo L, Leotta F, Mecella M. 2023. On the suitability of AI for service-based adaptive supply chains in smart manufacturing. *2023 IEEE International Conference on Web Services (ICWS). July 2–8, 2023. Chicago, IL, USA*. USA: IEEE. pp. 704–706 doi: 10.1109/icws60048.2023.00091

[6] Monti F, Silo L, Leotta F, Mecella M. 2023. Services in smart manufacturing: comparing automated reasoning techniques for composition and orchestration. *Service-Oriented Computing. SummerSOC 2023. Communications in Computer and Information Science*. vol 1847. Cham: Springer. pp. 69–83 doi: 0.1007/978-3-031-45728-9_5

[7] De Giacomo G, Favorito M, Leotta F, Mecella M, Silo L. 2023. Digital twin composition in smart manufacturing via Markov decision processes. *Computers in Industry* 149:103916

[8] Yadav N, Sardina S. 2011. Decision theoretic behavior composition. *AAMAS '11: The 10th International Conference on Autonomous Agents and Multiagent Systems, 2−6 May, 2011, Taipei, Taiwan, China*. USA: International Foundation for Autonomous Agents and Multiagent Systems. pp. 575−582 doi: 10.65109/zgyz4787

[9] Brafman RI, De Giacomo G, Mecella M, Sardina S. 2017. Service composition in stochastic settings. *AI*IA 2017 Advances in Artificial Intelligence. Lecture Notes in Computer Science*. Cham: Springer. pp. 159–171 doi: 10.1007/978-3-319-70169-1_12

[10] Marrella A, Mecella M, Sardiña S. 2018. Supporting adaptiveness of cyber-physical processes through action-based formalisms. *AI Communications* 31(1):47−74

[11] De Giacomo G, Vardi MY. 2013. Linear temporal logic and linear dynamic logic on finite traces. *IJCAI '13: Proceedings of the Twenty-Third international joint conference on Artificial Intelligence, 3–9 August, 2013, Beijing, China*. USA: AAAI Press. pp. 854–860 https://dl.acm.org/doi/10.5555/2540128.2540252

[12] Debois S, Hildebrandt TT, Laursen PH, Ulrik KR. 2007. Declarative process mining for DCR graphs. *SAC '17: Proceedings of the Symposium on Applied Computing, 3−7 April, 2017, Marrakech Morocco*. USA: Association for Computing Machinery. pp. 759–764 doi: 10.1145/3019612.3019622

[13] Di Ciccio C, Montali M. 2022. Declarative process specifications: reasoning, discovery, monitoring. In *Process Mining Handbook. Lecture*

Notes in Business Information Processing. Vol. 448. Cham: Springer. pp. 108–152 doi: 10.1007/978-3-031-08848-3_4

[14] Dumas M, Fournier F, Limonad L, Marrella A, Montali M, et al. 2023. AI-augmented business process management systems: a research manifesto. *ACM Transactions on Management Information Systems* 14(1):1−19

[15] De Giacomo G, Favorito M, Leotta F, Mecella M, Silo L. 2021. Digital twins composition via Markov decision processes. *Proceedings of the 1st Italian Forum on Business Process Management, co-located with the 19th International Conference of Business Process Management (BPM 2021). Rome, Italy, 10th September, 2021*. Vol. 2952. CEUR Workshop Proceedings. pp. 44–49 https://ceur-ws.org/Vol-2952/paper_297a.pdf

[16] De Giacomo G, Favorito M, Leotta F, Mecella M, Silo L. 2022. Modeling resilient cyberphysical processes and their composition from digital twins via Markov Decision Processes. In *Proceedings of the Workshop on Process Management in the AI Era (PMAI 2022), co-located with 31st International Joint Conference on Artificial Intelligence and the 25th European Conference on Artificial Intelligence (IJCAI-ECAI 2022). Wien, Austria, 23 July, 2022*. Vol. 3310. CEUR Workshop Proceedings. pp. 101–104 https://ceur-ws.org/Vol-3310/paper7.pdf

[17] De Giacomo G, Favorito M, Leotta F, Mecella M, Monti F, et al. 2023. AIDA: a tool for resiliency in smart manufacturing. *Intelligent Information Systems. CAiSE 2023. Lecture Notes in Business Information Processing*. Vol. 477. Cham: Springer. pp. 112–120 doi: 10.1007/978-3-031-34674-3_14

[18] Chatterjee K, Majumdar R, Henzinger TA. 2006. Markov decision processes with multiple objectives. In: *STACS Lecture Notes in Computer Science*. Vol. 3884. Berlin, Heidelberg: Springer. pp. 325–336 doi: 10.1007/11672142_26

[19] Chatterjee K, Katoen JP, Weininger M, Winkler T. 2020. Stochastic games with lexicographic reachability-safety objectives. In *Computer Aided Verification. CAV 2020. Lecture Notes in Computer Science*. Vol. 12225. Cham: Springer, 2020, pp. 398–420 doi: 10.1007/978-3-030-53291-8_21

[20] Hahn EM, Perez M, Schewe S, Somenzi F, Trivedi A, et al. 2021. Model-free reinforcement learning for lexicographic omega-regular objectives. In *Formal Methods. FM 2021. Lecture Notes in Computer Science*. Vol. 13047. Lecture Notes in Computer Science. Cham: Springer. pp. 142–159 doi: 10.1007/978-3-030-90870-6_8

[21] Skalse J, Hammond L, Griffin C, Abate A. 2022. Lexicographic multi-objective reinforcement learning. *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI-ECAI 2022)*. pp. 3430–3436 www.ijcai.org/proceedings/2022/0476.pdf

[22] Wray KH, Zilberstein S, Mouaddib AI. 2015. Multi-objective MDPs with conditional lexicographic reward preferences. *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence. 25−30 January, 2015, Austin, Texas*. USA: AAAI Press. pp. 3418−3424 https://ojs.aaai.org/index.php/AAAI/article/view/9647

[23] Skalse J, Abate A. 2023. On the limitations of Markovian rewards to express multiobjective, risk-sensitive, and modal tasks. In *39th Conference on Uncertainty in Artificial Intelligence, 31st July–4th August, 2023, Pittsburgh, PA, USA*. Vol. 216. pp. 1974–1984 https://dl.acm.org/doi/10.5555/3625834.3626019

[24] Puterman ML. 1994. *Markov decision processes: discrete stochastic dynamic programming*. New York, USA: John Wiley & Sons. 672 pp. https://dl.acm.org/doi/10.5555/528623

[25] Gao A, Yang D, Tang S, Zhang M. 2005. Web service composition using Markov Decision Processes. In *WAIM*. Vol. 3739. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2005, pp. 308–319 doi: 10.1007/11563952_28

[26] Chen K, Xu J, Reiff-Marganiec S. 2009. Markov-HTN planning approach to enhance flexibility of automatic web service composition. *2009 IEEE International Conference on Web Services. 6–10 July, 2009, Los Angeles, CA, USA*. USA: IEEE. pp. 9–16 doi: 10.1109/ICWS.2009.43

[27] Moustafa A, Zhang M. 2013. Multi-objective service composition using reinforcement learning. In *Service-Oriented Computing. ICSOC 2013. Lecture Notes in Computer Science*. Vol. 8274. Berlin, Heidelberg: Springer. pp. 298–312 doi: 10.1007/978-3-642-45005-1_21

[28] YChen Y, Huang J, Lin C, Shen X. 2019. Multi-objective service composition with QoS dependencies. *IEEE Transactions on Cloud Computing* 7(2):537−552

[29] Sadeghiram S, Ma H, Chen G. 2020. A user-preference driven lexicographic approach for multi-objective distributed Web service composition. *2020 IEEE Symposium Series on Computational Intelligence (SSCI). 1−4 December, 2020, Canberra, ACT, Australia*. USA: IEEE. pp. 791−797 doi: 10.1109/ssci47803.2020.9308222

[30] De Giacomo G, Favorito M, Silo L. 2024. Composition of stochastic services for LTL_f goal specifications. In *Foundations of Information and Knowledge Systems. Lecture Notes in Computer Science*. Vol. 14589. Cham: Springer. pp. 298–316 doi: 10.1007/978-3-031-56940-1_17

[31] De Giacomo G, Vardi MY. 2015. Synthesis for LTL and LDL on finite traces. *IJCAI'15: Proceedings of the 24th International Conference on Artificial Intelligence, July 25–31, 2015, Buenos Aires, Argentina*. USA: AAAI Press. pp. 1558–1564 https://dl.acm.org/doi/10.5555/2832415.2832466

[32] De Giacomo G, Di Stasio A, Tabajara LM, Vardi MY, Zhu S. 2022. Finite-trace and generalized-reactivity specifications in temporal synthesis. *Formal Methods in System Design* 61(2):139−163

[33] Aminof B, De Giacomo G, Di Stasio A, Francon H, Rubin S, et al. 2025. LTL_f synthesis under environment specifications for reachability and safety properties. *Information and Computation* 303:105255

[34] Aminof B, De Giacomo G, Rubin S, Vardi MY. 2025. LTLf+ and PPLTL+: extending LTLf and PPLTL to infinite traces. In: *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence. 16–22 August 2025, Montreal, Canada, with satellite event in Guangzhou, China, 29–31 August 2025*. ijcai.org, 2025, pp. 8447–8455 www.ijcai.org/proceedings/2025/0939.pdf

[35] Busatto-Gaston D, Chakraborty D, Majumdar A, Mukherjee S, Pérez GA, et al. 2023. Bi-objective lexicographic optimization in markov decision processes with related objectives. In *Automated Technology for Verification and Analysis. ATVA 2023. Lecture Notes in Computer Science*. Vol. 14215. Cham: Springer. pp. 203–223 doi: 10.1007/978-3-031-45329-8_10

[36] Fionda V, Greco G. 2018. LTL on finite and process traces: complexity results and a practical reasoner. *Journal of Artificial Intelligence Research* 63:557−623

[37] Brafman RI, De Giacomo G, Patrizi F. 2018. LTLf/LDLf non-markovian rewards. *Artificial Intelligence* 331:1771−1778

[38] De Alfaro L. 1997. *Formal verification of probabilistic systems*. PhD thesis. Stanford University, USA. https://searchworks.stanford.edu/view/3910936.

[39] Courcoubetis C, Yannakakis M. 1995. The complexity of probabilistic verification. *Journal of the ACM* 42(4):857−907

[40] Wells AM, Lahijanian M, Kavraki LE, Vardi MY. 2020. LTLf synthesis on probabilistic systems. *Electronic Proceedings in Theoretical Computer Science* 326:166−181

[41] De Giacomo G, Favorito M, Silo L. 2025. Service composition for ltl task specifications. *Information Systems* 133:102571

[42] De Giacomo G, De Masellis R, Montali M. 2014. Reasoning on LTL on finite traces: insensitivity to infiniteness. *Proceedings of the AAAI Conference on Artificial Intelligence* 28:1027−1033

[43] Kwiatkowska M, Norman G, Parker D. 2011. PRISM 4.0: verification of probabilistic real-time systems. In *Computer Aided Verification. CAV 2011. Lecture Notes in Computer Science*. Berlin, Heidelberg: Springer. pp. 585−591 doi: 10.1007/978-3-642-22110-1_47

[44] Hensel C, Junges S, Katoen JP, Quatmann T, Volk M. 2022. The probabilistic model checker storm. *International Journal on Software Tools for Technology Transfer* 24(4):589−610