SYNTHESIS OF SUPERVISORY CONTROLLERS FOR HYBRID SYSTEMS USING ABSTRACTION REFINEMENT

Olaf Stursberg

Process Control Laboratory (BCI-AST)
University of Dortmund, D-44221 Dortmund (Germany)
Email: olaf.stursberg@uni-dortmund.de

Abstract: For the computationally challenging task of synthesizing supervisory controllers for hybrid systems, this paper suggests to use guided abstraction refinement. Starting from a discrete abstraction of the hybrid model, so-called *candidate paths* are determined as possible controlled evolutions which satisfy given specifications for safety and goal attainment. A specialized validation procedure is used to check if the corresponding control strategies satisfy the specifications also for the original hybrid system. If not, the abstract model is refined according to the validation result. Since different validation methods (with different computational costs) are used, reachable hybrid sets have to be computed only for relatively few combinations of control inputs and locations of the hybrid automaton.

Keywords: Abstraction, Automaton, Logic Control, Reachability, Synthesis.

1. INTRODUCTION

This paper addresses the task of computing supervisory control strategies for hybrid systems with nonlinear continuous dynamics. Supervisory control is here understood as the closed-loop setting, in which the hybrid plant generates an event when its continuous state reaches an event set, and the controller immediately returns a discrete control action which determines the further evolution of the hybrid plant. A number of approaches considering this task have been published recently, see e.g. (Asarin *et al.*, 2000; Gonzalez *et al.*, 2001; Koutsoukos *et al.*, 2000; Koo *et al.*, 2001). Most of these approaches establish extensions of the supervisory control theory for discrete systems (Ramadge and Wonham, 1989) to hybrid models, and involve the step of determining which hybrid states are reachable under the influence of a discrete control action. Techniques to accomplish this step include level set methods (Trontis and Spathopoulos, 2003), viability methods (Bayen *et al.*, 2002), and computing ellipsoidal or polyhedral inclusions of reach sets (Stursberg and Krogh, 2003). However, these techniques are known to be computationally expensive, and to scale badly with the dimension of the continuous state space – an objective for improving the efficiency of controller synthesis is thus to reduce reach set computation as much as possible.

For this reason, a synthesis approach employing the concept of guided abstraction refinement is proposed here. As described in the context of verification (Clarke *et al.*, 2003), the principle is to use a discrete abstraction to search for evolutions (*counterexamples*) which violate a given property. Only for these evolutions the hybrid dynamics is investigated, in order to validate that the property is not satisfied for the hybrid model. As described in (Stursberg *et al.*, 2003), applying a set of different validation methods with different computational costs can lead to the advantage that reach set computation is required only for a small portion of the state space. If the steps of computing counterexamples, validation, and re-

fining the abstract model based on the validation result are applied iteratively, the analysis task can often be solved with a relatively low overall effort.

The concept is modified here for the purpose of controller synthesis: a discrete abstraction of the hybrid model is searched for *candidate paths* which encode control strategies that potentially fulfill safety and goal attainment specifications for the controlled system. The validation methods check if it is guaranteed that the hybrid evolutions, which correspond to the candidate path, meet the specifications. Also here, the outcome of the validation step is used to refine the abstract model (and thus affects the search for further candidate paths).

The method presented here is similar to the concept of *l-complete abstractions*, as described in (Moor *et al.*, 2002). However, the type of abstraction as well as the methods for validation and refinement follow different principles.

2. PROBLEM STATEMENT AND MODEL TRANSFORMATION

The investigation considers the following type of hybrid automaton with discrete inputs:

Def. 1: A hybrid automaton with discrete inputs is given by $HA = (X, Z, V, \gamma, inv, \Theta, g, r, f)$ with:

- the *continuous state space* $X \subseteq \mathbb{R}^{n_x}$ and $x \in X$;
- the finite set of *locations* $Z = \{z_1, \ldots, z_{n_z}\}$;
- the finite set $V = \{v_1, \ldots, v_{n_d}\}$ of *discrete inputs* $v_j \in \mathbb{R}^{n_v}$; an *input availability* mapping $\gamma : Z \to 2^V \setminus \emptyset$ assigns a set $V_z \subseteq V$ to $z \in Z$;
- the mapping $inv : Z \to 2^X$ which assigns a *polyhedral invariant* $inv(z_j) = \{x \mid \exists\ n_{p_j} \in \mathbb{N}, C_j \in \mathbb{R}^{n_{p_j} \times n_x}, d_j \in \mathbb{R}^{n_{p_j}}, x \in X : C_j \cdot x \leq d_j\}$ to each $z_j \in Z$;
- the finite *transition set* $\Theta \subseteq Z \times Z$; a transition from $z_1 \in Z$ to $z_2 \in Z$ is denoted by (z_1, z_2);
- a mapping $g : \Theta \to 2^X$ that associates a *guard* $g((z_1, z_2)) \subseteq X$ with each $(z_1, z_2) \in \Theta$ such that: $g((z_1, z_2)_j) = \{x \mid \exists\ n_{g_j} \in \mathbb{N}, C_j \in \mathbb{R}^{n_{g_j} \times n_x}, d_j \in \mathbb{R}^{n_{g_j}} : C_j \cdot x \leq d_j\}$. For all pairs of transitions $(z_1, \bullet) \in \Theta$, it is required that the corresponding guards are disjoint;
- a linear *reset function* $r : \Theta \times X \to X$, assigning $x' \in X$ to each $(z_1, z_2) \in \Theta$ and $x \in g((z_1, z_2))$;
- a *flow function* $f : Z \times X \times V \to \mathbb{R}^n$ that defines a continuous vector field $\dot{x} = f(z, x, v)$ for each $z \in Z$.

The states $x_k := x(t_k)$, locations $z_k := z(t_k)$, and inputs $v_k = v(t_k)$ are defined on an ordered time set $T = \{t_0, t_1, t_2, \ldots\}$, $t_k \in \mathbb{R}^{\geq 0}$, which contains the initial time t_0 and all times at which transitions occur. For a given series of inputs v_k, a *feasible run* of HA is a sequence $\phi_\sigma = (\sigma(t_0), \sigma(t_1), \sigma(t_2), \ldots)$ of hybrid states $\sigma_k := \sigma(t_k) = (z_k, x_k)$ according to:

- $\sigma(t_0) = (z_0, x_0)$ with $z_0 = z(t_0) \in Z$ and $x_0 \in inv(z_0)$, $x_0 \notin g((z_0, \bullet))$ for any $(z_0, \bullet) \in \Theta$.
- $\sigma(t_{k+1})$ follows from $\sigma(t_k)$ by:
  - (i) continuous evolution: $\chi : [0, \tau] \to X$, $\tau \in \mathbb{R}^{>0}$, where $\chi(t_0) = x_k$, $\dot{\chi}(t) = f(z_k, \chi(t), v_k)$ with an existing unique solution for $t \in [0, \tau]$, and $\chi(t) \in inv(z_k)$, but for all $g((z_k, \bullet)) \in \Theta$ and $t \in [0, \tau[$: if $\chi(t) \in g((z_k, \bullet))$, no t' with $t < t' \leq \tau$ exists such that $\chi(t') \notin g((z_k, \bullet))$;
  - (ii) discrete transition: $(z_k, z_{k+1}) \in \Theta$, $\chi(\tau) \in g(z_k, z_{k+1})$, and $x_{k+1} = r((z_k, z_{k+1}), \chi(\tau)) \in inv(z_{k+1})$;
  - (iii) input selection: $v_{k+1} \in V_{z_{k+1}}$. △

Two characteristics of this semantics are important: (a) A transition can occur for an arbitrary continuous state $\chi(\tau)$ in $g((z_k, \bullet))$, but it must be taken before the guard set is left. (This construction is suitable, e.g., to model by $g((z_k, \bullet))$ that the state which triggers the transition (a threshold) can only be measured with a limited precision.) (b) The discrete input v_k can be changed with a transition, but is constant otherwise.

The control task for this setting is defined as follows:

Def. 2: Given are:

- an initial state set Σ_0 with $(z_0, x_0) \in \Sigma_0$, $z_0 \in Z$, $x_0 \in X_0$ with a bounded polyhedral set $X_0 \subset inv(z_0)$ that is disjoint with all $g((z_0, \bullet))$,
- a set $\Sigma_F = \{\Sigma_{F,1}, \ldots, \Sigma_{F,n_F}\}$ of forbidden state sets $\Sigma_{F,j}$ with $(z, x) \in \Sigma_{F,j}$, such that $z \in Z$, $x \in X_{F,j}$ with a bounded polyhedron $X_{F,j} \subset inv(z_{F,j})$, and $\Sigma_{F,j} \cap \Sigma_0 = \emptyset$,
- and a goal set Σ_G, $(z_G, x) \in \Sigma_G$ with $z_G \in Z$, $x \in inv(z_G)$, and $\Sigma_G \cup \Sigma_F = \emptyset$.

The synthesis task is to find a control strategy $(v_0, v_1, v_2, \ldots)$ for HA such that the corresponding feasible run starting from every $(z_0, x_0) \in \Sigma_0$ leads into Σ_G (goal attainment), while no state $(z, x) \in \Sigma_{F,j}$, $\Sigma_{F,j} \in \Sigma_F$ is encountered (safety property). △

To simplify the further description, the open system HA is rewritten as a closed system that includes all possible control strategies: Since any $v_k \in V_z$ can possibly be applied in $z \in Z$, the closed system is defined with an extended location set containing pairs (z, v) with $v \in V_z$:

Def. 3: For HA as in Def. 1, the corresponding closed hybrid automaton is defined by: $HA^c = (X^c, Z^c, inv^c, \Theta^c, g^c, r^c, f^c)$ with:

- $X^c = X$;
- $Z^c = \{z_1^c; \ldots, z_{n_z^c}^c\}$ where for each $z \in Z$: $\forall\ v \in V_z : \exists\ z^c = (z, v) \in Z^c$;

- $inv^c : Z^c \rightarrow 2^X$ with $inv^c(z^c) = inv(z)$ for $z^c = (z, \bullet)$;
- $\Theta^c \subseteq Z^c \times Z^c$ such that $(z_1^c, z_2^c) \in \Theta^c$, $z_1^c = (z_1, \bullet)$, $z_2^c = (z_2, \bullet)$ if $\exists\ (z_1, z_2) \in \Theta$;
- $g^c : \Theta^c \rightarrow 2^X$ with $g^c((z_1^c, z_2^c)) = g((z_1, z_2))$ for $z_1^c = (z_1, \bullet)$, $z_2^c = (z_2, \bullet)$;
- $r^c : \Theta^c \times X \rightarrow X$ with $r^c((z_1^c, z_2^c), x) = r((z_1, z_2), x)$ for $z_1^c = (z_1, \bullet)$, $z_2^c = (z_2, \bullet)$;
- $f^c : Z^c \times X \rightarrow \mathbb{R}^n$ with $f^c(z^c, x) = f(z, x, v)$ for $z^c = (z, v)$, $v \in V_z$. △

The semantics for HA^c with extended hybrid states (z^c, x) is obvious from Def. 1. The control task according to Def. 3, is reformulated for HA^c as follows: The initial set is changed to Σ_0^c with $z_0^c = (z_0, v) \in \Sigma_0^c$, $v \in V_{z_0}$, $x_0 \in X_0 \subset inv(z_0)$. The goal set is modified to Σ_G^c with $z_G^c = (z_G, v) \in \Sigma_G^c$, $v \in V_{z_G}$, $x \in inv(z_g)$, and the forbidden set Σ_F^c is altered correspondingly. The sets of locations which involve initial, goal, or forbidden states, are denoted by Z_0^c, Z_G^c, and Z_F^c respectively. The control task renders then to transfer all states of Σ_0^c with $z_0^c = (z_0, v)$ for one(!) $v \in V_{z_0}$ into Σ_G^c while Σ_F^c is never encountered.

The closed hybrid automaton is referred to as *concrete model* C, and to simplify notation, the index 'c' is omitted for $inv^c, \Theta^c, g^c, r^c$, and f^c.

3. SYNTHESIS BASED ON ABSTRACTION REFINEMENT

The scheme of the procedure to synthesize a control strategy for C is shown in Fig. 1, and can be summarized as follows: First, an abstract model $A^{(0)}$ is obtained from C by an appropriate abstraction function (see Sec. 3.1). The latter essentially transfers the discrete dynamics of C to $A^{(0)}$, abstracts from the continuous part, and accounts for the state sets which are relevant for the specification $(\Sigma_0^c, \Sigma_G^c, \Sigma_F^c)$. The next step searches for a *candidate path* CP within the evolutions of $A^{(0)}$. A candidate path is a run of $A^{(0)}$ which leads from an abstract initial state to an abstract goal state, while not encountering an abstract forbidden state (Sec. 3.2). Since each abstract state in a candidate path corresponds to a location $z^c = (z, v)$ of C, it implicitly represents a control strategy $(v_0, v_1, v_2, \ldots)$. The following step of *validation* determines whether CP can be realized for C in the sense that every hybrid state reached along the corresponding abstract states in CP is eventually transferred into the goal set. As further explained in Sec. 3.3, the validation comprises the application of three different methods with different computational costs in order to reduce the overall effort. If CP is validated, a possible control strategy is found, and the procedure terminates (or can be resumed to search for another alternative strategy). If CP is found to be invalid, the

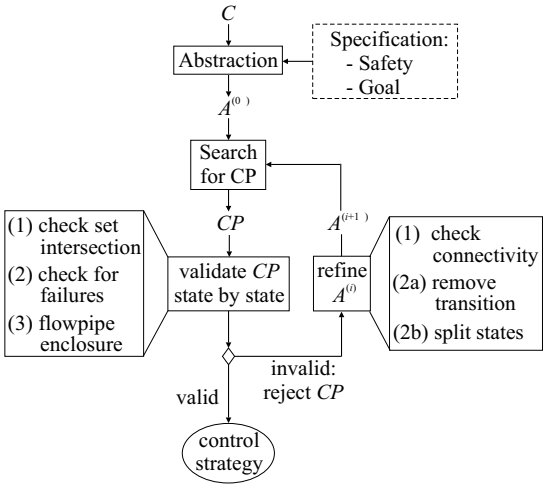

Fig. 1. The synthesis procedure.

strategy is rejected, and the procedure continues with a refinement step (Sec. 3.4). It first checks, if for the particular state of CP, in which the latter was found to be invalid, any evolution of C which corresponds to the transition in $A^{(i)}$ exists at all (*connectivity check*). If not, the transition is removed from $A^{(i)}$ (i is the iteration index). An additional optional means to update $A^{(i)}$ is to split states according to the validation result, if the third validation method (*flowpipe enclosure*) was applied. It adds details to $A^{(i)}$ in the sense that reachability information obtained from evaluating C is transferred to $A^{(i+1)}$.

The updated abstract model is searched for another candidate path, and the cycle of searching, validation, and refinement continues, until a suitable control strategy is found, no further candidate path can be determined, or an upper bound on the number of iterations is exceeded.

3.1 *Abstraction*

The abstraction step resembles the one used in (Clarke *et al.*, 2003) in the context of verification, but with modifications to consider Σ_0^c, Σ_G^c, and Σ_F^c. First, a transition system is introduced according to $TS = (\hat{S}, \hat{S}_0, \hat{E})$, consisting of a finite state set $\hat{S}$, an initial set $\hat{S}_0 \subset \hat{S}$, and a finite transition set $\hat{E} \subseteq \hat{S} \times \hat{S}$. A *run* of TS is given by $\hat{\phi} = (\hat{s}_0, \hat{s}_1, \hat{s}_2, \ldots)$ with $\hat{s}_i \in \hat{S}$ and $(\hat{s}_{i-1}, \hat{s}_i) \in \hat{E}$.

In order to obtain $A^{(0)} = (\hat{S}, \hat{S}_0, \hat{E})$ as a model of the type TS such it represents an abstraction of C, an abstraction function $\alpha : Z \times X \rightarrow \hat{S}$ is defined as follows:

(1) A state $\hat{s}_0 \in \hat{S}_0$ is introduced for any $z^c \in Z_0^c$, and $\hat{s}_0$ represents all hybrid states (z^c, x) of C with $z^c \in Z_0^c$ and $x_0 \in X_0$.
(2) $\hat{S}_F$ includes a state $\hat{s}_F$ for any $z^c \in Z_F^c$ such that it corresponds to all (z^c, x) with $z^c \in Z_{F,j}^c$ and $x \in X_{F,j}$

(3) $\hat{S}$ follows from $\hat{S} := \hat{S}' \cup \hat{S}_0 \cup \hat{S}_F$ with $\hat{S}'$ such that $\exists\ \hat{s} \in \hat{S}'$ for any $z^c \in Z^c$; $\hat{s} \in \hat{S}'$ represents the states (z^c, x) with: $x \in inv(z^c)$ if $z^c \notin Z^c_F$, or $x \in inv(z^c) \setminus X_{F,j}$ if $z^c \in Z^c_F$.

The set of abstract goal states $\hat{S}_G$ contains the states $\hat{s} \in \hat{S}$ which are assigned to $z^c_G \in Z^c_G$. The set $\hat{E}$ of A contains transitions $(\hat{s}_i, \hat{s}_l)$ for any of the following cases with $\hat{s}_i = \alpha(z^c_i, x_i)$, $\hat{s}_l = \alpha(z^c_l, x_l)$:

- $\exists\ (z^c_i, z^c_l) \in \Theta^c$,
- $\hat{s}_i \in \hat{S}_0$ and $z^c_i = z^c_l \in Z^c_0$,
- $\hat{s}_l \in \hat{S}_F$, $z^c_i = z^c_l \in Z^c_F$, and $x_i \in inv(z_{F,j}) \setminus X_{F,j}$, $x_l \in X_{F,j}$.

These assignments map every hybrid state of C onto at least one discrete state of A, and each transition of C has a correspondence in A. Thus, also every run of C corresponds to a run of A, such that the latter is an abstraction of the hybrid model.

3.2 *Search for a Candidate Path*

A candidate path can now be defined as a particular run of A:

Def. 4: Given the model A with the sets $\hat{S}_0$, $\hat{S}_F$, and $\hat{S}_G$, a *candidate path* of A is a run $CP = (\hat{s}_0, \hat{s}_1, \ldots, \hat{s}_p)$ with $\hat{s}_0 \in \hat{S}_0$, $\hat{s}_p \in \hat{S}_G$, and $\hat{s}_j \notin \hat{S}_F$ for all $j \in \{0, 1, \ldots, p\}$. △

To determine CP, a standard reachability algorithm using breadth-first search is applied. It returns one of the shortest candidate paths existing for the current abstract model in each iteration.

3.3 *Validation*

The validation aims to check for every pair $(\hat{s}_k, \hat{s}_{k+1})$ of subsequent states of CP (and the intermittent transition $(\hat{s}_k, \hat{s}_{k+1}) \in \hat{E}$) if a corresponding behavior of C exists in the following sense: In Fig. 2, let the set $I \subseteq inv(z^c_k)$ mark the set of hybrid states represented by $\hat{s}_k$, and $inv(z^c_{k+1})$ the set represented by $\hat{s}_{k+1}$. In order to validate the pair $(\hat{s}_k, \hat{s}_{k+1})$ of CP as being a feasible step of a control strategy, each continuous state $x \in I$ must be transformed into $inv(z^c_{k+1})$. This requires that the continuous dynamics $f(z^c_k, x)$ must first transfer $x \in I$ into a state x' in the guard set $g((z^c_k, z^c_{k+1}))$, and applying the reset $x'' = r((z^c_k, z^c_{k+1}, x'))$ must yield a state $x'' \in inv(z^c_{k+1})$. This refers to the case marked by (b) in Fig. 2; if the case (a) occurs, i.e. the trajectory does, e.g., not enter the guard set, then the step $(\hat{s}_k, \hat{s}_{k+1})$ of CP does not encode a feasible control strategy.

The validation scheme follows the objective to determine with an as small as possible effort whether

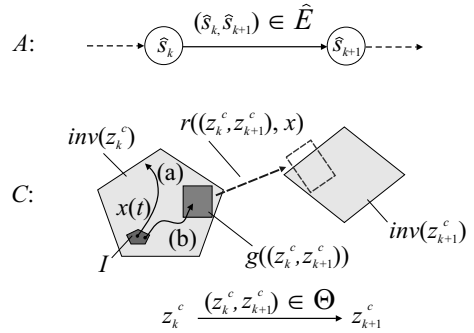

Fig. 2. Validating a transition of A for C.

a candidate path does not encode a feasible control strategy. Thus, a series of validation methods with an increasing effort to reject a candidate path is applied, but the methods are increasingly successful in invalidating a transition of CP.

The first method, referred to as *set intersection* in Fig. 1, checks the necessary condition that $r((z^c_k, z^c_{k+1}, x))$ maps at least one $x \in g((z^c_k, z^c_{k+1}))$ into $inv(z^c_{k+1})$. For a linear reset function and a polyhedral guard set (see Def. 1), this check can be carried out relatively easy by a linear transformation of $g((z^c_k, z^c_{k+1}))$ and by intersecting the result with the polyhedron $inv(z^c_{k+1})$.

The second validation method (listed as *failure check* in Fig. 1), represents the key step in efficiently rejecting an invalid candidate path. It poses an optimization problem that searches for a trajectory which starts in I, but does not eventually lead to $inv(z^c_{k+1})$ through the transition $(z^c_k, z^c_{k+1}) \in \Theta$. Let $T = \{x' \mid x'' \in inv(z^c_{k+1}) : r^{-1}(x'') = ((z^c_k, z^c_{k+1}), x'), x' \in g((z^c_k, z^c_{k+1}))\}$ denote a *target set*, as the subset of the guard which is mapped into $inv(z^c_{k+1})$ by r. For each continuous trajectory starting in $x_0 \in I$, the terminal state is defined as $x(t_f) = x_0 + \int_{t_0}^{t_f} f(z^c_k, x(\tau))d\tau$ such that one of the following criteria applies:

(1) $x(t_f) \in T$,
(2) $x(t_f) \in g((z^c_k, z^c_q))$ with $z^c_q \notin z^c_{k+1}$,
(3) $x(t_f) \in X_{F,j}$ for any F_j,
(4) $x(t_f) \notin inv(z^c_k)$, but $\lim_{t \to t_f} x(\tau) \in inv(z^c_k)$
(5) $x(t_f) \in inv(z^c_k)$ and $t \geq t_{max}$.

Additionally it is required that for $\tau \in [0, t_f[$ none of the sets in (1) to (3) is hit by $x(\tau)$. The parameter $t_{max} \in \mathbb{R}^{>0}$ denotes an upper time bound. Essentially, (1) refers to the desired case that $inv(z^c_{k+1})$ is reached, while the four other case render $(\hat{s}_k, \hat{s}_{k+1})$ invalid, since either a different guard set, a forbidden region, or the border of the invariant is reached, or t_{max} is elapsed. To determine if any of the cases (2) to (5) applies for any $x_0 \in I$, the following optimization problem is posed:

$$\max_{x_0 \in I} \|x(t_f) - x_{cent}(g((z^c_k, z^c_{k+1})))\|_2, \qquad (1)$$

with $x_{cent}(g((z_k^c, z_{k+1}^c)))$ denoting the center point of the guard set. The optimization is solved by nonlinear programming, in which the simulation of the continuous dynamics of C is embedded, and the simulation terminates according to the criteria for $x(t_f)$ as given above.

If the optimization according to Eq. 1 leads to the result that CP has to be rejected, the synthesis procedure continues with refinement (Sec. 3.4). If the validation was successful, the method *flowpipe enclosure* is applied to the validated step of CP. This method for computing reachable sets, which is in detail described in (Stursberg and Krogh, 2003), generates a sequence of hyper-rectangular polyhedra which completely enclose all continuous states in $inv(z_k^c)$ that can be reached from I by continuous evolution. The reason for applying this method is that the entry set for the location z_{k+1}^c can be obtained from intersecting the enclosing polyhedra with $g((z_k^c, z_{k+1}^c)$, and applying the reset function $r((z_k^c, z_{k+1}^c, x'))$ afterwards. Thus, this step produces the starting point for validating the next transition of CP.

3.4 *Refinement*

Three different situations have to be considered for refining $A^{(i)}$ in order to obtain an updated abstract model $A^{(i+1)}$:

(1.) If the method of checking for set intersection revealed that, for a transition $(\hat{s}_k, \hat{s}_{k+1})$ of CP, the corresponding transition $(z_k^c, z_{k+1}^c) \in \Theta$ of C can never be taken, $(\hat{s}_k, \hat{s}_{k+1})$ is removed from the set $\hat{E}$ of $A^{(i)}$.

(2.) If the method *failure check* returned the result that the transition $(\hat{s}_k, \hat{s}_{k+1})$ of $A^{(i)}$ is invalid, the transition cannot not be removed immediately from $\hat{E}$, since a smaller entry set into $inv(z_k^c)$ (through a different path) may lead to a positive validation result. However, if no continuous trajectory corresponding to $(\hat{s}_k, \hat{s}_{k+1})$ can be found, this transition is eliminated from $\hat{E}$. The investigation of the non-existence of such a corresponding trajectory (named *connectivity check* in Fig. 1) can be numerically performed by the optimization described in (Stursberg *et al.*, 2003). If a connecting trajectory is found but $(\hat{s}_k, \hat{s}_{k+1})$ of $A^{(i)}$ is invalid, CP is marked as *rejected* to exclude it from the further investigation.

(3.) If $(\hat{s}_k, \hat{s}_{k+1})$ has been validated, and the method *flowpipe enclosure* has returned an entry set I for the location z_{k+1}^c, refinement of $A^{(i)}$ by state splitting can be performed optionally. This means that the state $\hat{s}_{k+1}$ is replaced in $\hat{E}$ by two states $\hat{s}_a$ and $\hat{s}_b$, where $\hat{s}_a$ represents all hybrid states with z_{k+1}^c and $x \in I$, and $\hat{s}_b$ corresponds to all hybrid states with z_{k+1}^c and $x \in inv(z_{k+1}^c) \setminus I$. $\hat{s}_a$ has the same ingoing and outgoing transitions as $\hat{s}_{k+1}$. Also for $\hat{s}_b$, the set of outgoing transitions is identical to the one of $\hat{s}_{k+1}$, but the set of ingoing transitions is reduced by the transition from $\hat{s}_k$, corresponding to the result of the flowpipe computation. (The refinement function α is modified accordingly.)

4. APPLICATION TO A REACTOR SYSTEM

To illustrate the approach, it is applied to the startup procedure of a continuous chemical liquid-phase reactor. The system comprises a stirred tank with two inlets, one outlet, a cooling jacket, and a heating device. An exothermic reaction of two components D_1 and D_2 form a product D_3, and D_2 is supplied in excess. The task is to find a control strategy that drives the reactor from a state of being almost empty and at low temperature to nominal operation (filled, high temperature, and high yield). The relevant state variables are the liquid level (x_1), the temperature of the mixture (x_2), and the concentration (x_3) of component D_1. A vector $v := (F_2, F_3, K, H)^{\mathrm{T}}$ denotes the four discrete inputs: the valve settings for the inlet of component D_2 (F_2) and for the outlet (F_3), as well as the status of the cooling (K) and heating device (H). Each input can be switched between two values (low/high or on/off) such that 16 different input combinations are available. The hybrid dynamics is modelled by:

$$\dot{x}_1 = k_1 + F_2 - F_3, \tag{2}$$

$$\dot{x}_2 = (k_2(k_3 - x_2) + F_2(k_4 - x_2))/x_1 + k_5(k_6 - x_2)(k_7/x_1 + k_8)K \tag{3}$$

$$\dot{x}_3 = (k_9 - (k_{10} + F_2)x_3)/x_1 \tag{4}$$

but for the locations with $x_1 \geq 0.8$ an additional term for $\dot{x}_2$ has to be considered:

$$\dot{x}_2' = \dot{x}_2 + k_{11}(k_{12} - x_2)(k_{13} - k_{14}/x_1)H \tag{5}$$

and for locations with $(0, k_{15}, k_{16}) \cdot x \geq k_{17}$ the following reaction terms are relevant:

$$\dot{x}_2'' = \dot{x}_2' + x_3(k_{18} + k_{19}x_2^2) \tag{6}$$

$$\dot{x}_3' = \dot{x}_3 + k_{20} \cdot \exp(k_{21}/x_2) \tag{7}$$

(with constant parameters k_i). The 3-dim. continuous state space is additionally partitioned by a second threshold for x_1, and two more thresholds for x_2. The invariant and guard sets of the hybrid automaton follow from this partitioning. The initial hybrid model HA comprises 12 locations, 22 transitions, and seven forbidden sets (representing too high temperatures and liquid levels). The available input sets V_z for each location z are restricted to those inputs that are physically reasonable. Thus, the set Z^c of locations of HA^c contains 32 locations.

When applying the synthesis procedure proposed in this paper, a feasible control strategy is found by the 17^{th} candidate path. The over-approximated continuous reachable state set that for this strategy is shown in Fig. 3, and obviously all trajectories emerging from the initial set eventually lead to the goal set. The figure indicates six different phases (p_1 to p_6) , each of which refers to a period of time with constant discrete input. The input for this strategy are: $v_1 = v_2 = (1,0,0,0)^{\mathrm{T}}$ (i.e., high value for F_2, F_3 closed, cooling and heating switched off), $v_3 = v_4 = (0,1,0,1)^{\mathrm{T}}$ (low value for F_2, F_3 open, cooling off, heating on), $v_5 = (0,1,0,0)^{\mathrm{T}}$, and $v_6 = (0,1,1,0)^{\mathrm{T}}$. The first 16 candidate path were invalidated by the *failure check* method. The synthesis result was obtained in approx. 4 minutes on a PC with P-4 processor (1.5 GHz). It has to be mentioned that the computation time and the number of candidate paths rejected before finding a feasible solution does largely depend on the ordering of the sets V_z. (If, e.g., this order is rearranged such that above control strategy corresponds to the first investigated candidate path, the algorithm terminates already after 40 seconds.)

5. CONCLUSION

The proposed approach has the following advantages in comparison to methods which explore the hybrid state space in a breadth-first style by applying each available input in each location (starting from z_0): (a) The search is restricted to the paths for which the discrete dynamics signals success, and the search is performed in a depth-first style. (b) The validation method *failure check* was found to be very efficient in rejecting invalid strategies. (c) The refinement of A by eliminating transitions can exclude several further invalid candidate paths. An open point at this stage, and thus matter of current investigation, is whether the refinement by state splitting is indeed beneficial. It is also investigated if efficient heuristics can be developed for the order of choosing the discrete inputs (during the search for CP), depending on the outcome of preceding validations.

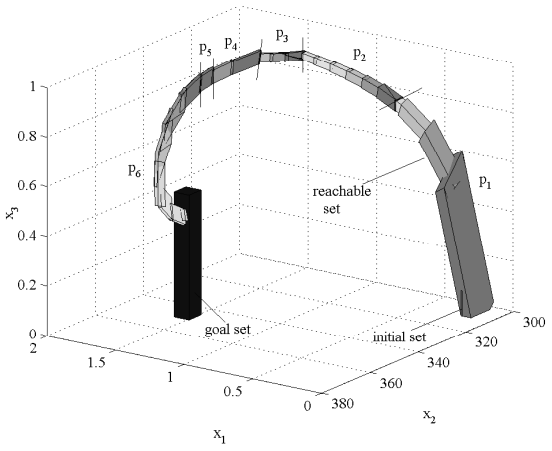

Fig. 3. Reachable continuous state set for the synthesized control strategy.

6. REFERENCES

Asarin, E., O. Bournez, T. Dang, O. Maler and A. Pnueli (2000). Effective synthesis of switching controllers for linear systems. *Proc. of the IEEE* **88**(7), 1011–1025.

Bayen, A.M., E. Crueck and C. Tomlin (2002). Guaranteed overapproximations of unsafe sets for continuous and hybrid systems. In: *Hybrid Systems - Comp. and Control*. Vol. 2289 of *LNCS*. Springer. pp. 90–104.

Clarke, E.M., A. Fehnker, Z. Han, B.H. Krogh, O. Stursberg and M. Theobald (2003). Verification of hybrid systems based on counterexample-guided abstraction refinement. In: *Tools and Algor. for the Constr. and Analysis of Systems*. Vol. 2619 of *LNCS*. Springer. pp. 192–207.

Gonzalez, J.M.E., A.E.C Cunha J.E.R. Cury and B.H. Krogh (2001). Supervision of event-driven hybrid systems: Modeling and synthesis. In: *Hybrid Systems - Comp. and Control*. Vol. 2034 of *LNCS*. Springer. pp. 247–260.

Koo, T.-J., G.J. Pappas and S. Sastry (2001). Mode switching synthesis for reachability specifications. In: *Hybrid Systems - Comp. and Control*. Vol. 2034 of *LNCS*. Springer. pp. 333–346.

Koutsoukos, X., P.J. Antsaklis, J.A. Stiver and M.D. Lemmon (2000). Supervisory control of hybrid systems. *Proc. of the IEEE* **88**(7), 1026–1049.

Moor, T., J.M. Davoren and J. Raisch (2002). Strategic refinements in abstraction based supervisory control of hybrid systems. In: *Proc. 6th Int. Workshop on Discrete Event Systems*. pp. 329–334.

Ramadge, P.J.G. and W.M. Wonham (1989). The control of discrete event systems. *Proc. of the IEEE* **77**(1), 81–98.

Stursberg, O., A. Fehnker, Z. Han and B.H. Krogh (2003). Specification-guided analysis of hybrid systems using a hierarchy of validation methods. In: *Proc. 15th IFAC Conf. Analysis and Design of Hybrid Sys.*. pp. 289–295.

Stursberg, O. and B.H. Krogh (2003). Efficient representation and computation of reachable sets for hybrid systems. In: *Hybrid Systems - Comp. and Control*. Vol. 2623 of *LNCS*. Springer. pp. 482–497.

Trontis, A. and M.P. Spathopoulos (2003). Hybrid control synthesis for eventuality specifications using level set methods. *Int. J. of Control* **76**(16), 1599–1627.