Control Design for Nondeterministic
Input/Output Automata

Y. Nke∗, J. Lunze∗
∗ Ruhr-University Bochum, Institute of Automation and Computer
Control (e-mail: nke@atp.rub.de).

Abstract: This paper presents a control design approach for Input/Output (I/O) automata. The main contribution consists of an explicit description of the control design procedure, which leads to an implementation-friendly architecture. Moreover, the concepts of well-posedness of the control loop and controllability are investigated. The level control of a chemical process is used as a case study for demonstration purposes.

Keywords: Discrete-event systems, input/output automata, supervisory control.

1. INTRODUCTION

The aim of this paper is to propose a control design approach for discrete-event systems. The system class considered here is represented by nondeterministic Input/Output (I/O) automata \( N_p \) (Fig. 1). Since technological systems usually work under safety and operating constraints, the aspect of modeling a specification \( S \) is handled in the I/O automata framework proposed in Nke and Lunze (2010). The control design aim is to find a controller, which is likewise described by an I/O automaton \( N_c \) such that the plant depicted in Fig. 1 satisfies the specification \( S \). The controllability of a system \( N_p \) for a given specification \( S \) is investigated.

Fig. 1. Control loop of I/O automata

Literature review. Several approaches to control design exist in literature and a comparative overview is given in Pinzon et al. (1999). All these methods have the common goal that the controller is designed so as to suppress dangerous states or events in order to satisfy safety requirements. The differences to the approach proposed in this paper lie in the way how the controller is derived for operating constraints and how the specification is modeled.

A widely used control design approach for standard automata was developed by Ramadge and Wonham (1989), which is called in the following RW-Theor (RWT). Some RWT-based approaches for I/O automata were proposed by Balemi et al. (1993); Perk et al. (2008) and Petreczky et al. (2009) who pointed out some limitations of the RW-Theory for I/O automata, namely: the lack of automatic synthesis, a poor implementation structure and the high computational complexity.

The approach proposed here contributes to the automatic synthesis of a controller for a given specification due to the fact the control design algorithm and the realization scheme of the feedback controller provide sufficient informations for implementation purposes. The computational complexity is a major problem of discrete-event system in general and is not the concern of this work. However, the common issues and differences between the approach of this paper and the RWT-based ones are highlighted in the following and motivate the development of a new approach.

The main difference between the I/O automata handled by RWT-based approaches and those presented here is the interpretation of I/O transitions. Ramadge and Wonham (1989); Balemi et al. (1993); Perk et al. (2008); Cassandras and Lafortune (2008); Petreczky et al. (2009) consider an I/O transition as a succession of an input event \( \sigma_i \) and an output event \( \sigma_o \). An I/O transition always consists of a sequence \( \sigma_i \sigma_o \) which takes at least 2 state transitions and 3 states into account. In the framework of this paper, only one state transition and 2 states are taken into account for each I/O transition labeled \( v/w \), which is more compact.

The model of the control loop is usually represented in the RW-Theory by a synchronous product of the plant automaton with the supervisor. The approach proposed next does not provide a model of the control loop but strictly handles the plant \( N_p \) and the controller \( N_c \) as two well distinguished entities. Hence the specification defined here is not designated for the control loop but for the plant only. However, two key properties are required for the control loop: determinism and nonblockingness.

Another crucial difference is in the interpretation of the role of the controller. In the RWT, the controller is a supervisor which is supposed to enable or disable the controllable events of the plant in order to satisfy the specification. The objective of the controller developed here is to explicitly react to the output of the plant to fulfill the specification. It is important to note that the feedback connection between the RWT-based supervisor and the plant is considered only for design purposes. In fact the obtained supervisor is not in an explicit feedback connection with the plant but is merged with the plant by...
using the synchronous product or the parallel composition. The automaton composed in this way is a generator with the language of the plant under control. This is the reason why a classical RTW-based supervisor is usually not ready to be used for I/O automata in its original version and can not be directly used as a controller \( \mathcal{N}_c \) in Fig. 1. The so called strong model matching problem for I/O automata is studied in Barrett and Lafortune (1998). Roussel and Giua (2005); Cantarelli and Roussel (2008) recently proposed an approach to extract a controller as a Mealy automaton out of a RTW-based supervisor. In order to use this approach in the context of I/O automata, it would be necessary to first convert I/O automata into standard automata, then to translate the specification into a language, build the supervisor and finally extract the needed controller. Instead, a straight-forward method to get a controller which is ready to be used and easy to be implemented is proposed in this work.

**Aim of the paper.** This work proposes an approach to enforce the specified nominal behavior without considering faults as in previous works. In the following, the focus is put solely on relevant control design aspects. There are three main objectives pursued for this sake:

- the design of a discrete event feedback controller \( \mathcal{N}_c \)
- an explicit realization scheme of the controller \( \mathcal{N}_c \)
- a controllability condition for the existence of \( \mathcal{N}_c \).

Since only controllable and observable events are considered here, the notions presented in this work also hold for partially observable and partially controllable automata considered in the literature.

Section 2 presents basic notions of the approach. A chemical plant introduced in Section 3 is used as a case study. Section 4 briefly presents the specification modeling. The derived specification automaton is used in Section 5 for the control design method. Experimental results illustrate the applicability of the approach in Section 6.

### 2. PRELIMINARIES

#### 2.1 Nondeterministic I/O automata

A nondeterministic I/O automaton \( \mathcal{N} \) is defined as a tuple

\[
\mathcal{N} = (Z, V, \mathcal{W}, L, Z_0)
\]

where each element has the following meaning:

- \( Z \) – Set of states
- \( V \) – Set of control inputs
- \( \mathcal{W} \) – Set of control outputs
- \( L \) – Characteristic function
- \( Z_0 \) – Set of possible initial states.

The dynamics of the automaton is given by the function

\[
L : Z \times W \times Z \times V \rightarrow \{0, 1\}
\]

\[
L(z', w, z, v) = \begin{cases} 
1, & \text{if } (z', w, z, v)! \\
0, & \text{else} 
\end{cases}
\]

where \((z', w, z, v)!\) means that the transition is defined i.e the system \( \mathcal{N} \) moves from state \( z \) with the input \( v \) to state \( z' \) and generates the output \( w \). The \( * \) symbol in the argument of \( L(\cdot) \) means that any value of the corresponding element can be considered according to its symbol set. According to the context, the subscript \( x \) of \( \mathcal{N}_x \) will be replaced by a \( c \) for the controller, an \( s \) for the specification or a \( p \) for the plant. The sets

\[
V_{ax}(x) = \{ v_x \in V_x : L_x(*, *, z_x, v_x) = 1 \}
\]

\[
V_{ax}(x, z) = \{ v_x \in V_x : L_x(z', *, *, z, v_x) = 1 \}
\]

\[
W_{ax}(x) = \{ w_x \in W_x : L_x(*, w, z_x, v_x) = 1 \}
\]

\[
W_{ax}(x, z) = \{ w_x \in W_x : L_x(z', *, w, z, v_x) = 1 \}
\]

are called the active input set or the active output set of \( \mathcal{N}_x \) from a state \( z_x \) or a state couple \((z'_x, z_x)\), respectively.

The output generation of an automaton \( \mathcal{N} \) is said to be deterministic when the output \( w \) is unique for a given state-input combination \((z, v)\). An I/O automaton \( \mathcal{N} \) with a deterministic output generation is said to be \( W \)-deterministic. The characteristic function is used in the following to express this determinism concept.

**Lemma 1.** (\( W \)-determinism of \( \mathcal{N} \)). A nondeterministic I/O automaton \( \mathcal{N} \) is \( W \)-deterministic iff

\[
\forall (z, v) \in Z \times V, \exists! w \in W : L(*, w, z, v) = 1.
\]

#### 2.2 Blocking automaton

A nondeterministic I/O automaton \( \mathcal{N} \) blocks when the characteristic function returns zero for a given state-input combination \((z, v)\). A control loop (Fig. 1) is said to be blocking whenever any automaton of the control loop blocks, that is either the plant \( \mathcal{N}_p \), the controller \( \mathcal{N}_c \) or both. This matter is handled by the following definition.

**Definition 1.** (Blocking nondeterministic automaton). For a given input sequence \( V(0 \cdots k) \) and a state sequence \( Z(0 \cdots k) \), a nondeterministic I/O automaton \( \mathcal{N} \) is said to be blocking if

\[
\exists k \in [0, k_e] : L(*, *, Z(k), V(k)) = 0, Z(k) \in Z.
\]

Consequently \( \mathcal{N} \) is said to be nonblocking for a given input sequence \( V(0 \cdots k) \) whenever \( 7 \) does not hold, namely if there exist states and output sequences \( Z(0 \cdots k) \) and \( W(0 \cdots k) \) so that \( \forall k \in [0, k_e], \exists Z(k+1), W(k) \in Z \times W : \forall k \in [0, k_e], \exists Z(k+1), W(k) \in Z \times W : \)

\[
\bigwedge_{k=0}^{k_e} L(Z(k+1), W(k), Z(k), V(k)) = 1.
\]

**Blocking control loop and well-posedness.** A control loop blocks if the plant \( \mathcal{N}_p \) and/or the controller \( \mathcal{N}_c \) blocks at a given state \( z_c \) or \( z_p \) for a given input \( v_c \) or \( v_p \) w.r.t Definition 1. In other words, there exists a state combination \((z_c, z_p)\) for which a given input combination \((v_c, v_p)\) leads to a blocking plant automaton \( \mathcal{N}_p \), a blocking controller automaton \( \mathcal{N}_c \) or both.

**Well-posedness of a control loop.** The question of the well-posedness of a control loop becomes relevant when considering a plant with a controller in a feedback connection for which

\[
w_p = v_c \land w_c = v_p
\]

holds. An “algebraic loop” then emerges as follows. During an evaluation of the control loop of Fig. 1, \( L_p(z'_p, w_p, z_p, v_p) = 1 \) must hold for \( \mathcal{N}_p \) whereas \( L_c(z'_c, w_c, z_c, v_c) = 1 \) must hold for \( \mathcal{N}_c \). Hence,
The system of equations (10) reflects the situation where the plant and the controller should switch from the states \( z_p \) and \( z_c \) to the states \( z_p' \) and \( z_c' \) respectively. The tuple \((w_p, w_c)\) triggering this transition also solves the system of equations. \( w_p \) and \( w_c \) are fixed points of the algebraic loop (10). If the sets of these fixed points are denoted by \( \hat{W}_p \) and \( \hat{W}_c \) then they satisfy the relations

\[
\hat{W}_c \subseteq W_{ac}(z_c) \cap V_{ap}(z_p), (z_c, z_p) \in Z_c \times Z_p \tag{11}
\]
\[
\hat{W}_p \subseteq W_{ac}(z_p) \cap V_{ap}(z_c), (z_c, z_p) \in Z_c \times Z_p. \tag{12}
\]

A control loop consisting of a plant automaton \( N_p \) and a feedback controller automaton \( N_c \) is said to be well-posed if for every state combination \((z_c, z_p)\) there is exactly one input combination \((v_c, v_p) = (\hat{w}_p, \hat{w}_c)\) solving the algebraic loop equations. Nke et al. (2009) proposed the concept of weak well-posedness which is adapted to the control design discussed here. In the case of the weak well-posedness, the solution of (11) must be unique whereas the solution of (12) need not. A control loop is said to be ill-posed if any of the fixed points sets is empty. These concepts are summarized in the following lemma.

**Lemma 2.** (Well-posedness). A control loop which consists of a plant \( N_p \) and a feedback controller \( N_c \) is called

- well-posed iff \( |\hat{W}_c| = |\hat{W}_p| = 1 \), \( \tag{13} \)
- weakly well-posed iff \( |\hat{W}_c| = 1 \land |\hat{W}_p| > 0 \), \( \tag{14} \)
- and ill-posed iff \( |\hat{W}_c| = 0 \lor |\hat{W}_p| = 0 \). \( \tag{15} \)

**Well-posedness and blocking.** Now it is possible to make a statement on the blocking property of a control loop by means of the well-posedness introduced above. A control loop is nonblocking if it is either well-posed w.r.t. (13) or weakly well-posed w.r.t. (14). A control loop is said to be blocking whenever it is ill-posed w.r.t. (15).

### 3. CASE STUDY: LEVEL CONTROL OF A CHEMICAL PLANT

Consider the simplified chemical plant of Fig. 2 performing a mixture preparation process. The control objective of the process is to fill the tank TM from level 0 up to level 4, then down to level 1 and back to level 4 in a cyclic way. The valves V1, V2, and V3 are the relevant actuators of the system controlled by the signal \( v_p \). The input \( v_p = 0 \) models the “close all valves” command whereas \( v_p = i \) is the “open valve Vi and close the other valves” command. The states \( z_p \) of the plant \( N_p \) are modeled by the states of the tank TM which vary from 0 (empty) up to 5 (full). Five level sensors (LS) permit a discrete measurement which is the output \( w_p \) of the level of the tank TM at each step. \( w_p \) describes the result of an inflow or an outflow of the educt, so that \( w_p = z_p' \) holds. The I/O automaton labeled with \( v_p/w_p \) is obtained as depicted in Fig. 3.

### 4. SPECIFICATION

A specification \( S \) describes how the plant \( N_p \) should behave under the influence of the controller \( N_c \). \( S \) can be expressed as follows (Nke and Lunze (2010)).

\[
(9) \Rightarrow \begin{cases}
L_p(z'_p, w_p, z_p, w_c) = 1 \\
L_c(z'_c, w_c, z_c, w_p) = 1. \tag{10}
\end{cases}
\]
5. FEEDBACK CONTROL DESIGN

5.1 Main idea

Problem 1. (Control problem). Given a plant modeled by an I/O automaton \( N_p \) and a specification \( S \). Find a controller \( N_c \) for a given specification \( S \) with the following requirements:

1. Fulfillment of the specification \( S \).
2. Weak well-posedness of the control loop w.r.t. \((14)\).
3. \( W \)-Determinism of the controller w.r.t. Lemma 1.

In Section 5.5, the existence of a controller automaton \( N_c \) respecting these three requirements will be referred to as the controllability of an automaton \( N_p \) for a specification \( S \).

The specification \( S \) is implemented by the specification automaton \( N_s \). The main idea of the control design procedure is to keep the same structure of the specification automaton \( N_s \) but to reverse the input/output events \( v_s/w_s \) to get the controller automaton \( N_c \), i.e., \( v_s/w_c = w_s/v_s \). However, this straightforward approach does not reveal how to derive a control law and how to explicitly enforce a control output to the plant. Moreover, a method enabling how to get a controller out of a set of control laws is necessary and addressed in the following.

5.2 Procedural description of the design method

A control law automaton \( A_c^{(i)} \) \((i = 1 \ldots \nu)\) is introduced as a single path with a unique combination of the state sequence \( Z_s(0 \ldots k_c) \) and an output sequence \( W_c(0 \ldots k_c) \) through the controller \( N_c \). Hence a controller \( N_c \) may consist of several control laws \( A_c^{(i)} \). For a specified state sequence \( Z_s(0 \ldots k_c) \) which must be enforced by the controller, the number of possible control laws \( \nu \) is given by

\[
\nu = \prod_{k=0}^{k_c-1} \sum_{v_s} \sum_{w_s} L_s(z', w, v) \tag{18}
\]

where \( z' = Z_s(k+1), z = Z_s(k) \). Equation (18) computes the product of the number of possible transitions for each state combination \((z',z)\) of the sequence \( Z_s(0 \ldots k_c) \).

The design of an I/O controller automaton \( N_c \) \((Z_c, V_c, W_c, L_c, Z_0c)\) consists of the following steps:

1. Build the specification automaton \( N_s \) w.r.t. \((17)\).
2. Find every control law \( A_c^{(1)}, \ldots, A_c^{(\nu)} \) by means of

\[
A_c^{(i)} = \text{Con}(N_s, S) \quad \text{with } i = 1 \ldots \nu, \tag{19}
\]

where the Con() operator works according to Algorithm 1 described in Section 5.3.
3. Build the feedback controller \( N_c \) with the characteristic function.

\[
L_c(z', w_c, v_c) = \nu \sum_{i=1}^{\nu} L_c^{(i)}(z', w_c^{(i)}, v_c^{(i)}) = 1 \tag{20}
\]

\( N_c \) may become nondeterministic even though it fulfills \( S \), however the nondeterminism of \( N_c \) should not concern the outputs generation but the states transition only. That is \( N_c \) must be at least \( W \)-deterministic w.r.t. Lemma 1. However, the \( W \)-determinism of \( N_c \) does not guarantee the uniqueness of the control output.

5.3 Control design operator: Con

The focus of this section is to present the Con() operator for each specification type introduced in Section 4. The control law design problem is derived from Problem 1 for the first and third requirements as follows:

- Given:
  - A plant automaton \( N_p \)
  - A specification \( S \)
- Find: A control law \( A_c^{(i)} \) \((i = 1 \ldots \nu)\) which generates a unique output at each step \( k \) while fulfilling \( S \).

Since the control law \( A_c^{(i)} \) represents a unique state sequence and a unique output sequence in the controller \( N_c \), the Con() operator consists of a central routine which guarantees the uniqueness of these sequences. This routine generates every possible control law \( A_c^{(i)} \) related to a specified state sequence \( \tilde{Z}_s(0 \ldots k_c) \) (Algorithm 1).

Algorithm 1. Control law design for a state sequence \( \tilde{Z}_s(0 \ldots k_c) \)

Input: \( N_s, \tilde{Z}_s(0 \ldots k_c) \)

Init: \( i = 1 \), compute \( v \) with \((18), v w T u p e l L i s t_{v w} = [ ] \)

\[ z = \tilde{Z}_s(k), z' = \tilde{Z}_s(k+1) \]

\[ VW_{s,w,v} = V_{as}(z', z) \times W_{as}(z', z) \]

\[ for \ \forall v w T u p e l = [ , w T u p e l ] \]

\[ v = w T u p e l(1), w = w T u p e l(2) \]

\[ if \ \forall L_s(z', w, v) == 1 \]

\[ w T u p e l L i s t_{v w} = \begin{bmatrix} w T u p e l L i s t_{v w} \\
                           w T u p e l 
\end{bmatrix} \]

end if

end for

\[ for \ \forall v w T u p e l = [ , w T u p e l ] \]

\[ v = w T u p e l(1), w = w T u p e l(2) \]

\[ i = 1 \]

\[ while \ i \leq \nu \]

\[ if \ L_c^{(i)}(z', z, *), \]

\[ L_c^{(i)}(z', z, *) == 0 \]

\[ i = i + \nu T u p e l L i s t_{v w} \]

else

\[ i = i + 1 \]

end if

end while

end for

Output: \( A_c^{(i)} \) \((i = 1 \ldots \nu)\)

The Con() operator described in Algorithm 1 generates control laws \( A_c^{(i)} \) provided that the corresponding state sequences \( \tilde{Z}_s \) are given for the considered specification type. The following explains how to obtain such a state sequence for each specification type:

- Con(\( N_s, z_F \)): find all state sequences \( \tilde{Z}_s \) in \( N_s \) from \( Z_{0s} \) to \( z_F \).
For the realization of this controller, the initial set is assumed to be a singleton (see the shift register \( SR \) in Fig. 5). However an extension to controller automata with several initial states by the set of initial states \( Z_{0c} \) is possible. Accordingly, the states flowing among the components in Fig. 5 have to be replaced by corresponding states sets. In addition, the \( V_{w}^{(i)}(z', z) \) and \( W_{w}^{(i)}(z', z) \) need to be extended to states sets too in order to compute \( \bar{V}_c(k) \) or \( w_c(k) \), respectively.

In Fig. 5 the counter \( k_s \) describes the steps of the generator represented by \( \bar{Z}_s(0 \ldots k_s) \) which must be forced by \( A_c^{(i)} \). At step \( k_s = 0 \), the condition \( k_s > 0 \) does not hold thus the select input of the connected multiplexer is 0 and the value 1 goes the select input of the next multiplexer. This enables to read the value \( Z_s(1) \) from the generator \( \bar{Z}_s(0 \ldots k_s) \). At each step \( k_s \) it generates the next state \( Z_s(k_s + 1) \) which has to be reached by the plant at the next step \( k_s + 1 \). Hence it is stored in the shift register \( SR \) for the next step. In addition, it is used together with the actual state of the plant \( \bar{Z}_s(k_s) \) in order to generate the needed control output \( w_c(k) \) and estimate the expected control inputs \( \bar{V}_c(k) \) by means of (6) and (4) respectively. If the actual output of the plant \( w_p(k) \) belongs to the expected ones i.e. \( w_p(k) \in \bar{V}_c(k) \) then the internal counter \( k_s \) of the generator \( \bar{Z}_s(0 \ldots k_s) \) is incremented, then the signals \( Z_s(k_s + 1) \) and \( Z_s(k_s) \) are updated.

If an unexpected output \( w_p(k) \) is generated by the plant, then the condition \( w_p(k) \in \bar{V}_c(k) \) does not hold. Consequently, the counter \( k_s \) would not be incremented, the current states \( Z_s(k_s + 1) \) and \( Z_s(k_s) \) would not be updated hence the controller output \( w_c(k) \) would not change.

Note that the realization scheme presented above can be directly applied for specifications of Type \( Z_s \) since \( \bar{Z}_s(0 \ldots k_s) = Z_s(0 \ldots k_s) \). For other specification types, the generator \( \bar{Z}_s(0 \ldots k_s) \) is built w.r.t. the corresponding \( Z_s \) sequence of Section 5.3.

5.5 Controllability conditions

W-Determinism of the feedback controller. This section proposes an answer to the question: When is the output generation of the controller deterministic? The next theorem proposes a criterion based on the specification automaton (Nke and Lunze (2010)).

**Theorem 3.** (W-Determinism of \( N_c \)). For a given feasible specification \( S \), described by the automaton \( N_s \), there exists a W-deterministic controller \( N_c \) if

\[
\forall (z_s, w_s) \in Z_s \times W_s, |V_{ws}(z_s)|_{w_s} = 1. \tag{21}
\]

**Controllability.** The controllability of a plant for a given specification should describe the possibility to find a feedback controller with the requirements stated in Problem 1.

**Definition 2.** (Controllability of a plant). A plant \( N_p \) is said to be controllable w.r.t. a specification \( S \) iff there exists a W-deterministic feedback controller \( N_c \) which can steer the plant \( N_p \) in a way that \( S \) is fulfilled and the control loop is weakly well-posed.

Since the first step of the control design method is to build the specification automaton \( N_s \), a necessary condition for the controllability of a plant is the feasibility of the considered specification. In addition, Theorem 3 gives a necessary and sufficient condition on the specification automaton under which one can obtain a W-deterministic controller. Hence it is an additional condition for the controllability of a plant. This is summarized by the following theorem.

**Theorem 4.** (Controllability). A plant \( N_p \) is controllable w.r.t. a specification \( S \) iff \( N_c \) satisfies (21).

**Proof.** \((\Rightarrow)\) If (21) does not hold, then \( N_c \) is not W-deterministic. Thus \( N_p \) is not controllable w.r.t. \( S \) according to Definition 2.

\((\Leftarrow)\) According to Definition 2, if \( N_p \) is not controllable w.r.t. \( S \) then \( N_c \) is not W-deterministic (includes \( S \) being not feasible) or the control loop is not weakly well-posed. \( N_c \) being not W-deterministic means that (21) does not hold because of Theorem 3. The control loop not being weakly well-posed means that it is either well-posed or ill-posed. Without loss of generality, well-posedness is not possible without weakly well-posedness which is a relaxed version. The case of the ill-posedness described through
(15) contradicts (21) by means of (17), showing that (21) also does not hold in this case. ■

6. APPLICATION

Feedback controller design for the chemical plant. The state sequence to consider in order to use Algorithm 1 is \( Z_i(1...8) = (0,1,2,3,4,3,2,1) \). This results in 8 control law automata \( \mathcal{A}^{(i)} \) \( (i = 1...8) \). This is due to the fact that it is possible to use either valve V2 or valve V3 in 4 different states namely 1, 2, 3 and 4. \( \mathcal{A}^{(i)} \) is depicted in Fig. 6 as an example. The last step consists of building the controller automaton \( \mathcal{N} \) by means of (20). The resulting controller \( \mathcal{N} \) which is able to steer the plant \( \mathcal{N}_p \) so that \( S \) (16) is fulfilled, is depicted in Fig. 7.

Experimental results. The execution of the control loop is done w.r.t. the control law \( \mathcal{A}^{(1)} \) enforcing (16) which is loaded in the generator \( Z_i(0...k_e) \) of Fig. 5. Figure 8 illustrates the behavior of the plant under control of \( \mathcal{A}^{(1)} \) (Fig. 6). The first and second plots show how the controller alternates between the valves V1 and V2 through \( w_c = 1 \) to fill TM and \( w_c = 2 \) to empty it, whereas the valve V3 is not used. The third plot represents the states trajectory of the level TM as specified by (16). The last plot shows its continuous evolution.

Hence, these experimental measurements demonstrate the theoretical correctness of the feedback controller and its applicability in an industrial environment.

7. CONCLUSION

The approach presented in this paper provides a feedback control design method for plants described by nondeterministic I/O automata. A necessary and sufficient condition for the controllability of a plant w.r.t. a specification is proposed. This approach delivers an implementation-ready controller which clearly reflects the causality of technical systems. This advantage needs to be studied regarding the complexity of the algorithm enabling the computation of such a controller. In addition, the proposed realization architecture is suitable to develop reconfiguration techniques of discrete-event controllers in the presence of faults. The related fault-tolerant issues for discrete-event systems are subject of ongoing research.

REFERENCES


