A Causal Analysis Method for Concurrent Hybrid Automata

Michael W. Hofbaur
Institute of Automation and Control
Graz University of Technology, Austria
michael.hofbaur@TUGraz.at

Franz Wotawa
Institute of Software Technology
Graz University of Technology, Austria
wotawa@ist.TUGraz.at

Abstract

Modern artifacts are typically composed of many system components and exhibit a complex pattern of continuous/discrete behaviors. A concurrent hybrid automaton is a powerful modeling concept to capture such a system's behavior in terms a concurrent composition of hybrid automata for the individual system components. Because of the potentially large number of modes of the concurrent automaton model it is non-trivial to validate the composition such that every possible operational mode leads to a causally valid dynamic model for the overall system. This paper presents a novel model analysis method that validates the automata composition without the necessity to analyze a prohibitively large number of modes. We achieve this by formulating the exhaustive causal analysis of hybrid automata as a diagnosis problem. This provides causal specifications of the component automata and enables us to efficiently calculate the causal relationships for their concurrent composition and thus validate a concurrent automaton model.

Introduction

Many modern artifacts, mobile robotic devices, space probes, or production plants exhibit complex patterns of behavior in order to satisfy the high demand on performance and durability. Key for the artifact's operation is a sophisticated control and automation system that orchestrates the many components of the artifact through closed loop, system-wide interaction. Our research objective is to provide a novel approach for model-based automation of complex systems. The goal is to build autonomous artifacts that reason quickly, extensively and accurately about the world and react to novel or unforeseen situations. Key for this approach is a modeling methodology that enables us to model the complex continuous/discrete behavior of individual system components and their interaction according to the system's blue-prints. In our previous work on hybrid estimation of complex systems (Hofbaur & Williams 2002; 2004; Hofbaur 2005) we developed such a modeling framework, the concurrent probabilistic hybrid automata (cPHA). Within this framework, we model individual system components as probabilistic hybrid automata (PHA) and describe the model for the overall system as a concurrent composition of PHAs which, together with the specification of the system's interconnection to the outside world, comprises the cPHA model of the artifact.

The composition methodology is in spirit of the reactive module framework of Alur and Henzinger (Alur & Henzinger 1999) with one major difference. We do not classify I/O variables of individual automata, that establish the interconnection to the other system components, as inputs or outputs. The system-wide context will specify whether a variable serves as an input or output for a component automaton. This relaxation of the automaton specification increases the expressiveness of our modeling framework and is particularly important within our context of autonomous automation where we allow the system to reconfigure itself to cope with novel operational or fault situations.

For example, we apply our automation framework to build fault tolerant micro-chemical plants. Figure 1 shows the simplified blue-print of such a system that continuously produces a chemical substance out of two reactants. The main reaction is due to the mixture of the substances in the micro-mixers M1 and M2. During normal operation, these devices receives raw substance A and B on the terminals 1 and 2, respectively, and provide a mixture at terminal 3 that leads to the output tank. This nominal operation implies that we model a micro-mixer with inputs 1 and 2 and the output 3.

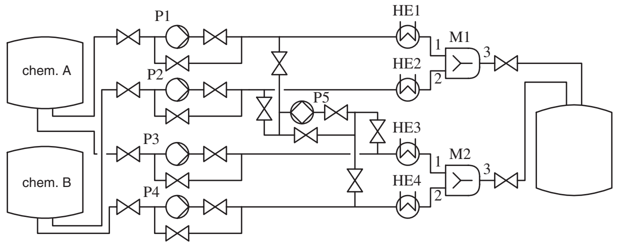

Figure 1: Micro-chemical plant.

In order to guarantee continuous chemical production, we can think of several reconfigurations that work round possible device-faults. For example, Figure 2 shows an un-conventional reconfiguration that works round a multiple fault that impairs the two pumps (P2, P3) and the two heat exchangers (HE2, HE4). The configuration reroutes the chemical substance B via the mixer M2 with closed output valve into the heat exchanger HE3 that is usually used for chemical substance A and uses the redundancy interconnection between the two processing branches to inject the substance B into the mixer M1. Thus, we use the mixer M2 not as a mixing device but only to re-route the chemical substance. This is equivalent to a mode where terminal 2 serves as the input and terminal 1 serves as the output. To model such an operational condition, it is therefore helpful not to pre-specify the causality of system components.

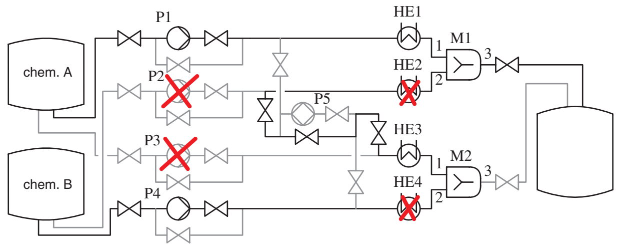

Figure 2: Reconfigured micro-chemical plant.

This flexibility of our modeling framework implies that we cannot directly check the interconnection topology for possible conflicts where two components specify a shared I/O variable as output variable, thus causing a causal conflict that indicates a modeling fault. Since our automation framework utilizes the hybrid model on-line to interpret the operational condition and to deduce an appropriate command sequence, we have to avoid situations where our component-based model fails to provide a valid description of the overall system. Typically, the number of modes for the overall system is typically very large (e.g. 10^6) and it is infeasible to perform a causal analysis for all of them individually. Therefore, we provide a novel mechanism that (a) deduces all possible causalities for individual components and (b) checks whether the composition of two components leads to causally valid overall models or not. By recursively applying the latter operation onto the system's component models, we can effectively check complex models of physical artifacts.

Related Work

Verification of hybrid systems, for example (Henzinger & Ho 1996; Chutinan & Krogh 1999; Mosterman, Biswas, & Sztipanovits 1998), almost always deals with the verification of the behavioral patterns that a hybrid system can possess (e.g. safety, stability, etc.) and not the verification of the model itself. Furthermore, typical frameworks for hybrid systems and their composition, such as (Alur & Henzinger 1999; Lynch, Segala, & Vaandrager 2001), define continuously valued I/O variables statically as input or output of the automaton. Our framework, however, does not make this assumption which adds to its expressiveness and flexibility.

Whereas, for example (de Kleer & Brown 1986; Iwasaki & Simon 1986; Nayak 1995), provide foundations and algorithms for causal analysis, (Travé-Massuyès & Pons 1997) first discussed the causal analysis problem for systems with multiple modes. They provide an algorithm for incremental generation of the causal structure that avoids the brute force approach of generating new causal structures for every different mode. The major difference of these traditional causal analysis methods to our work is that we intend to operate on the sub-system level, whereas traditional methods perform the analysis on the equation level. Causal analysis on the equation level classifies *one* variable out of the n variables that are related by the equation as the dependent (output) variable. On a subsystem level, however, it is possible that $m \leq n$ variables act as outputs to the subsystem. Thus, we cannot directly apply the standard causal analysis algorithms but build upon them instead.

The work of (Flaus & Gentil 2003) provides an alternative causal analysis method for hybrid systems on the sub-system level. The method classifies the I/O variables of individual automata as (a) inputs, (b) outputs or (c) indifferent variables that can serve either as an input or as an output. This analysis does not provide the level of detail that we are looking for since it fails to specify interdependences among the indifferent I/O variables as we shall see later in Section .

Causality Analysis of Concurrent Hybrid Automata

Hybrid Automata

We start by introducing a hybrid automata model that describes a generalized version of the probabilistic hybrid automata modeling paradigm that we introduced in (Hofbaur & Williams 2002; Hofbaur 2005) and omits details that are irrelevant within the context of this paper.

As a preliminary step towards defining a hybrid model for a complex artifact, we first define the hybrid automata (HA) model for individual system components. These models will then serve as the building blocks for an overall model, the concurrent hybrid automaton (cHA). More specifically, we define a component automaton as:

Definition 1 A *discrete-time hybrid automaton (HA)* $\mathcal{A}$ can be described as a tuple $\langle \mathbf{x}, \mathbf{w}, F, T, \mathcal{X}_d, \mathcal{U}_d, T_s \rangle$:

- $\mathbf{x}$ denotes the hybrid *state variables* of the automaton[1], composed of $\mathbf{x} = \mathbf{x}_d \cup \mathbf{x}_c$. The discretely valued state variables $\mathbf{x}_d = \{x_{d1}, \ldots, x_{dn_m}\}$ denote the *mode* of the automaton and have finite domain $\mathcal{X}_d = \{\mathbf{m}_1, \ldots, \mathbf{m}_l\}$. The continuously valued state variables $\mathbf{x}_c = \{x_{c1}, \ldots, x_{cn_x}\}$ capture the dynamic evolution of the automaton.
- The set of *I/O variables* $\mathbf{w} = \mathbf{w}_d \cup \mathbf{w}_c$ composed of discrete (input) variables $\mathbf{w}_d = \{u_{d1}, \ldots, u_{dn_c}\}$ (also called *command variables*) with finite domain $\mathcal{U}_d$ and the set of continuously valued I/O variables $\mathbf{w}_c = \{w_{c1}, \ldots, w_{cn_w}\}$.
- The set-valued function $F : \mathcal{X}_d \rightarrow 2^{\mathcal{F}_{DE}} \times 2^{\mathcal{F}_{AE}}$ specifies the *continuous evolution* of the automaton in terms of sets of *discrete-time difference equations* $F_{DE} \subseteq \mathcal{F}_{DE}$ and *algebraic equations* $F_{AE} \subseteq \mathcal{F}_{AE}$ for each mode $\mathbf{m}_j \in$

[1]When clear from context, we use lowercase bold symbols, such as $\mathbf{v}$, to denote a *set* of variables $\{v_1, \ldots, v_l\}$, as well as a *vector* $[v_1, \ldots, v_l]^T$ with components v_i.

$\mathcal{X}_d$. We call the set of equations $F(\mathbf{m}_i)$ the *raw model* of the HA at the mode $\mathbf{m}_i \in \mathcal{X}_d$. T_s denotes the sampling-period of the discrete-time difference equations[2].

- The set-valued function T specifies the discrete evolution (mode transitions) of the automaton.

In the following we denote the components of a specific HA specification $\mathcal{A}_j$ by using the same subscript, that is, $\mathbf{x}_j = \mathbf{x}_{d\,j} \cup \mathbf{x}_{c\,j}$, $\mathbf{w}_j = \mathbf{w}_{d\,j} \cup \mathbf{w}_{c\,j}$, F_j, T_j, $\mathcal{X}_{dj}$, $\mathcal{U}_{dj}$, T_{sj} but avoid double indexing for variables when referring to individual entities of $\mathbf{x}$ or $\mathbf{w}$.

To give an example for a HA specification consider the automaton

$$\begin{aligned}\mathcal{A}_1 =&\langle\{x_{d\,1}, x_{c\,1}, x_{c\,2}\}, \{w_{d\,1}, w_{c\,1}, w_{c\,2}, w_{c\,3}\},\\ &F_1, T_1, \{m_{11}, m_{12}\}, \mathcal{U}_{d1}, T_{s1}\rangle,\end{aligned} \tag{1}$$

with the following set of equations for the mode m_{11}:

$$\begin{aligned}F_{1,DE}(m_{11}) =&\{x_{c\,1,k+1} = 0.6\; x_{c\,1,k} + w_{c\,1,k} + w_{c\,3,k},\\ &x_{c\,2,k+1} = 0.1\; x_{c\,1,k} + 0.8\; x_{c\,2,k}\},\\ F_{1,AE}(m_{11}) =&\{w_{c\,2,k} = x_{c\,2,k} + w_{c\,1,k}\}.\end{aligned} \tag{2}$$

If we assume that $w_{c\,1,k}$ and $w_{c\,3,k}$ are input variables and $w_{c\,2,k}$ is an output variable we obtain the following mathematical model for $\mathcal{A}_1$ at the mode m_1:

$$\mathbf{x}_{c,k+1} = \begin{bmatrix} 0.6 & 0 \\ 0.1 & 0.8 \end{bmatrix} \mathbf{x}_{c,k} + \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \mathbf{u}_{c,k} \tag{3}$$

$$\mathbf{y}_{c,k} = \begin{bmatrix} 0 & 1 \end{bmatrix} \mathbf{x}_{c,k} + \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{u}_{c,k}, \tag{4}$$

where $\mathbf{u}_c := [w_{c\,1}, w_{c\,3}]^T$ specifies the vector of input variables and $\mathbf{y}_c := [w_{c\,2}]$ specifies the output. This causality is not unique. Because of the direct relation between $w_{c\,1}$ and $w_{c\,2}$ in the algebraic equation of (2) it is equally possible that $w_{c\,2}$ and $w_{c\,3}$ serve as the inputs of the automaton with output $w_{c\,1}$. The automaton's interconnection to other automata and to the outside world will determine the exhibited causality at mode m_{11}.

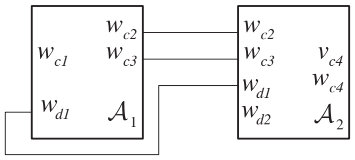

Figure 3: HA composition example.

Figure 3 shows such an interconnection of $\mathcal{A}_1$ with an automaton $\mathcal{A}_2$ that is specified as:

$$\begin{aligned}\mathcal{A}_2 =&\langle\{x_{d\,2}, x_{c\,3}, x_{c\,4}\}, \{u_{d\,1}, u_{d\,2}, w_{c\,2}, w_{c\,3}, w_{c\,4}\},\\ &F_2, T_2, \{m_{21}, m_{22}\}, \mathcal{U}_{d2}, T_{s2}\rangle.\end{aligned}$$

Let us assume that two automata $\mathcal{A}_1$ and $\mathcal{A}_2$ operate at the same sampling period ($T_{s1} = T_{s2}$) and have distinct state variables $\mathbf{x}_{c\,1} \cap \mathbf{x}_{c\,2} = \emptyset$. Then, the (concurrent) composition $\mathcal{A}_1 \parallel \mathcal{A}_2$ of two automata leads to a new automaton

$$\mathcal{A} = \langle \mathbf{x}, \mathbf{w}, F, T, \mathcal{X}_d, \mathcal{U}_d, T_s\rangle,$$

with components specified as follows:

$$\begin{array}{rcl} \mathbf{x} & := & \mathbf{x}_d \cup \mathbf{x}_c, \text{ with } \mathbf{x}_d := \mathbf{x}_{d\,1} \cup \mathbf{x}_{d\,2}, \\ & & \mathbf{x}_c := \mathbf{x}_{c\,1} \cup \mathbf{x}_{c\,2}, \\ \mathbf{w} & := & \mathbf{w}_1 \cup \mathbf{w}_2, \\ F & : & F(\mathbf{x}_d) := F_1(\mathbf{x}_{d\,1}) \cup F_2(\mathbf{x}_{d\,2}), \\ T & : & T(\mathbf{x}_d) := T_1(\mathbf{x}_{d\,1}) \times T_2(\mathbf{x}_{d\,2}), \\ \mathcal{X}_d & := & \mathcal{X}_{d1} \times \mathcal{X}_{d2}, \\ \mathcal{U}_d & := & \mathcal{U}_{d1} \times \mathcal{U}_{d2}, \\ T_s & := & T_{s1}. \end{array}$$

The composition does not ensure that the composite automaton $\mathcal{A}$ provides for every possible mode $\mathbf{m}_i \in \mathcal{X}_d$ a *complete* set of equations (raw model) $F(\mathbf{m}_i)$ that has as many equations as parameters, and no subset of equations has fewer parameters than equations (see (Nayak 1995), Definition 3.4). Therefore, we shall call two hybrid automata *compatible* if their composition provides raw models that represent complete sets of equations, more specifically:

Definition 2 (HA compatibility) We call two hybrid automata $\mathcal{A}_1$ and $\mathcal{A}_2$ with distinct state variables ($\mathbf{x}_{c\,i} \cap \mathbf{x}_{c\,j} = \emptyset$, $i \neq j$) and the same sampling period $T_{s\,1} = T_{s\,2}$ *strongly compatible*, if and only if the composite set of raw equations $F_1(\mathbf{x}_{d\,1}) \cup F_2(\mathbf{x}_{d\,2})$ specifies a *complete set of equations* for *every possible mode combination* $\mathbf{x}_{d\,1} \in \mathcal{X}_{d1}, \mathbf{x}_{d\,2} \in \mathcal{X}_{d2}$. If the set $F_1(\mathbf{x}_{d\,1}) \cup F_2(\mathbf{x}_{d\,2})$ specifies a complete set of equations for at *least one mode combination* $\mathbf{x}_{d\,1} \in M_1 \subset \mathcal{X}_{d1}$, $\mathbf{x}_{d\,2} \in M_2 \subset \mathcal{X}_{d2}$, then we call the automata *weakly compatible*.

The overall *concurrent hybrid automaton (cHA)* model for a physical artifact specifies the concurrent composition of its component automata $\mathcal{A} = \mathcal{A}_1 \parallel \mathcal{A}_2 \parallel \cdots \parallel \mathcal{A}_\zeta$ together with the interconnection to the outside world. The concurrent composition of ζ component automata, can be defined recursively as

$$\mathcal{A} = (\ldots((\mathcal{A}_1 \parallel \mathcal{A}_2) \parallel \mathcal{A}_3) \parallel \cdots \parallel \mathcal{A}_\zeta).$$

This leads to an exponentially increasing number of modes for the overall system model. Because of this potentially large number of modes, our model-based automation system performs most of the model analysis and control synthesis tasks on-line. Nevertheless, we want to validate the hybrid model at compile-time to ensure that the automation system does not drive the artifact into a state, where the hybrid model provides an unsolvable or even meaningless set of equations. Enumeration of all possible modes and consecutive analysis of the associated set of equations is not an option. Therefore, we do present an efficient compile-time analysis method that checks the compatibility of the automata composition. The analysis builds upon (a) an exhaustive causal analysis of component automata and (b) analysis of the causalities for a composite automaton $\mathcal{A}_1 \parallel \mathcal{A}_2$ based on the causalities of the individual automata $\mathcal{A}_1$ and $\mathcal{A}_2$.

[2]We use discrete-time equations to describe continuous dynamics since our automation framework currently supports discrete-time models only. However, the following causal analysis can be equally applied to continuous time models, where the dynamic part $\mathcal{F}_{DE}$ defines ordinary differential equations.

Calculating All Possible Causalities for a HA

To start our compatibility test, we first perform an exhaustive causal analysis of an individual hybrid automaton $\mathcal{A} = \langle \mathbf{x}, \mathbf{w}, F, ... \rangle$. This analysis separates the continuous I/O variables $\mathbf{w}_c \subset \mathbf{w}$ of $\mathcal{A}$ into disjoint sets of *input variables* $\mathcal{U} \subset \mathbf{w}_c$ and *output variables* $\mathcal{Y} \subset \mathbf{w}_c$. Thus, the causal analysis specifies the *independent* I/O variables ($\mathcal{U}$) and the *dependent* I/O variables ($\mathcal{Y}$) of the automaton according to the equations F.

Recall the hybrid automaton (1) given above with the continuous I/O variables $\mathbf{w}_c = \{w_{c\,1}, w_{c\,2}, w_{c\,3}\}$. The intuitive analysis of the raw model for mode m_{11} (2) provided one possible separation of $\mathbf{w}_c$ into the two sets $\mathcal{U} = \{w_{c\,1}, w_{c\,3}\}$ and $\mathcal{Y} = \{w_{c\,2}\}$. However, we noted that this separation is not unique and that other separations exist. Our aim is to find *all of them* and record them as

$$\mathcal{C} = \{C_1, C_2, \dots C_\eta\}, \tag{5}$$

where C_i specifies a specific I/O causality of $\mathcal{A}$, more specifically:

Definition 3 We define an *I/O causality* C_i of a hybrid automaton $\mathcal{A}_j$ as the triple $\langle \mathcal{U}_i, \mathcal{Y}_i, \mathcal{M}_i \rangle$. The set $\mathcal{M}_i \subseteq \mathcal{X}_{d\,j}$ specifies the modes where the set $\mathcal{U}_i \subseteq \mathbf{w}_{c\,j}$ denotes those I/O variables $w_{c\,\iota} \in \mathcal{U}_i$ that serve as the input variables (or *independent variables*) and the remaining continuous I/O variables $\mathcal{Y}_i := \mathbf{w}_{c\,j} \backslash \mathcal{U}_i$ are the output variables (or *dependent variables*) of the automaton.

An exhaustive causal analysis determines *all possible I/O causalities* for a hybrid automaton. This involves iterating through the set of modes $\mathcal{X}_d$ and determining for every mode $\mathbf{m}_i \in \mathcal{X}_d$ the possible input/output separations of $\mathbf{w}_c$ according to the set of equations $F(\mathbf{m}_i)$.

This approach seems feasible for HA models of individual system's components that typically have a moderate number of modes. Nevertheless, enumerating all possible I/O causalities for a hybrid automaton might seem counterproductive at the first view as the possible number of distinct I/O causalities increases exponentially with the number n_w of I/O variables (2^{n_w}), and of course, we have to evaluate the causalities for all l modes of the automaton. Thus, we need a method that explores possible I/O causalities carefully.

Efficient causal analysis algorithms, e.g. (Nayak 1995), provide only one possible causal mapping among the automaton's variables or signals a failure, whenever the set of equations is incomplete. However, it can efficiently test whether a subset of I/O variables $\mathcal{U}_i$ can serve as exogenous variables for a set of equations $F(\mathbf{m}_j)$ or not. Specifying I/O variables $w_{c\,j} \in \mathcal{U}_i$ as inputs adds extra constraints to the equation set. The causal analysis of this extended set can be used to classify the set of equations as *over-constrained*, whenever all of the variables $\mathcal{U}_i$ cannot serve as the inputs to the automaton at the particular mode $\mathbf{m}_j$ under investigation. This knock-out criteria allows us to utilize efficient algorithms from consistency-based diagnosis to carefully explore the 2^{n_w} possible input sets $\mathcal{U}_i \subseteq \mathbf{w}_c$.

The alternative sub-system based causal analysis method (Flaus & Gentil 2003) that classifies the I/O variables as inputs, outputs or indifferent variables fails to express the interdependence of the latter and does not provide the detailed analysis that we are looking for. For example, the variables $\{w_{c\,1}, w_{c\,2}\}$ in the automaton (2) at mode m_{11} can serve as input or output. However, enforcing $w_{c\,1}$ to act as an input for the automaton implies that $w_{c\,2}$ acts as output, and vice versa.

A diagnosis based exhaustive causal analysis

Model based diagnosis (Hamscher, Console, & de Kleer 1992) is one way to analyze artifacts whenever their expected behavior is inconsistent with observations (OBS). The artifact is modeled in terms of a set of components ($COMPS = \{c_1, \dots, c_\zeta\}$) and its system description (SD). The diagnostic problem is to determine possible *diagnoses* that are *minimal sets* Δ_i of *abnormally* working components ($\Delta_i \subseteq COMPS$). Model-based diagnosis solvers, such as Reiter's algorithm (Reiter 1987; Greiner, B., & Wilkerson 1989) or GDE (de Kleer & Williams 1987) utilize the fact that abnormal components are logically determined by the normal components of the system (Reiter 1987). This allows one to utilize a theorem prover (TP) as the underlying inference method that determines, whether the assumption that a set of components $\{c_1, \dots, c_j\}$ works correctly is consistent with model (SD) and the observations (OBS), or whether the set manifests a so-called *conflict*. These conflicts are then used to guide the exploration for possible diagnoses.

Our exhaustive causal analysis problem is very similarly structured. Out of a set of I/O variables $\mathbf{w}_c$, we have to identify maximal subsets $\mathcal{U}_i \subseteq \mathbf{w}_c$ that act as inputs to the automaton at a particular mode $\mathbf{m}_k$. This is equivalent to identifying minimal subsets $\mathcal{Y}_i = \mathbf{w}_c \backslash \mathcal{U}_i$ that describe possible outputs of the automaton. In order to do so, we test whether defining a set of possible inputs $\{w_{c\,1}, \dots, w_{c\,j}\}$ leads to an over-constrained set of equations

$$\hat{F} = \{\mathrm{ex}(w_{c\,i}) \,|\, w_{c\,i} \in \{w_{c\,1}, \dots, w_{c\,j}\}\} \cup F(\mathbf{m}_k). \tag{6}$$

With $\mathrm{ex}(w_{c\,i})$ we denote the exogenous variable specification of an I/O variable $w_{c\,i}$. A set of equations $\hat{F}$ is over-constrained if and only if there exists a subset $\bar{F} \subset \hat{F}$ that is complete. We calculate the set $\bar{F}$ with the bipartite-graph matching based causal analysis algorithm outlined in (Nayak 1995) and determine, whether $\bar{F}$ is a true subset of $\hat{F}$ based on the cardinality of the sets:

$$|\bar{F}| < |\hat{F}|. \tag{7}$$

Whenever this condition is satisfied, we declare $\{w_{c\,1}, \dots, w_{c\,j}\}$ to be a *conflict*. This analysis provides conflicts that are not necessarily minimal, i.e. also subsets of $\{w_{c\,1}, \dots, w_{c\,j}\}$ could imply an over-constrained set of equation $\hat{F}$. As a consequence, our causal analysis builds upon the modified Reiter algorithm (Greiner, B., & Wilkerson 1989).

Summing up, we reformulate exhaustive causal analysis for a hybrid automaton $\mathcal{A}$ at the mode $\mathbf{m}_k$ as a diagnosis-like problem. In analogy to a diagnosis problem ($SD, OBS, COMPS$) we write

$$SD = F(\mathbf{m}_k), \quad COMPS = \mathbf{w}_c, \quad OBS = \emptyset. \tag{8}$$

Note, that within this reformulation, SD is not a set of first-order sentences anymore and a theorem prover (TP) call checks input hypotheses $\{w_{c,1},\dots,w_{c,j}\}$. It does so by computing the extended set of equations (6), performing the causal analysis to deduce $\bar{F}$, and finally, returning either OK or signaling a conflict.

Let us recall the set of equations (2) to illustrate this operation. The Reiter algorithm checks first whether the set of all I/O variables $\mathbf{w}_c = \{w_{c\,1}, w_{c\,2}, w_{c\,3}\}$ is a possible input configuration for the automaton $\mathcal{A}_1$ at mode m_{11}. TP identifies this extended set to be over-constrained and declares $\{w_{c\,1}, w_{c\,2}, w_{c\,3}\}$ as a conflict. The diagnosis algorithm then tests the 3 subsets of the first conflict and obtains the following TP result:

TP call	input set	TP result
2	$\{w_{c\,2}, w_{c\,3}\}$	OK
3	$\{w_{c\,1}, w_{c\,3}\}$	OK
4	$\{w_{c\,1}, w_{c\,2}\}$	conflict

(9)

Only the fourth TP call returns an additional conflict $\{w_{c\,1}, w_{c\,2}\}$. Again, the algorithm investigates input sets that are subsets of $\{w_{c\,1}, w_{c\,2}\}$. None of these evaluations requires an additional causal analysis (TP call), since $\{w_{c\,1}\}$ and $\{w_{c\,2}\}$ represent subsets of the input sets for the TP calls 2 and 3, respectively. No other conflict needs further investigation and the exploration terminates. An intermediate result of the diagnosis algorithm is the so-called *hitting-set directed acyclic graph (HS-DAG)* that is shown in Figure 4 which, in this case, is a directed tree. Building this graph is

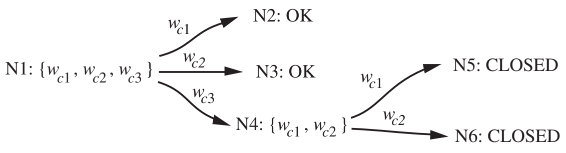

Figure 4: HS-DAG for $\mathcal{A}_1$ at mode m_1.

a systematic way to compute all conflicts with a minimum number of TP calls. The graph encodes the *minimal hitting sets of all conflicts* as paths (arc labels) from the root node (N1) to nodes that are labeled with OK (N2, N3). These minimal hitting sets represent the diagnoses $\Delta_1,\dots,\Delta_\eta$ sought for (Reiter 1987). In our example (Figure 4) we obtain the two 'diagnoses':

$$\Delta_1 = \{w_{c\,1}\},\ \ \Delta_2 = \{w_{c\,2}\}. \tag{10}$$

These diagnoses correspond to minimal sets of *output variables* $\mathcal{Y}_i$ for possible input/output configurations at mode m_1. By taking the set difference $\mathcal{U}_i = \mathbf{w}_c \backslash \Delta_i$ we obtain the corresponding sets of *input variables*

$$\mathcal{U}_1 = \{w_{c\,2}, w_{c\,3}\},\ \ \mathcal{U}_2 = \{w_{c\,1}, w_{c\,3}\}. \tag{11}$$

Consecutive analysis of the remaining mode m_{12} of $\mathcal{A}_1$ (details omitted) provides the following set of I/O causalities for the automaton $\mathcal{A}_1$:

$$\begin{aligned} C_{11} &= \langle\{w_{c\,2}, w_{c\,3}\}, \{w_{c\,1}\}, \{m_{11}\}\rangle, & (12)\\ C_{12} &= \langle\{w_{c\,1}, w_{c\,3}\}, \{w_{c\,2}\}, \{m_{11}, m_{12}\}\rangle. & (13)\end{aligned}$$

Automata Compatibility Analysis

Once we calculated the I/O causalities for two automata $\mathcal{A}_1$ and $\mathcal{A}_2$ we can proceed to determine, whether the automata are compatible or not. Recall that automata incompatibility is due to the fact that two automata determine the value of a shared I/O variable $w_{c\,i}$, i.e. both of them specify $w_{c\,i}$ as dependent (output) variable. Such a situation can be easily checked on the basis of the previously calculated automata I/O causalities.

Let $\mathcal{C}_1$ and $\mathcal{C}_2$ denote the I/O causalities of the automaton $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively. Then the automata at the causalities

$$C_{1i} = \langle\mathcal{U}_{1i}, \mathcal{Y}_{1i}, \mathcal{M}_{1i}\rangle \in \mathcal{C}_1, \quad C_{2j} = \langle\mathcal{U}_{2j}, \mathcal{Y}_{2j}, \mathcal{M}_{2j}\rangle \in \mathcal{C}_2 \tag{14}$$

are compatible, whenever the causalities do not share output variables. This condition is easily checked as:

$$\mathcal{Y}_{1i} \cap \mathcal{Y}_{2j} = \emptyset. \tag{15}$$

Whenever this condition holds for at least one combination of I/O causalities C_{1i} and C_{2j}, then the two automata are compatible. As a result, the composition $\mathcal{A}_1 \parallel \mathcal{A}_2$ has an I/O causality $\langle\mathcal{U}, \mathcal{Y}, \mathcal{M}\rangle$ with the entities:

$$\begin{aligned} \mathcal{U} &= \{\mathcal{U}_{1i} \cup \mathcal{U}_{2j}\} \backslash \{\mathcal{Y}_{1i} \cup \mathcal{Y}_{2j}\},\\ \mathcal{Y} &= \mathcal{Y}_{1i} \cup \mathcal{Y}_{2j},\\ \mathcal{M} &= \mathcal{M}_{1i} \times \mathcal{M}_{2j}. \end{aligned} \tag{16}$$

Checking all possible combinations of $C_{1i} \in \mathcal{C}_1$ and $C_{2j} \in \mathcal{C}_2$ and determining a combined I/O causality (16) for those C_{1i}, C_{2j} that satisfy (15) leads to the set of I/O causalities

$$\mathcal{C} = \{\langle\mathcal{U}_1, \mathcal{Y}_1, \mathcal{M}_1\rangle, \dots, \langle\mathcal{U}_\eta, \mathcal{Y}_\eta, \mathcal{M}_\eta\rangle\} \tag{17}$$

for the composite automata $\mathcal{A} = \mathcal{A}_1 \parallel \mathcal{A}_2$. This set can be used to classify compatibility as:

1. *incompatible* if $\mathcal{C} = \emptyset$,
2. *strongly compatible* if there exists for every mode $\mathbf{m}_j \in \mathcal{X}_d$ of the composite automata at least one I/O causality $C_i = \langle\mathcal{U}_i, \mathcal{Y}_i, \mathcal{M}_i\rangle$ such that $\mathbf{m}_j \in \mathcal{M}_i$, or
3. *weakly compatible* otherwise.

Causal Analysis of cHAs

As noted before, we intend to use the calculation of I/O causalities to determine compatibility of a network of hybrid automata that constitutes a concurrent composition of component automata and the interconnection to the outside world as shown in Figure 5. The interconnection to the outside world can be captured by introducing a new 'world automaton' $\mathcal{A}_0$ with dedicated I/O causality

$$C_0 = \langle\{y_{c\,1},\dots,y_{c\,n_y}\}, \{u_{c\,1},\dots,u_{c\,n_u}\}, \{m_0\}\rangle \tag{18}$$

that is wrapped round the component automata as shown in Figure 6 (n_y and n_u specifies the cHA's number of output and input variables, respectively). With $y_{c\,i}$ and $u_{c\,i}$ we denote a continuous I/O variable that always acts as an output or input of the cHA, respectively. Thus, analyzing a concurrent hybrid automata can be reformulated in terms of

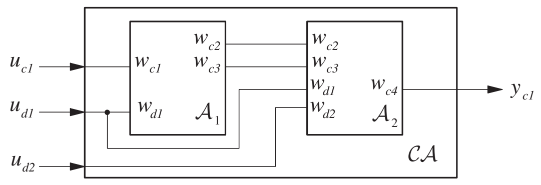

Figure 5: Concurrent hybrid automata.

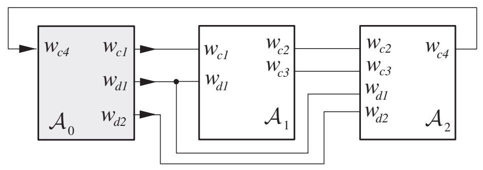

Figure 6: Concurrent hybrid automata as automata network.

compatibility analysis of a network of component automata $\mathcal{A}_1, \ldots, \mathcal{A}_l$ and the 'world automaton' $\mathcal{A}_0$.

We can perform this analysis recursively, starting with the world automaton $\mathcal{A}_0$ that has a unique I/O causality $\mathcal{C}_0 = \{C_0\}$ and one of its adjacent automata, e.g. $\mathcal{A}_1$. We then use the resulting composite automata $\mathcal{A}_0 \parallel \mathcal{A}_1$ and combine one of the next adjacent automata, and so forth. This operation computes the I/O causalities of the composite automaton according to (16) and can be seen as performing a breadth-first search over the possible causality combinations

$$\mathcal{C}_0 \times \mathcal{C}_1 \times \ldots \times \mathcal{C}_l.$$

Let us apply this procedure to the cHA as shown in Figure 5 with the previously introduced automata $\mathcal{A}_1$ and $\mathcal{A}_2$. For the automaton $\mathcal{A}_2$ we assume the set $\mathcal{C}_2 = \{C_{21}, C_{22}\}$ of I/O causalities with entities

$$C_{21} = \langle\{w_{c\,2}\}, \{w_{c\,3}, w_{c\,4}\}, \{m_{21}\}\rangle, \quad (19)$$
$$C_{22} = \langle\{w_{c\,3}\}, \{w_{c\,2}, w_{c\,4}\}, \{m_{21}, m_{22}\}\rangle, \quad (20)$$

and the world automaton $\mathcal{A}_0$ has the I/O causality

$$C_0 = \langle\{w_{c\,4}\}, \{w_{c\,1}\}, \{\}\rangle. \quad (21)$$

Figure 7 shows the breadth-first search tree for the causality

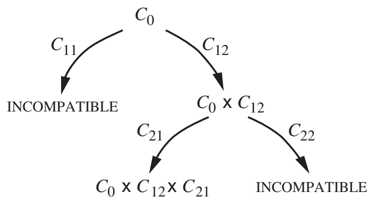

Figure 7: Breadth-first search tree for the cHA causality analysis.

analysis. The composition of $\mathcal{A}_0 \parallel \mathcal{A}_1$ declares $C_0 \times C_{11}$ incompatible, since both specify $w_{c\,1}$ as output variable and provides the causality

$$C_0 \times C_{12} = \langle\{w_{c\,4}, w_{c\,3}\}, \{w_{c\,1}, w_{c\,2}\}, \{m_{11}, m_{12}\}\rangle \quad (22)$$

as an intermediate result. The second expansion of the tree, which is equivalent to checking the composition $(\mathcal{A}_0 \parallel \mathcal{A}_1) \parallel \mathcal{A}_2$ leads to a unique overall I/O causality $C = C_0 \times C_{12} \times C_{21}$, more specifically:

$$C = \langle\{\}, \{w_{c\,1}, w_{c\,2}, w_{c\,3}, w_{c\,4}\}, \{m_{11}, m_{12}\} \times \{m_{21}\}\rangle. \quad (23)$$

This result specifies that the cHA specification shown in Figure 5 is *weakly compatible* since the scope of the I/O causality (23) does not cover a mode combination with $\mathcal{A}_2$ at the mode m_{22}.

Of course, efficiency of this exhaustive computation of the cHA causalities depends on the automata ordering for the recursive composition. The ordering has direct implications on the width of the search tree. This fact is best illustrated with a moderately complex cHA example composed of 8 HA components as shown in Figure 8. Without

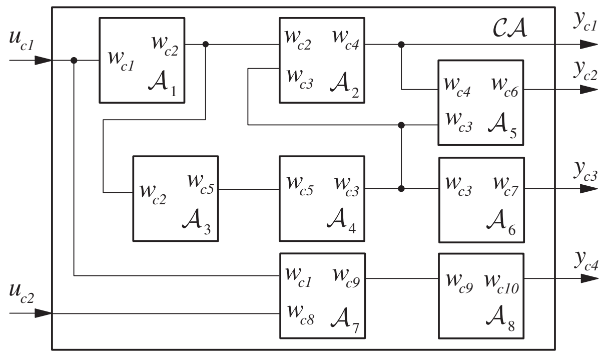

Figure 8: 8 component concurrent hybrid automata.

providing the low level detail for this example we note, that each HA has 3 modes of operation, thus, the cHA has a total number of $3^8 = 6561$ modes. In terms of I/O causalities we obtain through the application of the diagnosis based causal analysis method the causalities shown in Table 1. As a consequence, the overall cHA has at most $2{\cdot}3{\cdot}1{\cdot}2{\cdot}2{\cdot}2{\cdot}3{\cdot}2 = 288$ causalities.

Computing the system-wide valid causalities is done as follows: In a first step, we partition the composition into *independent concurrent HA clusters* and solve the causality computation for each cluster independently. For example, the cHA given in Figure 8 can be separated into a cluster with components $\mathcal{A}_1 || \mathcal{A}_2 || \ldots || \mathcal{A}_6$ and one with $\mathcal{A}_7 || \mathcal{A}_8$, so that:

$$\mathcal{C} = (\mathcal{C}_0 \times \mathcal{C}_1 \times \mathcal{C}_2 \times \mathcal{C}_3 \times \mathcal{C}_4 \times \mathcal{C}_5 \times \mathcal{C}_6) \times (\mathcal{C}_0 \times \mathcal{C}_7 \times \mathcal{C}_8)\ .$$

In the second step, we compute the valid causalities for the HA clusters using the following heuristic automata ordering: Search starts with the 'world automaton' $\mathcal{A}_0$ since no other automaton has fewer causalities. We then recursively select an automaton that is (a) connected to the composite automaton calculated thus far and (b) that has the fewest I/O causalities. Whenever this selection is not unique (because several connected automata share the same number of I/O causalities), we select the automaton that has the largest number of interconnecting variables.

Table 1: Component Causalities

HA	I/O causalities
$\mathcal{A}_1$	$C_{11} = \langle\{w_{c1}\}, \{w_{c2}\}, \{m_{11}, m_{12}, m_{13}\}\rangle$
	$C_{12} = \langle\{w_{c2}\}, \{w_{c1}\}, \{m_{12}, m_{13}\}\rangle$
$\mathcal{A}_2$	$C_{21} = \langle\{w_{c2}, w_{c4}\}, \{w_{c3}\}, \{m_{23}\}\rangle$
	$C_{22} = \langle\{w_{c3}, w_{c4}\}, \{w_{c2}\}, \{m_{22}\}\rangle$
	$C_{23} = \langle\{w_{c2}, w_{c3}\}, \{w_{c4}\}, \{m_{21}, m_{22}, m_{23}\}\rangle$
$\mathcal{A}_3$	$C_{31} = \langle\{w_{c2}\}, \{w_{c5}\}, \{m_{31}, m_{32}, m_{33}\}\rangle$
$\mathcal{A}_4$	$C_{41} = \langle\{w_{c5}\}, \{w_{c3}\}, \{m_{41}, m_{42}, m_{43}\}\rangle$
	$C_{42} = \langle\{w_{c3}\}, \{w_{c5}\}, \{m_{42}, m_{43}\}\rangle$
$\mathcal{A}_5$	$C_{51} = \langle\{w_{c4}, w_{c6}\}, \{w_{c3}\}, \{m_{53}\}\rangle$
	$C_{52} = \langle\{w_{c3}, w_{c4}\}, \{w_{c6}\}, \{m_{51}, m_{52}, m_{53}\}\rangle$
$\mathcal{A}_6$	$C_{61} = \langle\{w_{c7}, w_{c3}\}, \{w_{c8}\}, \{m_{61}, m_{62}\}\rangle$
	$C_{62} = \langle\{w_{c3}, w_{c7}\}, \{w_{c7}\}, \{m_{61}, m_{62}, m_{63}\}\rangle$
$\mathcal{A}_7$	$C_{71} = \langle\{w_{c1}, w_{c8}\}, \{w_{c9}\}, \{m_{71}, m_{72}, m_{73}\}\rangle$
	$C_{72} = \langle\{w_{c8}, w_{c9}\}, \{w_{c1}\}, \{m_{72}\}\rangle$
	$C_{73} = \langle\{w_{c1}, w_{c9}\}, \{w_{c8}\}, \{m_{73}\}\rangle$
$\mathcal{A}_8$	$C_{81} = \langle\{w_{c10}\}, \{w_{c9}\}, \{m_{82}, m_{83}\}\rangle$
	$C_{82} = \langle\{w_{c9}\}, \{w_{c10}\}, \{m_{81}, m_{82}, m_{83}\}\rangle$

In our example we thus perform the composition of the $\mathcal{A}_0, \ldots, \mathcal{A}_6$ component cluster as

$$(((((\mathcal{A}_0 || \mathcal{A}_1) || \mathcal{A}_3) || \mathcal{A}_5) || \mathcal{A}_6) || \mathcal{A}_4) || \mathcal{A}_2$$

and obtain a single overall causality

$$\mathcal{C} = \{C_0 \times C_{11} \times C_{23} \times C_{31} \times C_{41} \times C_{52} \times C_{62} \times C_{71} \times C_{82}\} \tag{24}$$

in a breadth-first search with maximal tree width of 6 that only explores 12 additional incompatible causalities.

Conclusion and Future Work

The clear separation between the computation of I/O causalities for the individual HA components and the consecutive recursive composition of the causalities is particularly helpful within an *object oriented* cHA modeling framework (Hofbaur 2005). For example, the micro-chemical plant of Figure 1 has 20 valves, all of them are *instances* of a particular *HA class* `valve`. Thus, in order to analyze the overall model we have to compute the I/O causalities for the HA class `valve` only once and can utilize this result for all 20 instances.

Alternatively to the breadth-first search approach to combine the individual I/O causalities we are currently investigating a reformulation of this task as a CSP problem. The latter would enable us to utilize standard CSP solvers and utilize structural properties of our concurrent automata through tree decomposition methods for CSPs (Gottlob, Leone, & Scarcello 2000), for example.

Regardless of which method, heuristic ordering or CSP / tree decomposition turns out to be superior for typical system topologies, we laid out the basis for an efficient algorithmic formulation of an exhaustive causal analysis of concurrent hybrid automata. Key to this methodology was the reformulation of the causal analysis problem as a diagnosis problem for which model based diagnosis provides efficient algorithms. Thus, we were able to show that the applicability of model-based diagnosis methods goes beyond the scope of pure diagnosis.

References

Alur, R., and Henzinger, T. 1999. Reactive modules. *Formal Methods in System Design* 15(1):7–48.

Chutinan, A., and Krogh, B. 1999. Verification of hybrid systems using polygonal flowpipe approximations. In *Hybrid Systems (HSCC 1999)*. 76–90.

de Kleer, J., and Brown, J. 1986. Theories of causal ordering. *Artificial Intelligence* 29:33–61.

de Kleer, J., and Williams, B. C. 1987. Diagnosing multiple faults. *Artificial Intelligence* 32(1):97–130.

Flaus, J.-M., and Gentil, S. 2003. Hybrid system automatic modeling for diagnostic purposes. In *Preprints of the 5th IFAC SAFEPROCESS Symposium*, 675–680.

Gottlob, G.; Leone, N.; and Scarcello, F. 2000. A comparison of structured CSP decomposition methods. *Artificial Intelligence* 124:243–282.

Greiner, R.; B., S.; and Wilkerson, R. 1989. A correction to the algorithm in Reiter's theory of diagnosis. *Artificial Intelligence* 41:79–88.

Hamscher, W.; Console, L.; and de Kleer, J., eds. 1992. *Readings in Model-Based Diagnosis*. Morgan Kaufmann.

Henzinger, T., and Ho, P. 1996. Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering* 22:181–201.

Hofbaur, M. W., and Williams, B. C. 2002. Mode estimation of probabilistic hybrid systems. In *Hybrid Systems (HSCC 2002)*. 253–266.

Hofbaur, M. W., and Williams, B. C. 2004. Hybrid estimation of complex systems. *IEEE Transactions on Systems, Man, and Cybernetics - Part B* 34(5):2178–2191.

Hofbaur, M. W. 2005. *Hybrid Estimation of Complex Systems*, volume 319 of *Lecture Notes in Control and Information Sciences*. Springer.

Iwasaki, Y., and Simon, H. 1986. Causality in device behavior. *Artificial Intelligence* 29:3–32.

Lynch, N.; Segala, R.; and Vaandrager, F. 2001. Hybrid I/O automata revisited. In *Hybrid Systems (HSCC 2001)*, volume 2034. 403–417.

Mosterman, P.; Biswas, G.; and Sztipanovits, J. 1998. A hybrid modeling and verification paradigm for embedded control systems. *Control Eng. Practice* 6(4):511–521.

Nayak, P. 1995. *Automated Modelling of Physical Systems*. Lecture Notes in Artificial Intelligence. Springer.

Reiter, R. 1987. A theory of diagnosis from first principles. *Artificial Intelligence* 32:57–95.

Travé-Massuyès, L., and Pons, R. 1997. Causal ordering for multiple mode systems. In *Proc. of the 11th Internat. Workshop on Qualitative Reasoning (QR97)*, 203–214.