State Tracking of Uncertain Hybrid Concurrent Systems[*]

Emmanuel Benazera, Louise Travé-Massuyès
LAAS-CNRS, 7 Avenue du Colonel Roche
31077 Toulouse cedex 4
Tl. + 33 5 61 33 63 02
Fax. + 33 5 61 33 69 36
ebenazer, louise@laas.fr

Philippe Dague
LIPN - UPRESA 7030
Université Paris 13
99 Av. J-B. Clément
93430 Villetaneuse France
dague@lipn.univ-paris13.fr

Abstract

In this paper we propose a dedicated component-based hybrid formalism, able to represent both continuous and discrete physical phenomena. The formalism merges concurrent automata with continuous uncertain dynamic models. Our framework is rather generic but focuses on the construction of intelligent autonomous supervisors by integrating a continuous/discrete interface able to reason on-line in any region of the physical system state-space, for behavior simulation, diagnosis and system tracking.

1 INTRODUCTION

In the past few years, numerous works have been presented to model embedded systems with hybrid models and reason about them for simulation, diagnosis [McIlraith *et al.*, 1999] or verification [Alur *et al.*, 1995] purposes. The modeling framework usually expresses the different operating modes of the system as a set of finite automata and associates to each mode continuous knowledge encoded through standard numeric differential equations. In this paper we propose a dedicated component-based hybrid formalism, able to represent and integrate continuous and discrete physical phenomena. The formalism merges concurrent automata with continuous bounded uncertain models that are preferred over stochastic models for robustness and guaranteed results reasons. However it is not sufficient to add continuous knowledge to automata, because moving between operating modes requires the automatic construction of the structure of the newly assembled continuous model. It means computing both the characterization of the region of the state-space of the operating mode (denoted as a *configuration*), and a proper causal ordering between the active variables in that mode. No prestudy of the behavior of the physical system is required to determine the state-space regions associated with the current system configuration(s) because the search at continuous level is casted into a boolean constraint satisfaction problem. A reasoning continuous/discrete interface (C/D I) is thus added, which provides an on-line generation of the characterization of the new model structure by making use of enhanced Truth Maintenance techniques [Williams and Nayak, 1997a] on the logical model. This is keypoint to our computing of the diagnosis of hybrid model where detection is provided by the continuous layer and state identification is performed at the discrete logical level by searching for the current configuration consistent with observations. At the same time, the logical framework allows the description of purely discrete component behavior in the same manner as in [Williams and Nayak, 1996]. In section 2 are described the discrete and the continuous layers; section 3 presents the interface that integrates both layers together; in section 4 are presented the algorithms required to reason about hybrid models and to track multiple trajectories in both simulation and diagnosis.

[*]This work is supported by CNES (French Space Research Center) and ASTRIUM.

2 Hybrid System Formulation

2.1 Hybrid Systems as Transition Systems

The set of all components of the physical system to be modeled is denoted by $Comps$. Every component in that set is described by a hybrid transition system. The set of all variables used to describe a component is denoted V and is partitioned in the following manner:

- $\Pi = \Pi_M \cup \Pi_C \cup \Pi_{Cond} \cup \Pi_D$ — set of discrete variables of 4 distinct types (Mode, Command, Conditional, Dependent),
- $\Xi = \Xi_I \cup \Xi_D$ — set of continuous variables of 2 distinct types (Input, Dependent).

Mode variables Π_M represent components nominal or faulty modes, such as *on* or *stuck off*. Command variables Π_C are endogeneous and exogeneous commands modeled as discrete events to the system (e.g. software commands). Continuous input variables Ξ_I are exogeneous continuous signals to the system determined by its environment (e.g. known inputs or disturbances). Conditional variables Π_{Cond} are specific discrete variables that represent conditions on continuous variables. Discrete and continuous dependent variables are all other variables. Finally the set Obs contains observable variables of the physical system. Each observable signal has an explicit sampling period. Our hybrid transition system is an extension of the standard transition system [Manna and Pnueli, 1992].

Definition 1 (Hybrid Transition System – HTS) *A Hybrid Transition System HTS is a tuple (V, Σ, T, C, Θ) with:*

- *$V = \Pi \cup \Xi$ set of all variables.$\forall v \in V$, the domain of v is $D[v]$, finite for variables in Π, $\mathcal{R}$ otherwise.*
- *Σ — set of all interpretations over V.*
  Each state in Σ assigns a value from its domain to any variable $v \in V$.
- *T — finite set of transition variables.*
  Each variable τ_m in T ranges over its domain $D[\tau_m]$ of possible transitions of the mode variable $m \in \Pi_M$. Each τ_m^i in $D[\tau_m]$ is a function $\tau_m^i : \Sigma \rightarrow 2^{\Sigma}$.
- *C — set of (qualitative or quantitative) continuous constraints over V.*
  Each constraint c in C at least depends on one mode variable in Π_M. $\forall m \in \Pi_M$, we note $C[m]$ the set of constraints associated to the variable m.
- *Θ — set of initial conditions.*
  Θ is a set of assertions over V such that they define the set of initial possible states, i.e. the set of states s in Σ such that $s \models \Theta$.

Note that in a HTS, due to the continuous constraints in C, some transitions can be triggered according to conditions over continuous variables. At the discrete/continuous interface level, these conditions have a corresponding discrete variable in Π_{Cond}, which captures their truth value.

States and Time

Considerations about time are central because both the discrete and the continuous frameworks use time representations that are different. At the continuous level, time is explicitely represented in the equations that represent the physical system behavior, we call it *physical time θ*. Physical time is discretized according to the highest frequency sensor, providing the HTS reference sampling period T_s. $x(kT_s)$, or $x(k)$ for short, specifies the value of the continuous state-variables in Ξ. We call *abstract time* the time at the discrete level. It is dated according to the occurrence of discrete events. At date t, the discrete state π_t of a HTS is the tuple (M_t, Q_t), where M_t is the vector of instances of mode variables, and Q_t the vector of instances of variables of Π in qualitative constraints. Discrete state-variables are in $\Pi \setminus \Pi_{Cond}$. Abstract time dates are indexed on physical time, which informs about how long a component has been in a given discrete state. If $t = kT_s$, then we write the indexed date t^k. When there is no ambiguity it is simply denoted by t. The *hybrid state s_{t^k}* of a HTS is the tuple $(\pi_{t^k}, x(k))$.

Transitions

Transitions describe changes between mode values over the time. $\forall m \in \Pi_M$, $\exists \tau_m \in T$, such that $D[\tau_m] = \{\tau_m^i \in T_N\} \cup \{\tau_m^j \in T_F\} \cup \{\tau^{id}\}$, with:

- T_N the set of *nominal* transitions,
- T_F the set of *faulty* transitions,
- τ^{id} the *id* transition.

Nominal transitions express switches from one nominal mode to another, whereas fault transitions move the system into a faulty mode. Because transitions cannot be considered as instantaneous against the frequency of the sensors, we introduce delays on nominal transitions. Delay $d_{\tau_m^i}$ is such that once a transition τ_m^i is *enabled* it is triggered after $d_{\tau_m^i} T_s$, i.e. after $d_{\tau_m^i}$ physical time units. While a transition is *enabled* and waiting for its delay to expire, it is said to be in *standby* state. A delay on transition can also be modeled by adding modes and clocks to the hybrid transition system [Henzinger, 1996]. We do not use this representation here because we think that it does not enforce the easy representation of a component as a transition system by creating modes that are irrelevant for the diagnosis purpose. Abrupt fault transitions have no explicit delay, i.e. their duration is one physical time unit. For a matter of simplification, the delay will be referred as d when there is no ambiguity.

Definition 2 (pre and post assertions) *For a given transition τ_m^i and a given state $s_{t^k} \in \Sigma$, we note assertions $pre(\tau_m^i) = m^j \wedge \phi^i_{\Pi_{C \cup Cond}}(s_{t^k})$ and $post(\tau_m^i) = m^{j'}$ where:*

- *m^j and $m^{j'}$ are two instances of the mode variable m,*
- *$\phi^i_{\Pi_{C \cup Cond}}(s_{t^k})$ is a logical condition over instances of variables of both Π_C and Π_{Cond} in s_{t^k}.*

We refer to the *guard* of a transition as the condition statement $\phi^i_{\Pi_{C \cup Cond}}$ that triggers the transition. Only fault transitions can be spontaneous, so their guard can be always true. Traditionnally, probabilities are also attached to every nominal and faulty transitions.

2.2 Component modes behavior

The behavior of a component is encoded as a set of nominal and faulty modes that exhibit a discrete or a continuous behavior depending on the component type. For discrete components, the behavioral model is given by a set of constraints over $\Pi_C \cup \Pi_D$ associated to each mode variable. Such equipments are usually software drivers as well as complex electronic devices. For continuous components, the continuous behavior is expressed by discrete-time continuous constraints over Ξ. Each constraint is attached to a mode of the transition system. Note that the *unknown* mode is rather specific as it has no constraints and thus covers all interpretations in Σ. The discrete-time continuous constraints are of the following form:

$$\left\{ \begin{array}{lcl} x(k+1) & = & Ax(k) + \sum_{j=0,\ldots,r} B_i u(k-j) \\ y(k+1) & = & Cx(k+1) \end{array} \right. \quad (1)$$

where $x(k)$, $y(k)$, and $u(k)$ represent the state vector of dimension n, observed variables vector of dimension p and control variables of dimension q at time kT_s, respectively; A, B_i and C are matrices of appropriate dimensions. Continuous constraints are encoded in a specific two levels formalism [Travé-Massuyès and Milne, 1997] which includes a causal model and an analytical constraint level. The causal model is obtained from equation (1) by expressing it as a set of *causal influences* among the continuous variables. The underlying operational model of dynamic influences is provided by the following equation:

$$\xi_j(k+1) = \sum_{p=0,\ldots,n-1} a_p \xi_j(k-p) + \sum_{q=0,\ldots,m} b_q \xi_i(k+1-q) \quad (2)$$

where ξ_i and ξ_j are continuous variables and n is the influence order and $m \leq n$ (causal link). Usually an equation is modeled by a set of influences of many types: dynamic, integral, static and constant. Dynamic influences are characterized with a gain K and a delay T_d, as well as with the response time T_r for a first order relation, the damping ratio ζ and the undamped natural frequency ω of the system for a second order relation. Other types also obey equation (2) with different parameters a_p and b_q. Every influence is conditioned by a logical assertion over variables in Π_{Cond}. An influence is said to be *active* when its condition is true. When necessary, uncertainties can be taken into account in the influence parameters and as additive disturbances. The first are represented by considering that parameters a_p and b_q have time independent bounded values, i.e. they are given an interval value. The latter can be introduced as a bounded value constant influence acting on ξ_j. From the superposition theorem that applies to the linear case, the computation of the updated value of variable $\xi_j \in \Xi$ in an equation eq consists of processing the sum of the activated influences from eq having exerted on ξ_j during the last time-interval. The prediction update of all the state and observed variables $x(k)$ and $y(k)$ from the knowledge of control variables $u(k)$ and influence activation conditions is performed along the causal model structure. Our representation of uncertainties leads to the prediction of continuous variable trajectories in the form of bounded envelopes. In other words, the system state $x(k)$ at every time instant $t = kT_s$ is provided in the form of a rectangle of dimension n.

Definition 3 (Causal system description – CD) *The causal system description associated to the set of continuous constraints of a HTS is a directed graph $G = (\Xi, I)$ where I is a set of edges supporting the influences among variables in Ξ, with their associated conditions and delays.*

The numerical intervals obtained from equation (2) are refined at the analytical model level with global constraints by performing a tolerance propagation algorithm [Hyvonen, 1992] on the set of variables.

2.3 Moving between modes

When a transition triggers, the component switches from one mode to another, the corresponding HTS needs to transfer its continuous state vector x as well. For that reason each transition τ^i_m is associated with a *mapping function* $l_{\tau^i_m} : \Sigma \rightarrow \Sigma$ over the dependent variables in V. It initializes the value of a subset of variables in the hybrid state resulting from applying τ^i_m to $s_{t^k_l}$ where l is the abstract time index. Other variables in $s_{t^k_l}$ keep their previous value. The identity mapping function is denoted l^{id}. Triggering a transition is a two steps operation [Mosterman and Biswas, 2000; Alur *et al.*, 1995]. First, mode change is performed by applying the transition τ^i_m to the current hybrid state and moving to the resulting mode after its delay has expired (*transition relation* $\overset{\tau^i_m}{\rightarrow}$).

$$\frac{\tau^i_m \in T,\, (s_{t^k_l}, s_{t^{k+d}_{l+1}}) \in \Sigma^2,\, s_{t^k_l} \models pre(\tau^i_m)}{s_{t^k_l} \overset{\tau^i_m}{\rightarrow} s_{t^{k+d}_{l+1}}} \qquad (3)$$

Second, initialization is performed by making use of the mapping function, and physical time goes on (*time-step relation* $\overset{\theta}{\rightarrow}$):

$$\frac{(\pi_{t_{l+1}}, x(k+d)) = l_{\tau^i_m}(s_{t^k_l})}{(\pi_{t_{l+1}}, x(k+d)) \overset{\theta}{\rightarrow} (\pi_{t_{l+1}}, x(\theta))} \qquad (4)$$

where $x(\theta)$ is the continuous state associated to discrete state π_{t_l} over the continuous time θ.

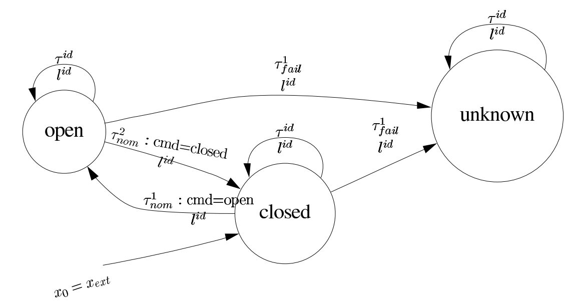
2.4 Hybrid Component System

Once components have been modeled as HTS, they need to be aggregated in a *Hybrid Component System* to model the entire physical plant. Components are instanciated and their models form reusable databases. Within the whole plant model, components are concurrent, i.e. able to evolve independently.

Definition 4 (Hybrid Component System – HCS) *A Hybrid Component System HCS is a tuple $(Comps, V, \Sigma, T, C, \Theta)$ with $Comps$ being a set of n components modeled as concurrent hybrid transition systems $H_i = (V_i, \Sigma_i, T_i, C_i, \Theta_i)$, $\left(\bigcup_{i=1,\cdots,n} V_i\right) = V$, $\Sigma \subseteq \bigotimes_i \Sigma_i$, $T = \bigcup_i T_i$, $C = \bigcup_i C_i$, $\Theta = \bigcup_i \Theta_i$.*

The n hybrid transition systems of a HCS are concurrent. Constraints and commands synchronize on shared variables in Π_D, Π_C and Ξ, then the automata communicate through shared variables and synchronize on transitions. The identity transition synchronizes stationary components when it is necessary for their constraints to be shared. The process is largely complexified by the introduction of delays on transitions and cannot be fully described in this paper. Basically it relies on the hypothesis that *we cannot track or diagnose a physical component while it is switching from one mode to another*, i.e. when one of its transitions is in *standby*, as the required transient models are often unknown or too complex. The consequence is that components only synchronize in their non-standby states.

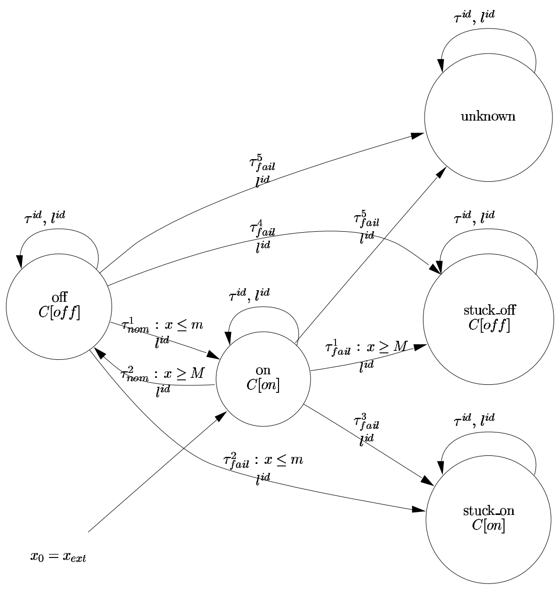
2.5 Example



where:
$C[open] : \dot{x} = aQ_o(\Delta x)$
$C[closed] : \dot{x} = aQ_c(\Delta x)$ where $\Delta x = x_e - x$.

Figure 1: room with unknown mode

Figure 1 shows the HTS of a room R submitted to a temperature source. It has two nominal modes: *open* (a door or a window is opened) and *closed*. The room temperature



where:
$C[off] : x_e = x_{ext}$
$C[on] : x_e = h$

Figure 2: thermostat with fault modes

x is influenced by the temperature of the source x_e according to a first-order differential equation which accounts for the room characteristics Q_c (closed) and Q_o (open). The actions that move the room from one mode to another are modeled as observed single discrete commands $cmd = open$ and $cmd = close$. Figure 2 presents the model of a thermostat T, with faulty and unknown modes, as well as required transitions. This thermostat switches according to the room temperature x (it should move to its *on* mode when the temperature $x \leq m$ to warm up the room, and back to its *off* mode when $x \geq M$ to cool it down). x is hence influenced by the heater setting temp h (on mode *on*) or by the outside temperature x_{ext} (on mode *off*). The temperature variation $\dot{x}$ is observed through a sensor with additive noise $\dot{x}_{noi}$. Initially, $x = x_{ext}$, the room is *closed* and the thermostat is *on*. Variables of the HCS are:

$$\begin{aligned}
R.mode \in \Pi_M &= (closed, open, unknown)\\
R.cmd \in \Pi_C &= (none, open, close)\\
R.c \in \Pi_{cond} &= (R.x \leq m, R.x > m \wedge R.x < M, R.x \geq M)\\
T.mode \in \Pi_M &= (off, on, stuck\_on, stuck\_off, unknown)\\
R.x \in \Xi_D &\in [-\infty, +\infty]\\
R.\dot{x} \in \Xi_D &\in [-\infty, +\infty]\\
R.\Delta x \in \Xi_D &\in [-\infty, +\infty]\\
R.\dot{x}_{noi} \in \Xi_I &\in [-1, 1]\\
R.Qc &\in [0.05, 0.15]\\
R.Qo &\in [0.02, 0.05]\\
R.a &\in [0.9, 1.1]\\
Obs &= \{\dot{x}\}
\end{aligned}$$

The feasible continuous states of Σ are specified by the influences in each HTS:

$$R.i_1\ (static) : \text{if } (R.mode = closed) \text{ then } R.\Delta x \overset{gain=Q_c}{\longrightarrow} R.\dot{x}$$

$$R.i_2\ (static) : \text{if } (R.mode = open) \text{ then } R.\Delta x \overset{gain=Q_o}{\longrightarrow} R.\dot{x}$$

$$R.i_3\ (integral) : R. \overset{gain=a}{\longrightarrow} R.x$$

$$R.i_4\ (static) : R.x \overset{gain=-1}{\longrightarrow} R.\Delta x$$

$$T.i_1\ (constant) : \text{if } (T.mode = on \vee T.mode = stuck\_on) \text{ then } T.h \longrightarrow R.\Delta x$$

$$T.i_2\ (constant) : \text{if } (T.mode = off \vee T.mode = stuck\_off) \text{ then } R.x_{ext} \longrightarrow R.\Delta x$$

$$T.i_3\ (constant) : T.x_{noi} \longrightarrow R.\dot{x}$$

Influences without explicit conditions are valid in all modes except for the *unknown* mode. T is represented as follows ($\bigcirc$ is the next operator from temporal logic):

$$\begin{aligned}
R.\tau^1_{nom} &: R.mode = closed \wedge R.cmd = open \quad \bigcirc \quad R.mode = open\\
R.\tau^2_{nom} &: R.mode = open \wedge R.cmd = closed \quad \bigcirc \quad R.mode = closed\\
R.\tau^1_{fail} &: R.mode = open \vee R.mode = closed \quad \bigcirc \quad R.mode = unknown
\end{aligned}$$

$$\begin{aligned}
T.\tau^1_{nom} &: T.mode = off \wedge R.x \leq m \quad \bigcirc \quad T.mode = on\\
T.\tau^2_{nom} &: T.mode = on \wedge R.x \geq M \quad \bigcirc \quad T.mode = off\\
T.\tau^1_{fail} &: T.mode = on \wedge R.x \geq M \quad \bigcirc \quad T.mode = stuck\_off\\
T.\tau^2_{fail} &: T.mode = off \wedge R.x \leq m \quad \bigcirc \quad T.mode = stuck\_on\\
T.\tau^3_{fail} &: T.mode = on \quad \bigcirc \quad T.mode = stuck\_on\\
T.\tau^4_{fail} &: T.mode = off \quad \bigcirc \quad T.mode = stuck\_off\\
T.\tau^5_{fail} &: T.mode = on \vee T.mode = off \quad \bigcirc \quad T.mode = unknown
\end{aligned}$$

Figure 3 presents the nominal CD of the HCS resulting from the composition of the two components.

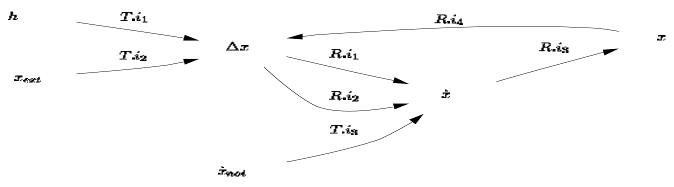


Figure 3: Causal nominal system description of the thermostat and room example

3 Continuous/Discrete Interface

3.1 Configurations

Depending on the mode at a given time, a HCS has its hybrid state that ranges over several continuous regions. These regions are known to be difficult to determine and compute, if not undecidable. We propose an on-line mechanism to keep track of the state-space partition by sheltering every continuous functional piece with a conjunction of logical conditions we denote as a *configuration*.

Definition 5 (HCS configuration) *A configuration for a HCS at time-step t^k is a logical conjunction $\delta_{t^k} = (\bigvee_i m^i) \wedge (\bigwedge_j \Pi^j_{Cond})$ where the m^i are instanciations of component modes in Π_M and the Π^j_{Cond} are variables of Π_{Cond}.*

The configurations are automatically drawn from conditions on both transition guards and influences that define structural changes in the model. A configuration can be attached to one or more modes in Π_M. In our example, the continuous state is easily partitioned by the thermostat's transitions into three regions determined by the three conditions on variable x, defining 18 configurations:

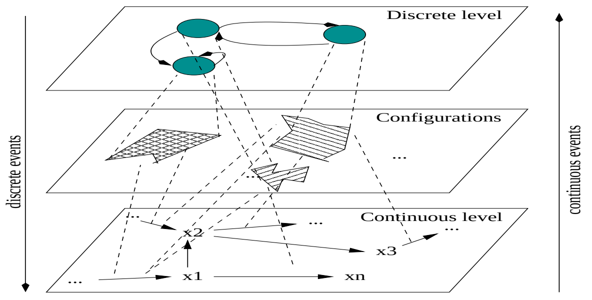


Figure 4: 3-layers interactions

$\mathcal{C}_1 \quad : \quad R.mode = closed \wedge T.mode = on \wedge R.x \leq m$
$\mathcal{C}_2 \quad : \quad R.mode = closed \wedge T.mode = on \wedge (R.x > m \wedge R.x < M)$
$\mathcal{C}_3 \quad : \quad R.mode = closed \wedge T.mode = off \wedge (R.x > m \wedge R.x < M)$
$\mathcal{C}_4 \quad : \quad R.mode = closed \wedge T.mode = off \wedge R.x \geq M$
$\ldots$

Whatever the complexity of the physical system is, it is easy to logically express the conditions as two modalities variables of Π_{Cond}, made of the condition and its negation, but it leads to partitions that are not optimal. However when reasoning on the HCS, configurations are sound because logically unconstrained solutions are ruled out when returning to the continuous values of variables. Note that the configuration associated to the *unknown* mode encompasses the overall state-space.

3.2 Causal ordering for static equations

When switching from one mode to another, some equations and variables are added or retracted according to the new configuration; in consequence, due to the possible presence of static continuous equations in the model, a proper causal ordering of variables is to be found when entering the new mode. A brute force approach would consist in generating a new causal structure for every different mode. The problem of performing an on-line incremental generation of the causal structure has been previously addressed [Travé-Massuyès and Pons, 1997] but it is solved here in a slightly different manner. This is done by first casting the problem into a boolean constraint satisfaction problem: every continuous equation and variable in the HCS is associated to boolean variables in Π whose truth values state if the variables or equations are active or not. Rules over the boolean variables are automatically built to represent the conditions of these activations and form a logical representation of the causal-ordering problem.

3.3 Overview

The previous configuration and causal ordering problems are solved on-line by using a truth maintenance system (TMS) to reason on the corresponding boolean constraint satisfaction problems. The context switching algorithms of [Williams and Nayak, 1997a] are useful because we are not interested in generating all configurations of the physical system but to switch from one to another as fast as possible. The HCS reacts to events, i.e. observations from sensors as well as commands, and propagates them to the model's discrete and continuous levels through the logical interface and the way back. Figure 4 sums up these interactions. The C/D I, made of the variables in Π_{Cond} associated to influence conditions and transition guards, as well as the causal ordering logical model, ensures the logical consistency of the changes triggered by the flow of events.

4 Simulation and Diagnosis of a Hybrid Component System

4.1 Simulation

A HCS simulation is a run of concurrent hybrid transition systems that generates possible nominal trajectories of the HCS according to issued commands and inputs over the time. The uncertainty on the continuous constraint parameters determines the precision of the computed envelopes that enclose the observed behavior of the physical system at each time step.

Sometimes the truth value of a condition in a configuration may be undetermined when checked against a rectangular enclosing of the continuous state-variables. The problem arises from the fact that some variables over which configurations rely are not measured. When the computed bounds of such a continuous variable ξ_i span over more than one configuration region relying on that variable, we say that the current *configuration is splitting the continuous state on variable* ξ_i. Figure

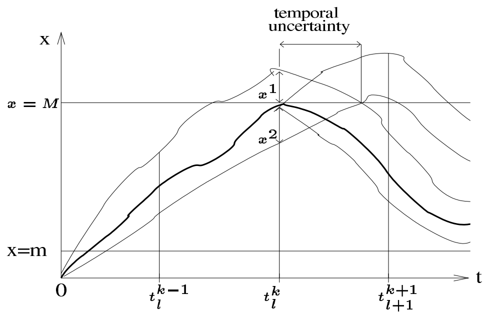


Figure 5: Transition guard split

5 shows a simple example of a configuration split for the thermostat example when crossing oat $x = M$. The current configuration splits on regions x^1 and x^2 and the two possible trajectories are tracked simultaneously. In applications, this situation happens rather frequently and multiple consecutive splits of a guard on the same variable can occur because sensor frequencies are usually beneath the *temporal uncertainty* induced by the envelopes. We first want to split the continuous state into logical branches then refine consequently the bounds on all continuous variables in every explored branch. For a given continuous variable ξ_i, the logical split from a configuration δ_{t^k} returns the set of possible configurations to be tracked:

$$[\delta_{t^k}](\xi_i) = \bigvee_j \left(\Pi^j_{Cond_{\xi_i}} \wedge \left(\bigwedge_n \Pi^n_{Cond} \right) \right) \tag{5}$$

where $\Pi^j_{Cond_{\xi_i}}$ are variables of Π_{Cond} relying on ξ_i and Π^n_{Cond} other conditions in δ_{t^k}. Relation (5) is used to compute the splitted areas because it is much faster than exploring the overall continuous state space. The following algorithm is applied on every tracked trajectory:

1. Search for a continuous variable ξ_i over which the current configuration δ_{t^k} is splitting.
2. Logically split the state-space with relation (5). To each configuration $\delta^j_{t^k}$ in $[\delta_{t^k}](\xi_i)$ we note its corresponding continuous region $x^j_{\xi_i}(k)$ and $\pi^j_{t^k,\xi_i}$ its corresponding discrete state.
3. Envelopes over variables in Ξ are refined in every region $x^j_{\xi_i}(k)$ by filtering them on the constraints defined by the conditions in the configuration [Hyvonen, 1992].
4. $(\pi^j_{t^k,\xi_i}, x^j_{\xi_i}(k))$ constitute new hybrid states enclosed in new trajectories to be tracked.

The three preceding steps are applied for remaining variables on the growing set of generated trajectories. Finally the resulting set of computed hybrid states is:

$$[s_{t^k}] = \bigotimes_{i,j}(\pi^j_{t^k,\xi_i}, x^j_{\xi_i}(k)) \tag{6}$$

In our example, the thermostat's configurations only split on the temperature x. On figure 5, until time-step t^k_l, the configuration of the HCS is

$$\mathcal{C}_2 : R.mode = closed \wedge T.mode = on \wedge R.x > m \wedge R.x < M$$

At time-step t^k_l, due to the crossing of $x = M$, the current configuration is splitted on x. A new partial hybrid state comes from equation (5):

$$R.mode = closed \wedge T.mode = on \wedge R.x \geq M$$

Then bounds of variable x are refined in each configuration by filtering the values with respective constraints $R.x > m \wedge R.x < M$ and $R.x \geq M$. As transition $T.\tau^2_{nom}$ turns *enabled*, the configuration is instantaneously ($T.\tau^2_{nom}$ has no delay) updated to:

$$\mathcal{C}_4 : R.mode = closed \wedge T.mode = off \wedge R.x \geq M \tag{7}$$

From that point the system tracks two distinct trajectories.

4.2 Fault Detection

The detection algorithm then uses the above estimation of the endogeneous continuous variable values in a *semi closed-loop* mode: it runs in *closed-loop* mode, i.e. resetting envelopes with observations, as long as no alarm is detected (a measured variable is out of its predicted bounds); when alarming, it performs pure *open-loop* simulation to avoid following the fault [Travé-Massuyès and Milne, 1997]. Note that the *closed-loop* and *open-loop* modes are identical for non dynamically influenced variables, as well as not relevant for non measured variables. Detection is also used to rule out misleading trajectories previously generated.

Figure 6 shows three scenarios with faults where detection is applied. On the first scenario the thermostat fails to switch at time-step 59 and sticks to its *on* mode. In the second scenario the constant $T.h$ is degraded from time-step 51 to a lower value, so the heater is slower to warm the room. Scenario three presents an abrupt fault characterized by a structural change in the thermostat model. For each scenario the system is tracked in *semi-closed* loop mode until the observation on $\dot{x}$ lies outside the computed bounds. Usually the diagnosis operation is triggered when no more tracked trajectories are consistent with observation, and a few steps after the fault has been detected, in order to avoid false alarms.

4.3 Diagnosis

When a fault is detected, a diagnosis comes back to find the current configuration of the HCS according to observations, inputs and commands. This must be performed over a finite temporal window to avoid loosing solutions [Nayak and Kurien, 2000].

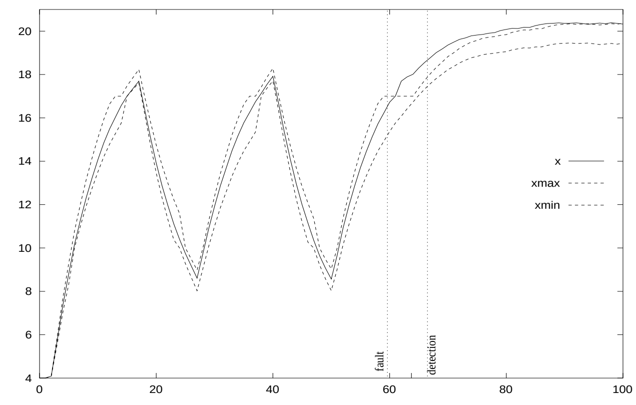
Definition 6 (HCS Diagnosis) *A diagnosis $diag(t)$ over m time-steps for a HCS is such that $diag(t) = \{\delta_t\}_{t=1,\cdots,m}$ with the consitency of:*

$$HCS \cup Obs_{t=1,\ldots,m} \cup \left(\bigcup_{t=1,\cdots,m} \delta_t \right) \tag{8}$$

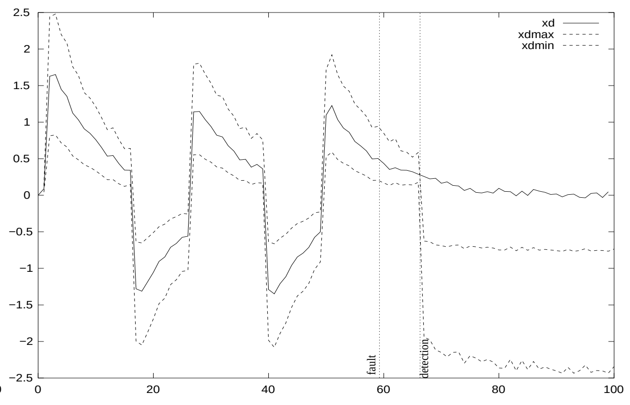
Solving relation (8) is a three steps operation. First, existing conflicts (a set of influences which cannot be unfaulty altogether) are exhibited from the causal system description (CD) of the HCS, each influence stamped with a temporal label and activation condition. They are then turned into diagnosis candidates by a failure-time oriented enhanced version of the hitting set algorithm [Travé-Massuyès and Jimenez, 2001]. Temporal information is drawn from the delays of the influences in CD. Second, at the configurations level, the TMS negates the activation conditions of the conflicting influences and fastly iterates through the logical remaining configurations to reinsure the consistency. Finally, every found configuration is checked against the past observations over a finite temporal window before being approved as in [Nayak and Kurien, 2000] except that candidate generation and consistency checks are interleaved from present time back to the beginning of the temporal window. Configuration solutions to the diagnosis problem contain a mode instanciation of every necessary component in the HTS explaining the observations.

When applied to the first scenario, the diagnosis starts as soon as $\dot{x}$ goes out of its bounds for all currently tracked trajectories: iterating through the system nominal CD from figure 3, at timestep 66 the influences in conflict are $\Gamma = \{T.i_3,\ T.i_2,\ R.i_1,\ R.i_3,\ R.i_4\}$. Relatively to the current configuration (7) it is equivalent to add the constraints $\Gamma_C = \{\bigvee_{m^i = D[T.mode]} T.mode = m^i, R.mode = closed, T.mode = off \vee T.mode = stuck\_off, \bigvee_{m^j \in D[R.mode]} R.mode = m^j\}$ which are activation conditions on the influences in conflict. In this case there are no delays in the equations and the conflicts are stamped with the current physical time.

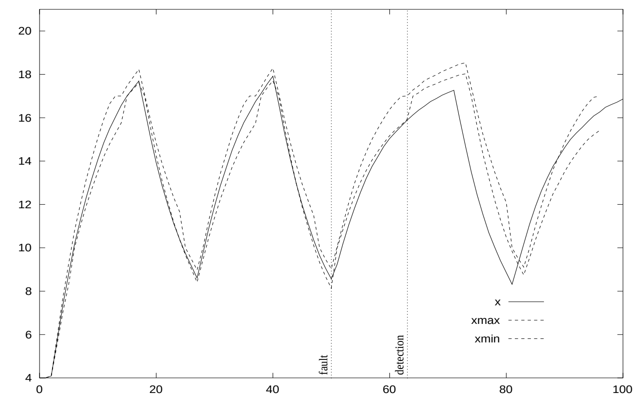
The TMS then seeks for consistency on both the configurations and the transition model starting from the current configuration by inserting the negation of the elements in Γ_C: $\Gamma_{\neg C} = \{T.mode = unknown, R.mode = open \vee R.mode = unknown, T.mode = on \vee T.mode =$



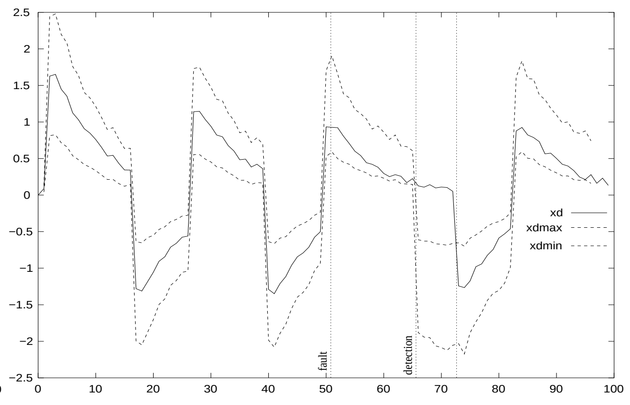
(a) Scenario 1, x: shows the last valid envelope before the fault on the controller is detected. Other envelopes have been discarded because as switching early the thermostat to mode *on* they did not match the observations anymore



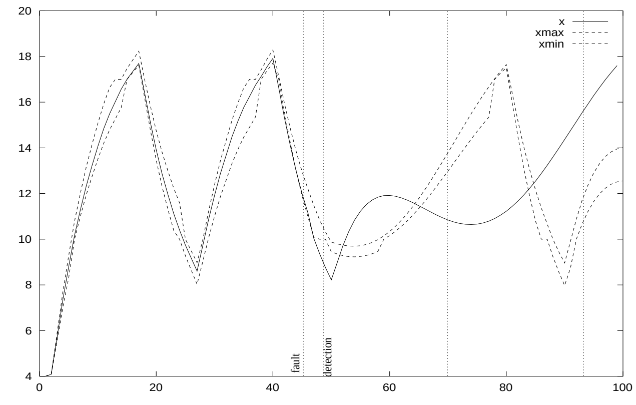
(b) Scenario 1, $\dot{x}$: by letting time progress the envelope will eventually catch the observation again.



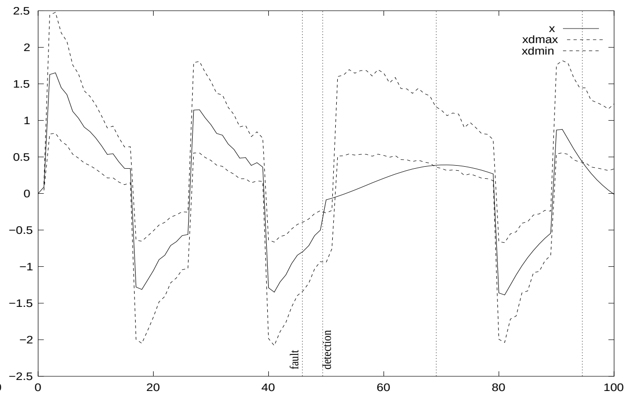
(c) Scenario 2, x



(d) Scenario 2, $\dot{x}$: The fault is not so abrupt as to be detected instantaneously. Moreover it is masked at time-step 73 (it goes *in* the predicted bounds again) for more than 20 time-steps. This is due to the fact that the thermostat's controller is still switching on valid thresholds.



(e) Scenario 3, x:



(f) Scenario 3, $\dot{x}$

Figure 6: Three fault scenarios

$stuck\_on \vee T.mode = unknown, R.mode = unknown\}$ and returns the following possible configurations ranked according to the probabilities attached to transitions and to the number of faults leading to them:

$$\begin{array}{rcl} 1 & : & (R.mode = closed) \wedge (T.mode = stuck\_on) \wedge (R.x \geq M) \\ 2a & : & (R.mode = closed) \wedge (T.mode = unknown) \wedge (R.x \geq M) \\ 2b & : & (R.mode = unknown) \wedge (T.mode = stuck\_on) \wedge (R.x \geq M) \\ 3 & : & (R.mode = unknown) \wedge (T.mode = unknown) \wedge (R.x \geq M) \end{array}$$

Other configurations with the thermostat in modes *on, stuck_off,* or the room in mode *open* are ruled out during the search process because there are no transitions or past observations and commands consistent with these configurations. Diagnosis 1 fits with the fault in the first scenario (thermostat took transition τ^3_{fail}). Scenarios 2 and 3 primarily lead to diagnosis 3 where the room and the thermostat are in the *unknown* mode. However this situation could be improved in scenario 2 by using parameter estimation techniques as proposed in [McIlraith *et al.*, 1999] because the structure of the model is still valid. Note that such faults could also result from the natural degradation of the monitored system and can be taken into account in causal models as in [Pons *et al.*, 1999]. Moreover, if Δx was to be observed, the room and the thermostat would be decoupled so that the room would not move to its unknown mode when the thermostat does.

5 Summary and Discussion

In this paper we extend previous work in the AI community [Williams and Nayak, 1996; Nayak and Kurien, 2000] by presenting a formalism that merges concurrent automata with continuous dynamical system models and reasons on its configurations using logical tools. The problem of reasoning about and diagnosing complex physical plants without extensive study of their continuous state-space is addressed. It is to be applied to the supervision of autonomous satellites as in [Benazera *et al.*, 2001]. At the moment the modeling and simulation tools have been implemented, including the engine that splits the configurations. The program generates a C++ runtime that is intended to be demonstrated on an autonoumous spacecraft test bench at CNES.

An advantage of this approach is that any conditions on transitions and influences (e.g. continuous functions) can be modeled and tracked without being directly observed. Moreover solutions to tracking and diagnosis are logically complete, and sound due to the continuous level structure and variables values. Finally on-line performances can be enhanced as the formalism allows the logical model to be precompiled before use by generating prime-implicants on transition guards [Williams and Nayak, 1997b] and influence conditions. However it still happens that trajectories cannot be discriminated due to too much imprecision on parameters that leads to overlaping envelopes. A solution to this problem is to merge such envelopes and corresponding trajectories. Another remark concerns the splits that occur and are not linked to any real mode or structure changes in the model: when starting the thermostat and room models with external temperature $x_{ext} < m$, a split occurs when first crossing at $x = m$. These splits however are sound and refine the bounds on continuous variables as they allow the system to reduce temporal uncertainty at the crossing point.

Further work will focuse on reconfiguration by reasoning on configurations with the same core algorithms as for diagnosis. This will be done by identifying a set of goal configurations and find under uncertainty a valid plan made of least costly endogeneous commands to reach these goals. In a near future more results are to come as our implementation is intended to be tested on spacecraft models and run on-board ground based satellite hardware.

6 Related Work

Few hybrid formalisms use interval-based uncertain models: however the model checking community has recently investigated this direction [Henzinger *et al.*, 2000]. Previous work on the diagnosis of hybrid systems include [McIlraith *et al.*, 1999] that uses parameter estimation and data fitting to refine the diagnosis. In [Williams *et al.*, 2002] unifies traditional continuous state observers with hidden Markov models belief update for single automatons in order to track hybrid systems with noise. [Asarin *et al.*, 2001] finds the conditions upon which a controller should switch the behavior of the system from one mode to another in order to avoid bad configurations and [Rinner and Kuipers, 1999] uses trackers on semi-quantitative models to monitor dynamic systems even with incomplete knowledge.

7 Acknowledgements

We very thankful to Marie-Claire Charmeau and Bernard Polle for providing information about application and valuable comments on this work.

References

[Alur *et al.*, 1995] R. Alur, C. Courcoubetis, N.Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A.Olivero, J.Sifakis, and S.Yovine. The algorithmic analysis of hybrid systems. In *Proceedings of the 11th International Conference on Analysis and Optimization of Discrete Event Systems*, pages 331–351, 1995.

[Asarin *et al.*, 2001] E. Asarin, O. Bournez, T. Dang, O. Maler, and A. Pnueli. Effective controller synthesis of switching controllers for linear systems. *Proceedings of the IEEE, Special Issue on Hybrid Systems*, 88:1011–1025, July 2001.

[Benazera *et al.*, 2001] E. Benazera, L. Travé-Massuyès, and P. Dague. Hybrid model-based diagnosis for autonomous spacecrafts. In *Proceedings of the first ESA Workshop on On-Board Autonomy, October 2001, Nordwijk, Netherlands*, pages 279 – 286, 2001.

[Henzinger *et al.*, 2000] T. A. Henzinger, B. Horowitz, R. Majumdar, and H. Wong-Toi. Beyond HYTECH: Hybrid systems analysis using interval numerical methods. In *HSCC*, pages 130–144, 2000.

[Henzinger, 1996] T. Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science (LICS '96)*, pages 278–292, New Brunswick, New Jersey, 1996.

[Hyvonen, 1992] E. Hyvonen. Constraint reasoning based on interval arithmetic: The tolerance propagation approach. *Artificial Intelligence*, 58(1-3):71–112, 1992.

[Manna and Pnueli, 1992] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems - Specification*. Springer-Verlag, 1992.

[McIlraith *et al.*, 1999] S. McIlraith, G. Biswas, D. Clancy, and V. Gupta. Towards diagnosing hybrid systems. In *Proceedings of the Tenth International Workshop on Principles of Diagnosis DX-99*, 1999.

[Mosterman and Biswas, 2000] P. J. Mosterman and G. Biswas. A comprehensive methodology for building hybrid models of physical systems. *Artificial Intelligence*, 121:171–209, 2000.

[Nayak and Kurien, 2000] P. Nayak and J. Kurien. Back to the future for consistency-based trajectory tracking. In *Proceedings of AAAI-2000, Austin, Texas*, 2000.

[Pons *et al.*, 1999] R. Pons, L. Travé-Massuyès, and M. Porcheron. Model-based diagnosis and maintenance of time-varying dynamic systems. In *Proceedings of the Tenth International Workshop on Principles of Diagnosis DX-99*, pages 211–219, 1999.

[Rinner and Kuipers, 1999] B. Rinner and B. Kuipers. Monitoring piecewise continuous behaviors by refining semi-quantative trackers. In *IJCAI*, pages 1080–1086, 1999.

[Travé-Massuyès and Jimenez, 2001] L. Travé-Massuyès and J.A. Jimenez. Fault detection and isolation in the ca-en system. Technical report, LAAS-CNRS, Toulouse, France, 2001.

[Travé-Massuyès and Milne, 1997] L. Travé-Massuyès and R. Milne. Tigertm: Gas turbine condition monitoring using qualitative model based diagnosis. *IEEE Expert Intelligent Systems & Applications*, 1997.

[Travé-Massuyès and Pons, 1997] L. Travé-Massuyès and R. Pons. Causal ordering for multiple modes systems. In *Proceedings of the Eleventh International Workshop on Qualitative Reasoning*, pages 203 – 214, 1997.

[Williams and Nayak, 1996] B. C. Williams and P. Nayak. A model-based approach to reactive self-configuring systems. In *Proceedings of AAAI-96, Portland, Oregon*, pages 971–978, 1996.

[Williams and Nayak, 1997a] B. C. Williams and P. Nayak. Fast context switching in real-time reasoning. In *Proceedings of AAAI-97, Providence, Rhode Island*, 1997.

[Williams and Nayak, 1997b] B. C. Williams and P. Nayak. A reactive planner for a model-based executive. In *Proceedings of IFCAI-97*, 1997.

[Williams *et al.*, 2002] B.C. Williams, M. Hofbaur, and T. Jones. Mode estimation of probabilistic hybrid systems. Technical report, Massachusset Institute of Technology, 2002.