Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

Modeling of asynchronous discrete-event systems as networks of input-output automata

Sebastian Drüppel *, **Jan Lunze** **, **Martin Fritz** ***

* *Robert Bosch GmbH, Corporate Sector Research and Advance Engineering, Mechatronic Engineering Group, D-70442 Stuttgart, Germany (Tel:+49-711-811-49402; e-mail: drueppel.sebastian@de.bosch.com).*
** *Ruhr-University Bochum, Institute of Automation and Computer Control, D-44780 Bochum, Germany (e-mail: lunze@atp.rub.de).*
*** *Robert Bosch GmbH, Automotive Aftermarket, D-73207 Plochingen, Germany (e-mail: martin.fritz@de.bosch.com).*

Abstract: A new approach for component-oriented modeling of asynchronous discrete-event systems is presented where input-output (I/O) automata are used for representing the components. Coupling signals are introduced to describe the interactions among the components. The resulting network of I/O-automata has a direct correspondence to the block diagram. By using the parallel composition rule known from standard automata modeling as an example, it is shown that the new model is applicable for at least the same class of asynchronous discrete-event systems as the known modeling formalisms.

Keywords: Discrete-event system, component-oriented modeling, coupling signals, standard automaton, input/output automaton

1. INTRODUCTION

The well known problem of *state space explosion* in the modeling of discrete-event systems can be overcome by component-oriented modeling philosophies, where the overall system is treated as a set of interacting components. The separate models that are set up for the components are coupled to represent the overall system (Sreenivas and Krogh (1991)).

This modeling way has several advantages in contrast to monolithic modeling. First, setting up models of smaller components is much easier than describing the composite overall system by a unique model. Second, component-oriented models retain the structure of the system, which can be used later in the analysis or the design tasks carried out by means of these models.

This paper proposes a new component-oriented modeling framework for asynchronous discrete-event systems that is motivated by the diagnosis of technical systems (Blanke et al. (2006)). The overall system is considered as a set of interacting input-output (I/O) automata. Coupling signals s_i and r_i are used to represent the interactions in an explicit way (Fig.1). The overall model retains the structure of the system and has a direct correspondence to the block diagram, which is widely used in the modeling of technological systems and control design.

The idea to use networks of I/O-automata has been followed by only a few authors. Three types of interconnections have been considered in literature: *side-by-side-composition (parallel connection)*, *cascade composition (series connection)* and *feedback composition* (Lee and Varaiya (2003) and Lunze (2006)). Due to the assumption that *all* components behave in *synchrony*, conditions have to be met to obtain a *well-defined* feedback connection (Neidig and Lunze (2005)). This paper will show that the structure of the system has not to be restricted to these three connection types but may be described by a more general interaction block $\boldsymbol{K}$ introduced in Fig. 1.

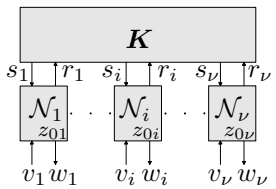

Fig. 1. Network of input/output-automata in block diagram form

This model distinguishes severely from component-oriented descriptions published in the past that likewise use automata to represent the components, but which define the interactions among the components in terms of events. The components are described by finite-state machines that will be referred to as *standard automata* here to distinguish them from I/O-automata. The main difference lies in the fact that the state transitions are represented by events rather than input symbols (Cassandras and Lafortune (1999) and Wonham (2005)). For component-oriented modeling, several composition operators have been proposed in literature (see Wenck and Richter (2004) for an overview). The main idea is to distinguish pri-

vate events that may occur in a single component, from common events that have to occur in all components. The *synchronous product* is used to describe the synchronization of the behavior of two components for all common events, whereas the *parallel composition* permits autonomous state changes of single components for their private events (cf. Section 3.1). For systems with higher priority events, the *prioritized synchronous composition* can be used (Heymann (1990)).

The paper proposes the new model in Sec. 2 and shows the relation to the modeling formalism with standard automata in Sec. 3.

2. A NEW MODELING FORMALISM FOR INTERCONNECTED DISCRETE-EVENT SYSTEMS

2.1 The I/O-automata network

The system is considered as the interconnection of a finite number of components C_i $(i = 1, \ldots, \nu)$. Each component is subject to two different kinds of interactions (Fig. 1). First, the control input v_i and control output w_i are used to model the interaction of the component C_i with its environment. Correspondingly, this interaction is represented for the overall system by the signal vectors

$$\boldsymbol{v} = (v_1 \ldots v_\nu)^T \in \mathcal{N}_v = \mathcal{N}_{v_1} \times \cdots \times \mathcal{N}_{v_\nu} \quad (1)$$

$$\boldsymbol{w} = (w_1 \ldots w_\nu)^T \in \mathcal{N}_w = \mathcal{N}_{w_1} \times \cdots \times \mathcal{N}_{w_\nu}. \quad (2)$$

Second, two coupling signals s_i and r_i are introduced for each component C_i to model the interaction with other components. In general, this signals may be vectors as C_i may affect more than one component via different signal paths and may have an effect on several other components. This arbitrary interaction is restricted in here to scalar interconnection signals to basically explain the modeling approach. Accordingly, the signal vectors

$$\boldsymbol{s} = (s_1 \ldots s_\nu)^T \in \mathcal{N}_s = \mathcal{N}_{s_1} \times \cdots \times \mathcal{N}_{s_\nu} \quad (3)$$

$$\boldsymbol{r} = (r_1 \ldots r_\nu)^T \in \mathcal{N}_r = \mathcal{N}_{r_1} \times \cdots \times \mathcal{N}_{r_\nu} \quad (4)$$

are related to one another by the interaction block $\boldsymbol{K}$.

2.2 Component model

Each component C_i is represented by an I/O-automaton $\mathcal{N}_i$. The definition of I/O-automata found in Lee and Varaiya (2003) and Lunze (2006) is extended to cope with the coupling signals (Fig. 2). In the component models

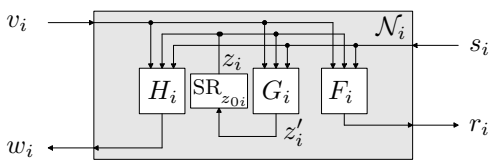

Fig. 2. Structure of the new I/O-automaton

$$\mathcal{N}_i = (\mathcal{N}_{z_i}, \mathcal{N}_{v_i}, \mathcal{N}_{w_i}, \mathcal{N}_{s_i}, \mathcal{N}_{r_i}, F_i, G_i, H_i, z_{0i}), \quad (5)$$

$\mathcal{N}_{z_i}$ is the set of states, $\mathcal{N}_{v_i}$ the set of control inputs, $\mathcal{N}_{w_i}$ the set of control outputs, $\mathcal{N}_{s_i}$ the set of interconnection inputs, $\mathcal{N}_{r_i}$ the set of interconnection outputs and $z_{0i} \in \mathcal{N}_{z_i}$ the initial state.

The dynamical properties of the component C_i are described by three functions:

- The **interconnection function** $F_i : \mathcal{N}_{z_i} \times \mathcal{N}_{v_i} \times \mathcal{N}_{s_i} \to \mathcal{N}_{r_i}$ specifies in each state z_i for all control inputs v_i and interconnection inputs s_i the interconnection output r_i
$$r_i = F_i(z_i, v_i, s_i). \quad (6)$$
- The **state transition function** $G_i : \mathcal{N}_{z_i} \times \mathcal{N}_{v_i} \times \mathcal{N}_{s_i} \to \mathcal{N}_{z_i}$ calculates in each state z_i the successor state z_i' for all control inputs v_i and interconnection inputs s_i
$$z_i' = G_i(z_i, v_i, s_i). \quad (7)$$
- The **output function** $H_i : \mathcal{N}_{z_i} \times \mathcal{N}_{v_i} \times \mathcal{N}_{s_i} \to \mathcal{N}_{w_i}$ specifies the control output w_i in each state z_i for all control inputs v_i and interconnection inputs s_i
$$w_i = H_i(z_i, v_i, s_i). \quad (8)$$

The block "SR" in Fig. 2 represents a shift register that is used to store the successor state z_i' calculated by (7) for one time step. It is initialized with z_{0i}.

2.3 The coupling model

The coupling model $\boldsymbol{K}$ is introduced to explicitly model the interaction of the system components. It is represented by a matrix $\boldsymbol{K}$ that links the interconnection outputs with the interconnection inputs

$$\boldsymbol{s} = \boldsymbol{K} \cdot \boldsymbol{r}. \quad (9)$$

The square matrix $\boldsymbol{K}$ has exactly one element K_{ij} in each row which has the value "1" to model the connection $s_i = r_j$. All other elements have the value "0". Hence, the relation

$$s_i = \boldsymbol{k}_i^T \cdot \boldsymbol{r} \quad (10)$$

holds for all interconnection inputs with $\boldsymbol{k}_i^T$ being the $i-th$ row of the matrix $\boldsymbol{K}$.

By this, every component may affect several other components via its interconnection output. The assumption to use scalar interaction signals restricts the interconnection in the opposite direction. There may be components that have no effect on any other component. However, an arbitrary interaction between the components is possible in the general case of vectorial interconnection signals.

2.4 Overall system model

The model of the overall system is a network of I/O-automata $\mathcal{NAN}$

$$\mathcal{NAN} = (\{\mathcal{N}_1, \ldots, \mathcal{N}_\nu\}, \mathcal{N}_z, \mathcal{N}_v, \mathcal{N}_w, \mathcal{N}_s, \mathcal{N}_r, \boldsymbol{K}, \boldsymbol{z}_0) \quad (11)$$

(Fig. 1) with the state

$$\boldsymbol{z} = (z_1 \ldots z_\nu)^T \in \mathcal{N}_z = \mathcal{N}_{z_1} \times \cdots \times \mathcal{N}_{z_\nu} \quad (12)$$

and the initial state

$$\boldsymbol{z}_0 = \{z_{0_1}, \ldots, z_{0_\nu}\}^T. \quad (13)$$

The control input $\boldsymbol{v} \in \mathcal{N}_v$ and the control output $\boldsymbol{w} \in \mathcal{N}_w$ are used to describe the interaction of the system with its environment. The interconnection input $\boldsymbol{s} \in \mathcal{N}_s$ and the interconnection output $\boldsymbol{r} \in \mathcal{N}_r$ are not seen from outside the system.

In general, the system may have loops, where the interconnection output of a component depends directly or via other components on itself. This results in the feedback problem known from Lee and Varaiya (2003) and Neidig

and Lunze (2005). The automata network is called *well-defined* if it is able to generate for an arbitrary sequence of control inputs $\boldsymbol{v}$ exactly one sequence of states and one sequence of control outputs $\boldsymbol{w}$. For a precise definition, the overall interconnection function

$$\boldsymbol{F}(\boldsymbol{z},\boldsymbol{v},\boldsymbol{s}) = \begin{pmatrix} F_1(z_1,v_1,s_1) \\ \vdots \\ F_\nu(z_\nu,v_\nu,s_\nu) \end{pmatrix}$$

is introduced.

Definition 1. The I/O-automata network (11) is *well-defined*, if two conditions are fulfilled:

(1) The following relation holds:
$$\mathcal{N}_{r_i} \subseteq \mathcal{N}_{s_j}. \tag{14}$$
(2) There exists for all control inputs $\boldsymbol{v} \in \mathcal{N}_v$ and all states $\boldsymbol{z} \in \mathcal{N}_z$ *exactly one* interconnection output $\boldsymbol{r} \in \mathcal{N}_r$ that solves
$$\boldsymbol{r} = \boldsymbol{F}(\boldsymbol{z},\boldsymbol{v},\boldsymbol{K}\cdot\boldsymbol{r}). \tag{15}$$

2.5 The notion of asynchrony

In modeling interconnected discrete-event systems by standard automata, asynchrony means that one component may carry out a state transition *independently* of the other components, if a *private event* of the component triggers the system (Cassandras and Lafortune (1999) and Lunze (2006)). The remaining components do not perform any state transition. To cover asynchrony, several composition operators like the *parallel composition* have been defined.

This situation occurs for I/O-automata that are coupled in the *side-by-side-composition* introduced in Lee and Varaiya (2003). There, the *stuttering symbol* called *absent* is used to allow for the reaction of a single component in the composition. The reacting component gets a non-vanishing input symbol due to which it performs a state transition and generates an output symbol. The stuttering symbol is given to the other component to remain in its current state and not to output a new symbol. As a result, one component performs an independent state transition. Recall that the stuttering symbol is also called the *empty symbol* and denoted by ε.

The formalism of the empty symbol ε is extended to cope with asynchrony in the modeling approach introduced in this paper. One I/O-automaton $\mathcal{N}_i$ of the network can perform an asynchronous state transition due to a control input $v_i \neq \varepsilon$ if all other components get the empty input symbol ε and if there is no interaction with the other components, i.e. if $s_i = \varepsilon$ and $r_i = \varepsilon$ hold. The interconnection functions F_i and the state transition functions G_i have to be defined accordingly for the empty control input or interconnection input symbols.

Asynchrony of one component can be extended to several components if there is an interaction between these components due to non-vanishing interconnection signals. The participating components may get in addition non-vanishing control inputs. In this case, the participating components perform a *synchronous* state transition while the non-participating components remain in their current states. Hence, the interacting components move *asynchronously* with respect to the remaining components. If all components are part of the interaction, the whole system performs a synchronous state transition.

2.6 Discussion

The new modeling formalism for interconnected discrete-event systems presented here allows for a *bottom-up* design of the model in four steps:

1. The system is decomposed into interconnected components.
2. Each component is *separately* modeled by an I/O-automaton without taking the other components into account.
3. The interconnection of the components is described by the coupling model.
4. The whole system model is composed of the component models and the coupling model.

There are four major advantages of this kind of modeling:

(1) Changes of the properties of one component only lead to a new component model but do not necessitate a change of other component models or of the interconnection relation.
(2) Changes in the structure of the overall system by e.g. adding or removing a component only results in a new coupling model but let the component models unaffected.
(3) A model library for component models can be used due to reasons (1) and (2).
(4) The cause-and-effect-chain of the system is explicitly covered by I/O-automata.

2.7 Example

To illustrate the modeling formalism introduced in this paper, the automaton graphs of $\mathcal{N}_1$ and $\mathcal{N}_2$ are shown in Fig. 3, where a directed edge from state z_i to its successor

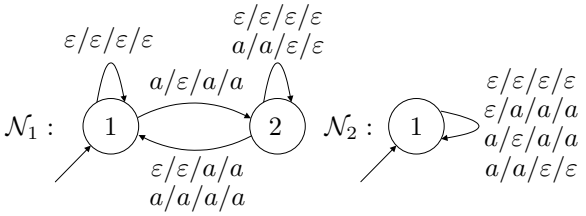

Fig. 3. Automaton graph of the I/O-automata $\mathcal{N}_1$ und $\mathcal{N}_2$

state z_i' is labeled by $v_i/w_i/s_i/r_i$, if (6) – (8) hold. The interconnection functions F_1 and F_2 are given in Tab. 1.

Table 1. Interconnection functions

(a) F_1

r_1	z_1	v_1	s_1
ε	1	ε	ε
a	1	a	a
ε	2	ε	ε
ε	2	a	ε
a	2	ε	a
a	2	a	a

(b) F_2

r_2	z_2	v_2	s_2
ε	1	ε	ε
a	1	ε	a
a	1	a	a
ε	1	a	ε

If the interconnection signals are related by

$$s_1 = r_2 \qquad\qquad s_2 = r_1, \tag{16}$$

the overall system represents a feedback connection and it has to be checked, whether the overall system is well-defined or not. From Fig. 3 follows directly, that (14) holds. Relation (15) has to be checked in each state $\boldsymbol{z}$

$$\boldsymbol{z} = (z_1, z_2)^T \in \mathcal{N}_z = \left\{(1,1)^T, (2,1)^T\right\} \tag{17}$$

for each input $\boldsymbol{v}$

$$\boldsymbol{v} = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \in \mathcal{N}_v = \left\{ \begin{pmatrix} \varepsilon \\ \varepsilon \end{pmatrix}, \begin{pmatrix} a \\ \varepsilon \end{pmatrix}, \begin{pmatrix} \varepsilon \\ a \end{pmatrix}, \begin{pmatrix} a \\ a \end{pmatrix} \right\}. \tag{18}$$

The results are shown in Tab. 2. It can be seen from Tab. 2(a) that (15) holds in state $\boldsymbol{z} = (1,1)^T$ for exactly one combination of the interconnection signals but not in state $\boldsymbol{z} = (2,1)^T$ (Tab. 2(b)). Hence, the overall system is not well-defined.

Table 2. Checking (15) for $\mathcal{N}_1$ and $\mathcal{N}_2$

(a) in $\boldsymbol{z} = (1,1)^T$

$(v_1, v_2)^T$	$r_1 = F_1(\cdot)$	$r_2 = F_1(\cdot)$
$(\varepsilon, \varepsilon)^T$	ε	ε
$(a, \varepsilon)^T$	a	a
$(\varepsilon, a)^T$	ε	ε
$(a, a)^T$	a	a

(b) in $\boldsymbol{z} = (2,1)^T$

$(v_1, v_2)^T$	$r_1 = F_1(\cdot)$	$r_2 = F_1(\cdot)$
$(\varepsilon, \varepsilon)^T$	ε a	ε a
$(a, \varepsilon)^T$	ε a	ε a
$(\varepsilon, a)^T$	ε a	ε a
$(a, a)^T$	ε a	ε a

A well-defined system is obtained by removing in the automaton graph of $\mathcal{N}_1$ at state 2 the transitions labeled $\varepsilon/\varepsilon/\varepsilon/\varepsilon$ and $a/a/a/a$. Then, there is exactly one combination of interconnection signals solving (15) for the state $(2,1)^T$. The resulting automaton is depicted in Fig. 4, where a directed edge is labeled by $\boldsymbol{v}/\boldsymbol{w}$.

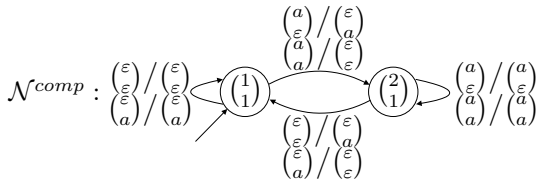

Fig. 4. Automaton graph of the composed I/O-automaton $\mathcal{N}^{comp}$

The state transitions of the overall system can be separated into synchronous and asynchronous moves.

(1) **Synchronous state transitions** occur, if both components get non-vanishing control input symbols. In the example, this is e.g. the case for a transition from state $\boldsymbol{z} = (1,1)^T$ to the successor state $\boldsymbol{z}' = (2,1)^T$ for the control input $\boldsymbol{v} = (a,a)^T$ and the generated control output $\boldsymbol{w} = (\varepsilon,\varepsilon)^T$. Another possibility is that there is no interaction due to non-vanishing input symbols, e.g. the self loop at state $\boldsymbol{z} = (2,1)^T$ for the control input $\boldsymbol{v} = (a,a)^T$ and the control output $\boldsymbol{w} = (\varepsilon/\varepsilon)^T$. It is also possible that one component gets a non-vanishing control input while the other one gets the empty symbol. The state transition from state $\boldsymbol{z} = (1,1)^T$ to the successor state $\boldsymbol{z}' = (2,1)^T$ is an example. Even though the control input $\boldsymbol{v} = (a,\varepsilon)^T$ has one vanishing element for the first component, the control output $\boldsymbol{w} = (\varepsilon,a)^T$ can have the non-vanishing element for the second one.

(2) An **asynchronous state transition** can be performed, if one component gets a non-vanishing control input while the other one gets an ε and there is no interaction between the components. This is for example the case for the self loop at state $\boldsymbol{z} = (2,1)^T$ for the control input $\boldsymbol{v} = (a,\varepsilon)^T$ the control output $\boldsymbol{w} = (a,\varepsilon)^T$.

3. COMPARISON OF NETWORKS OF I/O-AUTOMATA WITH COUPLED STANDARD AUTOMATA

The comparison of the network of I/O-automat introduced in Sec. 2 with coupled standard automata described in Lee and Varaiya (2003) and Wonham (2005) is presented in the following. For simplicity, it is assumed that the system consists of only two components C_1 and C_2.

3.1 Modeling of asynchronous discrete-event systems by standard automata

The component C_i $(i = 1,2)$ is modeled by the standard automaton $\mathcal{N}_i$, which is defined by the 4-tuple

$$\mathcal{N}_i = (\mathcal{N}_{z_i}, \Sigma_i, \delta_i, z_{0i}) \tag{19}$$

where $\mathcal{N}_{z_i}$ is the set of states, $z_{0i} \in \mathcal{N}_{z_i}$ the initial state and Σ_i the set of events. The state transition function $\delta_i : \mathcal{N}_{z_i} \times \Sigma_i \to \mathcal{N}_{z_i}$ specifies in each state z_i for all events σ_i the successor state z_i'

$$z_i' = \delta_i(z_i, \sigma_i). \tag{20}$$

For the composition of the two components the *parallel composition rule* is used. It yields the overall system model

$$\mathcal{N}_{\|} = (\mathcal{N}_{z_{\|}}, \Sigma_{\|}, \delta_{\|}, \boldsymbol{z}_{0\|}) \tag{21}$$

with the state $\boldsymbol{z}_{\|} = (z_1, z_2)^T \in \mathcal{N}_{z_{\|}} = \mathcal{N}_{z_1} \times \mathcal{N}_{z_2}$, the initial state $\boldsymbol{z}_{0\|} = (z_{01}, z_{02})^T$ and the set of events $\Sigma_{\|} = \Sigma_1 \cup \Sigma_2$. The introduction of the sets of common and private events

$$\Sigma^{com} = \Sigma_1 \cap \Sigma_2, \qquad \Sigma_i^{priv} = \Sigma_i \backslash \Sigma^{com},\ i = 1,2 \tag{22}$$

is needed to define the state transition function $\delta_{\|}$ of the overall system

$$\delta_{\|}\left(\begin{pmatrix} z_1 \\ z_2 \end{pmatrix}, \sigma\right) = \begin{cases} \begin{pmatrix} \delta_1(z_1,\sigma) \\ \delta_2(z_2,\sigma) \end{pmatrix}, & \text{if } \delta_1(z_1,\sigma)! \\ & \wedge\ \delta_2(z_2,\sigma)! \end{cases} \tag{23a}$$

$$\begin{cases} \begin{pmatrix} \delta_1(z_1,\sigma) \\ z_2 \end{pmatrix}, & \text{if } \sigma \in \Sigma_1^{priv} \\ & \wedge\ \delta_1(z_1,\sigma)! \end{cases} \tag{23b}$$

$$\begin{cases} \begin{pmatrix} z_1 \\ \delta_2(z_2,\sigma) \end{pmatrix}, & \text{if } \sigma \in \Sigma_2^{priv} \\ & \wedge\ \delta_2(z_2,\sigma)! \end{cases} \tag{23c}$$

$$\begin{cases} \text{undefined}, & \text{otherwise} \end{cases} \tag{23d}$$

that yields the successor state $\boldsymbol{z}_{\|}'$. The notation $\delta_i(z_i,\sigma)!$ means that the state transition function δ_i is defined for the arguments z_i and σ.

3.2 Interpretation of a network of I/O-automata as coupled standard automata

This section shows that the network of I/O-automata can be used to describe an asynchronous discrete-event system

modeled as the parallel composition of standard automata. As other composition rules can likewise be modeled by I/O-automata networks, the new model introduced in this paper is applicable at least for the same class of systems as known modeling methods that use standard automata.

To relate both modeling classes to one another, the events occurring in standard automata are interpreted as inputs to I/O-automata (like in the case where standard automata are used as acceptors of a language). It is assumed that the same input $v \in \mathcal{N}_{v_\parallel} = \mathcal{N}_{v_1} \cup \mathcal{N}_{v_2}$ is assigned to both components, which results in the special structure of the I/O-automata network shown in Fig. 5.

To re-build the parallel composition of two standard automata by the network depicted in Fig. 5, the interconnection signals s_i and r_i are used to enable or disable the movement of the automaton $\mathcal{N}_i$ by $\mathcal{N}_j$ and vice versa. Consequently, the interconnection inputs and outputs need to have two elements: $\mathcal{N}_{s_i} = \mathcal{N}_{r_i} = \{0,1\}$ $(i = 1,2)$. If $s_i = 1$ holds, the automaton $\mathcal{N}_i$ is allowed to move, which means that for this argument the state transition function G_i is defined. Otherwise $(s_i = 0)$, the movement of $\mathcal{N}_i$ is prohibited by automaton $\mathcal{N}_j$ with the consequence, that G_i is not defined for this argument. Analogously, $r_i = 0$ means that $\mathcal{N}_i$ sends a prohibition to $\mathcal{N}_j$ and to allow for the movement of $\mathcal{N}_j$ the automaton $\mathcal{N}_i$ must output $r_i = 1$.

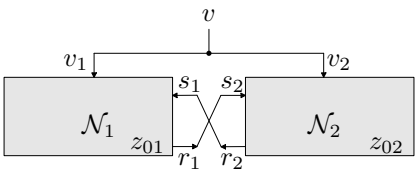

Fig. 5. Interpretation of a network of I/O-automata as coupled standard automata

The parallel composition rule (23) is realized by an appropriate choice of the interconnection function F_i of the I/O-automata as follows:

Definition 2. **Interconnection function for a parallel composition**
The interconnection function F_i^{PAC} is defined by

$$F_i^{PAC}(z_i, v_i) = r_i = \begin{cases} 1, & \text{if } G_i(z_i, v_i, 1)! \quad \vee\ v_i \notin \mathcal{N}_{v_i} \quad (24a) \\ 0, & \text{otherwise.} \quad (24b) \end{cases}$$

Interpretation of (24). The automaton $\mathcal{N}_i$ allows the movement of automaton $\mathcal{N}_j$ if its state transition function G_i is defined or if $v_i \notin \mathcal{N}_{v_i}$. The negation of this statement yields the condition for permitting the movement. Either G_i is not defined or the input symbol is not known to the automaton $\mathcal{N}_i$, i.e. its a private input symbol of the automaton $\mathcal{N}_j$.

As the network of I/O automata considered here has no control output, the automaton $\mathcal{N}_i$ is defined without an output function by the 7-tuple

$$\mathcal{N}_i = (\mathcal{N}_{z_i}, \mathcal{N}_{v_i}, \mathcal{N}_{s_i}, \mathcal{N}_{r_i}, F_i^{PAC}, G_i, z_{0i}). \quad (25)$$

Its structure is depicted in Fig. 6.

The I/O-automaton $\mathcal{N}_\parallel$ representing the parallel composition of $\mathcal{N}_1$ und $\mathcal{N}_2$ is defined by the 4-tuple

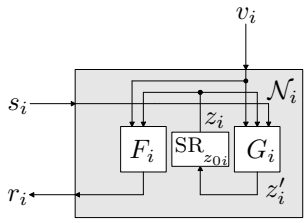

Fig. 6. Structure of the reduced I/O-automaton

$$\mathcal{N}_\parallel = (\mathcal{N}_{z_\parallel}, \mathcal{N}_{v_\parallel}, G_\parallel, \boldsymbol{z}_{0\parallel}) \quad (26)$$

with the set of states $\mathcal{N}_{z_\parallel} = \mathcal{N}_{z_1} \times \mathcal{N}_{z_2}$ and the initial state $\boldsymbol{z}_{0\parallel} = (z_{01}, z_{02})^T$. The state transition function of the network of I/O-automata

$$G_\parallel : \mathcal{N}_{z_\parallel} \times \mathcal{N}_v \to \mathcal{N}_{z_\parallel} \quad (27)$$

is obtained by the following procedure. First, the value of the interconnection outputs r_1 and r_2 are calculated by (24) in each state $\boldsymbol{z}_\parallel \in \mathcal{N}_{z_\parallel}$ for each input $v \in \mathcal{N}_{v_\parallel}$ and linked to the interconnection inputs as shown in Fig. 5. Next, the successor state $\boldsymbol{z}'_\parallel = (z'_1, z'_2)^T$ is calculated using (7). Hence, the state transition $\boldsymbol{z}'_\parallel = G_\parallel(\boldsymbol{z}_\parallel, v)$ is possible for the overall system.

3.3 Example

As a formal proof of the equality of both approaches cannot be given here due to space limitation, the comparison of the modeling formalisms is illustrated by an example (Fig. 7). If the automata graphs are interpreted for the standard automata, a directed edge from state z_i to z'_i labeled with σ_i is used for all possible state transitions $z'_i = \delta_i(z_i, \sigma_i)$. For the interpretation as I/O-automata, the edge is labeled with the pair v_i/s_i for all possible state transitions $z'_i = G_i(z_i, v_i, s_i)$. Hence, it contains in addition the value of the interconnection input.

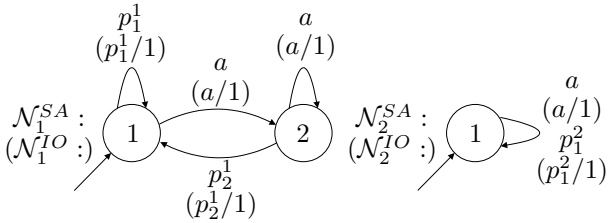

Fig. 7. Automaton graph of the standard automata $\mathcal{N}_1^{SA}$ and $\mathcal{N}_2^{SA}$ (and the I/O-automata $\mathcal{N}_1^{IO}$ and $\mathcal{N}_2^{IO}$)

The sets of events of the components are chosen to be

$$\Sigma_1 = \{a, p_1^1, p_2^1\} = \mathcal{N}_{v_1} \qquad \Sigma_2 = \{a, p_1^2\} = \mathcal{N}_{v_2}. \quad (28)$$

Note that common events have the same name and private events are denoted by p_x^i where the superscript i corresponds with the component C_i and the subscript $x = 1, 2, \ldots$ is used consecutively to enumerate the events of the component

$$\Sigma^{com} = \{a\} = \mathcal{N}_v^{com} \quad (29)$$

$$\Sigma_1^{priv} = \{p_1^1, p_2^1\} = \mathcal{N}_{v_1}^{priv} \quad \Sigma_2^{priv} = \{p_1^2\} = \mathcal{N}_{v_2}^{priv}. \quad (30)$$

The set of events of the overall system is given by

$$\Sigma_\parallel^{SA} = \{a, p_1^1, p_2^1, p_1^2\} = \mathcal{N}_{v_\parallel} \quad (31)$$

and the set of states $\mathcal{N}_{z_\parallel}$ by

$$\mathcal{N}_{z_\parallel} = \{(1,1)^T, (2,1)^T\} \quad (32)$$

with the initial state $\boldsymbol{z}_{0\|} = (z_{01}, z_{02})^T = (1,1)^T$.

The standard automaton $\mathcal{N}_{\|}^{SA}$ depicted in Fig. 8 is obtained from the application of the parallel composition rule (23) to $\mathcal{N}_1^{SA}$ and $\mathcal{N}_2^{SA}$.

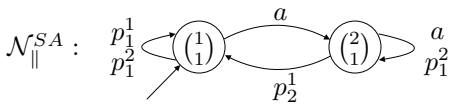

Fig. 8. Automaton graph of the composed standard automaton $\mathcal{N}_{\|}^{SA}$

The construction procedure to obtain the I/O-automaton $\mathcal{N}_{\|}^{IO}$ is shown step-by-step in Tab. 3. Comparing this results with the standard automaton depicted in Fig. 8 shows the equality of the two models.

Table 3. Construction of the composed I/O-automaton $\mathcal{N}_{\|}^{IO}$

$\boldsymbol{z}_{\|} = \binom{z_1}{z_2}$	v	$r_1 = s_2$	$r_2 = s_1$	$G_1(\cdot)!$?	$G_2(\cdot)!$?	$\boldsymbol{z}'_{\|} = \binom{z'_1}{z'_2}$	Entry in $G_{\|}^{IO}$
$(1,1)^T$	a	1	1	y	y	$(2,1)^T$	$\binom{2}{1} = G_{\|}^{IO}\left(\binom{1}{1}, a\right)$
$(1,1)^T$	p_1^1	1	1	y	n	$(1,1)^T$	$\binom{1}{1} = G_{\|}^{IO}\left(\binom{1}{1}, p_1^1\right)$
$(1,1)^T$	p_2^1	0	1	n	n	-	-
$(1,1)^T$	p_1^2	1	1	n	y	$(1,1)^T$	$\binom{1}{1} = G_{\|}^{IO}\left(\binom{1}{1}, p_1^2\right)$
$(2,1)^T$	a	1	1	y	y	$(2,1)^T$	$\binom{2}{1} = G_{\|}^{IO}\left(\binom{2}{1}, a\right)$
$(2,1)^T$	p_1^1	0	1	n	n	-	-
$(2,1)^T$	p_2^1	1	1	y	n	$(1,1)^T$	$\binom{1}{1} = G_{\|}^{IO}\left(\binom{2}{1}, p_2^1\right)$
$(2,1)^T$	p_1^2	1	1	n	y	$(2,1)^T$	$\binom{2}{1} = G_{\|}^{IO}\left(\binom{2}{1}, p_1^2\right)$

3.4 Discussion

The preceding subsection has shown that it is possible to define a special class of networks of I/O-automata introduced in Sec. 2 to rebuild the parallel composition rule known from modeling of asynchronous discrete-event systems by coupled standard automata. This fact points to three major advantages of the modeling formalism introduced in this paper in comparison to the standard automata modeling approach:

(1) The new modeling formalism can be proved to be powerful enough to cover all composition operators known from standard automata modeling (for a survey on composition operators see e.g. Wenck and Richter (2004)).
(2) The cause-and-effect-chain of the system under consideration is explicitly modeled by networks of I/O-automata. For standard automata modeling, the cause-and-effect-chain of the system is implicitly covered in the choice of the events.
(3) The way of modeling standard automata is a *top-down* approach. The most important modeling step and the major disadvantage with respect to I/O-automata modeling is the requirement that the events of the whole system have to be chosen and categorized from an overall system point of view. This violates the principles of component-oriented modeling. In contrast to this, the formalism introduced in Sec. 2 can be used in a *bottom-up* approach and purely component-oriented where the definition of the component models are done completely separately.

4. CONCLUSION

In this paper, a new component-oriented modeling formalism for asynchronous discrete-event systems has been introduced. The concept of I/O-automata for describing the components has been extended to model the interactions among the components by coupling signals. An interaction block $\boldsymbol{K}$ has been introduced to generally describe the structure of the system by linking the coupling signals. Hence, the resulting network of I/O-automata has a direct correspondence to the block diagram.

A special case of the network of I/O-automata has been derived to rebuild the parallel composition known from modeling discrete-event systems by standard automata. The equality of both approaches has been illustrated by an example.

In *future work*, both approaches will be formally proven to be equal for use with the parallel composition rule. Moreover, it will be shown that the new modeling formalism introduced in here is even more general then modeling asynchronous discrete-event system by standard automata because special cases have been used for the comparison and not the general form of the I/O-automata networks.

Since it is the author's aim to use the modeling formalism introduced in Sec. 2 for diagnosis, two *major extensions* will be made. First, vectorial coupling signals will be introduced to realize the most general form of interaction between the components of a technical system. Second, relations will be used instead of the functions F_i, G_i and H_i to cope with nondeterminism.

REFERENCES

M. Blanke, M. Kinnaert, J. Lunze, and M. Staroswiecki. *Diagnosis and Fault-Tolerant Control*. Springer-Verlag, 2006.

C. G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, 1999.

M. Heymann. Concurrency and discrete event control. In *IEEE Control Systems Magazine 10*, 1990.

E. A. Lee and P. P. Varaiya. *Structure and Interpretation of Signals and Systems*. Addison-Wesley, 2003.

J. Lunze. *Ereignisdiskrete Systeme*. Oldenbourg Verlag, 2006.

J. Neidig and J. Lunze. Direct feedback in automata networks. In *Proceedings of the 16th IFAC World Congress*, 2005.

R. Sreenivas and B. Krogh. On condition / event systems with discrete state realization. In *Discrete Event Dynamic Systems: Theory and Application*, 1991.

F. Wenck and J. H. Richter. A composition oriented perspective on controllability of large scale des. In *Proceedings of the 7th International Workshop on Discrete Event Systems (WODES)*, 2004.

W. M. Wonham. Supervisory control of discrete-event systems. Technical report, University of Toronto, Dept. of ECE, Systems Control Group, 2005.