Fault-Hiding Control Reconfiguration for a Class of Discrete Event Systems

Th. Wittmann ∗ J. H. Richter ** T. Moor ∗

∗ Friedrich-Alexander Universität Erlangen-Nürnberg
** Siemens AG, I IA ATS 43, Nürnberg

Abstract: Fault-hiding control reconfiguration aims at hiding a fault from the nominal controller while the reconfigured closed-loop system possesses admissible behaviour. The necessary degrees of freedom are created by placing a reconfiguration block between nominal controller and faulty plant. We aim at a guaranteed non-conflicting, complete and controllable behaviour of the self-reconfiguring closed-loop system, in particular for an arbitrary solution to the nominal control problem. Thereby, the nominal controller design and the design of the reconfiguration block are completely decoupled. This is desirable from a practical perspective, since in this way additional fault-tolerant control capabilities can be retrofit to an existing control system. In this paper, we propose a self-reconfiguring control architecture, state our reconfiguration problem in terms of finite languages and address the synthesis of discrete event dynamic reconfiguration blocks. To illustrate our results, we provide a running example.

Keywords: Discrete Event Systems, Fault-Tolerant Control, Control Reconfiguration, Fault-Hiding

1. INTRODUCTION

Industrial PLCs are programmed by experts in practice, and this is not expected to change in the near future. In this paper we thus introduce a concept for fault-tolerant control of discrete event processes that can coexist with manually programmed PLCs and that can be retrofit to existing controlled plants.

Inspired by the fault-hiding principle, we place a reconfiguration block between controller and faulty plant. Thereby, we influence the signals in the nominal closed-loop system in order to hide a fault from the controller. The reconfiguration block shall be inactive during nominal operation and take over control only after fault occurrences. Motivated from industrial applications, where the actual controller implementation is unknown, we demand the reconfiguration block to be functional for an arbitrary but specification-conforming controller.

Much research in recent years has focused on control reconfiguration, see [Blanke et al., 2006] and subsequent literature. In [Schmidt, 2012] control reconfiguration is considered in the context of reconfigurable machine tools. Fault-tolerant control based on control reconfiguration is reported in [Paoli et al., 2008, 2011, Nke and Lunze, 2013]. The fault-hiding principle was originally developed for linear continuous-time system [Lunze and Steffen, 2006] and later extended to selected classes of non-linear continuous-time systems [Richter et al., 2011]. However, to the authors best knowledge there is no approach to fault-hiding control reconfiguration in discrete event systems, which is our main contribution.

Our approach relies on the supervisory control theory (SCT), as proposed in [Ramadge and Wonham, 1987, 1989]. In this paper the nominal controller is a solution to a control problem under partial observation [Lin and Wonham, 1988], which we extend by an operator interface. Thereby, integrating a reconfiguration block to hierarchically structured control systems is simplified. Furthermore, we use fault-accommodating control [Wittmann et al., 2012] to address sensor, actuator and plant faults, thus we do not rely on a separate diagnosis or controller switching mechanism. Our design goals w.r.t. the self-reconfiguring closed-loop system are non-conflictingness, completeness, controllability and specification conformance. We set up a suitable candidate for the closed-loop system and thereof we extract an admissible reconfiguration block by projection.

Our main results consist in sufficient conditions for the synthesis of a reconfiguration block that ensures non-conflictingness, completeness, controllability and the conformance with the design specifications for an arbitrary nominal controller.

The remainder of this paper is structured as follows. A concise revision of formal languages and SCT is provided in Section 2. In Section 3 we state the nominal control problem and recall fault-accommodating control. In Section 4 we address fault-hiding control reconfiguration in discrete event systems, and illustrate our results using an example.

2. PRELIMINARIES

Let Σ be a finite alphabet, i.e., a finite set of symbols (or events) σ ∈ Σ. A finite sequence of symbols σi ∈ Σ, i ≤ n is called a string s = σ1σ2 ... σn, where n ∈ N. The length of a string s ∈ Σ* is denoted |s| ∈ N. The empty string ε is characterised by |ε| = 0. The Kleene-closure Σ* is the
set of all finite strings over the alphabet $\Sigma$ including the empty string.

If, for two strings $s, r \in \Sigma^*$, there exists $t \in \Sigma^*$ such that $s = rt$, we say $r$ is a prefix of $s$, and write $r \preceq s$. A formal language (or short a language) $L \subseteq \Sigma^*$ over $\Sigma$, is a subset of the Kleene-closure $\Sigma^*$. The prefix of a language $L$ is defined by $\text{pre} L = \{ r \in \Sigma^* | \exists s \in L: r \preceq s \}$. A language $L$ is prefix-closed if $\text{pre} L = L$.

The natural projection $p_o: \Sigma^* \to \Sigma_\text{c}$ with $\Sigma_\text{c} \subseteq \Sigma$, is defined iteratively: (1) let $p_o \epsilon = \epsilon$; (2) for an arbitrary $s \in \Sigma^*$ and $\sigma \in \Sigma$, let $p_o (s \sigma) = p_o (s) \sigma$ if $\sigma \in \Sigma_\text{c}$, or, if $\sigma \not\in \Sigma_\text{c}$, then $p_o (s \sigma) = p_o (s)$. The set-valued inverse $p_o^{-1}$ of $p_o$ is defined by $p_o^{-1}(r) = \{ s \in \Sigma^* | p_o (s) = r \}$ for $r \in \Sigma_\text{c}$. We use the convention that projections and inverse projections are denoted $p_o$ and $p_o^{-1}$, respectively, with a subscript to indicate the respective range and domain. When extended to languages, the projection distributes over unions, and the inverse projection distributes over unions and intersections.

The synchronous composition of two languages $L_1 \subseteq \Sigma_1^*$, $L_2 \subseteq \Sigma_2^*$ is defined by $L_1 \circ L_2 := (p_1^{-1} L_1) \cap (p_2 L_2)$, where $p_1$ and $p_2$ denote the natural projections from $\Sigma = (\Sigma_1 \cup \Sigma_2)^*$ to $\Sigma_2^*$ and $\Sigma_1^*$, respectively.

Given two languages $L, K \subseteq \Sigma^*$, and a set of uncontrollable events $\Sigma_{\text{uc}} \subseteq \Sigma$, we say $K$ is controllable w.r.t. $(L, \Sigma_{\text{uc}})$, if $K \subseteq \Sigma_{\text{uc}} \circ \text{pre} L \subseteq \text{pre} K \cap L$ [Ramadge and Wonham, 1987]. With $\Sigma_{\text{c}} \subseteq \Sigma$ the set of observable events, we say $K$ is prefix-normal w.r.t. $(L, \Sigma_{\text{c}})$, if $K \subseteq \text{pre} L \cap (p_1^{-1} p_1 \text{pre} K)$. A language $K \subseteq \Sigma^*$ is complete, if for all $s \in \Sigma^*$ there exists $\sigma \in \Sigma$ such that $s \sigma \in K$ [Kumar et al., 1992] and $\Sigma - \Sigma_{\text{c}}$ complete, with $\Sigma_{\text{c}} \subseteq \Sigma$. It's proof is omitted for brevity but available at request.

Fact 2. Given a nominal control problem $(L_N, E_N)$ and a candidate language $K_N \subseteq \Sigma^*_N$ satisfying

$\quad \Sigma_{\text{uc}} := \Sigma_{\text{uc}} \cup \Sigma_{\text{lo}}, \text{ i.e.}$

$\quad \text{pre} K_N \cap \text{pre} L_N \subseteq \text{pre} K_N$

$\quad (p_1^{-1} p_1 \text{pre} L_N) \cap \text{pre} K_N$

$\quad \text{Relative Closedness w.r.t. } L_N, \text{ i.e.}$

$\quad K_N = (p_1^{-1} p_1 \text{pre} L_N)$

$\quad \text{Completeness, i.e.}$

$\quad (\forall s \in \text{pre} K_N \exists \sigma \in \Sigma) [\exists \sigma \in \text{pre} K_N]$

$\quad \text{Specification Conformance, i.e. } K_N \subseteq E_N,$

then $H_N = \text{pre} K_N$ is a solution to the given nominal control problem. Conversely, if $H_N \subseteq \Sigma^*_N$ is a solution to the given nominal control problem, then the nominal closed-loop system $L_N \parallel H_N$ possesses the properties (K1)-(K5). Furthermore, given $H_N = \text{pre} K_N$, where $K_N$ denotes the suprema...
Example 1. Consider a machine that processes a single workpiece using one of two different modes. Process start and process completion are to be reported to the high-level operator. Table 1 summarizes the relevant events.

<table>
<thead>
<tr>
<th>event</th>
<th>semantics</th>
<th>alphabet</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1/a2</td>
<td>start process 1/2</td>
<td>$\Sigma_{\text{CON}}$</td>
</tr>
<tr>
<td>A1/A2</td>
<td>completion process 1/2</td>
<td>$\Sigma_{\text{UCON}}$</td>
</tr>
<tr>
<td>ah/Ah</td>
<td>report process start/completion</td>
<td>$\Sigma_{\text{HI}}$</td>
</tr>
</tbody>
</table>

An automaton representation of the nominal plant $L_N$ is drawn in Fig. 2. The nominal specification $E_N$ and the

Fig. 2. Nominal plant model $L_N$

minimally restrictive nominal controller $H_{N}^*$ are pictured in Fig. 3. Note that the semantics of the high-level events $ah$ and $A_h$ are defined by the specification $E_N$ and implemented solely by the controller $H_{N}^*$.

Fig. 3. Nominal specification $E_N$ and nom. controller $H_{N}^*$

3.2 Fault-Accommodating Modelling and Control

Fault-accommodating models [Wittmann et al., 2012] are a general modelling framework for systems subject to spontaneously occurring sensor, actuator and plant faults. We model a fault by a distinct low-level event $F$. Given a nominal plant model $L_N \subseteq \Sigma^*$ and a model of the fault’s history together with its impact on the plant $L_D \subseteq \Sigma_F^*$, with

$$\Sigma_{FP} := \Sigma_F \cup \{F\} = \Sigma_{\text{CON}}^* \cup \Sigma_{\text{UCON}} \cup \Sigma_{\text{HI}} \cup \Sigma_{\text{LO}} \cup \{F\},$$

then we call a pair $(L_N, L_D)$ a fault-accommodating model. With a given fault-accommodating model $(L_N, L_D)$ we associate the fault-accommodating behaviour $^2 $ $L_F = L_N \cup L_D$. We could show that

$$L_F = L_N F \Sigma_F^* \cap L_D$$

holds, given that $\text{pre } L_D \cap \Sigma_N^* \subseteq \text{pre } L_N$ and $L_D \cap \Sigma_N^* \subseteq L_N$.

Example 2. (Ex. 1 cont’d.) Both modes can be subject to a sensor fault, i.e. starting in mode 1 is confirmed with ending in mode 2. The fault-accommodating model $L_F$ is pictured in Fig. 4. Observe that the states 1,2 and 3 correspond to the nominal plant model but the fault may only occur in the states 2 and 3. The remaining states describe the fault’s impact on the plant.

Fig. 4. Fault-accommodating plant $L_F$

Using fault-accommodating models, the problem of fault-tolerant control can be reduced to a standard control problem, similar to the nominal control problem declared over a different overall alphabet

$$\Sigma_F := \Sigma_N \cup \{F\} = \Sigma_{\text{CON}} \cup \Sigma_{\text{UCON}} \cup \Sigma_{\text{HI}} \cup \Sigma_{\text{LO}} \cup \{F\}.$$ Formally, a fault-accommodating control problem is a pair $(L_F, E_F)$, where $L_F \subseteq \Sigma_F^*$ denotes a fault-accommodating plant model and $E_F \subseteq \Sigma_F^*$ denotes a fault-accommodating specification, and a solution to the fault-accommodating control problem is a fault-accommodating controller $H_F \subseteq \Sigma_{FC}^*$. $H_F$ is non-conflicting, complete, controllable w.r.t. $(L_F, \Sigma_{FC})$, $\Sigma_{FC} = \Sigma_{\text{UCON}} \cup \Sigma_{\text{LO}} \cup \{F\}$ and conformal with $E_F$. Note that we use $E_F$ to define the semantics of the operator interface in the case of a fault, see Ex. 3. The resulting closed-loop system is pictured in Fig. 5.

Example 3. (Ex. 2 cont’d). After a fault, the process at hand is no longer available and the controller may switch to the alternative mode without reporting back to the operator. The fault-accommodating specification $E_F$ and the respective minimally restrictive fault-accommodating controller $H_F$ are pictured in Fig. 6. Note that $H_F$ is admissible for the nominal as well as the faulty plant. In this example the specification is weak enough to retain the semantics of the high-level events.

In industrial practice an expert solution is replaced by a fault-accommodating controller, which is undesirable. However, following the concept of fault-hiding control reconfiguration introduced in the next section, we can benefit from expert experience up to the fault occurrence.

4. FAULT-HIDING CONTROL RECONFIGURATION

This section develops a formal framework for the design of a reconfiguration block $R$ that operates the fault-accommodating plant $L_F$ in a suitable way, while it mimics nominal plant behaviour w.r.t. the virtualised (decoupled) nominal controller $H_N$. The resulting self-reconfiguring closed-loop structure is pictured in Fig. 7.
The behaviour of the self-reconfiguring closed-loop system is given by $L$.

$$H_{v} \subseteq \Sigma_{V}^{*}$$

$$\Sigma_{VCON} \uplus \Sigma_{VUCON}$$

$$R \subseteq \Sigma_{R}$$

$$\Sigma_{CON} \uplus \Sigma_{UCON}$$

$$\Sigma_{LO}$$

$$\{ F \}$$

$$L_{v} \subseteq \Sigma_{v}^{*}$$

Fig. 7. Self-reconfiguring closed-loop system

To create the necessary degrees of freedom, we formally decouple the nominal controller from the fault-accommodating plant by introducing virtual events. Each virtual event has a unique physical counterpart with corresponding semantics. A virtual controller is defined over virtual events and the high-level events, but follows the nominal controller’s dynamical laws.

4.1 Self-reconfiguring Closed-Loop Structure

Formally, we introduce the set of virtual controllable events $\Sigma_{VCON}$ and the set of virtual uncontrollable events $\Sigma_{VUCON}$ disjoint with $\Sigma_{v}$ and in a bijective relationship with the corresponding virtualised specification $\rho(\Sigma_{CON}) = \Sigma_{VCON}$.

We extend this relationship to strings by the function $h : \Sigma^{*} \rightarrow \Sigma_{v}^{*}$, defined by $h(\epsilon) = \epsilon$ and

$$h(\sigma\sigma') = \begin{cases} h(\sigma)\rho(\sigma) & \text{if } \sigma \in \Sigma_{CON} \cup \Sigma_{UCON} \\ h(\sigma)\sigma' & \text{otherwise} \end{cases}$$

The extension of the function $h(\sigma)$ to languages is straightforward.

For a concise notation, we introduce the virtual overall alphabet $\Sigma_{v} = \Sigma_{VCON} \cup \Sigma_{VUCON} \cup \Sigma_{HI} \cup \Sigma_{LO}$, the virtual plant alphabet $\Sigma_{v}^{*} = \Sigma_{VCON} \cup \Sigma_{VUCON} \cup \Sigma_{LO}$, and the virtual controller alphabet $\Sigma_{VC} = \Sigma_{VCON} \cup \Sigma_{VUCON} \cup \Sigma_{HI}$.

Given a nominal controller $H_{n}$, a nominal plant model $L_{n}$ and a nominal specification $E_{n}$, we obtain the corresponding virtualised controller, the virtualised plant and the virtualised specification as follows:

$$L_{v} = h(L_{n}) \subseteq \Sigma_{v}^{*}$$

$$E_{v} = h(E_{n}) \subseteq \Sigma_{v}^{*}$$

$$H_{v} = h(H_{n}) \subseteq \Sigma_{VC}^{*}$$

The bijective relation between nominal events and their virtual counterparts, implies that $H_{v}$ solves the control problem $(L_{v}, E_{v})$ if and only if $H_{n}$ solves $(L_{n}, E_{n})$.

The virtual controller $H_{v}$ and the fault-accommodating plant $L_{v}$ are linked via a reconfiguration block $R \subseteq \Sigma_{R}$, where

$$\Sigma_{R} = \Sigma_{CON} \cup \Sigma_{UCON} \cup \Sigma_{VCON} \cup \Sigma_{VUCON}$$

The behaviour of the self-reconfiguring closed-loop behaviour is given by $L_{v} \parallel R \parallel H_{v}$.

4.2 Formal Reconfiguration Problem

We consider a reconfiguration block to be admissible if the resulting self-reconfiguring closed-loop system is non-conflicting, complete, controllable w.r.t. $L_{v} || H_{v}$ and if it satisfies a given specification language $E \subseteq \Sigma^{*}$, where

$$\Sigma = \Sigma_{CON} \cup \Sigma_{UCON} \cup \Sigma_{HI} \cup \Sigma_{LO} \cup \Sigma_{VCON} \cup \Sigma_{VUCON} \cup \{ F \}$$

We subdivide the specification $E$ according to $E = E_{v} || E_{n}$, with a fault-accommodating specification $E_{v}$ and a reconfiguration specification $E_{n}$. In this paper we use $E_{n}$ to enforce inactivity conditions as in [Richter, 2011], i.e. one-by-one event dispatching before the occurrence of a fault; formally,

$$E_{n} = \{ \{ \rho(\sigma)\sigma \mid \sigma \in \Sigma_{CON} \} \cup \{ \sigma\rho(\sigma) \mid \sigma \in \Sigma_{UCON} \} \cup \Sigma_{HI} \cup \Sigma_{LO} \}^{*} F \Sigma^{*}$$

Obviously, given a virtualised controller $H_{v}$ and a fault-accommodating plant $L_{v}$ the corresponding reconfiguration block is the solution to the control problem $(L_{v} \parallel H_{v}, E)$. However, in industrial practice the nominal control strategy is designed and implemented by human experts. Thus, we may not assume that a sufficiently precise controller model is available and we are faced with the design of a reconfiguration block that is functional for an arbitrary nominal controller.

Definition 3. A reconfiguration problem is a tuple $(L_{n}, E_{n}, L_{v}, E_{v}, E_{n})$, where $L_{n} \subseteq \Sigma_{n}$ is a nominal plant model, $E_{n} \subseteq \Sigma_{n}$ is a nominal specification, $L_{v} \subseteq \Sigma_{v}$ is a fault-accommodating plant model. Furthermore, $E_{v} \subseteq \Sigma_{v}$ is a fault-accommodating specification and $E_{n} \subseteq \Sigma_{n}$ is a reconfiguration specification, with $E = E_{v} || E_{n}$ relatively closed w.r.t. the formal plant $L_{v} \parallel H_{v}$, i.e. $E = (pre E) \cap L_{v} \parallel H_{v}$. A solution to a reconfiguration problem is a reconfiguration block $R \subseteq \Sigma_{R}$ with

$$(R0)$$ Closedness, i.e. $R = pre R$,

such that for an arbitrary $H_{v} = h(H_{n})$ where $H_{v} \subseteq \Sigma_{v}^{*}$ solves $(L_{v}, E_{v}, E_{n})$, the self-reconfiguring control loop $L_{v} \parallel R \parallel H_{v}$ satisfies

$$(RC1)$$ Nonconflictingness, i.e.

$$(pre L_{v}) \parallel R \parallel H_{v} = pre (L_{v} \parallel R \parallel H_{v})$$

$$(RC2)$$ Completeness, i.e.

$$(\forall s \in (pre L_{v}) \parallel R \parallel H_{v} \mid s \in \Sigma) \mid (s \in (pre L_{v}) \parallel R \parallel H_{v})$$

$$(RC3)$$ Controllability w.r.t. $(L_{v} || H_{v}, \Sigma_{uc})$, where

$$\Sigma_{uc} = \Sigma_{VCON} \cup \Sigma_{VUCON} \cup \Sigma_{HI} \cup \Sigma_{LO} \cup \{ F \}$, i.e.

$$(pre L_{v}) \parallel R \parallel H_{v} \Sigma_{uc} \cap (pre L_{v}) \parallel H_{v} \subseteq (pre L_{v}) \parallel R \parallel H_{v}$$

$$(RC4)$$ Specification Conformance, i.e. $L_{v} \parallel R \parallel H_{v} \subseteq E$. □

4.3 Reconfiguration Block Synthesis

Note that $H_{v}^{\Sigma}$ is an upper bound to all virtualised solutions $H_{v}$ to the nominal control problem $(L_{n}, E_{n})$. Since $H_{v}$ is unknown, we use $H_{n}$ instead. We interpret $L_{v} || H_{v}$ as a formal plant model controlled by the reconfiguration block $R$. However, a solution to the control problem $(L_{v} || H_{v}^{\Sigma}, E)$ is not necessarily a solution to the reconfiguration problem $(L_{n}, E_{n}, L_{v}, E_{v}, E_{n})$, since the liveness properties non-conflictingness and completeness are not guaranteed for an arbitrary nominal controller.

Lemma 4. Given is a reconfiguration problem $(L_{n}, E_{n}, L_{v}, E_{v}, E_{n})$ and the virtualised minimally restrictive solution

$$H_{v}^{\Sigma} = \Sigma_{VCON} \cup \Sigma_{VUCON} \cup \Sigma_{VCON} \cup \Sigma_{VUCON} \cup \{ F \} \Sigma^{*}$$

for $(L_{v} || H_{v}^{\Sigma}, E)$. □
Consider the reconfiguration block
\[ R = p_n \circ K. \]  
(3)

For any virtualised solution to the nominal control problem \( H_v \subseteq \Sigma_v \), the self-reconfiguring closed-loop system \( L_r \parallel R \parallel H_v \) possesses the properties (RC2)-(RC4). Furthermore, \( R \) is closed (R0).

Proof. We choose \( K \subseteq \Sigma^* \) with the properties (M1)-(M7) and start with the proof of two auxiliary statements
\[ \text{pre } K = (\pre L_r) \parallel R \parallel H_v^\prime, \]  
(4)
\[ L_r \parallel R \parallel H_v^\prime, \]  
(5)
proven by the deductions
\[ \text{pre } K = (\pre p_{vv} \circ \pre L_r) \cap (\pre p_v \circ H_v^\prime) \cap (p_n \circ K), \]
\[ = (\pre p_{vv} \circ L_r) \cap (\pre p_v \circ R) \cap (p_v \circ H_v^\prime), \]
\[ = (\pre L_r) \parallel R \parallel H_v^\prime, \]
\[ = (\pre L_r) \parallel R \parallel H_v^\prime, \]
\[ = L_r \parallel R \parallel H_v^\prime. \]

Ad (R0): (R0) directly follows from the definition of \( R \).

In order to establish (RC2)-(RC4), we choose an arbitrary virtualised solution \( H_v \subseteq \Sigma_v \) to the nominal control problem \( (L_v, E_v) \).

Ad (RC2): Picking an arbitrary \( s \in (\pre L_r) \parallel R \parallel H_v \), we need to establish the existence of a \( \sigma \in \Sigma \) such that \( \sigma \in (\pre L_r) \parallel R \parallel H_v \). In order to structure the following proof we introduce the set of events enabled by \( H_v \), \( \gamma_H = \{ \sigma \in \Sigma_v \mid \pre p_v (\sigma) \in H_v \} \), and the set of events simultaneously enabled by \( L_r \) and \( R \), \( \gamma_{LR} = \{ \sigma \in \Sigma_v \cup \Sigma_{vv} \mid \pre p_{vv} (\sigma) \in \pre L_r \text{ and } (p_v \circ \sigma) \in R \} \). Since \( H_v \) solves a standard control problem, the closed-loop system \( L_v \parallel H_v \) is complete. Referring to Eq. (4), completeness (M4) of \( K \) implies \( \gamma_{LR} \neq \emptyset \). We distinguish the following cases:

Case 1a \( (\gamma_H \cap \Sigma_v) \neq \emptyset \), \( H_v \) can evolve independently of \( L_r \): Picking \( \sigma \in (\gamma_H \cap \Sigma_v) \), we obtain \( \sigma \in (\pre L_r) \parallel R \parallel H_v \), as \((\Sigma_v \cup \Sigma_{vv}) \cap \Sigma_H = \emptyset \) holds, which implies (RC2). Thus, for the sequel we may assume \( \gamma_H \subseteq \Sigma_v \cup \Sigma_{vv} \).

Case 1b \( (\gamma_{LR} \cap \Sigma_{vv}) \neq \emptyset \), \( L_r \parallel R \) can evolve independently of \( H_v \): Pick \( \sigma \in (\gamma_{LR} \cap \Sigma_{vv}) \), we obtain \( \sigma \in (\pre L_r) \parallel R \parallel H_v \), as \( \Sigma_v \cap \Sigma_{vv} = \emptyset \) holds, which implies (RC2). Thus, for the sequel we may assume \( \gamma_{LR} \subseteq \Sigma_v \cup \Sigma_{vv} \).

In summary, we have shown that Case 1 implies completeness whenever \( L_r \) and \( R \parallel H_v \) can evolve independently. It remains to study cases where \( L_r \parallel R \) and \( H_v \) must evolve together, namely \( \gamma_{LR} \cap \Sigma_H \) is nonempty.

Case 2a \( (\gamma_H \cap \Sigma_{vv}) \neq \emptyset \): Assume that we can choose \( \sigma \in (\gamma_H \cap \Sigma_v) \). Observe that \( \pre p_v (\sigma) = \pre p_v (s) \in L_r \) and \( \pre p_v (\sigma) \in H_v \), the latter one from \( H_v \subseteq H_v \). Thus, controllability (M1) implies \( \sigma \in \pre K \) and we conclude \( \sigma \in (\pre L_r) \parallel R \parallel H_v \), in particular \( \sigma \in \gamma_{LR} \) which implies (RC2). The remaining case is characterised by \( \gamma_H \subseteq \Sigma_v \cup \Sigma_{vv}, \) i.e. the controller waits for an uncontrollable event.

Case 2b \( (\gamma_{LR} \cap \Sigma_{vv}) = \emptyset \): Together with \( \gamma_{LR} \subseteq \Sigma_v \cup \Sigma_{vv} \) this case implies \( \gamma_{LR} \subseteq \Sigma_v \). Referring to Eq. (4) and \( H_v \subseteq H_v \), we have \( s \in \pre K \). From (M6) we obtain \( \pre K \subseteq \pre p_v \circ R \parallel L_r \). Since \( H_v \) solves a standard control problem for the plant \( L_r \), closed-loop completeness (NC2) implies the existence of \( \sigma \) such that \( \pre p_v (\sigma) \in \pre L_r \), and \( \pre p_v (\sigma) \in \pre H_v \). In particular, we must have \( \sigma \in \Sigma_v \cup \Sigma_{vv} \), i.e. the controller waits for a sensor event. Hence, weak sensor consistency (M5) implies that there exits \( t \in (\Sigma_v \cup \Sigma_{vv})^* \) such that \( \sigma t \sigma' \in \pre K \) for some \( \sigma' \in \Sigma_v \cup \Sigma_{vv} \). Referring to Eq. (4), we have \( (\pre p_v \circ \sigma t \sigma') \in \pre L_r \) and \( (p_v \circ \sigma t \sigma') \in R \). Then, \( \gamma_{LR} \subseteq \Sigma_v \cup \Sigma_{vv} \) implies \( t = \epsilon \) and we obtain \( \sigma' \in \gamma_{LR} \). Since \( p_v (s) \in (\pre L_r) \parallel H_v \) and the nominal closed-loop system \( L_v \parallel H_v \) is controllable w.r.t. \( (L_v, \Sigma_v) \) (from NC3) we obtain \( p_v \circ \sigma t \sigma' \in (L_r \parallel H_v) \), in particular \( \sigma \in \pre p_v \circ L_r \). This concludes the proof of completeness (RC2).

Ad (RC3): Pick any \( s \in (\pre L_r) \parallel R \parallel H_v \) and any \( \sigma \in \Sigma_{vv}, \) with \( \Sigma_{vv} := \Sigma_H \cup \Sigma_{vv} \cup \Sigma_v \cup \Sigma_{vv} \cup \Sigma_v \), such that \( s \in (\pre L_r) \parallel H_v \). From the supremality of \( H_v \), we have \( s \in (\pre L_r) \parallel H_v \) and together with (Eq. 4) we obtain \( s \in (\pre L_r) \parallel R \parallel H_v \subseteq (\pre L_r) \parallel R \parallel H_v \), which implies (RC3).

Ad (RC4): To establish (RC4), note that from the supremality of \( H_v \) and Eq. (5) we have \( L_r \parallel R \parallel H_v \subseteq (L_r \parallel H_v) \parallel K \). Since \( K \) is relatively closed w.r.t. the plant \( (M3), \) together with (M7), we have \( K \subseteq (E \cap (L_r \parallel H_v)) \). Since \( E \) is relatively closed w.r.t. \( L_r \parallel H_v \), we obtain \( K \subseteq E \), concluding the proof of specification compliancy (RC4).
4.4 Result Validation

Example 4. (Ex. 3 cont’d). In the first step we virtualise the minimally restrictive solution to the nominal control problem to obtain $H^*_V$, see Fig. 8.

As an additional design requirement we impose the inactivity condition $E_R$ according to Eq. 2, see Fig. 9.

The supremal sublanguage $K$ of $L_V \parallel H^*_V$ w.r.t. (M1)-(M7) is non-empty and the respective reconfiguration block $R = p_R \preceq K$ is pictured in Fig. 10.

In order to evaluate the behaviour of the self-reconfiguring closed-loop system we choose a restrictive solution $H_V$ to the nominal control problem according to Fig. 11. The behaviour of the resulting self-reconfiguring closed-loop system $L_V \parallel R \parallel H_V$ is shown in Fig. 12.

All languages in Example 4 are closed, thus from Corollary 5 we can guarantee admissible behaviour of the self-reconfiguring closed-loop system for an arbitrary virtual solution to the nominal control problem.

Consider the self-reconfiguring closed-loop system in Fig. 12. During nominal operation (states 1-6) the reconfiguration block is inactive and nominal control commands are dispatched one-by-one. After the fault the reconfiguration block mimics a functional nominal plant by generating the events $a1v$ and $A1v$ and thus hides a fault from the virtual controller.

From a theoretical perspective, we have established a framework for fault-hiding control reconfiguration for discrete event systems based on SCT. Our main results are sufficient conditions for the synthesis of a reconfiguration block, guaranteeing admissible behaviour of the reconfiguring closed-loop system for an arbitrary nominal controller. Due to fault-accommodating models we do not depend on external diagnosis or controller switching mechanisms but our results are effectively restricted to closed languages. Current research addresses extension of the fault-hiding control reconfiguration framework for not necessarily closed plants, and its experimental evaluation.

From a practical perspective, fault-tolerant control capabilities can be retrofit to existing control systems, even if they are hierarchically structured. Since our reconfiguration block works with an arbitrary nominal controller, its design is independent of the actual nominal controller implementation: a desirable property for industrial applications.

REFERENCES