Application of a Model-based Reconfiguration Approach for the ISS COLUMBUS Environmental Control and Life Support System (ECLSS)

Benjamin Kelm, Kaja Balzereit, Lukas Moddemann, Stephan Myschik, Oliver Niggemann

Application of a Model-based Reconfiguration Approach for the ISS COLUMBUS Environmental Control and Life Support System (ECLSS)

Benjamin Kelm[1] and **Kaja Balzereit**[2] and **Lukas Moddemann**[3]
Stephan Myschik[1] and **Oliver Niggemann**[3]
[1] Institute of Aeronautical Engineering, Universität der Bundswehr, Munich, Germany
e-mail: firstname.lastname@unibw.de
[2] Fraunhofer Center for Machine Learning, IOSB-INA, Lemgo, Germany
[3] Institute of Automation Technology, Helmut Schmidt University, Hamburg, Germany

Abstract

Cyber-Physical Systems (CPS) are subject to various faults due to failing actuators, sensors or structural components. The increasing size and complexity of modern systems result in cost- and time-intensive manual fault handling. To enable systems to adapt to faults autonomously, reconfiguration, i.e. the identification of a new valid configuration that recovers operation, is necessary.

This paper presents an extension of the recently published reconfiguration algorithm *AutoConf* and the application to the Environmental Control and Life Support System (ECLSS) of the COLUMBUS module aboard the ISS. The implementation draws on a qualitative system model formulated in propositional logic. The corresponding satisfiability problem is solved by a state of the art SAT-Solver. The extension consists of three contributions, namely a health status implementation, a dynamic causal graph and a problem-specific formulation of serial dependencies of the actuators. Both a static and dynamic evaluation (integrated simulation) of the extended reconfiguration algorithm is presented for 73 fault cases, covering a wide range of faults.

1 Introduction

The Environmental Control and Life Support System (ELCSS) of the International Space Station (ISS) can be classified as a Cyber-Physical System, since its mechanisms and operational modes are controlled and monitored by an algorithm. Although there are Fault Detection, Isolation and Reconfiguration (FDIR) procedures implemented on the component level, a system-wide automatic fault-handling of ECLSS is nonexistent, requiring quick and continuous engineering support. Despite being designed as a very robust system, during the past decade operation was interrupted because of insufficiently handled sensor, structure and actuator faults. Since ECLSS is of vital importance for the astronauts and experiments aboard the ISS, it is worthwhile to investigate, whether the safety and reliability can be increased by a system-wide reconfiguration approach.

To assess this hypothesis the project (K)ISS[1] implements a software stack of Anomaly Detection, Diagnosis and Reconfiguration for the automatic fault handling.

The task of reconfiguration is to transfer the system from an invalid to a valid configuration, that is to recover a system from a fault by automatically adapting the system configuration so that operation within the specification of the system can be maintained [1].

This paper presents the extension and application of the recently published algorithm *AutoConf* by Balzereit and Niggemann [2] for automated reconfiguration to the ECLS System. The algorithm performs a reconfiguration in two steps: First a logical formula which represents the reconfiguration problem is created, which is then solved using a satisfiabilty (SAT) solver.

AutoConf requires a qualitative process model in the form of a causal graph which can be translated into propositional logic. With the description of the system goals (e.g. maintaining a specified temperature within the cabin) and further constraints the problem of reconfiguration can be transferred into a satisfiability (SAT) problem, i.e. finding an assignment of variables that satisfy the logical formula. A reconfiguration typically needs to be identified from a large parameter space. Since the choices are usually binary, the search space is increasing exponentially. To solve the formula and find a new configuration efficiently a state of the art SAT solver (Z3, Microsoft Research) [3] is used. The choice for this solver is mainly due to its wide user base and support of SMT, which might be used when expanding the algorithm.

AutoConf implements such an approach, yet exhibits a few limitations when applied to the specific system at hand. Some of these shall be addressed as adaptations in this paper, which lead to the following research questions:

RQ1: Which extensions to the existing reconfiguration algorithm AutoConf are necessary to accurately model the multi-physical system ECLSS? Usually a reconfiguration task is triggered by an anomaly detection followed by a diagnosis, which identifies the faulty component. This component is no longer available to the system, and thus, should be excluded from the new configuration. Additionally, in physical processes, the impact of a control variable typically depends on the state variables it is influencing. The flow direction through a valve, e.g., generally depends on the direction of the pressure differential. Finally, serial dependencies that require a set of control variables (e.g. valve positions) to be set to achieve a certain objective are commonly found in real world systems.

RQ2: How can a (quantitative) simulation be integrated

[1] The project (K)ISS is part of dtec.bw®
https://dtecbw.de/home/forschung/hsu/projekt-kiss

for a dynamic evaluation of the (qualitative) reconfiguration? A static evaluation of a reconfiguration solution does not account for inherent dynamics of the system. To validate the correct system response to a given fault, the reconfiguration algorithm is integrated into a physical simulation of ECLSS.

The contribution of this paper is threefold:

1. Showcasing the application of *AutoConf* to a real-world system, addressing specific constraints and challenges.
2. Further development and extension of *AutoConf* to support actuator health statuses, dynamic causal graphs and serial dependencies.
3. Evaluating the resulting reconfiguration algorithm within a simulation of the ECLS system.

This paper is structured as follows: First, the related work is discussed and our approach is classified within the FDIR and control research area. Next, we will provide a description of the system at hand, namely ECLSS, and the simulation model thereof, which will be used for the dynamic validation of the reconfiguration algorithm. Thirdly, the extensions to the existing *AutoConf* algorithm will be presented, followed by the evaluation, where both the static as well as dynamic results will be discussed. The article ends with a conclusion and an outlook on possible future work.

2 Related Work

This section provides an overview over literature in related research areas, namely fault tolerant control and qualitative approaches to reconfiguration.

2.1 Fault Tolerant Control

Fault-Tolerant Control (FTC) is concerned with designing controllers that maintain system operation even in the presence of faults. The system dynamics are modeled using ordinary differential equations; a controller uses this model to continuously calculate new system inputs that establish desired system behavior [4]. Blanke et al. [1] separated FTC techniques into *robust* and *adaptive* control. Robust controllers contain a single controller whose parameters are chosen to handle as many faults as possible. In case of a fault, the controller is not adapted but an a priori choice of controller shall enable tolerance towards most faults. Adaptive controllers, on the other hand, change their parameters with the faulty situation. Thus, the plant's behavior is adapted to the faulty situation and the effects of the fault may be mitigated. For this purpose, the system behavior needs to be estimated. Ma et al. [5] estimated the system dynamics using fuzzy logic.

However, as FTC mainly operates on quantitative models, its applicability towards hybrid systems is limited: discrete system behavior coming from different discrete operation modes is expensive, as for each discrete mode, a new control needs to be defined. This makes handling unforeseen faults nearly impossible. In addition, major faults requiring structural system adaptations cannot be handled by FTC properly.

2.2 Qualitative Approaches

Qualitative Simulation is concerned with the estimation of future system behavior, given a qualitative system description. Instead of ordinary differential equations, information about the monotony of system variables and landmark values, i.e. values representing significant system regions, are used [6].

Crow and Rushby [7] established the research area of model-based reconfiguration. They extended Reiter's diagnosis algorithm [8] towards the identification of a system adaptation for discrete systems. This idea was taken up in further research and led to an integration of AI-based diagnosis in control approaches [9].

Similarly, Blanke et al. [1] emphasized the need for reconfiguration as an automated adaption of a system's controller, to enable efficient fault handling.

One approach combining AI-based reconfiguration and modern control theory has been published by Balzereit and Niggemann [2]. Their algorithm *AutoConf* is based on a satisfiability solver identifying an input mask, that enables cyber-physical production systems (CPPS) to maintain production in the presence of faults. Since their approach has been developed for CPPS, this work extends it to further application areas.

3 ECLS System Description and Model

The COLUMBUS module is the biggest contribution of the European Space Agency (ESA) to the International Space Station. The purpose of COLUMBUS is to serve as a unique platform for different fields of research: Human physiology, biology, fundamental physics, material sciences and fluid physics. Furthermore, external experiment facilities allow the long-term and non-perturbed observation of the Earth and the universe. The European laboratory is operated by the COLUMBUS Control Center at the German Space Operations Center nearby Munich [10].

3.1 ECLS System Overview

The most critical and vital system of the COLUMBUS module is the Environmental Control and Life Support System (ECLSS), whose topology is shown the process flow diagram in figure 1. It consists of a supply (ISFA) and return (IRFA) fan assembly, a redundant pair of cabin fan assemblies (CFA 1/2), a temperature control valve (TCV), which distributes the airflow into two redundant cooling and condensation cores (Core 1 and 2) within the condensate heat exchanger (CHX) to cool and dehumidify the air.

The airflow is then channeled into the cabin, where it mixes with the cabin air. To refresh the air and ensure smoke detection, a minimum volumetric flow rate has to be passed by the smoke detectors (SD 1/2) and is returned by the ISFA and recycled in part through the CFAs. The thermal control system (TCS) is composed of the Cores, the coolant and external heat exchangers and is controlled by the redundant cabin temperature control units (CTCU 1/2).

Additionally, there are multiple sensors, measuring the volumetric airflow (AFS), pressure differentials across fans and filter (ΔP or DPS), partial pressure of O_2 and of CO_2 gas (PPOS/PPCS), cabin temperature (CTS 1-6), humidity (HS 1/2) and the total pressure (TPS 1-4). In the following section, we will define a theoretical model to represent ECLSS.

3.2 Theoretical Model - Hybrid Automaton

ECLSS is a mixed discrete-continuous system - exhibiting analog and continuous properties combined with digital, discrete controls. It can thus be modeled as a hybrid automaton $\mathcal{H}$, which is defined below.

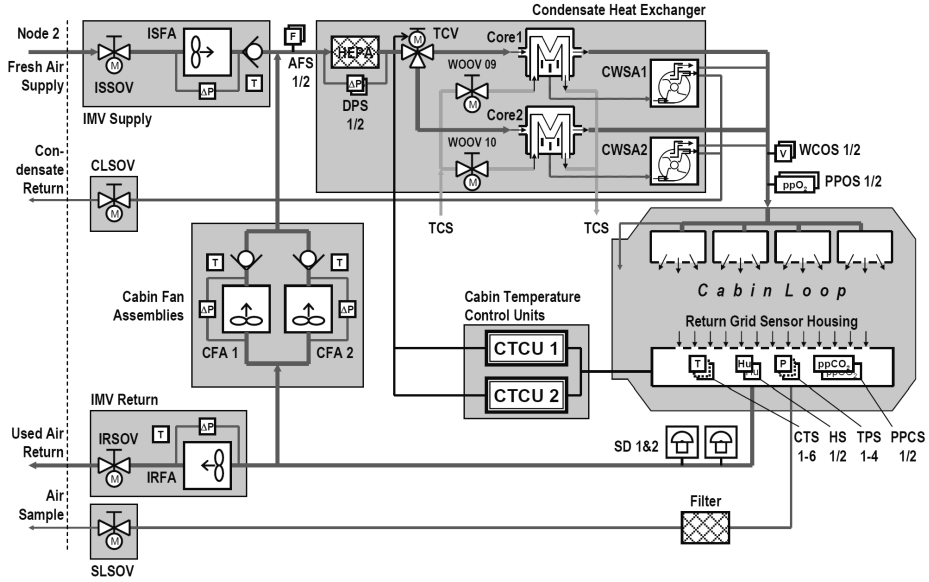

Figure 1: Process flow diagram of the ISS ECLS System (Cabin Loop) from Doyé [10]

Definition 1. *$\mathcal{H}$ shall be defined (according to McIlraith et al. [11]) by the tuple $(I, X, \boldsymbol{x}^0, \mathcal{F}, \Sigma, \Phi)$ where:*

- $I = \{\boldsymbol{i}_1, ..., \boldsymbol{i}_k\}$ is the set of input variables of size k. In this work, I only takes binary values. Every combination of inputs $\boldsymbol{i}$ defines a *control mode* $\mu \in \mathcal{M}$,
- $X \subset \mathbb{R}^n$ describes the set of continuous state variables, influenced by the binary inputs I. The initial state is expressed by $\boldsymbol{x}^0$.
- $\mathcal{F} = \{f_{\mu_1}, ..., f_{\mu_m}\}$ is a finite set of functions describing the system dynamics in each control mode over time $t \in \mathbb{R}$.
- $\Sigma = \{\sigma_1, ..., \sigma_p\}$ is the set of discrete *actions* that transitions the system between control modes
- $\Phi : \Sigma \times \mathcal{M} \times X \to \mathcal{M} \times X$ is the a transition function that maps an action, mode and state into a new mode and initial state.

The figure 2 below shows the graph of a simple hybrid automaton for illustration purposes.

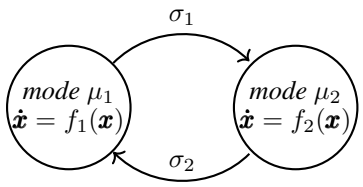

Figure 2: A simple hybrid automaton.

Definition 2. *A configuration of the hybrid system is defined as the tuple $(\boldsymbol{x}, \boldsymbol{i})$ of input and state variables.*

A CPS is usually operated by a control program P, which adjusts the inputs $\boldsymbol{i}$ to match the states $\boldsymbol{x}$ to specified reference values $\boldsymbol{w}$ [1].

Definition 3. *A configuration is valid if the deviation from the reference value $\boldsymbol{\epsilon} = |\boldsymbol{w} - \boldsymbol{x}|$ remains within certain state limits $\Delta\boldsymbol{\epsilon}$.*

Given a valid configuration the system satisfies its *system goal* via a control program. If the limits are violated due to a plant component fault, the configuration is *invalid*, preventing the system goal to be reached and requiring a *reconfiguration*.

Definition 4. *Given an invalid configuration $(\boldsymbol{x}^0, \boldsymbol{i}^0)$ reconfiguration is a function $f_R : I_R \to I_R$ so that the configuration $(\boldsymbol{x}, f_R(\boldsymbol{i}^0))$ is valid within a specified reconfiguration time Δt.*

4 Development and Application of the Extended Solution Algorithm

This section covers the brief presentation of the reconfiguration algorithm *AutoConf* and the algorithmic extension needed for the application to ECLSS.

4.1 *AutoConf* — The Original Approach

AutoConf was recently presented by Balzereit and Niggemann [2] and is divided into two steps: In the first step a logical formula which represents the reconfiguration problem is created. In the second step, this formula is solved by a SAT solver.

For the first step of creating the logical formula, also called *qualitative system model* (QSM), causal graphs G are required. This causal graph qualitatively defines which inputs affect which states of the system. As *AutoConf* only deals with binary inputs (e.g. valve opened or closed), we will define these as $B = \{\boldsymbol{b}_1, ..., \boldsymbol{b}_k\}$. The causal graph is first divided into two subgraphs of positive $G^+ = (V, E^+)$ and negative $G^- = (V, E^-)$ influences on the state variables. The nodes of the graphs consist of the states and inputs $V = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_n, \boldsymbol{b}_1, ..., \boldsymbol{b}_k\}$. The edges in the positive graph E^+, connecting inputs and states, indicate a significant *increase* of that state when that input is activated: $E^+ = \{(\boldsymbol{b}_j, \boldsymbol{x}_i) \,|\, j \in \{1, ...k\}, i \in \{1, ...n\}\}$. Similarly the edges of the negative graph E^- correspond to a significant decrease in the state variable. The causal graph thus integrates a qualitative description of the system dynamics (e.g. opening a valve will increase the respective state in the connected reservoir).

The algorithm then traverses the graph, encoding the causality into propositional logic. Therefore the state limits $\Delta\boldsymbol{\epsilon}$ are transferred into symbols low_{x_i} and $high_{x_i}$, being true if x_i is below the lower limit or above the upper limit. These, respectively, *imply* certain inputs to be activated or deactivated. For example, if a reservoir exceeds its limit, the formula will imply an opening of an outflow *or* a closing of an inflow. These constraints are formulated using the binary logical connectives (negation [$\neg$], conjunction [$\wedge$] and disjunction [$\vee$]).

In the second step, a logical SAT solver is used to solve the logical formula, utilizing logical reasoning. If the formula is *satisfiable* then there exists an assignment of input variables, that yields the formula true. The assignment corresponds to the new configuration needed to achieve a valid system state within a specified reconfiguration time Δt. If the formula is not satisfiable, then a reconfiguration cannot be performed and the system might be shut down. Thus, given a causal graph and a system observation, the reconfiguration of the system can be handled automatically.

4.2 *AutoConf_extd* — Extension of *AutoConf* for Real-World Systems

The application of *AutoConf* to ECLSS - a real-world system - gives rise to certain extensions of the algorithms to adequately reconfigure the system. Therefore, we will first define the (continuous) system states $\boldsymbol{x}$ and the (binary) input variables $\boldsymbol{b}$ for ECLSS, consisting of the following components (cf. section 3.1):

$$\boldsymbol{x} = [T_c, \phi_c, \dot{V}_{AFS}, p_c]^T \tag{1}$$

$$\begin{aligned}\boldsymbol{b} = [&b_{\mathrm{ISFA}}, b_{\mathrm{IRFA}}, b_{\mathrm{CFA}_1}, b_{\mathrm{CFA}_2}, ... \\ &b_{\mathrm{TCV}_1}, b_{\mathrm{TCV}_2}, b_{\mathrm{C}_1}, b_{\mathrm{C}_2}]^T.\end{aligned} \tag{2}$$

It is important to note, that the temperature control valve (TCV), which splits the airstream into the two cores, is handled as two binary inputs. The activation of a certain input corresponds to a specified maximum actuation, e.g. for the fans that might be the maximum continuous speed according to specifications. The reference values with permissible deviation of the states (under normal operation) are given by

$$\boldsymbol{w} \pm \Delta\boldsymbol{\epsilon} = \begin{bmatrix} 295\,\mathrm{K} & -4\,\mathrm{K} & +3\,\mathrm{K} \\ 0.5 & -0.2 & +0.2 \\ 500\,\mathrm{m^3/h} & -50\,\mathrm{m^3/h} & +1000\,\mathrm{m^3/h} \\ 101.3\times10^3\,\mathrm{Pa} & -1300\,\mathrm{Pa} & +1700\,\mathrm{Pa} \end{bmatrix} \tag{3}$$

The following presents the three main extensions necessary for a valid reconfiguration strategy.

Health status implementation

AutoConf does not include a component health status implementation. Yet, this is a vital functionality for determining whether and how a system can be reconfigured. For example, a broken actuator cannot be used for reconfiguration. Since we can manipulate the system only through the actuators (input variables), we limit our component health status to those. The health status will be implemented as an additional constraint on the existing formula. We define a boolean *health status vector* $\boldsymbol{h}$ corresponding to the input vector in (2) as

$$\begin{aligned}\boldsymbol{h} = [&h_{\mathrm{ISFA}}, h_{\mathrm{IRFA}}, h_{\mathrm{CFA}_1}, h_{\mathrm{CFA}_2}, ... \\ &h_{\mathrm{TCV}_1}, h_{\mathrm{TCV}_2}, h_{\mathrm{C}_1}, h_{\mathrm{C}_2}]^T.\end{aligned} \tag{4}$$

The additional constraint added to the logical formula, called *System Model* (SM), is presented below.

Algorithm 1 *AutoConf_GenSM_healthStatus*

1: **Input:** $X, B, low, high, POS, NEG, SM$
2: **Output:** SM
3: **for** $j \in \{1, 2, ...k\}$ {for every input $\boldsymbol{i}_k$} **do**
4: $\quad SM := SM \cup (\boldsymbol{h}_j \Rightarrow \boldsymbol{b}_j)$
5: **end for**
6: **for** $l \in \{1, 2, ...n\}$ {for every state variable $\boldsymbol{x}_l$} **do**
7: $\quad SM := SM \cup \left[\boldsymbol{high}_l \Rightarrow \bigvee_{\boldsymbol{b}_j \in NEG_l} (\boldsymbol{b}_j^0 \wedge \boldsymbol{h}_j) \Rightarrow \boldsymbol{b}_j)\right]$
$\quad$ {constraint to inhibit closing outflows}
8: $\quad SM := SM \cup \left[\boldsymbol{low}_l \Rightarrow \bigvee_{\boldsymbol{b}_j \in POS_l} (\boldsymbol{b}_j^0 \wedge \boldsymbol{h}_j) \Rightarrow \boldsymbol{b}_j)\right]$
$\quad$ {constraint to inhibit closing inflows}
9: **end for**
10: **return** SM

Given the state X and input I variables, the state sets of deviated state variables low and $high$, as well as the positive POS and negative NEG sets of inputs and the system model, the line 4 of algorithm 1 ensures that only healthy inputs are being used, whereas inputs are automatically closed for the new configuration. The next two lines inhibit a closing of outflows (negative input), in case of a high system state (cf. line 7) and a closing of inflows, where a low system state (cf. line 8). The conjunction with the health status variable enables faulty inputs to be closed regardless of this additional constraint.

Dynamic causal graph

ECLSS features intensive (pressure, temperature, humidity) and a extensive state variable (volumetric flow rate). For intensive state variables inflows can have both a positive or negative influence. For example, is the inflow temperature from the ISS higher than within the COLUMBUS cabin, the increase of the ISFA fan speed, will yield in a temperature rise in the cabin. To accomodate intensive state variables, we will define the incidence matrix dynamically. Alternatively, there is the possibilty to alter the state variables to the corresponding extensive state variables (e.g. instead of the temperature (intensive) the enthalpy (extensive) of a system might be considered). Formally, the causal graph is altered depening on the state of the system and the inflow conditions. An edge is part of the positive subgraph $e_i \in E^+$ if it contributes significantly, i.e. above a certain threshold defined by expert knowledge, to the increase of the respective state variable.

Serial actuator dependencies

The third extension of *AutoConf* deals with the serial dependencies of actuators. The TCV for example has to allow for airflow to pass through in order for the cooling cores (Core 1/2) to be able to transfer heat. These additional constraints are highly dependent on the specific system topology. For this paper, an individual treatment will suffice, although an generalization in the form of a dependency matrix might be beneficial for more complex systems. The first serial constraint pertains to the need for a minimum airflow through the cabin. Either ISFA or the recirculation fans (CFAs) need to be active and the TCV must be opened to either position.

$$[b_{\mathrm{ISFA}} \vee (b_{\mathrm{CFA}_1} \vee b_{\mathrm{CFA}_2})] \wedge (b_{\mathrm{TCV}_1} \vee b_{\mathrm{TCV}_2}) \tag{5}$$

The second serial constraint links the TCV position and the cores together via the implication, that a use of a cooling core requires the TCV to pass air though it.

$$(b_{C_1} \Rightarrow b_{\mathrm{TCV}_1}) \vee (b_{C_2} \Rightarrow b_{\mathrm{TCV}_2}) \tag{6}$$

With the extensions described above, the algorithm, which we will denote as *AutoConf_extd*, is able to model ECLSS sufficiently for the task of reconfiguration. In the following we will show the application to an example fault case to illustrate the process.

4.3 Application of *AutoConf_extd* to an Example Fault Case

For illustration purposes we will consider one hypothetical failure case: Suppose an accident during an experiment within the COLUMBUS module occurs. As a consequence the cooling core 1 fails and due to gas leakage the pressure in the cabin has increase beyond the threshold. Also, the hatch has been closed following the accident. The state of the system before reconfiguration is given by the system state

$$\begin{aligned} \boldsymbol{x}^0 &= [T_c, \phi_c, \dot{V}_{AFS}, p_c]^T \\ &= [303\,\mathrm{K}, 0.50, 500\,\mathrm{m}^3/\mathrm{h}, 103.5 \times 10^3\,\mathrm{Pa}]^T \end{aligned} \tag{7}$$

and the input configuration

$$\begin{aligned} \boldsymbol{b}^0 = [&b_{\mathrm{ISFA}}, b_{\mathrm{IRFA}}, b_{\mathrm{CFA}_1}, b_{\mathrm{CFA}_2}, \dots \\ &b_{\mathrm{TCV}_1}, b_{\mathrm{TCV}_2}, b_{\mathrm{C}_1}, b_{\mathrm{C}_2}]^T \\ &= [1,0,0,1,1,0,1,0]^T. \end{aligned} \tag{8}$$

We thus have only ISFA, CFA2 and one cooling branch (TCV1, C1) activated, which corresponds to the default configuration, where the used air is returned over the hatch opening.

We also find, by an underlying fault diagnosis algorithm, that two actuators have failed. The health state is given by

$$\boldsymbol{h}^0 = [1,1,1,0,1,1,0,1]^T. \tag{9}$$

With the above inputs, the reconfiguration algorithm first classifies the inputs into in- and outflows via the dynamic incidence matrix. These are then formulated via the *AutoConf_extd* into a logical set of formulas, which incorporates the following idea: "Which inputs do I need to open or close to bring the corresponding state within acceptable bounds?" The following line is an excerpt of the full logical formula.

$$T_{high} \Rightarrow \neg b_7^0 \wedge b_7 \vee \neg b_8^0 \wedge b_8 \vee \neg b_1^0 \wedge b_1, \tag{10}$$

It shows the implications of a high temperature, which are to switch on either one of the cooling cores (b_7 or b_8) or to switch on the ISFA fan. The negation of the pre-reconfigured inputs (b^0) is necessary to exclude from the search space those inputs which are already reconfigured. Please note, that since we're dealing with high temperature fault case, the dynamic incidence matrix will assign the colder inflowing air via the ISFA a positive value, modeling a sink.

These implications are set up for all state variables and for both cases - one for the violation of the lower and upper boundary. The serial actuator dependencies are also added to the logical formula to refine the model and internal flow structures.

This logical formula is then handed over to Z3 and checked for satisfiability. If it is satisfiable, a model can be retrieved, i.e. an input assignment, that satisfies the logical formula. In this sample fault case, the logical formula is satisfiable and the algorithm proposes a new input configuration to recover the system

$$\boldsymbol{b} = [1,1,0,0,0,1,0,1]^T. \tag{11}$$

By switching on the ISFA, the pressure can be reduced, and by acticating the second cooling branch (TCV2 and C2) the temperature can be lowered. Note that in this implementation, there is no guarantee for minimal cardinality of the solution, since the input space is rather small.

If the logical formula would not be satisfiable, a shut down of the system is invoked. Alternatively, it would be possible to lower the system requirements by removing constraints, thus prioritizing certain state variables.

5 Results

To validate the reconfiguration algorithm and system model, we will present both a static as well as a dynamic evaluation.

5.1 Static Evaluation

The static evaluation of the reconfiguration algorithm developed above applied to ECLSS will highlight two aspects of the result. First, we will present the results for a comprehensive fault case list in some detail. In a second part, we will compare these results to the original formulation of *AutoConf* and classify deviations of reconfigurability.

Reconfiguration with *AutoConf_extd*

Table 1 shows the results for 73 selected fault cases. The fault list is split into categories of single faults, affecting only one state or actuator (input), double and multiple faults. The faults are further differentiated into types of state limit violations (e.g. due to leakage or external disturbances), actuator faults (e.g. due to valves being stuck) and combined faults (state limit violation and actuator fault). For each fault the number of cases (# **cases**) and the number of reconfigurable faults (# **rcfg.**) is given. It is important to note, that all reconfigurable faults were succesfully identified and reconfigured by *AutoConf_extd* according to the definition of a valid configuration (cf. definition 3). For the other cases, insufficient redundancy prevents a reconfiguration to a valid system state.

Table 1: Static reconfiguration results for the *AutoConf_extd* algorithm for 73 selected faults

Fault category	**# cases**	**# rcfg.**
Single	**19**	**15**
Limit violation	9	7
Actuator fault	5	5
Combined faults	5	3
Double	**38**	**29**
Limit violation	12	11
Actuator fault	15	11
Combined faults	11	7
Multiple	**16**	**9**
Sum total	**73**	**53**

Although the fault case list is not exhaustive, it does cover a wide range of faults. About $78\,\%$ of the single faults are reconfigurable. Some single faults (such as a low temperature) are not directly reconfigurable, since ECLSS provides

Table 2: Comparison of reconfiguration results for the *AutoConf_extd* vs. *AutoConf* algorithm - with and without health status.

		AutoConf _extd	*AutoConf* with health status			
Fault category	**# cases**	**# rcfg.**	**# TP**	**# TN**	**# FP**	**# FN**
Single	**19**	**15**	**11**	**4**	**4**	**0**
Limit violation	9	7	5	2	2	0
Actuator fault	5	5	5	0	0	0
Combined	5	3	1	2	2	0
Double	**37**	**29**	**11**	**5**	**19**	**2**
Limit violation	12	11	5	1	5	1
Actuator fault	14	11	4	2	7	1
Combined	11	7	2	2	7	0
Multiple	**16**	**9**	**2**	**2**	**11**	**1**
Sum total	**73**	**53**	**24** *33%*	**11** *15%*	**34** *47%*	**3** *4%*
		AutoConf _extd	*AutoConf* without health status			
Sum total	**73**	**53**	**18** *25%*	**9** *13%*	**45** *63%*	**0** *0%*

no direct way to heat the incoming air. It is important to note, that the failure of any actuator can be handled, since the system exhibits a minimum twofold redundancy.

The double faults constitute the largest category, of which about $76\,\%$ are reconfigurable. The double actuator and combined faults exhibit lower reconfigurability, since the failure of two actuators of the same type (e.g. both cooling cores) may cause the loss of that function. Understandably, the multiple fault cases have the lowest reconfigurability of about $56\,\%$, since multiple state limit violations and actuator faults restrict the solution space severely. Since the application at hand is a unique system, no direct comparison or benchmark is applicable, which makes it hard to assess the performance of the algorithm. However, in the next section we will show the relative performance increase by comparing the extended algorithm to the original formulation.

Comparison of the extended and original algorithm

Table 2 compares the results of the extended algorithm *AutoConf_extd* with the original algorithm *AutoConf*, both *with* and *without* the health status implementation. The deviations are shown as a binary classification with respect to the correct fault reconfiguration result by *AutoConf_extd*. The columns (from left to right) denote the fault category, the number of cases, the number of reconfigurable cases (baseline for comparison), and the binary classification of the deviations pertaining to the original algorithm, consisting of True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN).

For the classification, first the original algorithm *AutoConf* is solved for each fault case and a new configuration (input variable assignment) is obtained. The configuration is then assigned to the extended algorithm formula *AutoConf_extd*. If the extended formula is satisfiable, the original algorithm has found a valid configuration and the result is a True Positive (TP). If it is not satisfiable, *AutoConf* has found a invalid configuration - a False Positive (FP). If neither of the formulas is satisfiable, we obtain a True Negative (TN), and if the extended algorithm found a valid configuration, but the original algorithm has not, we obtain a False Negative (FN).

The comparison for the original algorithm *with* health implementation is presented in detail for each fault category. Overall, about $49\,\%$ of the fault cases were identified correctly as either reconfigurable (TP) or unreconfigurable (TN). Yet, a large part ($51\,\%$) of the input assignments by *AutoConf* do not satisfy the extended formula, since they neither take the dynamic causal graph nor the serial actuator dependencies into account (cf. section 4.2). The number of False Positives is especially high for the multiple fault category, since the limited solution space causes a more frequent violation of the serial actuator dependencies. False Negatives (FN) are almost non-existent since the original algorithm is less constraining towards a solution.

The comparison for the original algorithm *without* health implementation shifts the imbalance further toward False Positives, since the formula does not take faulty actuators into account. The total number of misclassified faults is about $63\,\%$, indicating a significantly improved fault handling by the extended algorithm.

5.2 Dynamic Evaluation

This section deals with the dynamic evaluation of the reconfiguration algorithm. The results show the effects of the reconfiguration algorithm on a continuous simulation of ECLSS. First, the simulation architecture and implementation are presented and then the system response for certain faults is assessed.

Simulation Architecture

Figure 5 shows the architecture of the ECLSS simulation within MATLAB/Simulink using the Simscape library for physical modeling. The airflow through the components is modeled as moist air (MA), implementing real gas properties with partial water loading. The coolant is modeled as a thermal liquid. The signals on the bottom left are the control inputs I for the actuators (i.e. valves and fans). The individual components of ECLSS are modeled as follows: On the left , the ISS node 2 is implemented as a reservoir, setting constant boundary conditions. The fans (ISFA, IRFA and CFAs) are modeled by a constant volumetric flow rate sources with a flow resistance as a bypass to account for pressure potential equalization. The condensate heat exchanger (CHX) transfers the heat of the moist air flow to the coolant. If the dewpoint is reached, water will condensate on the cooling cores, lowering the absolute water loading of the air. The sources on the right implement a constant heat and moisture injection into the COLUMBUS cabin. Finally, the output X on the right reports the state of the system.

Figure 3 shows the ECLSS time response to the reconfiguration algorithm. The reconfiguration step is called every $10s$. The fault-case specific values for the states and inputs are initialized in the beginning. The selected fault-case is of the category double limit violation - the cabin temperature T_{cab} is too low and the relative humidity Φ_{cab} too high. The state limits are indicated by dashed horizontal lines.

Dynamic System Response to Reconfiguration

The input reconfiguration sequence for all 8 inputs is shown in figure 4. In the first call of the reconfiguration, the algorithm finds a new valid configuration by activating both ISFA and TCV_1. Following this initial reconfiguration, the algorithm fails to further reconfigure the system. Although there is an insufficient airflow $\dot{V}_{cab}$ and the CFAs could be activated to resolve that, this solution is not found. The underlying logical formula is unsatisfiable, because the first two states *block* a solution for the third state - all possible inputs have been already assigned, yet the state still remains

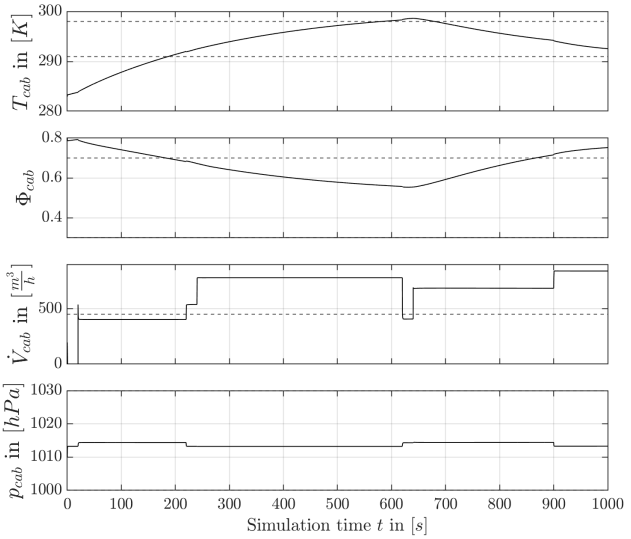

Figure 3: Simulation of ECLSS response to reconfiguration of fault-case #30, double limit violation (low T_{cab} + high Φ_{cab})

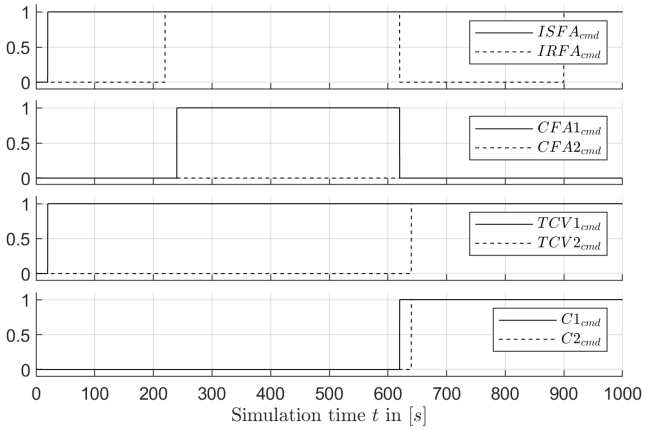

Figure 4: Input reconfiguration sequence of fault-case #30, double limit violation (low T_{cab} + high Φ_{cab})

beyond its limit, yielding the whole formula unsatisfiable. Until at approx. $t = 200s$ both the temperature and humidity enter into their valid range and the algorithm promptly activates the CFA1 and IRFA. This behavior is seen also for other fault-cases. One solution to this behavior would be a partial building of the logical formula, updating only those elements, which have not yet been reconfigured. Another solution would be to increase the reconfiguration time Δt to allow enough time to return to a valid system state.

At time $t = 640s$ the temperature exceeds the upper limit and both cooling cores are activated. However, also CFA1 and IRFA are deactivated again, causing the airflow to fall below its limit. This is a undesirable behavior caused by a questionable selection of the state variable, the volumetric airflow $\dot{V}_{cab}$. The volumetric airflow does not satisfy the thermodynamic definition of a *state variable*, rather it is known as a *process variable* [12]. A jump in the input causes, with almost no delay, a jump in the output, so that $\dot{V}_{cab}$ does not exhibit accumulative behavior. The problem of a low volumetric airflow is solved at the next reconfiguration step by the activation of TCV2. One solution for this issue is to choose another state variable to monitor the airflow, for example the kinetic energy of the flow within the cabin. A transfer of the state limits and required sensors however requires further expert knowledge and makes the application less straightforward. Another solution could be the explicit declaration of state and process variables and differentiating the logical propositions related to these.

5.3 Limitations

The evaluations above have exposed the limitations of the developed algorithm and propositional logic based approaches in general. The static evaluation has demonstrated the strong dependency on a valid system model. If system constraints are overlooked, a high number of misclassified reconfigurations can be the result (cf. table 2). The manual effort required to generate fine-tuned qualitative system model, including additional constraints, increases with complexity and size. Here, it could be beneficial to deploy system identification methods based on symbolic regression to find the qualitative system dynamics. Recent machine learning algorithms for sparse dynamic identification, like SINDYc [13], show promising results for the data-driven learning of input-output relations as needed for the incidence matrix.

The dynamic evaluation has shown two limitations of the current implementation. The first limitation is a *blocking behavior* and arises, when a state is out of limit, but the system has already been reconfigured. For the part of the formula dealing with that state, no new configuration can be found, which will yield the whole logical formula *unsatisfiable*. Although another state might be reconfigurable, the part of the formula corresponding to the first limit violation is blocking the solution. This only occurs with consecutive execution of the algorithm and can be solved relatively easy by a piece-wise generation of the formula. The second limitation discovered in the dynamic evaluation is the necessity of well-defined state variables. Process flow variables like $\dot{V}_{cab}$ do not exhibit accumulative behavior and thus must be handled differently.

6 Conclusion and Outlook

Cyber-Physical Systems are subject to various faults, such as broken actuators, leakages or strong external forcing. Nowadays fault handling is mostly done manually which is time and cost intensive. That is especially true for the ever-growing size and complexity of modern systems. However, in many cases, the system goal could still be reached by adapting the configuration of the system. Whereas control can only deal with faults forseen at design time, reconfiguration enables systems to adapt to unforseen faults. [2]

This article presents an extension of the recently published reconfiguration algorithm *AutoConf* and it's application to the safety-critical Environmental Control and Life Support System of the COLUMBUS module aboard the ISS. The extension consists of three contributions, namely a health status implementation, a dynamic causal graph and a problem-specific formulation of serial dependencies of the actuators (cf. *RQ1*). These extensions allow for a sufficiently precise qualitative system model (QSM) representing the real-world system ECLSS. To validate the extended algorithm *AutoConf_extd* a static and dynamic evaluation was performed on a fault case list with 73 entries covering

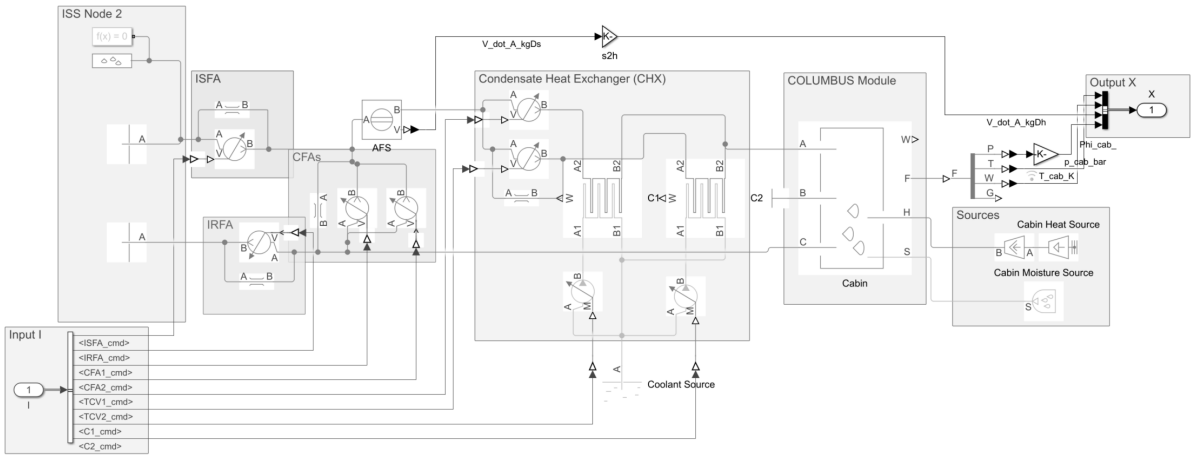

Figure 5: Simplified moist air process flow diagram of the ECLSS model - implemented in MATLAB/Simulink Simscape

a wide range of faults. The static evaluation shows a significant improvement in the reconfiguration of faults over the original algorithm (cf. table 1). The dynamic evaluation, for which reconfiguration was integrated into a physical simulation of ECLSS, showcases the continuous fault-handling (cf. *RQ1*). Two main limitations were identified: A suboptimal blocking behavior when the reconfiguration algorithm is executed repeatedly and the necessity of selecting a well-defined state variable, which is not always possible. (cf. section 5.3).

Future research may be conducted in the automatic data-driven generation of the qualitative system model, including system-specific constraints by sparse regression based system identification approaches, as developed by Brunton et al. [13]. This is of particular interest since the generation of the specific causal graph, although only qualitative in nature, still requires human expertise and extensive model checking effort. As indicated in the limitations, the algorithm does not always find the *best* solution. A further extension by incorporating a cost function might be developed to identify the cost-minimal solution. Further, the integration of *AutoConf* with a control system remains to be shown, which would operate on the input mask of the reconfigured system. Finally, the interaction of reconfiguration with diagnosis can be studied further to make use of all available data, widening the solution space.

Acknowledgments

This work was funded by dtec.bw® - Centre for Digitalisation and Technology Research of the Federal Armed Forces of Germany (BMVg).

References

[1] Mogens Blanke, Marcel Staroswiecki, Michel Kinnaert, and Jan Lunze. Diagnosis and fault-tolerant control. 2016.

[2] Kaja Balzereit and Oliver Niggemann. Autoconf: A new algorithm for reconfiguration of cyber-physical production systems. *IEEE Transactions on Industrial Informatics*, 2022.

[3] Leonardo de Moura and Nikolaj Bjørner. Z3: An Efficient SMT Solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 4963, pages 337–340. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[4] Bohui Wang, Bin Zhang, and Rong Su. Optimal tracking cooperative control for cyber-physical systems: Dynamic fault-tolerant control and resilient management. *IEEE Transactions on Industrial Informatics*, 17(1):158–167, 2021.

[5] Hui Ma, Qi Zhou, Lu Bai, and Hongjing Liang. Observer-based adaptive fuzzy fault-tolerant control for stochastic nonstrict-feedback nonlinear systems with input quantization. *IEEE Transactions on Systems, Man, and Cybernetics*, 49(2):287–298, February 2019.

[6] Benjamin Kuipers. Qualitative simulation. *Artificial intelligence*, 29(3):289–338, 1986.

[7] Judith Crow and John M Rushby. Model-based reconfiguration: Toward an integration with diagnosis. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, 1991.

[8] Raymond Reiter. A theory of diagnosis from first principles. *Artificial intelligence*, 32(1):57–95, 1987.

[9] Louise Travé-Massuyès. Bridging control and artificial intelligence theories for diagnosis: A survey. *Engineering Applications of Artificial Intelligence*, 27:1–16, 2014.

[10] Julian Doyé. An Advanced Columbus Thermal and Environmental Control System. In *SpaceOps 2012 Conference*, Stockholm, Sweden, June 2012. American Institute of Aeronautics and Astronautics.

[11] Sheila McIlraith, Gautam Biswas, Dan Clancy, and Vineet Gupta. Towards Diagnosing Hybrid Systems. *Working Notes of the AAAI 1999 Spring Symposium Series: Hybrid Systems and AI*, pages 128– 135, 1999.

[12] Christa Lüdecke and Dorothea Lüdecke. *Thermodynamik: physikalisch-chemische Grundlagen der thermischen Verfahrenstechnik*. Springer-Lehrbuch. Springer, Berlin Heidelberg, softcover reprint of the hardcover 1st edition 2000 edition, 2000.

[13] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Sparse Identification of Nonlinear Dynamics with Control (SINDYc). *IFAC-PapersOnLine*, 49(18):710–715, 2016.