Available online at www.sciencedirect.com

ScienceDirect

IFAC-PapersOnLine 48-27 (2015) 351–357

Modeling and Optimization of Hybrid Systems

Bengt Lennartson * Oskar Wigström * Sarmad Riazi *
Kristofer Bengtsson *

* *Department of Signals and Systems, Chalmers University of Technology, SE-412 96 Göteborg, Sweden (e-mail: bengt.lennartson@chalmers.se)*

Abstract: In this paper a predicate transition model for discrete event systems is generalized to include continuous dynamics, and the result is a modular hybrid predicate transition model. Based on this model a hybrid Petri net, including explicit differential equations and shared variables, is also proposed. It is then shown how this hybrid Petri net model can be optimized based on a simple and robust nonlinear programming formulation. The procedure only assumes that desired sampled paths for a number of interacting moving devices are given, while originally equidistant time instances are adjusted to minimize a given criterion. This optimization of hybrid systems is also applied to a real robot station with interacting devices, which results in about 30% reduction in energy consumption.

Keywords: discrete-event systems, Petri net, hybrid systems, optimization

1. INTRODUCTION

A generic model for hybrid systems is proposed based on modular components and transition predicates on shared variables. This modeling framework is an extension of a recently presented state-vector transition model, Lennartson et al. (2014a), that unifies models for discrete event systems, such as automata and Petri nets, optionally including discrete shared variables. The extended hybrid predicate transition model (HPTM) is shown to be both flexible, general and easy to adapt to different types of hybrid automata and hybrid PNs. Ordinary hybrid PNs (HPNs) David and Alla (2001) combine continuous and discrete PNs. Continuous PNs were originally introduced as an approximation of discrete PNs with many tokens, but they also work well for modeling continuous flow systems. Logical conditions on the continuous flows, represented by discrete places and tokens, result in HPNs that are purely graphical models.

Different simplifications and generalizations of this hybrid model have been proposed such as differential PNs, Demongodin and Koussoulas (2006), differential hybrid PNs, Sousa and Lima (2008), and differential predicate transition nets, Villani et al. (2007). For pure flow systems, the graphical information in HPNs on the continuous states gives a valuable insight. In this paper it is shown that logical conditions on the continuous flows are preferably expressed as predicates on continuous states and discrete variables, instead of complex discrete place-transition formulations. Moreover, a modular formulation of HPNs is demonstrated to give more readable models for larger systems.

We also propose *hybrid Petri nets including differential equations* (HPNDs), where differential equations (DEs) are introduced in a similar way is in hybrid automata Alur et al. (1993). The parallel behavior of HPNs means, however, that some restrictions are introduced, where an HPND must be complete and nonconflicting to guarantee a well formulated continuous time model. HPNDs are appropriate especially for systems that have no pure flow character, such as moving devices with double integrator and/or resonance dynamics.

Explicit DEs are also used in differential predicate transition nets, Villani et al. (2007), which are related to high level PNs where tokens involve data. Our proposed HPNs, including both explicit differential equations and shared variables, are on the other hand based on ordinary PNs. They have a more simple and modular structure, where shared variables are introduced to easier express complex logical relations between different local HPN models.

The proposed extensions of hybrid Petri nets are used for optimization of hybrid systems. Barton and Lee (2002) argue that hybrid optimal control problems are very difficult to solve, due to the combinatorial explosion in the discrete decision space, combined with the infinite dimensional continuous trajectories in each mode. However, a general and efficient method for optimization of hybrid systems, recently presented in Wigström and Lennartson (2014), is further developed in this paper for moving devices with a given path. A procedure adjusting the time axis by a simple nonlinear programming formulation is shown to work extremely well for energy optimization of robot stations, including shared and interacting zones. The energy consumption is evaluated on real industrial robots with impressive reduction of the energy consumption.

The main contributions of this paper are as follows: A recently presented predicate transition model is generalized to modular hybrid predicate transition systems. A new type of modular HPNs including differential equations and shared variables is also proposed, demonstrated and applied to a real robot station with interacting moving devices. Furthermore, a simple and powerful strategy for optimization of hybrid systems is proposed for moving devices, where the path is given but not the complete trajectory. Applying this method shows that up to 30% energy reduction can be obtained for real industrial and interacting robots.

In Section 2 the hybrid predicate transition model is presented, followed by different types of hybrid Petri nets, presented in

⋆ This work was carried out within the Wingquist Laboratory VINN Excellence Centre within the Area of Advance - Production at Chalmers, and supported by VINNOVA and the Swedish Science Foundation. The support is gratefully acknowledged.

2405-8963 © 2015, IFAC (International Federation of Automatic Control) Hosting by Elsevier Ltd. All rights reserved.
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2015.11.199

Section 3. In Section 4 it is shown how hybrid systems can be optimized, and conclusions are given in Section 5.

2. GENERIC MODEL FOR HYBRID SYSTEMS

A generic model for hybrid systems is presented in this section, based on modular components and transition predicates on shared variables. It is an extension and clarification of a preliminary formulation in Lennartson et al. (2014b).

2.1 Hybrid Predicate Transition Model

First, observe that a subset W of a set X can also be defined by the *predicate* mapping $\mathcal{W} : X \to \mathbb{B}$ as $\mathcal{W}(x) = 1$ iff $x \in W$ and $\mathcal{W}(x) = 0$ iff $x \notin W$.

The proposed *hybrid predicate transition model* (HPTM) is based on a universal ordered set (tuple) of variables $(x_1, \dots, x_n)$. This tuple includes both discrete and continuous variables, where the domain of definition for each variable x_j is X_j. A subset of these variables is included in a tuple x for which the HPTM is defined.

Definition 1. (Hybrid Predicate Transition Model). A hybrid predicate transition model G is a 9-tuple

$$G = \langle \Omega_x, \Omega_{x_c}, X, \Sigma, T, \mathcal{X}_c, \mathcal{X}_i, \mathcal{X}_m, \mathcal{X}_{inv} \rangle, \tag{1}$$

where

(i) $\Omega_x = \{j_1, \dots, j_n\}$ is the index set for the tuple $x = (x_{j_1}, \dots, x_{j_n})$.

(ii) $\Omega_{x_c} = \{\ell_1, \dots, \ell_{n_c}\}$ is the index set for the tuple $x_c = (x_{\ell_1}, \dots, x_{\ell_{n_c}})$ of continuous state variables, also being a subset of the variables in x.

(iii) $X = X_{j_1} \times \cdots \times X_{j_n}$ is the domain of definition for x.

(iv) Σ is a finite set of events.

(v) T is a finite set of transitions. Each transition is a tuple $(\sigma, \mathcal{C})$, where $\sigma \in \Sigma$ and $\mathcal{C} : X \times X \to \mathbb{B}$ is a predicate on the current value x and the next value $\acute{x}$.

(vi) $\mathcal{X}_c : X \times X_c \to \mathbb{B}$ is the continuous state predicate on the current value x and the time derivative of the continuous state vector $\dot{x}_c$, where X_c is the domain of definition for x_c.

(vii) $\mathcal{X}_i : X \to \mathbb{B}$ is a predicate, defining possible initial values of x.

(viii) $\mathcal{X}_m : X \to \mathbb{B}$ is a predicate, defining marked (desired) values of x.

(ix) $\mathcal{X}_{inv} : X \to \mathbb{B}$ is a predicate, defining desired invtariants of x that must always be satisfied. □

The tuple x is normally divided into three parts $x = (x_c, u_c, x_d)$, including the continuous state variables in x_c, but also continuous input signals in u_c and discrete variables in x_d. Thus, $\Omega_{x_c} \subseteq \Omega_x$ and the domain of definition X includes both continuous and discrete domains. The reason to introduce the index set Ω_x is that variables can be arbitrarily shared between different local models. The predicates are generated by boolean expressions, including conjunction $\wedge$, disjunction $\vee$, and negation $\neg$, while relations between variable values involve the operators $=, \neq, <, >, \leq$, and $\geq$.

A transition $(\sigma, \mathcal{C})$ is *enabled* when the predicate $\mathcal{C}(x, \acute{x})$ is true. When the enabled transition is executed the event σ occurs. Also note the condition on the next value $\acute{x} \in X$. Assuming that $X = \{0, 1, \dots, n\}$, this means that the conditions $\acute{x} = x + 1$ and $\acute{x} = x - 1$ implicitly include the additional guards $x < n$ and $x > 0$, respectively. These guards on the current value of x do not need to be explicitly introduced, since they are achieved by the domain of definition for $\acute{x}$.

Keep-current-value semantics When no condition on $\acute{x}_j$ is included in $\mathcal{C}(x, \acute{x})$, it is assumed that x_j keeps its current value. Therefore, consider the index set

$$\Omega_{\mathcal{C}} = \{j \mid \text{condition on } \acute{x}_j \text{ in } \mathcal{C}(x, \acute{x})\}.$$

Any expression involving $\acute{x}_j$, such as $\acute{x}_j = x_j + 2$ or $\acute{x}_j < 3$, implies that $j \in \Omega_{\mathcal{C}}$. No condition on $\acute{x}_j$ in $\mathcal{C}(x, \acute{x})$ yields $j \in \Omega_x \setminus \Omega_{\mathcal{C}}$, and the variable x_j will be assumed to keep its current value, i.e. $\acute{x}_j = x_j$. Introducing the keep-current-value predicate

$$\mathcal{C}_{cv}(x, \acute{x}) \equiv \bigwedge_{j \in \Omega_x \setminus \Omega_{\mathcal{C}}} \acute{x}_j = x_j, \tag{2}$$

the *complete transition predicate* for transition $(\sigma, \mathcal{C})$ becomes $\Phi(x, \acute{x}) \equiv \mathcal{C}(x, \acute{x}) \wedge \mathcal{C}_{cv}(x, \acute{x})$. This results in the following set of explicit state transition relations $\{(x, \sigma, \acute{x}) \in X \times \Sigma \times X \mid \exists (\sigma, \mathcal{C}) \in T : \Phi(x, \acute{x})\}$.

Continuous dynamics The predicate $\mathcal{X}_c(x, \dot{x}_c)$ typically includes a nonlinear state space model $\dot{x}_c = f(x)$, where the dependency on the discrete part in x may result in different dynamics depending on the actual value of x_d. More general forms like the differential algebraic equation $g(x, \dot{x}_c) = 0$ and differential inclusions such as $\dot{x}_c \geq f_1(x) \wedge \dot{x}_c \leq f_2(x)$ may be included in the predicate $\mathcal{X}_c(x, \dot{x}_c)$ as well. The continuous state variables can also be updated at the transitions as state jumps, where resetting clocks is a classical example.

2.2 Synchronous Composition

The synchronous composition of HPTMs is defined based on Hoare's full synchronous composition Hoare (1978), but extended to include shared variables.

Definition 2. (Synchronous composition of HPTMs). Let $G^k = \langle \Omega_x^k, \Omega_{x_c}^k, X^k, \Sigma^k, T^k, \mathcal{X}_c^k, \mathcal{X}_i^k, \mathcal{X}_m^k, \mathcal{X}_{inv}^k \rangle$, $k = 1, 2$, be two HPTMs, including their individual tuple of variables x^k. The synchronous composition of G^1 and G^2 is then defined as

$$G^1 \| G^2 = \langle \Omega_x^1 \cup \Omega_x^2, \Omega_{x_c}^1 \cup \Omega_{x_c}^2, X, \Sigma^1 \cup \Sigma^2, T, \mathcal{X}_c^1 \wedge \mathcal{X}_c^2, \mathcal{X}_i^1 \wedge \mathcal{X}_i^2, \mathcal{X}_m^1 \wedge \mathcal{X}_m^2, \mathcal{X}_{inv}^1 \wedge \mathcal{X}_{inv}^2 \rangle,$$

where the domain of definition X and the corresponding tuple of variables x are defined based on the index set $\Omega_x^1 \cup \Omega_x^2$, according to Definition 1. The transition $(\sigma, \mathcal{C}) \in T$ is defined for each combination of $(\sigma, \mathcal{C}^k) \in T^k$, $k = 1, 2$, such that

$$\mathcal{C}(x, \acute{x}) \equiv \begin{cases} \mathcal{C}^1(x^1, \acute{x}^1) \wedge \mathcal{C}^2(x^2, \acute{x}^2), & \sigma \in \Sigma^1 \cap \Sigma^2 \\ \mathcal{C}^1(x^1, \acute{x}^1), & \sigma \in \Sigma^1 \setminus \Sigma^2 \\ \mathcal{C}^2(x^2, \acute{x}^2), & \sigma \in \Sigma^2 \setminus \Sigma^1 \end{cases}. \tag{3}$$

□

The reason why the synchronization in (3) only involves the predicate condition $\mathcal{C}$, but not the keep-current-value predicate $\mathcal{C}_{cv}$, is that shared variables can be updated in different HPTMs. The keep-current-value predicate is therefore a global property that can be determined first after all local models have been synchronized. Then, the set $\Omega_{\mathcal{C}}$ for the global model and the complete transition predicate $\Phi(x, \acute{x})$ are determined.

It is reasonable to assume that a continuous state variable is only updated in one local model. Thus, it is assumed that the continuous state vectors x_c^1 and x_c^2, representing the continuous state of corresponding local model, do not have any shared variables. This means that $\Omega_{x_c^1} \cap \Omega_{x_c^2} = \emptyset$. Logical conditions on these state variables are typically transferred between the local models by discrete variables. This is illustrated in the following example, where it will also be shown how modular representations of hybrid systems are easily formulated by HPTMs.

Example 1. (Two robots with shared zone). Consider two one degree of freedom (1-DOF) robots with identical continuous dynamics

$$\begin{bmatrix} \dot{\theta}^k \\ \dot{\omega}^k \end{bmatrix} = \begin{bmatrix} \omega^k \\ -d\cos(\theta^k) + u^k \end{bmatrix}, \tag{4}$$

where d is the distance to the center of gravity, and $k = 1, 2$ for the two robots. Also, suppose that these robots are to move through a shared zone. The robots are in this shared zone when $|\theta^k| \leq \pi/4$ and outside otherwise. To avoid collision between the two robots, the shared zone is considered as a common resource that needs to be booked to be allowed to access the zone. This is easily represented by a shared discrete variable $Z \in \{0, 1\}$, where $Z = 0$ means that the zone is booked, and $Z = 1$ the opposite.

The hybrid system involving the two robots is modeled as two local HPTMs G^k. In Fig. 1 the local G^k is graphically represented as an automaton, including the shared variable Z for the discrete part and the continuous state invariants. The initial location is outside the shared zone ($|\theta^k| > \pi/4$), and the common continuous-time dynamics is given by the continuous state predicate $\dot{\theta}^k = \omega^k \wedge \dot{\omega}^k = -d\cos(\theta^k) + u^k$. Moreover, notice the individual tuple of variables $x^k = (\theta^k, \omega^k, u^k, Z)$, which includes both the continuous state variables θ^k and ω^k, the input signal u^k, and the shared discrete resource variable Z that is common for G^1 and G^2, resulting in the total hybrid system $G = G^1 \| G^2$. □

Compared to the well-established hybrid automaton, see Alur et al. (1993), this HPTM is more general, clean and flexible. The logical behavior can be expressed not only by transitions between specific locations, but also by combining locations with discrete variables, in this example the shared zone variable Z. Also note the modular formulation, where all variables are local, except for the shared variable Z.

The HPTM is not bounded to any specific graphical representation. Our experience shows, however, that focusing the graphical model on parts of the discrete behavior, including some shared discrete variables, as Z in Fig. 1, is often preferable.

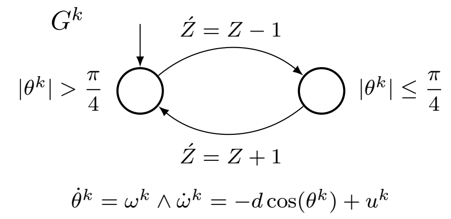
G^k
$Z' = Z - 1$
$|\theta^k| > \frac{\pi}{4}$
$|\theta^k| \leq \frac{\pi}{4}$
$Z' = Z + 1$
$\dot{\theta}^k = \omega^k \wedge \dot{\omega}^k = -d\cos(\theta^k) + u^k$

Fig. 1. Local HPTM for a 1-DOF robot, where the discrete part is modeled as an EFA, including the discrete shared variable Z representing the shared zone between the two robots.

The continuous part can often be modeled by a common set of differential equations, where mode specific details are handled by adding local discrete variables that are updated at the discrete transitions. A graphical representation also involving continuous time dynamics is an additional option included in the next section, where different versions of hybrid PNs are presented.

3. HYBRID PETRI NETS

Hybrid Petri nets, including different combinations of graphical and equation based formulations, will be discussed in this section. They are all formally defined as HPTMs, but with different graphical representations.

3.1 Ordinary Hybrid Petri Nets

In an ordinary hybrid Petri net, David and Alla (2001), both the discrete and the continuous dynamics are represented graphically. This includes both discrete places (circles) and continuous places (double circles), as well as discrete transitions (filled thin rectangles) and continuous transitions (empty thin rectangles).

To illustrate a HPN, consider the tank process in Fig. 2, where liquid is added and removed from three tanks by input and output on-off valves (open $v_k = 1$, closed $v_k = 0$, $k = 1, 2, 3$). Furthermore, vessels on a conveyor belt are filled with liquid by pumping from the third tank. The check valve is open when the height h_1 in Tank 1 is higher than the height h_2 in Tank 2. To examplify the continuous flow, observe that the height in Tank 2 is defined by the differential equation

$$\dot{h}_2 = \begin{cases} \dfrac{1}{A_2}\big(v_2 q_2 + q_{12} - v_3 q_3\big) & h_1 > h_2, \\ \dfrac{1}{A_2}\big(v_2 q_2 - v_3 q_3\big) & h_1 \leq h_2. \end{cases}$$

Example 2. (HPN for tank process). An HPN for the tank process in Fig. 2 is shown in Fig. 3. The flow rates q_1, q_{12}, q_2, and

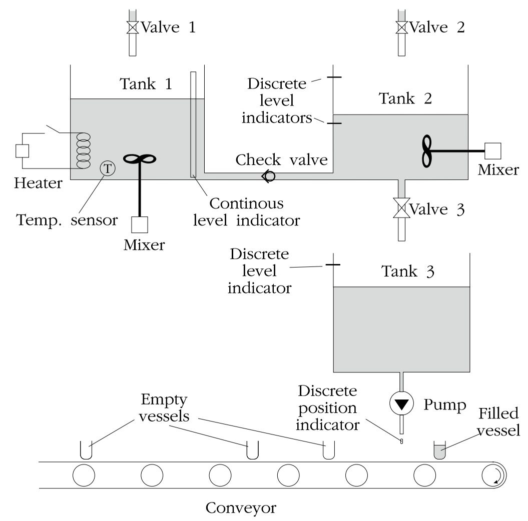

Fig. 2. A tank process including three tanks and a conveyor where vessels are filled with liquid, cf. Pettersson and Lennartson (1995).

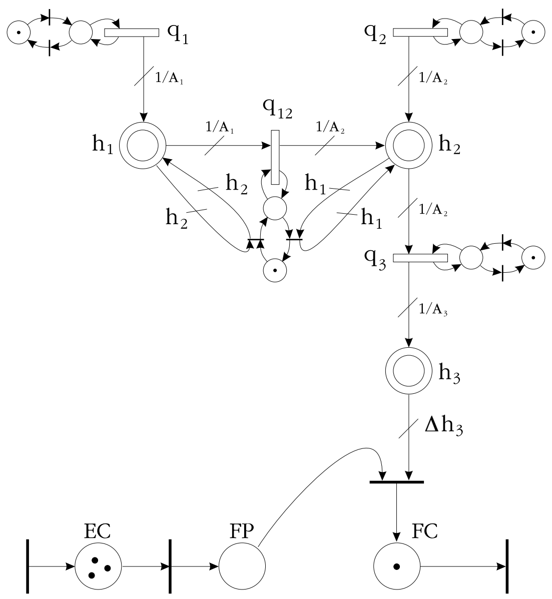

Fig. 3. An HPN for the tank process in Fig. 2, cf. Pettersson and Lennartson (1995).

q_3 are defined at the continuos transitions, including weighting factors at the edges. The on-off valves are represented by discrete decision places that control the continuous transitions by self-loops. Moreover, the transitions on the check valve model the on condition $h_1 > h_2$ and the off condition $h_1 \leq h_2$ by self-loops on the continuous height places h_1 and h_2, respectively.

The discrete transition above place FC is fired when $h_3 \geq \Delta h_3$ and there is a discrete marking in place FP (filling position). When this transition is fired the height h_3 is immediately reduced by a quick pumping action. This is modeled as a continuous state jump, such that the updated height $\acute{h}_3 = h_3 - \Delta h_3$. □

3.2 Modular Hybrid Petri Nets with Shared Variables

Instead of modeling all discrete conditions graphically, some of them can be expressed by predicates in an HPN. An HPN can also be separated into local *hybrid Petri nets including shared variables* (HPNSVs). This is illustrated in the following example.

Example 3. (HPNSVs for tank process). Consider the HPN in Fig. 4, where the on-off valves have been simplified to predicates on the discrete variables v_1, v_2, and v_3, and the predicate on the check valve depends the heights h_1 and h_2. Obviously, it is easier to understand the predicates in Fig. 4, especially the one on the check valve, compared to corresponding graphical representations in Fig. 3.

A modular version of this HPN is given by three HPNSVs in Fig. 5, where shared variables h_1, h_2, and q_{12}, and a shared event σ are used to synchronize the behavior between the local HPNSVs. Note that the modularization is not unique. The right continuous flow including Tank 2 and Tank 3 can also be divided into two HPNSVs, one for each tank. Alternatively, all three tanks can be composed into one HPNSV. □

Comparing the HPNs in Fig. 4 and Fig. 5 shows that for flow systems of moderate size the global model in Fig. 4 may be

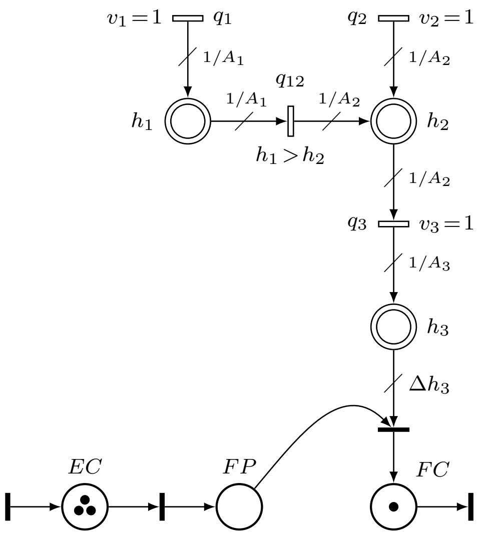

Fig. 4. An HPN including explicit predicates, modeling the same behavior as the HPN in Fig. 3.

preferable. For larger systems, benefits are on the other hand achieved by dividing global HPNs into smaller communicating modules, which sizes depend on the actual application.

3.3 Hybrid Petri Nets with Explicit Differential Equations

For systems without a natural flow, *hybrid Petri nets including explicit differential equations* (HPNDs) are often preferable. Differential equations (DEs) are then introduced at discrete places in a similar way as DEs are included in hybrid automata, optionally including shared variables, see Fig. 1. There is, however, one fundamental difference, where a hybrid automaton has only one active or executing location at a time, and the DEs in a specific location are only executed when that location is active.

PNs, on the other hand, may include parallel behavior, where tokens are distributed in various places at the same time. For a place p_i with a number of tokens m_i, the place is called an *active place* when $m_i > 0$. In the proposed HPND it is assumed that all DEs included in an active place are executed, as long as

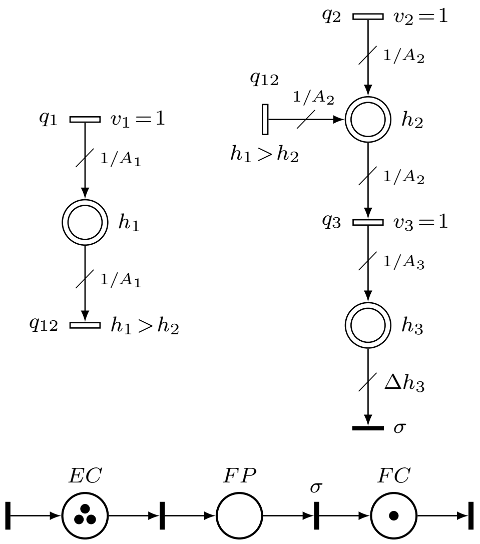

Fig. 5. A number of HPNSVs, modeling the same behavior as the HPNs in Fig. 3 and 4.

this place is active. A number of places may be active at the same time, and DEs are generally distributed arbitrarily among the places, including a number of places without any DEs. Thus, there is a potential risk that different DEs for the same continuous state variable are defined in different places, which happen to be active at the same time. In the same way, some continuous state variables may lack DEs for some specific token combinations. Therefore, two formal conditions are introduced to avoid these problems.

In each place p_i of an HPND, local DEs, or more generally conditions on $\dot{x}_c$, are defined by a local *continuous-state place-predicate* $\mathcal{X}_{ci}(x, \dot{x}_c)$, and the set

$$\Omega_{\mathcal{X}_{ci}} = \{j \mid \text{condition on } \dot{x}_j \text{ in } \mathcal{X}_{ci}(x, \dot{x}_c)\}$$

is the index set for the continuous variables x_j for which there are conditions on their time derivative in $\mathcal{X}_{ci}$. No condition on $\dot{x}_c$ in a place p_i means that $\mathcal{X}_{ci}(x, \dot{x}_c) \equiv$ True and $\Omega_{\mathcal{X}_{ci}} = \varnothing$. The time varying set $\Omega_m = \{i \,|\, m_i > 0\}$ is the index set for the currently active places p_i, where there is at least one token ($m_i > 0$). This set is updated after each discrete transition in the HPND.

When there are conditions on all continuous state derivatives for each reachable marking vector m, an HPND it is said to be *complete*. At the same time, conditions on the derivative of one specific state variable $\dot{x}_j$ should not be included in more than one active place, to avoid conflicts in the continuous state space solution. An HPND without such conflicts for any continuous state variable and for all reachable markings, is said to be *nonconflicting*. These two properties are now formalized in the following definition.

Definition 3. (Complete and nonconflicting HPND). An HPND is *complete*, if for each reachable marking vector m

$$\bigcup_{i \in \Omega_m} \Omega_{\mathcal{X}_{ci}} = \Omega_{x_c}, \tag{5}$$

and *nonconflicting*, if for each reachable marking vector m

$$\Omega_{\mathcal{X}_{ci}} \cap \Omega_{\mathcal{X}_{cj}} = \varnothing, \quad i, j \in \Omega_m, \quad i \neq j. \tag{6}$$

□

The following example illustrates these properties for an HPND version of the two robots in Example 1.

Example 4. (HPND for two robots with shared zone). An HPND corresponding to the two 1-DOF robots in Fig. 1 is shown in Fig. 6, where the continuous dynamics has been simplified to a double integrator process, and the invariants have been replaced by entry and exit points for the shared zone. The continuous state vector is $x_c = (\theta^1, \dot{\theta}^1, \theta^2, \dot{\theta}^2)$, and $x = (x_c, m)$ where m is the marking vector. Therefore, $\Omega_{x_c} = \{1, 2, 3, 4\}$, and we see that the conditions (5) and (6) are satisfied for the initial marking m_0, since $\Omega_{m_0} = \{1, 3, 5\}$ and $\Omega_{\mathcal{X}_{c1}} = \{1, 2\}$, $\Omega_{\mathcal{X}_{c3}} = \{3, 4\}$, while $\Omega_{\mathcal{X}_{c5}} = \varnothing$. In the same way these conditions are satisfied for the other reachable markings, where $\Omega_m = \{2, 3\}$ and $\Omega_m = \{1, 4\}$. Thus, (5) and (6) are satisfied for all reachable marking vectors, and the HPND is complete and nonconflicting. □

For a complete and nonconflicting model, the predicate $\mathcal{X}_c$ in Definition 1 can be expressed as $\mathcal{X}_c \equiv \bigwedge_{i \in \Omega_m} \mathcal{X}_{ci}$. Furthermore, if two local HPNDs G_1 and G_2 are complete and nonconflicting, the synchronized version $G_1 \| G_2$ will be the same. This means that modular HPNDs are formally defined as HPTMs, according to Definition 1.

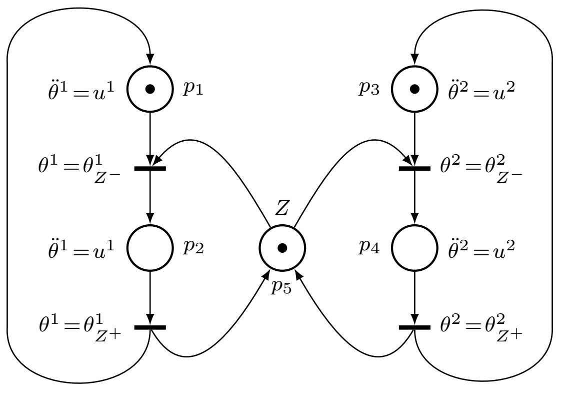

Fig. 6. HPND for the two 1-DOF robots in Fig. 1.

Combining HPNDs and HPNSVs results in *hybrid Petri nets including explicit differential equations and shared variables* (HPNDVs). This is a flexible modeling language, where graphical modeling can be used where it has its strength, typically showing continuous and/or discrete flows between places. Complicated logical conditions related to mutual exclusion, synchronization, precedence relations, and continuous states are preferably expressed by predicates on discrete and continuous variables. Logical conditions between places, far away from each other, generate complex graphical models. These models may be much harder to understand than modularized HPNDVs with added explicit transition predicates on variables, whose names can be chosen to clearly express the meaning of for instance resource booking and conditions on continuous variables, as in Fig. 5. In the next section it is shown how robot systems modeled as hybrid Petri nets can be optimized, especially concerning energy consumption.

4. OPTIMIZATION OF HYBRID SYSTEMS

The objective in optimization of hybrid systems is to compute optimal or sub-optimal trajectories for the systems. In the context of hybrid systems, a trajectory can be regarded as a sequence of discrete modes, as well as a set of continuous trajectories describing the continuous dynamics for each discrete mode. The combinatorial explosion in the discrete decision space, combined with the infinite dimensional continuous trajectories in each mode makes the hybrid optimal control problem very difficult to solve, Barton and Lee (2002). Several approaches to solve this challenging problem have been proposed. These fall into two categories, indirect and direct methods. The field of indirect methods include different variations of Dynamic Programming (DP), e.g. Branicky et al. (1998); Hedlund and Rantzer (2002). While these approaches guarantee a globally optimal trajectory, they also suffer from the 'curse of dimensionality'.

In this paper, direct methods are considered, where local optimality is sufficient. The problem is split into two parts, searching the discrete mode sequence search space, and computing the optimal continuous trajectories for each discrete candidate sequence. Take for example, the Hybrid Maximum Principle (HMP), in which a set of necessary conditions for optimality is posed, Sussmann (1999); Shahid Shaikh and Caines (2007). Within a neighborhood of mode trajectories, a combinatorial algorithm searches for the optimal sequence, and for each mode trajectory, the optimal continuous-time state trajectories are determined using for instance collocation, Benson (2005), or a shooting method, Riedinger et al. (2005). The discrete search space may be modeled using mixed integer constraints. In combination with the continuous dynamics, this results in a monolithic non-convex Mixed Integer Nonlinear Program (MINLP), Avraam et al. (1998); Barton and Lee (2002) or a Generalized Disjunctive Program, Oldenburg and Marquardt (2008).

4.1 Collocation and Integrated CP/NLP Optimization

In collocation, the continuous states and inputs are approximated as polynomials, and a common approximation is to use Lagrange polynomials on the interval $[-1, 1]$ given by

$$L_\ell(t) = \prod_{\substack{i=0 \\ i\neq\ell}}^{N} \frac{t-t_i}{t_\ell - t_i}, \qquad \ell = 0, ..., N.$$

We see that $L_\ell(t_i) = 1$ if $\ell = i$ and $L_\ell(t_i) = 0$ if $\ell \neq i$, and the continuous state vector $x_c(t)$ is approximated by $X_c(t)$ as

$$x_c(t) \approx X_c(t) = \sum_{\ell=0}^{N} L_\ell(t) x_c(t_\ell).$$

where $X_c(t_\ell) = x_c(t_\ell)$. Differentiation yields the approximated continuous state derivatives $\dot{X}_c(t_i) = \sum_{\ell=0}^{N} \dot{L}_\ell(t_i) x_c(t_\ell)$. Now, assume that a hybrid system occupies a mode during an interval $[t_m, t_{m+1}]$, and transform the time scale above from $t \in [-1, 1]$ into $[t_m, t_{m+1}]$. This continuous model approximation can be applied to any nonlinear state space model and optimization criterion. Including a sequence of mode transitions, all state and input variables as well as mode starting times t_m are determined by solving coupled NLPs for multiple modes. Adding different discrete sequences, the optimization of the total hybrid system becomes a MINLP.

Instead of solving the MINLP by traditional relaxation methods, we combine in Wigström and Lennartson (2014) Constraint Programming (CP) for the discrete sequences with NLPs for the continuous dynamics. This integrated approach has mainly been used to solve optimization problems involving CP and MILP, Lombardi and Milano (2012). In our integrated algorithm, decisions in CP are branched upon just like in a branch and bound MINLP algorithm, but instead of solving a relaxation, constraints are propagated in each node until a criterion is satisfied. The good performance of the integrated approach compared to classic MINLP is mainly due to CP's ability to rule out infeasible branches at a much earlier stage. Additionally, if a branch is feasible, the propagation may still speed up the search by inferring and fixating decisions otherwise handled by branching.

Applied to an example where four AGVs are to move through an intersection without collisions, cf. Wigström and Lennartson (2014), the integrated algorithm manage to reduce the number of NLPs by more than a factor 4 compared to an ordinary but efficient MINLP solver. The added time of running the CP routines was less than 10% of the running time for the algorithm. Thus, an integrated CP/NLP approach is a strong candidate for solving MINLP problems that results when hybrid systems are optimized by numerical methods and nonlinear criteria.

This collocation and integrated CP/NLP optimization approach can be directly applied to modular HPNDVs, for instance for energy optimization of coordinated moving devices. An important industrial application is multi robot stations where robots are acting in the same area, Vergnano et al. (2012). The proposed method is recommended when arbitrary but collision free optimal paths are requested. However, when the path is already given, a much simpler method will now be introduced.

4.2 Given Paths for Moving Devices

Consider a set of moving devices, where each device D^k includes a number of joints with joint positions in a vector θ^k, also called the pose. For an industrial robot with six degrees of freedom, $\dim(\theta^k) = 6$. Assume that the pose is given at a number of individual time instances $t_0^k, \ldots, t_N^k$ for each moving device. The pose at time t_ℓ^k for device D^k is denoted θ_ℓ^k, and all poses $\theta_0^k, \ldots, \theta_N^k$ define a path. Given a path for each device, the goal is to determine corresponding velocities and accelerations for all devices such that an overall criterion as minimized. The velocities are estimated as

$$\dot{\theta}_\ell^k = (\theta_{\ell+1}^k - \theta_\ell^k)/(t_{\ell+1}^k - t_\ell^k)$$

In the same way, the accelerations are estimated as

$$\ddot{\theta}_\ell^k = (\dot{\theta}_{\ell+1}^k - \dot{\theta}_\ell^k)/(t_{\ell+1}^k - t_\ell^k)$$

When the joints are moved by servo motors, the joint motor currents are assumed to be expressed by the vector function

$$i_\ell^k = \alpha^k \circ \ddot{\theta}_\ell^k + \beta^k \circ \dot{\theta}_\ell^k + \gamma^k \circ \cos\theta_\ell^k + \delta_\ell^k$$

where $\circ$ = Hadamard (entry-wise) vector multiplication. The parameters in γ^k and δ^k are included to model the effect of gravitational forces. The goal is to minimize the sum of weighted squared currents

$$\sum_{k=1}^{r} \sum_{\ell=0}^{N-1} (w_\ell^k \circ i_\ell^k)^T i_\ell^k \tag{7}$$

which is an approximate normalized measure of the electrical energy consumption, where the vector w_ℓ^k introduces individual weights on the different joints of the moving devices.

Assume that θ_ℓ^k, i_ℓ^k and t_ℓ^k are given for $\ell \in \{0, \ldots, N\}$, either based on physical experiments or simulations. Including the velocity and acceleration estimates, the parameter vectors α^k, β^k, γ^k, and δ_ℓ^k are first determined by least squares estimation for each individual robot path. The time instances $t_1^k, \ldots, t_{N-1}^k$ are then adjusted (decision variables) such that the criterion (7) is minimized, including constraints on maximal permissive velocities, accelerations and jerks (the derivative of the accelerations). Modifying these time instances, where the different poses θ_ℓ^k are passed, means that new velocities and accelerations will be obtained.

This optimization procedure has been applied to industrial robots with a given path and an original equidistant sampling interval $t_{\ell+1}^k - t_\ell^k = \Delta = 12$ ms. Applying this optimization procedure on an individual robot results in an NLP. Furthermore, the resulting trajectory, with adjusted non equidistant sampling instances t_ℓ for the given robot poses, is interpolated and resampled with the original sampling period 12 ms. This modified trajectory is then converted to trajectory control code that is possible to directly execute by KUKA-robots.

A real KUKA robot KC30 is moved from home to fully extended position first based on standard point-to-point robot commands. Based on the resulting path and currents as input, the proposed optimal solution results in about 20% reduction of the energy consumption. The reason for this significant reduction is that the built-in goal to keep constant velocity by ordinary robot commands is replaced by something close to constant jerk in the optimized motion. Observe that this optimization procedure does not require any detailed physical model, involving inertia and damping matrices that are hard to obtain from robot vendors. The proposed optimization only needs a densely sampled path including motor current measurements, which are easily obtained from real robot experiments.

By extending the optimization to two industrial robots with a shared zone modeled as the HPND in Fig. 6, where the poses θ^1 and θ^2 are now six dimensional vectors, some more conditions need to be included. In this paper we focus specifically on the mutual exclusion problem, where the entry and exit points θ^k_{Z-} and θ^k_{Z+} for the two robots are assumed to be given, and the corresponding time variables at these points in the original paths t^k_{Z-} and t^k_{Z+} are identified. In the optimization, the values of these time variables may be changed, but the following condition

$$t^1_{Z+} < t^2_{Z-} \vee t^2_{Z+} < t^1_{Z-} \tag{8}$$

is included to guarantee that either the first robot will exit the zone before the second one enters, or the opposite. The resulting optimization problem is in this case a MINLP, but the only alternative scenarios that need to be evaluated in this problem are the two given by the alternative in (8).

Performing the optimized trajectory on the physical robots results in an energy reduction of about 30%. In this case the reduction is also a result of replacing normal waiting time (to avoid collisions), followed by full speed in the original solution, by slow speed motions and no waiting in the optimized version. The computations for this robot station only take a few seconds, and the energy reduction is impressive. Even for motions up to 30 seconds (2500 time points), the computation takes roughly about a minute.

5. CONCLUSIONS

A recently proposed predicate transition model for discrete event systems has been generalized in this paper to include continuous dynamics, resulting in a hybrid predicate transition model (HPTM). To obtain a hybrid model involving graphical elements, either hybrid automata or hybrid Petri nets can be adapted to the HPTM. In the latter case we propose a hybrid model where both explicit differential equations are added to discrete places, and shared discrete variables are introduced to simplify the graphical part of the discrete logics. Special conditions have been introduced for such hybrid PNs to guarantee that a complete and nonconflicting continuous model is formulated. Based on the proposed hybrid PN, a simple optimization procedure has also been formulated and applied to energy optimization of real industrial robots. Applying this routine on interacting robots shows that the energy consumption can be reduced up to 30%.

REFERENCES

Alur, R., Courcoubetis, C., Henzinger, T., and Ho, P. (1993). Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In *Lecture Notes in Computer Science*, volume 736, 209–229. Springer-Verlag.

Avraam, M., Shah, N., and Pantelides, C. (1998). Modelling and optimisation of general hybrid systems in the continuous time domain. *Computers & chemical engineering*, 22, 221–228.

Barton, P.I. and Lee, C.K. (2002). Modeling, simulation, sensitivity analysis, and optimization of hybrid systems. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 12(4), 256–289.

Benson, D. (2005). *A Gauss pseudospectral transcription for optimal control*. Ph.D. thesis, Massachusetts Institute of Technology.

Branicky, M., Borkar, V., and Mitter, S. (1998). A unified framework for hybrid control: model and optimal control theory. *Automatic Control, IEEE Transactions on*, 43(1), 31–45.

David, R. and Alla, H. (2001). On hybrid Petri nets. *Discrete Event Dynamic Systems*, 11, 9–40.

Demongodin, I. and Koussoulas, N.T. (2006). Differential Petri net models for industrial automation and supervisory control. *IEEE Transactions on Systems, Man, and Cybernetics – Part C*, 36(4), 543–553.

Hedlund, S. and Rantzer, A. (2002). Convex dynamic programming for hybrid systems. *Automatic Control, IEEE Transactions on*, 47(9), 1536–1540.

Hoare, C.A.R. (1978). *Communicating sequential processes*, volume 21 of *Series in Computer Science*. ACM.

Lennartson, B., Basile, F., Miremadi, S., Fei, Z., Hosseini, M.N., Fabian, M., and Åkesson, K. (2014a). Supervisory Control for State-Vector Transition Models - A Unified Approach. *IEEE Transaction on Automation Science and Engineering*, 11(1), 33–47.

Lennartson, B., Wigström, O., Fabian, M., and Basile, F. (2014b). Unified model for synthesis and optimization of discrete event and hybrid systems. In *12th IFAC - IEEE International Workshop on Discrete Event Systems (WODES'14)*.

Lombardi, M. and Milano, M. (2012). Optimal methods for resource allocation and scheduling: a cross-disciplinary survey. *Constraints*, 17, 51–85.

Oldenburg, J. and Marquardt, W. (2008). Disjunctive modeling for optimal control of hybrid systems. *Computers and Chemical Engineering*, 32(10), 2346 – 2364.

Pettersson, S. and Lennartson, B. (1995). Hybrid modeling focused on hybrid Petri nets. In *Proc. 2nd European Workshop on Real-time and Hybrid systems*, 303–309.

Riedinger, P., Daafouz, J., and Iung, C. (2005). About solving hybrid optimal control problems. *IMACS05*.

Shahid Shaikh, M. and Caines, P. (2007). On the hybrid optimal control problem: Theory and algorithms. *Automatic Control, IEEE Transactions on*, 52(9), 1587–1603.

Sousa, J.R.B. and Lima, A.M.N. (2008). Modeling and simulation of systems of multiple sources of energy by differential hybrid Petri nets. In *Proc. IEEE International Symposium on Industrial Electronics (ISIE 2008)*, 1861–1866.

Sussmann, H.J. (1999). A maximum principle for hybrid optimal control problems. In *Decision and Control, 1999. Proceedings of the 38th IEEE Conference on*, volume 1, 425–430.

Vergnano, A., Thorstensson, C., Lennartson, B., M., P.F., Pellicciari, Leali, F., and Biller, S. (2012). Modeling and Optimization of Energy Consumption in Cooperative Multi-Robot Systems. *IEEE Transaction on Automation Science and Engineering*, 9(2), 423–428.

Villani, E., Miyagi, P.E., and Valette, R. (2007). *Modelling and Analysis of Hybrid Supervisory Systems – A Petri Net Approach*. Advances in Industrial Control. Springer-Verlag.

Wigström, O. and Lennartson, B. (2014). An integrated CP/OR method for optimal control of modular hybrid systems. In *12th IFAC - IEEE International Workshop on Discrete Event Systems (WODES'14)*.