A New DES Control Synthesis Approach Based on Structural Model Properties

Hans-Christian Lapp and Hans-Michael Hanisch, *Senior Member, IEEE*

***Abstract*—This work proposes a novel approach for controller synthesis on the shop-floor level for discrete-event systems. The approach is based on a modular modeling formalism and structural properties of models of uncontrolled plant behavior that are designed using this modeling formalism. The approach takes advantage of the transition invariants of the underlying Petri net components of the model and uses the event interconnections of transitions that are part of the transition invariants. The result is the so-called transition invariant graph that represents a new structural property of the modeled system. The synthesis procedure takes models of the uncontrolled plant behavior and formal specifications of the desired cyclic plant behavior under control and forbidden states as well. From these ingredients, the transition invariant graph is computed from which the admissible trajectories are extracted. This is performed by partial reachability analysis. An example taken from a lab-scaled manufacturing system illustrates the methodology and shows the application. The complexity of the synthesis procedure is significantly reduced. That makes it feasible to be used for even larger systems of real industrial scale.**

***Index Terms*—Control synthesis, discrete-event systems, invariants, net condition/event systems.**

I. Introduction

THE goal of control or supervisory control design is to realize desired, or avoid undesired, plant behavior. Since Ramadge and Wonham [1] introduced the Supervisory Control Theory (SCT), much research has been done regarding supervisor control design for discrete event systems (DES). The framework of Ramadge and Wonham uses finite state machines to model the DES and the supervisor.

The basic SCT synthesizes supervisors on a higher control level, assuming that there are underlying controllers on the shop-floor level that perform the control tasks of the single components. These supervisors are able to prevent some events in the DES, but they cannot force them [2]. The synthesis algorithms need correct models of system and specification which is not easy to capture in industrial context [3], [4]. A usual solution to these remarks is a high abstraction level of the supervisor. This often leads to the introduction of extra control instances [2], [3] to reach nondeterminism regarding the implementation and which are able to force events in the system. Resulting synthesized solutions are often hard to implement, e.g., on a PLC [3]. Hence, some difficulties became apparent, concerning the modeling of complex systems and an appropriate abstraction level of the DES and supervisor modeling, especially for supervisory control for low abstraction levels.

To handle this, extensions of SCT have been proposed for control synthesis [2]–[6].

The approach presented in this contribution is intended for automation on the shop-floor level. This means a low abstraction level, respectively, a high level of detail, even for basic plant components. Therefore, the background of the approach presented in this contribution is the synthesis of distributed discrete process control on shop-floor level, which was introduced in [7].

Disregarding the difference in between supervisor synthesis and control synthesis, a general challenge is the raising complexity of DES modeling and the synthesis procedures corresponding to an increasing complexity regarding real-scale DES.

Because the main focus of this contribution is in reducing the complexity of the supposed synthesis procedure, the following excerpt of related work is not straightforward classified into supervisor and control synthesis. Instead, it should give an overview over the methodologies used in supervisor and control synthesis with attention to the used modeling formalisms.

The approach of Feng *et al.* [8] is taking into account the modular component-based product structure of manufacturing systems to derive a hierarchy of decentralized supervisors and coordinators.

Charbonnier *et al.* [2] introduced a supervised control concept and use the tool Grafcet to bear down the difficulties coming up with basic SCT and finite automata formalism.

Gouyon *et al.* [4] are taken into account the structure of the DES for their approach for modular control synthesis. They combine the SCT formal framework with automation engineering methods based on the re-use of control objects.

The benefits in using Petri nets (PNs) combined or instead of automata are an implicit state description and a more compact modeling of concurrently occurring events. Therefore, many approaches utilize PNs for DES modeling and supervision or controller design; a survey is given by Holloway *et al.* [9]. The main drawback of PNs are possibly large reachable state spaces, which cause raising complexity.

Uzam and Wonham [10] introduced a hybrid approach. They coupled Ramadge and Wohnham supervisors (automata) to DES

Manuscript received December 15, 2011; revised May 21, 2012, August 01, 2012; accepted August 20, 2012. Date of publication August 27, 2012; date of current version October 14, 2013. This work was supported by the Deutsche Forschungsgemeinschaft (DFG) under Grant HA 1886/17. Paper no. TII-11-1013.

The authors are with the Institute of Computer Science, Martin-Luther University of Halle-Wittenberg, 06099 Halle, Germany (e-mail: hans-christian.lapp@informatik.uni-halle.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/TII.2012.2215615

modeled with PN to close the gap in between and to benefit from the advantages of both formalisms.

Iordache and Antsaklis gave a survey [11] over supervision based on place invariants (SBPI). There, structural properties of PN, namely place invariants [12], and the compact system representation of PN as matrices are the starting points for a methodology to avoid complete reachable state-space computation and to efficiently calculate a supervisory control. Flochová [6] also takes advantage of this PN structure-based methodology to synthesize and implement control and supervisory control in an IEC 61131-3 compliant environment.

Li and Zhou [13] determine liveness-enforcing supervisors based on structural properties called siphons and mixed integer programming. Therefore, only siphons which need to be monitored are considered.

Another structure-based approach was introduced by Missal and Hanisch [14].

They analyze the pre-regions of forbidden states in plant model's state space, to synthesize distributed control. This work takes advantage of the modular Net Condition/Event Systems, which have extensions of PN as building blocks.

For the approach of Vasiliu *et al.* [15], an off-line calculation of the reachability graph for the PN plant model is necessary. However, the introduced concept of over-states constitutes a simplification, which allows an efficient feedback controller synthesis.

In summary, reducing the complexity of supervisor or control design procedure is done, amongst others, via model reduction [10], model abstraction [5], [8], utilizing model structure properties and mathematical model representation [6], [11], [13], model simplifications [15], or taking into account the structure of the DES model [4], [8], [16].

The specification of undesired plant behavior in terms of forbidden states is also common to the presented approaches, excepting SCT. In the case of PN, these specifications can be reduced to a set of inequalities [6], [11].

The introduced approach differs from the presented related research works in two points. First, the specification is basically given as the desired cyclic process behavior of the plant, contrary to a forbidden behavior specification. Second, the introduced structural analysis methodology, which is the focus of this paper, incorporates the advantages of a hierarchical modular modeling formalism and the application of known PN properties in a simple comprehensible way. This makes the approach easy to understand from the automation engineering point of view as well as from theoretical point of view. As a result, the complexity of the supposed synthesis procedure is reduced significantly. Fig. 1 shows the scheme of the supposed synthesis procedure. The structural analysis approach presented in this paper enhances the step distributed synthesis. Transformation and export is about converting the intern control model representation into data compliant to the standard IEC 61499. These data can be imported by several standard compliant tool suites[1] to implement the control code on control devices.

This contribution is structured as follows. Section II introduces the underlying modular modeling formalism. The structural analysis approach is presented in Section III. The enhancing of the supposed synthesis procedure is described in Section IV. An example is presented in Section V, followed by the conclusions.

[1]For some examples, see http://www.iec61499.org/tools.htm

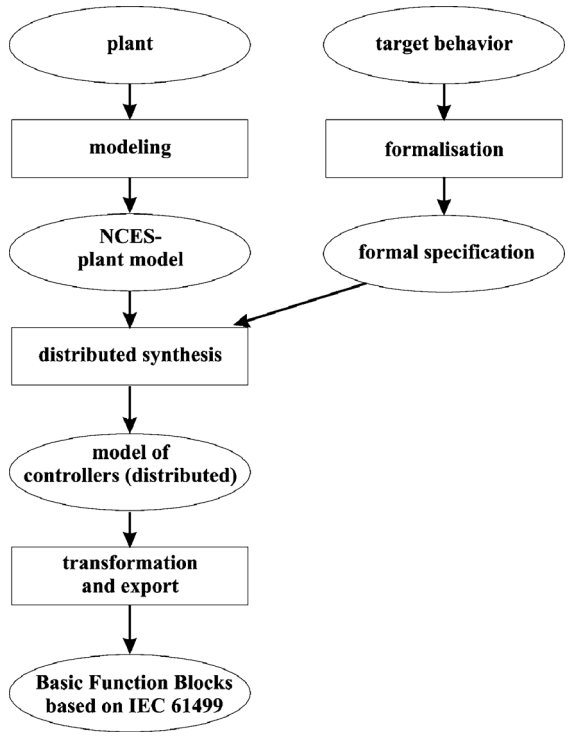

Fig. 1. Scheme of the control synthesis procedure.

II. Modeling Formalism

For DES and specification modeling safe Net Condition/Event Systems ($_S NCES$) are used, a one-bounded (and therefore called *safe*) subclass of Net Condition/Event Systems, which were introduced in [17]. A crucial advantage of $_S NCES$ is their modular structure. The behavior of plant components is encapsulated in reusable modules. Modules are interconnected via signal arcs. Thus, $_S NCES$ offer a natural way to model even large-scale systems, regarding their composition of basic components and groups of components. In [14], recommendations for a well-formed modular plant modeling with $_S NCES$ are given.

A. $_S NCES$ Model

Two kinds of modules are defined within $_S NCES$. The first kind are basic modules. They are supposed for modeling basic plant components (e.g., cylinders or sensors). The second kind are composite modules, which are composed of basic modules and/or other composite modules. Fig. 2 shows a basic module, which is used in the example in Section V. It represents the behavior of a clamping cylinder. Fig. 3 shows an example of a composite module, which is composed of basic modules; amongst others, it contains the basic module depicted in Fig. 2. The composite module represents the component group for

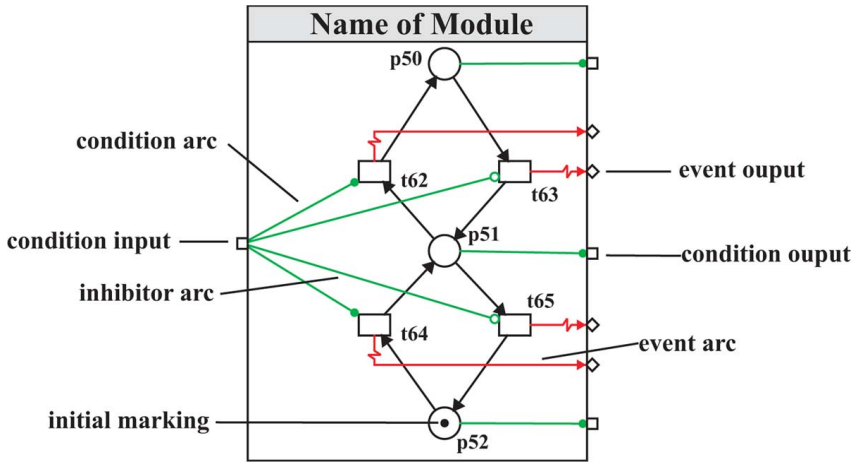

Fig. 2. Example for a safe Net Condition/Event System basic module.

a cylinder, including the basic plant components actuator, cylinder, and end positions sensors.

The complete definition of ${}_S NCES$ would be too large for this contribution, thus, for more formal, details see [16]–[18]. Some fundamental definitions, with examples in Fig. 2, are given in the following.

Definition 2.1: A basic module $\mathcal{M}_\mathcal{B}$ is a tuple, given by

$$ {}_S NCEM = \{P, T, F, CN, EN, C^{\text{in}}, E^{\text{in}}, C^{\text{out}}, E^{\text{out}}, CI^{\text{arc}}, EI^{\text{arc}}, CO^{\text{arc}}, EO^{\text{arc}}, em, m_0\} $$

where P, T, F are (common to PN) sets of places, transitions, and ordinary arcs, $CN \subseteq P \times T, EN \subseteq T \times T$ are the sets of condition and event signals (e.g., $p_{51} \longrightarrow$ condition output), $C^{\text{in}}, E^{\text{in}}, E^{\text{out}}, C^{\text{out}}$ are the sets of signal in-/outputs, $CI^{\text{arc}} \subseteq C^{\text{in}} \times T, EI^{\text{arc}} \subseteq E^{\text{in}} \times T$ are the sets of condition and event input arcs (e.g., condition input $\longrightarrow t_{62}$), $CO^{\text{arc}} \subseteq P \times C^{\text{out}}, EO^{\text{arc}} \subseteq T \times E^{\text{out}}$ are the sets of condition and event output arcs (e.g., $t_{63} \dashrightarrow$ event output), $em : T \rightarrow \{\boxtimes, \boxtimes\}$ is the event mode for every transition, m_0 is the initial marking of the places.

The event mode *em* expresses if the firing of a transition depends on a disjunctively $\boxtimes$ or conjunctively $\boxtimes$ combination of its input arcs.

The defined PN and signal components are arranged in a modular structure, which is shown in Fig. 2.

Definition 2.2: A composite module $\mathcal{M}_\mathcal{C}$ consists of submodules, which can be also composite modules or/and basic modules (Fig. 3). The submodules of $\mathcal{M}_\mathcal{C}$ are interconnected and connected with the inputs and outputs of $\mathcal{M}_\mathcal{C}$ via signal arcs.

Definition 2.3: If an ${}_S NCE$ Module has no inputs or outputs it is called safe Net Condition/Event System $\mathcal{M}_\mathcal{S}$.

An input state is defined for the signal inputs of ${}_S NCES$ modules as follows.

Definition 2.4: The input state (is) of an ${}_S NCEM$ is a mapping is: $C^{\text{in}} \cup E^{\text{in}} \rightarrow \{0, 1\}$ assigning a value of $\{0, 1\}$ to each signal input.

The semantics of ${}_S NCES$ is given in terms of steps.

Steps are sets of transitions interconnected via event signals. An event signal synchronizes two transitions in one direction ($t_i \dashrightarrow t_j$) under the enabling conditions.

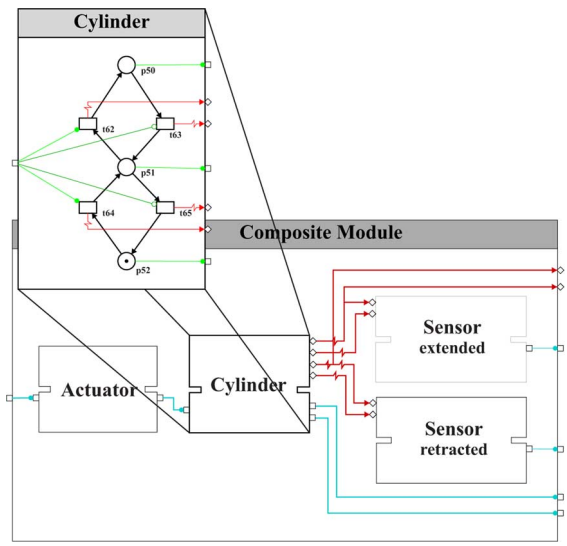

Fig. 3. Example for basic and composite modules. From left to right, the basic modules represent the behavior of an actuator, a clamping cylinder, and its two end position sensors.

For enabling of a transition and a step, only the marking of places and the input state of the module are of interest.

Definition 2.5: A transition $t \in T$ of an ${}_S NCEM$ is given as follows:

1) *marking-enabled* at a marking m iff $(\forall p \in P \text{ with } (p,t) \in F : m(p) = 1) \wedge (\forall p \in P \text{ with } (t,p) \in F : m(p) = 0)$;
2) *condition-enabled* at a marking m and an input state *is* iff $\forall p \in P \text{ with } (p,t) \in CN : m(p) = 1$ and $\forall c^{\text{in}} \in C^{\text{in}} \text{ with } (c^{\text{in}}, t) \in CI^{\text{arc}} : is(c^{\text{in}}) = 1$.

A transition is marking-enabled if all pre-places are marked and all post-places are unmarked. A transition is condition-enabled if all places that are connected via condition arcs are marked and all connected condition inputs have the value one.

With these terms, we can define sets of event-interconnected transitions that are called steps in general and enabled steps in particular.

Definition 2.6: Let $\mathcal{M}$ be an ${}_SNCEM$ with the marking m and the input state *is* and $\xi \subset T$ a nonempty set of transitions within $\mathcal{M}$.

ξ is a *step* iff:

1) $|\xi \cap (T_E)| = 1$, while $T_E := \{t \in T \mid \nexists t' \in T : (t', t) \in EN\}$;
2) for every transition $t \in \xi$ with $t \notin (T_E)$ holds:
   - $em(t) = \boxtimes \wedge ((\exists t^{'} \in \xi : (t^{'}, t) \in EN) \vee (\exists e^{\text{in}} \in E^{\text{in}} \text{ with } (e^{\text{in}}, t) \in EI^{\text{arc}} : is(e^{\text{in}}) = 1))$ or
   - $em(t) = \boxtimes \wedge ((\forall t^{'} \text{ with } (t^{'}, t) \in EN : t^{'} \in \xi) \wedge (\forall e^{\text{in}} \in E^{\text{in}} \text{ with } (e^{\text{in}}, t) \in EI^{\text{arc}} : is(e^{\text{in}}) = 1))$; and
3) all transitions are free of conflicts to each other.

Ξ is the set of steps within $\mathcal{M}$.

ξ is called *enabled step* under m and *is* iff ξ is marking and condition-enabled under m and is and there is no set of transitions with $\xi' = \xi \cup \{t\}$, which is also a step and marking and condition-enabled under m and *is*.

The defined enabled steps are always maximal steps and contain exactly one trigger transition. Conflict transitions must not be part of the same step.

The effect of firing an enabled step on the marking of the net is defined as follows.

Definition 2.7: Let M be an ${}_SNCEM$ with the marking m and the input state *is*.

If ξ is an enabled step under m and *is*, than ξ is enabled to fire. The successor marking m' is determined for $p \in M$ to

$$m'(p) = \begin{cases} 1, & \text{if } \exists t \in \xi : (t, p) \in F \\ 0, & \text{if } \exists t \in \xi : (p, t) \in F \\ m(p), & \text{else.} \end{cases}$$

An enabled transition is forced to fire by an incoming event signal.

B. Specification Model

Next to the formal plant model, two kinds of formal specifications are used in the presented approach. The first kind are forbidden states, which are *safety critical states* concerning human beings, environmental and plant safety. They are defined in terms of *state predicates* [16] and they are the starting points for a backward search algorithm for controllable steps avoiding these states [16].

The second kind are formal specifications of the plant's *desired process behavior*, or more precisely, the desired transformation of work piece (WP) properties during a process cycle. Work piece properties can be geometric properties or work piece positions, etc. These specifications are partial orders over the desired process behavior and is given in terms of ${}_SNCES$. Fig. 4 shows an example cutout of such a specification module. These modules are connected with event arcs to the model of the uncontrolled plant behavior. For example, the lower left event input *WP leaving* in the module in Fig. 4 is intended to receive an event from a corresponding plant component. With many of such specification modules for example, it is possible to specify work piece presence with a desired order at different plant locations. This kind of specification is already comprehensible "by view" compared with a set of inequalities.

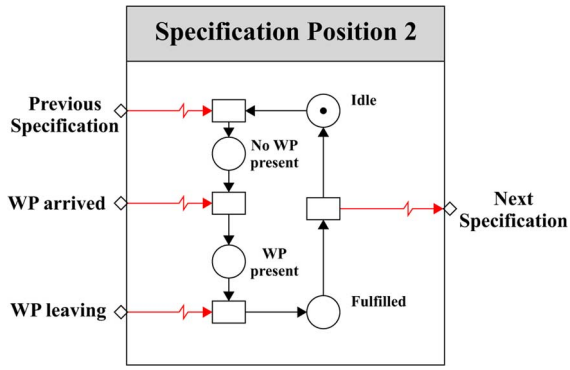

Fig. 4. Example for formal specification in terms of partial order over work piece properties.

The modularity of ${}_SNCES$ supports the supposed distributed control synthesis in such a way, that specification modules can be assigned to each relevant plant component or assembly group. This mapping can be used during the supposed distribution of the synthesized control. Currently, the ${}_SNCES$ specification modules are derived manually. But this step can be automated if an unambiguous name assignment from a design framework, e.g., a SysML-based specification [19], to the ${}_SNCES$ model and specification modules is guaranteed.

III. Structural Analysis Approach

For analyzing the structure of PN, rich methodologies exist [12], [20], e.g., regarding net properties like liveness or deadlocks or structural properties. The modeling formalism used in this contribution consists of PN components and signal components (see Section II.A). Thus, in the following known PN structural properties are adopted to ${}_SNCES$ for extending ${}_SNCES$ analysis methodologies according to structural properties.

A. Basics

Synthesis of process control is about cyclic process behavior. Cyclic, in this case, means recurring behavior of the plant and its components. Even the basic components (e.g., cylinders, valves, etc) of the plant have to return in their initial states. The behavior of these basic components is modeled as basic modules (Definition 2.1), which have to return into their initial states again. Plant behavior and components which are necessary for starting up the plant are excluded from the considerations of this contribution, because they are not part of the relevant cyclic process behavior. The initial state of a basic module is implicitly given by the initial marking of its PN component and is *invariant* regarding the firing sequence of its transitions (if well-formed modeling was fulfilled [16]). This invariance property can be described by *transition-invariants* (t-invariant).

B. PN Structural Properties

The potential of using PN invariants is discussed in [20]. According to the concept of well-formed *NCES* [14], the PN component of every basic module is live and thus covered by a t-invariant.

Definition 3.1: Let N be a PN and C the corresponding incidence matrix. A t-invariant $\vec{x}$ of N is every non-trivial integer solution of the equation system $C \cdot \vec{x} = \vec{0}$.

In the case of $_S$*NCES*, C corresponds with the incidence matrix of the composed $_S$*NCES* model. It covers only the PN components without the signal interconnections.

To utilize the t-invariants for the presented analysis methodology, one has to take into account the signal interconnections between the $_S$*NCE* Modules, too. Such a signal connection starts at a transition of a t-invariant of a basic module and triggers the firing of an enabled transition in a subsequent basic module. Thus, conclusions about the plant behavior are possible, based only on the model structure without any dynamic effects of the plant. In the following, these conclusions are used to reduce the computational complexity of state space calculation.

C. New Representation of Structural Properties

Incorporating the previously described PN structural properties and the signal components of $_S$*NCES*, we define a new graphical representation.

Definition 3.2: Let $\mathcal{M}_\mathcal{S}$ be an $_S$*NCES* and $TI^\mathcal{M}$ the set of all t-invariants of $\mathcal{M}_\mathcal{S}$. Then, the *transition invariant graph* (*TIG*) is defined as the tuple $TIG := (V, E)$ with:

- $V \equiv TI^\mathcal{M}$ is the set of t-invariants. The particular TIs are displayed as boxes, labeled with the unique number of the t-invariant they represent.
- E is the set of directed arcs representing event interconnections between t-invariants.
- There is exactly one simple directed arc between TI_j and TI_k with $TI_j, TI_k \in TI^\mathcal{M}$ iff
  at least one event interconnection $t_x \rightharpoonup t_y$ exists in $\mathcal{M}_\mathcal{S}$ with $t_x \in TI_j$ and $t_y \in TI_k$. In other words, a *TIG* is no multigraph.

All necessary information for generating the *TIG* is implicitly given in the plant model. Hence, the *TIG* is some kind of aggregation of these information into a new graphical representation, which eases the following analysis steps. Regarding its purpose, the *TIG* is an abstraction without loss of information.

During building the *TIG*, one has to pay attention to two special cases:

1) nested t-invariants within one basic module;
2) event-synchronized places (described in [16]).

Their transformation into *TIG* representation was described in detail in [7]. The *TIG* has, of course, properties like paths and cycles as well. They are defined in the following.

Definition 3.3: Event-coupled t-invariants path $(EKTIP)$. Let $\mathcal{M}_\mathcal{S}$ be an $_S$*NCES* and $TI^\mathcal{M}$ the set of all t-invariants of $\mathcal{M}_\mathcal{S}$. A sequence $EKTIP := \{TI_1, \ldots, TI_n\}$ is called *event-coupled t-invariants path* iff

- $\forall TI_i \in EKTIP : TI_i \in TI^\mathcal{M}$;
- $\forall TI_i \in EKTIP \,|\, 2 \leq i \leq n : \exists t_x \in TI_{i-1} \wedge \exists t_y \in TI_i : t_x \rightharpoonup t_y$ holds.

Definition 3.4: Event-coupled t-invariants cycle $(EKTIZ)$. Let $\mathcal{M}_\mathcal{S}$ be an $_S$*NCES* and $TI^\mathcal{M}$ the set of all t-invariants of $\mathcal{M}_\mathcal{S}$. A sequence $EKTIZ := \{TI_1, \ldots, TI_n\}$ is called *event-coupled t-invariants cycle* iff

- the same conditions as for an $EKTIP$ and
- $\exists t_m \in TI_n \wedge \exists t_n \in TI_1 : t_m \rightharpoonup t_n$ holds.

For instance, Fig. 5 presents a cutout of an $_S$*NCES* and the corresponding *TIG*. For readability reasons, the $_S$*NCES* is depicted in terms of $_S$*NCES* Modules, without showing all the PN components. Furthermore, only the relevant event interconnections are shown. The composite module in the middle represents the transfer behavior of a rotary table. This table "transfers" the work piece property *presence* along its four table positions. The work piece presence at each table position is represented by the four basic modules around the transfer module. The corresponding *TIG* is depicted in Fig. 5(b). There, each t-invariant of the $_S$*NCES* is represented by a numbered node. For example, in Fig. 5(a) WP property *presence* at position 3 is modeled as basic module. Because this module contains only one t-invariant, it is represented in the *TIG* in Fig. 5(b) by a single node (TI_{31}). Due to comprehensiblity, the *TIG* nodes are clustered corresponding to the positions of their corresponding t-invariants in the $_S$*NCES*. For distinction, the diamond-shaped nodes represent the t-invariants which model the work piece property *presence*. The gray colored nodes represent the t-invariants of the (composite) transfer module, which transfers the property *presence* from one table position to the next.

At first view, the *TIG* in Fig. 5(b) shows a cycle. This $EKTIZ$ represents the behavior of the rotary table. Contrarily, this fact is not easy to see in the $_S$*NCES* in Fig. 5(a), because of its hierarchical composition. The reason for this is that the *TIG* is a lossless abstraction of the modularity of $_S$*NCES*. Thus, the *TIG* is an appropriate base to ease subsequent analysis steps, especially if considering large-scale models.

IV. Reducing Complexity

A. Application of the TIG

Section I pointed out the problem of complexity during control synthesis. The approach introduced in the previous section can help to handle this, as it is explained in the following.

As described in Section II-B, the process behavior specification is given in terms of $_S$*NCES* Modules and is linked via event arcs to the model of the uncontrolled plant behavior.

If we calculate the *TIG* of the original plant model linked to the specification, the result will be the abstracted graphical description of causal dependecies described in the previous section, which also contains dependencies between the original plant model and the specification model.

Let $EKTIP^\mathcal{M}$ denote the set of all $EKTIP$s of the *TIG*. Then, this set represents these causal dependencies within the plant model. But only $EKTIP$s complying with the process behavior specification are of interest, regarding the control synthesis according to the specified cyclic process behavior. For example, if a work piece should be ejected after passing all table positions in the cutout example in Fig. 5, paths ranging from *WP Properties 1* to *WP Properties 4* are interesting for subsequent usage. A whole cycle, e.g., starting and ending in *WP Properties 1*, is not desired because this means a work piece stays on the rotary table and is not ejected.

The constraints for extracting "desired" $EKTIP$s are explained in the following. The extraction of relevant paths out

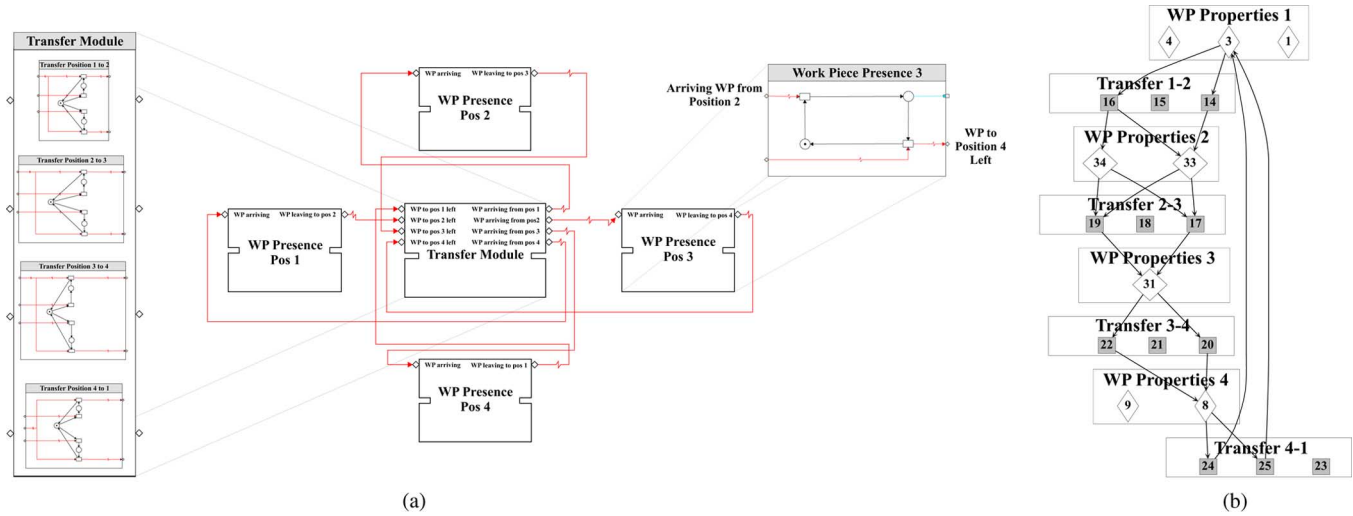

Fig. 5. Little example for transforming an ${}_S NCES$ into a *TIG*. (a) ${}_S NCES$. Rotary table (center) with four table positions. It transfers work piece property *presence* (rest of the modules) from one table position to another. (b) *TIG*. Diamond-shaped nodes: work piece *presence*, gray nodes: transfer of work piece property *presence*.

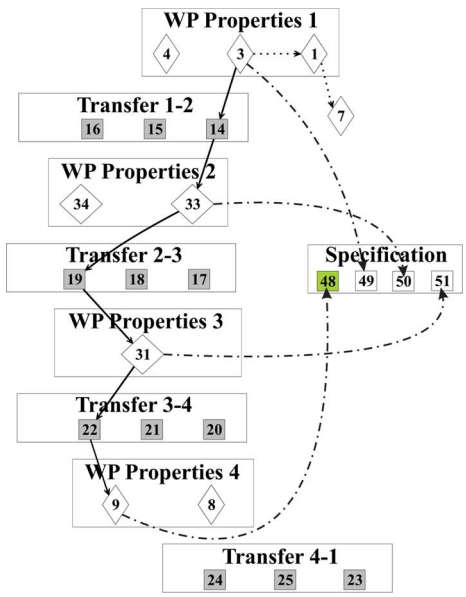

Fig. 6. Extraction of relevant $EKTIP$s. Solid line: relevant $EKTIP$; dotted line: rejected $EKTIP$; dotted–dashed line: "passing condition" for relevant $EKTIP$s.

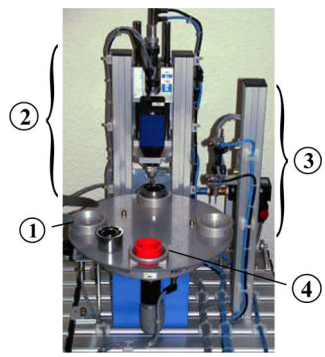

Fig. 7. Processing station of a laboratory-scale production system. The table positions are numbered clockwise from 1 to 4 beginning at the left.

of $EKTIP^{\mathcal{M}}$ relies on the fact, that t-invariants of the specification modules are event-connected with corresponding t-invariants "of interest" in the ${}_S NCES$ model. This is a structural feature, which can be examined easily.

Let $EKTIP^{\text{comp}} \subset EKTIP^{\mathcal{M}}$ denote the set of specification complying $EKTIP$s. $EKTIP^{\text{comp}}$ is generated by extracting $EKTIP$s out of $EKTIP^{\mathcal{M}}$, which meet the following two conditions:

- Passing all t-invariants of the specification in the correct order.
- $EKTIP$ ends in the t-invariant of specification, which represents last step of desired process behavior.

For the example in Fig. 5, this is shown in Fig. 6. There, the process specification is about work piece presence at every table position. The specification is modeled so that this fact is expressed by a basic module, respectively a t-invariant, for each table position $(TI_{48}, TI_{49}, TI_{50}, TI_{51})$. For example, the $EKTIP$ drawn with solid edges is a relevant one. It passes (depicted by the dot-dashed edges) the t-invariants of the specification in the correct order and ends in TI_{48}, which represents the last step of this example process cycle. Hence, this $EKTIP$ complies the specified process behavior on the basis of the given uncontrolled plant behavior model and is inserted into the set $EKTIP^{\text{comp}}$. The other $EKTIP$ $\{TI_3, TI_1, TI_7\}$ (dotted edges) is rejected because it does not comply the above given two conditions. It represents the setting of a sensor, which is not relevant for control synthesis.

B. Partial Reachability Analysis

In the previous section, the set $EKTIP^{\text{comp}}$ was generated, which contains *TIG* paths covering the whole process behavior specification based on the structure of the uncontrolled plant model. Each $EKTIP \in EKTIP^{\text{comp}}$ complies the specified process behavior, regarding completeness and order (passing of all t-invariants of the specification in the correct order). Thus, the set $EKTIP^{\text{comp}}$ represents some kind of desired plant behavior according to the causal dependencies of the plant model.

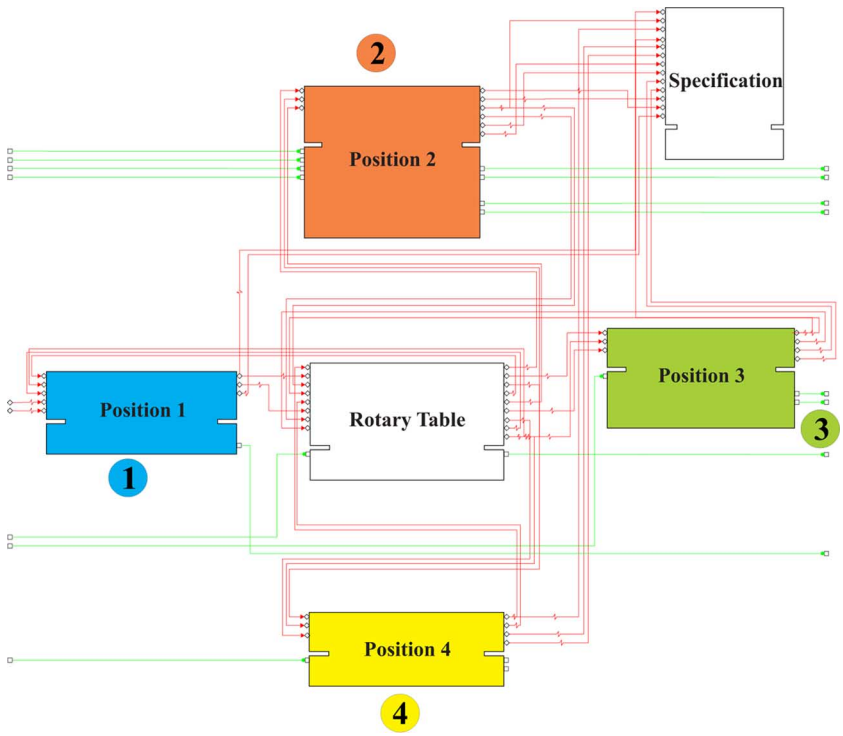

Fig. 8. $_S$*NCES* model of the processing station (Fig. 7). Each component group, which is shown in Fig. 7 and the specification are represented by a composite module. Due to readability, not every detail is labeled.

Remember, an $EKTIP \in EKTIP^{\text{comp}}$ is a sequence of t-invariants. Each of them represents a set of transitions. Thus, the transitions within $EKTIP^{\text{comp}}$ are representing state transitions of a *desired subspace* of the reachable state space of the uncontrolled plant behavior.

Contrarily, specified forbidden states are *undesired points* within the reachable state space.

Regarding complexity of control synthesis, it is beneficial to analyse only desired and not forbidden parts of the state space to determine a control model. Thus, the set of process specification compliant $EKTIP$s and the set of forbidden states are incorporated as conditions into a common state space computation as follows: State transitions are rejected/not computed if their subsequent state:

- is a forbidden state;
- has got an enabled step ξ, which does not contain at least one transition
  $t \in TI : (TI \in EKTIP \wedge EKTIP \in EKTIP^{\text{comp}})$.

Hence, only a part of the state space is computed. That is why this step is called *partial reachability analysis* (pRA). The state transitions, which were rejected during pRA are representing boundaries in the desired state space. They are the starting points for the controller synthesis in the supposed process control synthesis procedure [7], [14].

The pRA reduces the computational costs by reducing the amount of state space, which has to be computed. The significant effect of the pRA will be shown with an example in the following section.

V. Example

Fig. 7 shows a processing station of a modular production system[2] on the laboratory scale. It consists of three plant modules: a rotary table with four positions, a drilling module ②and a drilled hole testing module ③ Its purpose is to drill an incoming workpiece ① Afterwards, the hole testing module checks the result. Rejected workpieces are left on the rotary table to repeat processing, and accepted ones are ejected ④to the next plant station.

The $_S$*NCES* model (Fig. 8) of this processing station consists of 73 places and 92 transitions, which are hierarchically organized in in 32 basic modules (e.g., cylinders, actuators, or work piece properties) and 17 composite modules. Fig. 8 depicts the highest module level of the $_S$*NCES* model, consisting of one composite module for each table position, one for the table itself and in the upper right a composite module representing the specification of desired process behavior. Due to readability the highest hierarchical level of the $_S$*NCES* model is shown. To get an imagination of the model interior view remember Fig. 5(a). The basic module details shown there are contained in one of two composite submodules of the rotary table module shown in Fig. 8. Fig. 9 shows the corresponding *TIG*. The *TIG* contains the example cutout previously shown in Fig. 5(b). For clarity, the t-invariants are grouped according to the assembling groups they represent.

The whole specification for this example consists of a set of forbidden states and a process behavior specification in terms of $_S$*NCES* (in the upper right in Fig. 8). The forbidden states concern safety constraints, e.g., the driller has to be retracted if the

[2][Online]. Available: http://aut.informatik.uni-halle.de/forschung/testbed/

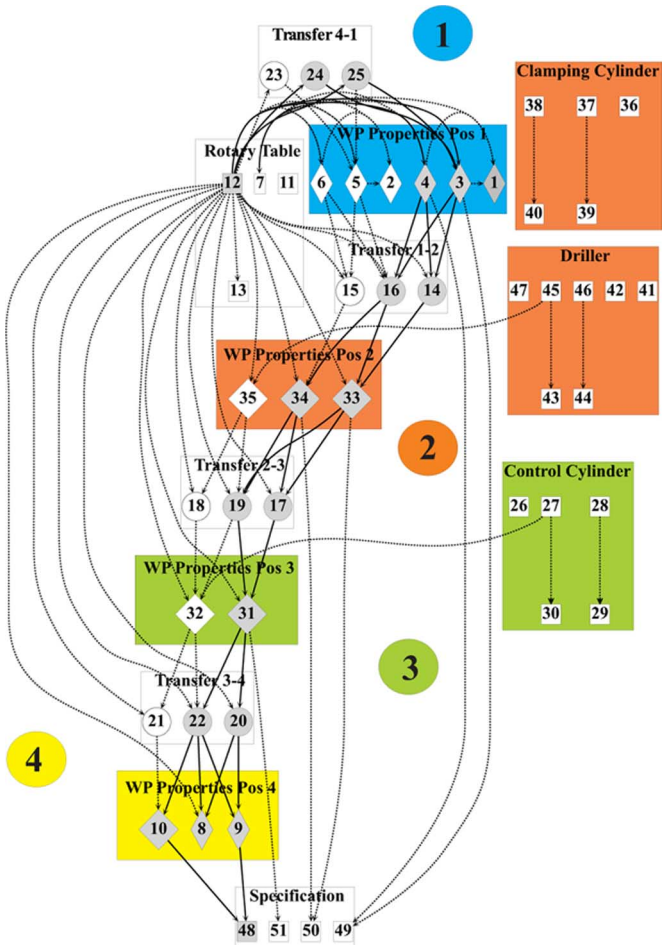

Fig. 9. *TIG* constructed from the ${}_S NCES$ model (Fig. 8) of the processing station.

table rotates. The specification of the desired process behavior is about the work piece property *presence*, which is "transfered" between the table positions. The specification (basic) modules (represented in Fig. 9 by the t-invariants $TI_{48}^{\text{spec}}, \ldots, TI_{51}^{\text{spec}}$) are connected to the original plant model using already existing events.

The *TIG* itself already allows some causal analysis. As mentioned before, it reflects the structural dependencies of the plant model. The gray highlighted t-invariants represent the work piece property *presence*. It is modeled within each table position (diamond-shaped t-invariants) and within the rotary table (circle-shaped t-invariants), where it is "transfered" from one table position to another.

Regarding the specification over the work piece property *presence*, we are interested in $EKTIP$s ranging from t-invariants representing ① to TI_{48}^{spec}, which belongs to the specification and marks the last step (processed work piece is leaving/new one arrives) of the desired cyclic process behavior. Such $EKTIP$s are collected in the set $EKTIP^{\text{comp}}$, like described in the previous Section IV-A. For the example *TIG* in Fig. 9, the $EKTIP$s of this set are represented by thick solid edges.

According to the presented pRA (Section IV-B), the extracted set $EKTIP^{\text{comp}}$ and the forbidden state specifications for the example in Fig. 7 are used to limit the state space analysis and lead to the following results, which are presented in Table I.

TABLE I
COMPARISON OF RG SIZE REGARDING THE USED SPECIFICATION AND ANALYSIS METHODOLOGY

specification	reachability graph			
	nodes	[%]	edges	[%]
none	9,216	100	81,408	100
forbidden states	2,880	31.25	20,512	25.2
specification modules	255	2.77	1,792	2.2
both	*192*	*2.08*	*1,216*	*1.49*

There, the size of the state space of the interconnected uncontrolled plant behavior model and the specification model is expressed by the node and edge numbers of the corresponding complete and partial reachability graphs (pRG). Compared to the use of only forbidden states specification, the presented approach reduces the size of the resulting pRG significantly. It will ease the handling and analysis during the following steps of the supposed synthesis procedure, which means a reduced complexity.

The lower size of the pRG can be interpreted as follows. By applying the presented approach, undesired or not reasonable trajectories through the state space are avoided, e.g., every cylinder of the plant station could extend and retract before anything else happens, because we got a model of the uncontrolled plant behavior. Regarding the desired process behavior, this makes no sense or even fulfill a forbidden state. Hence, such a trajectory is rejected.

The border states of the pRA result and the specified forbidden states are the starting points for a backward search for controllable steps avoiding these states. The determined controllable steps will be the base for determining a control model ensuring the desired plant behavior. In a further step it should transformed into IEC 61499 compliant data.

VI. CONCLUSION

Complexity of models and related algorithms has always been and will always be a fundamental, crucial issue for the application of any formal method to systems of real size and complexity. Reducing the complexity is therefore the only solution to pave the way for application of formal methods to discrete controller design. We therefore proposed a novel approach for complexity reduction that is based on a modular modeling formalism and structural properties of models of uncontrolled plant behavior that are designed using this modeling formalism. The approach takes advantage from the t-invariants of the underlying Petri net components of the model and uses the event interconnections of transitions that are part of the t-invariants. As a result, we get the so-called transition invariant graph that represents a superset of the possible cyclic plant trajectories. Along with specifications of desired and forbidden plant behavior, we extract the trajectories of the plant that fulfill the specifications. From these trajectories, we plan to extract a set of distributed controllers that would enforce the correct plant behavior in closed loop. This is subject of future research. Even without having this accomplished, the transition invariant graph may support manual controller design in a conventional way done by experienced engineering staff. The reduction of

complexity using the method even for small academic examples is significant (see Table I). After having embedded the approach into the whole synthesis framework shown in Fig. 1, we hope to be able to apply the framework to systems of industrial size and complexity in manufacturing.

REFERENCES

[1] P. Ramadge and W. Wonham, "The control of discrete event systems," *Proc. IEEE*, vol. 77, no. 1, pp. 81–97, Jan. 1989.

[2] F. Charbonnier, H. Alla, and R. David, "The supervised control of discrete-event dynamic systems," *IEEE Trans. Control Syst. Technol.*, vol. 7, no. 2, pp. 175–187, Mar. 1999.

[3] J.-M. Roussel and A. Giua, "Designing dependable logic controllers using the supervisory control theory," in *Proc. 16th IFAC World Congress*, Praha, Czech Republic, Jul. 2005, p. CDROM paper no. 04427.

[4] D. Gouyon, J.-F. Pétin, and A. Gouin, "A pragmatic approach for modular control synthesis and implementation," *Int. J. Production Res.*, vol. 2, no. 14, pp. 2839–2858, 2004.

[5] E. Dumitrescu, M. Ren, L. Piétrac, and E. Niel, "A supervisor implementation approach in discrete controller synthesis," in *Proc. IEEE Int. Conf. Emerging Technolog. Automation*, Hamburg, Germany, Set. 2008, pp. 1433–1440.

[6] J. Flochová, "A petri net based supervisory control implementation," in *Proc. IEEE Int. Conf. Syst., Man Cybern.*, Washington, DC, Oct. 2003, vol. 2, pp. 1039–1044.

[7] T. Winkler, H.-C. Lapp, and H.-M. Hanisch, "A new model structure based synthesis approach for distributed discrete process control," in *Proc. 9th IEEE Int. Conf. Ind. Inf.*, Lisbon, Portugal, 2011, pp. 527–532, IEEE IES.

[8] L. Feng, K. Cai, and W. M. Wonham, "A structural approach to the nonblocking supervisory control of discrete-event systems," *Int. J. Manuf. Technol.*, vol. 41, no. 11-12, pp. 1152–1168, 2009.

[9] L. Holloway, B. Krogh, and A. Giua, "A survey of petri net methods for controlled discrete event systems," *Discrete Event Dynamic Syst.: Theory Applic.*, vol. 7, no. 2, pp. 151–190, 1997.

[10] M. Uzam and W. Wonham, "A hybrid approach to supervisory control of discrete event systems coupling RW supervisors to petri nets," *Int. J. Adv. Manuf. Technol.*, vol. 28, no. 7-8, pp. 747–760, 2006.

[11] M. V. Iordache and P. J. Antsaklis, "Supervision based on place invariants: A survey," *Discrete Event Dynamic Systems*, vol. 16, no. 4, pp. 451–492, 2006.

[12] W. Reisig, , W. Reisig, Ed., *Petri Nets: An Introduction*, ser. Monographs in Theoretical Computer Science. An EATCS Series. Berlin, Germany: Springer, 1985, vol. 4.

[13] Z. Li and M. Zhou, "Two-stage method for synthesizing liveness-enforcing supervisors for flexible manufacturing systems using petri nets," *IEEE Trans. Ind. Inf.*, vol. 2, no. 4, pp. 313–325, Nov. 2006.

[14] D. Missal and H.-M. Hanisch, "A modular synthesis approach for distributed safety controllers, part A & B," in *Proc. 17th IFAC World Congress*, Seoul, Korea, Jul. 2008, vol. 17, no. 1, pp. 14 473–14 484.

[15] A. I. Vasiliu, A. Dideban, and H. Alla, "Control synthesis for manufacturing systems using non-safe petri nets," *J. Control Eng. Appl. Inf.*, vol. 11, no. 2, pp. 43–50, June 2009.

[16] D. Missal, , H.-M. Hanisch, Ed., *Formal Synthesis of Safety Controller Code for Distributed Controllers*. Berlin: Logos Verlag, 2012, vol. 9.

[17] H.-M. Hanisch and M. Rausch, "Synthesis of supervisory controllers based on a novel representation of condition/event systems," in *Proc. Conf. Syst., Man Cybern.*, Vancouver, BC, Canada, Oct. 1995, vol. 4, pp. 3069–3074.

[18] L. Pinzon, M. Jafari, H.-M. Hanisch, and P. Zhao, "Modelling admissible behavior using event signals," *IEEE Trans. Syst., Man Cybern. B, Cybern.*, vol. 34, no. 3, pp. 1435–1448, Mar. 2004.

[19] M. Hirsch, , H.-M. Hanisch, Ed., *Systematic Design of Distributed Industrial Manufacturing Control Systems*. Berlin, Germany: Logos Verlag, 2010, vol. 6.

[20] P. Starke and S. Roch, "Analysing signal-net systems," Humboldt-Universität zu Berlin, Berlin, Germany, Tech. Rep.162, Sep. 2002, Informatikberichte.

Hans-Christian Lapp received the Diploma degree in computer science from the Technical University of Ilmenau, Ilmenau, Germany, in 2006.

He was then a Scientific Assistant with the Institute of Computer Science, University of Leipzig, Leipzig, Germany. Since 2009, he has been with the Chair for Automation Technology, Martin-Luther-University Halle-Wittenberg, Halle, Germany. His research interest is in modeling and distributed control synthesis.

Hans-Michael Hanisch (SM'01) received the Ph.D. degree in chemical engineering from Polytechnical Institute Merseburg, Merseburg, Germany, in 1983, and the Habilitation in automation technology from the University of Dortmund, Dortmund, Germany, in 1995.

From 1994 until 1999, he was a Professor with the University of Magdeburg, Magdeburg, Germany. Since 2000, he has been with Martin-Luther-University Halle-Wittenberg, Halle, Germany, where he is a Full Professor of the Chair for Automation Technology. His current research interests include discrete event and hybrid systems, modeling, analysis and verification of distributed systems, control synthesis, reconfigurable control, and others.