Sound and Complete Reconfiguration for a Class of Hybrid Systems

Kaja Balzereit
Fraunhofer Center for Machine Learning
Fraunhofer IOSB-INA
Lemgo, Germany
kaja.balzereit@iosb-ina.fraunhofer.de

Oliver Niggemann
Helmut-Schmidt-University
Hamburg, Germany
oliver.niggemann@hsu-hh.de

Abstract

Reconfiguration is the automated recovery from a fault in hybrid systems such as tank systems from process engineering. Until today, many heuristics for this purpose have been developed. However, the reliability of these heuristics is not discussed or approximated using artificial systems.

In this article, soundness and completeness properties are defined in the context of reconfiguration. In addition, an algorithm based on the transformation of the reconfiguration problem to search and a subsequent solution using Best-First Search is presented. It is shown that this algorithm is sound and complete for a class of hybrid systems.

1 Introduction

Inputs of hybrid systems, i.e. systems characterized continuous variables such as tank levels and temperatures and discrete variables such as positions of switches, usually are controlled by a program leading the system to a goal, specified by the operator. Faults in hybrid systems, like a stuck pipe or a heating element failing, may lead to the control program becoming invalid, such that the system goal is no longer reached. Reconfiguration describes the automated adaptation of the system configuration by setting inputs to fixed values or exchanging faulty components [Balzereit and Niggemann, 2021a].

Model-based diagnosis [Reiter, 1987], which is in the scope of AI research for many years now, is a task closely related to reconfiguration: Diagnosis is concerned with the identification of the root cause of a fault. For this purpose, as for reconfiguration, logic-based approaches are used [Metodi *et al.*, 2014].

Properties like soundness, i.e. the property of only returning valid solutions, and completeness, i.e. the property of handling all faults, have been extensively discussed in the context of diagnosis [Feldman *et al.*, 2010] and qualitative reasoning [Travé-Massuyès *et al.*, 2003]. For reconfiguration, however, just few research exists. Until now, no discussion of central properties of reconfiguration algorithms exists.

In this paper, two central research hypotheses concerning reconfiguration for hybrid systems are discussed:

RH1: For a class of hybrid systems, a sound and complete reconfiguration algorithm can be created.

Soundness guarantees that the outputs of the reconfiguration algorithm are only correct solutions. Completeness guarantees that all faults can be handled. Soundness and completeness are central properties of logic-based algorithms [Kartha, 1993]. Thus, the strengths and limitations of algorithms can be examined and statements about reliability can be made [Travé-Massuyès *et al.*, 2003]. However, soundness and completeness for hybrid systems can only be shown with respect to the hybrid system model, e.g. a hybrid automaton. Real-world systems act in a nondeterministic environment such that it is not possible to predict every possible fault including a correct reconfiguration. In this paper, it is examined for which hybrid systems soundness and completeness can be guaranteed.

RH2: Search is suitable for formalizing and solving reconfiguration problems for a hybrid systems.

Search has been researched extensively in the area of Artificial Intelligence (AI) and been used successfully for solving various problems [Forbus and De Kleer, 1993]. We show an abstraction of the reconfiguration problem for hybrid systems into a discrete, finite search space and discuss the advantages and limitations of this approach.

The contribution of this article is threefold: (1) For the first time, the terms *soundness* and *completeness* are defined in the context of reconfiguration for hybrid systems. These definitions are derived from the definition of soundness and completeness for diagnosis algorithms. (2) A solution approach consisting of two steps is presented. First, the reconfiguration problem for hybrid systems, described by a hybrid system model, e.g. a hybrid automaton, is transformed into a search problem. Second, the search problem is solved using a new algorithm which is based on Best-First Search. (3) We proof soundness and completeness of the new algorithm for a specific class of hybrid systems.

The paper is structured as follows: In Section 2, the related work is discussed. Then, reconfiguration for hybrid systems is defined and formalized as search (Section 3). After that, in Section 4 the logic-based solution algorithm is presented. Decidability, soundness and completeness of the algorithm are shown in Section 5. Finally, conclusion and outlook are given (Section 7).

2 Related Work

The related work to this topic can be separated into two main areas: Qualitative Modeling and soundness and completeness properties of algorithms. Reconfiguration needs to reason about the causal coherences, i.e. the effects of changing an input variable on the state variables of the system. Hence, techniques from Qualitative Modeling are used. Due to the reconfiguration being close to model-based diagnosis [Reiter, 1987], we mainly discuss the recent approaches from that research area.

2.1 Modeling Hybrid Systems Qualitatively

Modeling hybrid systems in a qualitative way has been researched for many years. De Kleer and Brown [1984] established the field of Qualitative Physics. Hybrid systems are abstracted such that quantitative values are no longer needed, but only qualitative information is used.

Qualitative Simulation (QS) is concerned with modeling hybrid systems in terms of Qualitative Differential Equations to predict future system behavior by discretizing continuous variables using representative landmark values [Kuipers, 2001]. Trave-Massuyes et al. [2003] discussed QS and similar abstraction formalisms in terms of soundness and completeness. For some classes of hybrid systems and some abstraction formalism, the preservation of soundness and completeness properties can be shown whilst for other, no abstraction preserving soundness and completeness exists. Provan [2009] presented an abstraction method for hybrid systems preserving the completeness of diagnosis. The hybrid system is described in terms of propositional logic, allowing for the diagnosis of faults.

Benazera et al. [2009] divided the state space of hybrid systems into different regions to allow for drawing conclusions about system properties.

Hybrid Bond Graphs have been used for modeling the causal coherences necessary for diagnosis [Narasimhan and Biswas, 2007]. In addition, continuous observations have been integrated into the qualitative reasoning approach. The approach has been extended to the usage of Hybrid Minimal Structurally Overdetermined Sets which indicate the presence of a fault [Khorasgani and Biswas, 2017].

Stern et al. [2019] and Matei et al. [2020] presented modeling approaches relying on a combination of a learning method and information about the system topology. Thus, structural information is used to represent the causal coherences of the system.

Kong et al. [2015] showed an approach on the reachability analysis of hybrid systems. The approach operates on logical formulae representing the hybrid system and safe and unsafe areas. However, the modeling efforts of this approach are very high since hybrid systems are described in a detailed way using DAE.

The representation of hybrid systems in a qualitative way has been shown to be suitable for diagnosis. The choice of the most suitable modeling formalism depends on the information available. Our approach is compatible with all of these modeling approaches since from all of these models, a transformation to a search space can be done where our approach operates on.

2.2 Soundness and Completeness

Soundness and completeness are key aspects of diagnosis algorithms: Proving soundness and completeness guarantees for finding exclusively all correct diagnoses [Feldman *et al.*, 2010]. Siddiqi et al. [2007] presented a hierarchical diagnosis algorithm based on qualitative reasoning and proved its soundness and completeness. Feldman et al. [2010] presented a sound and complete SAT-based algorithm for model-based diagnosis. The search is accelerated by random guesses and flipping of binary variables. Stern et al. [2012] presented a conflict-directed search for model-based diagnosis. Soundness and completeness of the SAT-based algorithm is shown. Metodi et al. [2014] presented a SAT-based approach to model-based diagnosis which is based on splitting the system into cones. Also this approach is shown to be sound and complete.

In this paper, we combine the idea of using qualitative representations of hybrid systems with a BFS-based algorithm to create a sound and complete algorithm for the reconfiguration of a class of hybrid systems. Therefore, we transform the reconfiguration problem of hybrid systems into a search problem to find an assignment to the input variables recovering a valid configuration.

3 Formalization of Reconfiguration

In the following, we use the established formalism of hybrid automata to describe hybrid systems to derive a definition of reconfiguration.

Definition 1 (Hybrid Automaton (according to [McIlraith *et al.*, 1999; Lygeros *et al.*, 1999]))**.** A *hybrid automaton* H is a tuple $(X, I, \boldsymbol{x}_0, \mathcal{F}, \Sigma, \Phi)$ with

- I being a set of input variables. I consists of discrete I_D and continuous I_C variables $I = I_D \cup I_C$. $\boldsymbol{i}(t)$ refers to an input at time t, $\boldsymbol{i}_D(t)$ refers to a discrete input at time t, $\boldsymbol{i}_C(t)$ refers to a continuous input at time t. Every combination of discrete inputs $\boldsymbol{i}_D$ defines a *mode* $\mu \in \mathcal{M}$.
- $X \subset \mathbb{R}^n$ with $n \in \mathbb{N}$ denoting state variables. $\boldsymbol{x}(t)$ refers to a state describing the continuous behavior at time t. $\boldsymbol{x}(t = 0) = \boldsymbol{x}_0$ are initial states,
- $\mathcal{F}$ is a finite set of functions $\{\boldsymbol{f}_{\mu_1}, \boldsymbol{f}_{\mu_2}, ...\}$ with $\boldsymbol{f}_{\mu_i} : X \times \mathbb{R} \times I_C \to X$ describing the continuous behavior in mode μ_i over time $t \in \mathbb{R}$,
- Σ is a finite set of discrete events $\{\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2, ...\}$ with $\sigma_i : I_D \to I_D$ which transition the system between modes,
- $\Phi : \Sigma \times \mathcal{M} \times X \to \mathcal{M} \times X$ is a transition function mapping an action, a mode and a state into a new mode and initial state.

If all functions $\boldsymbol{f}_{\mu_i}$ are independent from t, so $\boldsymbol{f}_{\mu_i} : X \times I_C \to X \;\; \forall \mu_i \in M$, the automaton is called *time-invariant*. Otherwise, it is called *time-variant*. The set of output variables OP is defined by the observable states, $\boldsymbol{op}(t)$ refers to an output at time t.

For better readability, variables depending on t are shortened to the variable itself, e.g. $\boldsymbol{i}$ instead of $\boldsymbol{i}(t)$.

Definition 2 (Configuration)**.** Given a state $\boldsymbol{x}$ and an input $\boldsymbol{i}$, the tuple $(\boldsymbol{x}, \boldsymbol{i})$ is called a *configuration* of the hybrid system. $C = X \times I$ denotes the set of all configurations of a system.

The input variables I of a hybrid systems usually are controlled by a program P, e.g. a control unit or a production plan. Usually, to define the system goal, reference values on output variables are specified by human operators. Then, the program controls the system by a sequence of input changes which establish the wanted reference values [Blanke *et al.*, 2006].

Definition 3 (Reference Value)**.** A reference value $r \in \mathbb{R}$ on an output variable defines the requirement that needs to be met such that the system goal is reached.

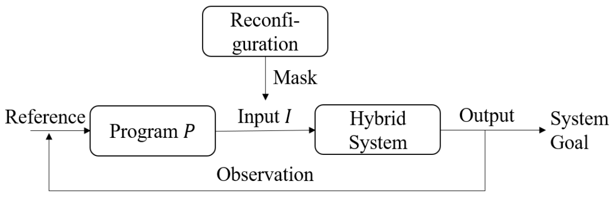


Figure 1: Reconfiguration masks some inputs of the hybrid system such that a control program can reach the system goal.

Definition 4 ((Control) Program)**.** Given a reference value $r \in \mathbb{R}$ and outputs $\boldsymbol{op}$, a program is a function $P : \mathbb{R} \times OP \rightarrow I$ assigning values to the input variables I to reach a user-specified system goal G. The set of all programs P for a system is described by $\mathcal{P}$.

There are configurations, from that a given system goal G cannot be reached by a program P, e.g. after a fault led to the program becoming invalid. Nowadays, at most known faults are handled by P while often no fault handling is done at all. Hence, when a fault occurs, reconfiguration becomes necessary to alter the current configuration to a new configuration from that the system goal becomes reachable again. Usually, the areas from that control programs of hybrid systems can reach a system goal are described by multidimensional polytopes [Maïga *et al.*, 2015].

Definition 5 (Fault)**.** A fault changes the system such that the program P no longer reaches the system goal [Blanke *et al.*, 2006].

Please note that this definition limits the term *faults* to deviations leading the system to no longer reach the system goal, i.e. deviating configuration (see Definition 2). Deviations that are mitigated by control are not within the scope of this article.

To recover a system from a fault, two steps are done (see Figure 1). First, reconfiguration changes the system to a new configuration by *masking* some inputs, i.e. applying a function f_R on a subset of the inputs $I_R \subset I$ that may no longer be changed by the control program. The new configuration shall enable that a - possibly adapted - production program P' can reach the system goal [Balzereit and Niggemann, 2020]. Second, an adapted program P' is identified which works on the remaining inputs $I \setminus I_R$ that are not masked and controls the system to reach the system goal. The identification of an adapted program P' (step II) can be done using well-known control methods. However, the identification of adequate reconfiguration operations is still an open research topic [Blanke *et al.*, 2006].

Definition 6 (Validity)**.** Given a system goal G, a configuration $(\boldsymbol{x}, \boldsymbol{i})$ is *valid* iff G can be reached by at least one program $P \in \mathcal{P}$. Otherwise, the configuration is *invalid*. C_v denotes the set of valid configurations, $C_i = C \setminus C_v$ denotes the set of invalid configurations.

Definition 7 (Reconfiguration)**.** Given an invalid configuration $(\boldsymbol{x}, \boldsymbol{i}) \in C_i$, *reconfiguration* is a function $\boldsymbol{f}_R : I_R \rightarrow I_R$ with $I_R \subset I$ such that $(\boldsymbol{x}, \boldsymbol{f}_R(\boldsymbol{i})) \in C_v$. Thus, a valid configuration is recovered and the system goal can be reached by a program $P' \in \mathcal{P}$.

Hence, three central questions of reconfiguration are: *(Q1) How are the sets C_v and C_i determined?* C_v defines the area of configurations allowing for a program $P \in \mathcal{P}$ reaching the system goal. Depending on the available programs P, this area varies. *(Q2) How is I_R, i.e. the set of input variables that is affected by reconfiguration, determined?* To minimize the intervention into the system and maximize the remaining control options, the set I_R shall be chosen as small as possible. *(Q3) How is f_R determined?* In many cases, setting input variables to fixed values may already allow for reaching the system goal. However, in some cases more complex functions for f_R might be necessary.

Definition 8 (Cardinality)**.** The *cardinality* of a reconfiguration $\boldsymbol{f}_R : I_R \rightarrow I_R$ is defined by $|I_R|$.

Definition 9 (Minimal Reconfiguration)**.** A reconfiguration f_R is called *minimal* if no reconfiguration $\tilde{I}_R \subset I_R$ exists, such that a function $\tilde{\boldsymbol{f}}_R : \tilde{I}_R \rightarrow \tilde{I}_R$ exists, that is also a reconfiguration.

Definition 10 (Minimal-Cardinality Reconfiguration)**.** A reconfiguration r is of *minimal-cardinality* if no reconfiguration $\tilde{\boldsymbol{f}}_R$ exists, such that $|\tilde{I}_R| < |I_R|$.

Definition 11 (Soundness)**.** A reconfiguration algorithm is *sound* if every output of the algorithm is a reconfiguration according to Definition 7.

Definition 12 (Completeness)**.** A reconfiguration algorithm is *complete* if for every invalid configuration a reconfiguration is found.

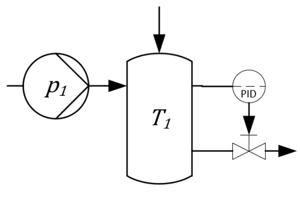
4 Solution Algorithm

The solution algorithm contains three steps: First (step I), reconfiguration is formalized as search. Second (step II), the transformation to search from a hybrid automaton is shown as an example. Last (step III), the search for a reconfiguration is solved using an adaptation of BFS.

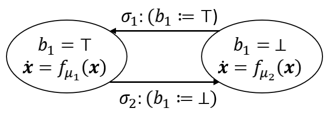
We address the questions *(Q1) - (Q3)* as follows: *Q1:* The sets C_v and C_i can take arbitrarily nonlinear shapes. However, in practical applications, the set from that a control reaches the system goal can be approximated using linear restrictions on state variables [Vännman and Albing, 2007; IEC, 2017]. In the following, we discuss assumptions suitable for real-world applications. *Q2:* Our approach is limited to hybrid systems where control works exclusively on continuous I_C and reconfiguration works exclusively on discrete input variables I_D. Thus, the controlled system behavior for each mode $\mu \in M$ is described by the functions $\boldsymbol{f}_\mu$. Without this limitation, the reconfiguration algorithm would also need to take the interdependencies of masking continuous variables on the functions $\boldsymbol{f}_\mu$ into account. The reconfiguration algorithm takes the current discrete input $\boldsymbol{i}_D$ into account and minimizes the number of discrete input variables that are masked. *Q3:* Since our approach works exclusively on discrete input variables, input variables are set to fixed, discrete values. WLOG we assume the discrete input variables to be binary. Non-binary discrete variables with finite value ranges are transformed to binary representations. $\boldsymbol{b}$ denotes the binarized inputs $\boldsymbol{i}_D$. B denotes the set of these. $\tilde{\Sigma} = \{\tilde{\sigma}_1, \tilde{\sigma}_2, ...\}$ with $\tilde{\sigma}_i : B \rightarrow B$ denotes the set of events on the binarized inputs.

In addition, we make the following assumption:

Assumption 1. *We assume that for every configuration $(\boldsymbol{x}, \boldsymbol{b})$ at least one reconfiguraton exists, e.g. by defining safety configurations that can always be reached (total shutdown or similar).*



(a) Piping and Instrumentation Diagram



(b) Hybrid Automaton H_R with $\boldsymbol{i}_D = (\boldsymbol{b}_1)^T$, $\boldsymbol{i}_C = (\boldsymbol{i}_v)^T$, $\boldsymbol{x} = (\boldsymbol{x}_1)^T$.

Figure 2: Running Example

Running Example The system in Figure 2a serves as running example. It consists of a pump p_1 delivering water to a tank T_1. The binary variable b_1 describes if the pump is turned on or off. T_1 has an outflowing valve which opening degree is described by $\boldsymbol{i}_v$. It is controlled by a PID controller and delivers water to a consumer.

The hybrid automaton H_R describing the running example consists of two modes $\mathcal{M} = \{\mu_1, \mu_2\}$ is shown in Figure 2b. Let $p \in \mathbb{R}$ be the amount of water p_1 delivers, $flow_{\text{in}}$ be a constant inflow, and $c(\boldsymbol{x})$ be the control function for the PID-controller of the output valve. Then $\boldsymbol{f}_{\mu_1}(\boldsymbol{x}) = 1/A \cdot (flow_{\text{in}} + p - c(\boldsymbol{x}))$ describes the change of water level when the pump is running and $\boldsymbol{f}_{\mu_2}(\boldsymbol{x}) = 1/A \cdot (flow_{\text{in}} - c(\boldsymbol{x}))$ describes the change of water level when the pump is not running. H_R is time-invariant. The goal of the system is to deliver a specific amount of water via the valve, which can be reached by the controller while the tank level is in between 0.2 and 0.6m.

4.1 Step I: Reconfiguration as Search

The main benefits of transformation reconfiguration into search problem are the following [Balzereit and Niggemann, 2021b]: (i) The reconfiguration algorithm becomes independent from modeling formalisms of hybrid systems. These formalisms may vary with the use case due to variation in available information. (ii) Research on algorithms to solve search problem has come to a sophisticated level. Our approach is based on a simple adaptation of BFS. However, it is compatible with more matured algorithms like A* or similar.

Definition 13 (Search Space)**.** A *search space* is a tuple (S, O) where $S = \{s_1, s_2, ...\}$ is a set of *search nodes* and $O = \{o_l : s_j \to s_k, l \in \mathbb{N}\}$ with $s_j, s_k \in S$ is a set of *operations*, which change the search nodes [Pearl and Korf, 1987].

Search Nodes

For reconfiguration, the search nodes $s_i \in S$ represent configurations $(\boldsymbol{x}, \boldsymbol{b})$. In this article, we are concerned with *time-invariant* systems. Hence, the timed behavior of the system is modeled in the state variables since states store the past system behavior. For *time-variant* systems, time needs to be represented explicitly in the search space.

Operations

The operations $o_l \in O$ represent the possible changes to the configuration, i.e. the changes to the input variables I. In the following, discrete reconfiguration for operations working on binarized input variables is defined. Hence, an operation $o_l \in O$ describes the change of the values of binary input variables B.

Discrete Reconfiguration

Definition 14 (Discrete Reconfiguration)**.** Given a search space (S, O) with $s_j = (\boldsymbol{x}, \boldsymbol{b})$ being configurations, an initial node s_0, and operations $O_B = B \times B$ changing the binary input variables, *discrete reconfiguration* is a sequence of operations R in O_B that recovers a valid search node, so the search node after applying the operations $s^* = o_l(...o_1(s_0)), o_1, ...o_l \in R$ is valid. Hence, discrete reconfiguration is defined by a sequence of operations through the search space, starting from an invalid node and ending at a a valid one.

A discrete reconfiguration can be mapped to a reconfiguration as in Definition 7: The set I_R is given by the binary variables B which are affected by an operation $o_B \in O_R$. The function $\boldsymbol{f}_R$ then is given by the values to which the variables are set.

The dimension of the search space depends (1) on the dimension of state variables $|X|$ and (2) on the dimension of the discrete input variables $|B|$. For each state and each binary input, one search node is created. Then, the number of search nodes is given by $2^{|B|} \cdot |X|$. Since the states are continuous, the dimension is infinite. By default, in each node $|B|$ operations, one for each change of a binary variable, can be applied. Then, the search space contains $2^{|B|}$ operations for each search node.

Definition 15 (Path)**.** A path P in the search space (S, O) is a sequence of operations $P = (s_0 \to s_1), ...(s_{k-1} \to s_k)$ with $s_i, i \in \{1, 2, ...k\}$ being pairwise disjoint. For better readability, in the following the search nodes in a path are separated by a comma.

Theorem 1. *Let R be a sequence of operations $R = (s_0, s_1), (s_1, s_2), ...(s_{k-1}, s_k)$ with $s_0, s_1, ...s_{k-1}$ being invalid and s_k being valid. Then R is a minimal reconfiguration if and only if R is a path.*

Proof. if-part: Proof via contradiction. Assume R is a path but R is not a minimal reconfiguration. Since R leads from an invalid node s_0 to a valid node s_k, R is a reconfiguration. So, R must not be minimal. Thus, $\exists \tilde{R}$ which is a reconfiguration from s_0 to s_k containing only operations that are in R. As a consequence, R must contain cycles which is a contradiction to Definition 15.

only-if-part: Proof via contradiction. Assume R is a minimal reconfiguration but R is not a path. Then R contains at least one node twice, WLOG this node is s_j. So, $R = ((s_0, s_1), ..., (s_{j-1}, s_j), (s_j, s_{j+1}), ...(s_{j+o}, s_j), (s_j, s_m)...(s_{k-1}, s_k))$. Then $\exists \tilde{R} = ((s_0, s_1), ..., (s_l, s_j), (s_j, s_m)...(s_{k-1}, s_k))$ leading to a reconfiguration $\tilde{R}$ containing less operations than R, but only operations which are also in R, which contradicts the minimality of R. □

The length of a path P, so the number of operations in P, is the cardinality of the corresponding reconfiguration. Thus, the minimal-cardinality reconfiguration is defined by the shortest path to a valid node.

4.2 Step II: Generation of Search Space from Hybrid Automaton

The transformation of the reconfiguration problem for hybrid systems, described by a hybrid system model, e.g. a hybrid automaton, into the search problem plays an important role when it comes to soundness and completeness. Hence, as an example, we show the transformation for a hybrid system modeled as a hybrid automaton [McIlraith *et al.*, 1999].

The key aspect is the definition of valid and invalid configurations, so if the system goal can be reached by a program P (see Definition 6). The areas from where it is guaranteed that a program P reaches the system goal theoretically can take every shape. However, in practice, often simple linear restrictions are suitable [Vännman and Albing, 2007; IEC, 2017]. Hence, the following assumption is made:

Assumption 2. *The area from that a program can reach the system goal can be described using bound values on observable state variables, so* $\boldsymbol{lb}_i \leq \boldsymbol{x}_i \leq \boldsymbol{ub}_i \rightarrow \exists P \in \mathcal{P}$ *with* $\boldsymbol{lb}_i, \boldsymbol{ub}_i$ *being bound values for the state variable* $\boldsymbol{x}_i$*.*

Thus, given a current state, it can be checked whether the system goal can be reached. However, if the system will recover to such a state in a reconfiguration time Δt is an open question. This question could be answered using a simulation of the system. But such a simulation usually is not available since not enough knowledge about the system exists. Qualitative Simulation (QS) is concerned with predicting future system states using reduced knowledge [Kuipers, 2001]. Unfortunately, QS suffers from a explosion of predictions such that even incorrect ones are generated [Kuipers, 1985].

Our approach predicts only the area of the system variables in the next step using quantitative information. We discuss, for which class of hybrid automata the presented reconfiguration algorithm is still sound and complete.

In the following, first a theorem on predicting future system states is presented. Then, it is discussed, for which automata this is applicable.

The theorem uses functions $\boldsymbol{f}_{\mu,\tau}$ describing the system behavior in mode μ in the presence of the fault τ. These function can either be learned if data about faults is available [Khoo *et al.*, 2020], generated using temporal causal graphs and fault signatures [Narasimhan and Biswas, 2007] or modeled as additive and multiplicative faults, as usual in Fault-Tolerant Control [Zhou and Frank, 1998].

Theorem 2. *Given a hybrid automaton H with $\boldsymbol{x}(t)$ being the current state and μ being the current mode, a fault τ, and a set of programs $\mathcal{P}$ satisfying Assumption 2. If*

$$\boldsymbol{f}_{\mu,\tau}(\boldsymbol{x})_i \cdot \Delta t + 0.5\dot{\boldsymbol{f}}_{\mu,\tau}(\boldsymbol{\xi}_i) \cdot \Delta t^2 \geq \boldsymbol{lb}_i - \boldsymbol{x}_i, \quad (1)$$

$$\boldsymbol{f}_{\mu,\tau}(\boldsymbol{x})_i \cdot \Delta t + 0.5\dot{\boldsymbol{f}}_{\mu,\tau}(\boldsymbol{\xi}_i) \cdot \Delta t^2 \leq \boldsymbol{ub}_i - \boldsymbol{x}_i \quad (2)$$

with $\boldsymbol{\xi} \in [\boldsymbol{x}(t), \boldsymbol{x}(t+\Delta t)]$ being the remainder value from the Taylor approximation holds, then a program $P \in \mathcal{P}$ exists that reaches the system goal.

Proof. Let $\boldsymbol{x}_i < \boldsymbol{lb}_i$. Using the Taylor approximation

$$\boldsymbol{x}_i(t+\Delta t) = \boldsymbol{x}_i(t) + \dot{\boldsymbol{x}}_i(t) \cdot \Delta t + 0.5\ddot{\boldsymbol{x}}_i(\nu) \cdot \Delta t^2 \quad (3)$$

$$= \boldsymbol{x}_i(t) + \boldsymbol{f}_{\mu,\tau}(\boldsymbol{x})_i \cdot \Delta t + 0.5\dot{\boldsymbol{f}}_{\mu,\tau}(\boldsymbol{\xi})_i \cdot \Delta t^2 \quad (4)$$

with $\nu \in [t_0, t_0 + \Delta t]$ and $\boldsymbol{x}(\nu) = \boldsymbol{\xi}$ follows. Then, from (1)

$$\boldsymbol{x}_i(t+\Delta t) \geq \boldsymbol{x}_i(t) + \boldsymbol{lb}_i - \boldsymbol{x}_i(t) = \boldsymbol{lb}_i \quad (5)$$

follows (analogously for $\boldsymbol{x}_i > \boldsymbol{ub}_i$). Thus $\forall \boldsymbol{x}_i \in X$ $\boldsymbol{lb}_i \leq \boldsymbol{x}_i(t+\Delta t) \leq \boldsymbol{ub}_i$ holds, such that the system goal can be reached at time $t + \Delta t$ due to Assumption 2. □

Since showing property [(1), (7)] is not possible for complex functions and unknown $\boldsymbol{\xi}$ we use an approximation which holds for functions with a bounded derivative function $2\alpha \leq \dot{\boldsymbol{f}}_{\mu_j} \leq 2\beta$ and adapt the constraints to

$$\boldsymbol{f}_{\mu,\tau}(\boldsymbol{x})_i \cdot \Delta t + \alpha \cdot \Delta t^2 \geq \boldsymbol{lb}_i - \boldsymbol{x}_i, \quad (6)$$

$$\boldsymbol{f}_{\mu,\tau}(\boldsymbol{x})_i \cdot \Delta t + \beta \cdot \Delta t^2 \leq \boldsymbol{ub}_i - \boldsymbol{x}_i. \quad (7)$$

Theorem 3. *Given a hybrid automaton H with states $\boldsymbol{x}$, mode μ and function $\boldsymbol{f}_{\mu,\tau}$ with a bounded derivative $2\alpha \leq \dot{\boldsymbol{f}}_{\mu,\tau} \leq 2\beta$, H satisfies [(1), (2)] if the constraints [(6), (7)] are satisfied.*

Proof. Trivial. □

Using this property, the transformation for a hybrid automata with bounded functions $\boldsymbol{f}_{\mu,\tau}$ is shown in Algorithm 1. First (l. 1), the search space is created from the cartesian product of the state space X and the input space B. The initial configuration is created from the initial state $\boldsymbol{x}_0$ and the initial input $\boldsymbol{b}_0$ (l. 2).

The set of operations is initialized (l. 3), and then, for each event (l. 4) and for each input (l. 5), the resulting input from the event is identified (l. 6). From that, the search nodes are identified (l. 7) and the operation is added to O (l. 8).

The set of valid search nodes is defined by the set of search nodes that satisfy [(6), (7)].

The evaluation function which is used in Best-First Search Algorithm is created from the reciprocal Hamming distance of the binary input variables.

Please note that at this point, S_v and f_{eval,s_0} do not need to be enumerated for all search nodes but can be evaluated during the search.

The created search space and the additional information is returned (l. 13).

Algorithm 1: *HybridAutomatonToSearchSpace*

```
   Input: $H = (X, I, \boldsymbol{x}_0, \mathcal{F}, \Sigma, \Phi), \boldsymbol{b}_0$
   Output: $(S, O), s_0, S_v, f_{\text{eval},s_0}$
   // search nodes
1  $S := X \times B$
2  $s_0 := (\boldsymbol{x}_0, \boldsymbol{b}_0)$
   // operations
3  $O = \emptyset$
4  for $\tilde{\boldsymbol{\sigma}}_i \in \tilde{\Sigma}$ do
5      for $\boldsymbol{b}_i \in B$ do
6          $\boldsymbol{b}_o := \tilde{\boldsymbol{\sigma}}_i(\boldsymbol{b}_i)$
7          $o := (\boldsymbol{x}, \boldsymbol{b}_i) \rightarrow (\boldsymbol{x}, \boldsymbol{b}_o)\ \forall \boldsymbol{x}, (\boldsymbol{x}, \boldsymbol{b}_i), (\boldsymbol{x}, \boldsymbol{b}_o) \in S$
8          $O := O \cup \{o\}$
9      end
10 end
   // validity of search nodes
11 $S_v := \{s_i \in S | s_i$ satisfies [(6), (7)$\}$
   // evaluation function
12 $f_{\text{eval},s_0}(s_i) := \|\boldsymbol{b}_i \oplus \boldsymbol{b}_0\|_1^{-1}$ where $s_i = (\boldsymbol{x}_i, \boldsymbol{b}_i)$
13 return $(S, O), s_0, S_v, f_{\text{eval},s_0}$
```

4.3 Step III: Best-First Search to Identify Solution

The solution algorithm is based on a BFS to identify a valid configuration [Pearl, 1984]. BFS uses a evaluation function f_{eval,s_0} which assigns each search node s_i a heuristic value $f_{\text{eval},s_0}(s_i) \in \mathbb{R}$ rating the node.

To enable minimal-cardinality solutions, f_{eval,s_0} is defined by the reciprocal distance of $s_i = (\boldsymbol{x}_i, \boldsymbol{b}_i)$ to $s_0 = (\boldsymbol{x}_0, \boldsymbol{b}_0)$ calculated by $|\boldsymbol{b}_i \oplus \boldsymbol{b}_0|^{-1}$.

Please note that due to [(1), (2)] the validity of a configuration (and thus of a search node) varies with the states $\boldsymbol{x}$. However, since the validity does not need to be known apriori but is evaluated in *BFReconf*, this does not pose any problems.

The algorithm uses the functions *PathTo*, which identifies a path from between two nodes using a list of parent nodes, and *Successors*, which returns, given a node and the available operations, the connected nodes.

Algorithm 2: *BFReconf*

```
Input: $(S, O), s_0, S_v, f_{\text{eval},s_0}$
Output: $R = ((s_0, ...), ...(..., s^*))$
1  OPEN = $\{s_0\}$
2  POPEN = $\emptyset$
3  CLOSED = $\emptyset$
4  while OPEN $\neq \emptyset$        // start solving loop
5  do
6      $s^* := \text{argmax}_{s_i \in \text{OPEN}}\{f_{\text{eval},s_0}(s_i)\}$   // find best node
7      CLOSED := CLOSED $\cup\, s^*$
       // return if valid
8      if $s^* \in S_v$ then
9          $R$ = PathTo$(s_0, s^*, \text{POPEN})$
10         return $R$
11     end
       // examine successors
12     for $\tilde{s} \in$ Successors$(s^*, O)$ do
13         if $\tilde{s} \notin$ CLOSED then
14             OPEN := OPEN $\cup\, \tilde{s}$
15             POPEN := POPEN $\cup\, (\tilde{s}, s^*)$
16         end
17     end
18 end
```

The solution algorithm *BFReconf* is given the initial configuration $s_0 = (\boldsymbol{x}, \boldsymbol{b})$ and the evaluation function f_{eval} as input and outputs a new configuration $s^* = (\boldsymbol{x}, \boldsymbol{b}^*)$. First, the sets OPEN, POPEN, CLOSED, where POPEN stores the parent nodes of the nodes in OPEN, are initialized (l. 1, 2, 3). In the solving loop (l. 4), first the node with the highest f_{eval}-value in OPEN is identified (l. 6) and added to CLOSED (l. 7). If the identified node s^* is valid (l. 8), the function *PathTo* identifies a path from s_0 towards s^* using the list of parent nodes POPEN (l. 9) and returns it (l. 10). Otherwise, the successors of s^* are examined (these are given by the changes to the binary variables) (l. 12) and added to OPEN (l. 14), if the node is not already in CLOSED (l. 13). The information about the parent node s^* is stored in POPEN (l. 15).

5 Theoretical Results

Theorem 4. *Let (S, O) be the search space created from Algorithm HybridAutomatonToSearchSpace from a time-invariant hybrid automaton with bounded functions $\boldsymbol{f}_{\mu,\tau}$ and $\mathcal{P}$ be a set of programs which satisfies Assumption 2. Then Algorithm BFReconf is sound.*

Proof. *BFReconf* outputs an $s^* \in C_v$ for which one of two things holds:

Either $\boldsymbol{lb}_i \leq \boldsymbol{x}_i \leq \boldsymbol{ub}_i \quad \forall \boldsymbol{x}_i \in X$ holds. Then, due to Assumption 2, the system goal can be reached.

Otherwise the constraints [(6), (7)] hold such that due to Theorems 3 and 2 the system goal can be reached. □

Theorem 5. *Given a time-invariant hybrid system and an invalid configuration s_0, BFReconf is complete.*

Proof. Due to Assumption 1, for every configuration a reconfiguration, so a path to a valid configuration, exists. Since *BFReconf* iterates over all connected nodes until a valid node is found, *BFReconf* is complete. □

6 Practical Application

The running example is used to show the practical application of the presented method.

Let $A = 1m^2$ be the area of the tank, $p = 0.12m^3/s$ be the volume of the pump when activated and $flow_{\text{in}} = 0.08m^3/s$ be, $c((x)) = a_v \cdot \delta_c \cdot \sqrt{2g\boldsymbol{x}_1}$ be the outflow through the controlled valve with $a_v = 0.05$ being the area of the valve, $\delta_c \in [0, 1]$ being the opening degree set by the PID controller, and $g = 9.81m/s^2$ being the gravitational acceleration. The bound values are set to $\boldsymbol{lb}_1 = 0.2, \boldsymbol{ub}_1 = 0.6$, the reconfiguration time Δt is set to $1s$. For every $\boldsymbol{x}_1 \in [\boldsymbol{lb}_1, \boldsymbol{ub}_1]$ the configuration is valid.

The search space (S, O) consists of $S = \mathbb{R} \times \{0, 1\}$ and $O = \{o_l : (\boldsymbol{x}, b) \to (\boldsymbol{x}, \neg b) | x \in \mathbb{R}, b \in \{0, 1\}\}$.

We consider the fault case of a leak in the tank τ. Then, $f_{\mu_{1/2},\tau}(\boldsymbol{x}) = f_{\mu_{1/2}}(\boldsymbol{x}) - 1/A \cdot (a_\tau \sqrt{2g\boldsymbol{x}})$ with $a_\tau = 0.01$ being the area of the leak. Given a configuration $\boldsymbol{x}_1 = 0.15, b_1 = \bot$, the configuration is labeled invalid since the approximation (6)

$$\boldsymbol{f}_{\mu,\tau}(0.15) + \alpha \geq 0.05 \tag{8}$$

with $\alpha = 0.006$ being the lower bound of the derivative $\dot{\boldsymbol{f}}_{\mu,\tau}$ is not satisfied. Then, *BFReconf* checks the configuration $\boldsymbol{x}_1 = 0.15, b_1 = \top$ which satisfies [(6),(7)] and returns the operation $o_R = (0.15, \bot) \to (0.15, \top)$ as reconfiguration.

The constraints [(6),(7)] perform a quadratic approximation using the bound values of $\dot{\boldsymbol{f}}_{\mu,\tau}$ and check, if both approximations lie between the bound values $\boldsymbol{lb}, \boldsymbol{ub}$. Figure 3 shows the approximation for the running example with activated pump. As can be seen, after the reconfiguration time of $1s$, both approximations lie between $\boldsymbol{lb}$ and $\boldsymbol{ub}$ such that the configuration is valid.

7 Conclusion

In this article, for the first time, the terms soundness and completeness are defined in the context of reconfiguration for hybrid systems. A reconfiguration algorithm based on the transformation to a search problem and subsequent solution using and adaptation of Best-First Search is presented. Soundness and completeness are shown for a class of hybrid systems. In future work, an A* search algorithm will

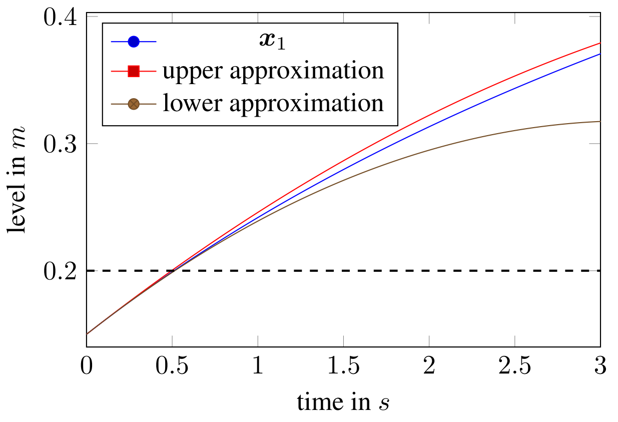


Figure 3: Lower and upper quadratic approximation of tank level of Running Example.

be used such that not only a valid but the best reconfiguration is identified.

Acknowledgment

This work was developed within the Fraunhofer Cluster of Excellence "Cognitive Internet Technologies" and was partly funded by the German Federal Ministry for Education and Research grant 01IS19082B.

References

[Balzereit and Niggemann, 2020] Kaja Balzereit and Oliver Niggemann. Modeling quantitative effects for the automaed reconfiguration of hybrid systems. In *31th International Workshop on Principles of Diagnosis (DX'20)*, 2020.

[Balzereit and Niggemann, 2021a] Kaja Balzereit and Oliver Niggemann. Gradient-based reconfiguration of cyber-physical production systems. In *4th IEEE International Conference on Industrial Cyber-Physical Systems*, 11 2021.

[Balzereit and Niggemann, 2021b] Kaja Balzereit and Oliver Niggemann. Reconfiguring hybrid systems using sat, 2021.

[Benazera and Travé-Massuyès, 2009] Emmanuel Benazera and Louise Travé-Massuyès. Set-theoretic estimation of hybrid system configurations. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(5):1277–1291, 2009.

[Blanke *et al.*, 2006] Mogens Blanke, Michel Kinnaert, Jan Lunze, and Marcel Staroswiecki. *Diagnosis and fault-tolerant control*, volume 2. Springer, 2006.

[De Kleer and Brown, 1984] Johan De Kleer and John Seely Brown. A qualitative physics based on confluences. *Artificial intelligence*, 24(1-3):7–83, 1984.

[Feldman *et al.*, 2010] Alexander Feldman, Gregory Provan, and Arjan Van Gemund. Approximate model-based diagnosis using greedy stochastic search. *Journal of Artificial Intelligence Research*, 38:371–413, 2010.

[Forbus and De Kleer, 1993] Kenneth D Forbus and Johan De Kleer. *Building problem solvers*, volume 1. MIT press, 1993.

[IEC, 2017] IEC. *IEC 61360 - Standard data element types with associated classification scheme*, 2017.

[Kartha, 1993] G Neelakantan Kartha. Soundness and completeness theorems for three formalizations of action. In *IJCAI*, volume 93, pages 724–729. Citeseer, 1993.

[Khoo *et al.*, 2020] Teck Ping Khoo, Jun Sun, and Sudipta Chattopadhyay. Learning fault models of cyber physical systems. In *International Conference on Formal Engineering Methods*, pages 147–162. Springer, 2020.

[Khorasgani and Biswas, 2017] Hamed Khorasgani and Gautam Biswas. Structural fault detection and isolation in hybrid systems. *IEEE Transactions on Automation Science and Engineering*, 15(4):1585–1599, 2017.

[Kong *et al.*, 2015] Soonho Kong, Sicun Gao, Wei Chen, and Edmund Clarke. dreach: δ-reachability analysis for hybrid systems. In *International Conference on TOOLS and Algorithms for the Construction and Analysis of Systems*, pages 200–205. Springer, 2015.

[Kuipers, 1985] Benjamin Kuipers. The limits of qualitative simulation. In *IJCAI*, volume 9. Citeseer, 1985.

[Kuipers, 2001] Benjamin Kuipers. Qualitative simulation. *Encyclopedia of physical science and technology*, 3:287–300, 2001.

[Lygeros *et al.*, 1999] John Lygeros, Claire Tomlin, and Shankar Sastry. Controllers for reachability specifications for hybrid systems. *Automatica*, 35(3):349–370, 1999.

[Maïga *et al.*, 2015] Moussa Maïga, Nacim Ramdani, Louise Travé-Massuyès, and Christophe Combastel. A comprehensive method for reachability analysis of uncertain nonlinear hybrid systems. *IEEE Transactions on Automatic Control*, 61(9):2341–2356, 2015.

[Matei *et al.*, 2020] Ion Matei, Johan de Kleer, Alexander Feldman, Rahul Rai, and Souma Chowdhury. Hybrid modeling: Applications in real-time diagnosis. *arXiv preprint arXiv:2003.02671*, 2020.

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

[Metodi *et al.*, 2014] Amit Metodi, Roni Stern, Meir Kalech, and Michael Codish. A novel sat-based approach to model based diagnosis. *Journal of Artificial Intelligence Research*, 51:377–411, 2014.

[Narasimhan and Biswas, 2007] Sriram Narasimhan and Gautam Biswas. Model-based diagnosis of hybrid systems. *IEEE Transactions on systems, man, and cybernetics-Part A: Systems and humans*, 37(3):348–361, 2007.

[Pearl and Korf, 1987] Judea Pearl and Richard E Korf. Search techniques. *Annual Review of Computer Science*, 2(1):451–467, 1987.

[Pearl, 1984] Judea Pearl. Heuristics: intelligent search strategies for computer problem solving. 1984.

[Provan, 2009] Gregory Provan. Model abstractions for diagnosing hybrid systems. In *Proceedings of the 20th International Workshop on Principles of Diagnosis, DX-09, Stockholm, Sweden*, pages 321–328, 2009.

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

[Siddiqi *et al.*, 2007] Sajjad Ahmed Siddiqi, Jinbo Huang, et al. Hierarchical diagnosis of multiple faults. In *IJCAI*, volume 7, pages 581–586, 2007.

[Stern and Juba, 2019] Roni Stern and Brendan Juba. Safe partial diagnosis from normal observations. *Thirty-Third AAAI Conference on Artificial Intelligence*, 2019.

[Stern *et al.*, 2012] Roni Stern, Meir Kalech, Alexander Feldman, and Gregory Provan. Exploring the duality in conflict-directed model-based diagnosis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, 2012.

[Travé-Massuyès *et al.*, 2003] Louise Travé-Massuyès, Liliana Ironi, and Philippe Dague. Mathematical foundations of qualitative reasoning. *AI magazine*, 24(4):91–91, 2003.

[Vännman and Albing, 2007] Kerstin Vännman and Malin Albing. Process capability indices for one-sided specification intervals and skewed distributions. *Quality and Reliability Engineering International*, 23(6):755–765, 2007.

[Zhou and Frank, 1998] Dong-Hua Zhou and PM Frank. Fault diagnostics and fault tolerant control. *IEEE Transactions on aerospace and electronic systems*, 34(2):420–427, 1998.