Modeling Quantitative Effects for the Reconfiguration of Hybrid Systems

Kaja Balzereit
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 task of recovering a valid system state after an error has occurred, which led to an invalid system state. Especially for hybrid systems, identifying the necessary changes to restore valid system functioning is challenging: Hybrid systems contain continuous and discrete variables, leading to an infinite search space which, in addition, suffers from combinatorial explosion. Existing approaches to the reconfiguration problem mostly require a pre-definition of faults and a large amount of expert knowledge and thus, enable the system to adapt only to known faults.

This paper presents a novel approach which does not need a pre-definition of faults such that the system is enabled to adapt even to unknown faults. It works on an representation of the reconfiguration problem in a logical calculus. Therefore, the hybrid system is modeled in first-order logic. To integrate continuous variables, which have infinite domains, they are discretized using intervals. The approach is shown to reconfigure faults on simulated systems from process engineering. This way, the reconfiguration problem of hybrid systems can be modeled and solved efficiently.

1 Introduction

Hybrid systems offer many possibilities to adapt to changed circumstances like alternative paths or parameter settings. However, the full potential often is not reached since the control software mostly handles only a predefined set of faults with predefined reconfiguration instructions [1]. Such that the control is adaptable also to unforeseen faults, it needs to identify the necessary changes to reconfigure the system while the system is operating. However, there are some aspects making this challenging: Hybrid systems have an infinite state space consisting of continuous and discrete variables, which lead to a combinatorial explosion since the number of discrete combinations rises exponentially with the number of variables. Logic can be used to handle the problem of combinatorial explosion [2]. Using logical constraints, the space of valid discrete combinations is reduced extremely, such that a suitable solver. e.g. a SAT solver, often can solve the problem in a short time even though the runtime rises exponentially with the number of variables for the worst case.

However, explicitly modeling a hybrid system in logic requires a large amount of manual efforts and expert knowledge, which is hardly available [3]. Until now, only few approaches to the automated reconfiguration of hybrid systems has been published [4]. Existing solutions rely on creating a single, static system model that is only applicable to a predefined set of faults. But for hybrid systems predicting every possible fault often is not possible since they operate in a non-deterministic environment [5]. The lack of an existing solution comes from some unsolved research challenges, three of which follow.

RQ1: What is a suitable model of a hybrid system for the task of reconfiguration? Since creating a logical model from expert knowledge is not possible for most cases, a model is needed that on the one hand is expressive enough to perform reconfiguration, and on the other hand can be generated from available information (e.g. a topology description from a P&ID). Existing models like hybrid automata [6] or state space representations [7] which can be learned are not expressive enough for reconfiguration: A model suitable for reconfiguration (1) needs to be suitable with the binary concept of valid and invalid behavior and (2) needs to include information about the impacts of changing input variables on the system variables such that a valid state is recovered. Therefore, not only the information about the coherence between an input and a state variable like it is used for diagnosis is needed [5], but also the direction (increase/decrease) that the state variable is changed to.

RQ2: How can the continuous variables be suitably discretized such that reconfiguration can be performed efficiently? Explicitly modeling reconfiguration instructions for every possible state of the hybrid system is not feasible since the state space in general is infinite. Discretization can be used to separate continuous data into a finite number of intervals and thus reduce the complexity of the specific problem. Interval-based methods have been shown to improve the performance of various learning methods [8] . However, for reconfiguration, intervals do not only need to group similar values but also contain qualitative information about validity, i.e. if the interval contains valid or invalid values. Until now, no research has been done on the generation of intervals for reconfiguration.

RQ3: Are gradients of the state variables necessary to handle the system dynamics? The behavior of hybrid systems in general is dynamic, i.e. the consequences of a change do not manifest directly but evolve over time. This has impact on the decision of a reconfiguration method since the consequences of a reconfiguration may manifest later.

Gradients of state variables give information about the direction the state variables evolve and thus, can give information if validity will be recovered after a fault has occurred. However, since reconfiguration works on a system abstraction, a prediction of future system states using the gradient can lead to an explosion of the state space: Various possible future system states, including the correct one but also numerous incorrect states may be predicted [9]. Therefore, this research question addresses the need and value of integrating the gradient into the reconfiguration approach.

The main contribution of this paper is twofold: (1) A novel modeling formalism of hybrid systems in a first-order logic (FOL) such that reconfiguration can be performed efficiently is presented. The system description can be drawn from a topology description of the system such that the need for expert knowledge is reduced. An integration of continuous values into this calculus using interval decomposition to enable reconfiguration is presented. The resulting formula is satisfiable if and only if the reconfiguration problem has a solution; (2) The need of gradients to model the dynamic system behavior is examined by evaluating the gradient-based solution approach using examples from process engineering.

The paper is structured as follows: First, the existing methods for control and reconfiguration of hybrid systems are discussed. After that, the problem is formalized and the solution approach is presented. Then, the approach is evaluated using representative hybrid systems from process engineering. Finally, a conclusion is given.

2 Related Work

Models of Hybrid Systems

There are many approaches to represent hybrid systems like hybrid automata [6], state space representations [7], or formulations in logic [10]. However, such that reconfiguration can be performed efficiently, the creation of the system should be possible from available information, e.g. the system topology, and the model should be suitable with a binary definition of valid behavior.

Tsude et al. [7] discussed reconfiguration strategies for hybrid systems based on a state space representation of the system. Thus, the control was enabled to recover from faults using physical redundancy. However, due to the usage of an explicit model, large optimization problems need to be solved continuously.

In the field of automated planning, a hybrid system is described in terms of states, actions and events [11]. For every possible action and event, the preconditions and effects need to be defined. Crawford et al. [12] presented a concept for the online reconfiguration of manufacturing systems using automated planning, enabling the systems to adapt to both, changing external circumstances and faults Grastien [13] chose a finite state machine as a system description and solved the reconfiguration problem using a combination of automated planning and model-based diagnosis (MBD). However, using planning to solve reconfiguration problems requires the creation of an explicit system model from expert knowledge, which, however, is not available in most cases.

The goal of fault-tolerant control (FTC) is to enable a system to accommodate faults using methods from control theory. Therefore, a state-space representation of the system is used and an optimal adjustment for known faults is searched [14]. Thus, actuator and sensor faults can be handled. However, FTC approaches are mainly based on a fixed system model. Major system changes or perturbances like a failing system component require a new system model which leads to high modeling efforts [15].

MBD is a task closely related to the task of reconfiguration. The challenge how to model a hybrid system has been addressed by Matei et al. [3] and Stern et al. [5]. Both approaches rely on a combination of a learning method and information about the system topology. Thus, the complex behavior of the hybrid system is represented and structural information is used to represent the causal coherences of the system. These approaches have shown that the system topology serves well at delivering information about causal coherences and creating a qualitative system representation. Hence, we choose a similar approach for the problem of automated reconfiguration. However, for the task of reconfiguration, not only the fact that a variable has a wrong value needs to be known but also if the variable needs to be decreased or increased to recover a valid value. In addition, the impact of input variables on the state variables needs to be modeled such that reconfiguration is enabled to identify the changes necessary to recover a valid state. Thus, we present an adaption of the existing modeling formalisms. The formalism includes the handling of continuous variables but works on an abstraction of the system such that the efforts for solving the reconfiguration problems are reduced.

Interval Decomposition

Applying interval decomposition to continuous data has shown to improve the performance of optimal control methods [16] and different learning methods [8]. Continuous data can be decomposed into discrete intervals using naive methods like equal-width or equal-frequency decomposition, statistics-based methods, information maximization based methods [8], and entropy-based methods [17]. All of these approaches use interval decomposition to group similar values to improve the performance of methods from machine learning. However, for reconfiguration, suitable intervals must be assigned with qualitative information about the validity of the values in the interval, i.e. which intervals indicate valid and which invalid behavior. None of the existing methods aims at separating non-faulty from faulty intervals.

Gradients

Kuipers established the field of Qualitative Simulation (QS) in that a system is abstracted to a qualitative differential equation system [18]. Therefore, real variables are separated into intervals and information about variable monotony is used to predict future system states. Thus, also the behavior of systems for which just few expert knowledge is available can be simulated. However, due to the high system abstraction QS may predict various system behaviors, even incorrect ones [19]. Our approach also works on an abstraction for that monotony information in form of the sign of the gradient is used, but no simulation of the system is done. Instead, the gradient information is used to determine if the value of a variable will be higher or lower at a future point of time.

Gradients have also been used in diagnosis to integrate information about dynamic system behavior allowing for the diagnosis of incipient faults [20]. However, calculating the gradient from historical data suffers from system noise. Our

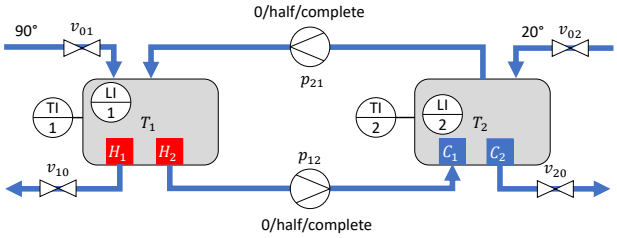

Figure 1: Running Example: The system contains two tanks, two heating elements, two cooling elements and two connecting pipes. Every tank has an inflow with either hot (90 degree) or cold (20 degree) fluid and an outflow.

approach also uses information about the gradient to draw conclusions about the dynamic behavior but omits to calculate the gradient explicitly but relies solely on the sign.

3 Problem Formalization

Following, the problem of reconfiguration of hybrid systems is formalized in an analogously to the formalization of diagnosis given by Feldman et al. [21]. The two-tank system in Figure 1 serves as a running example. The system is an adaption of the Titration and Neutralisation Plant which has already been used for the evaluation of reconfiguration methods [15]. In tank T_1, two heating units, H_1 and H_2, are placed. In tank T_2, two cooling units, C_1 and C_2, are placed. The tanks each have one external inflow, v_{01} and v_{02}, and are connected via pipes which can be controlled by the pumps p_{21} and p_{12}. These pumps can either be run on half or full power, or close the connection completely. The temperature and fluid level are measured in each tank. The goal of the system is to deliver fluid between 65 and 75 degrees through v_{10} and fluid between 10 and 20 degrees through v_{20} while keeping the fluid level in each tank between 30 and 40.

To perform reconfiguration a description of the hybrid system to draw conclusions about the impact of changing an input variable and a definition of valid behavior is needed. In this paper, hybrid systems that can be described by a DAE in the form

$$0 = F(\dot{\mathbf{x}}, \mathbf{x}, \mathbf{u}, t), \mathbf{y} = H(\mathbf{x}, \mathbf{u}, t) \quad (1)$$

where $\mathbf{x}$ describes state variables, $\mathbf{u}$ describes input variables, $\mathbf{y}$ describes output variables, and t describes the time are considered. It is assumed that the function H is invertible such that the values of the state variables $\mathbf{x}$ can be calculated from the output variables $\mathbf{y}$ (full observability). Therefore, in the following, $\mathbf{x}$ is handled as a vector of known values.

In this approach, the input variables $\mathbf{u}$ are discretized such that they can be represented by binary variables $\mathbf{b}$ since the task is not to control a system in an optimal way but to restore a valid state. Therefore, exact numerical values mostly are not necessary [15]. For example in the tank system, the goal is not to reach one exact water level but to recover from a water level causing damage to the system. Therefore, the pumps and valves do not need to be controlled in an optimal way, which would require very detailed numerical information, but a binary on-off control to reach the valid interval again is suitable. Following, $\mathbf{b}$ denotes the vector of binary input variables, B denotes the set of these. X denotes the set of state variables.

Definition 1. A *system description* SD is a formula in FOL over the input variables B and state variables X.

Given a set of variables V, an assignment $\alpha : V \to \mathbb{R}$ is a function that maps a value to all variables in V. Thus, observation of the system variables can be represented by assignments. Such that a reconfiguration method is able to distinguish between valid and invalid system states, a definition of validity is required.

Definition 2. Given an assignment α_X to the state variables X a *definition of validity* w is a function $w : \alpha_X \to \{\top, \bot\}$ which returns $\top$ if the state is valid and $\bot$ if the state is invalid.

A fault is a disturbance that leads from a valid to an invalid state. In this approach actuator faults (like a pipe stuck), internal system faults (like a leaking tank), and external faults (like a cold environment leading to a cooling of the water) are considered. However, sensor faults which lead to faulty output values are not handled, the outputs are assumed to represent the current system state.

The task of a reconfiguration method is described as given an assignment to the state variables, e.g. current sensor measurements, the goal is to find a truth assignment to the input variables that restores valid behavior. However, the system often cannot be recovered from a valid state directly but some time delay is needed. Hence, a reconfiguration problem can be formalized as following:

Definition 3. A *hybrid reconfiguration problem* HRP is defined as a tuple $(B, X, SD, w, \alpha_B, \alpha_X)$ where here SD is a system description over the input variables B and the state variables X, w is a definition of validity, α_B is an assignment to B, and α_X is an assignment to X with $w(\alpha_X) = \bot$. The solution to a HRP is an assignment α_B^* to the input variables of the HRS that changes the system to a valid state in a finite time t^*, so $w(\alpha_X^*) = \top$ where α_X^* are the observed state variables from the system with input α_B^* after time t^*.

4 Solution Approach

In process plants, even if there are errors, the end product often can still be produced as before using physical redundancies, e.g. by using a second reactor. Or, in other cases, at least a safe state can be reached. For this, first the invalid system state has to be detected (using the definition of validity w), then the alternative production route has to be implemented by switching valves, pumps and reactors (using the inputs B).

Figure 2 shows the main challenges of reconfiguration for hybrid systems: First, a system description needs to be abstracted from the real world system. As described in *RQ1* this description on the one hand needs to be suitable with a binary concept of validity and on the other hand needs to contain information about the impact of changing an input variable on the state variables of the system. Logic allows for modeling the binary concept of valid and invalid states and has successfully been used for the similar problem of diagnosis. However, logical models in diagnosis only contain the information *that* a variable has an impact on another one [22]. For reconfiguration, also information *how* a variable influences another is needed, e.g. in a positive or negative way. Hence, in section 4.1 we present an extension of existing modeling approaches to enable reconfiguration.

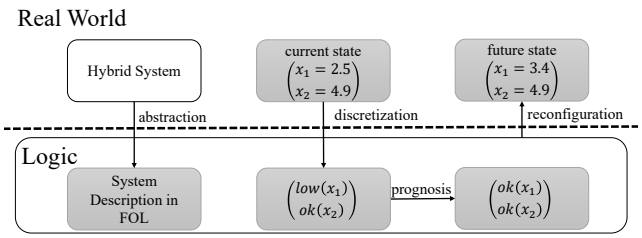

Figure 2: Basic concept of the reconfiguration method. Arrows describe steps of the solution approach. Gray boxes contain information.

As presented in *RQ2*, infinite domains coming from continuous system variables need to be integrated into the logical calculus. Here, a discretization of the state variables using intervals is presented. Thus, the infinite domains are reduced into a finite number of intervals. In addition, intervals can be assigned with an ordinal relation encoding information if the value is okay, too high or too low, which is close to human intuition [23] (section 4.2). This simplification is suitable since valid behavior in real-world systems often is expressed in terms of valid intervals [24]. In contrast to QS the intervals here are not used for the simulation of the system but only for the indication of valid and invalid areas.

Since the task of reconfiguration is to recover a valid state from an invalid one, a reconfiguration method always needs to perform some kind of prognosis, i.e. a prediction how changing system inputs will manifest in the future (see *RQ3*) . Therefore, information about the gradient of the state variables is used (section 4.3). However, using the gradient has some drawbacks: First, gradients are sensitive to system noise. Second, the sign of the gradient is used to predict the future direction of the state variables, which may be false for dynamically changing systems [19]. However, in this approach, only the sign of the gradient is used to predict if the value in the next step will be higher or lower. Thus, the impact of noise is reduced. And in contrast to QS, a prognosis is only done for the next time step and not for a complete simulation. Hence, the risk of having improper predictions and a state explosion is low.

The system, the definition of validity based on intervals, the information from the prognosis, and the current system values α_B, α_X are input to the solution algorithm (section 4.4). This algorithm determines updated input variables α_B^* and returns these to the hybrid system to recover a valid state.

4.1 System Description

The system description consisting of a FO-formula enables the reconfiguration method to determine the necessary changes leading to a valid output. The binary variables form the space of possible changes that can be initiated by reconfiguration. Here, subformulas like $\mathbf{b}_{ij} \rightarrow inc(\mathbf{x}_j)$ and $\mathbf{b}_{ij} \rightarrow dec(\mathbf{x}_i)$ form the system description where $dec(\cdot), inc(\cdot)$ are predicate symbols describing that the state variable $\mathbf{x}_i, \mathbf{x}_j \in X$ will increase/decrease when opening the connection $\mathbf{b}_{ij}$. Manually modeling these formulas requires a large amount of expert knowledge which is hardly available [3]. Other approaches like probabilistic graphical models require failure probabilities for every system component [25] which is also not available for most systems [5].

One possible way is to gather these formulas from a state space representation. However, the function F may not always be known. Hence, the presented approach relies on information about the system topology which is available for most systems [5]. How to describe a system topology has been discussed in the context of qualitative physics [26], bond graph modeling [27], and cyber-physical production systems [28]. Referring to these, the terms *entity*, *component* and *connection* are defined in the following.

Systems process different kinds of materials or energies. An electrical circuit for example processes electrical energy, a tank system processes fluid and thermal energy. Even though the materials differ, the systems underlie the same principles [26].

Definition 4. An *entity* $\tau \in T$ describes one kind of material or energy (e.g. fluid, electrical or thermal energy). Each state variable x_i is assigned to exactly one entity, thus entities create a partition on the set of state variables based on their physical units. T denotes the set of all entities in a system.

Definition 5. A *component* c is a constituent which stores or processes energy/material of at least one entity. Mathematically seen, a component is described by a subset of equations of (1). Since the number of equations and variables may be unbalanced for this subset, the unknown variables are inputs to the component. These input variables are either input variables $\mathbf{u}$ of the overall system or output variables $\mathbf{y}_{c_j}$ from connected components c_j.

It is assumed that one component is assigned a maximum of one state variable per entity (e.g. a tank may only have one water level). $x_{\tau,c}$ denotes the state variable of entity τ and component c.

Definition 6. A *connection* is a constituent which connects the variables corresponding to one entity τ of two components c_1, c_2 by setting the sum of flow variables (e.g. electrical current or a fluid flow) to zero $f_{\tau,c_1} + f_{\tau,c_2} = 0$ and equating the potential variables (e.g. electrical voltage or pressure) $p_{\tau,c_1} = p_{\tau,c_2}$.

In hybrid systems, there may be multiple connections between components (e.g. two pumps connecting the same two tanks). Hence, the topology of a system is represented by a multi-graph. The nodes represent system components, the edges represent connections. The topology is separated according to the entities of the system, e.g. one graph describing the fluid behavior and one graph describing the temperature behavior.

To derive the formula describing the system, the connections of the system, that can be controlled (opened/closed), are assigned with directions indicating the flow of energy/material. If opening a connection increases the state variable of the corresponding component, it is modeled as an ingoing edge, otherwise as an outgoing one. For the running example, the direction of the connection p_{21} is from tank T_2 to tank T_1 since this is the flow of water when the pump is activated. This direction usually can be drawn from a process description or a system specification [28]. However, some edges might be used bi-directional, e.g. a valve connecting two tanks where the flow direction depends on the level of the tanks. To model these, the edge is split up to two directed edges which are assigned constraints indicating when the specified direction is valid.

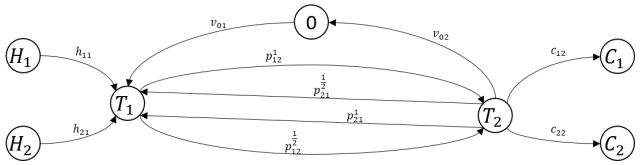

Figure 3: Graph representing the temperature structure.

Each connection (i, j) is assigned with a binary variable b_{ij} which is true if the connection is opened and false otherwise. These binary variables form the set of input variables, i.e. the variables that can be manipulated by the reconfiguration method. For each connection, according to the edges in the multi-graphs the formulas $\mathbf{b}_{ij} \rightarrow dec(\mathbf{x}_{\tau,i})$ and $\mathbf{b}_{ij} \rightarrow inc(\mathbf{x}_{\tau,j})$ are drawn describing that by opening the connection the state variable of the out-flowing component is decreased and the state variable of the in-flowing component is increased. Thus, for every connection of every multi-graph, two formulas are drawn which together form the system description.

Running Example The two-tank system processes two entities, fluid and temperature. Thus, the system topology is described by two directed multi-graphs. Figure 3 shows the directed graph representing the temperature structure. Connections which deliver cold material, and thus, decrease the temperature in a component, are modeled as outgoing edges, even though the fluid flow is in the opposite direction. The inflows v_{01}, v_{02} are inflows from an external constituent, which is represented by component 0 in the system. The pumps can be either run on full or half power or shut down completely. To model these using binary variables, each opening state is modeled as an individual edge. Since the pump can only be run on one opening state at the same time, the constraint $\top \rightarrow \bigoplus_{k \in I} \mathbf{b}^k_{ij}$ where I describes the set of opening states, is necessary.

4.2 Discretization using Intervals

As in control theory [14] and automated planning [11], reconfiguration needs information about the system goal to distinguish between valid and invalid states. In consistency-based diagnosis, non-faulty behavior is encoded by satisfiability of a logical formula [29]. For reconfiguration, this goal is defined by valid and invalid areas of the state variables (see Definition 2). Theoretically, this areas can take every shape, even nonlinear ones. However, in real-world systems, valid areas usually are represented using intervals [24]. Hence, the presented approach works on validity defined in intervals on output variables. There have been many approaches to the decomposition of continuous variables into intervals. However, the scope of these approaches is to group similar data to improve the performance of learning methods. This differs from the scope of intervals needed for reconfiguration. These intervals need to be assigned with qualitative information, e.g. an ordinal relation allowing for an intuitive interpretation of the value (e.g. too low, too high or okay) [23]. Usually, this information comes from quality and safety requirements, physical limitations and process specifications [28].

To integrate this information into FOL predicates encoding if a variable is too low, okay or too high $(low(\cdot), ok(\cdot), high(\cdot))$ are assigned with a truth value by the constraints $\mathbf{x}_{\tau,c} < lb_{\tau,c} \rightarrow low(\mathbf{x}_{\tau,c})$, $lb_{\tau,c} \leq \mathbf{x}_{\tau,c} \leq ub_{\tau,c} \rightarrow ok(\mathbf{x}_{\tau,c})$, and $\mathbf{x}_{\tau,c} > ub_{\tau,c} \rightarrow high(\mathbf{x}_{\tau,c})$. Since $\mathbf{x}_{\tau,c}$, $lb_{\tau,c}$ and $ub_{\tau,c}$ are known the truth values of these constraints can be evaluated a-priori and do not need to be handled by the SAT solver. Thus, every observation α_X implies a truth assignment to the predicates $low, ok, high$ for every state variable. We use α^{int}_X to denote this truth assignment. Hence, the observation α_X is discretized such that the current system state is represented in terms of valid and invalid intervals.

The system state is valid, if every variable is between its variable bounds. Thus, the definition of validity is given by $w := \top$ if $\mathbf{x}_{\tau,c} \in [lb_{\tau,c}, ub_{\tau,c}] \forall \mathbf{x}_{\tau,c} \in X, \bot$ else.

4.3 Prognosis using Gradient Information

The presented reconfiguration approach is based on the assumption that the future interval of a state variable can be predicted from the current state and information about the gradient. In detail, from $low(\mathbf{x}_{\tau,c}) \wedge pos(\dot{\mathbf{x}}_{\tau,c})$, where $pos(\cdot)$ is a predicate symbol which is true if the sign of $\dot{\mathbf{x}}_{\tau,c}$ is positive, can be drawn that $\mathbf{x}_{\tau,c}$ will be ok in a finite time. Thus, no further reconfiguration is necessary. The sign of the gradient is determined using the sign of the backwards difference quotient. To recover variables which are low but do not satisfy the above constraint, i.e. the gradient sign is either zero or negative, further actions for recovery are necessary. To increase a variable, either a binary variable leading to an increase needs to be activated or a binary variable leading to a decrease needs to be deactivated. This is represented by

$$low(\mathbf{x}_{\tau,c}) \wedge \neg pos(\dot{\mathbf{x}}_{\tau,c}) \rightarrow \bigvee_{b_{ij} \in IN} (\neg pre(\mathbf{b}_{ij}) \wedge \mathbf{b}_{ij}) \vee \bigvee_{b_{ij} \in DE} (pre(\mathbf{b}_{ij}) \wedge \neg \mathbf{b}_{ij}) \quad (2)$$

where the pre-operator returns the value of a variable in the prior time step, IN, DE denote the binary variables leading to an increase/decrease which are drawn from the system description SD. The decrease of variables which are $high$ is performed analogously. The conjunction of the constraints (2) for every state variable for low and $high$ is denoted by Re.

4.4 Overall Solution using SAT

Algorithm 1: IntervalReconf

Input: $HRP = (B, X, SD, w, \alpha_B, \alpha^{\text{int}}_X), Re$
Output: α^*_B or *error*

$n^{\text{ub}} \leftarrow 1$
while $n^{\text{ub}} \leq |\text{B}|$ **do**
  $\beta \leftarrow (\sum_{(i,j)} (\mathbf{b}_{ij} \oplus pre(\mathbf{b}_{ij})) \leq n^{\text{ub}})$
  if $\text{SAT}(SD \wedge \alpha^{\text{int}}_X \wedge (Re \wedge \alpha_B) \wedge \beta)$ **then**
    $\alpha^*_B \leftarrow \text{MODEL}(SD \wedge \alpha^{\text{int}}_X \wedge (Re \wedge \alpha_B) \wedge \beta)$
    return α^*_B
  else
    $n^{\text{ub}} \leftarrow n^{\text{ub}} + 1$
// no solution found
return *error*

Algorithm 1 shows how the overall solution is done. The functions SAT and MODEL are functions of the used SAT solver and check if the given formula is satisfiable or return

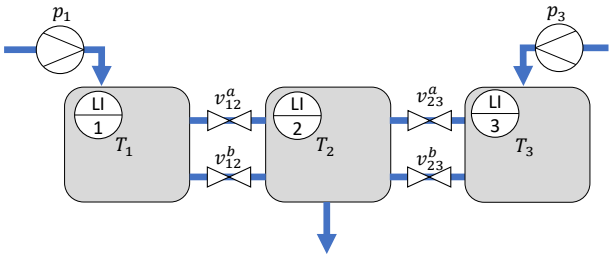

Figure 4: Three-Tank problem: The system consists of three tanks, four valves and two pumps. The valves v_{12}^a, v_{23}^a are located at a height of 30 cm, the valves v_{12}^b, v_{23}^b are located at the bottom. Thus, the flows through the valves depend on the water levels in the tanks. The goal of the system is to deliver water through the outflow of T_2.

the corresponding model, if it exists. The hybrid reconfiguration problem HRP and the reconfiguration events Re are input to the algorithm. The core of the algorithm is the satisfiability check of the formula in line 4 which consists of the system description SD, α_X^{int} which assigns the predicates indicating the actual interval a truth value, $Re \wedge \alpha_B$ which encodes how to perform reconfiguration where α_B denotes the previous values of the binary variables, and the constraint β containing an upper bound for binary changes n^{ub} to maintain system stability (l. 3). Starting with 1 (l. 1), the upper bound is increased (l. 8) until a satisfying assignment to the formula is found or the number of possible changes is reached (l. 2). This is a cardinality constraint which can be transformed to conjunctive normal form such that satisfiability can still be checked by a standard solver [30]. If a satisfying assignment exists (l. 4), it is identified (l. 5) and returned (l. 6). Otherwise, no reconfiguration exists and an error is returned (l. 9).

5 Evaluation

Until today, no benchmark of hybrid systems with injected faults exists [3]. Thus, the running example and the benchmark problem as presented by Heiming et al. [31] consisting of three tanks, four valves and two pumps are used for evaluation. Both systems have already been used to serve as examples for reconfiguration problems [15]. They are simulated in Modelica [32], the satisfiability of the logical formula is checked using Z3 [33].

For reconfiguration, faults are not separated according to the component they affect but to their effect on the state variables of the system. Continuous faults lead to a continuous change of at least one state variable, for example a leak in a tank. Discrete faults lead to an abrupt change of at least one state variable, for example due to someone adding some extra water to one tank. Temporal effects, like a creeping loss of temperature are handled as soon as the first variable reaches an interval bound.

Illustrative Example on Two-Tank-System

In case of no faults, the two-tank system from the running example operates in the standard configuration with the inflows v_{01}, v_{02} and the outflows v_{10}, v_{20} being open and both heating elements and both cooling elements being turned on. The pumps are not used. As can be seen in Figure 5, without faults the water levels and the temperatures in both tanks are constant. At time $t = 60$s, the water level in T_1

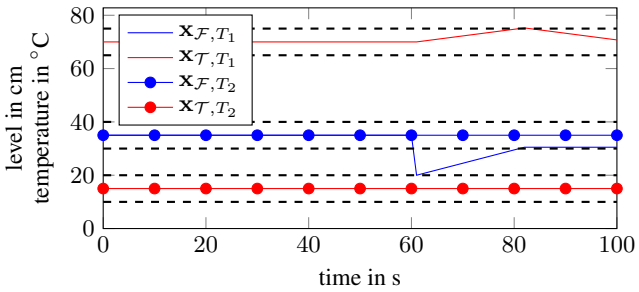

Figure 5: Water level and temperature in tanks T_1 and T_2 with abrupt drop of water level in T_1 at $t = 60s$. The black, dashed lines indicate the bound values for the valid areas.

Table 1: Amount of correctly reconfigured test cases.

test cases	# cases	# reconf.	amount in %
continuous	34	34	100
discrete	32	32	100
multiple continuous	4	4	100
multiple hybrid	5	4	80
multiple discrete	7	4	57

decreases abruptly by 42%. The gradient-based reconfiguration method is applied and the outflow v_{10} is closed. Thus, the temperature in the tank rises, since the heaters heat up less water. When the temperature reaches the upper limit of 80 degrees (at approximately 80s), the heating element H_1 is turned off and the temperature falls again. The outflow is opened since a valid water level is reached again.

5.1 Empirical Evaluation

In total, 80 faults are injected into the systems. In both systems, continuous faults are represented by leaks in one tank (2 cases per system), one of the valves being stuck in either open or closed position (8 cases per system) and one of the pumps being stuck on full power or blocked (4 cases per system). Discrete faults are abrupt drops and rises (5 test cases each with varying intensity from 20% to 70% of current water level per system) of the water level of one tank, e.g. by manual adding cold or hot water to the system.

In the running example, which processes temperature in addition to fluid, further faults are injected. Temperature losses and rises (1 case each) and failing heating and cooling elements (4 cases) are continuous faults, discrete drops and rises of temperature (6 test cases each with varying intensity from 7% to 42% of current temperature) are discrete faults. Additionally, multiple simultaneous faults are simulated which are either a combination of two continuous faults (4 test cases combining heating/cooler failures with stuck valves/pumps), a continuous and a discrete fault (5 test cases combining heater/cooler failures with abrupt water loss) or a combination of two discrete faults (5 test cases combining water and temperature loss, 1 test case per system combining water loss in two tanks).

The results of the evaluation of the test cases are shown in Table 1. All of the continuous, discrete and multiple continuous faults are solved correctly. When it comes to multiple faults with at least one discrete fault, the method is only capable of handling 80% (mixed hybrid) to 57% (mixed discrete) fault since multiple faults often lead to goal conflicts: Not only one state variable needs to be changed, but multiple ones which can lead to contradicting constraints.

Since no prioritization of goals is done, the reconfiguration method tries to recover every invalid state variable simultaneously which is not always possible.

6 Conclusion

Reconfiguration is the task of restoring valid behavior after a fault has occurred. Especially for hybrid systems, the problem of reconfiguration is not yet solved. To handle the complexity of hybrid system an abstraction of the hybrid system in FOL, gathered from a state space representation, is used. Therefore, continuous variables are decomposed into a finite number of intervals that are integrated into FOL, which is capable of handling the problem of combinatorial explosion. To integrate the dynamic behavior information about the gradient of state variables is included.

The results show that the reconfiguration method is capable of handling single faults and even multiple continuous faults very well. Since the presented approach omits the need for a quantification of the gradient and a prediction is only done for the next time step, the risk of false predictions due to noise or a state space explosion are low. Especially the multiple discrete faults are hard to handle for the reconfiguration method since they often lead to goal conflicts. Adding a prioritization to the goals will enable the methods to handle the faults one by one. Since the faults simulated concern every component and connection in the systems and even combined faults are considered, we believe that the examined faults cover a wide range of possible faults. Additionally, since no assumption on the application area of process engineering is done, we assume the approach to be transferable to a wide range of hybrid systems. To corroborate these assumptions, we will evaluate the approach using examples from other areas, including real-world systems.

At the moment, full system observability is required. However, this requirement may not always be satisfied, especially for real-world systems. How to define validity for partially observable systems and how to solve the corresponding reconfiguration problem will be examined in the future.

References

[1] Sylviane Gentil, Jacky Montmain, and Christophe Combastel. Combining fdi and ai approaches within causal-model-based diagnosis. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(5):2207–2221, 2004.

[2] Leonid I Perlovsky. Conundrum of combinatorial complexity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(6):666–670, 1998.

[3] 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.

[4] Kaja Balzereit and Oliver Niggemann. Automated reconfiguration of cyber-physical production systems using satisfiability modulo theories. In *3rd IEEE International Conference on Industrial Cyber-Physical Systems*, 11 2020.

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

[6] Oliver Niggemann, Benno Stein, Asmir Vodencarevic, Alexander Maier, and Hans Kleine Büning. Learning behavior models for hybrid timed systems. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[7] Kazuro Tsuda, Domenico Mignone, Giancarlo Ferrari-Trecate, and Manfred Morari. Reconfiguration strategies for hybrid systems. In *Proceedings of the 2001 American Control Conference.(Cat. No. 01CH37148)*, volume 2, pages 868–873. IEEE, 2001.

[8] Alberto Cano, José María Luna, Eva L Gibaja, and Sebastián Ventura. Laim discretization for multi-label data. *Information Sciences*, 330:370–384, 2016.

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

[10] Hamed Khorasgani, Gautam Biswas, and Daniel Jung. Structural methodologies for distributed fault detection and isolation. *Applied Sciences*, 9(7):1286, 2019.

[11] Michael Cashmore, Daniele Magazzeni, and Parisa Zehtabi. Planning for hybrid systems via satisfiability modulo theories. *Journal of Artificial Intelligence Research*, 67:235–283, 2020.

[12] Lara S Crawford, Minh Binh Do, Wheeler S Ruml, Haitham Hindi, Craig Eldershaw, Rong Zhou, Lukas Kuhn, Markus PJ Fromherz, David Biegelsen, Johan de Kleer, et al. On-line reconfigurable machines. *AI Magazine*, 34(3):73–88, 2013.

[13] Alban Grastien. Self-healing as a combination of consistency checks and conformant planning problems. In *DX@ Safeprocess*, pages 105–112, 2015.

[14] 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.

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

[16] Hamid Reza Marzban. Optimal control of linear multi-delay systems based on a multi-interval decomposition scheme. *Optimal Control Applications and Methods*, 37(1):190–211, 2016.

[17] Rebelo de Sá CF Pinho, C Soares, AJ Knobbe, et al. Entropy-based discretization methods for ranking data. *Information Sciences*, 329:16, 2016.

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

[19] Vittorio Brusoni, Luca Console, Paolo Terenziani, and Daniele Theseider Dupré. A spectrum of definitions for temporal model-based diagnosis. *Artificial Intelligence*, 102(1):39–79, 1998.

[20] Lee Barford, Eric Manders, Gautam Biswas, Pieter Mosterman, Vishnu Ram, and Joel Barnett. Derivative estimation for diagnosis. *HP LABORATORIES TECHNICAL REPORT HPL*, 1999.

[21] 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.

[22] Gregory Provan. Hierarchical model-based diagnosis. *Proc. DX'01*, pages 167–174, 2001.

[23] Alan Mathison Turing. Systems of logic based on ordinals. *Proceedings of the London mathematical society*, 2(1):161–228, 1939.

[24] 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.

[25] Ali Abdollahi, Krishna R Pattipati, Anuradha Kodali, Satnam Singh, Shigang Zhang, and Peter B Luh. Probabilistic graphical models for fault diagnosis in complex systems. In *Principles of Performance and Reliability Modeling and Evaluation*, pages 109–139. Springer, 2016.

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

[27] Peter J Gawthrop and Geraint P Bevan. Bond-graph modeling. *IEEE Control Systems Magazine*, 27(2):24–45, 2007.

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

[29] 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.

[30] Niklas Eén and Niklas Sörensson. Translating pseudo-boolean constraints into sat. *Journal on Satisfiability, Boolean Modeling and Computation*, 2(1-4):1–26, 2006.

[31] B Heiming and J Lunze. Definition of the three-tank benchmark problem for controller reconfiguration. In *1999 European Control Conference (ECC)*, pages 4030–4034. IEEE, 1999.

[32] Hilding Elmqvist, Sven Erik Mattsson, and Martin Otter. Modelica: The new object-oriented modeling language. In *12th European Simulation Multiconference, Manchester, UK*, 1998.

[33] Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340. Springer, 2008.