Invariant-Based Supervisory Control of Switched Discrete Event Systems

Invariant-based Supervisory Control of Switched Discrete Event Systems

Spyros Reveliotis and Zhennan Fei

***Abstract*—This paper introduces the notion of *switched Discrete Event Systems (s-DES)* and investigates its representational and computational potential in (i) the description and the analysis of the underlying DES behavior, (ii) the specification of the posed control requirements, and (iii) the eventual computation of the necessary control function. More specifically, it is shown that the potential decomposition of the overall DES behavior in a well defined set of "operational modes" enables the specification of control requirements and the synthesis of the corresponding control laws in a modular and distributed manner that takes full advantage of the aforementioned decomposition. The work is motivated by the need to cope with DES operating under a number of failing modes that result from non-catastrophic failures and repairs, and also DES that might evolve their operation through a number of "stages". Furthermore, the technical developments of the paper and their representational and computational power are further highlighted by an application example that is drawn from the area of robot pursuit on time-varying graphs; however, due to space considerations, this example is provided in an electronic supplement to the paper.**

Keywords: Discrete Event Systems, Switched Systems, Supervisory Control, Distributed Algorithms

I. Introduction

The notion of "switched (hybrid) systems" has been used extensively in the past control literature [1]. Under the existing theory, this notion describes time-driven dynamical systems where the governing equations change their structure abruptly at certain time-points, either due to the mere passage of time, or due to the transition of the considered system among particular sections of the underlying state space [2]. The distinct sets of equations that govern the system behavior during specific time intervals, or over specific sections of the underlying state space, define the set of the "operational modes" for this system, and the logic that determines the system transitions among its operational modes defines a "transitional structure" that is represented by some sort of (finite state) automaton [3]. The resultant formalization of this class of systems has given rise to some very interesting problems regarding the definition and the rigorous investigation of various notions of stability, reachability and other properties for the underlying dynamics, and the synthesis of control policies that will enforce these properties. The reader is referred to [1] for a nice tutorial introduction to the theory of switched / hybrid dynamical systems, and a comprehensive survey of the relevant literature.

S. Reveliotis is with the School of Industrial & Systems Engineering, Georgia Institute of Technology, email: spyros@isye.gatech.edu. Zhennan Fei is with Prover Technology, Stockholm, Sweden. S. Reveliotis was partially supported by NSF grant ECCS-1405156.

In this work, however, we are interested in switched systems where the modal dynamics are not "time-driven" but "event-driven" [3]. In other words, the system dynamics at each of its operational modes are represented by a "local" (finite state) automaton that characterizes the transition of this system through its underlying state space in response to various events that take place in its environment. Furthermore, these local dynamics are switched in an autonomous manner that is driven by an additional set of events that can take place uncontrollably at various system states. Hence, similar to the case of the hybrid dynamics that have been studied in the past, any trajectory characterizing the overall system behavior can be partitioned into a number of segments that are defined by the occurrence of two consecutive events that correspond to a modal switching of the underlying dynamics, and each segment evolves over the same state space but according to a different transitional structure. In the sequel, we shall refer to such a switched system as a *"switched Discrete Event System (s-DES)"*.

The astute reader will also notice that, according to the above description, an s-DES is essentially a (finite state) automaton with its state space partitioned / organized by the additional structure that is implied by the notion of the "(operational) modes" that was described in the previous paragraph, and with its dynamics further driven and shaped by the uncontrollable transitions that take place among these modes. Hence, the logical behavior of an s-DES can be analyzed and controlled through the corresponding DES theory [3]. In view of this fundamental observation, *the main objective of this paper is to show that the potential decomposition of the operation of any given DES into a set of modes along the aforementioned lines can enable the specification of control requirements and the synthesis of control laws for these systems in a modular and distributed manner that takes full advantage of the aforementioned decomposition*.

Hence, the results presented in this paper can be classified together with the existing results on modular and hierarchical DES supervisory control (SC) theory [3], [4] that seek to further regiment and facilitate the supervisor synthesis process by taking advantage of special structure that might exist in the specifications and the underlying system. Along these lines, our work presents particular affinity to the works presented in [5], [6], [7], [8] that also seek to design invariant-enforcing state feedback-based supervisors for the underlying DES. On the other hand, in the case of the s-DES structure that is considered in this work, the DES special structure involved is conceptually simpler and flatter than the hierarchical structures

mentioned above, since essentially we address the dynamics of the same physical entity under different operational regimes. Also, by seeking to enforce a local invariant-based specification for each mode, the sought supervisors decompose naturally across these modes. But since there are uncontrollable transitions among these modes, the synthesis procedures of these supervisors must interact with each other, and this interaction typically will be much more arbitrary and multidirectional than the corresponding interaction taking place in the synthesis procedures that are employed by hierarchical SC theory. We propose to support this interaction by means of a message-passing asynchronous distributed algorithm that possesses a separate thread for the synthesis of each local modal supervisor.

DES that are particularly amenable to the aforementioned decomposition of the underlying logical dynamics and the corresponding distribution of the control function, are those that are susceptible to non-catastrophic failures, i.e., failures that might alter the system behavior but do not lead to the termination of its operation. Machines on a manufacturing shop-floor (or any other sort of mechanical equipment) that have to experience some downtime due to a failing component or lack of certain consumables provide a typical example of the aforementioned type of failure. Furthermore, many DES modeling complex operations can be decomposed into a set of "operational regimes" or "stages" that correspond to particular phases towards the execution of a more major task; and these phases may be organized sequentially according to a prespecified execution plan, or they may evolve in a more random manner driven by exogenous input and events.

A nice exposition of the existing literature on fault-tolerant supervisory control for DES is provided by [9]. However, most of this literature employs a linguistic modeling framework and focuses on the case of a single but possibly unobservable failure. Furthermore, the main problem addressed in that regime, is to define diagnosability and controllability properties, as well as the necessary monitoring and control processes, that will enable the system to observe certain safety and maybe liveness requirements in its post-failure behavior. On the other hand, the s-DES model that is considered in this work allows the modeling of many different types of failures as well as the potential recovery from (some of) these failures. Furthermore, the occurrence of the various failing and recovery events is observable,[1] and the tracing of these events defines the operational mode of the considered DES at any given instant.

As already mentioned, in the resulting operational regime, the control specifications are expressed in terms of a set of predicates that are defined on the underlying state space; each predicate corresponds to a single failing mode, and it must be observed by the considered DES while finding itself in that mode. Also, the resulting SC problem can be addressed, in principle, by the standard DES SC theory [3], but the state space to be employed by the corresponding formulation involves the replication of the original DES state space for each considered failing mode, and therefore, its size can be exorbitant, even by the standards of the state spaces that are usually handled by DES theory. Hence, the emphasis of this work is placed on the development of the aforementioned distributed algorithm that can master this excessive complexity through a decomposing scheme that takes advantage of the special structure and the regularity that exists in the considered problem.

In view of the above positioning of the paper theme and the intended contribution, the presented developments are organized as follows: Section II provides a formal definition of the s-DES, the underlying dynamics, and the corresponding SC problem that is addressed in this work. Subsequently, Section III characterizes further the well-posedness of the considered SC problem and a notion of "optimal supervision" for it, by taking a more traditional, monolithic perspective on this problem and tapping upon the corresponding results of classical SC theory. Section IV presents the main result of this work, i.e., the aforementioned distributed algorithm for the computation of the optimal supervisor, where the distribution of the presented computation is taking place over the distinct operational modes of the considered s-DES. Finally, Section V concludes the paper developments, and outlines some directions for future work. In addition, an electronic supplement to this paper, accessible at http://www2.isye.gatech.edu/~spyros, provides (i) some supportive technical developments for Section III, and a demonstrative example that highlights the representational and computational advantages established by the s-DES concept and the proposed distributed algorithm.[2]

II. Switched discrete event systems and the considered SC problem

Switched discrete event systems: In the context of this work, a *switched Discrete Event System (s-DES)* $\mathcal{G}$ is formally defined by a finite state automaton (FSA) $\mathcal{G} \equiv \langle X \times S,\ \Sigma \cup E,\ \delta,\ (x_0, s_0)\rangle$, where:

1) The finite sets S and X define respectively the sets of operational *states* and *modes* of the considered DES $\mathcal{G}$. Hence, the complete – or *"global"* – state of DES $\mathcal{G}$ at any instant is characterized by the current operational state and the running mode.
2) The *event* set of $\mathcal{G}$ consists of two event types: The event set Σ triggers transitions of $\mathcal{G}$ within its operational state space S, without impacting its mode. On the other hand, the event set E collects the set of events that cause a modal change in $\mathcal{G}$. Furthermore, the set Σ is partitioned into the sets Σ^c and Σ^u defining, respectively, the sets of *controllable* and *uncontrollable* events in Σ. On the other hand, all the events in E are *uncontrollable* events.
3) The state transition function $\delta : X \times S \times (\Sigma \cup E) \to 2^{X \times S}$ describes the transition of $\mathcal{G}$ among its various

[1] since, for instance, it is not difficult to monitor the operational status of the machines and the other mechanical equipment that were mentioned in the previous paragraphs through the deployment of some sensing capability, or even by human supervision and inspection.

[2] We also notice that a preliminary and abbreviated version of this paper has appeared in the proceedings of the 5th Conference on the Analysis and Design of Hybrid Systems (ADHS 2015).

global states $(x,s) \in X \times S$ upon the occurrence of various events in $\Sigma \cup E$. The considered definition of δ supports the modeling of nondeterministic behavior, while by setting $\delta(x,s,q) = \emptyset$ for some triplets $(x,s,q) \in X \times S \times (\Sigma \cup E)$, we can also model potential *infeasibility* of some event q in a global state (x,s). Finally, the transition function δ is further qualified by the following condition:

$$\forall x \in X, \forall s \in S, \forall q \in \Sigma \cup E,\ \forall (x',s') \in \delta(x,s,q),\\ x' = x, \quad \text{if } q \in \Sigma \quad \wedge \quad x' \neq x, \quad \text{if } q \in E \tag{1}$$

Equation 1 expresses the aforestated fact that the events in Σ cannot impact the mode of $\mathcal{G}$, and establishes a notion of "locality" for the transitions that are effected by these events in the context of the global state space $X \times S$. On the other hand, the events in E cause transitions among the local subspaces that correspond to the various modes of DES $\mathcal{G}$.

4) Finally, (x_0, s_0) initializes $\mathcal{G}$ by specifying an initial mode $x_0 \in X$ and an initial state $s_0 \in S$.

In the following, we further assume that δ is extended to apply to (i) entire subsets of the global state space $G \subseteq X \times S$, and to (ii) strings of $(\Sigma \cup E)^*$, the Kleene closure of $\Sigma \cup E$, in the standard / natural manner. Also, whenever the state set G is a singleton $\{(x,s)\}$, we shall write $\delta(x,s,w)$ instead of the more complicated form $\delta(\{(x,s)\},w)$. Finally, we define:

$$Reach(\mathcal{G}) \equiv \{(x,s) \in X \times S : \exists w \in (\Sigma \cup E)^* \text{ s.t. } (x,s) \in \delta(x_0,s_0,w)\} \tag{2}$$

The considered SC problem: In this work, we want to control the considered DES $\mathcal{G}$ so that its dynamics observe a set of invariants $\mathcal{I}_x,\ x \in X$, where each $\mathcal{I}_x$ is defined by a predicate $\mathcal{P}_x$ that must be satisfied by the states $(x,s),\ s \in S$. Hence, $\forall x \in X,\ \mathcal{I}_x \equiv S_x = \{s \in S : \mathcal{P}_x(x,s) = \texttt{TRUE}\}$, and the SC problem addressed in this work can be formally characterized as follows:

Definition 1: **– The considered SC problem:** Given an s-DES $\mathcal{G} = \langle X \times S,\ \Sigma \cup E,\ \delta,\ (x_0,s_0)\rangle$ and a set of predicates $\mathcal{P}_x,\ x \in X$, represented by the state subsets $S_x = \{s \in S : \mathcal{P}_x(x,s) = \texttt{TRUE}\}$, develop a supervisor $\hat{\Gamma}$, in the form of a total function from $X \times S$ to 2^{Σ^c}, such that the transition function $\hat{\Gamma}/\delta$ that is obtained from the transition function δ by setting

$$\forall x \in X,\ \forall s \in S,\ \forall q \in \Sigma \cup E,\\ \hat{\Gamma}/\delta(x,s,q) = \begin{cases} \emptyset, & \text{if } q \in \Sigma^c \wedge q \notin \hat{\Gamma}(x,s) \\ \delta(x,s,q), & \text{o.w.} \end{cases} \tag{3}$$

induces the DES $\hat{\Gamma}/\mathcal{G} \equiv \langle X \times S,\ \Sigma \cup E,\ \hat{\Gamma}/\delta,\ (x_0,s_0)\rangle$ such that

$$Reach(\hat{\Gamma}/\mathcal{G}) \subseteq \{(x,s) : x \in X \wedge s \in S_x\} \tag{4}$$

In addition, we request that the synthesized supervisor $\hat{\Gamma}$ is maximally permissive; i.e., there must be no other supervisor Γ such that (i) the induced DES $\Gamma/\mathcal{G}$ satisfies the condition of Eq. 4, and (ii)

$$\forall (x,s) \in X \times S, \quad \hat{\Gamma}(x,s) \subseteq \Gamma(x,s) \tag{5}$$

with the inclusion holding true for at least one state (x,s).

Also, in order to facilitate the subsequent discussion, we shall set

$$\{(x,s) : x \in X \wedge s \in S_x\} \equiv \Omega \tag{6}$$

Finally, in the following, for any given supervisor Γ, we shall refer to the induced DES $\Gamma/\mathcal{G}$ as the *"controlled"* DES $\mathcal{G}$ (under supervision by Γ), and any supervisor Γ such that the controlled DES $\Gamma/\mathcal{G}$ satisfies the condition of Eq. 4 will be characterized as a *"correct"* supervisor.

III. Well-posedness and optimal supervision for the defined SC problem

As remarked in the introductory section, the SC problem formulated in Section II can be addressed, in principle, through the standard SC theory, especially, the seminal results that are presented in [5], [10] on predicate enforcement through state-feedback control. Hence, in this section, we overview some key results from that theory that characterize the notions of "well-posedness" and the "optimality" for the considered SC problem. However, instead of adopting the predicate calculus [11] that is adopted in the aforementioned works, in the following exposition we have opted for a more conventional line of semantics and notation, since this approach (i) might render the corresponding results more accessible to the general readership, and (ii) facilitates the statement of our main results through the standard concepts and terminology that are employed by the theory of distributed algorithms (c.f., for instance, [12]).[3]

Well-posedness of the considered SC problem: We begin our analysis of the SC problem of Definition 1 by providing the necessary and sufficient condition for the existence of a correct supervisor. For this, consider an s-DES $\mathcal{G} \equiv \langle X \times S,\ \Sigma \cup E,\ \delta,\ (x_0,s_0)\rangle$, and the supervisor $\Gamma^{\emptyset}$ defined by:

$$\forall (x,s) \in X \times S, \quad \Gamma^{\emptyset}(x,s) = \emptyset \tag{7}$$

Then, for any global state $(x,s) \in X \times S$, the *"uncontrollable reach"* $Reach^u(x,s;\mathcal{G})$ of this state is defined by

$$Reach^u(x,s;\mathcal{G}) \equiv Reach(\langle X \times S,\ \Sigma \cup E,\ \Gamma^{\emptyset}/\delta,\ (x,s)\rangle) \tag{8}$$

Also, we define the uncontrollable reach for the execution of a controllable event $\sigma \in \Sigma^c$ at some state $(x,s) \in X \times S$ by

$$Reach^u(x,s,\sigma;\mathcal{G}) = \bigcup_{(x,s') \in \delta(x,s,\sigma)} Reach^u(x,s';\mathcal{G}) \tag{9}$$

In view of the above definitions, the complete feasibility condition for the SC problem of Definition 1 can be expressed

[3] The electronic supplement to this paper, posted at http://www2.isye.gatech.edu/~spyros, contains a more expansive treatment of the material that is presented in this section, including complete formal proofs for the presented results.

as follows [5]:

Proposition 1: There exists a correct supervisor for the SC problem of Definition 1 if and only if (iff) $\forall(x,s) \in Reach^u(x_0,s_0;\mathcal{G}),\ s \in S_x$ (or, equivalently, $(x,s) \in \Omega$).

In the following, unless otherwise specified, we shall assume that all the addressed instantiations of the considered SC problem are feasible.

Pertinent correct supervisors and their disjunction: Next we proceed to define a notion of "supervisor disjunction" for the considered SC problem that will eventually enable us to formalize the notion of "maximal permissiveness" among the class of the considered supervisors. A first derivation of the following results can be traced in [5].

Definition 2: A supervisor Γ for an s-DES $\mathcal{G}$ with required control invariants $\mathcal{I}_x$, expressed by the state sets $S_x,\ x \in X$, is characterized as *"pertinent"* iff

$$\forall(x,s) \in \Omega,\ \forall\sigma \in \Gamma(x,s),\quad Reach^u(x,s,\sigma;\mathcal{G}) \subseteq \Omega \quad (10)$$

The next proposition establishes that it is reasonable to limit our search for a correct supervisor within the class of pertinent supervisors.

Proposition 2: Consider an s-DES $\mathcal{G}$ with required control invariants $\mathcal{I}_x$ that are expressed by the state sets $S_x,\ x \in X$. Then, for any correct supervisor Γ, there exists a pertinent correct supervisor Γ' such that (i) $Reach(\Gamma'/\mathcal{G}) = Reach(\Gamma/\mathcal{G})$ and (ii) $\forall(x,s) \in Reach(\Gamma/\mathcal{G}),\ \Gamma'(x,s) = \Gamma(x,s)$.

Definition 3: Given two correct supervisors Γ_1 and Γ_2, the *"disjunctive"* supervisor $\Gamma_1 \vee \Gamma_2$ is obtained by setting

$$\forall(x,s) \in X \times S,\quad (\Gamma_1 \vee \Gamma_2)(x,s) \equiv \Gamma_1(x,s) \cup \Gamma_2(x,s) \quad (11)$$

The disjunction of two correct supervisors for the considered SC problem is not necessarily a correct supervisor; an elucidating example of this fact is provided in the electronic supplement to this paper. However, this complication can be circumvented by restricting the underlying synthesis process to the space of *pertinent* correct supervisors.

Proposition 3: Consider an s-DES $\mathcal{G}$ with required control invariants $\mathcal{I}_x$ that are expressed by the state sets $S_x,\ x \in X$, and two pertinent correct supervisors $\Gamma_1,\ \Gamma_2$. Then, the disjunctive supervisor $\Gamma_1 \vee \Gamma_2$ is a pertinent correct supervisor for the considered problem.

Optimal supervision for the considered SC problem: The closure of the pertinent correct supervisors under disjunction, that was established by Proposition 3, further implies that, for any feasible instantiation of the SC problem of Definition 1, the *maximally permissive* pertinent correct supervisor is defined *uniquely* on the critical states $(x,s) \in X \times S$ with $Reach^u(x,s;\mathcal{G}) \subseteq \Omega$ (i.e., the global states that can observe the imposed invariants). In the sequel, we shall denote the maximally permissive supervisor w.r.t. these critical states by Γ^*, and we shall also refer to it as the *"optimal"* supervisor. For the remaining states $(x,s) \in X \times S$ that can violate uncontrollably the imposed invariants, the sought optimal supervisor can be left undefined, or, for better specificity, one can set $\Gamma^*(x,s) = \emptyset$, and this is the practice that we shall adopt herein.

A basic iterative algorithm for the computation of Γ^*: The computation of the optimal pertinent correct supervisor Γ^* for any feasible instantiation of the considered SC problem can be performed through a fixed-point iteration that is in line with the classical Ramadge & Wonham SC theory [13]. More specifically, we define the operator $\mathcal{F}$ upon the subsets of $X \times S$, that when applied on any given subset G of this set, returns

$$\mathcal{F}(G) = \{(x,s) \in G : \forall q \in \Sigma^u \cup E,\ \delta(x,s,q) \subseteq G\} \quad (12)$$

Also, we use the notation $\mathcal{F}^i$ to denote the i-fold composition of this operator with itself. Then, we have the following theorem:

Theorem 1: Consider an s-DES $\mathcal{G}$ with required control invariants $\mathcal{I}_x$ that are expressed by the state sets $S_x,\ x \in X$, and further assume that $Reach^u(x_0,s_0;\mathcal{G}) \subseteq \Omega$. Then, the limit $\lim_i \mathcal{F}^i(\Omega)$ is obtained in a finite number of iterations and it is a non-empty subset of the set Ω containing the initial global state (x_0,s_0). Furthermore, the sought supervisor Γ^* can be defined as follows:

$$\forall(x,s) \in X \times S,\quad \Gamma^*(x,s) = \begin{cases} \{\sigma \in \Sigma^c : \delta(x,s,\sigma) \subseteq \lim_i \mathcal{F}^i(\Omega)\},\ (x,s) \in \lim_i \mathcal{F}^i(\Omega) \\ \\ \emptyset,\ \text{o.w.} \end{cases}$$

The above computation of Γ^* is quite straightforward and, as already remarked, in line with similar existing results in DES SC theory. But for many DES $\mathcal{G}$ of the considered class, this computation may be practically challenged by the very large size of the underlying state space $X \times S$. Hence, it is useful to seek alternative algorithms for the computation of the target supervisor Γ^* that are amenable to a more distributed implementation. Such a distributed scheme can speed up the overall computation through parallelization, and, even more importantly, it can control more effectively the corresponding memory requirements. In the sequel, we present a distributed algorithm for the computation of Γ^* that takes advantage of the distribution of the control predicate that specifies the imposed invariant across the modes of the underlying s-DES $\mathcal{G}$.

IV. An efficient distributed algorithm for the computation of Γ^*

The presented algorithm distributes the computation of the target set $\lim_i \mathcal{F}^i(\Omega)$ over $|X|$ agents $\mathcal{A}_x,\ x \in X$, with each agent $\mathcal{A}_x$ pursuing the computation of the subset of $\lim_i \mathcal{F}^i(\Omega)$ that corresponds to mode x. Furthermore, these agents communicate by (i) message passing [12] that enable them to share partial results of their computation that will be useful in the computation conducted by some other agents, and also through (ii) some shared variables [12] that enable them to track the status of the overall computation, and define a terminating condition for their endeavor.

Next, we detail the proposed distributed algorithm by discussing (a) the data that is possessed by every agent $\mathcal{A}_x$, (b) the primary local variables that it employs during its computation, (c) the variables that it shares with the other agents, (d) the structure of the messages that it exchanges with the other agents, and (e) the primary algorithm that is executed by each agent. Once all these aspects of the considered algorithm have been fully described, we shall proceed to establish its correctness.

Each agent $\mathcal{A}_x,\ x \in X$, avails of the following data during the execution of its computation:

1) The control predicate $\mathcal{P}_x$ corresponding to mode x.
2) The restriction of the transition function δ to $\{x\}\times S\times\Sigma$, that encodes the transitions taking place in the subspace that corresponds to mode x. This restriction will be denoted by δ_x, and furthermore, for representational economy, δ_x will be considered as reduced to a two-argument function, defined on $S \times \Sigma$.[4]
3) The restriction of the function δ to $\{x\} \times S \times E$, that encodes the uncontrollable transitions in mode x resulting in a mode change. This restriction will be denoted by δ_x^E, and similar to the case of the function δ_x that was defined in the previous item, δ_x^E will be considered as a two-argument function, defined on $S \times E$.
4) The function $\Delta_x : S \to 2^X$ with $\Delta_x(s) \equiv \{x' \in X : \exists e \in E \text{ and } s' \in S \text{ s.t. } (x, s) \in \delta(x', s', e)\}$; in plain terms, $\Delta_x(s)$ provides the set of modes x' that have uncontrollable transitions from their subspace $\{(x', s') : s' \in S\}$ to state (x, s).
5) Finally, for the needs of the subsequent discussion, we also define $\hat{\Delta}_x \equiv \bigcup_{s\in S} \Delta_x(s)$.

The local variables that are possessed by agent $\mathcal{A}_x$ are as follows:

1) Q_x is a subset of S. This variable is initialized at the set $\{s \in S : P_x(s) = \texttt{TRUE}\}$, and at the end of the overall computation it will contain the target set $\{s \in S : (x, s) \in \lim_i \mathcal{F}^i(\Omega)\}$.
2) $M_{x,x'},\ \forall x' \in \hat{\Delta}_x$. At various stages of the computation of agent $\mathcal{A}_x$, each of these variables will contain subsets of S with the following property: Each state $s \in M_{x,x'}$ corresponds to a global state (x, s) that (i) has been identified by agent $\mathcal{A}_x$ as not belonging to the target set $\lim_i \mathcal{F}^i(\Omega)$, and (ii) can be reached from the corresponding mode x' through some uncontrollable event $e \in E$. Hence, $M_{x,x'}$ will be passed, as a message, to the corresponding agent $\mathcal{A}'_x$, that will proceed to the necessary state eliminations from its own set $Q_{x'}$.
3) $MessageQueue_x$ is a FIFO queue that collects all the messages $M_{x',x}$ that are sent to agent $\mathcal{A}_x$ by the other agents $\mathcal{A}_{x'}$, $x' \in X \setminus \{x\}$.

Besides their local variables, the agents $\mathcal{A}_x,\ x \in X$, also share the following two arrays of variables:

1) $COUNT[|X|, |X|]$, with $COUNT[x, x']$ reporting the number of messages sent by agent $\mathcal{A}_x$ to the agent $A_{x'}$ that have not been picked up for processing by agent $A_{x'}$.
2) $DONE[|X|]$, an array of Boolean variables, with $DONE[x]$ being set to TRUE or to FALSE by agent $\mathcal{A}_x$ depending on whether it is in an "idling" or a "working" mode.

The above two sets of variables are respectively initialized to 0 and FALSE, and as remarked at the beginning of this section, they will be used in order to detect the completion of the overall computation. On the other hand, the communication taking place among the various agents by means of the messages $M_{x,x'}$ that were discussed above, is assumed to occur through reliable FIFO channels that deliver the dispatched messages with a finite latency [12]. The reception of these messages by the destination agent and their placement in the corresponding message queue can be handled either by a thread that runs in parallel to the main computational thread of this agent, or by "interrupts" to the main computational process that are triggered by the arriving messages.

Next we turn to the description of the main algorithm that is run by each agent $\mathcal{A}_x,\ x \in X$. The pseudo-code for this algorithm is presented in Fig. 1, and it involves four major stages. The first stage concerns the initialization of the agent's local variables that control the corresponding algorithmic thread(s), and also, the initialization of the shared variables that are (primarily) controlled by this agent. This stage also collects in the set Q_x all the operational states $s \in S$ that satisfy the local predicate $\mathcal{P}_x$. Finally, the initialization stage concludes with agent $\mathcal{A}_x$ informing the other agents $\mathcal{A}_{x'}$, $x' \in \hat{\Delta}_x$, of the states $s \in S \setminus Q_x$ that violate the predicate $\mathcal{P}_x$ and can be reached uncontrollably from the subspaces of the global state space $X \times S$ that correspond to agents $\mathcal{A}_{x'}$; this is performed by composing and sending the corresponding messages $M_{x,x'}$, and updating the counters $COUNT[x, x']$ accordingly.

Upon its completion, the initialization stage passes control to a second stage that is perceived as the main computational stage of each local thread. The primary task of this stage is to update the variable Q_x according to a fixed-point computation that starts with the current value of Q_x and applies iteratively upon this variable an operator $\mathcal{F}_x$ that can be perceived as a localized version of the original operator $\mathcal{F}$ of Eq. 12 in the computational scope of agent $\mathcal{A}_x$; more specifically, the operator $\mathcal{F}_x$ is defined as follows:

$$\forall Q \subseteq S,\ \ \mathcal{F}_x(Q) \equiv \{s \in Q : \forall \sigma \in \Sigma^u,\ \delta_x(s, \sigma) \subseteq Q\} \quad (13)$$

A second task of this stage is to compose the messages $M_{x,x'},\ x' \in \hat{\Delta}_x$, with the corresponding states $s \in S$ that have been removed from Q_x during the aforementioned computation, and send these messages to their destination agents while updating accordingly the corresponding variables $COUNT[x, x']$.

Upon the completion of its two primary tasks, the second

[4]Formally, the reduction of δ_x from a three-argument to a two-argument function, can be obtained through the "existential quantification" of its first (constant) argument.

Input: $\mathcal{P}_x,\ \delta_x,\ \delta_x^E,\ \Delta_x,\ \hat{\Delta}_x$
Output: Q_x
/* INITIALIZE */
$MessageQueue_x := NILL; \quad DONE[x] :=$ `FALSE`;
for all $x' \in \hat{\Delta}_x$ **do**
  $COUNT[x, x'] := 0$;
end for
$Q_x := \{s \in S : \mathcal{P}_x(s) =$ `TRUE`$\}$;
if $Q_x \neq S$ **then**
  for all $x' \in \hat{\Delta}_x$ **do**
    $M_{x,x'} := \{s \in S \setminus Q_x : x' \in \Delta_x(s)\}$;
    if $M_{x,x'} \neq \emptyset$ **then**
      $COUNT[x, x'] + +$; send$(M_{x,x'})$;
    end if
  end for
end if
/* COMPUTE */
$\tilde{Q}_x := Q_x$;
repeat
  $\hat{Q}_x := Q_x$;
  $Q_x := \mathcal{F}_x(\hat{Q}_x) \equiv$
  $\{s \in \hat{Q}_x : \forall \sigma \in \Sigma^u, \delta_x(s, \sigma) \subseteq \hat{Q}_x\}$;
until $Q_x = \hat{Q}_x$;
if $Q_x \neq \tilde{Q}_x$ **then**
  for all $x' \in \hat{\Delta}_x$ **do**
    $M_{x,x'} := \{s \in S : s \in \tilde{Q}_x \setminus Q_x \wedge x' \in \Delta_x(s)\}$;
    if $M_{x,x'} \neq \emptyset$ **then**
      $COUNT[x, x'] + +$; send$(M_{x,x'})$;
    end if
  end for
end if
/* PROCESS MESSAGE QUEUE */
repeat
  if $MessageQueue_x \neq NILL$ **then**
    $DONE[x] :=$ `FALSE`; $\tilde{Q}_x := Q_x$;
    while $MessageQueue_x \neq NILL$ **do**
      Pop next message $M_{x'x}$ from $MessageQueue_x$;
      $COUNT[x', x] - -$;
      $Q_x := Q_x \setminus \{s \in S\ :\ \exists s' \in M_{x',x},\ \exists e \in E,$
      $(x', s')\ \in\ \delta_x^E(s, e)\}$;
    end while
    if $Q_x \neq \tilde{Q}_x$ **then**
      Go to "COMPUTE";
    else
      $DONE[x] :=$ `TRUE`;
    end if
  end if
until $\big(\sum_{x,x' \in X} COUNT[x, x'] = 0\big)\ \wedge$
$\bigwedge_{x \in X} DONE[x]$;
/* TERMINATE */
return Q_x;

Fig. 1. The algorithm executed by each agent $\mathcal{A}_x,\ x \in X$.

stage passes control to a third stage which is in charge of processing any newly received messages. More specifically, this third stage processes each message $M_{x',x}$ that is currently available in $MessageQueue_x$, by removing its contents from the set Q_x and updating accordingly the corresponding counter $COUNT[x', x]$. If these updates have resulted in a reduction of the set Q_x, this third stage passes the control back to the second stage for the computation of the new fixed point of $\mathcal{F}_x$ w.r.t. the current value of Q_x. If, on the other hand, Q_x was not altered by the processing of the new messages, then, the algorithm sets the variable $DONE[x] =$ `TRUE` and proceeds to check its terminating condition. This condition is uniform for all agents $\mathcal{A}_x,\ x \in X$, and requests that (i) there are no messages pending for processing, and (ii) all agents are idling; more formally,

$$\Big(\sum_{x,x' \in X} COUNT[x, x'] = 0\Big) \wedge \bigwedge_{x \in X} DONE[x] \tag{14}$$

If the condition of Eq. 14 is met, then the algorithm proceeds to its fourth and final stage, that returns the current set Q_x, terminates all its running threads, and exits. On the other hand, if the aforementioned condition is not met, the algorithm gets into a loop that continuously checks for the reception of new messages or the eventual satisfaction of the terminating condition. In particular, if the considered loop is broken by the arrival of new messages, the algorithm sets the variable $DONE[x] =$ `FALSE` and goes back to the beginning of the third stage, for the processing of these new messages according to the logic that was outlined in the previous paragraphs.

We close the presentation of the proposed distributed algorithm of Fig. 1 with the next theorem which establishes that (i) the considered algorithm will terminate in finite time, and that (ii) the sets Q_x returned by the agents $\mathcal{A}_x,\ x \in X$, upon their termination, constitute a correct distributed representation of the target set $\lim_i \mathcal{F}^i(\Omega)$ of Theorem 1.

Theorem 2: The algorithm depicted in Fig. 1 will terminate in finite time for all agents $\mathcal{A}_x,\ x \in X$, and the sets Q_x that are returned upon termination, will satisfy the following property:

$$\forall x \in X, \quad Q_x = \{s \in S : (x, s) \in \lim_i \mathcal{F}^i(\Omega)\} \tag{15}$$

Proof: To prove the results of Theorem 2, first it is important to notice that, due to its symmetrical structure, the terminating condition of Eq. 14 is priced uniformly for every agent $\mathcal{A}_x$, and whenever it is set to `TRUE`, all agents are in an idling mode, having set their corresponding variable $DONE[x] =$ `TRUE`. Furthermore, it is easy to check, by tracing the logic of the stage "PROCESS MESSAGE QUEUE", that once the terminating condition is set to true, it cannot be negated by the action of any agent, and therefore, all the agents $\mathcal{A}_x$ will have to proceed from their current idling status to their termination.

On the other hand, an agent $\mathcal{A}_x$ may exit its idling status by receiving some messages $M_{x',x}$ in its queue $MessageQueue_x$. But the origination and delivery of any such message $M_{x',x}$ by agent $\mathcal{A}'_x$ implies the reduction of

the cardinality of the corresponding state set $Q_{x'}$. Since (a) all state sets $Q_x,\ x \in X$, are initialized to finite contents, (b) these contents can only be reduced during the execution of the algorithm, and (c) any single pass of the stages "COMPUTE" and "PROCESS MESSAGE QUEUE" (i.e., lines 14–39 in Fig. 1) is a finite computation, it follows that the entire algorithm will terminate in finite time.

Next, we prove the correctness of the algorithm, i.e., the validity of Eq. 15. An argument very similar to that establishing the main result of Theorem 1 can establish that, for any value of the variable set Q_x, the fixed-point computation of the stage "COMPUTE" in the code of Fig. 1 will return the subset of Q_x collecting all of its states that have no uncontrollable paths to $S \setminus Q_x$. This remark together with the initial value of the sets $Q_x,\ x \in X$, further imply that the first execution of the stage "COMPUTE" by each agent $\mathcal{A}_x$ will remove from the corresponding set Q_x all states that have uncontrollable paths to states $s \in S \setminus S_x$ (i.e., to states in S that do not satisfy the predicate $\mathcal{P}_x$). On the other hand, the messages $M_{x,x'}$ exchanged by the agents $\mathcal{A}_x$, and the corresponding processing of these messages that takes place in stages "PROCESS MESSAGE QUEUE" and "COMPUTE", seek to eliminate from each state set Q_x any states s corresponding to global states (x, s) with emanating transition sequences to some predicate-violating state (x', s') that contain some event(s) $e \in E$. Hence, to establish the claim of Theorem 2, we need to show that the considered distributed algorithm generates and processes all the messages $M_{x,x'}$ that are necessary for the aforementioned eliminations.

To prove this last result, for any given global state $(x, s) \notin \lim_i \mathcal{F}^i(\Omega)$, let $\#(x, s)$ denote the minimum number of events $e \in E$ in any uncontrollable transition sequence that leads from (x, s) to $X \times S \setminus \Omega$; we shall establish the sought result through an induction on this state index. The base case of $\#(x, s) = 0$ was already established in the previous discussion about the computation that takes place during the first visit of the stage "COMPUTE" by each agent $\mathcal{A}_x,\ x \in X$. Next, suppose that the algorithm identifies and eliminates from the corresponding set Q_x all the global states $(x, s) \notin \lim_i \mathcal{F}^i(\Omega)$ with $\#(x, s) \leq n$, and consider a state $(\hat{x}, \hat{s}) \notin \lim_i \mathcal{F}^i(\Omega)$ with $\#(\hat{x}, \hat{s}) = n + 1$. By the definition of the state $(\hat{x}, \hat{s})$, there is a state (x', s') with $\#(x', s') = n$, and (x', s') is reachable from $(\hat{x}, \hat{s})$ through a transition sequence we where $w \in \Sigma^u$ and $e \in E$. By the inductive hypothesis, state s' has been eliminated from Q'_x by agent $\mathcal{A}_{x'}$, and furthermore, this elimination has been communicated to the agent $\mathcal{A}_{\hat{x}}$ by a message $M_{x',\hat{x}}$ during the execution of the corresponding computational stage by agent $\mathcal{A}_{x'}$. Since agent $\mathcal{A}_{x'}$ was in stage "INITIALIZE" or "COMPUTE" during the composition and dispatching of the aforementioned message $M_{x',\hat{x}}$, $DONE[x'] = \texttt{FALSE}$ and, according to the remarks in the opening paragraph of this proof, agent $\mathcal{A}_{\hat{x}}$ cannot be in its terminating stage. Furthermore, the increase of the counter $COUNT[x', \hat{x}]$ by one unit upon the creation of the aforementioned message $M_{x',\hat{x}}$ implies that the terminating condition of Eq. 14 cannot be evaluated to TRUE until agent $\mathcal{A}_{\hat{x}}$ has processed the message $M_{x,\hat{x}}$. Let $\tilde{s}$ denote the next-to-last state visited by the aforementioned transition sequence we on a path that leads from $(\hat{x}, \hat{s})$ to (x', s').[5] State $\tilde{s}$ is in $Q_{\hat{x}}$ during the processing of the considered message $M_{x',\hat{x}}$, since, otherwise, the considered state $\hat{s}$ would have been eliminated from $Q_{\hat{x}}$ by the fixed-point iteration of stage "COMPUTE" during the elimination of the state $\tilde{s}$. But then, the processing of $M_{x',\hat{x}}$ by agent $\mathcal{A}_{\hat{x}}$, in stage "PROCESS MESSAGE QUEUE", will lead to the elimination of state $\tilde{s}$ from $Q_{\hat{x}}$, and this elimination subsequently will trigger the execution of the stage "COMPUTE" by the same agent; state $\hat{s}$ will be eliminated from the set $Q_{\hat{x}}$ during this stage (unless $w = \epsilon$, in which case, $\hat{s} \equiv \tilde{s}$). □

An example application of the s-DES modeling framework and of the algorithm of Fig. 1 for the computation of the optimal strategy in a robot-pursuit problem that takes place on a time-varying graph, is provided in the electronic supplement to this paper. The corresponding developments reveal quite vividly, both, the representational and the computational efficiencies that are established by this new modeling framework, but they could not be included in this document due to the imposed space limitations.

V. Conclusions

This paper introduced the s-DES concept, defined an invariant-based SC problem for this class of DES, and developed a distributed algorithm for computing the corresponding maximally permissive supervisor. The proposed algorithm constitutes a decomposing scheme that is suggested by the s-DES structure, and it can effect substantial gains in terms of the required computational time and memory compared to the corresponding requirements that are posed by the monolithic algorithms provided by the classical DES SC theory. The motivational ideas underlying the presented work and the aforementioned representational and computational gains are further highlighted through the application of the derived results on a robot-pursuit problem that evolves on a time-varying graph, and it is provided in an electronic supplement to this paper.

Our future work will seek to further develop, formalize and assess the potential of the presented results for the particular application of "robust" deadlock avoidance, that was introduced in [14] and is further discussed in [15]. Such a development involves the establishment of (some notion of) non-blocking behavior for each operational mode, and necessitates the extension of the current theory and the algorithm of Fig. 1 to cases where the invariant-defining predicates $\mathcal{P}_x$, $x \in X$, cannot be specified independently at each mode x by simple subsets of the local state space S. We shall also consider additional specific applications that are amenable to the structure and the analytical and computational benefits of the s-DES concept. Finally, on the more theoretical side, an interesting direction is the integration of the concepts and the computational developments presented in this work with

[5] The last qualification of $\tilde{s}$ is necessary since the transitions of the considered DES $\mathcal{G}$ are nondeterministic.

the assumptions and the results on fault-tolerant SC that are overviewed in [9].

REFERENCES

[1] H. Lin and P. Antsaklis. Hybrid dynamical systems: An introduction to control and verification. *Foundations and Trends in Systems and Control*, 1:1–172, 2014.

[2] D. Liberzon. *Switching in Systems and Control*. Birkhäuser, Boston, MA, 2003.

[3] C. G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems (2nd ed.)*. Springer, NY,NY, 2008.

[4] W. M. Wonham. Supervisory control of discrete event systems. Technical Report ECE 1636F / 1637S 2013-14, Electrical & Computer Eng., University of Toronto, 2014.

[5] P. J. G. Ramadge and W. M. Wonham. Modular feedback logic for discrete event systems. *SIAM Journal on Control and Optimization*, 25:1202–1218, 1987.

[6] Y. Brave and M. Heymann. Control of discrete event systems modeled as hierarchical state machines. *IEEE Trans. on Automatic Control*, 38:1803–1819, 1993.

[7] B. Gaudin and H. Marchand. Safety control of hierarchical synchronous discrete event systems: a state-based approach. In *Proceedings of the 13th Mediterranean Conference on Control and Automation*, pages 889–895, 2005.

[8] C. Ma. *Non blocking supervisory control of state tree structures*. PhD thesis, University of Toronto, Toronto, Canada, 2004.

[9] T. Moor. Fault-tolerant supervisory control. In *Proceedings of the 5th Intl. Workshop on Dependable Control of Discrete Systems*, pages 17–24. IFAC, 2015.

[10] R. Kumar, V. Garg, and S. I. Marcus. Predicates and predicate transformers for supervisory control of discrete event systems. *IEEE Trans. on Automatic Control*, 38:232–247, 1993.

[11] E. W. Dijkstra. *A Discipline of Programming*. Prentice Hall, Englewood Cliffs, NJ, 1976.

[12] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, San Francisco, CA, 1996.

[13] P. J. G. Ramadge and W. M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77:81–98, 1989.

[14] M. Lawley and W. Sulistyono. Robust supervisory control policies for manufacturing systems with unreliable resources. *IEEE Trans. on R&A*, 18:346–359, 2002.

[15] S. Reveliotis. *Real-time Management of Resource Allocation Systems: A Discrete Event Systems Approach*. Springer, NY, NY, 2005.