Decentralized Control of Distributed Discrete Event Systems With Linear Dependency Structure

Christian Hillmann* and Olaf Stursberg*

***Abstract*— This paper introduces an approach to synthesizing optimal decentralized controllers for distributed discrete event systems with a linear structure. Any subsystem (except of the last in the structure) demands a sequence of outputs from the next adjacent subsystem, in order to realize its own path into a goal state. This dependency is considered in the synthesis procedure for obtaining a local state-feedback controller for any subsystem. These local controllers for discrete-event systems resemble the typical state-feedback control structure of linear discrete-time continuous-valued systems. Any local controller is computed by algebraic computations, it communicates with controllers of adjacent subsystems, and it aims at transferring the corresponding subsystem into the goal with a minimal sum of transition costs. As is shown also for an example, the computational effort can be significantly reduced compared to the synthesis of a centralized controller.**

I. Introduction

The motivation for the class of distributed discrete event systems (DES) addressed in this paper stems from the structure observed, e.g., in industrial production processes with uni-directional supply schemes. Consider the example of an assembly process consisting of two machines, where the first machine assembles parts which are produced by the second machine. 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. One can easily imagine dependency schemes of similar type which involve more than two production units. The objective of algorithmically synthesizing discrete event controllers which make explicit use of the specific dependency structure is the subject of this work.

With respect to existing work on control synthesis for DES, it seems fair to tag the *supervisory control theory* (SCT) according to [1] as the most established approach. 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 exists, including approaches to decentralized control [2], [3], [4], hierarchical structures [5], [6], [7] including consistency [8], communication aspects [9], [10], [11], concurrency [12], modularity and abstraction [13], and the transformation in PLC programs [14].

* Institute of Control and System Theory, Dept. of Electrical Engineering and Computer Science, University of Kassel (Germany). {hillmann, stursberg}@uni-kassel.de

A second important class of techniques for obtaining decentralized controlles for DES start from models specified as Petri-nets. For example, the work reported in [15], [16], [17], [18] proposes different means to consider distributed structures of the plant, and/or to produce controllers which adhere to principles of decentralization where local controllers are assigned to modules or subsystems of the plant.

In contrast to the afore-mentioned classes of approaches, this paper follows the lines of algebraic controller computation as proposed in [19]. The main idea there is to transfer principles of discrete-time linear time-invariant (LTI) systems to the domain of DES, and in particular to model distributed and hierarchical structures of DES by algebraic descriptions. These ideas were used in [20] to obtain online-reconfigured feedback controllers that account for the occurrence of failures or changing goal specifications. As in [21], the work in [20] 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 approaches, however, were formulated for monolithic systems only, but not for distributed setting of DES.

In this paper, the algebraic computation of optimal state feedback controllers for distributed systems is addressed, and in particular for the linear dependency structures mentioned above. These structures allow computing local controllers of subsystems separately, what may considerably reduce the overall computational effort, compared to the synthesis of centralized controllers. In addition, the scheme naturally leads to a set of local controllers to be implemented on separate hardware. Thus, the outcome of the procedure is compatible to the typical hardware structure of larger processes, i.e. it does not require an aditional step of splitting a large (centralized) controller into local instances assigned to the plant subsystems. Of course, the decentralized controllers have to account for the dependencies among the subsystems, such that appropriate communication between the local controllers has be established.

The paper is organized as follows: after introducing the distributed system structure, the algebraic system representation is described in Sec. II. In Section III, the considered control problem is formulated and the structure of the local control laws is introduced. As a main contribution, Sec. IV proposes an algorithm for separate and sequential synthesis of the subsystem controllers. This part also discusses computational complexity and sketches different extensions. Section V illustrates the procedure by an example from the domain of assembly processes, followed by some conclusions.

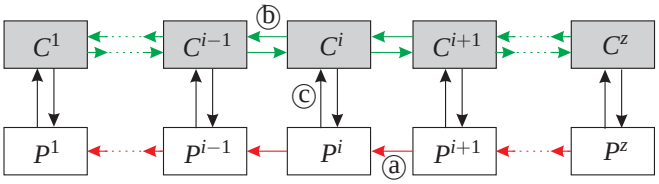

Fig. 1. Distributed system structure consisting of z feedback loops (P^i,C^i) and with linear dependency: ⓐ any subsystem P^i $(i \neq z)$ depends on outputs of P^{i+1}, ⓑ any controller C^i $(i \in \{2,\ldots,z-1\})$ communicates with the neighboring controllers C^{i-1} and C^{i+1}; ⓒ any pair (P^i,C^i)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

II. System Modeling

The subject of this investigation are distributed processes which are suitably modeled by discrete event dynamics and which consist of z subsystems according to $\mathscr{P} = \{P^1,\ldots,P^i,\ldots,P^z\}$. Figure 1 clarifies the interconnection structure of the subsystems: each subsystem P^i forms a control loop together with a local controller C^i in the typical understanding that the current state of P^i triggers a control input of the controller C^i such that a transition of P^i leads to a desired next state. The dependency structure considered here adds the following: (1.) a transition of P^i (for $i \in \{1,\ldots,z-1\}$) may depend on a particular output provided by P^{i+1}, and (2) the controller C^i (for $i \in \{2,\ldots,z-1\}$) 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. The upcoming sections describe in detail how the subsystems are modelled, how costs are introduced, and how the interconnection and communication between subsystems are taken into account during controller synthesis (offline) and controller execution (online).

A. Definition of Subsystems

The discrete event dynamics of any subsystem $P^i \in \mathscr{P}$ is defined as follows:

Def. 1: The plant model $P^i = (T,X^i,U^i,Y^i,W^i,f^i,g^i)$ consists of an ordered domain $T = \{0,1,2,...\}$ with an index $k \in T$ referring to an event time, the finite set of discrete states $\xi^i_k \in X^i = \{1,...,n_i\} \subset \mathbb{N}$, the finite set of discrete inputs $v^i_k \in U^i = \{1,...,m_i\} \subset \mathbb{N}$, the finite set of discrete outputs $y^i_k \in Y^i = \{1,...,q_i\} \subset \mathbb{N}$, a deterministic state transition function $f^i : X^i \times U^i \to X^i$, a dependency matrix $W^i \in \{0,1,...,q_{i+1}\}^{m_i \times n_i}$ between P_i and P_{i+1}, and an output function $g^i : X \to Y^i$ (often referred to as *Moore-type*).

For $i = z$, the dependency matrix is set to a zero matrix $W^i = \{0\}^{m_i \times n_i}$. □

To simplify notation, the sets X^i, U^i, and Y^i are introduced as ordered index sets (which may, of course, enumerate e.g symbolic identifiers). An element $W^i(a,b) = y^{i+1} > 0$ of the dependency matrix W^i encodes that the subsystem P^{i+1} must provide the output $y^{i+1} \in Y^{i+1}$ to enable in P^i the discrete transition from the state $b \in X^i$ under the effect of the input $a \in U^i$. An entry $W^i(a,b) = 0$ means, however, that the transition from the state $b \in X^i$ upon the input $a \in U^i$ is independent of the output of the subsystem P^{i+1}.

Def. 2: For an initial state $\xi^i_0 \in X^i$ and an input sequence $\phi^i_u := \{v^i_0, v^i_1, ...\}$ with $v^i_k \in U^i$ and $k \in T$, the elements of an admissible run $\phi^i_x := \{\xi^i_0, \xi^i_1, ...\}$ and a corresponding output sequence $\phi^i_y := \{y^i_0, y^i_1, ...\}$ of P^i according to Def. 1 follow from:

$$\xi^i_{k+1} = \begin{cases} f^i(\xi^i_k, v^i_k), & \text{if } W^i(v^i_k, \xi^i_k) \in \{0, \xi^{i+1}_k\} \\ \xi^i_k, & \text{else} \end{cases} \quad (1)$$

$$y_{k+1} = g^i(\xi^i_{k+1}) \quad (2)$$

□

The else-statement in the first assignment refers to the case that no input triggers a state transition.

Since Sec. IV proposes a method for synthesizing state feedback controllers and in order to simplify notation, the outputs are ommitted in the sequel according to the following assumption.

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

In consequence, we have $X^i = Y^i$, i.e. any occurence of y^i_k in Def.s 1 and 2 can be expressed in terms of x^i_k. In particular, the dependency of P^i from P^{i+1} as modeled by W^i can be understood now such that a transition of P^i requires P^{i+1} to be in a specific state x^{i+1}_k. In context of the manufacturing process mentioned in the beginning, it seems justifiable to assume that a part being required for assembly is eventually available.

Assumption 2: In addition, we assume that the dynamics of all P^i, $i \in \{1,\ldots,z\}$ operate on the same time domain $(k \in T)$, and that all P^i iterate their states synchronously. □

Given the typical cycle time of control hardware and the order of magnitudes larger average time between events, this assumption is certainly justifiable. (The understanding is that k enumerates the actual state transitions, not the hardware cycles.)

B. Algebraic Formulation

The control synthesis to be described below is structurally similar to that of (optimal) state feedback controllers for linear discrete-time continuous-valued systems. It thus makes sense to represent the dynamics of P^i in a format amenable to algebraic matrix operations, compare to the scheme proposed in [20] for monolithic DES.

Def. 3: For P^i with state set X^i as introduced in Def. 1, let the state vector $x^i_k \in \{0,1\}^{n_i \times 1}$ identify the plant state according to: $x^i_{k,j} = 1$ if $\xi^i_k = j$ is the active state in k, and $x^i_{k,j} = 0$ otherwise. For any input $l \in U^i$, a state transition matrix $F^i_l \in \{0,1\}^{n_i \times n_i}$ is defined such that for any pair $h,j \in X^i$ applies: $F^i_l(j,h) = 1$ if $j = f^i(h,l)$, and $F^i_l(j,h) = 0$ otherwise. It is required that $F^i_l(j,j) = 1$ if $F_l(p,j) = 0$ for all $p \neq j$, $p \in \{1,\ldots,n_i\}$ (meaning that the state with index j has a self-loop, if no outgoing transition exists for the input $l \in U^i$). □

The interpretation of $F_l^i(j,h)=1$ obviously is that P^i can transition from state $\xi_k^i=h$ into state $\xi_k^i=j$ upon event l, if in addition the dependency condition encoded by W^i is satisfied. A run of P^i is now defined in algebraic form as follows:

Def. 4: For P^i, let $\mathscr{F}^i=\{F_1^i,\ldots,F_{m_i}^i\}$ denote the set of state transition matrices as introduced before. Given an initial vector $x_k^i\in\{0,1\}^{n_i\times 1}$ with $\sum_{j=1}^{n_i}x_{k,j}^i=1$ and an input sequence $\phi_u=(v_0^i,v_1^i,v_2^i,\ldots)$ as in Def. 2, an admissible run $\phi_x^i=(x_0^i,x_1^i,x_2^i,\ldots)$ of P^i over T satisfies:

$$x_{k+1}^i=\begin{cases}F_j^i\cdot x_k^i, & \text{if } v_k^i=j \text{ and } W^i(v_k^i,\xi_k^i)\in\{0,\xi_k^{i+1}\}\\ x_k^i, & \text{else}\end{cases} \tag{3}$$

□

For the computation of the control law for P^i, it is necessary that P^{i+1} can indeed deliver the output signals to P^i that are encoded in W^i. Thus, we require that any discrete state $\xi^i\in X^i$ can be transferred into any other state $\hat{\xi}^i\in X^i$ by at least one finite input sequence. As a preparation for stating this assumption formally, a *reachability matrix* R^i, is introduced as:

$$R^i:=\sum_{p=1}^{n_i}\Big(\sum_{l=1}^{m_i}F_l^i\Big)^p. \tag{4}$$

It encodes (for the case that the dependency conditions are satisfied) that for any pair of states taken from X^i, a sequence of inputs exists which transfers the plant from the first into the second state. Similarly as introduced in [20] for monolithic systems, a subsystem P^i can then be classified as *completely controllable*, if $R(q,s)>0$ for any pair $q,s\in\{1,...,n_i\}$.

Assumption 3: Any subsystem $P^i\in\mathscr{P}$ is completely controllable. □

This assumption is justifiable in the sense that a process in which a unit cannot deliver the output required in another depending unit has to be characterized as not properly engineered.

C. Local transition costs

The control task to be precisely defined in the next section considers a performance measure in the sense that any subsystem should be transferred into a goal state with minimum costs. For this reason, local costs $\pi(\xi_k^i,\xi_{k+1}^i,v_k^i)$ are assigned to any transition $f^i(\xi_k^i,v_k^i)=\xi_{k+1}^i$. While different interpretations of such costs may be reasonable for a given application, the most obvious ones are the time or the control effort in terms of energy (or a different consumed resource) which is necessary to transfer the system from ξ_k^i to ξ_{k+1}^i by the use of the input v_k^i.

Def. 5: Given P^i, a *local cost matrix* $\Pi_j^i\in\mathbb{R}_{\geq 0}^{n_i\times n_i}$ is defined for any $j\in U^i$, such that (i) $\Pi_j^i(q,p)=\pi(p,q,j)$, (ii) $\Pi_j^i(q,p)=\infty$, if the transition from $\xi_k^i=p$ to $\xi_{k+1}^i=q$ is infeasible with $v^i=j$ (i.e. $F_j^i(q,p)=0$), and (iii) $\Pi_j^i(p,p)=0$ for all $p\in X^i$, $j\in U^i$ (i.e. self-loops do not incur costs). Furthermore, the matrix Π_{opt}^i denotes the *minimal local transition costs* for P^i, where the entry $\Pi_{opt}^i(q,p)$ encodes the minimally possible local costs for the transfer of P^i from state $\xi^i=p$ to $\xi^i=q$. □

As will become apparent in the next section, the realization of paths with the costs in the matrix Π_{opt}^i is the very objective of the feedback controller.

III. Control Task

Given the dependency structure of $\mathscr{P}$ as defined before, the transitions of subsystem 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, we thus focus in the following on just two subsystems indicated by $i\equiv A$ and $i+1\equiv B$. The section IV-C will later show, however, that the extension to larger dependency chains is straightforward.

Problem 1: Given a pair (ξ_F^A,ξ_F^B) of goal states of the two subsystems P^A and P^B, the control task is to compute for both subsystems local feedback control laws which generate for **any** initialization $\xi_0^A\in X^A$ and $\xi_0^B\in X^B$ input sequences $\phi_u^A=(v_0^A,\ldots,v_{d^A-1})$ and $\phi_u^B=(v_0^B,\ldots,v_{d^B-1})$ which lead to runs $\phi_x^A=(\xi_0^A,\ldots,\xi_{d^A}^A)$ with $\xi_{d^A}^A=\xi_F^A$ and $\phi_x^B=(\xi_0^B,\ldots,\xi_{d^B}^B)$ with $\xi_{d^B}^B=\xi_F^B$ such that the global path costs:

$$J_{global}=\sum_{i=1}^{d^A}\Pi_{v_{i-1}^A}(\xi_i^A,\xi_{i-1}^A)+\sum_{i=1}^{d^B}\Pi_{v_{i-1}^B}(\xi_i^B,\xi_{i-1}^B) \tag{5}$$

are minimal. □

Intuitively speaking, the objective is to equip both subsystems P^A and P^B with local controllers C^A and C^B, which establish the transfer from an arbitrarily chosen local initial state into the respective local goal state such that the sum of the transfer costs for P^A and P^B is as small as possible.

Before the particular structure of the local controllers is introduced, it is shown that the combination of goal states ξ_F^A and ξ_F^B is reachable from an arbitrary pair of initial states of the two subsystems.

Theorem 1: The system $\mathscr{P}=\{P^A,P^B\}$ with dependency structure as introduced before is completely controllable if both of the subsystems P^A and P^B on their own are completely controllable according to Assumption 3. □

Proof 1: Since the transitions of subsystem B are independent of the current state of subsystem P^A and subsystem P^B is completely controllable, a sequence ϕ_u^B of inputs exists to transfer B from an arbitrary initial state ξ_0^B into ξ_F^B. This also means that P^B can deliver any arbitrary output sequence ϕ_y^B (and thus run ϕ_x^B) to subsystem P^A, i.e. any condition formulated for P^A in terms of the dependency matrix W^A can be satisfied by P^B. Since P^A itself is completely controllable as well, a sequence of inputs ϕ_u^A exists which transfers P^A into an arbitrary goal state ξ_F^A. □

The type of state feedback control law to be determined within this paper is chosen to be:

$$v_k^A=u^A\cdot K^A(\xi_k^B)\cdot x_k^A\ \in U^A \tag{6}$$

$$v_k^B=u^B\cdot K^B\cdot x_k^B\ \in U^B \tag{7}$$

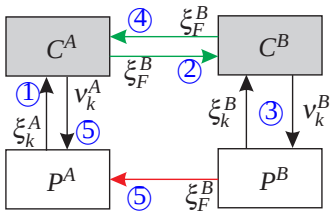

Fig. 2. Online-execution for one state transition of P^A including the provision of ξ_k^B by P^B. The numbers indicate the order of information processing.

where u^i is the row vector of all local input indices in U^i (in ascending order) and $K^i \in \{0,1\}^{m_i \times n_i}$ is a local controller matrix, compare to [20]. The interpretation of these laws is (identically to state feedback in continuous systems) that the multiplication of a feedback matrix $u^i \cdot K^i$ with the current state x_k^i selects the input value from the set U^i. The multiplication is thus equivalent to choosing one of the possible events in U^i to trigger the desired transition. It is important to note here that, in general, the input generated by the control law of a certain subsystem (except of the one with highest index) depends on the current plant state of the own subsystem and the current plant state of the subsystem with the next higher index. For the specific case of P^A und P^B, this means that the controller matrix $K^A(\xi_k^B)$ depends on the current plant state ξ_k^B of subsystem B, while K^B is independent of the state of another subsystem. For this reason, K^A actually consists of n_B single controller matrices. For simplifying notation, the set of all of these matrices are referenced by the expression K^A, while the one specific controller matrix in K^A corresponding to a particular ξ^B is denoted by $K^A(\xi^B)$.

The online-execution of the decentrally controlled distributed system takes place as visualized in Fig. 2: When C^A receives the information from p^A that state ξ_k^A is reached ①, C^A sends the *request* to C^B that P^B has to reach ξ_F^B (as temporary goal state of P^B) ②. This state is encoded in K^A in order to realize a cost-optimal path of P^A into its goal state x_F^A. Then, C^B realizes a path of P^B into ξ_F^B (possibly several steps in ③). The controller C^B communicates to C^A the information that the requested state is reached ④. Eventually, the control input v_k^A supplied by C^A together with ξ_F^B send by P^B triggers the state transition in P^A ⑤.

Figure 3 illustrates the flow of information for the offline-computation of the control laws. In the first step ①, the matrix K^B of controller C^B is synthesized, and Π_{opt}^B contains

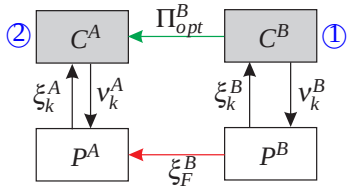

Fig. 3. Information flow within the offline synthesis of K^A and K^B.

the optimal transfer costs (according to Def. 5) for any pair of initial and goal state in X^B. Since the synthesis of K^B does not depend on any other subsystem, the online-procedure presented in [20] can be used for the transfer in a temporary goal state ξ_F^B. The more intricate task of synthesizing K^A, see step ② in Fig. 3, which builds upon the matrix Π_{opt}^B, is subject of the next section.

IV. Controller Synthesis

This section describes the algorithm to compute the controller matrices by which a subsystem reaches a goal state with minimal costs. In addition, it is shown that the controller matrices can be computed separately if the matrix of optimal transition costs from the adjacent subsystem with the next higher index is available, i.e. if Π_{opt}^B is known for computing K^A. The positive effect on the computational effort necessary for decentralized controller synthesis is discussed in the last part of the section.

A. Synthesis algorithm for two subsystems

Algorithm 1 for computing the controller matrix K^A follows the dynamic programming principle [22]. It can be interpreted as a version of the Bellman-Ford algorithm (see [23]) with reversed transition through the graph of P^A with additional consideration of the dependency on the state of the subsystem P^B.

As input data, Alg. 1 requires the transition cost matrices Π_v^A of P^A for all $v \in U^A$, the dependency matrix W^A, the matrix Π_{opt}^B of optimal path costs of subsystem B, and a pair of goal states ξ_F^A and ξ_F^B. The algorithm starts with initializing the entries of the controller matrices $K^A(\xi^B)$ to 0 for all $\xi^B \in X^B$, and the cost matrix H_0 is initialized to ∞ for any entry. In the course of the algorithm, the matrix $H_i \in \mathbb{R}_{\geq 0}^{n_A \times n_B}$ is iteratively updated to contain upon termination in the element $H_i(j,l)$ the minimized costs for transferring $\mathscr{P}$ from the state pair $j \in X^A$ and $l \in X^B$ to the specified pair of goal states. The transfer requires at most $i \cdot n_B$ steps since, in the worst case, P^A has to wait for n_B steps until the relevant dependency condition of the respective step is satisfied. This is due to the fact that the number of transitions to transfer the completely controllable subsystem P^B between two arbitrary states is at most n_B. The initialization of the cost matrix H_0 is modified to zero for the element which refers to the pair of goal states (ξ_F^A, ξ_F^B).

In the first step of the main while-loop of Alg. 1, the cost matrix H_i is initialized to the result of the previous iteration. In any iteration, the objective is to check if a cost reduction in H_i is possible for any combination of discrete states $\{1,\ldots,n_A\} \in X^A$, discrete inputs $\{1,\ldots,m_A\} \in U^A$, and discrete states $\{1,\ldots,n_B\} \in X^B$. In detail, the costs for a transition of P^A from state k to state j triggered by the input m are computed and compared to the best costs computed so far. For this transition, two cases have to be considered: P^A requires that P^B is currently in a specific discrete state ($W^A(m,k) > 0$), or the transition from k to j by input m is independent of the current state of P^B ($W^A(m,k) = 0$).

Data: Transition cost matrices Π_v^A, dependency matrix W^A, the matrix Π_{opt}^B, goal states ξ_F^A and ξ_F^B
Result: Control matrices $K^A(\xi^B)$ to transfer P^A and P^B to the goal states with minimized total costs

$K^A(\xi^B) = 0^{m_A \times n_A}$ for all $\xi^B = 1,...,n_B$;
$H_0 = \infty^{n_A \times n_B}$;
$H_0(\xi_F^A, \xi_F^B) = 0$;
$control = 0$;
$i = 1$;
while $control = 0$ **do**
  $H_i = H_{i-1}$;
  for $k = 1 : n_A$ **do**
    for $m = 1 : m_A$ **do**
      for $l = 1 : n_B$ **do**
        $j = (1 : n_A) \cdot F_m^A(:,k)$;
        if $W^A(m,k) > 0$ **then**
          $r = W^A(m,k)$;
          if $H_{i-1}(j,r) + \Pi_m^A(j,k) + \Pi_{opt}^B(r,l) <$ $H_i(k,l)$ **then**
            $H_i(k,l) =$ $H_{i-1}(j,r) + \Pi_m^A(j,k) + \Pi_{opt}^B(r,l)$;
            $K^A(l)(:,k) = 0$;
            $K^A(l)(m,k) = 1$;
          end
        else
          if $H_{i-1}(j,l) + \Pi_m^A(j,k) < H_i(k,l)$ **then**
            $H_i(k,l) = H_{i-1}(j,l) + \Pi_m^A(j,k)$;
            $K^A(l)(:,k) = 0$;
            $K^A(l)(m,k) = 1$;
          end
        end
      end
    end
  end
  if $H_i = H_{i-1}$ **then**
    $control = 1$;
  else
    $i = i + 1$;
  end
end

Algorithm 1: Computation of $K^A(\xi_k^B)$.

For the first case, the discrete state $r = W^A(m,k) \in X^B$ of P^B (as required for the transition of P^A) is determined. Then, the costs for the considered transition are computed by $\Pi_m^A(j,k) + \Pi_{opt}^B(r,l)$. This value is composed of the local transition costs $\Pi_m^A(j,k)$ and the optimal transition costs $\Pi_{opt}^B(r,l)$ assigned to a transition necessary to satisfy the dependency condition of the considered transition of P^A. If this transition leads to a cost reduction for the momentarily investigated combination of states k^A and l^B, i.e. if:

$$H_{i-1}(j,r) + \Pi_m^A(j,k) + \Pi_{opt}^B(r,l) < H_i(k,l) \qquad (8)$$

applies, the value of $H_i(k,l)$ is updated to the lower value. In this case, the entries of the controller matrix are updated by replacing a possibly existing non-zero entry of K^A for the currently investigated state by zero, and by setting the entry $K^A(l)(m,k)$, which corresponds to the momentarily investigated pair of state and input, to 1.

The procedure for the case of $W^A(m,k) = 0$ is very similar: since $W^A(m,k) = 0$ encodes that the currently investigated transition of P^A is independent of the current state of P^B, only the local transition costs of P^A are determined, and H_i is checked for a cost reduction. If the costs are lowered, the controller matrix is updated.

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

The controller matrices K^B for the subsystem P^B can be computed by a simplified variant of Alg. 1: since no dependencies to another subsystem are present, the dependency matrix W^B is a zero-matrix and H_i is one-dimensional. This means that the case $W^B(m,k) > 0$ never occurs, and only those for-loops are required, which correspond to the state and input sets of the subsystem P^B. This variant of Alg. 1 leads to the simplified synthesis procedure presented in [20].

The *local* controller matrices K^A and K^B determined by these algorithms transfer the system from *arbitrary* initial states ξ_0^A and ξ_0^B into the specified pair of goal states ξ_F^A and ξ_F^B with minimal *global* costs J_{global} according to (5).

B. Effort estimation of the controller synthesis procedure

The computation of the controller matrices K^A and K^B by the method introduced in this paper consists of two parts. The first part includes to compute Π_{opt}^B and K^B. The computational effort for this task is of the order $\mathcal{O}(n_B^2 m_B + n_B^3)$. The computation of K^A represents the second part, and its computational effort grows with $\mathcal{O}(n_A^2 m_A n_B)$. Hence, the overall computational effort is of the order $\mathcal{O}(n_B^2 m_B + n_A^2 m_A n_B + n_B^3)$.

A natural algorithmic competitor of the procedure proposed here is a centralized design, i.e. to compute the parallel composition of P^A and P^B, and to apply the procedure as mentioned above for P^B to the composition. The computation of local control laws for the composed system consists of 4 steps: First, the composition and the corresponding transition cost matrices Π_i^P have to be determined, leading to an effort of the order $\mathcal{O}(n_A n_B m_A m_B)$. This step is followed by the determination of the matrices of optimal one-step transition costs with $\mathcal{O}(n_A^2 n_B^2 m_A m_B)$. The third step computes the controller matrix K for the composed system with an effort of $\mathcal{O}(n_A^2 n_B^2)$ (according to the procedure in [20]). Finally, the controller matrices K^A and K^B have to extracted from K, what incurs computational costs of $\mathcal{O}(n_A n_B)$. In total, the effort for this alternative to compute the local control laws can be summarized to be of order $\mathcal{O}(n_A^2 n_B^2 m_A m_B)$.

C. Extensions of Algorithm 1

The algorithm 1 is formulated for the case of one specified pair of goal states ξ_F^A and ξ_F^B. However, the algorithm succeeds also in computing the controller matrix K^A for arbitrary sets of goal states by setting the value 0 to all entries of H_0. For the example of a pair $\xi_{F,1}^A$ and $\xi_{F,1}^B$ of goal states as well as a pair $\xi_{F,2}^A$ and $\xi_{F,2}^B$ respectively, the

entries $H_0(\xi^A_{F,1}, \xi^B_{F,1})$ and $H_0(\xi^A_{F,2}, \xi^B_{F,2})$ have to be set to zero, while the rest remains the same.

As already mentioned, the proposed synthesis procedure is also applicable for systems $\mathscr{P}$ with more than $z = 2$ subsystems, but the computations require slight modifications: While for $z = 2$, the computation of K^A requires the matrix Π^B_{opt}, the case for $z = 3$ with $\mathscr{P} = \{P^A, P^B, P^C\}$ requires a matrix $\Pi^{B,C}_{opt}$ when determining K^A. The matrix $\Pi^{B,C}_{opt}$ contains the optimal path costs for any combination of initial states ξ^B_0, ξ^C_0 and goal states ξ^B_F, ξ^C_F with $\xi^B_0, \xi^C_F \in X^B$ and $\xi^C_0, \xi^C_F \in X^C$. In addition, the controller matrix K^A depends on the current state ξ^B_k of P^B and the current state ξ^C_k of P^C. Thus, the matrix $H_0 \in \mathbb{R}^{n_A \times n_B \times n_C}$ has three dimensions, such that the algorithm has to comprise a further for-loop over the states in $X^C = \{1^C, ..., n^C_C\}$.

V. Illustrative Example

The method introduced before is now illustrated by application to a relatively small section of a larger manufacturing process. It consists of two linearly dependent machines, where P^B represents a bending machine which can produce parts of two different shapes. Fig. 4 shows the transition graph of P^B containing the discrete states, discrete inputs, and costs of the transitions representing the bending process. Starting from an initial state $\xi^B = 1^B$, two different shapes (represented by $\xi = 2^B$ respectively $\xi = 3^B$) can be produced. The notation of the transition from $\xi^B = 1^B$ to $\xi^B = 2^B$ encodes that this transition is triggered by the local input $v^B = 1^B$ 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 varnishing process within P^B leads to the final products: $\xi^B = 4^B$ represents a red product, and $\xi^B = 5^B$ represents a blue product.

The other machine (modeled as P^A and supplied by P^B) mounts the bent parts and a base plate to one new product according to two different specifications: The first end product consists of two blue components, while the second one consists of two blue components and one red component. Fig. 5 shows the transition graph corresponding to P^A, containing the discrete states, discrete inputs, costs of transitions, and the dependency conditions. The mounting process can be understood as follows: From the initial state

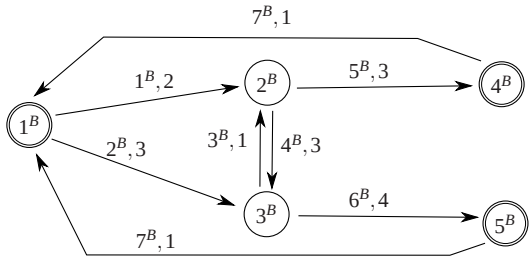

Fig. 4. Graph of the bending machine P^B. The transitions are labelled by the discrete input v^B_k, and the local transition costs $\pi(\xi^B_k, \xi^B_{k+1}, v^B_k)$. No self-loop transitions are shown.

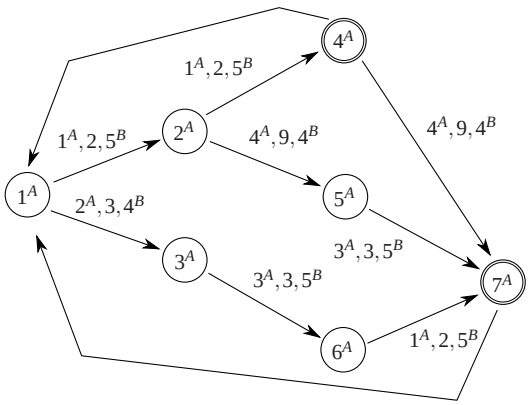

Fig. 5. Graph of the mounting machine P^A. The transition labeling includes the discrete input v^A_k, the transition costs $\pi(\xi^A_k, \xi^A_{k+1}, v^A_k)$, and the entry of the dependency matrix W^A. For the sake of clarity, self-loop transitions are not shown.

$\xi^A = 1^A$, a bent blue product ($\xi^A = 2^A$) or a bent red product ($\xi^A = 3^A$) is mounted to the base plate.

Since no buffer is present in between the machines P^A and P^B, it is important that machine B produces a part only if this is required currently by machine P^A. The transitions are here denoted such that, e.g., the transition from $\xi^A = 1^A$ to $\xi^A = 2^A$ encodes that this transition is triggered by the local input $v^A = 1^A$, entails costs of 2, and requires that machine P^B is currently in state 5^B. All other transitions in this graph either correspond to mounting a red product or a blue product. It is important to note that mounting a red product requires another tool than for a blue product. As the tool change entails additional costs, the transition costs from 6^A to 7^A (which is structurally identical to the transition from 5^A to 7^A) are higher. Note that all inputs which do not change the discrete state have zero costs.

Based on the transition graphs, the dependency matrix W^A and the matrix Π^B_{opt} of optimal path costs are computed:

$$W^A = \begin{bmatrix} 5 & 5 & 0 & 0 & 0 & 5 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 & 5 & 0 & 0 \\ 0 & 4 & 0 & 4 & 0 & 0 & 0 \end{bmatrix}, \quad \Pi^B_{opt} = \begin{bmatrix} 0 & 4 & 5 & 1 & 1 \\ 2 & 0 & 1 & 3 & 3 \\ 3 & 3 & 0 & 4 & 4 \\ 5 & 3 & 4 & 0 & 6 \\ 7 & 7 & 4 & 8 & 0 \end{bmatrix}$$

The controller matrices of machine P^B are computed, and obtained for the example of a goal state $\xi^B_F = \xi^B_4$ to:

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

Subsequently, Alg. 1 can be used to compute the controller matrices $K^A(\xi^B_k)$ for the pair of reference states $\xi^A_F = 7^A$ and

$\xi_F^B = 1^B$. For the example of $\xi_k^B = 2^B$, the result is

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

Now consider that the current states are $\xi_k^A = 2^A$ and $\xi_k^B = 2^B$. The path with minimal local costs for P^A is $2^A \to 4^A \to 7^A$. Nevertheless, the controller C^A chooses the input $v_k^A = 4^A$, which corresponds to the path $2^A \to 5^A (\to 7^A)$. The reason for this choice is that the incurred local costs for the corresponding path of P^B are lower as the contribution for the path $2^A \to 5^A \to 7^A$ of subsystem P^A.

The time for the computation of this control law with an implementation of Algorithm 1 in Matlab is 5 milliseconds (Intel® Core^{TM} i5 CPU @ 2.67 GHz x 4). For comparison, the solution with parallel composition of P^A and P^B and subsequent computation of a centralized controller (as sketched in Sec. IV-B) requires 0.31 seconds for the same example, i.e. the effort is by a factor of 60 higher already for this small example when using a centralized design.

VI. Conclusion

The paper has proposed a method for computing local controller matrices for DES with linear dependency structure. The local controller matrices generate discrete inputs such that a global cost criterion is minimized. The sequential computation of the local conrollers enabled by the dependency structure leads to a significant reduction of the computational effort compared to parallel composition and computation of a centralized control law (which may be implemented in decentralized fashion afterwards). This is shown exemplarily for the computation of a control law for a structure consisting of two subsystems P^A and P^B. The reason for the significant effort reduction is the separation of tasks, since the controller K^B can be determined independently of P^A. As a further result of the computation of the local controller matrix for P^B, the matrix Π_{opt}^B is generated. This matrix enables the controller of P^A to compute local controller matrices without explicitely considering the dynamical processes of P^B. The main advantage is that the number m_B of possible discrete input values of P^B is only relevant for the local synthesis of K^B. For Alg. 1, the number m_B is not relevant, i.e. the algorithms for computing K^A and K^B are less dependent on each other as in centralized computation.

Topics of current investigations are the extensions to tree-like dependency structures as well as the use for bi-directional interconnections of subsystems.

Acknowledgements

The authors gratefully acknowledge partial financial support by the European Commission through the project UnCoVerCPS under grant number 643921.

References

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

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

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

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

[5] K. Wong and W. Wonham, "Hierarchical control of discrete-event systems," *Discrete Event Dynamic Systems*, vol. 6, no. 3, pp. 241–273, 1996.

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

[7] C. Baier and T. Moor, "A hierarchical and modular control architecture for sequential behaviours," *Discrete Event Dynamic Systems*, pp. 1–30, 2014.

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

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

[10] A. Mannani, Y. Amin, and P. Gohari, "Distributed extended finite-state machines: communication and control," in *8th Int. Workshop on Discrete Event Systems*, 2006, pp. 161–167.

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

[12] M. Heymann, "Concurrency and discrete event control," *IEEE Control Systems Magazine*, vol. 10, no. 4, pp. 103–112, 1990.

[13] R. Hill and D. Tilbury, "Modular supervisory control of discrete-event systems with abstraction and incremental hierarchical construction," in *8th Int Workshop on Discrete Event Systems*, 2006, pp. 399–406.

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

[15] M. Iordache and P. Antsaklis, "Decentralized control of petri nets with constraint transformations," in *American Control Conf.*, 2003, pp. 314–319.

[16] H. Hu, M. Zhou, and Y. Liu, "maximally permissive distributed control of Petri net modeling automated manufacturing systems," in *IEEE Int. Conf. on Automation Science and Engineering*, 2013, pp. 1151–1156.

[17] H. Hu, C. Chen, R. Su, M. Zhou, and Y. Liu, "Distributed supervisor synthesis for automated manufacturing systems using Petri nets," in *IEEE Int. Conf. on Robotics and Automation*, 2014, pp. 4423–4429.

[18] J. Ye, Z. Li, and A. Giua, "Decentralized supervision of petri nets with a coordinator," *IEEE Trans. on Systems, Man and Cybernetics: Systems*, vol. 45, no. 6, pp. 955–966, 2015.

[19] O. Stursberg, "Hierarchical and distributed discrete event control of manufacturing processes," in *17th IEEE Conf. on Emerging Technologies & Factory Automation*, 2012, pp. 1–8.

[20] C. Hillmann and O. Stursberg, "Algebraic synthesis for online adaptation of dependable discrete control systems," in *Dependable Control of Discrete Systems*, vol. 4, no. 1, 2013, pp. 61–66.

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

[22] R. Bellman, *Dynamic Programming*. Princeton University Press, 1957.

[23] ——, "On a routing problem," *Quarterly of Applied Mathematics*, vol. 16, no. 1, pp. 87–90, 1958.

[24] M. Dogruel and U. Ozguner, "Controllability, reachability, stabilizability and state reduction in automata," in *IEEE Int. Symp. on Intelligent Control*, 1992, pp. 192–197.