Towards Optimal Supervisory Controller Synthesis of Stochastic Nondeterministic Discrete-Event Systems

Jasen Markovski and Rong Su

Abstract—We synthesize supervisory controllers for discrete-time stochastic discrete-event systems with unrestricted nondeterminism by employing data-based control requirements and by abstracting from the stochastic aspects of the original system. This approach enables the use of standard synthesis tools, like Supremica, whereas the probabilistic behavior of the supervised system can be analyzed by using probabilistic model checking, e.g., by employing PRISM. We develop a process theory that provides for the appropriate abstractions of the probabilistic timed behavior of the original system and for compositional model transformation from Supremica to PRISM. We illustrate the proposed framework on a case study dealing with movement coordination of automated guided vehicles in pipeless plants.

I. INTRODUCTION

Supervisory control theory [22], [7] investigates automated supervisory control software synthesis based on discrete-event models of the uncontrolled system and the control requirements. Supervisory controllers ensure safe and non-blocking functioning of the system by observing and coordinating the discrete-event behavior of its distributed components. The models of the supervisory controllers are referred to as supervisors, which are synthesized based on a model of the uncontrolled system, known as a plant, and the control requirements. The synchronization of these processes results in the so-called supervised plant, modeling the supervised system.

The task of the supervisors is to ensure that system behavior is allowed by the control requirements and that no deadlock or livelock is present in the supervised plant [22], [7]. However, supervisors come with no guarantees with respect to liveness or performance properties of the plant, with the exception of fundamental nonblocking behavior. The above issues have been addressed in several extensions of the theory, which introduced quantitative aspects like probabilities [15], [9] and stochastic delays [12], [14] in the plant and/or the supervisor. These approaches treat controllability, i.e., conditions that ensure existence of a supervisor that respects the control requirements, and optimality at the same time.

With respect to controllability, the events that model system activities are split into two disjoint sets of controllable and uncontrollable events. The former model interactions with the actuators of the system and can be disabled, whereas the latter model observation of sensors or interaction with the user or the environment and must be preserved. With respect to optimality, the supervisor must ensure optimal behavior with respect to a given set of performance measures. Variants of this scheduling problem are investigated in the formal methods community using probabilistic extensions of temporal logics like [6].

Treating controllability and optimality at the same time, typically results in computationally expensive algorithms [6] and often requires (re)definition of standard performance metrics to cast them in the language-based supervisory control domain [9], [14]. Therefore, we propose to decouple the supervisor synthesis procedure from ensuring optimality to efficiently perform both tasks. By abstracting from the stochastic behavior, we are able to synthesize supervisors using standard synthesis tools. The synthesized discrete-event supervisor is then coupled with the original stochastic plant to ensure safety. Afterwards, we employ probabilistic model checking of discrete-time Markov decision processes to compute the bounds on performance and dependability of the stochastic nondeterministic discrete-event model of the supervised system.

The proposed synthesis-centric model-based systems engineering framework is supported by a process theory for synchronizing probabilistic timed processes with data. Unlike our previous work that investigated extensions with exponentially-distributed delays [18], [20], we opt for discrete-time stochastic processes due to more accessible tool support. Additionally, we do not treat the stochastic delays syntactically as in [20], but we abstract from them. This is an advantage, as we do not have to rely on the memoryless property of Markovian delays as in [18], [20], generalizing our approach. To capture the notion of controllability, we rely on an appropriate extension of the behavioral relation partial bisimulation [23], [4].

We illustrate our approach by revisiting a case study that deals with movement coordination of automated guided vehicles in a chemical pipeless plant. The materials needed to complete a product are transported in the system by means of automated guided vehicles, which safe movement coordination is postulated as a supervisory control problem. We synthesize a supervisor for the model of the pipeless plant using Supremica [1] and extend it with Markovian delays. Then, we analyze the stochastic supervised plant using the probabilistic model checker PRISM [13].

II. SYNCHRONIZING PROBABILISTIC TIMED PROCESSES

We present a process theory for synchronizing probabilistic timed processes with data that is suitable for model-
ing stochastic plants with unrestricted nondeterminism, and which restrictions can be employed to model supervisors with data-based observations and the underlying performance model.

We denote the data variables by \( V \) and their domain by \( D(X) \) for \( X \in V \). We require the variables to have finite domains, as this can be employed to deliver more efficient supervisor synthesis [25], which is supported by the synthesis tool [1]. By \( F(V) \) we denote the arithmetical expressions over a set of variables \( V \subseteq V \). To be able to evaluate the expressions, we employ a valuation function \( \delta \in \Delta(V) \), for \( \Delta(V) = \{ \delta | \delta : V \rightarrow D(V) \} \), that keeps tracks of the variable assignments. The expressions are evaluated by employing the evaluation function \( v_\delta : F(V) \rightarrow D(V) \) that depends on the current valuation \( \delta \). We define Boolean expressions over the set of variables \( V \) by \( B(V) \), where the atomic propositions are formed by predicate symbols over the data assignments, like \( \{ <, \leq, =, \neq, \geq, > \} \), the logical constants false \( F \) and true \( T \), and the set of standard logical operators. To evaluate the Boolean expressions we employ the evaluation function \( v_\delta : B(V) \rightarrow \{ F, T \} \) that also depends on \( \delta \in \Delta \). We note that whenever it is clear from the context, we omit the set of variables \( V \) and write \( \Delta \) for \( \Delta(V) \). In addition to the data variables, we employ recursion variables to define recursive processes and we denote the set of these variables by \( W \).

The process terms \( P \) are induced by \( P \) with:

\[
P ::= 0 \mid 1 \mid a[v].P \mid \bigoplus_{i \in I} \pi_i.P \mid \gamma : P \mid P + P \mid P \parallel_B P \mid \mu X.E
\]

(1)

for an action (or event) \( a \in A \), partial update function \( \nu : V \rightarrow F \), \( 0 < \pi_i \leq 1 \) for \( i \in I \) such that \( \sum_{i \in I} \pi_i = 1 \), \( \gamma \in \Delta \), and \( A, B \subseteq A \). The process term \( \mu X.E \) denotes the solution of the guarded recursive specification \( E \) with respect to the recursive variable \( X \in W \). Guarded recursive equations always prefix recursive variables with actions or timed delays and guarantee finite unique solutions [3], [2], [5]. The set of guarded recursive specifications is given by \( G \), whereas the set of recursive variables of the specification \( E \) is denoted by \( W(E) \).

A guarded recursive specification \( E \in G \) is defined as \( E = \{ X = g | X \in W(E), g \in GP \} \), where the set of guarded terms \( GP \) is induced by \( G \):

\[
G ::= 0 \mid 1 \mid a[v].U \mid a[v].G \mid \bigoplus_{i \in I} \pi_i.G \mid \bigoplus_{i \in I} \pi_i.U \mid \gamma : G \mid G + G,
\]

for \( U ::= X \mid U + U \) for \( X \in W \), \( a \in A \), \( \nu : V \rightarrow F \), \( 0 < \pi_i \leq 1 \) for \( i \in I \) such that \( \sum_{i \in I} \pi_i = 1 \), and \( \gamma \in \Delta \).

The process theory contains two constant processes. The deadlock process, denoted by 0, has no outgoing transitions, whereas successful termination option, denoted by 1, enables the sequential composition and models so-called marked states that denote that the system can successfully terminate its execution [22], [7]. The action-prefixed process, corresponding to \( a[v]P \), executes a delayable action transition labeled by \( a \in A \) and continues behaving as \( P \), and after the transition is taken, some variables are updated according to \( v \). The probabilistic timed delay prefix is introduced in combination with a probabilistic choice operator specified by \( \bigoplus_{i \in I} \pi_i.P_i \) for \( 0 < \pi_i \leq 1 \) for some finite index set \( I \), and it executes a unit timed delay with probability \( \pi_i \), after which the process continues behaving like \( P_i \). We employ this operation to form geometrically-distributed delays as in discrete-time Markov processes [8]. The variables can be employed to form process guards, resulting in the guarded command prefixed term \( \gamma : g \), where \( \gamma \) is allowed to begin its execution only when the guard \( \gamma \in \Delta \) is satisfied. The alternative composition, given by \( p + q \), makes a nondeterministic choice between action transitions or action transitions and probabilistic timed delays. Once an action transition or a probabilistic timed delay is chosen from \( p \) or \( q \), respectively, the alternative composition continues to behave as the remainder of the chosen process. We compose processes employing automata- or CSP-like synchronous parallel composition of [10], [22], [7], denoted by \( P \parallel_B Q \), which synchronizes actions of \( A \cap B \), and interleaves the actions of \( (A \setminus B) \cup (B \setminus A) \). The parallel composition enables synchronous advancement of time by synchronizing the probabilistic timed delays, if both processes can delay, or it interleaves the passage of time with the delayable synchronizing actions, when one component waits for synchronization.

We give semantics in terms of synchronizing probabilistic timed processes, which bear a resemblance to the Interactive Probabilistic Chains of [8]. Unlike the approach of [8], we do not consider internal actions, and we extend the theory with data variables. To keep track of the data variables and their updates, we couple the processes with an environment \( \sigma = (\delta, \alpha) \), where \( \delta \in \Delta \) keeps track of the valuations, whereas \( \alpha \in 2^V \) keeps track of which variables have been updated, as this is needed for correct semantics of the synchronous parallel composition. We write \( \sigma \in \Sigma \) for short, wherever the components of the environment are not needed.

The discrete stochastic processes are defined by a success-ful-termination option predicate \( \downarrow \subseteq P \times \Sigma \), an action transition relation \( \rightarrow \subseteq (P \times \Sigma) \times A \times (P \times \Sigma) \), and a probabilistic timed multi-transition \( \nrightarrow \subseteq (P \times \Sigma) \times (0, 1] \times (P \times \Sigma) \times N \). We write \( (p, \sigma) \downarrow \) for \( (p, \sigma) \in \downarrow \), \( (p, \sigma) \nrightarrow \) for \( (p, \sigma) \in \nrightarrow \), and \( n \)-times \( (p, \sigma) \nrightarrow (p', \sigma') \) for \( ((p, \sigma), (p', \sigma')) \in \nrightarrow \), and \( n \)-times \( (p, \sigma) \nrightarrow (p', \sigma') \) whenever there does not exist \( (p', \sigma') \in P \times \Sigma \) and \( a \in A \) such that \( (p, \sigma) \nrightarrow (p', \sigma') \), and \( (p, \sigma) \nrightarrow (p', \sigma') \) whenever there does not exist \( (p', \sigma') \in P \times \Sigma \) and \( a \in A \) such that \( (p, \sigma) \nrightarrow (p', \sigma') \).

We define \( \downarrow \), \( \rightarrow \), and \( \nrightarrow \) using structural operational semantics [3]. We define some auxiliary notation to present the operational rules more compactly. By \( f \circ f \) we denote the composition of the functions \( f \) and \( f \), where \( (g \circ f)(x) = g(f(x)) \) for \( x \in D(f) \). By \( f|_D \), we denote the restriction of the domain of function \( f \) to \( D \), i.e., \( f|_D = \{ x \mapsto f(x) \mid x \in D(f) \cap D \} \). By \( f|_g \) we define the replacement of the function \( f \) by the function \( g \) on their common domain, i.e., \( f|_g = f|_{D(f) \setminus D(g)} \cup g \). We depict the operational rules in Fig. 1, where for the sake of compactness of presentation we
omit the symmetrical operational rules and we only denote them in brackets next to the rule number.

To be able to relate our approach to language-based approaches to supervisory control theory, we also introduce the multi-step action labeled transition relation $\rightarrow\in (P \times \Sigma) \times A^* \times (P \times \Sigma)$, where $A^*$ is the set of strings or traces of $A$. By $\varepsilon$ we denote the empty trace and by $\ell_1, \ell_2 \in A^*$, we hold that $(p, \sigma) \ell_1 \varepsilon (p, \sigma)_1$, whereas for every non-empty trace $\ell a$, for $a \in A^*$ and $a \in A$, it holds that $(p, \sigma) \ell a \rightarrow (p', \sigma')$ if there exist $(p'', \sigma'') \in P \times \Sigma$ such that $(p, \sigma) \ell a \rightarrow (p'', \sigma'')$, $(p', \sigma') \in (p'', \sigma'')$.

The underlying behavioral relation that we employ is a probabilistic extension of partial bisimulation for discrete-event systems with data, which has been employed to capture controllability of nondeterministic plants in [4]. This relation is parameterized with a so-called bisimulation action set $B \subseteq A$. The actions inside this set are required to be bisimulated, whereas the rest of the actions are simulated. We need some preliminary notions to properly handle the probabilistic timed delays. Given a relation $R$, we write $R^{-1} \triangleq \{(q, p) \mid (p, q) \in R\}$ for the inverse relation. We require that $R$ is a reflexive and a transitive relation. Then, also $R^{-1}$ and $R \cap R^{-1}$ are reflexive and transitive. Moreover, $R \cap R^{-1}$ is symmetric, making $R \cap R^{-1}$ an equivalence relation. We employ this equivalence to ensure that the exiting accumulative probabilities of equivalent states to the same equivalence classes coincide as it is required by Markovian lumping [11], [8]. For $(p, \sigma) \in P \times \Sigma$ and $C \subseteq P$, we define $P((p, \delta), C) \triangleq \sum_{\pi \in (p, \sigma) \rightarrow (p', \sigma')} \pi$. Recall that $\rightarrow$ is a multi-transition relation, so the summation is performed over all multi-transitions between $p$ and $p'$. Moreover, note that the probabilistic timed delays are independent of the valuation of the data variables. Nonetheless, the operational rules require that we compute the accumulative rate within a given context.

Now, we directly apply the approach of [2] and require that the relation holds for every possible environment $\sigma \in \Sigma$. This makes the relation data-independent and equivalent processes do not depend on the initial valuation of the data variables. A reflexive and transitive relation $R \subseteq P \times P$ is a probabilistic partial bisimulation for the bisimulation action set $B \subseteq A$ if for all $(p, q) \in R$ and $\sigma \in \Sigma$, it holds:

1) $(p, \sigma) \not\rightarrow (q, \sigma)$;
2) if $(p, \sigma) \rightarrow (q', \sigma')$ for $a \in A$, then exist $q' \in P$ such that $(q, \sigma) \rightarrow (q', \sigma')$ and $(p', q') \in R$;
3) if $(q, \sigma) \rightarrow (q', \sigma')$ for $b \in B$, then exist $p' \in P$ such that $(p, \sigma) \rightarrow (p', \sigma')$ and $(p', q') \in R$;
4) $P((p, \sigma), C) = P((q, \sigma), C)$ for all $C \subseteq P/(R \cap R^{-1})$.

If $R$ is a partial bisimulation relation such that $(p, q) \in R$, then $p$ is partially bisimilar to $q$ with respect to $B$ and we write $p \equiv_B q$. It is not difficult to show that probabilistic partial bisimilarity is a preorder for the process terms in $P$ and it is a congruence for the given operators, by repeating the results of [4], [8]. This makes $\equiv_B$ a congruence, so we can define a standard term model for the process theory. Also note that if $B = A$, then mutual partial bisimulation $\equiv_A$ reduces to the probabilistic bisimulation of [8], when disregarding internal transitions. By abstracting from the probabilistic timed delays, we can also reduce the relation to strong bisimulation, when $B = A$, and strong simulation, when $B = \emptyset$ [4].
III. CONTROLLABILITY

To be able to model controllable and uncontrollable activities of the system, we split the set of action labels to the set of controllable C and uncontrollable U action labels, such that C ∪ U = A and C ∩ U = ∅. We model the system as a set of concurrent components, which synchronization amounts to a plant p, which is represented by some process p ∈ P, as the plant can have unrestricted behavior. On the other hand, the supervisor must amount to a deterministic non-probabilistic process as it models a supervisory controller that keeps track of the state of the plant by observing the data valuations and/or events and sends back unambiguous control signals. Similarly to the plant, we are not concerned with the form of the supervisor, i.e., it can be a monolithic process, or it can be distributed over several local supervisors, or even impose a hierarchical structure [7]. In any case, the possibly distributed supervisors synchronize to form the deterministic non-probabilistic process s ∈ P that models its behavior. We say that a process p is deterministic if for all σ ∈ Σ and ℓ ∈ A* it holds that |{(p′, σ′) ∈ P × Σ | (p, σ) → ℓ γ (p′, σ′)}| ≤ 1. We employ a strong variant of determinism that does not account for variable valuations since the supervisor comprises no variables.

We can summarize the form of the supervisor as µS.E_S, where the guarded recursive specification is given by:

\[ E_S = \{ S = \sum_{c \in C} \gamma_c : c.S + 1 \} \]  \hspace{1cm} (2)

The supervisor of form (2) implements the supervision actions by the guards γ_c ∈ B for c ∈ C, whereas it always enables the successful termination option. We note that the uncontrollable events are not part of the alphabet of the supervisor, so they are interleaved by default in every state. Also, it should be clear that the process specified by E_S is deterministic.

The supervised plant can be specified by p \parallel s, where s = µS.E_S, for some recursive specification of form (2). We employ probabilistic partial bisimilarity to ensure that no termination options of the plant are disabled by the supervisor, that all uncontrollable events of the reachable states are enabled by the supervisor, and that the branching structure and the probabilistic behavior of the original system is preserved in the allowed states. The role of the bisimulation action set is taken by the set of uncontrollable events, as they should not be disabled in any state of the supervised plant. We note that the successful termination option is enabled in all states of the supervised plant that comprise states of the original plant that can successfully terminate. This is a stronger requirement than the original setting of [22] and it follows from the fact that successful termination must be preserved in the congruence induced by the probabilistic partial bisimulation precongruence. Alternatively, we can introduce marked states independent of the process theory and require the same condition for coreachability as [22], but then we will lose the connection with the behavioral connection. The above conditions can be succinctly captured by the following refinement relation between the supervised

\[ (p, σ) \models \neg γ \Rightarrow \not\exists γ \]  \hspace{1cm} (23)

\[ (p, σ) \models \not\exists γ \Rightarrow γ \]  \hspace{1cm} (24)

\[ v_s(γ) = F \]  \hspace{1cm} (25)

\[ v_s(γ) = T, (p, (δ, α)) \not\models γ \Rightarrow α \not\models \not γ \]  \hspace{1cm} (26)

\[ v_s(γ) = T \]  \hspace{1cm} (27)

\[ (p, σ) \models γ \]  \hspace{1cm} (28)

\[ p \parallel s \subseteq u p. \]  \hspace{1cm} (3)

In absence of probabilistic behavior, equation (3) reduces to controllability of nondeterministic systems [4]. If, in addition, p is deterministic, then equation (3) reduces to standard language controllability of [23], [22], [7].

As discussed in the introduction, we opt to decouple the treatment of controllability from the treatment of optimality. Thus, the control requirements, that specify the allowed behavior of the system, disregard the probabilistic behavior and only consider the data valuations and the enabled labeled transition relations. The data-based control requirements, denoted by the set R, have the following syntax induced by R:

\[ R := \alpha \Rightarrow γ \mid γ \Rightarrow α \mid γ, \]  \hspace{1cm} (29)

for α ∈ A and γ ∈ B, and where \neg denotes logical negation. A given control requirement r ∈ R is satisfied with respect to process p ∈ P within the valuation σ ∈ Σ, notation (p, σ) \models r, according to the operational rules depicted in Fig. 2.

To be able to employ standard synthesis tools for supervisor synthesis, we have to abstract from the stochastic behavior, i.e., we have to abstract from the probabilistic timed delays. Recall that the control requirements only consider the states and the action labeled transitions, whereas the probabilistic timed delays do not affect the data valuations. This makes the abstraction from the probabilistic timed delays safe in the sense that the core discrete-event behavior of the abstracted system will not change. Intuitively, the probabilistic choice is replaced by a nondeterministic one, whereas the duration of time is abstracted from and the action labeled transitions remain delayable. We note that the synthesis tool supports nondeterministic plants [1].

To abstract from the stochastic behavior, we employ the functions abs for process terms in P, and the function abs_{X,E} for the guarded recursive specification in E ∈ G, where the leading variable is X ∈ W(E). Both functions are defined in Fig. 3, where by \sum_{i \in I} p_i, we denote the process term \sum_{i \in I} p_i for I = \{i_1, \ldots, i_n\}. We briefly comment on the rules that formalize the discussion from above. The abstraction passes through every operator, leaving everything intact, except for the probabilistic choice, which is abstracted to a nondeterministic choice. The transition from abs to abs_{X,E} is given when we abstract from the stochastic behavior in solutions of recursive specifications µX.E. Now, the abstracted supervised plant fulfills the controllability conditions for nondeterministic discrete-event
\[
\begin{align*}
\text{abs}(0) &= 0 & \text{abs}(a.p) &= a.\text{abs}(p) \\
\text{abs}(1) &= 1 & \text{abs}(\bigoplus_{i \in I} \pi_i.p_i) &= \sum_{i \in I} \text{abs}(p_i) \\
\text{abs}(p + q) &= \text{abs}(p) + \text{abs}(q) \\
\text{abs}(p \parallel q) &= \text{abs}(p) \parallel \text{abs}(q) \\
\text{abs}(\mu X.E) &= \mu X.\text{abs}(X.E(g_X)), \text{ if } X = g_X \in E \\
\text{abs}_{X,E}(0) &= 0 & \text{abs}_{X,E}(1) &= 1 \\
\text{abs}_{X,E}(a.p) &= a.\text{abs}_{X,E}(p) \\
\text{abs}_{X,E}(p + q) &= \text{abs}_{X,E}(p) + \text{abs}_{X,E}(q) \\
\text{abs}_{X,E}(Y) &= \text{abs}_{Y,E}(g_Y), \text{ if } Y = g_Y \in E \\
\text{abs}_{X,E}(\bigoplus_{i \in I} \pi_i.g_i) &= \sum_{i \in I} \text{abs}_{X,E}(h_i), \\
\end{align*}
\]

Fig. 3. Abstraction from probabilistic timed prefix in \( P \) and \( G \)

Fig. 4. Core of the proposed model-based systems engineering framework

systems from [4], if it holds that
\[
\text{abs}(p) \parallel s \leq_0 \text{abs}(p),
\]
i.e., the abstraction of the supervised plant is partially bisimulated by the abstraction from the supervised plant [4]. Note that \( \text{abs}(s) = s \), implying that \( \text{abs}(p) \parallel s = \text{abs}(p) \parallel s \). What remains to be shown is that every supervisor for the abstracted version of the plant is also a supervisor for the original stochastic plant, and vice versa.

**Theorem 1:** A deterministic process \( s \in S \) of form (2) is a supervisor for a plant \( p \in P \) if and only if it is a supervisor for \( \text{abs}(p) \), i.e., \( p \parallel s \leq_0 \text{abs}(p) \) if and only if \( \text{abs}(p) \parallel s \leq_0 \text{abs}(p) \).

Theorem 1 enables us to employ standard tools for supervisor synthesis in order to synthesize a supervisor for the original stochastic plant.

**IV. TOWARDS OPTIMAL SUPERVISION**

We depict the core of the proposed model-based systems engineering framework in Fig. 4. We extend previous proposals of [24], [21] with discrete-time stochastic features in the form of probabilistic timed delays. In Fig. 4, we presume that the modeling and abstraction processes are successfully completed, whereas for in-depth analysis of the process of formalization of the control requirements and the specification of the plant, we refer to [24], [21]. We begin with the abstracted plant that we model in Supremica [1] together with the data-based control requirements. The synthesized supervisor is employed to obtain the stochastic supervised plant that is compositionally transformed to PRISM [13]. To this end, we developed the model transformation tool Suprema2DTM [19]. This tool employs the Suprema specification of the plant and the synthesized supervisor and together with the probabilistic information forms a corresponding set of PRISM modules, which synchronization forms the final discrete-time Markov decision process. Recall that the CSP-like synchronization of PRISM [13] conforms with the synchronous parallel composition of Fig. 1.

We illustrate the proposed framework by coordinating the movement of automated guided vehicles in a pipeless chemical plants. The main feature of the pipeless plants is that instead of usage of pipes for transportation purposes, the needed materials are transferred by means of automated guided vehicles. As an illustration, we consider a prototype of a pipeless plant described in [16], [17] that produces and mixes colors. Vehicles are employed for carrying the materials, which are carried through the system by the vehicles to the stations in the system. The generator of vessels supplies the vessels, the filling stations fill them, the mixing stations mix the material, the exit station takes the vessels, whereas the vehicles can wait in a designated area for the next assignment. The colors are made by means of recipes, which specify the order in which colors should be poured and mixed. The vehicles have embedded control for safe moving, e.g., there is no spilling due to acceleration, docking, and collision avoidance while transiting in the pipeless plant [17].

The supervisory control problem that we consider is safe movement coordination of the vehicles such that a set of recipes is fulfilled. Vehicle movement is safe if no vehicle or vessel is damaged and no material is spilled outside the vessel. The pipeless plant can comprise multiple stations and automated guided vehicles. Vehicles have no memory, so the supervisor must keep track of their positions and status of the recipes. We can summarize the control requirements as follows. No empty vehicle is allowed at a filling station to prevent spilling of liquids. No vehicle that is carrying a vessel is allowed to visit a vessel generator station or a mixing station that is busy mixing another vessel to prevent vessels from breaking. No two vehicles are allowed to simultaneously transit or occupy the same station to prevent collision while docking. Finally, the system must successfully complete all recipes, leaving all stations empty, while all vessels have exited the system and the vehicles are parked in the waiting station.

We illustrate the process-algebraic specification of the plant, whereas the complete plant and the specification of the control requirements, as well as the transformation tools are available from [19]. The mixing station is modeled by the process \( \mu X.M \), where \( M \) is given by
\[
\begin{align*}
X &= gM[VM \rightarrow 1].Y, \\
Y &= \pi_1.Y \oplus \pi_2.mix.tM[VM \rightarrow 0].X,
\end{align*}
\]
and \( \oplus \) denotes the probabilistic choice operation, whereas \( gM \) models the reception of a vessel, \( mix \) denotes mixing, and \( tM \) denotes delivery. The probabilities \( \pi_1 \) and \( \pi_2 \) indicate the average time of mixing. In our example, the average time
is 10 time units, so $\pi_1 = \frac{9}{10}$ and $\pi_2 = \frac{1}{10}$. The variable $VM$ follows the state of the mixing station and it is 1 if the station is busy, and 0 if the station is empty.

We specify the control requirements by employing the observation variables. For example, the event $ar11$ denotes that the recipe 1 is assigned to vehicle 1. The vehicle can accept a task only if it not scheduled for another recipe, which is observed by the variable $AR1$, where $AR1 = 0$ denotes that the vehicle is available, and if the vehicle is empty, i.e., it is not busy delivering a vessel, which is indicated by the variable $AV1$, where $AV1 = 0$ denotes an empty vehicle. Thus, we compose the control requirement:

$$ar11 \Rightarrow AR1 = 0 \land AV1 = 0,$$

where $\land$ denotes logical conjunction.

Once the supervisor synthesis procedure is complete and the stochastic supervised plant is transformed to a Markov decision process, we can analyze this process using PRISM. The main features for analyzing these types of processes are the formulas $Pmin=?[F=\pi \cdot State]$ and $Pmax=?[F<\pi \cdot State]$ that compute the minimal and the maximal probability of reaching a set of states within a given time bound $t \geq 0$, where the time bound can be omitted to denote the situation where this set of states is eventually reached.

A pipeless plant comprising two vehicles, two filling stations, one mixing station, and two recipes has successfully complete its task if a state is reached that has the following labels: $A1F$ and $A2F$, which denote that the vehicles are empty, $RG2$, which denotes that both recipes have started, $R1SR1$ and $R2SR2$, which denotes that both recipes have finished, and $EDone$, which denotes that all vessels have exited the system. If we wish to compute the maximal probability with which we can finish both recipes within 200 times units, then we can employ the query:

$$Pmax=?[F<200("A1F") \land("A2F") \land("EDone") \land("R1SR1") \land("R2SR2") \land("RG2")],$$

which returns the probability of 0.76. This is the maximal performance that we can obtain from the supervised system for the experimental values we took.

V. CONCLUSION

We developed a model-based systems engineering framework that enables validation of the performance of discrete-time stochastic supervised systems. The framework is based on a process theory for probabilistic timed synchronizing processes with unrestricted nondeterminism. We extended the behavioral preorder partial bisimulation to the new setting and employed it to capture the notion of controllability. We also defined a suitable abstraction from the stochastic behavior, which enabled us to employ standard tools for the synthesis of the supervisor. We analyze the system using probabilistic model checking, for which we developed a compositional model transformation tool. We illustrated the system on a case study involving movement coordination of automated guided vehicles in a pipeless plant, for which we estimated the optimal performance.

REFERENCES