Towards Plan Recognition in Hybrid Systems*

Diego Aineto[1], Eva Onaindia[1], Miquel Ramírez[2], Enrico Scala[3], Ivan Serina[3]

[1] VRAIN, Universitat Politècnica de València
{dieaigar,onaindia}@upv.es
[2] Univesity of Melbourne
miquel.ramirezunimelb.edu.au
[3] Università degli Studi di Brescia, Italy
{enrico.scala,ivan.serina}@unibs.it

Abstract. Hybrid systems model software-based control systems to deal with real-world applications that exhibit an interaction of discrete and continuous dynamic behaviour. Hybrid automata (HA) is a widely used formalism for model checking of hybrid systems including reachability analysis, safety and stability verification. Given a HA model and a set of state observations received at certain times, we focus on determining if the HA accepts a trajectory that recognizes the observations. We formulate this task as a planning problem encoded in PDDL+ and use a planner to find a hybrid plan that satisfies the observations while following the HA dynamics. Ultimately, we show the adequacy of PDDL+ to model HAs with two well-known examples in the literature on HAs.

Introduction

Hybrid system exhibits a combination of a discrete and continuous behavior which is characteristic of many software-based control systems. Discrete-time linear control systems are widely used as the abstraction of choice for computer controlled physical systems. This paradigm models systems whose continuous dynamics are described by a linear differential equation in combination with transforming measured physical quantities into digital format to denote the logical modes of operation [12].

The first hybrid systems model the software as a finite-state machine that interacts with the physical world through converters. This model later evolved into Hybrid Automata (HA). The HA paradigm builds upon the timed automata model that enables the description of finitely many states where each state represents infinitely many states of the timed automaton [2].

Hybrid systems arise in varied contexts such as manufacturing, automotive engine control, traffic control, communication networks and computer synchronization among others. Industrial applications usually require safety and stability properties and HAs have been widely used as a modeling and verification framework for cyber-physical systems [4].

In this work we present an approach to decide whether a hybrid system modeled as an HA accepts a sequence of discrete-time state observations. That is, determine if

the observed behaviour of a system is actually modeled by a given HA. In discrete environments modeled as finite-state machines, *plan recognition* is the task of predicting the plan of an acting agent based on its observed actions [3, 7, 11]. We leverage the advances on plan recognition and posit our problem as an extension of existing techniques to mixed discrete-continuous domains.

For our purposes, we use PDDL+, a powerful modelling language to express discrete and continuous change. Planning in mixed discrete-continuous domains is a challenging problem and it has received increasing attention in the planning community since it allows to model effectively real-world domains. In this paper, we present a mapping scheme that transforms the elements of an HA into the constructs of PDDL+. We also present an example of the transformation scheme to an scenario of a gas burner modeled through an HA, and run the PDDL+ encoding with the planner ENHSP [9]. The results show the suitability of a planning-based approach for building hybrid trajectories that comply with the state observations.

Background

This section describes background knowledge. The first section is devoted to present the HA paradigm to model hybrid systems. The second section presents PDDL+, the planning language we will use to modelling continuous change .

Hybrid automata

In general, the *state* of a hybrid system is defined by the values of its continuous variables and a discrete mode. The state changes either continuously, according to a flow condition, or discretely according to a *jumps* which correspond to the change of state in an automaton that transitions in response to external events or to the continuous evolution. Continuous flow is permitted as long as so-called *invariants* hold, while discrete transitions can occur as soon as given jump conditions are satisfied or events occur.

A Hybrid automata (HA) defines trajectories (behaviours) of a hybrid system. It is formally defined as a finite state machine with a finite set of continuous variables whose values are described by a set of ordinary differential equations (ODEs). We adopt the formulation of HA as defined in [5].

A HA with polynomial dynamics is a tuple $H = \langle \mathsf{Loc}, \mathsf{Lab}, \mathsf{Edg}, X, \mathsf{Init}, \mathsf{Inv}, \mathsf{Flow}, \mathsf{Jump}, \mathsf{Final} \rangle$ where:

- Loc is a set of discrete variables also called *locations* of H
- Lab is a set of labels
- Edg is a set of edges to denote the discrete transitions
- $X = \{x_1, \ldots, x_n\}$ a set of real-valued variables
- $\mathsf{Init} : \mathsf{Loc} \rightarrow \mathsf{PConstr}(X)$ (polynomial constraints over X), states the possible valuations of X when H starts from a location l
- $\mathsf{Inv} : \mathsf{Loc} \rightarrow \mathsf{PConstr}(X)$ is the invariant of location l, which constrains the possible valuations of X when the control of H is at l
- $\mathsf{Flow} : \mathsf{Loc} \rightarrow \mathsf{PConstr}(X \bigcup \dot{X})$ constrains the valuation evolution of X according to the rate of change given by their time derivative $\dot{X}$

- $\mathsf{Jump} : \mathsf{Edg} \rightarrow \mathsf{PConstr}(X \bigcup X^+)$ gives the jump condition of edge e and X^+ represents updates at the conclusion of a discrete transition
- $\mathsf{Final} : \mathsf{Loc} \rightarrow \mathsf{PConstr}(X)$ is the final condition of location l.

Depending on the analysis question at hand, final conditions can either specify the *unsafe states* of the system or the *desired states* of the system [4].

The timed transition system defined by a hybrid automata H is a tuple $[\![H]\!] = \langle S, S_0, S_f, \Sigma, \rightarrow \rangle$. The set $S = \{(l, v) |\ l \in \mathsf{Loc}, v \in [\![\mathsf{Inv}(l)]\!]\}$ is the state space of H; $S_0 = \{(l, v) \in S |\ v \in [\![\mathsf{Init}(l)]\!]\}$ denotes the set of initial states (S_f is the set $v \in [\![\mathsf{Final}(l)]\!]$); $\Sigma = \mathsf{Lab} \bigcup \mathsf{Time}$ ($\mathsf{Time} \in \mathbb{R}^{\geq 0}$) is a finite set of labels. Finally, $\rightarrow \in S \times \Sigma \times S$ is the transition relation; that is, tuples of the form $(l, v) \rightarrow_\sigma (l', v')$, $\sigma \in \Sigma$, such that every tuple belongs to one of the two following cases:

- **discrete transition**: $e = (l, \sigma, l') \in \mathsf{Edg}$, $\sigma \in \mathsf{Lab}$, and $(v, v') \in [\![\mathsf{Jump}(e)]\!]$
- **continuous transition**: $\sigma \in \mathsf{Time}$, $l = l'$, and there exists $\delta \in \mathbb{R}^{\geq 0}$ and a continuously differentiable function $\xi : [0, \delta] \rightarrow \mathbb{R}^X$ such that $\xi(0) = v$, $\xi(\delta) = v'$ and $\forall t \in [0, \delta]$, $(\xi(t), \dot{\xi}(t)) \in [\![\mathsf{Flow}(l)]\!]$ and $\xi(t) \in [\![\mathsf{Inv}(l)]\!]$

This notion of hybrid system is a blend between the automata theoretic models used for verification of software systems and the differential equations used in control [4]. The paths contained in the timed transition system of H describe in a precise manner the possible trajectories of the hybrid system modeled by H.

We now show an example of the behaviour of the temperature in a room controlled by an HA (thermostat) with two locations $\mathsf{Loc} = \{\mathsf{l_{ON}}, \mathsf{l_{OFF}}\}$, two labels $\mathsf{Lab} = \{\mathsf{ON}, \mathsf{OFF}\}$, and two edges that connect both locations and a variable x that represents the room temperature [8]. Following Table 1, when the thermostat is on, the temperature increases at a rate given by the differential equation $\dot{x} = 5 - 0.1x$; when the thermostat is off the room cools down at a rate given by the differential equation $\dot{x} = -0.1x$. The heater remains on while the temperature is ≤ 22 and it remains off while the temperature is ≥ 18. The thermostat turns on the heater when the temperature decreases below 19C and turns off the heater when the temperature increases above 21C. Initially, the temperature of the room is 15C.

	$\mathsf{l_{ON}}$	$\mathsf{l_{OFF}}$
Inv	$x \leq 22$	$x \geq 18$
Flow	$\dot{x} = 5 - 0.1x$	$\dot{x} = -0.1x$
Edg	$\mathsf{l_{ON}} \rightarrow_{\mathsf{OFF}} \mathsf{l_{OFF}}$	$\mathsf{l_{OFF}} \rightarrow_{\mathsf{ON}} \mathsf{l_{ON}}$
Jump	$x \geq 21$	$x \leq 19$
	$x' = x$	$x' = x$
Init	$x = 15$	–

Table 1. Example of the behaviour of a thermostat for monitoring the temperature in a room described as an HA

A hybrid trajectory that illustrates the evolution of the room temperature according to the above HA is a sequence $q_0 \sigma_0 q_1 \sigma_1 \ldots \sigma_{n-1} q_n$, where $q_i = (l_i, v_i)$, $q_0 \in S_0$,

$q = q_n$, $\sigma_i \in \Sigma$ and $(q_i, \sigma_i, q_{i+1}) \in \rightarrow$. For example, the following path is accepted by the HA: $\langle \mathsf{l_{ON}}, 15\rangle \rightarrow_{1.9} \langle \mathsf{l_{ON}}, 21.08\rangle \rightarrow_{\mathsf{OFF}} \langle \mathsf{l_{OFF}}, 21.08\rangle \rightarrow_{1.1} \langle \mathsf{l_{OFF}}, 18.88\rangle \rightarrow_{\mathsf{ON}} \langle \mathsf{l_{ON}}, 18.88\rangle \rightarrow_{0.8} \langle \mathsf{l_{ON}}, 21.28\rangle \rightarrow_{\mathsf{OFF}} \langle \mathsf{l_{OFF}}, 21.28\rangle \rightarrow_{1.2} \langle \mathsf{l_{OFF}}, 18.86\rangle$.

Mixed Discrete-continuous Planning Domains in PDDL+

PDDL+ [6] (Planning Domain Definition Language) is the language of reference in the planning community to represent problems containing a mixture of discrete and continuous variables together with changes that can be either continuous or instantaneous. Continuous changes are caused by the execution of autonomous processes while instantaneous changes are caused by events or actions. While actions are what is in control of the agents, PDDL+ processes and events describe what happens in the world when some condition is met. Processes last for as long as their preconditions are met. A process is something like gravity's effect on an object which increases the velocity of the object until it reaches the ground. Events correspond to an instantaneous transition that happens the instant their preconditions are met, usually with the effect of transforming their state such that their preconditions are no longer met. Events are uncontrollable, and in the previous example, we might consider an event to be the object hitting the ground when the position of the object reaches some value, e.g., $x = 0$.

A PDDL+ problem Π is formally defined by the tuple $\langle X, F, A, P, E, I, G\rangle$ where:

- X is a set of numeric variables taking values from $\mathbb{Q}$
- F is a set of Boolean variables
- A is a set of actions. Each $a \in A$ is a pair $\langle pre(a), \mathit{eff}(a)\rangle$ where $pre(a)$ is a propositional formula containing both numeric and Boolean conditions. A numeric condition is a comparison of the form $\xi \geq 0$ with ξ being a numeric expressions over X. A Boolean condition is either $l = \top$ or $l = \bot$ where $l \in F$. $eff(a)$ is a set of numeric and propositional assignments
- E is a set of events. An event is structured as an action
- P is a set of processes. A process $p \in P$, as an action and an event, is a pair $\langle pre(p), \mathit{eff}(p)\rangle$. While the preconditions of a process have the same structure of those used for actions and events, the effect's set only contain *continuous effects*. A continuous effect is a time-dependent effect of the form $\dot{x} = \xi$ where ξ is a function representing the derivative of x
- I is an assignment of all variables in X and F, called the initial state
- G is the goal. Its structure is equivalent to that used for action, process and event preconditions

A plan π for a planning instance Π is a finite set of pairs $(t, a) \in \mathbb{Q} \times A$, where t is the timepoint at which a is executed. Each plan induces a continuous trajectory of states that tells us the value of all the variables in X and F for each timepoint t'. Such a trajectory starts at the initial state and is determined by the processes and events that are triggered spontaneously as time goes by, together with the changes prescribed by the actions from the plan. Recall that both processes and events are compulsory executed as their preconditions are met. When a process is triggered, a continuous update of the numeric variables is triggered according to the prescribed derivatives. When an event

is triggered, the state instantaneously changes according to the event's effects. It is required that for each state, the triggered events are non-mutex, so as to avoid non-determinism in how events are sequenced.

A plan π is said to be a solution for a problem Π if each action $(t, a) \in \pi$ is such that a is executable at time t provided the continuous trajectory induced by the triggered events and processes from Π, and there is a $t_g \geq 0$ such that the state at that timepoint satisfies the goal and there is no pair (t_n, a_n) in π such that $t_n > t_g$.

More details on the semantics of PDDL+ can be found into the paper by [6].

We consider for the scope of this paper a slightly variant of PDDL+. In such a variant we can also encode global constraints as those defined in [9, 10]. A global constraint is a propositional formula as that used in action's precondition. PDDL+ extended with global constraints (hereinafter simply PDDL+) is formally the tuple $\langle X, F, A, P, E, I, G, C \rangle$ where all the elements but C are defined as above, whilst C is a set of global constraints. A plan is a valid solution plan if it is a valid plan for the basic version of PDDL+ and, in addition, the induced continuous trajectory also satisfies all constraints from C for any timepoint up to the point in which the goal is satisfied.

Mapping HA to PDDL+

The theory of HA provides an ideal formal basis for the development of a semantics for PDDL+. The work [6] uses the syntax of HAs as a semantic model to construct a formal interpretation of PDDL+ semantics and builds a mapping from planning constructs to constructs of the corresponding HA. In this work, we posit the reverse problem: building a formal mapping of HA semantics into PDDL+ constructs.

Let be $H = \langle \mathsf{Loc}, \mathsf{Lab}, \mathsf{Edg}, X, \mathsf{Init}, \mathsf{Inv}, \mathsf{Flow}, \mathsf{Jump}, \mathsf{Final} \rangle$ a hybrid automata and $\Pi = \langle A, P, E, I, G, X, F, C \rangle$ a PDDL+ problem. A mapping between H and Π is depicted below.

- The first observation is that Time and X of an HA are values in $\mathbb{R}$ whereas numeric variables in Π (including time) are rational values ($\mathbb{Q}$). This means PDDL+ plans are restricted to expressing only a subset of the transitions possible in $[\![H]\!]$, mainly because planners can only make use of timestamps with a finite representation, so there are only countably many plans that can be expressed [6]. Defining an additional metric in planning problems to specify the set of acceptable plans may be an interesting research line to overcome this shortcoming.
- A **location** $l \in \mathsf{Loc}$ is simply represented with a Boolean variable of F to denote whether or not the hybrid automata H is in location l.
- The set of actions A of Π are the constructs to encode the **edges** of H. Specifically, given $a \in A$ and $e = (l, \sigma, l') \in \mathsf{Edg}$:
  - the action name is an item of Lab
  - $pre(a)$ encodes the guards (conditions) of $\mathsf{Jump}(e)$ and the condition that H is in location l
  - $eff(a)$ encodes the reset (updates) of $\mathsf{Jump}(e)$; additionally, the location of H is updated to l'

The set A is the source of non-determinism in planning. In contrast, non-deterministic HAs often appear when different locations partially share the invariants, or the guards of an edge $e = (l, \sigma, l')$ partially overlap with $[\![Inv(l)]\!]$ (in this latter case, if overlapping occurs at only one point on the frontier of $[\![Inv(l)]\!]$, then the HA is deterministic).
For many of the HA typically discussed in the literature, it commonly happens that for any pair of outgoing edges of a location l, $e_1 = (l, \sigma_1, l_1)$ and $e_2 = (l, \sigma_2, l_2)$, it holds $[\![Jump(e_1)]\!] \bigcap [\![Jump(e_2)]\!] = \emptyset$. Particularly, given the following grammar rule that defines a formula φ

$$\varphi ::= \mathsf{PTerm}(X) \bowtie 0 |, \bowtie \in \{<, \leq, =, \geq, >\} \tag{1}$$

, if we restrict $\mathsf{PConstr}(X)$ to a subset of $\mathsf{PConstr}(X)$ defined by conjunctive formulas ($\varphi \wedge \varphi$), and there is no other source of non-determinism, we can then ensure a deterministic discrete behaviour of the HA. In this case, the edges of a location can be represented with PDDL+ events ($E \in \Pi$) instead of actions, thus reducing the branching factor of the search.

- It must hold $I \in \mathsf{Init}(l)$ for some l of H. This means the assignments of the numeric variables X of Π in the initial state must hold in some state $s \in S_0$. Additionally, a Boolean variable of F is used to represent that H is initially at l.
- It must hold $G \in \mathsf{Final}(l)$ for some l of H. This means the assignments of the numeric variables that appear in G must hold in some state $s \in S_f$. Additionally, a Boolean variable of F is used to represent that H is finally at location l.
- **Invariants** of a location are represented with PDDL+ constraints ($C \in \Pi$), encoding an implication of the form $if\ \mathsf{Loc} = l\ then\ \mathsf{Inv}(l)$. Specifically, given a valuation for $X \in H$, the constraints of PDDL+ verify that $\forall x \in X$, $v(x) \in [\![Inv(l)]\!]$. This effectively constraints any continuous trajectory of a solution plan for Π to comply with the invariant conditions of the hybrid automata.
- The **continuous** flows that occur in a location of H are naturally represented with processes of PDDL+. More specifically, for each location l we devise a process whose precondition constrains the process to be active only when the hybrid automata is in state l; we do so by preconditioning the process with the Boolean variable from F relative to l (see above). Then we set the effects of the process with derivatives constraints expressed by $\mathsf{Flow}(l)$, which are the differential constraints exhibited by the hybrid system when in that specific location.

An example of the mapping between the elements of a HA and a PDDL+ planning problem is shown in the section **Illustrative Example: Gas burner**.

Decoding problem

Our objective is to verify if an automata H accepts a hybrid path that satisfies a sequence of discrete-time state observations. This problem, known as *observation decoding*, has been addressed to uncover the state trajectory of a planning agent from a sequence of gapped observations with probabilistic sensor models [1]. In this paper, however, we

assume that observations contain precise measurements for all variables in X, and we leave the management of uncertainty and partial observations for future work.

We note that the sampling rate of the real system that H is simulating may not match every possible change of mode in $[\![H]\!]$; that is, since the observations input may be produced at any unknown rate, this leads to scenarios in which two consecutive observations may belong to the same location of H or scenarios in which two consecutive observations belong to two non-consecutive states of a hybrid path accepted by H. This highlights the unbounded nature of the decoding problem and justifies the use of planning for solving it.

Definition 1 (Observation) *An observation of an automata H is a pair $o = \langle t, \phi \rangle$, where $t \in \mathbb{T}$ is a time point and ϕ is a measurement of variables X.*

Definition 2 (Observation sequence) *We define an observation sequence of length K as $\omega = (o_k)_{k=1}^{K}$, where each $o_k = \langle t_k, \phi_k \rangle$, is a measurement (valuation) ϕ for variables X that was collected at time t_k. For any pair of consecutive observations $\langle o_k, o_{k+1} \rangle$, it holds that $t_{k+1} > t_k$.*

Let $\tau = q_0\sigma_0q_1\sigma_1 \ldots \sigma_{n-1}q_N$ be a hybrid trajectory. Since every transition label belongs to Lab or Time, $\sigma_i \in \mathbb{R}^{\geq 0}$. We define the time at which the system reaches state q_i recursively as $\mathsf{clock}(q_0) = 0$ and $\mathsf{clock}(q_i) = \mathsf{clock}(q_{i-1}) + \sigma_{i-1}, 1 \leq i \leq N$.

Definition 3 (Decoding problem) *Given a hybrid automata H and an observation sequence $w = (\langle t_1, \phi_1 \rangle, \langle t_2, \phi_2 \rangle \ldots, \langle t_K, \phi_K \rangle)$, the solution to the decoding problem $\langle H, w \rangle$ is a hybrid trajectory $\tau = (l_0, v_0)\sigma_0(l_1, v_1)\sigma_1 \ldots \sigma_{n-1}(l_N, v_N)$ such that there exists only one state $q_i \in \tau$ that satisfies $t_k = \mathsf{clock}(q_i)$ and $\phi_k = v_i$.*

Among all trajectories accepted by H that solve the decoding problem, we aim to find $\tau^* = min_{\tau \in H} f(q_0, q_1, \ldots, q_N)$, the hybrid trajectory that minimizes the path length. The ability to find the optimal path will depend on the planner functionalities. Intuitively, this means finding a trajectory $q_0\sigma_0q_1\sigma_1 \ldots \sigma_{n-1}q_N \in \Sigma^*$ that minimizes the number of discrete transitions. Since planners are typically designed for this metric, this amounts to finding a PDDL+ plan with minimal plan length.

Illustrative example: Gas burner

In this section we present the PDDL+ encoding of a well-known hybrid system formalized as an hybrid automaton. We solved the problem with the planner ENHSP [9], a heuristic search planner that can read and solve the targeted PDDL+. We run ENHSP with default parameters (corresponding to running a plain state-space search guided by the AIBR heuristic) but change the discretization-based approach so as to represent the example of interest with a suitable time granularity. In particular, we set the discretization factor of the Gas burner example to 0.1s.

The hybrid system consists of a single gas-burner that is used for heating alternatively two water tanks. The HA that models this problem is shown in Figure 1. The problem description as well as the figure are borrowed from [5].

Figure 1 shows that the HA comprises three locations: in l_1 the burner is heating tank_1 whose temperature is represented with variable x_1; in l_2 the burner increases the temperature x_2 of tank_2; in location l_0 the burner is switched off. The dynamics in each location is given by predicates ON_i and OFF_i ($i = 1, 2$): the constants a_i model the heat exchange rate of tank i with the room, b_i model the heat exchange rate between the two tanks and h_i depends on the power of the gas-burner. In this example, constant values are $h_1 = h_2 = 2$, $a_1 = a_2 = 0.01$ and $b_1 = b_2 = 0.005$.

With initial values $x_1 = 0$, $x_2 = 50$ and starting in location l_1, the burner heats up tank_1 until it reaches a temperature of 100 degrees. At this time, the temperature of tank_2 is below 80 degrees and the HA takes the discrete transition toggle to location l_2, where the burner heats up tank_2 until it reaches 100 degrees. Since x_1 is still above 80 degrees, the HA transits via $\mathrm{turnoff}_2$ to l_0 where the burner is off. It briefly remains here until x_1 falls to 80 degrees, at which point it takes transition turnon_1 to l_1 where the burner heats up tank_1.

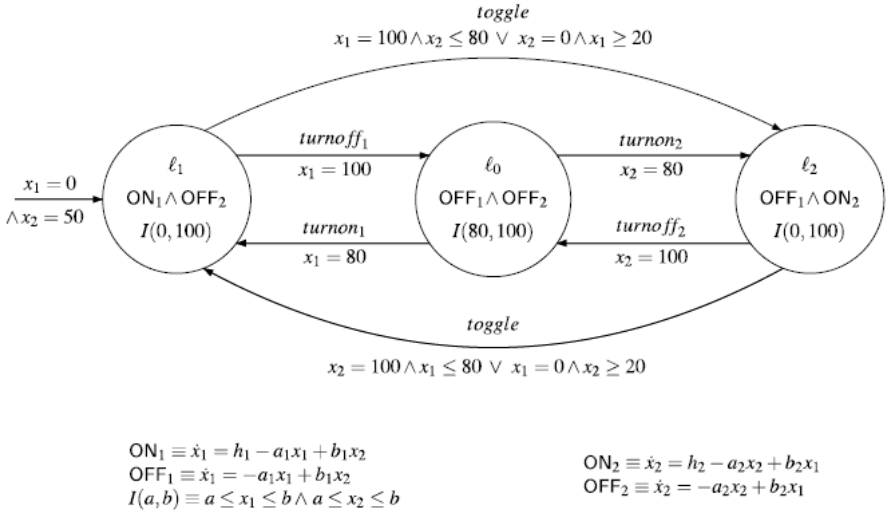

Fig. 1. HA representing a gas burner task (the figure is taken from [5])

Figure 2 shows the process that encodes the Flow in location l_1, where $\#t$ is a syntactical device to explicit the fact that the effect is a time-dependent effect. Figure 3 depicts the edge (l_1, v_1, l_2) represented as a PDDL+ action. When the HA is in l_1 and the temperature of tank_1 reaches 100C, the planner chooses between action toggle or action $\mathrm{turnoff}_1$. In contrast, when the HA is at l_0 and x_1 takes value 80C, the required jump to l_1 is represented with an event since there is no other available choice (see Figure 4).

Figure 5 shows the mandatory conditions of $\mathsf{Inv}(l_0)$ that must be satisfied when the system is in l_0. Finally, Figure 6 shows the implementation of an event to capture the

```
(:process flow_l1
    :parameters ()
    :precondition
       (and
        (l1)
       )
    :effect
       (and
        (increase (x1) (* #t (+ 2 (+ (* -0.01 (x1))(* 0.005 (x2))))))
        (increase (x2) (* #t (+ (* -0.01 (x2))(* 0.005 (x1)))))
       )
)
```

Fig. 2. Flow of location l_1 encoded as a process

```
(:action toggle_l1_l2
    :parameters ()
    :precondition
        (and
            (l1)
            (or
              (and (>= (x1) 99) (<= (x1) 101) (<= (x2) 81))
              (and (>= (x2) -1) (<= (x2) 1) (>= (x1) 19))
            )
        )
    :effect (and (not (l1)) (l2))
)
```

Fig. 3. Edge toggle from l_1 to l_2 encoded as an action

input observation. We note that for an observation $o = \langle t, \phi \rangle$, where $\phi = [x_1, x_2]$ represents the temperature of tank_1 and tank_2, respectively, (i) we set the time granularity of o to 0.1s; and (ii) the precision of the temperature reading is set to ± 1 its actual value.

```
(:event turn_on1
    :parameters ()
    :precondition (and
                          (l0)
                          (>= (x1) 79)
                          (<= (x1) 81)
                  )
    :effect (and (not (l0)) (l1))
)
```

Fig. 4. Edge $turnon_1$ from l_0 to l_1 encoded as an event

We tested a decoding problem with observation sequence $w = (\langle 50, [87.4, 40.2] \rangle, \langle 100, [85.7, 96.3] \rangle, \langle 150, [80.8, 98.4] \rangle, \langle 200, [81.8, 94.6] \rangle$, where each tuple corresponds

```
(:constraint inv_l0
    :parameters ()
    :condition (and
                    (or
                        (not (l0))
                        (and
                            (<= (x1) 101)
                            (>= (x1) 79)
                            (<= (x2) 101)
                            (>= (x2) 79)
                        )
                    )
                )
)
```

Fig. 5. Invariant of l_0 encoded as a constraint

to an observation, being the first element the time of the observation and the values in brackets the values of x_1 and x_2. The ENHSP planner found a plan in 127s corresponding to a hybrid trajectory accepted by the HA that explains the four observations (see Figure 7). The 2013 actions of the plan are broken down into 2000 wait actions corresponding to 200s of continuous flow with a time discretization of 0.1s, 9 actions and events for the discrete transitions, and 4 events for the validation of the observations. The validate events are used for reading the observations and the `waiting` lines denote the continuous trajectory in a location. The hybrid trajectory starts in location l_1, which satisfies the first observation. At $t = 60$, the HA jumps to location l_2. Subsequently, at $t = 94.0$ the action $\mathrm{turnoff}_2$ is executed and the system enters location l_0. At $t = 97$ the event turnon_1 is triggered as the temperature of tank_1 cools down to 80. The second observation matches a state whose location is l_1. The plan goes on until the last observation is validated.

```
(:event validate
    :parameters ( ?x - observation)
    :precondition
       (and
            (not (observed ?x))
            (> (time_observation ?x) (- (running_time) 0.05))
            (< (time_observation ?x) (+ (running_time) 0.05))
            (>= (x1) (- (x1_observation ?x) 1))
            (<= (x1) (+ (x1_observation ?x) 1))
            (>= (x2) (- (x2_observation ?x) 1))
            (<= (x2) (+ (x2_observation ?x) 1))
      )
    :effect (and (observed ?x))
)
```

Fig. 6. Event for validating an input observation

```
0.0: -----waiting---- [50.0]
50.0: (validate o1)
50.0: -----waiting---- [60.2]
60.2: (toggle_l1_l2)
60.2: -----waiting---- [93.8]
93.8: (turn_off2)
93.8: -----waiting---- [96.8]
96.8: (turn_on1)
96.8: -----waiting---- [100.0]
100.0: (validate o2)
100.0: -----waiting---- [108.7]
108.7: (turn_off1)
108.7: -----waiting---- [135.2]
135.2: (turn_on2)
135.2: -----waiting---- [147.6]
147.6: (turn_off2)
147.6: -----waiting---- [149.7]
149.7: (turn_on1)
149.7: -----waiting---- [150.0]
150.0: (validate o3)
150.0: -----waiting---- [162.0]
162.0: (turn_off1)
162.0: -----waiting---- [190.5]
190.5: (turn_on2)
190.5: -----waiting---- [200.0]
200.0: (validate o4)
Plan-Length:2013
```

Fig. 7. Plan returned by ENHSP for the observation sequence

We illustrate now a portion of the hybrid trajectory corresponding to the plan in Figure 7 until $t = 108.7$. We show the states traversed by the HA and highlight in bold the states that satisfy the first two observations (`(validate o1)` and `(validate o2)`). A reminder that we are using a precision of $0.1s$ for the time and ± 1 for the temperature of the tanks.

$\langle l_1, [0, 50]\rangle \rightarrow_{50} \langle l_1, [\mathbf{87.1}, \mathbf{40.3}]\rangle \rightarrow_{10.2} \langle l_1, [100, 40.9]\rangle \rightarrow_{\text{toggle}}$
$\langle l_2, [100, 40.9]\rangle \rightarrow_{33.6} \langle l_2, [81.9, 99.1]\rangle \rightarrow_{\text{turnoff}_2} \langle l_0, [81.9, 99.1]\rangle \rightarrow_3$
$\langle l_0, [81, 97.4]\rangle \rightarrow_{\text{turnon}_1} \langle l_1, [81.1, 97.4]\rangle \rightarrow_{3.2} \langle l_1, [\mathbf{86.2}, \mathbf{95.6}]\rangle \rightarrow_{8.7}$
$\langle l_1, [99.6, 91.5]\rangle \rightarrow_{\text{turnoff}_1} \langle l_0, [99.6, 91.5]\rangle \rightarrow_{26.5} \ldots$

Conclusions and Discussion

This work presents a first step towards recognition of trajectories in cyber-physical systems that exhibit a hybrid dynamics. Our proposal lies in using the Hybrid Automata formalism to model the system behaviour, mapping the semantics of a HA into a PDDL+ planning problem and then solving the problem with a planner. As a result,

we obtain a plan which defines the trajectory that matches the discrete-time state observations gathered from the hybrid system.

One question underlying our proposal is the added value of encoding the semantics of a HA into a mixed discrete-continuous planning problem over other techniques such as for example the use of verification tools for HAs. In this regard, model-checking tools are widely used in formal verification of safety properties of hybrid systems. This is commonly addressed as a reachability analysis that computes the set of all possible states the system can reach and checks whether every execution of the system always stays within a set of safe states. There are several distinctive features in our problem that justifies the use of planning technology:

- between two gathered observations, the hybrid system may go through an indefinite number of states; that is, the length of the trajectory that links two observations is unknown
- state reachability is a satisfiability problem aimed at verifying whether or not the system reaches a safe state. In plan recognition, we seek to optimize the length of the trajectory or any other metric.

The paper illustrates the behaviour of a simple HA that models a gas burner to heat two water tanks, its encoding in PDDL+ and resolution with the ENHSP planner. We have recently initiated an experimentation with more complex examples that feature composition of various hybrid automata. The promising results highlight that the PDDL+ modeling of the problem allows synchronization of HAs wherein each presents a different dynamics. All in all, this paper is a first step towards the modeling of HA with planning technology.

References

1. Diego Aineto, Sergio Jiménez, and Eva Onaindia. Observation decoding with sensor models: Recognition tasks via classical planning. In *30th International Conference on Automated Planning and Scheduling ICAPS*, pages 11–19. AAAI Press, 2020.
2. Rajeev Alur and David L. Dill. Automata for modeling real-time systems. In *Automata, Languages and Programming, 17th International Colloquium, ICALP90*, volume 443 of *Lecture Notes in Computer Science*, pages 322–335. Springer, 1990.
3. Sandra Carberry. Techniques for plan recognition. *User Model. User-Adapt. Interact.*, 11(1-2):31–48, 2001.
4. Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem, editors. *Handbook of Model Checking*. Springer, 2018.
5. Laurent Doyen, Goran Frehse, George J. Pappas, and André Platzer. Verification of hybrid systems. In Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem, editors, *Handbook of Model Checking*, pages 1047–1110. Springer, 2018.
6. Maria Fox and Derek Long. Modelling mixed discrete-continuous domains for planning. *J. Artif. Intell. Res.*, 27:235–297, 2006.
7. Miquel Ramírez and Hector Geffner. Plan Recognition as Planning. In *21th International Joint conference on Artifical Intelligence, IJCAI*, pages 1778–1783. AAAI Press, 2009.
8. Jean-François Raskin. An introduction to hybrid automata. In Dimitrios Hristu-Varsakelis and William S. Levine, editors, *Handbook of Networked and Embedded Control Systems*, pages 491–518. 2005.

9. Enrico Scala, Patrik Haslum, Sylvie Thiébaux, and Miquel Ramírez. Interval-based relaxation for general numeric planning. In *ECAI 2016 - 22nd European Conference on Artificial Intelligence*, volume 285 of *Frontiers in Artificial Intelligence and Applications*, pages 655–663. IOS Press, 2016.
10. Enrico Scala, Miquel Ramírez, Patrik Haslum, and Sylvie Thiébaux. Numeric planning with disjunctive global constraints via SMT. In *Proceedings of the Twenty-Sixth International Conference on Automated Planning and Scheduling, ICAPS 2016, London, UK, June 12-17, 2016*, pages 276–284. AAAI Press, 2016.
11. Gita Sukthankar, Robert P. Goldman, Christopher Geib, David V. Pynadath, and Hung Bui. *Plan, Activity, and Intent Recognition: Theory and Practice*. Morgan Kaufmann, 2014.
12. Paulo Tabuada. *Verification and Control of Hybrid Systems - A Symbolic Approach*. Springer, 2009.