Identifying Model-Based Reconfiguration Goals through Functional Deficiencies

Emmanuel Benazera1 and Louise Travé-Massuyès2

Abstract. Model-based diagnosis is now advanced to the point autonomous systems face some uncertain and faulty situations with success. The next step toward more autonomy is to have the system recovering itself after faults occur, a process known as model-based reconfiguration. After faults occur, given a prediction of the nominal behavior of the system and the result of the diagnosis operation, this paper details how to automatically determine the functional deficiencies of the system. These deficiencies are characterized in the case of uncertain state estimates. A methodology is then presented to determine the reconfiguration goals based on the deficiencies. Finally, a recovery process interleaves planning and model predictive control to restore the functionalities in prioritized order.

1 Introduction

Model-based autonomous systems already face faulty situations with some success: they detect and diagnose faults by either identifying potential candidates for their own physical state [6] or reasoning on their structural and behavioral knowledge [5]. The next step toward more autonomy is to have the system recovering itself after faults occur, a process known as model-based reconfiguration (MBRconf). Automated reconfiguration comprehends three steps: goal identification, goal selection, recovery. Goal identification searches for a set of potential states of the system where the fault effects are inhibited; goal selection is the process of deciding the best of these states, denoted goal states; recovery searches for the chain of actions that may turn the physical system state into the desired goal states. Recent architecture design for autonomy [11] puts the goal identification and selection processes outside the scope of a model-based diagnoser, in the hands of upper decisional levels. The aim of this paper is to produce an automated goal identification/selection/recovery methodology that takes better advantage of the system model. Due to several factors, MBRconf is a challenging problem:

- The state of the system cannot be uniquely determined in all situations. Recent model-based monitoring/diagnosis systems tracks several potential non-faulty/faulty state estimates simultaneously [12, 2]. Moreover, the set of state estimates is the result of a selection process as the total number of possible states is too large to be explored. The ambiguity is however mitigated by the fact that the number of state estimates is typically small.
- Faults effects may differ from one state estimate to the other. For this reason, pre-compiled policies may fail recovering the system by proposing an improper command when the state is uncertain.
- Nowadays, embedded digitally controlled systems have complex behaviors characterized by a preeminence of discrete switches in their dynamics. They are modeled as hybrid systems, that exhibit both discrete and continuous dynamics.

Referring to the faulty states as the estimates that result from the diagnosis operation, as opposed to the nominally predicted states, we propose to compare the faulty states and the predicted states and thus determine the functional deficiencies caused by the faults. In this context, functional deficiencies are variable instances in one or more predicted states and that have been lost in one or more faulty states. Our approach aims at minimizing the size of a functionality to recover while maximizing its coverage of the estimates. The contributions of this paper are threefold. First, we show how this strategy leads to a finite set of disjoint functional deficiencies, and characterize them. Second, we propose a methodology to identify potential goals from the deficiencies based on a productive analogy with model-based diagnosis, reasoning at a single point in time, despite the system continuous dynamics. Third, we show how to interleave conformant planning and model predictive control to bring the system's hybrid dynamics from the initial faulty (uncertain) state to the potential goal state.

2 Hybrid Model-Based State Prediction and Diagnosis

In this section we introduce a comprehensive formalization of model, state and uncertainty. The autonomous system is considered a model-based system, i.e. that has a structural and behavioral knowledge of itself.

Definition 1 (Model-Based System). A model-based system $A$ is a tuple $(C, M, T, X, E)$, where $C$ is a set of modeled components, $M$ a set of finite discrete variables as component behavioral modes, $T$ a set of transitions among these modes, $X$ the set of continuous variables partitioned in state variables $X_e$, output (observed) variables $X_o$ and input variables (commands) $X_i$. $E$ a set of continuous static/differential equations over $X$.

In this paper we use a hybrid description of the physical system's state. The hybrid state $s$ is the tuple $(M, X)$. Instances of variables $v$ in $M \cup X$ are noted $(v = v^p)$, or $v^j$ for short. The hybrid state's discrete side abstracts the physical system as a set of mode instances $M = \bigcup_k C_k.m^k$ where $C_k.m^k$ is an instance of a variable $m \in M$ of component $C_k \in C$. The continuous state $X$ is made of instances $x^j$ of continuous variables of $\mathcal{X}_e$. Instances of observed
variables of $X_v$ are noted $y$ (vector $Y$), and $\tilde{y}$ (vector $\tilde{Y}$) denotes the measured value. Commands are noted $u$ (vector $U$). We consider a discrete-time model of the form:

$$E: \begin{cases} X(k+1) = f(X(k), U(k)) \\ Y(k) = g(X(k), U(k)) \\ 0 \leq h(X(k), U(k)) \end{cases}$$ \hspace{1cm} (1)$$

System $A$'s behavior is described with rules of the form $\bigwedge \epsilon_i \phi_i$ if $\phi$, where $\epsilon_i \in E$ and $\phi$ is a conjunction of equalities/inequalities over functions of variables in $M \cup X$. A set $T = \{\tau_1, \tau_2, \ldots, \tau_{\text{trans}}\}$ of transitions is specified for each mode $m$. Each transition $\tau$ is enabled according to a guard $\phi(\tau)$ and may trigger with probability $p(\tau)$ whenever the guard is satisfied. $T(s_1, s_2)$ denotes the set of transitions that moves $A$ from $s_1$ to $s_2$.

Given the ability $A$ has to predict and diagnose its own behavior, we respectively note $P(A)$ the prediction of the hybrid system's nominal state, and $D(A)$ the diagnosis result after a fault occurs. Note that when fault modes are present, the diagnosis may become a state identification problem, and $P(A)$, $D(A)$ may result from the same engine. Uncertainty on the physical system's state requires us to consider $P(A)$ and $D(A)$ as sets of hybrid states. We denote $S = (P(A), D(A))$.

**Example (Pressure expansion system).** Figure 1 pictures our case study: a two valves system that limits water pressure between input flow $Q_0$ and flow output $Q$. An electric switch $S$ powers valve $V_2$ when pressure $P_0$ equals or exceeds threshold $P^*$. $V_2$ opens when powered. $S$, $V_1$ and $V_2$ have two nominal operational modes open and closed, and two faulty modes stuck_CLOSED, stuck_OPEN. $Q_0$ and $Q$ are measured. $P_0 \geq P_{\text{atm}}$ is the only input to the system. $P_{\text{atm}}$ denotes the atmospheric pressure.

Our assumption scenario faults occur when the prediction of the nominal state is uncertain, i.e. the uncertainty on the pressure does not allow to discriminate between two predicted states:

$${s}_N^p: \begin{cases} Q_0 > 0, P_0 < P^* \\ V_1. m = \text{open} \\ V_2. m = \text{closed} \end{cases} \quad \text{and} \quad {s}_N^g: \begin{cases} Q_0 > 0, P_0 > P^* \\ V_1. m = \text{open} \\ V_2. m = \text{closed} \end{cases}$$

After observing $Q_0 > 0 \land Q = 0$, $A$ returns diagnosis, based on the knowledge of the nominal states above:

$${s}_P: \begin{cases} Q_0 > 0, P_0 < P^* \\ V_1. m = \text{stuck_CLOSED} \\ V_2. m = \text{closed} \end{cases} \quad \text{and} \quad {s}_P^g: \begin{cases} Q_0 > 0, P_0 > P^* \\ V_1. m = \text{stuck_CLOSED} \\ V_2. m = \text{closed} \end{cases}$$

$s_P$ is the faulty state diagnosed from $s_N^p$ while $s_P^g$ has been deduced from $s_N^g$. Hybrid states in $P(A) = (s_N^p, s_N^g)$ and $D(A) = (s_P, s_P^g)$ contain enough information for the autonomous system to extract its functional deficiencies.

### 3 Functional Deficiencies

Given a belief on a model-based system $A$, we extend $P(A)$ and $D(A)$ by the states probabilities such that $P(A) = ((s_N^p, p(s_N^p)), \ldots, (s_N^g, p(s_N^g)))$ is the set of the nominal predicted states, and their associated probabilities, and $D(A) = ((s_P, p(s_P)), \ldots, (s_P^g, p(s_P^g)))$ is the set of faulty states from diagnosis, and their attached probabilities. Given a variable $v$, we note $s(v)$ its value in state $s$. Any set of nominal and faulty states in $S$ is denoted a reconfiguration set. We want to find a set $F$ of prioritized observable instances in $M \cup X$ that are the functional deficiencies between states in $P(A)$ and $D(A)$, and thus need to be recovered. The general idea that is developed in this section has been inspired by the model-based reconfiguration of logical functions in [14].

#### 3.1 Deficient variable instances

Given two states $(s_N, s_F)$ respectively from $P(A)$ and $D(A)$, and a variable $v$, we note $L(s_N(v), s_F(v))$ the measure of the common ground of $v$'s value in each state. We say that variable whose instances in a pair of nominal/faulty states have less common ground than observable variables that discriminated among these states, are deficient. We write it as follows:

$$L(s_N(v), s_F(v)) \leq \frac{\sum_{y \in Y_{\text{misb}}} L(s_N(y), s_F(y))}{\text{nbr}(Y_{\text{misb}})}$$ \hspace{1cm} (2)$$

where nbr$(Y_{\text{misb}})$ is the number of misbehaving observed variables. A misbehaving $y$ is an observed variable that triggered a fault detection, thus discriminating $s_N$ from $s_F$: $y$'s value in $s_F$ better fits $\tilde{y}$ than its value in $s_N$. When relation 2 is satisfied, we say $L(s_N(v), s_F(v))$ is deficient. The definition of $L$ depends on the nature of the variables and the expression of the uncertainty in the model.

In the case variable domains are discrete, as in [16], variable instances have attached boolean labels. Misbehaving variables are observables labeled 1 in $s_N$ and 0 in $s_F$. We set up $L(s_N(v), s_F(v)) = \frac{\sum_{y \in Y_{\text{misb}}} L(s_N(y), s_F(y))}{\text{nbr}(Y_{\text{misb}})}$.

This corresponds to the general case of tracking multiple states simultaneously.

Flows $0 > 0$ are abstracted from their real values for an improved readability.
I otherwise.

In case variable instances are numerical intervals, as in [2], we say $y$ is misbehaving if $p(y | S') p(T(s_N, S_f)) \geq p(y | S_N)$, i.e., if its likelihood is higher in the diagnosed estimate than in the nominally predicted one, given the probability of changing mode. Here $p(T(s_N, S_f)) = p(s_N(\phi_1, \ldots, \phi_r)) \prod_{i=1}^{r} p(\tau_i)$ where $r$ is the number of transitions leaving $S_N$ and reaching $S_f$. Given that $s_N \sim N(m_N, \theta_N)$ and $s_f \sim N(m_F, \theta_F)$, we define $L$ as the measure of the common space enclosed by both density functions $f_N, f_F$. Given $\rho^1, \rho^2$ the two intersection points of these curves, and considering that $\theta_F \geq \theta_N$ (otherwise, the notations are inversed):

$$L(s_N(v), s_F(v)) = \int_{-\infty}^{\rho_1} f_N(v) dv + \int_{\rho_2}^{+\infty} f_N(v) dv + \int_{\rho_1}^{\rho_2} f_F(v) dv$$

(3)

$\rho^1, \rho^2$ are solutions of $f_N(v) = f_F(v)$. In the general case, at the curve intersection points, the Mahalanobis metric $(v - m)^T \theta^{-1} (v - m)$ of both estimates is identical.

### 3.2 Functional Deficiencies

Based on deficient variables, we build the functional deficiencies.

**Definition 2 (Functional deficiency).** A functional deficiency $F$ for a model-based system $A$ over a set of hybrid states $S = (P(A), D(A))$ is a set of variable instances of $M \cup X$ that hold in some states of $P(A)$, and that are deficient in some states of $D(A)$. We denote as $S(F)$ the reconfiguration set associated to $F$. Consider $F$ as a conjunction of $n$ mean value instances as follows:

$$F = \bigwedge_{j=1, \ldots, n} \left( \sum_{i=1, \ldots, p} p(s^1_i \cap s^k_i)(v^i) \right)$$

(4)

then $(s^1_i, s^k_i) \in S(F)$ iff $L(s^1_i(v), s^k_i(v))$ is deficient for all $i, k$.

In other words, $S(F)$ includes all nominal and faulty states whose pairs show a deficiency for all the instances of $F$. $F$ is said to be complete w.r.t. a reconfiguration set $S'$ iff $S' = S(F)$. The complete $F$ over $S$ is unique.

**Property 1.** If $F, F'$ are complete functional deficiencies, then if $F' \in F, S(F') \subseteq S(F)$.

Given two tuples $(F_1, S(F_1))$ and $(F_2, S(F_2))$, we write:

$$(F_1, S(F_1)) \cap (F_2, S(F_2)) = (F_1 \cap F_2, S(F_1) \cup S(F_2))$$

(5)

$$(F_1, S(F_1)) \cup (F_2, S(F_2)) = (F_1 \cup F_2, S(F_1) \cap S(F_2))$$

(6)

We note $F_1 \cap F_2, F_1 \cup F_2$ for short. From now on, we consider a functional deficiency to be complete when not explicitly mentioned otherwise. Also, we sometimes write a functional deficiency as the conjunction of its elements. The tuple $(F, S(F))$ is denoted a reconfiguration tuple. Finally, it is possible to prioritize a functional deficiency:

$$pr(F) = \sum_{i=1}^{n} \sum_{j=1}^{f} p(s^1_i)p(s^k_j), (s^1_i, s^k_j) \in S(F)$$

(7)

**Definition 3 (Core functional deficiency).** The core functional deficiency $F^c$ has its elements satisfied in all states of $P(A)$ and deficient in all states of $D(A)$. The core function is unique for a given set $S$, and its priority is equal to 1.\[8\]

Note that at least all misbehaving variables in states of $S(F)$ do belong to the core deficiency, as does $Q = 0$ in our example.

### 3.3 Minimal functionalities over maximal reconfiguration sets

This section develops a characterization of functional deficiencies whose size is minimal, while deficient over the largest number of state estimates. The reason is that the autonomous system certainly wants to operate minimal changes while covering the maximum states. We begin by characterizing a complete functional deficiency of minimal size.

**Definition 4 (Minimal functional deficiency).** A functional deficiency $F$ is minimal if it exists no functional deficiency $F'$ such that $F' \subset F$ and $S(F') = S(F)$.

We then characterize the maximal reconfiguration set.

**Definition 5 (Maximal reconfiguration set).** A functional deficiency $F$ has a maximal reconfiguration set $S(F)$ if it exists no other functional deficiency $F'$ such that $S(F) \subset S(F')$ and $F' \subset F$.

The search for minimal functional deficiencies over maximal reconfiguration sets leads to a set of functional deficiencies denoted minmax. A minimal functional deficiency represents the minimal set of variable instances that are deficient over the maximum set of pairs of nominal/faulty states.

**Proposition 1.** Given two minmax functional deficiencies $F$ and $F'$ such that $F' \cap F \neq \emptyset$, then $S(F') = S(F)$.

Proof. If $F'' = F' \cap F$ and $F'' \neq \emptyset$, then $F'' \subset F$ and from definition 4, applied to $F$, it comes $S(F) = S(F'')$. Similarly, $S(F') = S(F')$, so $S(F) = S(F')$.

According to definition 2, the completeness of two functionalities $F$ and $F'$ implies that if $S(F) = S(F')$, then $F = F'$. The previous proposition implicitly focuses the search on distinct minmax functionalities. Thus functional deficiencies may be characterized as disjoint sets of variable instances. This result brings flexibility to the reconfiguration process under uncertainty, but is mitigated as the disjoint functions are not independent from each other w.r.t. to the hybrid dynamics. In other words, they may not be recovered independently. In reference to the recovery (planning) operation, these functionalities are no serializable goals.

**Proposition 2.** The core functional deficiency $F^c$ is minmax.

Proof. This is trivial from definitions 4 and 5. $F^c$ is also complete with $S(F^c) = S$.

\[8\] Note that in this expression, there is no notion of fault criticality. Every faulty state is assumed to have equal criticality but the probability of the state is taken into account.

\[8\] Given that $P(A)$ and $D(A)$ have their states probabilities summing to 1.
1. Compute the complete \( F \) w.r.t. each reconfiguration set \( (s^*_N, s^*_F) \), compute \( F^C \), and add them all to the agenda.
2. Iterate through the tuples \((F_1, F_2)\) in the agenda.
3. If \( F^C \cap F_1 = \emptyset \), \( F_1 \rightarrow F \in \{ F_1 \cap F^C \} \).
4. Else if \( F_1 \cap F_2 \neq \emptyset \), create a new function \( F^* = F_1 \cap F_2 \) and add it to the agenda. Do \( F_1 \rightarrow F \in F^C \).
5. Else if \( F_1 = F_2 \), \( S(F_1) = S(F_2) = S(F_1) \cup S(F_2) \) and remove the remaining function \( F_1 \) from the agenda.
6. \( F_1 \) is minimax when it does not intersect with other functions anymore. It is removed to the agenda and returned.

**Algorithm 1:** Computing minimax functional deficiencies

The computation of the minimax functional deficiencies is performed with algorithm 1. Its main principle is to progressively reduce simple non-minimax deficiencies. The first step updates the deficiencies for each combination of two states of \( S \) using the measure of relation 2, and computes the core function. Iterating through this set, step 3 prunes out any deficiency of its intersection with \( F^C \). Step 4 prunes out non-disjoint functionalities of their intersection. Step 5 merges the reconfiguration sets of similar deficiencies.

A word on complexity: given \( p \) nominal and \( q \) faulty states, resulting in \( f \) minimax deficiencies, the first step finds \( pq + 1 \) complete deficiencies. Studying the loop that starts at step 2, we consider an iteration checks all intersections among the \( F_i \) currently in the agenda. Noting \( n_j \) the number of checks at iteration \( j \), we have \( n_j = \lambda_j \sum_{i=1}^{j-1} i^{-1} \), with \( \lambda_j = \frac{n_{j-1}}{\sum_{i=1}^{j-1} i} \), and \( e_j \) is the number of functions eliminated (or added, \( e \) negative). Noting \( \lambda = \frac{1}{2} \sum_{j=1}^{\infty} \lambda_j \), where \( \xi \) is the total number of iterations, we write \( \lambda \approx \frac{1}{j^2} \). It appears that if \( D(A) \) is computed w.r.t. \( P(A) \), then in general \( f = pq \). From that it comes \( \xi \approx \sum_{j=1}^{\infty} j \). Finally, the total number of computed intersections is around \( \sum_{i=1}^{\infty} n_j \), with \( n_0 = pq + 1 \). The algorithm is better understood by developing our example. Step 1 gives:

\[
\begin{align*}
\left( s^1_N, s^1_F \right) : & \quad F_1 = (V_1, m = open) \land Q_1 > 0 \land Q > 0 \\
\left( s^2_N, s^2_F \right) : & \quad F_0 = P_0 < P^* \quad \land (s^1_N, s^1_F) \\
\left( s^3_N, s^3_F \right) : & \quad F_0 = P_0 < P^* \quad \land (s^1_N, s^1_F) \\
\left( s^4_N, s^4_F \right) : & \quad F_0 = P_0 < P^* \quad \land (s^1_N, s^1_F) \\
\left( s^5_N, s^5_F \right) : & \quad F_0 = P_0 < P^* \quad \land (s^1_N, s^1_F) \\
\left( s^6_N, s^6_F \right) : & \quad F_0 = P_0 < P^* \quad \land (s^1_N, s^1_F) \\
\left( s^7_N, s^7_F \right) : & \quad F_0 = P_0 < P^* \quad \land (s^1_N, s^1_F) \\
\left( s^8_N, s^8_F \right) : & \quad F_0 = P_0 < P^* \quad \land (s^1_N, s^1_F) \\
\left( s^9_N, s^9_F \right) : & \quad F_0 = P_0 < P^* \quad \land (s^1_N, s^1_F) \\
\end{align*}
\]

We have \( F_1 = F^C \) so \( F_1 \) can be eliminated. Then reducing other functions with \( F^C \):

\[
\begin{align*}
F_2 &= \quad P_0 < P^* \quad \land (V_0, m = closed) \\
F_3 &= \quad P_0 < P^* \quad \land (V_0, m = open) \\
F_4 &= \quad P_0 < P^* \quad \land (V_0, m = closed) \land (V_2, m = open) \land Q_2 > 0 \\
F_5 &= \quad (V_2, m = open) \land Q_2 > 0 \\
F_6 &= \quad (V_2, m = closed) \land Q_2 > 0 \land (V_2, m = open)
\end{align*}
\]

1. \( F_2 \cap F_3 = F_0 < P^* \land (S.m = open), F_7 \leftarrow F_0 < P^* \land (S.m = open), F_9 \leftarrow F_0 < P^* \land (S.m = closed) \\
2. \( F_2 \cap F_4 = \emptyset, F_2 \cap F_5 = \emptyset, F_2 \cap F_6 = \emptyset, F_5 \leftarrow F_7 \leftarrow F_3 \leftarrow F_7 \leftarrow F_0 < P^* \land (S.m = open) \land Q_1 > 0 \land Q > 0 \\
3. \( F_3 \cap F_4 = \emptyset, F_3 \cap F_5 = \emptyset, F_3 \cap F_6 = \emptyset, F_3 \leftarrow F_7 \leftarrow F_9 \leftarrow F_2 \leftarrow F_0 < P^* \land (S.m = closed) \land Q_1 > 0 \land Q > 0 \\
4. \( F_4 \cap F_5 = F_0 < P^* \land (S.m = open) \land Q_1 > 0 \land Q > 0 \\
5. \( F_5 \leftarrow F_7 \leftarrow F_9 \leftarrow F_2 \leftarrow F_0 < P^* \land (S.m = closed) \land Q_1 > 0 \land Q > 0 \\
6. \( F_6 \leftarrow F_7 \leftarrow F_9 \leftarrow F_2 \leftarrow F_0 < P^* \land (S.m = closed) \land Q_1 > 0 \land Q > 0 \\
\]

Finally, the minimax functions are:

\[
\begin{align*}
F^* &= (V_1, m = open) \land Q_1 > 0 \land Q > 0 \land (S.m = open) \land (S.m = closed) \land (S.m = closed) \\
F_0 &= (S.m = closed) \land (S.m = closed) \land (S.m = closed) \land (S.m = closed)
\end{align*}
\]

At this point, a possible extension to the functional deficiencies is to distinguish the continuous reduction of \( F \), that is its reduction to variables in \( X \), from the hybrid deficiency (made of both discrete and continuous instances). Intuitively, as the modes are relaxed, there exist more states that satisfy the continuous reduction to a deficiency, than the hybrid deficiency. For this reason, we say the latter leads to reset solutions (as modes deficiencies are explicitly set up to be recovered), as opposed to redundancy solutions (modes are unspecified, several component mode switches may be activated to recover the continuous deficiencies). We note \( F \) the continuous reduction to \( F^* \).

### 4 Reconfiguration of Functional Deficiencies

This section focuses on reconfiguring a functional deficiency by identifying a set of goal states, and planning a recovery to those states. Ideally, a goal state specifies a value to all component modes, and may be inferred from the functional deficiency. In the case of a hybrid uncertain state however, the constraints in the form of continuous static/differential equations prevent a unique identification of the modes from a given continuous state point. Hence we propose to rely on an intrinsic property of hybrid systems, that is that the conditional statements \( \phi \) naturally partition their behavioral space into hybrid regions that we refer to as configurations. We refer the reader to [2] for a formalization of these regions.

In the following, we denote as the goal functional deficiency \( F^* \) the functional deficiency to be recovered. Its selection is part of the recovery process, and is detailed at the end of the section. For now, a simple \( F^* \) is \( F^C \) as its priority is maximal, and it covers all state estimates.

Identifying the hybrid regions that enclose the values of \( F^* \) is sufficient to form goals that we refer to as configuration goals (instead of goal states). They correspond to reduced sets of both component modes and equalities/inequalities over continuous variables.
Then, we must ensure that the goals are reachable by both the continuous and discrete dynamics, respectively equations $E$ and transitions $T$.

4.1 Configurations identification

We first enhance the model representation, then determine the goal configurations through a process similar to the model-based diagnosis consistency approach. Indeed, reconfiguration can be viewed as the problem of identifying components whose reconfiguration is sufficient to restore acceptable behavior, when diagnosis is the problem of identifying components whose abnormality is sufficient to explain observed malfunctions [4].

4.1.1 Causal-graph of influences

A first difficulty lies in equations in $E$ that may demand a time-analysis for determining continuous variable values that are not set in $F^*$. A second problem lies in the non-existence of a bijection between modes $M$ and a particular continuous region of the state-space, as constrained by $E$. These problems can be tackled by first enhancing the model-based formalism with a causal representation of $E$.

Definition 6 (Causal-Graph of Influences). The causal-graph of influences of a set of equations $E$ is an oriented graph $G = (X, I)$ where the variables in $X$ form a set of nodes $x_k$ and $I$ a set of arcs among these variables.

The causal-graph is a representation of relations among variables in $E$ that holds at any time step.

Definition 7 (Causal Influence). A causal influence in $I$, $I_{i,j} = (x_i, x_j, b, \phi)$, is a directed arc between two variables $x_i$ and $x_j$, with $b$ the sign of the influence and $\phi$ its activation condition.

Influences are drawn from the implicit causality in $E$. Variables that are subject to no influence are referred to as the inputs of $G$.

Figure 2 pictures the causal-graph of the pressure expansion system. In the following we replace equations in $E$ with $G$.

In general some work is required to extract the causality from static relations [15]. $b = (-1, 1)$ stores the numerical positive or equal/negative influence among variables. $\phi$'s truth value in the hybrid state determines the activation/deactivation of the influence in the graph. Unconditioned, the influence is permanently activated. The activation conditions represent the causality changes in the dynamics.

Definition 8 (Configuration). A configuration for $G$ (and by extension $A$) is of the form $\bigwedge_\phi \phi_i$.

A configuration delimits a region of behavior of $A$. In our example, $V_1.m = \text{open} \land V_2.m = \text{open} \land P_0 \leq F^* \land P_0 \geq P_1 \land P_0 \geq P_2 \land S.m = \text{closed}$ is a nominal configuration of the system.

4.1.2 Building configuration goals from functional deficiencies

We write the MBD theory based on consistency [13] where for the reconfiguration purpose, observations are replaced with functional deficiencies. A deficiency $F^*$ has been characterized w.r.t. the state uncertainty. We are now searching for the minimal sets of conditions that are sufficient to restore $F^*$.

Figure 2. Pressure expansion system causal-graph

Definition 9 (Reconfiguration candidate). A reconfiguration candidate for $A$ given $F^*$ is defined as a minimal set $\Delta = \{I_1^\Delta, \ldots, I_n^\Delta\} \subseteq I$ of influences such that

$$A \cup F^* \cup \neg \phi_1^\Delta \cup \cdots \cup \neg \phi_n^\Delta$$

is consistent.

Definition 10 (Reconfiguration conflict). A reconfiguration conflict for $A$ given $F^*$ is a set $\lambda = \{I_1^\lambda, \ldots, I_n^\lambda\}$ of influences such that

$$A \cup F^* \cup \phi_1^\lambda \cup \cdots \cup \phi_n^\lambda$$

is not consistent.

From $G \cup F^*$, we seek for reconfiguration conflicts in $G$ that are such that influences in a conflict cannot be activated together given $F^*$. For a deficient variable (node) $x_j$ of $F^*$, we call ascending influences the influences that belong to the paths from the inputs/other deficient variables, to $x_j$. An ascending influence for $x_j$ is noted $\lambda^j = \{I_i, \phi_i\}$. A conflict for $x_j$ is thus the set $\lambda^j$ of its ascending influences $\{\lambda_i^j\}_{j=1, \ldots, n_j}$. $\Lambda = \{\{\lambda_i^j\}\}_{j=1, \ldots, n_j}$ is the collection of conflicts over all deficient variables of $F^*$. The minimal set of influences $\Delta$ that are candidates to the reconfiguration is obtained similarly to the diagnoses in the MBD theory by computing the hitting sets ($HS$) over $\Lambda$ [13]. We note $\Delta_p = (I_{\phi_A} \land \mathcal{E}_a \phi_A)$ a diagnostic candidate, where $I_{\phi_A}$ is a set of influences. Consequently, $\Delta = \{\Delta_{\phi_A}\}_{\phi_A \in X_\phi}$. We note $\neg \Delta = \{\neg \Delta_{\phi_A}\}_{\phi_A \in X_\phi}$.

Algorithm 2: Identifying reconfiguration candidates (Goals)

1. Apply $F^*$ to $G$.
2. Apply $S_p(F^*)$ to $G \setminus F^*$.
3. Get the conflicts $\Lambda$.
4. Compute $\Delta = HS(\Lambda)$.
5. $\neg \Delta \land F^*$ are goal configurations.

Consider our example again. Reconfiguring $F^* = F^a$ with algorithm 2 implies $\phi_1^\lambda$ is satisfied (step 1), and based on remaining variable instances in states in $S_p(F^*)$ the configuration of the subgraph $G \setminus F^*$ ($G$ deprived of nodes and axis to nodes in $F^*$) is determined, in that case $\neg \phi_2^\lambda$ is satisfied (step 2). Tracing the ascending
influences in \( G \), it comes two sets of conflicts (one per continuous variable instance in \( F^c \)):

\[
\lambda_2 = \{ Q \rightarrow Q_1, Q, Q_3, Q \rightarrow Q_2, 0, P_2 \rightarrow P_{atm} \}
\]

\[
\lambda_{Q_1} = \{ Q_1 \rightarrow P_1, Q_1 \rightarrow P_1, P_1 \rightarrow P_{atm} \}
\]

\( \phi_1 \) is satisfied in \( F^c \), and influences over \( Q, P_1 \) and \( P_2 \) are activated in all configurations, so it simplifies to:

\[
\lambda_2 = \{ Q_2 \rightarrow Q_2, 0 \} \quad \Lambda = \{ \lambda_Q, \lambda_{Q_1} \}
\]

It comes \( \Delta = \{ \{ \phi_2 \} \} \) and \( \phi_2 \land F^c \) thus is a valid goal configuration (step 5).

Reconfiguring the continuous reduction \( \bar{F}^c \) leads to more opportunities: \( \phi_1 \) is no more satisfied and \( \lambda_{Q_1} = \{ -\phi_1 \} \), thus \( \Delta = \{ \{ -\phi_1, -\phi_2 \} \} \) and configuration goals are given by \( \phi_1 \land \phi_2 \land F^c \).

### 4.2 Recovery

The recovery operation aims at bringing the system into the regions defined by the configuration goals. In our case, due to the hybrid dynamics, this process implies a chain of transitions exist to the component mode goals, while the continuous dynamics ensure the transition guards are successively satisfied. Sets of component transitions \( T_0, \ldots, T_p \) must satisfy

\[
A \cup D(A) \cup T_0 \cup \ldots \cup T_p \cup F^* \cup \neg \Delta \tag{10}
\]

is consistent, where the current time of the system is set to \( k_0 \) and the initial state belongs to \( D(A) \). \( Pl = \{ T_0, \ldots, T_p \} \) is a plan for the recovery. Noting \( k_0 \) the time at which transition \( T_p \) triggers, the continuous dynamics must satisfy

\[
\begin{align*}
X(k_0) & \cup \phi_0 \\
E(X(k_0)) & \cup \phi_1 \\
E(X(k_1)) & \cup \phi_2 \\
& \vdots \\
E(X(k_{p-1})) & \cup \phi_p \\
E(X(k_p)) & \cup F^*
\end{align*} \tag{11}
\]

are consistent, where \( E(X(k_0)) \) refers to the dynamics of relation (1), is conditioned by \( \phi_{p+1} \), and \( X(0) = \sum\phi_0 \in D(A) p(\phi_0) X_p(0) \). We say relations (10) and (11) define a hybrid system planning problem. To our knowledge, the planning of hybrid systems has received no attention yet. We believe that its development will be made necessary by several on-line applications.

Relation (10) defines a probabilistic conformant planning problem [8], where a set of transitions must bring the system to a set of predetermined goals, under uncertainty and without observing directly the system state. The plan maximizes the probability of the goal configuration given the initial belief state \( D(A) \). In our example, a stuck valve cannot be re-opened, so no plan exists for functionalities \( F^c \). A plan exists to \( \bar{F}^c \) for some initial states, \( Pl = \{ \tau_2, \tau_21 \} \). \( F_0 \) has a plan \( Pl = \{ \tau_3 \} \).

Relation (11) defines a control problem where the continuous dynamics must be forced to successive \( \phi_d \) through available inputs. A model predictive control problem (MPC) solves on-line a finite horizon open-loop optimal control problem subject to system dynamics and constraints involving states and controls. Based on measurements obtained at time \( k \), the future dynamic behavior of the system is predicted over a fixed horizon, and the controller determines the input such that a performance criterion is optimized. This technique fits well within the model-based autonomous system framework, given two key elements are already present, the model \( A \), and the state predictor (or estimator) \( \bar{P}(A) \). By using control and measurement horizons of a single time step, a basic formulation of the MPC problem at time \( k \) is

\[
U^*(k + 1) = \min_{U} \sum_{i=1}^{k+1} J(X(k), U(k))
\]

\[
J(X(k), U(k)) = \int_{k}^{k+1} F(X(t), U(t)) dt
\]

\[
F(X, U) = (X - X_s)^T Q(X - X_s) + (U - U_s)^T R(U - U_s)
\]

\[
X(k + 1) = f(X(k), U^*(k))
\]

where \( Q \) and \( R \) denote positive definite symmetric weighting matrices, and \( U^*(k + 1) \) is the optimal input used in the prediction at \( k + 1 \). Considering \( \bar{F} \) over \( X \) in the form \( \bar{F}(X) \geq 0 \), we note \( \bar{F}(X) + \varepsilon \) if its reduction to an equality, where \( \varepsilon \) is a term that ensures the threshold is later satisfied. The function is evaluated at \( k \) with \( \bar{F}(k) : \bar{F}(X(k)) + \varepsilon \), and we note its inverse \( \bar{F}^{-1}(k) \). The MPC application to the control objective \( \bar{F} \) sets the setting point \( X_s, U_s \) to \( (\bar{F}^{-1}(k), 0) \). In our example, \( \tau_3 \)'s guard has \( \bar{F}^{-1}(k) = F^* + \varepsilon \).

Again, we face the fact that \( P(A)(k) = \{ s^1, \ldots, s^2 \} \) likely contains multiple state estimates. Thus the minimization must apply to each \( F(X^*(k), U(k)) \), returning \( U^*(k + 1) \). We merge the optimized input candidates according to the states estimated probabilities:

\[
U^*(k + 1) = \sum_{i=1}^{\#} p(X^i(k)) U^*(k + 1) \tag{12}
\]

Finally, when \( \phi_j \) is reached, transition \( T_1 \) should trigger, and MPC then focuses on \( \phi_{j+1} \). The last MPC set-point is \( F^* \).

Solving this control problem however requires more research. First, the MPC community itself seeks for better integration of modern state estimation techniques within the control loop [10]. Second, \( \phi \)'s inverse is a problem in practice. The control could focus on bringing the system state back to the geometrical center of the goal configuration region instead. This is yet to be explored. Third, optimality and especially, stability problems, if far out of the scope of this paper, must be tackled in the case of control based on multiple state estimates. Modern hybrid state estimators should be coupled with powerful techniques such as Quasi-Infinite Horizon NMPC [3]. Note that recent developments also pave the way for powerful stability and safety/reachability analysis of these controllers [1].

### 4.3 Reaching the goals: safety and convergence

Considering the context of a faulty system, the reconfiguration process should likely be safe, not making the situation worse. In our case, the goal configurations identification may produce multiple solutions, while not ensuring that they are reachable. In this section we improve algorithm 2 by reducing the number of goal solutions that are guaranteed to be reachable under monotonous continuous dynamics. To ensure the latter, and given a variable \( v \) that appears in \( F^* \) (instance \( v^* \)), the sign of \( \langle S_N(u), S_F(v) \rangle \) is studied, where \( \langle S_N, S_F \rangle \) is the reconfiguration set of \( F^* \). Here, we use \( v^* - \sum_{\phi \in D(A)} p(\phi^e) e_{\phi}(v) \). Algorithm 2 is modified such that \( \Lambda \) becomes \( \Lambda^- \); the set of influences to be deactivated, while \( \Lambda^+ \), the set of influences to be activated is constructed as follows:
Given a path of ascending influences \( \{I_{i_1}, \ldots, I_{i_k}\} \) from \( x_1 \) to \( x_2 \) involved in \( F^* \), if \( x_1(S_N(x_1) - S(x_1)) \prod_{i_{k-1}}b_{i_{k-1}} > 0 \), then for each \( \phi_k \) that is not satisfied, add \( I_{i_{k+1}} \) to \( \Lambda^+ \).

- Otherwise, if the above criterion is not satisfied, while \( \phi_k \) is, then add \( I_{i_{k+1}} \) to \( \Lambda^- \).

This corresponds to activating every ascendant path whose combined influences have a beneficial effect to the restoration of \( F^* \). The approach is conservative as the test equality to 0 is not considered.

1: Apply \( F^* \) to \( G \).
2: Apply \( SE(F^*) \) to \( G \) \( \to F^* \).
3: Get the conflicts \( \Lambda^+ \), \( \Lambda^- \).
4: Compute \( \Delta^+ = HS(\Lambda^+) \) and \( \Delta^- = HS(\Lambda^-) \).
5: Do \( \Delta = \Delta^+ \lor \Delta^- \) and eliminate inconsistent configurations.
6: \( \Delta \land F^* \) are goal configurations.

Algorithm 3: Identifying reconfiguration candidates (SafeGoals)

Back to our example, we reconfigure \( F_3 \) to \( Q_2 > 0 \). Step 3 of algorithm 3 gives \( \lambda_{Q_2}^{+} = \{Q_2 \in P_0 \}, \lambda_{Q_2}^{-} = \{Q_4 \in -P_0 \} \), thus \( \Delta^+ = \{\{P_0\}\} \), \( \Delta^- = \{\{\neg P_0\}\} \). The solution is the same as returned by algorithm 2 but it is now ensured that opening \( V_2 \) brings the flow back into the right direction.

The safety may not be ensured when negative and positive effects to a variable are activated via the same condition, as over reductions in prioritized order. In our example, \( P_{stem} \) was not considered being a constant, a numerical analysis would have been required here.

4.4 Prioritized selection of functional deficiencies

Our general strategy to the reconfiguration of the functional deficiencies explores reset solutions first, then redundancy solutions (continuous reductions) in prioritized order. In case of plan failure the next deficiency is selected (algorithm 4). In our example, \( s^F_k \) and \( s^P_k \) have much lower probability than \( s^F_k \) as they correspond to doublefaults. \( F^* \) is subject to plan failure. \( F_0: S.m = closed \) is its own goal configuration and has a plan \( \tau_3 \) whose guard is \( P_0 \geq P^* \). MPC generates the pressure input \( P_0 \) to reach that level. Note that depending on the real initial state, the reconfiguration may have no effect. The operation does not harm the system though we consider that maintaining a nominal level of pressure does not harm even the fault system, and may help discriminate among the estimates. For example, if reconfiguring \( F_0 \) fails, \( s^F_k \), and potentially \( s^P_k \) are eliminated.

Algorithm 4: Prioritized selection of functional deficiencies

5 Summary, Existing works and Perspectives

We've presented a methodology to the automated reconfiguration of functional deficiencies. The deficiencies are identified by comparing predicted and diagnosed states, and then partitioned and prioritized over the state estimates. Goals are further identified from the deficiencies. Planning and MPC techniques are used in common to move the system toward the goals.

REFERENCES


