Factory Monitoring and Control with Mixed Hardware/Software, Discrete/Continuous Models

Paul Maier, Martin Sachenbacher
Technische Universität München
Boltzmannstraße 3, 85748 Garching, Germany
{maierpa,sachenba}@in.tum.de

Abstract

Many complex systems today, such as robotic networks, automobiles and automated factories, consist of hardware components whose functionality is extended or controlled by embedded software and which exhibit continuous dynamics. We address the problem of monitoring and control in such systems with a twofold contribution. First, we extend Probabilistic Hierarchical Constraint Automata (PHCA), introduced in previous work as a means to compactly describe uncertain hardware and complex software behavior, to hybrid PHCA (HyPHCA). These allow to model continuous behavior in the form of differential equations. Continuous behavior can be conservatively approximated with discrete Markov chains, and in previous work we showed how to transform PHCA monitoring into a constraint optimization problem that can be solved using off-the-shelf reasoners. Our second contribution is to show how to combine these and additional known methods to use a HyPHCA to monitor the internal state and plan for contingencies in a rich class of mixed hardware/software, discrete/continuous systems. Preliminary results of our approach for an industrial filling station scenario demonstrate its feasibility.

Introduction

Many complex systems today, such as robotic networks, automobiles, and automated factories consist of hardware components whose functionality is extended or controlled by embedded software. In such a system, the problem of monitoring its internal state under partial observations and planning for contingencies can be addressed by model-based diagnosis and planning, utilizing a Hidden Markov Model (HMM) of the system's internal behavior. [10] introduced Probabilistic Hierarchical Constraint Automatons (PHCA) as compact HMM encoding, which allows to model uncertain hardware behavior as well as complex software behavior; in previous work [9], we introduced an encoding of PHCAs as soft constraints and a decomposition-based optimization algorithm to efficiently compute best system trajectories over a window of N time steps. However, many real-world components, like the silo of a filling station shown in figure 1, involve not only discrete behavior but also continuous dynamics. Therefore, we are extending PHCAs to Hybrid PHCAs (HyPHCAs) that allow modeling of continuous behavior as Ordinary Differential Equations (ODEs). The main challenge is then to make trajectory estimation on HyPHCAs tractable.

Our example is an industrial filling station [2] which fills a granulate material in bottles. The bottles are transported to and from the station on a conveyor belt. A pneumatic arm puts bottles from the conveyor onto a swivel and back when they are finished. The swivel positions the bottles below a silo, where they are filled by a screw mechanism powered by an electrical motor. A sensor (binary signaled) indicates when the silo is empty. We created a simplified model of the station (shown in figure 1), consisting only of the silo and the sensor model. The silo fill level, during filling, is continuously modeled as $\dot{u}_{\mathrm{lvl}} = -\mathrm{fR} * u_{\mathrm{lvl}}$ (where fR is the fill rate). This equation, while not realistic, demonstrates that our approach can handle such equations.

We consider a scenario (shown in table 1) ranging over 10 time steps ($\triangle t = 2s$). The silo, with initial fill level of 50 units, receives commands to fill two bottles. Within the first 7 time steps, the motor switch breaks such that the motor continues running, emptying the silo (motor-switch-fault). At t_0 the sensor indicates an empty silo. The monitoring problem is to choose among three possible hypotheses explaining the signal: (1) the silo emptied nominally (2) the silo emptied too quickly due to the motor-switch-fault or (3) the sensor is stuck-on. A model which respects the continuous behavior allows a reasoner to detect an inconsistency with the sensor signal: the silo couldn't have emptied nominally, without the motor running. Thus, hypothesis (1) is ruled out. Since the sensor fault is much less likely than the motor fault[1], the reasoner correctly assumes hypothesis (2) as most probable. The control problem is to find suitable actions to deal with

[1]The (failure) probabilities of our model are chosen manually. The vision is that models such as ours will be derived automatically from engineering models, using domain knowledge to generate the probabilities.

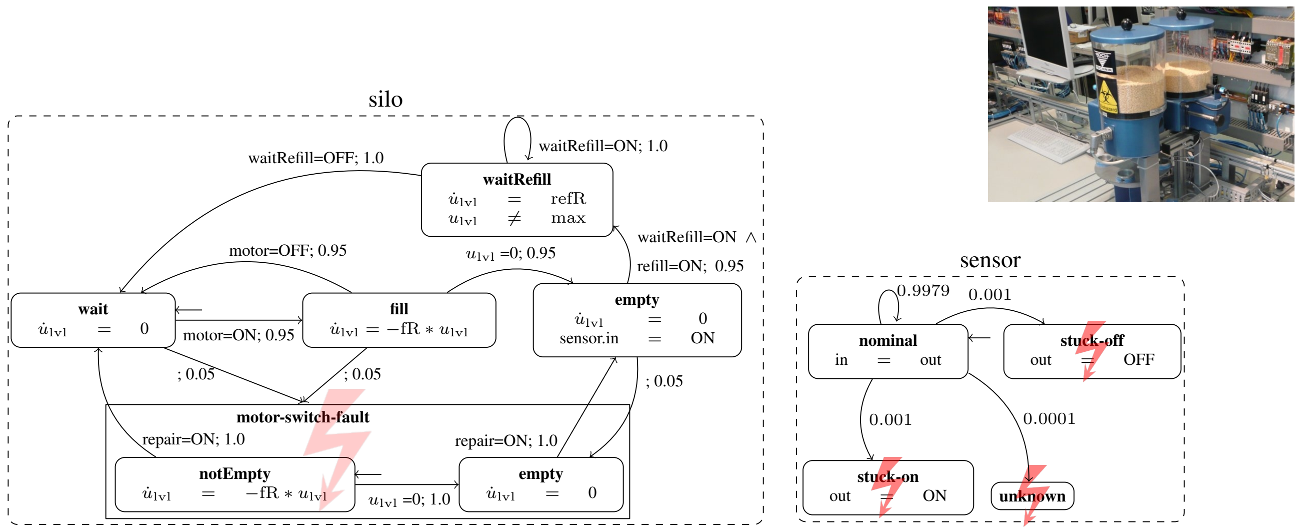
silo
waitRefill=OFF; 1.0
waitRefill=ON; 1.0
waitRefill
$\dot{u}_{\mathrm{lvl}} = \mathrm{refR}$
$u_{\mathrm{lvl}} \neq \max$
waitRefill=ON ∧
refill=ON; 0.95
motor=OFF; 0.95
u_{lvl} =0; 0.95
wait
$\dot{u}_{\mathrm{lvl}} = 0$
motor=ON; 0.95
fill
$\dot{u}_{\mathrm{lvl}} = -\mathrm{fR} * u_{\mathrm{lvl}}$
empty
$\dot{u}_{\mathrm{lvl}} = 0$
sensor.in = ON
; 0.05
; 0.05
; 0.05
motor-switch-fault
repair=ON; 1.0
repair=ON; 1.0
notEmpty
$\dot{u}_{\mathrm{lvl}} = -\mathrm{fR} * u_{\mathrm{lvl}}$
u_{lvl} =0; 1.0
empty
$\dot{u}_{\mathrm{lvl}} = 0$
sensor
0.9979
0.001
nominal
in = out
stuck-off
out = OFF
0.001
0.0001
stuck-on
out = ON
unknown

Figure 1. HyPHCA modeling the silo and the silo empty sensor of a filling station (photo upper right). The bolt indicates failure states (e.g., silo.motor-switch-fault, sensor.stuck-on). All variables may be used for inter-component communication, see e.g. sensor.in in location silo.empty.

the fault and reach a control program goal. In this case the goal is that at t_3 in the future, the silo must have a fill level between 5 and 10 and be in its initial location **wait**.

In the following, we describe an approach, combining several well known methods, which allows to deduce the correct fault hypothesis as well as which actions when to take to reach the goal. In contrast to existing work [6, 7] we do not develop a specific algorithm tailored to our modeling formalism, but instead transform the monitoring and control problems to a constraint optimization problem, which can be solved by off-the-shelf reasoners. The main advantage is that a large body of existing, well refined constraint optimization algorithms can be exploited, and new developments can be incorporated easier.

From Hybrid to Abstracted Discrete Models

A Probabilistic Hierarchical Constraint Automaton (PHCA) [10] is defined as a tuple $A = \langle \Sigma, P_\Theta, \Pi, \mathcal{C}, P_T \rangle$. $\Sigma = \Sigma_c \cup \Sigma_p$ is a set of primitive and composite locations, where the latter denote sub-PHCAs. A location may be marked or unmarked; marked locations represent active execution branches. Π is a set of dependent, observable and commandable variables with finite domains and $\mathcal{C}[\Pi]$ denotes the set of finite domain constraints over Π. P_Θ is a probability distribution over sets of start locations $\Theta_i \in \Sigma$ and $P_T(l_i)$, for each $l_i \in \Sigma_p$, is a probability distribution over a set of transition functions $T(l_i) : \Sigma_p^{(t)} \times \mathcal{C}[\Pi]^{(t)} \rightarrow 2^{\Sigma^{(t+1)}}$. Each transition function maps a marked location into a set of locations to be marked at the next time point, provided that the transition's guard constraint is entailed. $\mathcal{T}$ denotes the set of all transitions. A *PHCA State* at time t is a set of marked locations called a marking $m^{(t)} \subset \Sigma$. A series of $N+1$ PHCA states is a system trajectory, representing its evolution over the course of N time steps. We refer to [10] for a more detailed description.

The PHCA formalism does not support a time model other than discrete, ordered time points. Therefore we define a *clocked PHCA* $A_{\triangle t}$, which is simply a PHCA A where all execution steps take an equal amount of time $\triangle t$. Its time semantics are such that the state of time point t_i is held within $[t_i, t_i + \triangle t)$, and all transitions with entailed guards for this execution step are taken instantaneously at $t_i + \triangle t$.

Systems with mixed discrete/continuous behavior can be modeled using the well known Hybrid Automata [4], capturing continuous system evolution with ODEs. They however don't support hierarchical structure and probabilistic behavior. Therefore we define a hybrid extension to PHCAs called HyPHCAs.

Definition (HyPHCA)

A Hybrid PHCA, or HyPHCA, is a tuple $HA = \langle \Sigma, P_\Theta, \Pi, \mathcal{U}, \mathcal{C}, \mathrm{flow}, P_T \rangle$. $\mathcal{U} = U \cup \dot{U} \cup U'$ is a set of real-valued variables $U = \{u_1, \dots, u_n\}$, their first derivatives $\dot{U} = \{\dot{u}_1, \dots, \dot{u}_n\}$ and a set $U' = \{u'_1, \dots, u'_n\}$ representing values of U at the end of discrete change. $\mathcal{C} : \Sigma \rightarrow \mathcal{C}[\Pi \cup U \cup U']$ is a function associating locations with constraints over discrete and/or real-valued variables. $\mathcal{C}[\Pi \cup U \cup U']$ is the set of constraints over $\Pi \cup U \cup U'$. $\mathrm{flow} : \Sigma \rightarrow \mathrm{flow}[U \cup \dot{U}]$ is a function associating locations with constraints over real-valued variables and their derivatives in the form of *linear ODEs*. $\mathrm{flow}[U \cup \dot{U}]$ is the set of these differential equations. Σ, P_Θ, Π and P_T are defined as for a PHCA.

Our abstraction-based approach to hybrid system mon-

	Time step										
	Past							Present	Future		
Variable	t_{-7}	t_{-6}	t_{-5}	t_{-4}	t_{-3}	t_{-2}	t_{-1}	t_0	t_1	t_2	t_3
sensor.out	OFF	OFF	OFF	OFF	OFF	OFF	OFF	ON	-	-	-
silo.motor	ON	OFF	ON	OFF	OFF	OFF	OFF	OFF	-	-	-
silo.repair	OFF	OFF	OFF	OFF	OFF	OFF	OFF	**ON**	**OFF**	**OFF**	**OFF**
silo.waitRefill	OFF	OFF	OFF	OFF	OFF	OFF	OFF	**OFF**	**ON**	**OFF**	**OFF**
silo.refill	OFF	OFF	OFF	OFF	OFF	OFF	OFF	**OFF**	**ON**	**OFF**	**OFF**
silo location	**wait**	**fill**	**wait**	**m-s-f**	**m-s-f**	**m-s-f**	**m-s-f**	**m-s-f.e**	**empty**	**waitRefill**	wait (goal)
sensor location	**nom.**	**nom.**	**nom.**	**nom.**	**nom.**	**nom.**	**nom.**	**nom.**	**nom.**	**nom.**	**nom.**
u_{lvl}	$[45, 50)$	$[\mathbf{45}, \mathbf{50})$	$[\mathbf{25}, \mathbf{30})$	$[\mathbf{25}, \mathbf{30})$	$[\mathbf{10}, \mathbf{15})$	$[\mathbf{5}, \mathbf{10})$	$[\mathbf{0}, \mathbf{5})$	$[\mathbf{0}, \mathbf{5})$	$[\mathbf{0}, \mathbf{5})$	$[\mathbf{0}, \mathbf{5})$	$[10, 15)$ (goal)

Table 1. The table shows the monitoring/control results for our example scenario (discretization with 10 partition elements of u_{lvl}). The rows show: Known sensor values (1 row), known commands (4 rows), marked locations for sensor and silo (2 rows) and finally the fill level (1 row). Table entries are variable values; bold values are derived automatically by our method. Legend: nom. $\rightarrow$ nominal; m-s-f $\rightarrow$ motor-switch-fault.notEmpty; m-s-f.e $\rightarrow$ motor-switch-fault.empty

itoring relies on converting a HyPHCA HA to a clocked PHCA $A_{\triangle t}$. The evolution of continuous variables $u \in U$ in between two time points t_i and t_{i+1} (where $t_{i+1} = t_i + \triangle t$) is mapped onto discrete, unguarded transitions between locations of a special clocked PHCA $A_{\triangle t}^{\text{Markov}}$. It has only primitive locations, corresponding to cells of a quantization of the continuous state space, and encodes a Markov chain that conservatively approximates the continuous evolution. Given a location $L_{A_{\triangle t}}$ of a clocked PHCA $A_{\triangle t}$ which captures the discrete part of a location L_{HA} of HA (omitting real-valued variables and ODEs), a PHCA $A_{\triangle t}^{\text{Markov}}$ abstracting the continuous behavior $\text{flow}(L_{HA})$ is embedded as a composite location into $L_{A_{\triangle t}}$. We quantize the continuous state space (including time) with equally sized grid cells, but in our approach this method can be easily replaced with more sophisticated ones (e.g., [5]).

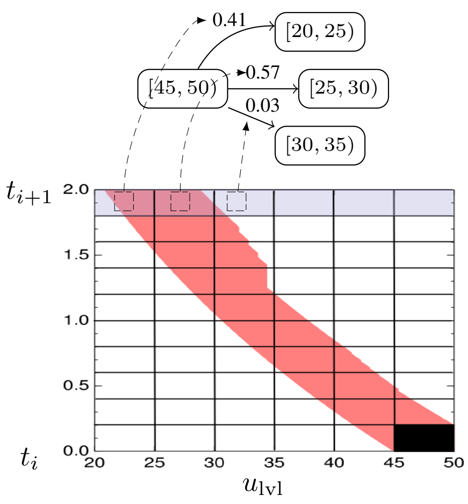

Figure 2. Reachable set R_{start} for $\dot{u}_{\mathrm{lvl}} = -\mathrm{fR} * u_{\mathrm{lvl}}$ starting from the marked grid cell Q_{start}. Above: the derived PHCA $A_{\triangle t}^{\text{Markov}}$.

To conservatively estimate transition probabilities of $A_{\triangle t}^{\text{Markov}}$ we use a well known method which employs reachability analysis [7], illustrated in figure 2. We recap this method shortly. The quantized state space is divided in time slices corresponding to partition elements of the time interval $[t_i, t_{i+1}]$. Start locations of transitions of $A_{\triangle t}^{\text{Markov}}$ are associated with quantization cells within the first time slice in $[t_i, t_{i+1}]$ and destination locations with the last time slice. Let now Q_{start} be the quantization cell of start location L_{start} and $Q_{x,t_{i+1}}$ the cells of all possible destination locations L_x. The approximation R_{start} is computed, a set which is as small as possible yet guaranteed to include all continuous states reachable from Q_{start} within $[t_i, t_{i+1}]$. Now the probabilities for the transitions L_{start} to destination locations L_x are computed as

$$P(L_x | t_{i+1}, L_{\text{start}}) = \frac{V(Q_{x,t_{i+1}} \cap R_{\text{start}})}{V(\bigcup_x Q_{x,t_{i+1}} \cap R_{\text{start}})},$$

where $V()$ measures the volume of the given set. Currently we use PHAVer [3] for reachability analysis, but different approaches can be employed (e.g., [8]).

Best System Trajectories for Monitoring and Control

Given a clocked PHCA $A_{\triangle t}$, partial observations, known commands and a goal state $m^{(t_{i+n})}$, we combine the problems of system monitoring and finding goal achieving commands into a single problem of finding the most probable system trajectory over N time steps which is consistent with the observations and contains $m^{(t_{i+n})}$. From this trajectory the goal achieving commands can be easily derived. We frame this problem as a discrete constraint optimization problem (COP) $\mathcal{R} = (X, D, C)$ with transition probabilities as preferences by translating $A_{\triangle t}$ to soft-constraints following our framework in [9]. This unfolds $A_{\triangle t}$ over N time steps as follows: $X = \{X_1, ..., X_n\}$ is a set of variables with corresponding set of finite domains $D = \{D_1, \ldots, D_n\}$. For all time points $t = 0..N$, it consists of $\Pi^{(t)} \subseteq X$ encoding PHCA variables, auxiliary variables (needed to, e.g., encode hierarchical structure) and a set of binary variables $Y = \{X_{L_1}^{(t_i)}, X_{L_2}^{(t_i)}, \ldots\} \subseteq X$ representing location markings of $A_{\triangle t}$. Y is the set of solution variables of $\mathcal{R}$. $C = \{C_1, \ldots, C_n\}$ is a set of constraints (S_i, F_i) with scope $S_i = \{X_{i1}, \ldots, X_{ik}\} \subseteq X$ and a constraint function $F_i : D_{i1} \times \cdots \times D_{ik} \to [0,1]$ mapping partial assignments of variables in S_i to a probability value in $[0,1]$. For all time steps $t = 0..N$, hard constraints in C (F_i evaluates to $\{0,1\}$ only) encode hierarchical structure as well as consistency of observations and commands with locations and transitions, while soft constraints in C encode probabilistic choice of initial locations at $t = 0$ and probabilistic transitions. $\mathcal{R}$ then consists of $O(N(|\mathcal{T}| + |\Sigma| + |\Pi|))$ variables and $O(N(|\Sigma| + |\mathcal{T}|))$ constraints. The k-best solutions to $\mathcal{R}$ are assignments to Y which, extended to all variables X, maximize the global probability value in terms of the functions F_i. These assignments correspond to the most probable PHCA system trajectories and their extension to X provides assignments to, e.g., goal achieving commands. All described steps up to now — discretizing, generating Markov chains, COP encoding — can be done offline. Online, we iteratively add observations and known commands to $\mathcal{R}$ and solve it, yielding the k most likely system trajectories.

Experimental Results

We created COP instances with different discretizations for u_{lvl} (2, 10 and 25 partition elements) for our example scenario and two small variations, and solved them using Toulbar2[2]. We tried its default and a second, decomposition based configuration. The problem size was for all instances (unfolded over 10 time steps) 843 variables and 920 constraints. For 10 partition elements of u_{lvl} table 1 shows the variable assignments the solver deduced from the given observations and goals in bold face. The generated solution correctly identifies the motor-switch-fault and provides the necessary commands to reach the goal: $\text{repair} = \text{ON}$ for t_0, $\text{refill} = \text{ON}$ and $\text{waitRefill} = \text{ON}$ for t_1 and $\text{waitRefill} = \text{OFF}$ for t_2. Table 2 shows the average runtime. The columns show results for the example scenario (1) and the two variations: diagnose motor-switch-fault only (2) and nominal behavior (3). As one would expect, a slight increase in runtime can be seen for the more fine grained discretization (25). The variations take roughly the same time as the main scenario. Small differences are probably due to the fact that the variations are the same COP with some constraints omitted. E.g., when diagnosing the motor-switch-fault only, the goal is omitted. This makes the problem slightly harder because more future evolutions are possible. We expected the offline decomposition of the problem to lower online computation effort, but surprisingly, it had a negative effect in our scenario, with yet unknown cause.

[2]`https://mulcyber.toulouse.inra.fr/projects/toulbar2/`

Toulbar2 config.	Discreti-zation	Scenarios		
		1	2	3
default params	2	0.016s	0.026s	0.028s
	10	0.014s	0.026s	0.016s
	25	0.030s	0.054s	0.032s
with tree decomp.	2	0.126s	0.172s	0.200s
	10	0.122s	0.156s	0.164s
	25	0.138s	0.178s	0.178s

Table 2. Runtime results (mean time in sec.) for the example scenario and its variations.

Conclusion

We introduced HyPHCAs, an extension to PHCAs, as a modeling framework for mixed discrete/continuous systems and showed how to combine several existing methods to offline abstract the differential equations of the HyPHCA to Markov chains encoded as PHCAs, embed them in the discrete part of the HyPHCA, encode the discrete abstraction with soft-constraints and online monitor and control the system by solving a constraint optimization problem. Our experimental results demonstrate the feasibility of the approach. Our next step is to develop an estimator module which iteratively shifts the time window ([9]) to monitor systems over long time periods and verify our results on larger factory settings such as [1]. In this and in other settings, accurate model-based monitoring and control can only be achieved by considering both hybrid hardware and software behavior.

References

[1] M. Buss, M. Beetz, and D. Wollherr. CoTeSys - Cognition for Technical Systems. In *Proc. HAM*, 2007.

[2] S. Dominka. *Hybride Inbetriebnahme von Produktionsanlagen — Von der Virtuellen zur Realen Inbetriebnahme (in German)*. PhD thesis, TUM, 2007.

[3] G. Frehse. Phaver: Algorithmic Verification of Hybrid Systems Past Hytech. In *Proc. HSCC*, pages 258–273, 2005.

[4] T. Henzinger. The Theory of Hybrid Automata. In *Proc. LICS*, pages 278–292, New Brunswick, New Jersey, 1996.

[5] M. W. Hofbaur and T. Rienmüller. Qualitative Abstraction of Piecewise Affine Systems. In *Proc. QR*, 2008.

[6] M. W. Hofbaur and B. C. Williams. Mode Estimation of Probabilistic Hybrid Systems. In *HSCC*, pages 253–266, Stanford, California, USA, 2002. Springer.

[7] J. Lunze and B. Nixdorf. Representation of Hybrid Systems by Means of Stochastic Automata. *Mathem. and Comp. Modelling of Dyn. Sys.*, 7:383–422, Dec. 2001.

[8] M. Althoff, O. Stursberg, and M. Buss. Online Verification of Cognitive Car Decisions. In *Proc. IEEE IV*, 2007.

[9] T. Mikaelian, B. C. Williams, and M. Sachenbacher. Model-based Monitoring and Diagnosis of Systems with Software-Extended Behavior. In *Proc. AAAI*, 2005.

[10] B. C. Williams, S. Chung, and V. Gupta. Mode Estimation of Model-Based Programs: Monitoring Systems with Complex behavior. In *Proc. IJCAI*, pages 579–590, 2001.