Decentralized Optimal Control of Distributed Interdependent Automata With Priority Structure

Olaf Stursberg, *Member, IEEE*, and Christian Hillmann

***Abstract*— For distributed discrete-event systems (DESs), which are specified by a set of coupled automata, the centralized synthesis for a composed plant model is often undesired due to a high computational effort and the need to subsequently split the result into local controllers. This paper proposes modeling and synthesis procedures to obtain optimal decentralized controllers in state-feedback form for distributed DES. In particular, this paper addresses the DES with priority structures, in which subsystems with high priorities are supplied with the output of subsystems with lower priority. If the subsystem dependencies have linear or treelike structures, the synthesis of the subsystem controllers can be accomplished separately. Any local controller is computed by algebraic computations, it communicates with controllers of adjacent subsystems, and it aims at transferring the corresponding subsystem into goal states with a minimal sum of transfer costs. As is shown for an example, the computational effort can be significantly reduced compared with the synthesis of centralized controllers following the composition of all subsystem models.**

***Note to Practitioners*—In industrial practice, controllers to realize sequential procedures for processes such as multistep production schemes are very often designed manually, i.e., the designer selects a sequence of control actions based on an intuition of what has to be triggered next in order to get to a goal state. Such procedure (leading to controllers implemented often on a PLC) is of combinatorial nature and thus typically time-consuming and error-prone. To avoid several iterations over testing and correcting the controllers, the algorithmic and model-based synthesis is proposed as a reasonable alternative in this paper: a distributed discrete process is first modeled by modular discrete-event models, which explicitly account for the dependencies among the process units. While it is often difficult for a given process to manually select one out of many feasible control strategies, the algorithms for control synthesis proposed in this paper employ optimization to determine the behavior, which is most favorable with respect to costs associated with the model.**

***Index Terms*— Automata, decentralized control, discrete-event systems (DESs), optimization, production automation, supervisory control.**

I. Introduction

THE common engineering principle of *divide-and-conquer* is standard in automation of larger processes. For example, the manual design and implementation of supervisory or sequential controllers are typically accomplished separately for the logical units (subsystems or modules) of the overall process. Modeling the process dynamics by discrete-event systems (DESs) is also typically tractable only if subsystems are modeled separately, and then coupled by appropriate signals or variables [1]. The algorithmic and model-based computation of DES controllers often follows a scheme of first computing a composition of the subsystem models, then generating a controller for the monolithic model, and finally partitioning the controller according to hardware infrastructure used for implementation [2]. For the reasons of computational efficiency (and also to render the step of partitioning superfluous), the distributed computation of local subsystem controllers is a reasonable alternative, as discussed already in [3]. This paper follows the idea of decentralized control synthesis and combines it to the principles of optimization DES with particular structures.

The structures under consideration are motivated by the dependencies among subsystems, which can be observed, e.g., in industrial production processes with unidirectional supply schemes. Consider the example of an assembly process consisting of two machines, where the first machine assembles parts which are produced by a second one. When designing a discrete-event controller to establish the assembly procedure for the first machine, the second one must deliver the required parts at appropriate instances of time. The behavior and control objectives of the second machine must thus be aligned to the control strategy for the first. This also means that the controller of the first machine is entitled to define (and communicate) sequences of goal states to the controller of the second machine. With respect to the overall control objectives, the first machine can be identified as the one for which the control goal has to be satified with high priority, while the second machine can be classified as subordinated. One can easily imagine priority schemes of similar type, which involve more than two production units.

With respect to existing and relevant work on control synthesis for the DES, the approach of *supervisory control theory* (SCT) according to [4] is well established. In a language-based setting, algorithms based on the SCT generate controllers to formulate the set of behaviors, which is permissible according to a given specification. A large number of extensions of the SCT exist, including approaches to decentralized control [5]–[7], hierarchical structures [8]–[10] including consistency [11], communication aspects [12], [13], concurrence [14], modularity [15], as well as the transformation into PLC programs [16]. The work in [17] addresses the issue of reducing the computational complexity of the synthesis task by defining tailored and relatively small modular

Manuscript received December 21, 2015; revised June 1, 2016 and October 18, 2016; accepted January 26, 2017. Date of publication March 6, 2017; date of current version April 5, 2017. This paper was recommended for publication by Editor J. Wen upon evaluation of the reviewers' comments. This work was supported by the European Commission through the Project UnCoVer-CPS under Grant 643921.

The authors are with the Control and System Theory Group, Department of Electrical and Computer Science, Universität Kassel, Kassel 34121, Germany (e-mail: stursberg@uni-kassel.de).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASE.2017.2669893

abstractions. All of these approaches do not include the notion of minimizing transition cost (and thus selecting one out of several permitted behaviors), in opposite to the objective of this paper.

The approaches reported in [18] and [19] (together with preceding papers by the same authors) establish a theory of optimal control for the DES within the framework of the SCT. This is extended to time-varying DES and the notion of near-optimal solutions computed online in [20]. The work in [21] considers, in particular, so-called disabling costs and proposes an approach to optimal supervisory control for this setting. However, in contrast to what is reported here, the aforementioned papers do not consider plant modularity and decentralized control, i.e., they aim at optimizing permissible languages for a centralized controller instance. Sadid *et al.* [22] formulate a multiobjective optimization problem for decentralized DES, in order to solve the problem of collision avoidance of a set of autonomous vehicles. Opposed to the work presented in the paper at hand, [22] uses evolutionary algorithms to approximate the optimal solution, and it does not consider dependency structures among subsystems. The latter point also holds true for the work in [23], which addresses the synthesis of optimal supervisors for cyclic processes (as frequently occurring in manufacturing processing) with interleaving tasks.

While the above listing of relevant work refers to finite-state automata, another relevant class of techniques for obtaining decentralized controllers for DES starts from Petri net models. For example, the work reported in [24] and [25] proposes different means to consider distributed structures of the plant, and/or to produce controllers adhering to the principles of decentralization, but optimizing behavior is not considered. The approach in [26] combines the controller synthesis for manufacturing systems with optimization, but rather optimizes the use of resources than the costs associated with the transfer into goal states. The work in [27] explicitly relates to the design of distributed supervisors represented by Petri nets, but does not use principles of optimal control.

In contrast to the aforementioned appoaches, this paper follows the lines of algebraic controller computation as proposed in [28]. The main idea there is to the transfer principles of discrete-time linear time-invariant systems to the domain of the DES, and, in particular, to model distributed and hierarchical structures of the DES by algebraic descriptions. These ideas were used in [29] to obtain online-reconfigured feedback controllers, which account for the occurrence of failures or changing goal specifications. As in [30], the work in [29] employs DES models with a notion of transition costs to enable an ordering of feasible solutions and thus the computation of cost-optimal controllers. Both the approaches, however, were formulated for monolithic systems only, but not for a distributed setting of the DES.

In this paper, the algebraic computation of optimal state-feedback controllers for distributed systems is addressed, and, in particular, for linear and treelike dependence structures. These structures allow computing local controllers of subsystems separately, what may considerably reduce the overall computational effort, compared with the synthesis of centralized controllers. In addition, the scheme naturally leads to a set of local controllers to be implemented on separate hardware. In a preliminary version of this paper [31], an algorithm for a linear structure of interacting the DES was already proposed. This paper extends this work by an algorithm for terminal (independent) subsystems, the proof of optimality of the local controllers, and the consideration of tree structures, where each subsystem may depend on the output of several subsystems with lower priority.

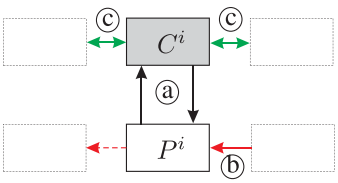

Fig. 1. Subsystem i as part of a larger structure. ⓐ: control loop of plant P^i and controller C^i. ⓑ: unidirectional plant coupling. ⓒ: controller communication.

This paper is organized such that Section II first introduces the modeling of subsystems and dependencies, the task of optimizing transfer costs for subsystems, and an algorithm for controller synthesis of independent subsystems. Section III focuses on linear system structures, presents a corresponding synthesis algorithm, and discusses the computational effort. While Section IV extends the consideration to treelike system structures, Section V illustrates the procedure for an example from the domain of assembly processes and demonstrates the computational complexity, and Section VI concludes this paper with a discussion.

II. Modeling and Control of a DES Subsystem

The objective of this paper is to investigate the controller synthesis for distributed DESs, in which several subsystems are interconnected by particular, directed dependencies. Before the overall system structure is introduced, this section first describes the model and the control task for a single subsystem. Consider Fig. 1 as illustration of such a subsystem with index i, consisting of a plant model P^i and a controller C^i. Both form a local control loop, which can be subject to coupling among subsystems on the plant level, as well as communication between controllers. The subsequent sections will clarify that the unidirectional coupling among subsystems, as shown in Fig. 1, prepares the priority structures envisaged in this paper.

A. Subsystem Definition

The plant P^i is modeled as deterministic finite-state automaton according to Definition 1.

Definition 1: The subsystem model $P^i = (T, U^i, X^i, Y^i, W^i, f^i, g^i)$ consists of an ordered time domain $T = \{0, 1, 2, \ldots\} \subset \mathbb{N} \cup \{0\}$ with $k \in T$ referring to an event time, and finite sets of discrete inputs $v_k^i \in U^i = \{1, \ldots, m_i\} \subset \mathbb{N}$, discrete states $\xi_k^i \in X^i = \{1, \ldots, n_i\} \subset \mathbb{N}$, and discrete outputs $y_k^i \in Y^i = \{1, \ldots, q_i\} \subset \mathbb{N}$. A deterministic state transition function is denoted by $f^i : X^i \times U^i \to X^i$, a

dependence matrix by $W^i \in \{0, 1, \ldots, q'\}^{m_i \times n_i}$, and an output function of *Moore-type* by $g^i : X \to Y^i$. □

The state, input, and output sets mentioned previously are specified as ordered index sets to ease notation—obviously, any index may represent a symbolic identifier of the respective quantity, as e.g., a name aligned to the physical state encoded by ξ_k^i.

Including the dependence matrix W^i into the definition of P^i is motivated by modeling that the state transitions of P^i may be dependent on another subsystem, say exemplarily P^{i+1}. For $a \in \{1, \ldots, m_i\}$ and $b \in \{1, \ldots, n_i\}$, the entry $W^i(a, b) = y^{i+1} > 0$ models that for P^i, the discrete transition out of state $b \in X^i$ and triggered by the input $a \in U^i$ depends on the output $y^{i+1} \in Y^{i+1}$ of the subsystem P^{i+1}. The value $W^i(a, b) = 0$ encodes, in contrast, that the same state transition is not dependent on an output of P^{i+1}. Admissible behavior of P^i is defined as follows.

Definition 2: Given P^i according to Definition 1, an initial state $\xi_0^i \in X^i$, and an input sequence $\phi_u^i := \{\nu_0^i, \nu_1^i, \ldots\}$ with $\nu_k^i \in U^i \cup \{0\}$, the elements of an *admissible run* $\phi_x^i := \{\xi_0^i, \xi_1^i, \ldots\}$ and a corresponding output sequence $\phi_y^i := \{y_0^i, y_1^i, \ldots\}$ follow for $k \in T$ from:

$$\xi_{k+1}^i = \begin{cases} \xi_k^i, & \text{if } \nu_k^i = 0 \\ f^i\left(\xi_k^i, \nu_k^i\right), & \text{if } \nu_k^i \in U^i,\ W\left(\nu_k^i, \xi_k^i\right) = 0 \\ f^i\left(\xi_k^i, \nu_k^i\right), & \text{if } \nu_k^i \in U^i,\ W^i\left(\nu_k^i, \xi_k^i\right) \in Y^{i+1} \end{cases} \tag{1}$$

$$y_{k+1}^i = g^i\left(\xi_{k+1}^i\right). \tag{2}$$

□

In (1), the case $\nu_k^i = 0$ models that no input is supplied in step k and the state remains unchanged, while the second and third cases encode state transitions, which are triggered by an input and are independent of P^{i+1}, or dependent on it, respectively.

As the objective of this paper is to present techniques to synthesize optimal state-feedback controllers for the DES, the following parts simplify the notation by abstracting from the presence of plant outputs.

Assumption 1: Any event of a subsystem P^i, $i \in \{1, \ldots, z\}$ is assumed to be observable, the state ξ_k^i to be fully measurable, and the output function g^i to be defined as identity function. □

This assumption implies $X^i = Y^i$ such that any occurence of y_k^i in Definitions 1 and 2 can be expressed in terms of x_k^i. With respect to the dependence of P^i on the subsystem P^{i+1}, the third case in (1) changes to $W^i(\nu_k^i, \xi_k^i) \in X^{i+1}$, i.e., the dynamics of P^i depends on the state of the linked subsystem.

Coupling the states of P^{i+1} to state transitions of P^i obviously requires to relate the time domains of the two subsystems. Assumption 2 establishes this relation in the sense that the subsystems (including their controllers) operate synchronously.

Assumption 2: The dynamics of P^i, $i \in \{1, \ldots, z\}$, and that of all subsystems to which P^i is coupled according to (1) are defined on the same time domain T, i.e., $k \in T$ is identical at any time for all of these subsystems, which thus iterate their states synchronously. □

This assumption is motivated by the typical situation that the cycle times of communication and control hardware are by orders of magnitude smaller than the average time between plant events (as modeled by f^i). If one abstracts from the cycles without plant events and includes only indices into T for times at which at least one P^i changes the state, the assumption is justifiable.

B. Algebraic Model Formulation

This paper proposes algorithms for controller synthesis, which bear similarity to procedures for computing (optimal) state-feedback controllers for linear discrete-time continuous-valued systems using dynamic proagramming. As an alternative option to the often used language-based model, we propose an algebraic system representation, which enables a synthesis procedure based on algebraic matrix operations (to be implemented, e.g., in MATLAB), which lead to a straighforward implementation of the local controllers, and for which an extension to hybrid controllers is possible. The following definitions to represent the dynamics of P^i extend the descriptions proposed in [29] for monolithic DES to the case of distributed system structures.

Definition 3: Let P^i be given according to Definition 1. For any state $\xi_k^i \in X^i$, define a *state vector* $x_k^i \in \{0, 1\}^{n_i \times 1}$ such that

$$x_{k,j}^i = 1, \text{ and } x_{k,p}^i = 0 \ \ \forall\ p \neq j,\ \ p \in \{1, \ldots, n_i\} \tag{3}$$

if and only if $\xi_k^i = j \in \{1, \ldots, n_i\}$ is the active state of P^i in k. A *state transition matrix* $F_l^i \in \{0, 1\}^{n_i \times n_i}$ is introduced for any input $l \in U^i$, such that for any pair $h, j \in X^i$ applies

$$F_l^i(j, h) = \begin{cases} 1, & \text{if } j = f^i(h, l) \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

By requiring that $F_l^i(j, j) = 1$ if $F_l(p, j) = 0$ for all $p \neq j$, $p \in \{1, \ldots, n_i\}$, a self-loop for the state with index j is provided, if no outgoing state transition exists for $l \in U^i$. □

The value $F_l^i(j, h) = 1$ is to be interpreted such that P^i can be transitioned from state $\xi_k^i = h$ into state $\xi_k^i = j$ if the input l is applied. Since f^i was defined to be deterministic in Definition 1, $\sum_{j=1}^{n_i} F_l^i(j, h) = 1$ applies for all $h \in \{1, \ldots, n_i\}$ and $l \in U^i$. The possible additional dependence of the transition on the state of subsystem P^{i+1} is considered within the following algebraic definition of a run of P^i.

Definition 4: For P^i, let $\mathcal{F}^i = \{F_1^i, \ldots, F_{m_i}^i\}$ denote the set of state transition matrices as introduced before. Given an initial vector $x_0^i \in \{0, 1\}^{n_i \times 1}$ with $\sum_{j=1}^{n_i} x_{0,j}^i = 1$ and an input sequence $\phi_u = (\nu_0^i, \nu_1^i, \nu_2^i, \ldots)$ as in Definition 2, a run $\phi_x^i = (x_0^i, x_1^i, x_2^i, \ldots)$ of P^i over T is *admissible*, if it satisfies

$$x_{k+1}^i = \begin{cases} x_k^i, & \text{if } \nu_k^i = 0 \\ F_j^i \cdot x_k^i, & \text{if } \nu_k^i \in U^i,\ W\left(\nu_k^i, \xi_k^i\right) = 0 \\ F_j^i \cdot x_k^i, & \text{if } \nu_k^i \in U^i,\ W^i\left(\nu_k^i, \xi_k^i\right) \in X^{i+1}. \end{cases} \tag{5}$$

□

For control of the distributed system structures to be discussed later, it is important that a subsystem of lower priority

can deliver the outputs (or states, respectively) on which a coupled plant relies. For example, a particular state of P^{i+1} is necessary for the progress of P^i, if $W^i(\nu_k^i, \xi_k^i) \in X^{i+1}$ encodes dependence. If one wants to model that a subsystem is able to deliver any signal that may be required by another, a conservative approach is to imply that any state $\xi^i \in X^i$ can be transferred into any other $\hat{\xi}^i \in X^i$ by at least one finite input sequence. For this purpose, an *accumulated state transition matrix* over all inputs F^i and a *reachability matrix* R^i are defined

$$F^i = \sum_{l=1}^{m_i} F_l^i, \quad R^i := \sum_{p=1}^{n_i} \left(F^i\right)^p. \tag{6}$$

The first matrix encodes with $F^i(j,h) = 1$ that the transition from $\xi_k^i = h$ to $\xi_k^i = j$ is possible with at least one input. In addition, to this one-step reachability, R^i formalizes the possibility of transferring P^i between an arbitrary pair of states. Note that the computations of F^i and R^i use Boolean arithmetic.[1] If the dependence conditions are satisfied, a value $R^i(j,h) = 1$ models that state j is reachable from state h by at least one input sequence in at most n_i state transitions. Similar, as described for monolithic systems, in [29], the subsystem P^i can be classified as *completely controllable*, if $R^i = 1^{n_i \times n_i}$.

C. Local Transition Costs and Control Task

The general objective of this paper is to establish discrete-event state-feedback controllers, which optimally drive the subsystems into a designated goal state. This section defines the underlying performance measure and the control task, both still confined to one subsystem. Performance is introduced by minimizing the costs of transferring the system between initial and goal state. For this purpose, transition costs $\pi(\xi_k^i, \xi_{k+1}^i, \nu_k^i)$ are defined for any transition $\xi_{k+1}^i = f^i(\xi_k^i, \nu_k^i)$ specified for P^i through the state transition function (or the set of state transition matrices $\mathcal{F}^i$, respectively). Possible interpretations of such transition costs are the time, the control effort, and/or the energy required to steer P^i from ξ_k^i to ξ_{k+1}^i by the use of the control input ν_k^i (i.e., π can encode state and control costs).

Definition 5: For any input $j \in U^i$ of the subsystem P^i, the *transition cost matrix* $\Pi_j^i \in \mathbb{R}_{\geq 0}^{n_i \times n_i}$ is defined such that $\Pi_j^i(q,p) = \pi(p,q,j)$ is the cost of the transition $f^i(p,j) = q$. The value $\Pi_j^i(q,p) = \infty$ is assigned if the transition is infeasible for input j (i.e., $F_j^i(q,p) = 0$), and self-loops do not incur costs: $\Pi_j^i(p,p) = 0$ for all $p \in X^i$, $j \in U^i$.

The matrix of *minimum transition costs*

$$\Pi^i := \left\{q, p \in \mathcal{X}^i : \min_{j \in \mathcal{U}^i} \Pi_j^i(q,p)\right\} \tag{7}$$

contains in any entry $\Pi^i(q,p)$ the minimum cost of the state transition $f^i(p,j) = q$ over all $j \in U^i$ values. The associated matrix

$$\Pi_{\mathcal{U}^i}^i := \left\{q, p \in \mathcal{X}^i : \Pi_{\mathcal{U}^i}^i = 0 \text{ if } F^i(p,q) = 0, \text{ and: } \Pi_{\mathcal{U}^i}^i(q,p) = \operatorname*{argmin}_{j \in \mathcal{U}^i} \Pi_j^i(q,p) \text{ if } F^i(p,q) = 1\right\} \tag{8}$$

of best inputs encodes with the element $\Pi_{\mathcal{U}^i}^i(q,p)$ the index of the input minimizing the transition costs if the transition is feasible, and zero otherwise.

Finally, the matrix Π_{opt}^i denotes the *minimal transfer costs* for P^i, such that $\Pi_{\text{opt}}^i(q,p)$ specifies the minimal costs for transferring the subsystem from the state $\xi^i = p$ into $\xi^i = q$ over all possible input sequences defined for P^i, which realize an admissible run from p to q. □

Note that determining the latter matrix Π_{opt}^i involves to compute cost-optimal runs—the following statement formalizes this computation as the control task to be solved for an independent subsystem P^i.

Problem 1: Let a subsystem P^i be given as *independent* of other subsystems in the sense that $W^i = 0^{m_i \times n_i}$, and let ξ_F^i denote the goal state. The task is to compute a state-feedback controller, which realizes for any arbitrary initialization $\xi_0^i \in X^i$ an input sequence $\phi_u^i = (\nu_0^i, \ldots, \nu_{d^i-1})$ that leads to an admissible run $\phi_\xi^i = (\xi_0^i, \ldots, \xi_{d^i}^A)$ such that the following hold.

1) The final state is the goal $\xi_{d^i}^i = \xi_F^i$.
2) The state-feedback control law has the structure

$$\nu_k^i = u^i \cdot K^i \cdot x_k^i \ \in U^i \tag{9}$$

with a row vector $u^i = [1, 2, \ldots, m_i]$ of all input indices in U^i and the controller matrix $K^i \in \{0,1\}^{m_i \times n_i}$.
3) The costs of ϕ_x^i are minimal over all admissible runs to transfer P^i from ξ_0^i to ξ_F^i

$$\Pi_{\text{opt}}^i\left(\xi_F^i, \xi_0^i\right) := \min_{\phi_u^i} \sum_{j=1}^{d^i} \Pi_{\nu_{j-1}^i}^i\left(\xi_j^i, \xi_{j-1}^i\right). \tag{10}$$

□

As is usual for state-feedback controllers of continuous systems, the law (9) should be interpreted such that for a given current state x_k^i, the product $u^i \cdot K^i$ selects the control input ν_k^i to be applied, in order to trigger the next state transition. Note that the selection of K^i according to the solution of (10) produces the ξ_F^ith row of the matrix Π_{opt}^i, and establishes an optimal controller for P^i with goal state ξ_F^i.

D. Synthesis Algorithm for Independent Subsystems

In order to solve problem 1, the algorithm in Fig. 2 computes the controller matrix K^i and the part of Π_{opt}^i referring to the goal state ξ_F^i. It can be seen as a modified Dijkstra-algorithm, which explores the state transition structure of P^i backward starting from ξ_F^i (i.e., following the principles of dynamic programming). Lines 2–4 contain the initialization of K^i, of an auxiliary vector V, and an auxiliary vector G. $V(q) = 1$ denotes that state $\xi^i = q$ still has to be explored, and $G(q)$ stores the final minimum costs for the transfer from q to ξ_F^i. The loop in lines 5–10 initializes another auxiliary

[1] For $a \in \{0,1\}$, $b \in \{0,1\}$, let: 1) $a+b = 0$ if and only if $(a = 0) \wedge (b = 0)$, and $a + b = 1$ otherwise and 2) $a \cdot b = 1$ if and only if $(a = 1) \wedge (b = 1)$, and $a \cdot b = 0$ otherwise.

ALGORITHM 1

1: **Input:** F^i, Π^i, $\Pi^i_{\mathcal{U}}$, ξ^i_F

2: $K^i := 0^{m_i \times n_i}$
3: $V := 1^{n_i \times 1}$, $V(\xi^i_F) := 0$
4: $G := \infty^{n_i \times 1}$, $G(\xi^i_F) := 0$
5: **for** $q = 1 : n_i$ **do**
6: **if** $F^i(\xi^i_F, q) \cdot V(q) = 1$ **then**
7: $H(q) := \Pi^i(\xi^i_F, q)$
8: $j := \Pi^i_{\mathcal{U}^i}(\xi^i_F, q)$, $K^i(j, q) := 1$
9: **end if**
10: **end for**
11: **while** $V \neq 0^{n_i \times 1}$ **do**
12: $l := \operatorname{argmin}_p H(p) \cdot V(p)$
13: $V(l) := 0$
14: $G(l) := H(l)$
15: **for** $q = 1 : n_i$ **do**
16: **if** $F^i(l, q) \cdot V(q) = 1$ **then**
17: **if** $G(l) + \Pi^i(l, q) < H(q)$ **then**
18: $H(q) := G(l) + \Pi^i(l, q)$
19: $K^i(:, q) := 0$
20: $j := \Pi^i_{\mathcal{U}^i}(l, q)$, $K^i(j, q) := 1$
21: **end if**
22: **end if**
23: **end for**
24: **end while**

25: **Output:** K^i, $\Pi^i_{opt}(\xi^i_F, :) := G$

Fig. 2. Computation of K^i and $\Pi^i_{\text{opt}}(\xi^i_F, :)$ for an independent subsystem.

vector H for the predecessor states of ξ^i_F. Here, $H(q)$ denotes the lowest cost found so far for the transfer from $\xi^i = q$ to ξ^i_F. The loop also sets the entries of K^i for these states to the indices of the inputs, which lead to cost minimal transitions into ξ^i_F. The main loop from line 11 to line 24 first determines the node still to be explored, which momentarily has the smallest temporary cost to reach ξ^i_F. For this node (with index l), the cost value is final [thus mapped into $G(l)$]. For the predecessors of state l, it is checked whether a new best path into ξ^i_F is found via state l. If so, the cost value $H(q)$ is updated, as is the entry of K^i, which corresponds to the input to be applied in state q to obtain minimal costs.

Theorem 1: Given a plant P^i for which problem 1 has to be solved. Then, for any state $\xi^i_0 \in X^i$, $\xi^i_0 \neq \xi^i_F$, Algorithm 1 computes K^i such that P^i is transferred into ξ^i_F with minimal costs.

Proof: Assume that a controller $\bar{K}^i$ existed, which transfers P^i from ξ^i_0 into ξ^i_F with lower costs than K^i, i.e., leading to $\bar{\Pi}^i(\xi^i_F, \xi^i_0) < \Pi^i_{\text{opt}}(\xi^i_F, \xi^i_0)$. Let $\bar{\phi}^i_u$ and $\bar{\phi}^i_\xi$ denote the optimal input sequence and run obtained by $\bar{K}^i$, while ϕ^i_u and ϕ^i_ξ correspond to the controller K^i. Then, there must exist a state $\xi^i_k \in \bar{\phi}^i_\xi$, $\xi^i_k \in \phi^i_\xi$ with $\xi^i_k \neq \xi^i_F$, for which $\bar{K}^i(:, \xi^i_k) \neq K^i(:, \xi^i_k)$. The controller $\bar{K}^i$ triggers the transition into a state $\bar{\xi}^i_{k+1} \neq \xi^i_{k+1}$ with input $\bar{\nu}^i_k \in \bar{\phi}^i_k$, leading to cost-to-go of $\pi^i(\bar{\xi}^i_k, \bar{\xi}^i_{k+1}) + \Pi^i_{\text{opt}}(\bar{\xi}^i_F, \bar{\xi}^i_{k+1})$, while K^i leads by $\nu^i_k \in \phi^i_k$ to a successor ξ^i_{k+1} and costs $\Pi^i_{\text{opt}}(\xi^i_F, \xi^i_k)$. The above-mentioned assumption requires that $\pi^i(\bar{\xi}^i_k, \bar{\xi}^i_{k+1}) + \Pi^i_{\text{opt}}(\bar{\xi}^i_F, \bar{\xi}^i_{k+1}) < \Pi^i_{\text{opt}}(\xi^i_F, \xi^i_k)$. However, when the algorithm iterates the loop in line 11 to line 24 with $l = \xi^i_{k+1}$ and $q = \xi^i_k$, it only assigns the value 1 to the jth row of $K(:, q)$ with $j = \nu^i_k$, if $G(l) + \Pi^i(l, q) = \Pi^i_{\text{opt}}(\xi^i_F, \xi^i_k) + \pi^i(\xi^i_k, \xi^i_{k+1})$ is lower than a previously determined value $H(q)$. In addition, if $K(:, q)$ is not rendered in any subsequent iteration, there cannot exist a state $\bar{\xi}^i_k$, for which $\pi^i(\bar{\xi}^i_k, \bar{\xi}^i_{k+1}) + \Pi^i_{\text{opt}}(\bar{\xi}^i_F, \bar{\xi}^i_{k+1}) < \Pi^i_{\text{opt}}(\xi^i_F, \xi^i_k)$. Thus, there does not exist a strategy $\bar{\phi}^i_u$ and a controller $\bar{K}^i$, which steers P^i from ξ^i_0 into ξ^i_F with lower costs than $\Pi^i_{\text{opt}}(\xi^i_F, \xi^i_0)$, and the lemma follows from contradiction of the above-mentioned assumption. ■

If the algorithm is run once for any $\xi^i \in X^i$ chosen as the goal state ξ^i_F, the complete matrix of minimal transfer costs Π^i_{opt} is obtained for P^i.

Note that this algorithm can also be used for centralized controller synthesis of a distributed system with dependence: if two subsystems $P^1 = (T, U^1, X^1, Y^1, W^1, f^1, g^1)$ and $P^2 = (T, U^2, X^2, Y^2, 0^{m_2 \times n_2}, f^2, g^2)$ according to Section II-A and with P^1 depending on P^2 are composed, the result is an automaton $P^{12} = \{T, U^1 \cup U^2, X^1 \times X^2, Y^1 \times Y^2, 0^{(m_1+m_2)\times(n_1 \cdot n_2)}, f^{12}\}$ of the same type with $g^{12} : (X^1 \times X^2) \to (Y^1 \times Y^2)$ and $f^{12} : (X^1 \times X^2) \times (U^1 \cup U^2) \to (X^1 \times X^2)$. The fact that the dependence matrix is a zero matrix shows that P^{12} is independent, meaning that only the first two lines of (1) are relevant, and that Algorithm 1 is applicable to P^{12}. This application will be used in Section V to compare the computational complexity with the proposed procedure of the following sections.

A result similar to the one in Theorem 1 was already derived in [19] for a different system representation and for the more general case of systems with uncontrollable events. Since the focus of this paper is to provide solutions for distributed systems (rather than robustness against uncontrollable events), the scope of this paper is restricted to deterministic state transitions.

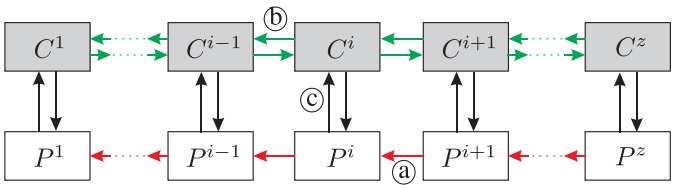

Fig. 3. Distributed system structure consisting of z feedback loops (P^i, C^i) and with linear dependence. ⓐ: any subsystem P^i $(i \neq z)$ depends on outputs of P^{i+1}. ⓑ: any controller C^i $(i \in \{2, \ldots, z-1\})$ communicates with the neighboring controllers C^{i-1} and C^{i+1}. ⓒ: any pair (P^i, C^i) exchanges local state and input information.

III. Control of Distributed Systems With Linear Structure

We now extend the consideration to distributed systems with several interconnected subsystems of the type P^i as introduced before. This section investigates the optimal control of processes consisting of z subsystems according to $\mathcal{P} = \{P^1, \ldots, P^i, \ldots, P^z\}$, which are interconnected in a linear structure as shown in Fig. 3. Any subsystem P^i forms a control loop with an assigned local controller C^i, both interacting as explained already by means of Fig. 1.

The dependence structure among the control loops adds the following. First, a state transition of P^i (for $i \in \{1,\dots,z-1\}$) may depend on a particular output provided by P^{i+1}. Then, the controller C^i (for $i \in \{2,\dots,z-1\}$) communicates with C^{i-1} and C^{i+1}. This communication is necessary to obtain the information which output has to be sent from P^i to P^{i-1}, and which output P^i requires from P^{i+1} for further execution. For this configuration, we say that P^i has *higher priority* than P^{i+1}. The following describes how the control task and the synthesis algorithm have to be extended to consider the subsystem interaction for this structure.

A. Task of Distributed Controller Synthesis

In contrast to the case of independent subsystems as in Section II-D, the distributed setting now involves to handle dependence matrices W^i different from zero matrices. Furthermore, control objectives (in particular costs) are formulated for the set of subsystems, i.e., the C^i values are not only tailored to local goals, but also consider the overall performance of $\mathcal{P}$. Except the more general case for W^i, the subsystem modeling remains the same as in Section II.

As mentioned in Section I, the motivation for considering the linear structure in Fig. 3 is to model plants in which P^{i+1} supplies P^i with material, or where P^{i+1} acomplishes a task for P^i, $i \in \{1,\dots,z-1\}$. This interpretation motivates Assumption 3.

Assumption 3: Any subsystem $P^i \in \mathcal{P}$, $i \in \{2,\dots,z\}$ is completely controllable: $R^i = 1^{n_i \times n_i}$. □

This assumption is justifiable in the sense that a process in which a unit cannot deliver the output required in another unit has to be characterized as not properly engineered. The same reasoning justifies the definition of deterministic state transition functions f^i.

In the dependence structure of $\mathcal{P}$ according to Fig. 3, the state transitions of P^i are only affected by the current plant state of P^{i+1}, but not any subsystem with higher index. To considerably simplify the notation for the following part, we restrict the presentation to only two subsystems indicated by $i = 1$ and $i = 2$. For P^2, the principles for independent subsystems as covered in Section II-D obviously apply. Section III-C will later show that the extension to chains with more than two subsystems is possible straightforwardly.

With respect to the controllability assumption for single subsystems, the following implication for the pair of P^1 and P^2 is possible.

Proposition 2: Let $\mathcal{P} = \{P^1, P^2\}$ be a pair of two connected subsystems for which an admissible run is a sequence of state pairs (ξ_k^1, ξ_k^2) according to Definition 2 with $W^1(\nu_k^1, \xi_k^1) \in X^2$ and $W^2 = 0^{m_2 \times n_2}$. The structure is completely controllable if P^1 and P^2 on their own are completely controllable according to Assumption 3. □

Proof: Since the transitions of P^2 are independent of the current state of P^1, and since subsystem P^2 is completely controllable, a sequence ϕ_u^2 of inputs exists to transfer P^2 from an arbitrary initial state ξ_0^2 into ξ_F^2. Thus, P^2 is able to deliver any arbitrary output sequence ϕ_y^2 (and thus admissible run ϕ_x^2) to subsystem P^1, i.e., any condition formulated for P^1 in terms of the dependence matrix W^1 is satisfiable by P^2. Since P^1 itself is completely controllable as well, a sequence of inputs ϕ_u^1 exists which transfers P^1 into an arbitrary goal state ξ_F^1. ■

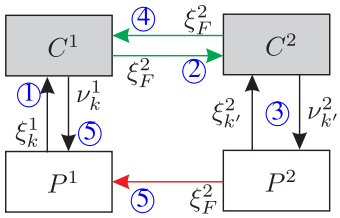

Fig. 4. Online-execution for one state transition of P^1 including the provision of ξ_k^2 by P^2. The numbers indicate the order of information processing.

The problem to be considered for the pair of subsystems can now be stated as follows.

Problem 2: For the two subsystems P^1 and P^2, let the goal states ξ_F^1 and ξ_F^2 be defined. The control task is to compute two local feedback control laws, which generate for **any** initialization $\xi_0^1 \in X^1$ and $\xi_0^2 \in X^2$ the input sequences $\phi_u^1 = (\nu_0^1, \dots, \nu_{d_1^1-1})$ and $\phi_u^2 = (\nu_0^2, \dots, \nu_{d_2^2-1})$, such that the following hold.

1) The admissible runs $\phi_x^1 = (\xi_0^1, \dots, \xi_{d_1^1}^1)$ with $\xi_{d_1^1}^1 = \xi_F^1$ and $\phi_x^2 = (\xi_0^2, \dots, \xi_{d_2^2}^2)$ with $\xi_{d_2^2}^2 = \xi_F^2$ are obtained.
2) ϕ_u^1 and ϕ_u^2 follow from controllers of the following type:
$$\nu_k^1 = u^1 \cdot K^1(\xi_k^2) \cdot x_k^1 \in U^1, \quad \nu_k^2 = u^2 \cdot K^2 \cdot x_k^2 \in U^2 \tag{11}$$
with vectors u^1 and u^2, and matrix K^2 as in problem 1, and $K^1(\xi_k^2) \in \{0,1\}^{m_1 \times n_1}$ for $\xi_k^2 \in X^2$.
3) The global path costs are minimal
$$J_g = \sum_{k=1}^{d_1^1} \Pi_{\nu_{k-1}^1}(\xi_k^1, \xi_{k-1}^1) + \sum_{k=1}^{d_2^2} \Pi_{\nu_{k-1}^2}(\xi_k^2, \xi_{k-1}^2). \tag{12}$$
□

Thus, the solution is targeted to provide local controllers C^1 and C^2 for P^1 and P^2, such that the latter are driven from an arbitrarily chosen initial state into the respective local goal state, while the sum of the transfer costs for both control loops is as small as possible. As in problem 1, the controllers are of a state-feedback type, where K^2 is independent of P^1, but the controller matrices $K^1(\xi_k^2)$ depend on the current state ξ_k^2 of subsystem P^2. Thus, C^2 comprises n_2 matrices, and the matrix set is denoted by $\mathcal{K}^1$.

Before the algorithmic solution to problem 2 is presented, the interactions of the two control loops during online execution are clarified by means of Fig. 4: when C^1 receives the information from P^1 that state ξ_k^1 is reached ①, C^1 sends the *request* to C^2 that P^2 has to reach ξ_F^2 as a temporary goal state ②. This state is encoded in K^1 in order to realize a cost-optimal path of P^1 into its goal state ξ_F^1. Then, C^2 realizes a path of P^2 into ξ_F^2 ③. If the path comprises more than one transition, the pair (P^1, C^1) waits in state ξ_k^1 until P^2 has reached ξ_F^2 (the index k' in Fig. 4 is meant to indicate that (P^2, C^2) evolve, while (P^1, C^1) wait in step k).

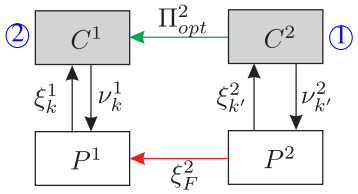

Fig. 5. Information flow within the offline synthesis of K^1 and K^2.

When P^2 attains ξ^2_F, C^2 communicates to C^1 that the requested state is reached ④. Eventually, the control input ν^1_k supplied by C^1 together with ξ^2_F send by P^2 triggers the state transition in P^1 ⑤.

B. Synthesis Algorithm for Two Subsystems

The offline determination of the distributed controllers C^1 and C^2 consists of two phases, for which the information flow is shown in Fig. 5: in step ①, the matrix K^2 of the controller C^2 is computed for any output (or state ξ^2_k, respectively) that the subsystem with higher priority, i.e., the pair (P^1, C^1), may request during online execution. If any state of P^2 may be requested, the result of the synthesis of C^2 has to be the full matrix Π^2_{opt} of optimal transfer costs for any pair of initial and goal states in X^2. Then, controller matrix $K^2(\xi^2_F)$ is required for a possible ξ^2_F, leading to a controller set $\mathcal{K}^2$. To obtain Π^2_{opt} and the controllers in $\mathcal{K}^2$, Algorithm 1 can be used for the relevant goal states ξ^2_F, since P^2 is independent of P^1. For the states of P^2, which may never be requested by (P^1, C^1), the corresponding rows of Π^2_{opt} may simply be set to $\infty^{1\times n_2}$, and the respective control matrices need not to be computed.

The more intricate synthesis task is that of obtaining K^1 (see step ② in Fig. 5). This task builds on Π^2_{opt} and is the subject of the following description. The algorithm for solution, which is shown in Fig. 6, employs the dynamic programming principle [32]. It may be understood as a version of the Bellman–Ford algorithm (see [33]) with reversed transition through the graph of P^1, and with additional consideration of the dependence on the states of P^2 as well as the costs incurred by this subsystem.

The input data of Algorithm 2 comprises the matrices of transition costs Π^1_ν of P^1 for all $\nu \in U^1$, the dependence matrix W^1, the matrix Π^2_{opt} of optimal path costs for P^2, and a pair of goal states ξ^1_F and ξ^2_F. The last entry in this list of inputs means that P^2 has to reach a specified goal state, after it has provided the necessary symbols to P^1. (If P^2 only has to supply P^1 but the final state is not important, line 3 of the algorithm is simply changed to $H_0(\xi^1_F, :) = 0^{1\times n_2}$).

In lines 2–5, the controller matrices $K^1(\xi^2)$ are initialized to zero matrices for all $\xi^2 \in X^2$, and the auxiliary matrix H_0 is initialized to ∞ for any entry, except the one referring to the pair of goal states. Similar to vector $H(q)$ in Algorithm 1, the matrix $H_a \in \mathbb{R}^{n_1\times n_2}_{\geq 0}$ (with iteration counter a) is iteratively updated with the temporary lowest costs to realize the transition into (ξ^1_F, ξ^2_F). Upon termination of the algorithm, the element $H_a(j,l)$ contains the minimized costs for transferring P^1 and P^2 from the state pair $j \in X^1$ and $l \in X^2$ to the specified pair of goal states. The transfer requires at most $a \cdot n_2$ steps, since, in the worst case, P^1 has to wait for n_2 steps until the relevant dependence condition of the respective step is satisfied. This is due to the fact that the number of state transitions to transfer the completely controllable subsystem P^2 between two arbitrary states is at most n_2.

ALGORITHM 2

1: **Input:** Π^1_ν, W^1, Π^2_{opt}, ξ^1_F, ξ^2_F
2: $K^1(\xi^2) := 0^{m_1\times n_1}$ for all $\xi^2 = \{1, ..., n_2\}$
3: $H_0 := \infty^{n_1\times n_2}$, $H_0(\xi^1_F, \xi^2_F) := 0$
4: $H_1 := H_0$
5: $a := 1$
6: **while** $(a = 1) \vee (H_a \neq H_{a-1})$ **do**
7: $H_a := H_{a-1}$
8: **for** $k = 1 : n_1$ **do**
9: **for** $m = 1 : m_1$ **do**
10: $j := \begin{bmatrix}1 & \cdots & n_1\end{bmatrix} \cdot F^1_m(:, k)$
11: **for** $l = 1 : n_2$ **do**
12: **if** $W^1(m,k) > 0$ **then**
13: $r := W^1(m,k)$
14: $H_{sum} := H_{a-1}(j,r) + \Pi^1_m(j,k) + \Pi^2_{opt}(r,l)$
15: **if** $H_{sum} < H_a(k,l)$ **then**
16: $H_a(k,l) := H_{sum}$
17: $K^1(l)(:,k) := 0$
18: $K^1(l)(m,k) := 1$
19: **end if**
20: **else**
21: **if** $H_{a-1}(j,l) + \Pi^1_m(j,k) < H_a(k,l)$ **then**
22: $H_a(k,l) := H_{a-1}(j,l) + \Pi^1_m(j,k)$
23: $K^1(l)(:,k) := 0$
24: $K^1(l)(m,k) := 1$
25: **end if**
26: **end if**
27: **end for**
28: **end for**
29: **end for**
30: $a := a + 1$
31: **end while**
32: **Output:** $K^1(\xi^2)$ for $\xi^2 \in X^2$

Fig. 6. Computation of the control matrix $K^1(\xi^2_k)$ to transfer P^1 and P^2 to the goal states with minimized total costs.

The first step of the main loop of the algorithm (line 7) copies the cost matrix of the previous iteration to H_a. Any iteration a checks then which entries of H_a can be reduced over the possible combinations of discrete states $k \in X^1$, discrete inputs $m \in U^1$, and discrete states $l \in X^2$ (loop beginning in lines 8, 9, and 11). Concretely, the cost for a transition of P^1 from state k to state j triggered by the input m is computed and compared with the best cost computed so far. For this transition, two cases have to be considered (lines 12 and 20): P^1 requires that P^2 is currently in a specific discrete state $[W^1(m,k) > 0]$, or the transition from k to j by input m is independent of the current state of P^2 $(W^1(m,k) = 0)$.

For the first case, the discrete state $r = W^1(m,k) \in X^2$ of P^2 is determined, which is required for the transition of P^1. Subsequently, the cost incurred by the transition is determined as $\Pi^1_m(j,k) + \Pi^2_{\text{opt}}(r,l)$. This value adds the local transition

cost $\Pi^1_m(j,k)$ to the optimal transition cost $\Pi^2_{\text{opt}}(r,l)$ assigned to the state transfer in P^2, necessary to satisfy the dependence condition of the considered transition of P^1. If this transition leads to a cost reduction for the momentarily investigated combination of states k of P^1 and l of P^2, i.e., if

$$H_{a-1}(j,r) + \Pi^1_m(j,k) + \Pi^2_{\text{opt}}(r,l) < H_a(k,l) \qquad (13)$$

applies, the value of $H_a(k,l)$ is updated to the new and lower value (line 16). In this case, the entries of the controller matrix are updated by replacing a possibly existing nonzero entry of $K^1(l)$ for the currently investigated state by zero, and by setting the entry $K^1(l)(m,k)$, which corresponds to the momentarily investigated pair of state k and input m, to 1 (lines 17 and 18).

The procedure for the case of $W^1(m,k) = 0$ is very similar: since $W^1(m,k) = 0$ encodes that the currently investigated transition of P^1 is independent of the current state of P^2, only the local transition costs of P^1 are determined, and H_a is checked for a cost reduction. If the costs are lowered, the value of $H_a(k,l)$ and the controller matrix are updated (lines 22–24).

The algorithm terminates at the end of a while-iteration if $H_a = H_{a-1}$ applies, i.e., if no further cost reduction can be determined. Since the longest path without cycles between two arbitrary states of P^1 does at most contain n_1 states and since cycles can only increase costs, the number of iterations of the while-loop is also limited to n_1.

The key benefit of Algorithm 2 can now be stated.

Theorem 3: Let problem 1 for $\mathcal{P} = \{P^1, P^2\}$ be given, while $\mathcal{P}$ satisfies Proposition 2. Then, Algorithm 2 computes the control matrices $\mathcal{K}^1$ to transfer P^1 from an arbitrary initial state $\xi^1_0 \in X^1$, $\xi^1_0 \neq \xi^1_F$ into ξ^1_F with minimally possible value of the costs in (12), provided K^2 was computed by Algorithm 1.

Proof: First, Proposition 2 guarantees that (P^2, C^2) is able to deliver any symbol r (according to line 13) that is required by P^1. Given Theorem 1, Algorithm 1 provides Π^2_{opt} such that this matrix encodes the least possible path costs for any transfer between a pair of states in P^2. Hence, whenever Algorithm 2 considers a state transition of P^1, which depends on P^2, the last term of H_{sum} in line 13 uses the least possible cost necessary for P^2 to provide the symbol r and to reach the goal state. This minimizes the second sum of (12).

Furthermore, Proposition 2 also ensures that $K^1(l)$, $l \in \{1, \ldots, n_2\}$ exists to obtain an admissible run of P^1 from any ξ^1_0 into ξ^1_F. That this run is cost minimal, which follows from the following reasoning (compared with that of the proof of Algorithm 1): Whenever the matrix $K^1(l)$ is updated in line 17/18 or 23/24 to a value 1 in row m and column k, the cost sum for the path from state j into ξ^1_F plus the cost for the transition from state k into l upon input m, plus the cost $\Pi^2_{\text{opt}}(r,l)$ for dependent transitions, is smaller than the costs determined before to transfer from state k to ξ^1_F. Since the loops run over all states and inputs of P^1 until no entry of H_a is further reduced, $H_a(k)$ contains upon termination, the minimum costs to transfer P^1 from k to ξ^1_F are obtained. Thus, the set $\mathcal{K}^1 = \{K^1(1), \ldots, K^1(n_2)\}$ produces for any pair of states (ξ^1_0, ξ^1_F) the path, which minimizes the first sum of (12), and hence also the minimal global path costs. ■

C. Effort Estimation and Extensions

The computational effort for determining the controllers C^1 and C^2 for $\mathcal{P} = \{P^1, P^2\}$ is now considered in order to assess the scalability of the controller synthesis with the size of the state and input sets of the subsystems. The computations for the independent subsystem according to Algorithm 1, i.e., of Π^2_{opt} and K^2, are (for given inputs) of the order $\mathcal{O}(n_2^2 m_2 + n_2^3)$. The first term refers to the computation of K^2 (of dimension $m_2 \cdot n_2$) for at most n_2 repetitions of the while-loop, and the second term to computing l at most for n_2 times. For the determination of $\mathcal{K}^1 = \{K^1(1), \ldots, K^1(n_2)\}$ using Algorithm 2, the computational effort grows (for given inputs) according to $\mathcal{O}(n_1^2 m_1 n_2)$. This effort follows from updating H_a at most $n_1 \cdot n_2$ times, and from the fact that the loops over m and l are embedded into the iteration to construct the matrices $K^1(l)$. Hence, the overall computational effort of the synthesis is of the order $\mathcal{O}(n_2^2 m_2 + n_1^2 m_1 n_2 + n_2^3)$.

An alternative to the proposed procedure of computing C^2 and C^1 sequentially is a centralized design consisting of four steps: First, the parallel composition of P^1 and P^2 is computed, leading to a larger model according to Definition 1 with state space of size $n_1 \cdot n_2$. This step together with the determination of the corresponding transition cost matrices requires an effort of the order $\mathcal{O}(n_1 n_2 m_1 m_2)$. The second step involves the computation of the matrices of optimal one-step transition costs with effort $\mathcal{O}(n_1^2 n_2^2 m_1 m_2)$. The third step determines the controller matrix K for the composed system, for which the procedure in Section II-D can be employed. (Note that the composed model is independent.) The controller synthesis incurs an effort of the order $\mathcal{O}(n_1^2 n_2^2)$. In order to obtain local controllers, the controller matrices K^1 and K^2 are extracted from K, which leads to costs of $\mathcal{O}(n_1 n_2)$. In total, the effort for this alternative option to compute the local control laws can be summarized to be of order $\mathcal{O}(n_1^2 n_2^2 m_1 m_2)$. The difference in absolute terms will be reported for an example in Section V.

With respect to possible extensions of Algorithm 2, it should be mentioned first that the algorithm in Fig. 6 is formulated for the case of one specified pair of goal states ξ^1_F and ξ^2_F. However, the algorithm succeeds as well in computing the set of controller matrices $\mathcal{K}^1$ for arbitrary sets of goal states by setting the value 0 to all corresponding entries of H_0. For the example of two pairs of goal states $(\xi^1_{F,1}, \xi^2_{F,1})$ and $(\xi^1_{F,2}, \xi^2_{F,2})$, respectively, the matrix elements $H_0(\xi^1_{F,1}, \xi^2_{F,1})$ and $H_0(\xi^1_{F,2}, \xi^2_{F,2})$ are set to zero, while the remaining elements are set to ∞.

As already mentioned, the proposed synthesis procedure is also applicable for systems $\mathcal{P}$ with more than $z = 2$ subsystems, but the computations require slight modifications: while for $z = 2$, the computation of K^1 requires the matrix Π^2_{opt}, the case for $z = 3$ with $\mathcal{P} = \{P^1, P^2, P^3\}$ requires a matrix $\Pi^{B,C}_{\text{opt}}$ when determining K^1. The matrix $\Pi^{B,C}_{\text{opt}}$ contains the optimal path costs for any combination of initial states ξ^2_0 and ξ^3_0 and

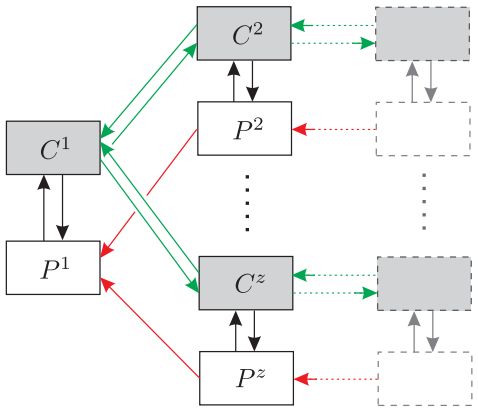

Fig. 7. Distributed system with tree structure where the feedback loop (P^1, C^1) depends on the loops of the subsystems (P^2, C^2) to (P^z, C^z) (each of which may depend on further subordinated subsystems).

goal states ξ_F^2 and ξ_F^3 with $\xi_0^2, \xi_F^2 \in X^2$ and $\xi_0^3, \xi_F^3 \in X^3$. In addition, the controller matrix K^1 depends on the current state ξ_k^2 of P^2 and the current state ξ_k^3 of P^3. Thus, the matrix $H_0 \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ has three dimensions, such that the algorithm has to comprise a further for-loop over the states in $X^3 = \{1^3, \ldots, n_3^3\}$.

IV. Control of Distributed Systems With Tree Structure

Another class of dependence structures, which is amenable to separate the controller synthesis for the subsystems, is that of treelike coupling (see Fig. 7). The subsystem P^1 depends on a set of subsystems P^2 to P^z, which itself may depend on one or more subsystems. The interpretation of this structure is that P^1 (the root of the tree) is the subsystem of highest priority in the sense that it sends out requests to and is supplied by P^2 to P^z. This requires that the dependence matrix W^1 has to be extended to comprise outputs (or states respectively) of the interconnected plant subsystems. According to Definition 1, a dependence matrix of a subsystem P^i depending on one subsystem P^{i+1} was defined so far by $W^i(a, b) \in \{0\} \cup Y^{i+1}$ for $a \in U^i$, $b \in X^i$. For the configuration shown in Fig. 7, i.e., P^1 depending on the outputs of P^2 to P^z, the definition of W^1 is changed to

$$W^1(a, b) \in \begin{bmatrix} \{0\} \cup Y^2 \\ \vdots \\ \{0\} \cup Y^z \end{bmatrix}, \quad a \in U^i, \ b \in X^i.$$

Thus, each element of the matrix maps into a vector of length $z - 1$. For $j \in \{2, \ldots, z\}$, the jth element $W_j^1(a, b)$ of the vector encodes whether the state transition of P^1 out of state a and triggered by the input b is independent of P^j [then $W_j^1(a, b) = 0$], or which output $y^j \in Y^j$ of P^j is necessary for the transition. In order to execute the transition, all subsystems P^j with $W_j^1(a, b) \neq 0$ must have supplied the specified output to enable the state transition. If the subsystems P^j themselves would depend on more than one subsystem, their dependence matrices would again be adopted correspondingly. Thus, arbitrary tree structures can be obtained. Except the modification of the dependence matrices, the plant modeling remains the same as in Section II.

For simplifying the description, the further discussion of the controller synthesis is tailored to the case $\mathcal{P} = \{P^2, P^2, P^3\}$, where P^1 depends on the outputs of P^2 and P^3. (The procedure can straightforwardly be extended to more than two supplying subsystems, i.e., $z > 3$. Also, the consideration of further layers of the tree, e.g., the dashed blocks on the right of Fig. 7, is possible by recursing from the right.)

Assume that P^2 and P^3 are independent and completely reachable, and that the corresponding controllers C^2 and C^3 have been synthesized according to the procedure of Section II-D. Furthermore, assume that $R^1 = 1^{n_1 \times n_1}$. Then, with the same reasoning as in the proof of Proposition 2, $\mathcal{P}$ is completely reachable. The task of computing the controller C^1 for P^1 can then be stated as follows.

Problem 3: Let the distributed system $\mathcal{P} = \{P^1, P^2, P^3\}$ be given with goal states ξ_F^1, ξ_F^2, and ξ_F^3. For P^2 and P^3, let the control matrices K^2 and K^3 and the matrices of minimum state transfer costs Π_{opt}^2 and Π_{opt}^3 be computed by Algorithm 1.

The task is to compute a set of control matrices $\mathcal{K}^1 = \{K^1(1), \ldots, K^1(q)\}$ for $q \in X^2 \times X^3$ such that for arbitrary initialization $\xi_0^1 \in X^1$, $\xi_0^2 \in X^2$, and $\xi_0^3 \in X^3$, the following holds true. The controllers $\{\mathcal{K}^1, K^2, K^3\}$ determine input sequences $\phi_u^1 = (\nu_0^1, \ldots, \nu_{d^1-1})$, $\phi_u^2 = (\nu_0^2, \ldots, \nu_{d^2-1})$, and $\phi_u^3 = (\nu_0^3, \ldots, \nu_{d^3-1})$ such that the following hold.

1) The admissible runs $\phi_x^1 = (\xi_0^1, \ldots, \xi_{d^1}^1)$ with $\xi_{d^1}^1 = \xi_F^1$ and $\phi_x^2 = (\xi_0^2, \ldots, \xi_{d^2}^2)$ with $\xi_{d^2}^2 = \xi_F^2$, and $\phi_x^3 = (\xi_0^2, \ldots, \xi_{d^3}^2)$ with $\xi_{d^3}^2 = \xi_F^3$ are obtained (where the upper bounds d^i of the indices indicate the length of the runs).
2) ϕ_u^1 is obtained from the control law

$$\nu_k^1 = u^1 \cdot K^1(\xi_q) \cdot x_k^1 \in U^1 \tag{14}$$

with the vector $u^1 = [1 \ \cdots \ m_1]$ and $K^1(q) \in \{0, 1\}^{m_1 \times n_1}$ for $q \in X^2 \times X^3$.
3) The following global path costs are minimal:

$$J_g = \sum_{k=1}^{d_1^1} \Pi_{\nu_{k-1}^1}(\xi_k^1, \xi_{k-1}^1) + \sum_{k=1}^{d_2^2} \Pi_{\nu_{k-1}^2}(\xi_k^2, \xi_{k-1}^2) + \sum_{k=1}^{d_3^3} \Pi_{\nu_{k-1}^3}(\xi_k^3, \xi_{k-1}^3). \tag{15}$$

□

This problem can be solved by an algorithm, which is obtained as a modification of Algorithm 2 with respect to considering the extended dependence structure (14). In the interest of space, Fig. 8 shows only the main part of Algorithm 2, which is modified compared with Algorithm 2, i.e., the lines 3–24 replace the lines 11–27 of Algorithm 2. In Algorithm 2, H_a is a 3-D matrix for storing the best costs found so far (similar to the case of three subsystems in linear structure, as mentioned at the end of Section III-C). The three indices of $H_a(j, l, p)$ refer to the states of the three subsystems: $j \in X^1, l \in X^2, p \in X^3$. The shown part of Algorithm 3 loops over the states of P^2 and P^3, and the

ALGORITHM 3 (Excerpt)

Input: Π^1_ν, W^1, Π^2_{opt}, Π^3_{opt}, ξ^1_F, ξ^2_F, ξ^3_F

...
for $l = 1 : n_2$ **do**
  for $p = 1 : n_3$ **do**
    $r_1 = W^1_1(m,k)$, $r_2 = W^1_2(m,k)$
    if $r_1 > 0$ **then**
      if $r_2 > 0$ **then**
        $\Pi_{pre} = H_{a-1}(j, r_1, r_2) + \Pi^2_{opt}(r_1, l) + \Pi^3_{opt}(r_2, p)$
      else
        $\Pi_{pre} = H_{a-1}(j, r_1, l) + \Pi^2_{opt}(r_1, l)$
      end if
    else
      if $r_2 > 0$ **then**
        $\Pi_{pre} = H_{a-1}(j, l, r_2) + \Pi^3_{opt}(r_2, p)$
      else
        $\Pi_{pre} = H_{a-1}(j, l, p)$
      end if
    end if
    if $\Pi^1_m(j,k) + \Pi_{pre} < H_a(k,l,p)$ **then**
      $H_a(k,l,p) := \Pi^1_m(j,k) + \Pi_{pre}$
      $K^1(l)(:,k) := 0$, $K^1(l)(m,k) := 1$
    end if
  end for
end for
...

Output: $K^1((\xi^2, \xi^3))$ for $(\xi^2, \xi^3) \in X^2 \times X^3$

Fig. 8. Part of the synthesis algorithm if P^1 depends on P^2 and P^3.

update of the controller matrix K^1 in line 21 as well as the update of H_a in line 22 is bound to the condition that a new best path from the state triple (k, l, p) into $(\xi^1_F, \xi^2_F, \xi^3_F)$ is found (line 19). The cost value Π_{pre} is an auxiliary variable, which is set depending on the condition whether the respective transition out of state $\xi^1 = k$ is depending on the state of P^2, and/or on the state of P^3, or is independent of the latter two subsystems ($r_1 = r_2 = 0$).

The fact that Algorithm 3 determines the set of controller matrices $K^1(\xi^2, \xi^3)$ such that the transfer costs into $(\xi^1_F, \xi^2_F, \xi^3_F)$ are minimal can be shown by similar arguments as in the proofs for Theorems 1 and 3; this is omitted here for brevity.

V. Illustrative Example

This section reports on the application of the synthesis scheme to a section of a larger manufacturing process, which complies with the structure considered in Section III-A. This system consists of two linearly dependent machines, where P^2 represents a bending machine that can produce parts of two different shapes and different colors. These parts are further processed in a mounting machine, which fixes the parts to base plates according to two different product specifications, both requiring different tools within the machine. While this process (sketched in Fig. 9) is small, it is a typical section of production chain and is suitable to illustrate the synthesis procedure.

Fig. 10 shows the transition graph of P^2 containing the discrete states, discrete inputs, and costs of the transitions representing the bending process. Starting from an initial state $\xi^2 = 1^2$, two different shapes (represented by $\xi = 2^2$ and $\xi = 3^2$, respectively) can be produced. The notation of the state transition from $\xi^2 = 1^2$ to $\xi^2 = 2^2$ encodes that this

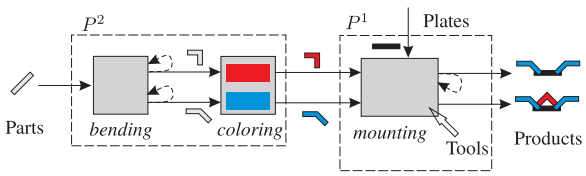

Fig. 9. Scheme of the production process.

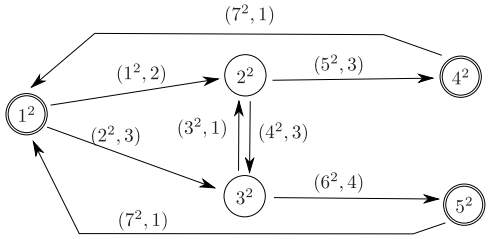

Fig. 10. Graph of the bending and coloring machine P^2 with state identifiers ξ^2_k specified within circles. The transitions are labeled by a pair $(\upsilon^2_k, \pi(\xi^2_k, \xi^2_{k+1}, \nu^2_k))$, consisting of the discrete input υ^2_k and the local transition costs $\pi(\xi^2_k, \xi^2_{k+1}, \nu^2_k)$. No self-loop transitions are shown for simplification.

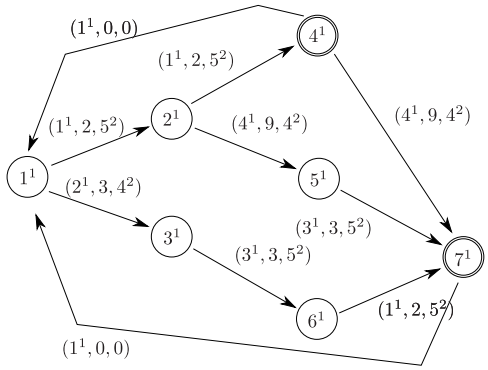

Fig. 11. Graph of the mounting machine P^1 with state identifiers ξ^1_k again specified within circles. The transition labeling in terms of triples $(\upsilon^1_k, \pi(\xi^1_k, \xi^1_{k+1}, \nu^1_k), W^1(m,k))$ includes the discrete input ν^1_k, the transition costs $\pi(\xi^1_k, \xi^1_{k+1}, \nu^1_k)$, and the entry of the dependence matrix W^1. Again, self-loop transitions are not shown.

transition is triggered by the local input $\nu^2 = 1^2$ and entails costs of 2. Note that after the first bending process, it is possible to reshape the two types to the respective other type with additional effort. Subsequently, a coloring step within P^2 leads to the final products: $\xi^2 = 4^2$ represents a red product, and $\xi^2 = 5^2$ represents a blue product. Any time when P^2 reaches one of the states 4^2 or 5^2, the production cycle of this unit is completed by returning to the state 1^2 through the input 7^2 (incurring costs of 1). This is simply achieved by specifying an intermediate goal $\xi_F 2 = 1^2$ for P^2.

The other machine is modeled as P^1 and depends on the supply by P^2. Depending on a given product specification, it mounts two or three of the parts supplied by P^2 to a base plate. The two considered product specifications are that two blue parts are mounted to the plate, or that in addition, one red part is added. Fig. 11 contains the transition graph to model the possible mounting sequences, and it shows for P^1 the discrete states, the discrete inputs, the costs of state transitions, and the dependence conditions. The mounting process can be understood as follows: from the initial state $\xi^1 = 1^1$, a bent

blue part or a bent red product is mounted to the base plate, leading to $\xi^1 = 2^1$ or $\xi^1 = 3^1$, respectively. Since no buffer is provided in between the two machines, it is important that the machine modeled by P^2 produces a part only if this is required currently by the machine represented by P^1. The state transitions are here denoted such that, e.g., the transition from $\xi^1 = 1^1$ to $\xi^1 = 2^1$ encodes that this transition is triggered by the local input $\nu^1 = 1^1$, entails costs of 2, and requires that machine P^2 is currently in state 5^2. The following transitions model the additional mounting of a one blue part (leading to the first product modeled by state 4^2) or fixing one red part and one more blue part (in different orders), leading to the state denoted by state 7^1. As the tool change entails additional costs, the transition costs from 5^1 to 7^1 is higher compared with the transition from 6^1 to 7^1. Note that all inputs, which do not change the discrete state have zero costs, and that P^1 can return to the initial state from the states 4^1 and 7^1.

Based on the transition graphs, the dependence matrix W^1 can be specified, and the matrix Π^2_{opt} of optimal transfer costs is obtained from Algorithm 1

$$W^1 = \begin{bmatrix} 5 & 5 & 0 & 0 & 0 & 5 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 & 5 & 0 & 0 \\ 0 & 4 & 0 & 4 & 0 & 0 & 0 \end{bmatrix}$$

$$\Pi^2_{\text{opt}} = \begin{bmatrix} 0 & 4 & 5 & 1 & 1 \\ 2 & 0 & 1 & 3 & 3 \\ 3 & 3 & 0 & 4 & 4 \\ 5 & 3 & 4 & 0 & 6 \\ 7 & 7 & 4 & 8 & 0 \end{bmatrix}.$$

The controller matrix for P^2 follows from the same algorithm, and is obtained for the example of $\xi^2_F = \xi^2_4$ to:

$$K^2 = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

This matrix should be interpreted according to (9) such that, e.g., for state 2^2 (corresponding to the second column), the input $\nu^1 = 5$ (referring to the fifth row) has to be applied.

Subsequently, Algorithm 2 can be used to compute the controller matrices $K^1(\xi^2_k)$ for the pair of reference states $\xi^1_F = 7^1$ and $\xi^2_F = 1^2$. For the example of $\xi^2_k = 2^2$, the result is

$$K^1(2^2) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Now consider that the current states are $\xi^1_k = 2^1$ and $\xi^2_k = 2^2$, when the goal specification $\xi^1_F = 7^1$ is supplied to the system. For the subsystem P^1 alone, the path $2^1 \to 4^1 \to 7^1$ incurs the lowest costs of 11. Nevertheless, the controller C^1 chooses the input sequence ($\nu^1_k = 4^1, \nu^1_k = 3^1$), which triggers the path $2^1 \to 5^1 \to 7^1$ for P^1 and $2^2 \to 4^2 \to 1^2 \to 3^2 \to 5^2 \to 1^2$ for P^2. The reason is that the total costs sum up to 24 for the latter pair of path, while the pair $2^1 \to 4^1 \to 7^1$ and $2^2 \to 3^2 \to 5^2 \to 1^2 \to 2^2 \to 4^2 \to 1^2$ leads to total costs of 25.

The time for the computation of this control law with an implementation of Algorithm 2 in MATLAB is 5 ms (Intel Core^{TM} i5 CPU @ 2.67 GHz $\times$ 4). For comparison, the solution with parallel composition of P^1 and P^2 and subsequent computation of a centralized controller (as sketched in Section II-D) requires 0.31 s for the same example. Thus, the effort for a centralized design is already by a factor of 60 higher for this small example, compared with the proposed algorithms. As one can see from the discussion in Section III-C, similar ratios can be expected for growing n_1, m_1, and m_2, while the gap can be expected to decrease for growing n_2.

VI. Conclusion

For different configurations of distributed DESs, this paper has proposed synthesis algorithms, which establish distributed controllers in state-feedback form. The local controllers provided for the subsystems are determined to realize the transfer of the subsystems into goal states from arbitrary inital states, i.e., not tailored only to a specific transfer from one initial into a goal state (as often the case for manually designed sequential controllers). In addition, if the algorithms are run with new goal states ξ^i_F, the adaptation to new specifications is easily achieved. In contrast to most other synthesis algorithms for the DES, the proposed procedures consider costs of state transitions and optimize the behavior of the distributed DES plant. The underlying principle is that of dynamic programming, which is common for systems with continuous-valued states.

A possible alternative approach would be to compose the finite-state machine of the subsystems to a single model, and to run the optimization for it. In contrast, to reduce the computational effort for the optimization, the algorithms advocated in this paper retain the distributed plant structure, and apply the optimization separately to the subsystems. For the considered priority structures of the plant, namely linear and treelike dependencies, the effort can be significantly reduced compared with the approach of parallel composition and optimization of a monolithic model. The reason for the effort reduction is that by working through the chain or tree of subsystems, while starting from those with lowest priority, the one-sided dependencies enable separate computation. In addition, the absence of cycles avoids the reiteration of already computed local controllers.

The local controllers are immediately obtained in decomposed form, i.e., the extraction from centralized controllers is not necessary. The transfer of the state-feedback controllers into the typical standard languages for implementing sequential controllers is a simple task and can be automated. The algebraic formulation of the plant dynamics and the control laws has proved useful in implementing several steps of the synthesis algorithms. One may argue that the various matrices (F^i, K^i, Π^i_{opt}, etc.) quickly grow in size for

larger n_i and m_i. It should be noted, however, that typically many of these matrices are sparse, i.e., particular and efficient representations and algebraic routines for sparse matrices can be used to further speed up the computations.

Current investigations comprise the exploration of efficient synthesis algorithms for distributed structures with bidirectional dependencies between subsystems. In addition, extensions to nondeterministic behavior (including uncontrollable state transitions) are matter of current research.

REFERENCES

[1] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. New York, NY, USA: Springer, 2008.

[2] P. J. G. Ramadge and W. M. Wonham, "The control of discrete event systems," *Proc. IEEE*, vol. 77, no. 1, pp. 81–98, Jan. 1989.

[3] K. Rudie and J. C. Willems, "The computational complexity of decentralized discrete-event control problems," *IEEE Trans. Autom. Control*, vol. 40, no. 7, pp. 1313–1319, Jul. 1995.

[4] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM J. Control Optim.*, vol. 25, no. 1, pp. 206–230, 1987.

[5] F. Lin and W. M. Wonham, "Decentralized supervisory control of discrete-event systems," *Inf. Sci.*, vol. 44, no. 3, pp. 199–224, 1988.

[6] K. Rudie and W. M. Wonham, "Think globally, act locally: Decentralized supervisory control," *IEEE Trans. Autom. Control*, vol. 37, no. 11, pp. 1692–1708, Nov. 1992.

[7] K. Schmidt, T. Moor, and S. Perk, "Nonblocking hierarchical control of decentralized discrete event systems," *IEEE Trans. Autom. Control*, vol. 53, no. 10, pp. 2252–2265, Nov. 2008.

[8] K. C. Wong and W. M. Wonham, "Hierarchical control of discrete-event systems," *Discrete Event Dyn. Syst.*, vol. 6, no. 3, pp. 241–273, 1996.

[9] Y. Brave, "Control of discrete event systems modeled as hierarchical state machines," *IEEE Trans. Autom. Control*, vol. 38, no. 12, pp. 1803–1819, Dec. 1993.

[10] C. Baier and T. Moor, "A hierarchical and modular control architecture for sequential behaviours," *Discrete Event Dyn. Syst.*, vol. 25, no. 1, pp. 95–124, 2015.

[11] H. Zhong and W. M. Wonham, "On the consistency of hierarchical supervision in discrete-event systems," *IEEE Trans. Autom. Control*, vol. 35, no. 10, pp. 1125–1134, Oct. 1990.

[12] G. Barrett and S. Lafortune, "Decentralized supervisory control with communicating controllers," *IEEE Trans. Autom. Control*, vol. 45, no. 9, pp. 1620–1638, Sep. 2000.

[13] A. Mannani and P. Gohari, "Decentralized supervisory control of discrete-event systems over communication networks," *IEEE Trans. Autom. Control*, vol. 53, no. 2, pp. 547–559, Mar. 2008.

[14] M. Heymann, "Concurrency and discrete event control," *IEEE Control Syst. Mag.*, vol. 10, no. 4, pp. 103–112, Jun. 1990.

[15] R. C. Hill and D. M. Tilbury, "Incremental hierarchical construction of modular supervisors for discrete-event systems," *Int. J. Control*, vol. 81, no. 9, pp. 1364–1381, 2008.

[16] M. Fabian and A. Hellgren, "PLC-based implementation of supervisory control for discrete event systems," in *Proc. 37th IEEE Conf. Decision Control*, vol. 3, Dec. 1998, pp. 3305–3310.

[17] K. W. Schmidt and J. E. R. Cury, "Efficient abstractions for the supervisory control of modular discrete event systems," *IEEE Trans. Autom. Control*, vol. 57, no. 12, pp. 3224–3229, Dec. 2012.

[18] R. Kumar and V. K. Garg, "Optimal supervisory control of discrete event dynamical systems," *SIAM J. Control Optim.*, vol. 33, no. 2, pp. 419–439, 1995.

[19] R. Sengupta and S. Lafortune, "An optimal control theory for discrete event systems," *SIAM J. Control Optim.*, vol. 36, no. 2, pp. 488–541, 1998.

[20] L. Grigorov and K. Rudie, "Near-optimal online control of dynamic discrete-event systems," *Discrete Event Dyn. Syst.*, vol. 16, no. 4, pp. 419–449, 2006.

[21] A. Ray, J. Fu, and C. Lagoa, "Optimal supervisory control of finite state automata," *Int. J. Control*, vol. 77, no. 12, pp. 1083–1100, 2004.

[22] W. H. Sadid, S. L. Ricker, and S. Hashtrudi-Zad, "Multiobjective optimization in control with communication for decentralized discrete-event systems," in *Proc. 50th IEEE Int. Conf. Decision Control, Eur. Control Conf.*, Dec. 2011, pp. 372–377.

[23] K. W. Schmidt, "Optimal supervisory control of discrete event systems: Cyclicity and interleaving of tasks," *SIAM J. Control Optim.*, vol. 53, no. 3, pp. 1425–1439, 2015.

[24] M. V. Iordache and P. J. Antsaklis, "Decentralized supervision of Petri nets," *IEEE Trans. Autom. Control*, vol. 51, no. 2, pp. 376–381, Feb. 2006.

[25] J. Ye, Z. Li, and A. Giua, "Decentralized supervision of Petri nets with a coordinator," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 6, pp. 955–966, Jun. 2015.

[26] H. Hu, M. Zhou, Z. Li, and Y. Tang, "An optimization approach to improved Petri net controller design for automated manufacturing systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 10, no. 3, pp. 772–782, Jul. 2013.

[27] H. Hu, C. Chen, R. Su, M. Zhou, and Y. Liu, "Distributed supervisor synthesis for automated manufacturing systems using Petri nets," in *Proc. IEEE Int. Conf. Robot. Autom.*, May/Jun. 2014, pp. 4423–4429.

[28] O. Stursberg, "Hierarchical and distributed discrete event control of manufacturing processes," in *Proc. 17th IEEE Conf. Emerg. Technol. Factory Autom.*, Sep. 2012, pp. 1–8.

[29] C. Hillmann and O. Stursberg, "Algebraic synthesis for online adaptation of dependable discrete control systems," *IFAC Proc. Vol.*, vol. 46, no. 22, pp. 61–66, 2013.

[30] K. M. Passino and P. J. Antsaklis, "On the optimal control of discrete event systems," in *Proc. 28th IEEE Conf. Decision Control*, Dec. 1989, pp. 2713–2718.

[31] C. Hillmann and O. Stursberg, "Decentralized control of distributed discrete event systems with linear dependency structure," in *Proc. IEEE Int. Conf. Autom. Sci. Eng.*, Aug. 2015, pp. 551–557.

[32] R. Bellman, *Dynamic Programming*. Princeton, NJ, USA: Princeton Univ. Press, 1957.

[33] R. Bellman, "On a routing problem," *Quart. Appl. Math.*, vol. 16, no. 1, pp. 87–90, 1958.

[34] A. Mannani, Y. Yang, and P. Gohari, "Distributed extended finite-state machines: Communication and control," in *Proc. 8th Int. Workshop Discrete Event Syst.*, 2006, pp. 161–167.

[35] M. Dogruel and U. Ozguner, "Controllability, reachability, stabilizability and state reduction in automata," in *Proc. IEEE Int. Symp. Intell. Control*, Aug. 1992, pp. 192–197.

[36] H. Hu, M. Zhou, and Y. Liu, "Maximally permissive distributed control of large scale automated manufacturing systems modeled with Petri nets," in *Proc. IEEE Int. Conf. on Autom. Sci. Eng.*, Aug. 2013, pp. 1145–1150.

Olaf Stursberg (M'06) received the Dipl.-Ing. and Dr.-Ing. degrees in engineering from Universität Dortmund, Dortmund, Germany, in 1996 and 2000, respectively.

After being a Postdoc at Carnegie Mellon University in 2001 and 2002, he was appointed as Oberingenieur (similar to Assistant Professor) with Universität Dortmund, from 2002 to 2006. In 2006, he joined Technische Universität München, Munich, Germany, as an Associate Professor of Automation Systems. Since 2009, he has been a Full Professor of Control and System Theory with the Department of Electrical Engineering and Computer Science, Universität Kassel, Kassel, Germany. His current research interests include the control of networked and cyber-physical systems, optimal and predictive control, and the control of hybrid, discrete event, and stochastic processes.

Christian Hillmann received the Dipl.-Ing. degree in mechatronics from Universität Kassel, Kassel, Germany, in 2014.

From 2014 to 2015, he was a Scientific Assistant with the Control and System Theory Group, Department of Electrical Engineering and Computer Science, Universität Kassel. He is now with the Automation Industry in Germany. His current research interests include the area of algorithmic synthesis of controllers for discrete event and cyber-physical systems.