Application of Hybrid I/O Automata in Safety Verification of Pitch Controller for Model Helicopter System*

Sayan Mitra[1] Yong Wang[2] Nancy Lynch[1]
Eric Feron[2]

[1]MIT Laboratory for Computer Science,
Cambridge, MA 02139, USA
{mitras,lynch}@theory.lcs.mit.edu

[2]MIT Laboratory for Information and Decision Systems,
Cambridge , MA 02139, USA
{y_wang, feron}@mit.edu

Abstract: This paper presents an application of the Hybrid I/O Automaton modelling framework [9] to a realistic hybrid system verification problem. A supervisory pitch controller for ensuring the safety of a model helicopter system is designed and verified. The supervisor periodically observes the plant state and takes over control from the user when the latter is capable of taking the plant to an unsafe state. The design of the supervisor is limited by the actuator bandwidth, the sensor inaccuracies and the sampling rates. Safety is proved by inductively reasoning over the executions of the composed system automaton. The paper also presents a set of language constructs for specifying hybrid I/O automata.

1 Introduction

Formal verification of hybrid systems is a hard problem. It has been shown that checking reachability for even a simple class of hybrid automata is undecidable [4]. Algorithmic techniques have been developed for several smaller subclasses of hybrid automata making automatic verification possible [1]. However these subclasses are too weak to represent realistic hybrid systems. Consequently the languages and tools, like HyTech [3], developed for algorithmic methods are not adequate for describing general hybrid systems. An alternative approach to verification is based on the hybrid Input/Output automaton (HIOA) model [10, 11, 9]. In this approach the properties of a system are derived by induction on the executions of the automaton model, see [6, 14, 8] for related earlier works. Being a more expressive model, hybrid I/O automata enables us to model a larger class of hybrid systems. Although at present there is no tool support for HIOA, we intend to extend the IOA Toolset [2] for checking HIOA code and also build theorem prover interfaces for HIOA to partially automate the verification process.

This paper presents the verification of a supervisory controller of a model helicopter system using the HIOA framework. The helicopter system (Figure 1) is manufactured by Quanser [5]. It is driven by two rotors mounted at the two ends of its body and it is attached to an arm which is fixed at one end. The helicopter can revolve about the fixed end of the arm and has three degrees

*Funding for this research has been provided by AFRL contract F33615-01-C-1850

of freedom. The rotor inputs are either controlled by the user with a joystick, or by controllers designed by the user. Students of Aeronautics and Astronautics at MIT experiment with different controllers for the helicopter. Controllers are often unsafe and damage the equipment by pitching the helicopter too high or too low. This is also a hazard for the users. Therefore the safety of the system is important. A supervisory controller is designed to prevent the helicopter from reaching unsafe states. The supervisor periodically observes the position and the velocity of the helicopter and overrides the user's controller by conservatively estimating the worst that might happen if the user is allowed to continue. The supervisor is limited by the actuator bandwidth, the sampling rate, and sensor inaccuracies. These factors also make the verification more complex.

This paper also describes a specification language for HIOA. In this language discrete transitions of hybrid I/O automata are specified in the usual precondition-effect style, and the continuous evolution is written in terms of constrained "state-space" models called activities. The language, to date is for manual use, it constitutes a first step for automating the verification process using HIOA.

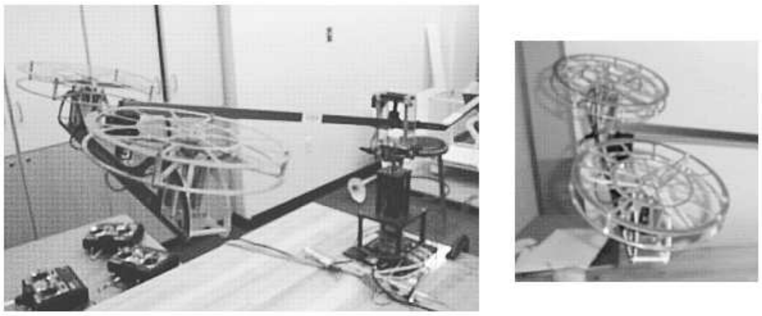

Figure 1: Helicopter model with three degrees of freedom.

The contributions of this paper are: (1) demonstration of a realistic application of the hybrid I/O automata based verification methodology, (2) design of the supervisory controller which ensures safety of the Quanser helicopter system along the pitch axis, and (3) a set of language constructs for specifying hybrid I/O automata.

In Section 2 we review the hybrid I/O automata model and describe the specification language. We present the HIOA models of the system components and the supervisor in Sections 3 and 4 respectively. We present the proof for safety of the system in Section 5. Concluding remarks and future directions for research are discussed in Section 6.

2 Hybrid I/O Automata

In this section we briefly review the HIOA mathematical model. For a complete discussion of the model refer to [9]. Earlier versions of the model appeared in [10] and [11].

2.1 The HIOA Model

A hybrid I/O automaton captures the hybrid behavior of a system in terms of discrete transitions and continuous evolution of its state variables. Let V be the set of variables of automaton $\mathcal{A}$. Each $v \in V$ is associated with a *(static) type* defining the set of values v can assume. A *valuation* $\mathbf{v}$ for V is a function that associates each variable $v \in V$ to a value in $type(v)$. A trajectory τ

of V is defined as a mapping $\tau : J \rightarrow val(V)$ where J is a left closed interval of time. If J is right closed then τ is said to be *closed* and its *limit time* is the supremum of the domain of τ, also written as $\tau.ltime$. Each variable $v \in V$ is also associated with a *dynamic type* (or *dtype*) which is the set of trajectories that v may follow. Dynamic types must satisfy the *time-shift, subinterval* and *pasting* closure properties described in [9]. A hybrid I/O automaton $\mathcal{A}$ is a tuple $(X, U, Y, Q, \Theta, H, I, O, \mathcal{D}, \mathcal{T})$ where

- X: set of *internal or state variables*, U : set of *input variables*, Y : set of *output variables*. The set of variables $V \triangleq U \cup Y \cup X$. The set of *locally controlled variables* $Z \triangleq X \cup Y$.
- H : set of *internal actions*, I : set of *input actions* , O : set of *output actions*. The set of actions $A \triangleq H \cup I \cup O$.
- $Q \subseteq val(X)$: a set of states
- $\Theta \subseteq Q$: non-empty set of *start states*.
- $\mathcal{D} \subseteq Q \times A \times Q$: set of *discrete transitions*. A transition $(\mathbf{x}, a, \mathbf{x}') \in \mathcal{D}$ is written in short as $\mathbf{x} \overset{a}{\rightarrow}_{\mathcal{A}} \mathbf{x}'$.
- $\mathcal{T}$: set of *trajectories* for V, such that for every trajectory τ in $\mathcal{T}$, and for every $t \in \tau.dom$, $\tau(t).X \in Q$. It is required that $\mathcal{T}$ is closed under prefix, suffix, and concatenation. The first state $\tau(0).X$ of trajectory τ is written as $\tau.fstate$. Similarly if $\tau.dom$ is finite then $\tau.lstate = \tau(\tau.ltime).X$.

In addition, a hybrid I/O automaton also satisfies: (1) the input action enabling property, which prevents it from blocking any input action and (2) the input trajectory enabling property, which ensures that it is able to accept any trajectory of the input variables either by allowing time to progress for the entire length of the trajectory or by reacting with some internal action before that.

An *execution* of $\mathcal{A}$ is a finite or infinite sequence of actions and trajectories $\zeta = \tau_0, a_1, \tau_1, a_2 \ldots$,where (1) each $\tau_i \in \mathcal{T}$, (2) $\tau_0.fstate \in \Theta$ and (3) if τ_i is not the last trajectory in ζ then τ_i is finite and $\tau_i.lstate \overset{a_{i+1}}{\rightarrow} \tau_{i+1}.fstate$. An execution is *closed* if the sequence is finite and the domain of the final trajectory is a finite closed interval. The length of an execution is the number of elements (actions and trajectories) in the sequence.

2.2 New Addition to HIOA Structure: *Activities*

In the earlier works [6, 14, 8] using the HIOA model, trajectories of automata were specified using an ad hoc mixture of integral, algebraic equations and English. While this form of specification is simple to read, it does not lend itself easily to systematic analysis, nor does it enforce a consistent style in writing specifications. The specification language [12] we use in this paper uses "state space" representation [7] of the trajectories. This representation is concise, natural, and widely used in the analysis of dynamical systems. To make this representation work, we have introduced extra structure into the basic HIOA model of [9].

2.2.1 Assumptions

We assume that the time domain is R. A variable v is *discrete* if its dynamic type is the pasting closure of the set of constant functions from left closed intervals of time to $type(v)$. A variable is *continuous* if its dynamic type is the pasting closure of the set of continuous functions from left closed intervals of time to R. For any set S of variables, S_d and S_a refers to the discrete and continuous subsets of S respectively. The following are the first two restrictions we impose on the HIOA model:

R1 Every variable is either discrete or continuous.

R2 Locally controlled discrete variables remain constant over trajectories, that is, $\tau.lval \lceil Z_d = \tau.fval \lceil Z_d$, for all $\tau \in \mathcal{T}$.

Hence in this model, the evolution of the locally controlled variables of a HIOA is completely specified by the discrete transitions and the evolution of the variables in Z_a. The state space representation is used to specify this evolution as explained in the next part.

2.2.2 State Model

Let e be a real valued algebraic expression involving the variables in $X \cup U$. For a given trajectory τ we use $\tau.e$ to denote the function with domain $\tau.dom$ that gives the value of the expression e at all times during trajectory τ. Given that v is a locally controlled continuous variable, a trajectory τ satisfies the algebraic equation

$$v = e,$$

if for every $t \in \tau.dom$, $\tau \downarrow v(t) = \tau.e(t)$.

If an algebraic equation involves a nondeterministic choice such as

$$v \in \; [e_1, e_2],$$

then trajectory τ satisfies the equation if for every $t \in \tau.dom$, $\tau \downarrow v(t) \in [\tau.e_1(t), \tau.e_2(t)]$.

If the expression e is integrable when viewed as a function, then τ satisfies the differential equation

$$\dot{v} = e,$$

if for every $t \in \tau.dom$, $\tau \downarrow v(t) = \tau \downarrow v(0) + \int_0^t \tau.e(t')\ dt'$.

A *state model* of HIOA $\mathcal{A}$ consists of $|Z_a|$ number of algebraic and/or differential equations with exactly one equation having v or $d(v)$ as its left hand side. The right hand sides of the equations are algebraic expressions involving the variables in $X \cup U$. It is also required that there are no circular relationships between the state variables.

A state model specifies[1] the evolution of every variable v in Z_a from some initial valuation. A trajectory τ satisfies a state model E if at all times in $\tau.dom$, all the variables in Z_a satisfy the differential and algebraic equations in E with $\tau(0)$ defining the initial valuations.

2.2.3 Activities

An *activity* α of HIOA $\mathcal{A}$ consists of three components:

1. An *operating condition* $P \subseteq Q$,
2. A *stopping condition* $P^+ \subseteq Q$, and
3. A a state model E for $\mathcal{A}$.

The set of trajectories defined by activity α is denoted by $[\alpha]$. A trajectory τ belongs to the set $[\alpha]$ if the following conditions hold:

[1] By *specifies* we mean restricts rather than uniquely determines. Due to possible nondeterminism in the state model, unique determination might be impossible.

- τ satisfies the state model E.
- For all $t \in \tau.dom$, $(\tau \downarrow X)(t) \in P$.
- If $(\tau \downarrow X)(t) \in P^+$ for $t \in dom(\tau)$ then τ is closed and $t = \tau.ltime$.

The set of trajectories of an automaton is defined to be the union of all the sets of trajectories specified by the activities of an automaton. Suppose automaton $\mathcal{A}$ has n activities, namely α_i for $i \in \mathcal{I}$, where $\mathcal{I}$ is an arbitrary index set with n elements. Then $\mathcal{T}_{\mathcal{A}} = \cup_{i \in\ \mathcal{I}}\ [\alpha_i]$.
With the present model, as defined so far, it would be possible for an automaton to switch activities over a single trajectory in an execution. Our final restriction on the HIOA model prevents such switches. In other words, the switching of activities (or state models) is brought about only by discrete steps:

R3 Operating condition of all the activities are disjoint, that is, $P_i \cap \mathcal{P}_j = \emptyset$ if $i \neq j$.

It can be proved that a set of trajectories specified by a set of activities respecting **R1**, **R2**, and **R3**, satisfy the prefix, suffix, and concatenation closure properties.

Lemma 2.1 *Suppose $\mathcal{T}$ is a set of trajectories specified by the activities α_i, $i \in \mathcal{I}$, where $\mathcal{I}$ is an index set. Then $\mathcal{T}$ is closed under prefix, suffix, and concatenation.*

Proof: $\mathcal{T} = \cup_{i=1}^{n}[\alpha_i]$ is closed under prefix and suffix because each of the sets $[\alpha_i]$ are closed under prefix and suffix. Let $\tau_0, \tau_1, \tau_2, \ldots$ be a sequence of trajectories in $\mathcal{T}$ such that, for each non-final index i, τ_i is closed and $\tau_i.lstate = \tau_{i+1}.fstate$. From the concatenation closure requirement of $\mathcal{T}$, it is necessary that $\tau = \tau_0 \frown \tau_1 \frown \tau_2 \ldots \in \mathcal{T}$.

Let $\tau_i \in [\alpha_j]$, $\tau_{i+1} \in [\alpha_k]$, where $j, k \in \mathcal{I}$, therefore $\tau_i.lstate \in P_j$ and $\tau_{i+1}.fstate \in P_k$. Let us assume for the sake of contradiction that $j \neq k$. From **R3**, $P_j \cap P_k$ must be empty. But we have $\tau_i.lstate \in P_j \cap P_k$, which contradicts our assumption. Therefore it must be the case that $j = k$. Therefore every trajectory in the sequence belongs to the same activity, say $[\alpha_j]$. As $[\alpha_j]$ is closed under concatenation, $\tau_0 \frown \tau_1 \frown \tau_2 \frown \ldots \in \mathcal{T}$. $\square$

2.3 Language Constructs

Our specification language is based on the above modified HIOA model. Variables are declared by specifying their names, types, dtypes, and optionally their initial valuations. For input variables initial valuations cannot be specified. Varibales declared with the **analog** keyword are continuous, else they are discrete. Algebraic expressions are written using the operators $+, -, *$, and $\backslash$. An expression involving nondeterministic choice, such as $v \in [e_1, e_2]$, is written as:

$$v = \textbf{choose}[e_1, e_2]$$

The derivative of a continuous variable x is written as $d(x)$. The discrete transitions are written in the precondition—effect style of the IOA language [2]. An activity $\alpha : (P, P^+, E)$ is written as:

activity α **when** P **evolve** E **stop at** P^+.

For automata with a single activity, if the operating condition P is not specified explicitly, then it is assumed to be the entire state space of the automaton. If the stopping condition P^+ is omitted then it is assumed to empty.

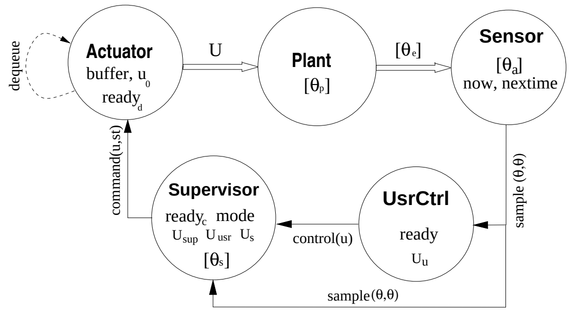

Figure 2: Components of Helicopter system. Continuous and discrete communication between components are shown by wide and thin arrows respectively. The internal variables are marked inside the circles. Internal actions are shown with dased self loops.

3 Specification of System Components

In this section we present a HIOA model of the helicopter system, except for the supervisory controller, which is in Section 4; the interaction among the different components of the system are shown in Figure 2. A nonlinear dynamical model of the helicopter with three degrees of rotational freedom can be found in [13]. In this paper we consider the pitch dynamics, which are critical for safety. The roll and yaw effects are eliminated by making the initial conditions and the disturbances along these axes to be zero and giving identical input to the two rotors. The pitch dynamics is described by the following differential equation :

$$\ddot{\theta} + \Omega^2 \cos\theta = U(t), \tag{1}$$

where Ω is the rotational inertia and U is the net input for the pitch axis. The `Plant` automaton specifies the evolution of the pitch angle (θ_p^0) and velocity (θ_p^1) of the helicopter in terms of the input U. We define three global types `RAD,` `RADPS` and `UTYPE` for variables representing angle, angular velocity and actuator output respectively. The constant $\Theta(\dot{\Theta})$ corresponds to the largest absolute value of any variable representing angle angle (angular velocity). The state variables θ_p^0 and θ_p^1 are initialized to some value from the set $\mathbf{U}$, which is defined in equation (5). The `Plant` automaton has a single activity *pitch_dynamics*, which describes the evolution of the locally controlled continuous variables and it operates over the entire state space. The state variables evolve according to equation (1), and the output variables copy values from the state variables.
The `Plant` is said to be safe at a given state if the pitch angle θ_p^0 is within the allowed limits θ_{min} and θ_{max}. We define the set of safe states as:

$$\mathbf{S} \triangleq \{s \mid \theta_{min} \leq s.\theta_p^0 \leq \theta_{max}\}, \tag{2}$$

The function of the `Sensor` automaton is to periodically convey the state of `Plant` to the controllers as observed by the physical sensors. It is parameterized by the sensor errors for pitch angle ϵ_0, and velocity ϵ_1, and the sampling period Δ. The values of the input variables θ_e^0, θ_e^1, are copied into θ_a^0 and θ_a^1 respectively. Value of the variable *now* increases monotonically with a constant rate of unity along all trajectories. The stopping condition of the *read* activity ensures that a *sample* action occurs after every Δ interval of time. The value of θ_d^0 (θ_d^1) is nondeterministically chosen to

type RAD = Real **suchthat** $(i : \texttt{RAD},\ |i| \leq \Theta)$
type RADPS = Real **suchthat** $(i : \texttt{RADPS},\ |i| \leq \dot{\Theta})$
type UTYPE = Real **suchthat** $(i : \texttt{UTYPE} \mid\ U_{min} \leq i \leq U_{max})$

hybridautomaton Plant(Ω : Real)

variables
  input analog U : UTYPE,
  internal analog θ_p^0 : RAD, θ_p^1 : RADPS, *initially* $(\theta_p^0, \theta_p^1) \in \mathbf{U}$,
  output analog θ_e^0 : RAD, θ_e^1 : RADPS

trajectories
  activity *pitch_dynamics*
    evolve $d(\theta_p^0) = \theta_p^1$; $d(\theta_p^1) = -\Omega^2 \cos\theta_p^0 + U$;
      $\theta_e^0 = \theta_p^0; \theta_e^1 = \theta_p^1$

Figure 3: HIOA specification of the plant

be within $\pm\epsilon_0$ ($\pm\epsilon_1$) of θ_a^0 (θ_a^1). This choice models the noise or the uncertainties in the sensing devices.

The `UsrCtrl` automaton, shown in Figure 5, models an arbitrary user controller. This automaton reads the *sample* action as input and triggers an output *control(u_d)* action, which communicates the output U_u of the user's controller to the supervisor. The output U_u is modeled as a nondeterministic choice over the entire range of possible values. This captures our assumption that the user is capable of issuing arbitrarily bad outputs. The design of a safe supervisor for this particular model of `UsrCtrl` ensures that the system is safe for any user designed controller because every controller must implement this specification of `UsrCtrl`. The `UsrCtrl` automaton does not have any continuous variables, and so the only activity *void* does not specify any state model. The stopping condition ensures that the trajectories terminate when *ready* is set to `true`.

Next, we present the `Actuator` automaton, which models the actuator and the D/A converter. The delay in the actuator response is modeled by a FIFO *buffer* of (u, st) pairs, where u is a command issued from `Supervisor`, and the scheduled time st is the time at which u is to be delivered to the plant. A *command(u,m)* action appends $(u, timer + \tau_{act})$ to *buffer* and a *dequeue* action copies *buffer*`.head.u` to u_o and removes *buffer*`.head`. The $ready_d$ flag is set when a new pair is added and it is reset when a pair is removed from *buffer*. The following properties of `Actuator` can be derived from its specification.

Invariant 3.1 *In any reachable state s of* `Actuator`*, for all* $0 \leq i < s.buffer.\texttt{size} - 1$, $s.now \leq s.buffer\texttt{[i].st} \leq s.buffer\texttt{[i+1].st} \leq s.now + \tau_{\mathsf{act}}$.

Proof: The base case is trivially true because $s.buffer = \{\}$. Consider a discrete steps of the form $s \xrightarrow{\pi} s'$. If π=*sample* or π =*control* then the invariant is preserved because none of the variables involved in it are changed by π.

Case 1: π =*command(u,t)*. From the code it follows that $s'.buffer = s.buffer + (u, s.now + \tau_{\mathsf{act}})$. Since $s'.now = s.now$, it follows from the inductive hypothesis that $s'.now \leq$

hybridautomaton `Sensor`(ϵ_0,ϵ_1, Δ : `Real`)

actions
  output *sample* (θ_d^0 : `RAD` , θ_d^1: `RADPS`)

variables
  input analog θ_e^0 : `RAD`, θ_e^1 : `RADPS`,
  internal analog θ_a^0 : `RAD` := 0, θ_a^1 : `RADPS` := 0,
    now: `Real` := 0,
  internal *next_time* : `Real` := Δ

discrete transitions
  output *sample* (θ_d^0 , θ_d^1)
  pre $now = next\_time \wedge$
    $\theta_d^0 \in [\theta_a^0 - \epsilon_0, \theta_a^0 + \epsilon_0] \wedge$
    $\theta_d^1 \in [\theta_a^1 - \epsilon_1, \theta_a^1 + \epsilon_1]$
  eff $next\_time := now + \Delta$

trajectories
  activity *read*
    evolve d(*now*) =*1*; $\theta_a^0 = \theta_e^0$; $\theta_a^1 = \theta_e^1$;
    stop at $now = next\_time$

Figure 4: HIOA specification of the sensor and A/D conversion circuit

$s'.buffer$.`nexttolast.st` $\leq$ $s'.buffer$.`last.st`$\leq$ $s'.now + \tau_{\texttt{act}}$. Therefore $s'.now \leq$ $s'.buffer$`[i].st` $\leq$ $s'.buffer$`[i+1].st`$\leq$ $s'.now + \tau_{\texttt{act}}$, for all $0 \leq i < s'.buffer$.`size` $- 1$.

Case 2: π =*dequeue*. From the code it follows that $s'.buffer$= $s.buffer$.`tail`. Since $s'.now = s.now$, it follows from the inductive hypothesis that $s'.now \leq s'.buffer$`[i].st` $\leq s'.buffer$`[i+1].st`$\leq s'.now + \tau_{\texttt{act}}$, for all $0 \leq i < s.buffer$.`size` $- 1$.

For the continuous part, consider a closed trajectory τ of `Actuator` with $s = \tau.fsate$, $s' = \tau.lstate$, and $t' = \tau.ltime$. From the inductive hypothesis it is known that $s.now \leq s.buffer$`[i].st` $\leq$ $s.buffer$`[i+1].st`$\leq s.now + \tau_{\texttt{act}}$, for all $0 \leq i < s.buffer$.`size` $- 1$. From the code it follows that $s'.now = s.now + t'$ and $s'.buffer = s.buffer$. We claim that $s'.now \leq s'.buffer$.`head.st` and therefore the invarint holds at s'. Suppose this was not the case, that is $s'.now > s'.buffer$.`head.st`. Then there would exist $t'' \in \tau.dom$ such that $t'' < t'$ and $\tau(t'').now = s'.buffer$.`head.st`. Since $\tau(t'')$ satisfies the stopping condition for activity *d2a* therefore $\tau.ltime = t''$, which contradicts our assumption.□

4 Supervisory Controller

The supervisory controller has to ensure that the `Plant` state stays in the safe region **S** defined in equation (2). A second requirement of the supervisor is to interfere as little as possible with the user's controller. In the next section we informally discuss the relevance of several different regions in the state space, their actual definitions appear in the following section.

4.1 Supervisor Strategy

The design principle of the supervisor is simple: allow the user to be in control in all plant states from which the supervisor is guaranteed to restore the plant to a safe state; in all other states block the user's controller, perform recovery, and return control to the user. The issue here is to find

hybridautomaton `UsrCtrl`

actions
  input *sample* (θ_d^0 : `RAD` , θ_d^1 : `RADPS`),
  output *control* (u_d : `UTYPE`)

variables
  internal θ_u^0: `RAD` $:= 0$, θ_u^1 : `RADPS` $:= 0$,
    U_u : `UTYPE` $:= 0$,
    ready : `Bool` := false

discrete transitions
  input *sample* (θ_d^0 , θ_d^1)
  eff $\theta_u^0 := \theta_d^0$; $\theta_u^1 = \theta_d^1$
    $U_u :=$ **choose** $[U_{min}, U_{max}]$;
    ready := true

  output *control* (u_d)
  pre $(u_d = U_u) \wedge ready$
  eff *ready* := false

trajectories
  activity *void*
    evolve stop at *ready*

Figure 5: Specification of User's Controller

the *safe operating region* $\mathbf{U}$, that is, the largest set of states in which the user can be allowed to operate without threatening the safety of the plant.

To intuitively explain our choice of $\mathbf{U}$, let us first examine a few candidate regions which are not suitable. Clearly any safe operating region has to be a subset of $\mathbf{S}$. Also, $\mathbf{S}$ itself is not suitable because the plant has non-zero inertia and the output from the actuator is limited between U_{min} and U_{max}. Consider a region $\mathbf{C} \subseteq \mathbf{S}$, from which all trajectories are contained in $\mathbf{S}$, provided that the input to the plant is *correct*. Here correct means that the output from the actuator is U_{min}, U_{max} or 0, whichever is most suitable for the safety of the plant. The region $\mathbf{C}$ is still not a safe operating region, since the supervisor cannot change the output of the actuator sooner than $\tau_{\texttt{act}}$ due to the delay in the *buffer*. Therefore the supervisor has to look ahead into the future for at least $\tau_{\texttt{act}}$ time, in order to ensure that the actuator output is correct in the worst case. Let us consider the set of states $\mathbf{R} \subseteq \mathbf{C}$ from which all reachable states over a period of $\tau_{\texttt{eff}}$ are within $\mathbf{C}$, with any input to the plant. From states within $\mathbf{R}$, the supervisor can, if required, override the user controller and issue correct recovery commands such that all future states in the next $\tau_{\texttt{eff}}$ period are within $\mathbf{C}$, after which the correct commands of the supervisor appear at the actuator output, which in turn ensures safety. This region $\mathbf{R}$ is close to what we want, except that the supervisor cannot observe the plant state directly and for that it has to depend on the periodic updates from the sensors which are prone to errors. For a given sensed state (θ_s^0, θ_s^1), the actual plant state is in the set:

$$P(s) = \{s \mid \theta_s^0 - \epsilon_0 \leq s.\theta_p^0 \leq \theta_s^0 + \epsilon_0 \wedge \theta_s^1 - \epsilon_1 \leq s.\theta_p^1 \leq \theta_s^1 + \epsilon_1\},$$

Finally, taking the errors and the delay into account we define the region $\mathbf{U}$ as follows: An observed state s is in $\mathbf{U}$ if starting from any state in $P(s)$ all the reachable states over a Δ interval of time are in $\mathbf{R}$. In Section 5 we shall prove that this choice of $\mathbf{U}$ ensures safety of the plant.

Switching back to the user's controller from the supervisor is delayed until the supervisor brings the plant state within a *inner* region $\mathbf{I} \subseteq \mathbf{U}$. This asymmetry in the switching is introduced to prevent high frequency chattering between the user and the supervisory controller.

type MODES = { usr, sup }

hybridautomaton Actuator(τ_{act})

actions
  input *command* (u : UTYPE)
  internal *dequeue*

variables
  internal u_o : UTYPE := 0, $ready_d$: Bool := false,
    buffer : seq of (u:UTYPE, st:Real, m:MODE) := {}
  output analog U : UTYPE := 0,
  input analog *now* : Real

discrete transitions
  input *command* (u)
  eff *buffer* $+:= (u, now + \tau_{\mathsf{act}})$;
    $ready_d$:= true

  internal *dequeue*
  pre *buffer*.head.st $=$ *now* $\wedge$ $ready_d$
  eff u_o := *buffer*.head.v;
    buffer := *buffer*.tail;
    $ready_d$:= false

trajectories
  activity *d2a*
  evolve $U = u_o$
  stop at *buffer*.head.st $=$ *now*

Figure 6: Actuator and D/A conversion

4.2 Regions of Control

In the previous section we described the intuitive meanings of the regions **C**, **R**, **U**, and **I**; here we present their definitions.

$$\mathbf{C} \triangleq \{s \mid \theta_{min} \le s.\theta_p^0 \le \theta_{max} \wedge \Gamma^-(s.\theta_p^0, 0) \le s.\theta_p^1 \le \Gamma^+(s.\theta_p^0, 0)\}, \tag{3}$$

$$\mathbf{R} \triangleq \{s \mid \theta_{min} \le s.\theta_p^0 \le \theta_{max} \wedge \Gamma^-(s.\theta_p^0, \tau_{\mathsf{act}}) \le s.\theta_p^1 \le \Gamma^+(s.\theta_p^0, \tau_{\mathsf{act}})\}, \tag{4}$$

$$\mathbf{U} \triangleq \{s \mid \theta_{min} + \epsilon_0 \le s.\theta_s^0 \le \theta_{max} - \epsilon_0 \wedge U^-(s.\theta_s^0) \le s.\theta_s^1 \le U^+(s.\theta_s^0)\}, \tag{5}$$

$$\mathbf{I} \triangleq \{s \mid \theta_{min} + \epsilon_0 \le s.\theta_s^0 \le \theta_{max} - \epsilon_0 \wedge I^-(s.\theta_s^0) \le s.\theta_s^1 \le I^+(s.\theta_s^0)\}. \tag{6}$$

Where the functions Γ^+ and Γ^-, are as follows:

$$\Gamma^+(\theta, \mathcal{T}) = -U_{mag}\mathcal{T} + \sqrt{2(\Omega^2 \cos\theta_{max} - U_{min})(\theta_{max} - \theta + \frac{1}{2}U_{mag}\mathcal{T}^2)}, \tag{7}$$

$$\Gamma^-(\theta, \mathcal{T}) = U_{mag}\mathcal{T} - \sqrt{2(U_{max} - \Omega^2)(\theta - \theta_{min} + \frac{1}{2}U_{mag}\mathcal{T}^2)}, \tag{8}$$

$$U_{mag} = U_{max} - U_{min} \tag{9}$$

$$U^+(\theta) = -\epsilon_1 + \Gamma^+(\theta + \epsilon_0, \tau_{\mathsf{act}} + \Delta), \tag{10}$$

$$U^-(\theta) = +\epsilon_1 + \Gamma^-(\theta - \epsilon_0, \tau_{\mathsf{act}} + \Delta), \tag{11}$$

$$I^+(\theta) = -2\epsilon_1 + \Gamma^+(\theta + 2\epsilon_0, \tau_{\mathsf{act}} + \Delta), \tag{12}$$

$$I^-(\theta) = +2\epsilon_1 + \Gamma^-(\theta - 2\epsilon_0, \tau_{\mathsf{act}} + \Delta). \tag{13}$$

From the definitions of Γ^+ and Γ^- the following properties can be shown,

Property 1 *Over the interval* $-\frac{\pi}{2} \le \theta \le \frac{\pi}{2}$ *the following properties hold :*

1. $\Gamma^{+}(\theta, \mathcal{T})$ *and* $\Gamma^{-}(\theta, \mathcal{T})$ *are monotonically decreasing with respect to* θ.

2. $\Gamma^{+}(\theta, \mathcal{T})$ *is monotonically decreasing with respect to* $\mathcal{T}$. $(\mathcal{T} \geq 0)$.

3. $\Gamma^{-}(\theta, \mathcal{T})$ *is monotonically increasing with respect to* $\mathcal{T}$. $(\mathcal{T} \geq 0)$.

4. $\Gamma^{+}(\theta_{max}, \mathcal{T}) < 0$ *and* $\Gamma^{-}(\theta_{min}, \mathcal{T}) > 0$ *for* $\mathcal{T} > 0$.

Using these properties, the following sequence of containments can be proved.

Property 2 $\mathbf{I} \subseteq \mathbf{U} \subseteq \mathbf{R} \subseteq \mathbf{C} \subseteq \mathbf{S}$

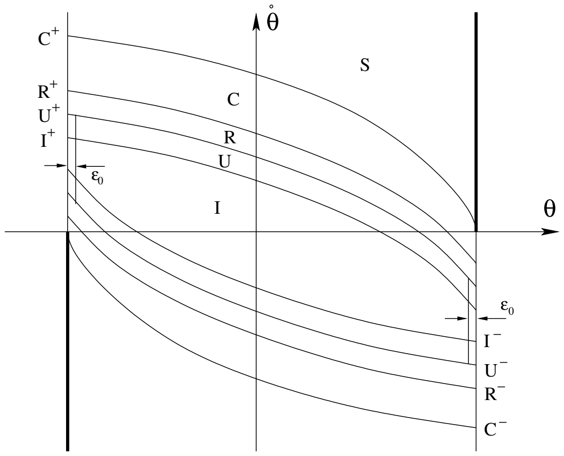

Figure 7: Regions in the statespace.

4.3 Supervisor Automaton

The `Supervisor` automaton (Fig. 8) copies the observed plant state into internal variables θ_s^0 and θ_s^1 when the *sample* action occurs. Based on this state information the tentative output U_{sup} to the actuator is decided. When the *control* action occurs, the supervisor copies the user's command into another internal variable U_{usr} and sets output command U_s and *mode* for the next Δ interval based on (θ_s^0, θ_s^1) and the current value of *mode*. If *mode* is `usr` and the observed state is in $\mathbf{U}$ then *mode* remains unchanged and U_s is set to U_{usr}. If the present state is not in $\mathbf{U}$ then *mode* is changed to `sup` and the U_s is set to U_{sup}. If *mode* = `sup` then U_s is copied from U_{sup} and the mode changes only when (θ_s^0, θ_s^1) is in $\mathbf{I}$. The *control* action enables the *command* output action by setting the $ready_c$ flag.

5 Analysis of Helicopter System

In this section we verify the safety of the helicopter system with the supervisory controller. Let $\mathcal{A}$ denote the composition of the `Plant`, `Sensor`, `UsrCtrl`, `Actuator`, and the `Supervisor` automata. The helicopter system is safe if all the reachable states of the $\mathcal{A}$ are contained within the region $\mathbf{S}$. We assume the following relationships amongst the different parameters in the model:

hybridautomaton `Supervisor`

actions
  input *sample* (θ_d^0: `RAD` θ_d^1: `RADPS`),
  input *control* (u_d : `UTYPE`),
  output *command* (u_d : `UTYPE`, m : `MODES`)

variables
  internal θ_s^0 : `RAD` $:= 0$, θ_s^1 : `RADPS` $:= 0$,
    U_{sup}, U_{usr}, U_s : `UTYPE` $:= 0$,
  internal $ready_c$: `Bool` := false, $mode$: `MODES` := `usr`
  internal analog rt : `Real` $:= 0$;

discrete transitions
  input *sample* (θ_d^0, θ_d^1)
  eff $\theta_s^0 := \theta_d^0$; $\theta_s^1 := \theta_d^1$;
    if $\theta_s^1 \geq I^+(\theta_s^0)$ **then** $U_{sup} := U_{min}$
    elseif $\theta_s^1 \leq I^-(\theta_s^0)$ **then** $U_{sup} := U_{max}$ **fi**

  output *command* (u_d, m)
  pre $ready_c \wedge (u_d = U_s) \wedge m = mode$
  eff $ready_c :=$ false

  input *control* (u_d)
  eff $U_{usr} := u_d$; $ready_c :=$ true
    if $mode =$ `usr` **then**
      if $(\theta_s^0, \theta_s^1) \in \mathbf{U}$ **then** $U_s := U_{usr}$
      else $U_s := U_{sup}$; $mode :=$ `sup` **fi**
    elseif $mode =$ `sup` **then**
      if $(\theta_s^0, \theta_s^1) \in \mathbf{I}$ **then** $U_s := U_{usr}$; $mode :=$ `usr`
      else $U_s := U_{sup}$ **fi fi**

trajectories
  activity *supervisor*
    when $mode =$ `sup`
    evolve $d(rt) = 1$ **stop at** $ready_c$

  activity *user*
    when $mode =$ `usr`
    evolve $rt = 0$ **stop at** $ready_c$

Figure 8: HIOA specification of supervisor automaton

1. $\theta_{min} < 0 < |\theta_{min}| < \theta_{max}$,
2. $U_{max} > \Omega^2$, $U_{min} \leq 0$.
3. For any *sample* action $s \xrightarrow{\pi} s'$,
   if $s.\theta_s^1 > I^+(s.\theta_s^0)$ then, $s'.\theta_s^1 \geq I^-(s'.\theta_s^0)$, and
   if $s.\theta_s^1 < I^-(s.\theta_s^0)$ then, $s'.\theta_s^1 \leq I^+(s'.\theta_s^0)$.

The first two are facts derived from the dimensions of the actual system. The third constraint is required to prevent the supervisor from holding control forever by jumping between the region above I^+ and the region below I^- over a single Δ interval of time. This condition imposes certain bounds on the values of $\tau_{\mathsf{act}}, \Delta$, and U_{mag}.

All the invariants in this paper are either derived from other invariants or proved by induction on the length of a closed execution of automaton $\mathcal{A}$. The induction for an invariant $\mathcal{I}$ consists of a base case, and an induction step. The base case tests that $\mathcal{I}$ is satisfied at all the initial sates of $\mathcal{A}$. The induction step consists of : (1) a discrete part—to test that for every discrete step $s \xrightarrow{\pi} s'$, from any reachable state s, preserves $\mathcal{I}$, and (2) a continuous part—to test that for any closed trajectory τ, starting from a reachable state s, $\mathcal{I}$ is preserved at the $\tau.lsate$. We shall use s and s' to denote the pre and the post states of discrete transitions, as well as $fstate$ and $lstate$ of closed trajectories, as will be clear from the context.

In the remainder of this section we first present some preliminary properties of the system, then we state the key invariants of $\mathcal{A}$, and present the proof of safety in the user and the supervisor modes.

5.1 Some Preliminary Properties

Property 3 *The discrete variables of $\mathcal{A}$ are not changed over any closed trajectory τ.*

Proof: Follows from the code of the components of $\mathcal{A}$.

Property 4 *For any discrete step $s \xrightarrow{\pi} s'$ of automaton $\mathcal{A}$, $s'.\theta_p^0 = s.\theta_p^0$ and $s'.\theta_p^1 = s.\theta_p^1$.*

Proof: From the code of `Plant` it follows that θ_p^0 and θ_p^1 are not altered by any discrete step.
Let us define a derived state variable $time\_left$ at a given state s as : $s.time\_left \triangleq s.next\_time - s.now$.

Invariant 5.1 *In every reachable state s of $\mathcal{A}$, $0 \leq s.time\_left \leq \Delta$.*

Proof: The base case holds trivially because $s.time\_left = \Delta$. For the discrete part of the induction we consider transitions $s \xrightarrow{\pi} s'$, where $\pi = sample$ action. Other actions do not alter any of the variables in the invariant. It follows from the code that $s.now = s.next\_time$, $s'.next\_time = s.next\_time + \Delta$, and $s'.now = s.now$. Therefore $s'.next\_time - s'.now = \Delta$.

For the continuous part, consider a closed trajectory τ with limit time $k \geq 0$, let $s.time\_left = t \in [0, \Delta]$. Let us assume for the sake of contradiction that $k > t$. Then $\tau \downarrow now(t) = \tau \downarrow next\_time(t)$, which satisfies the stopping condition of *read* activity, therefore $t = \tau.ltime$. This contradicts our assumption, and therefore $k \leq t$. From activity *read*, $s'.time\_left = t - k$. As $0 \leq t \leq \Delta$, we have $0 \leq s'.time\_left \leq \Delta$. □

Corollary 5.1 *The limit time of every trajectory of $\mathcal{A}$ is upper bounded by Δ.*

Lemma 5.1 *In any execution of $\mathcal{A}$, sample, control, and command actions occur only when $now = n\Delta$, for some integer $n > 0$.*

Corollary 5.2 *In every reachable state s, for all $0 \leq i < s.buffer.\mathtt{size} - 1$, $s.buffer\mathtt{[i+1].st} = s.buffer\mathtt{[i].st} + \Delta$.*

Lemma 5.2 *In any execution of $\mathcal{A}$, a dequeue action occurs when $timer = \tau_{\mathtt{act}} + n\Delta$, for every integer $n \geq 0$.*

Invariant 5.2 *In any reachable state s, $s.buffer.\mathtt{size} \leq \lceil \frac{\tau_{\mathtt{act}}}{\Delta} \rceil$.*

Proof: Consider any reachable state s such that $s.buffer \neq \{\}$. From Corollary 5.2, $s.buffer\mathtt{[i+1].st} = s.buffer\mathtt{[i].st} + \Delta$, for all $0 \leq i < s.buffer.\mathtt{size}$. From Invariant 3.1, $s.buffer.\mathtt{last.st} - s.buffer.\mathtt{head.st} \leq \tau_{\mathtt{act}}$. The property follows by showing a simple contradiction.
We define the quantity M as the maximum possible size of *buffer* in any reachable state, $M \triangleq \lceil \frac{\tau_{\mathtt{act}}}{\Delta} \rceil$.

5.2 User Mode

In this section we prove that $\mathcal{A}$ is safe in the user mode. We define a set of regions $\mathbf{A}_t$ for $0 \leq t \leq \Delta$,

$$\mathbf{A}_t \triangleq \{s \mid \theta_{min} + \leq s.\theta_p^0 \leq \theta_{max} \wedge \Gamma^-(s.\theta_p^0, \tau_{\mathtt{eff}} + t) \leq s.\theta_p^1 \leq \Gamma^+(s.\theta_p^0, \tau_{\mathtt{eff}} + t)\}, \tag{14}$$

and we prove the following properties.

Lemma 5.3 *The region $\mathbf{A}_t$ satisfies the following:*

1. $\mathbf{A}_0 = \mathbf{R}$,

2. $\mathbf{U} \subseteq \mathbf{A}_\Delta$,

3. If $0 \le t \le t' \le \Delta$ *then* $A_{t'} \subseteq A_t$.

Proof: For part 1, set $t = 0$ in equation (14).
For part 2, $\theta_s^0 - \epsilon_0 \le \theta_p^0 \le \theta_s^0 + \epsilon_0$ and $\theta_s^1 - \epsilon_1 \le \theta_p^1 \le \theta_s^1 + \epsilon_1$. Setting $t = \Delta$ we have:

$$\mathbf{A}_\Delta = \{s \mid \theta_{min} - \epsilon_0 \le s.\theta_s^0 \le \theta_{max} + \epsilon_0 \wedge \Gamma^-(s.\theta_p^0, \tau_{\mathtt{eff}} + \Delta) - \epsilon_1 \le s.\theta_s^1 \le \Gamma^+(s.\theta_p^0, \tau_{\mathtt{eff}} + \Delta) + \epsilon_1\}.$$

From Property 1, $\theta_p^0 \ge \theta_s^0 - \epsilon_0 \Rightarrow \Gamma^-(\theta_p^0, y) \le \Gamma^-(\theta_s^0 - \epsilon_0, y)$ and $\theta_p^0 \le \theta_s^0 + \epsilon_0 \Rightarrow \Gamma^+(\theta_p^0, y) \ge \Gamma^+(\theta_s^0 + \epsilon_0, y)$. Therefore,

$$\{s \mid \theta_{min} \le s.\theta_p^0 \le \theta_{max} \wedge \Gamma^-(s.\theta_s^0 - \epsilon_0, \tau_{\mathtt{eff}} + \Delta) + \epsilon_1 \le s.\theta_p^1 \le \Gamma^+(s.\theta_s^0 + \epsilon_0, \tau_{\mathtt{eff}} + \Delta) - \epsilon_1\} \subseteq \mathbf{A}_\Delta.$$

The left hand side is equal to $\mathbf{U}$ as defined in equation (5).
For part 3, we observe that in equation (14) Γ^+ and Γ^- are monotonically decreasing and monotonically increasing respectively with respect to t. Therefore if $0 \le t \le t' \le \Delta$ then $\mathbf{A}_{t'} \subseteq \mathbf{A}_t$. □

Lemma 5.4 *For any closed trajectory τ of $\mathcal{A}$, if $\tau.fstate \in \mathbf{A}_t$ then $\tau.lstate \in \mathbf{A}_{t-ltime(\tau)}$.*

Proof: Consider a closed trajectory τ. Assume that $s \in \mathbf{A}_t$. From the definition of $\mathbf{A}_t$ it follows that, $\theta_{min} \le s.\theta_p^0 \le \theta_{max}$ and $\Gamma^-(s.\theta_p^0,\ \tau_{\mathtt{eff}} + t) \ \le\ s.\theta_p^1 \ \le\ \Gamma^+(s.\theta_p^0,\ \tau_{\mathtt{eff}} + t)$. We conservatively estimate s' by considering the maximum and the minimum input U to `Plant`. First considering the maximum positive input, $U = U_{max}$, from the state model of `Plant` we get the upper bound on the acceleration at any state s'' in τ :
$d(s''.\theta_p^1) \le -\Omega^2 \cos\theta_{max} + U_{max}$. Integrating from t to t',

$$s'.\theta_p^1 \le (U_{max} - \Omega^2 \cos\theta_{max})t' + s.\theta_p^1, \tag{15}$$

$$s'.\theta_p^0 \le \frac{1}{2}(U_{max} - \Omega^2 \cos\theta_{max}){t'}^2 + s.\theta_p^1 t' + s.\theta_p^0. \tag{16}$$

Simplifying and using the definition of Γ^+ we get the bounds on $s'.\theta_p^0$ and $s'.\theta_p^1$.

$$s'.\theta_p^0 \le \theta_{max},\ \textit{and} \tag{17}$$

$$s'.\theta_p^1 \le \Gamma^+(s'.\theta_p^0,\ \tau_{\mathtt{eff}} + t - t'). \tag{18}$$

Likewise considering $U = U_{min}$, we get the lower bounds on $s'.\theta_s^0$ and $s'.\theta_s^1$.

$$s'.\theta_p^0 \ge \theta_{min},\quad \textit{and} \tag{19}$$

$$s'.\theta_p^1 \ge \Gamma^-(s'.\theta_p^0,\ \tau_{\mathtt{eff}} + t - t'). \tag{20}$$

Combining equations (17) (18) (19) and (20) we have $s' \in \mathbf{A}_{t-t'}$. □

Invariant 5.3 *In any reachable state s, $s.mode = \mathtt{usr} \ \wedge\ \neg s.ready \Rightarrow s \in \mathbf{A}_{s.time\_left}$.*

Proof: The base case holds because for any initial state s, $s.time\_left = \Delta$ and $s \in \mathbf{U} \subseteq \mathbf{A}_\Delta$. We have to consider three possible cases for discrete steps $s \xrightarrow{\pi} s'$: if $\pi = sample(x,y)$, then $s'.ready = \mathsf{true}$ and the invariant holds vacuously. if $\pi = control(x)$, assume $s'.mode = \texttt{usr}$, we have two sub-cases: if $s.mode = \texttt{usr}$, then from the code of the *control* action, $s \in \mathbf{U} \Rightarrow s' \in \mathbf{U} \subseteq \mathbf{A}_\Delta$. Since $s'.time\_left \leq \Delta$, $s' \in \mathbf{A}_{s.time\_left}$. Otherwise, if $s.mode = \texttt{sup}$, then $s \in \mathbf{I} \Rightarrow s' \in \mathbf{I} \subseteq \mathbf{A}_\Delta$, which implies that $s' \in \mathbf{A}_{s'.time\_left}$. if $\pi = command(x)$, assume $s'.mode = \texttt{usr} \wedge \neg s'.ready$, then $s.mode = \texttt{usr} \wedge \neg s.ready$. By inductive hypothesis $s \in \mathbf{A}_{s.time\_left}$, therefore $s' \in \mathbf{A}_{s'.time\_left}$.

For the continuous part, consider a closed trajectory τ with $\tau.ltime = t'$. Assume $s'.mode = \texttt{usr} \wedge \neg s'.ready$. As the valuations of *mode* and *ready* do not change over τ, $s.mode = \texttt{usr} \wedge \neg s.ready$. From the inductive hypothesis $s \in \mathbf{A}_{s.time\_left}$. Using Lemma 5.4, $s' \in \mathbf{A}_{s.time\_left} - t' = \mathbf{A}_{s'.time\_left}$. □

Invariant 5.4 *In any reachable state s, $s.mode = \texttt{usr} \Rightarrow s \in \mathbf{R}$.*

Proof: The base case holds because all initial states are in $\mathbf{U}$ and $\mathbf{U} \subseteq \mathbf{R}$. Consider any discrete transition $s \xrightarrow{\pi} s'$, with $s'.mode = \texttt{usr}$. We split the proof into two cases: If $\neg s'.ready$, then using Invariant 5.3, $s' \in \mathbf{A}_{s'.time\_left} \subseteq \mathbf{R}$. On the other hand, if $s'.ready$, then $\pi \neq control$, and $s.mode = \texttt{usr}$ since only the *control* action can change *mode*. So from the inductive hypothesis $s \in \mathbf{R}$. It follows that $s' \in \mathbf{R}$ from the Property 4.

For the continuous part consider a closed trajectory τ with $\tau.fstate = s$, $\tau.lstate = s'$, and $s'.mode = \texttt{usr}$. Once again there are two cases, if $\neg s'.ready$ then $s' \in \mathbf{R}$ by Invariant 5.3. Else if $s'.ready$, then $s.ready$ and $s.mode = \texttt{usr}$ because *ready* and *mode* does not change over the trajectories. Since s satisfies the stopping condition for activity *void* in `UsrCtrl`, therefore τ is a point trajectory, that is, $s' = s$. From the inductive hypothesis, $s \in \mathbf{R}$. Therefore $s' \in \mathbf{R}$. □

5.3 Supervisor Mode

The first invariant in this section states that in all reachable states that have *ready* set to false, if the sensed plant state is within I^+ and I^-, then the system is in the user mode.

Invariant 5.5 *In any reachable state s, $I^-(s.\theta_s^0) \leq s.\theta_s^1 \leq I^+(s.\theta_s^0) \ \wedge \ \neg s.ready \Rightarrow s.mode = \texttt{usr}$.*

Proof: The base case holds from initialization. Consider discrete steps $s \xrightarrow{\pi} s'$ with $I^-(s'.\theta_s^0) \leq s'.\theta_s^1 \leq I^+(s'.\theta_s^0)$. If $\pi = sample$, then $s'.ready = \mathsf{true}$ and therefore the invariant holds vacuously. If $\pi = control$, then it follows from the code that $I^-(s.\theta_s^0) \leq s.\theta_s^1 \leq I^+(s.\theta_s^0)$ and therefore $s'.mode = \texttt{usr}$. For *command* and *dequeue* actions and also for any trajectory of $\mathcal{A}$, the invariant is preserved because none of the variables involved in it are altered. □

Invariant 5.6 *In any reachable state s,*
if $s.\theta_s^1 > I^+(s.\theta_s^0)$ then $s.U_{sup} = U_{min}$, and
if $s.\theta_s^1 < I^+(s.\theta_s^0)$ then $s.U_{sup} = U_{max}$.

Proof: Immediate from the code of *sample* action. None of the other actions or activities alter any of the variables involved in the invariant.

Invariant 5.7 *In any reachable state s, $s.rt = n\Delta - s.time\_left$, for some integer $n \geq 1$.*

Proof: For the base case: $s.time\_left = \Delta$, $s.rt = 0$ and therefore the invariant holds for $n = 1$. Consider discrete step $s \xrightarrow{\pi} s'$ with $\pi = sample$. From the induction hypothesis it follows that $s.rt = n\Delta - s.time\_left$, for some $n \geq 1$; fix n. From the code it follows that $s'.rt = s.rt$,

$s.time\_left = 0$ and $s'.time\_left = \Delta$. Hence $s'.rt = (n+1)\Delta - s'.time\_left$. The invariant is preserved by all other discrete actions because the variables rt and $time\_left$ are not changed by them.

Consider a closed trajectory τ with $\tau.ltime = t'$. From induction hypothesis it follows that $s.rt = n\Delta - s.time\_left$, for some $n \geq 1$; fix n. Therefore $s'.rt = s.rt + t' = n\Delta - s.time\_left + t' = n\Delta - s'.time\_left$. $\square$

We define two predicates $\mathcal{Q}_k^+$ and $\mathcal{Q}_k^-$ that capture the progress made by the system while the actuator delays the delivery of commands issued by the supervisor. A state s satisfies $\mathcal{Q}_k^+$ (or $\mathcal{Q}_k^-$), if the last k commands in $s.buffer$ are equal to U_{min} (or U_{max} respectively). More formally , for any $k \geq 0$,

$$\mathcal{Q}_k^+(s) \triangleq \forall i,\ max(0, s.buffer.\mathtt{size} - k) \leq i < s.buffer.\mathtt{size}, \quad s.buffer[i].\mathtt{u} = U_{min}, \text{ and}$$
$$\mathcal{Q}_k^-(s) \triangleq \forall i,\ max(0, s.buffer.\mathtt{size} - k) \leq i < s.buffer.\mathtt{size}, \quad s.buffer[i].\mathtt{u} = U_{max}.$$

Clearly, for all $k > 0$, $\mathcal{Q}_k^+(s)$ implies $\mathcal{Q}_{k-1}^+(s)$, and therefore for any $k \geq s.buffer.\mathtt{size}$, $\mathcal{Q}_k^+(s)$ implies that $\mathcal{Q}_j^+(s)$ holds for all $j < s.buffer.\mathtt{size}$. Similar results hold for $\mathcal{Q}_k^-$. The next invariant states that every reachable state s in the supervisor mode, satisfies either $\mathcal{Q}^+_{\lceil \frac{s.rt}{\Delta} \rceil}(s)$ or $\mathcal{Q}^-_{\lceil \frac{s.rt}{\Delta} \rceil}(s)$, depending on whether s is above I^+ or below I^- respectively. In addition if $s.ready_d$ is $\mathtt{true}$, that is, s is in between a *command* action and a *dequeue* action, then $\mathcal{Q}^+_{\lceil \frac{s.rt}{\Delta} \rceil + 1}(s)$ or $\mathcal{Q}^-_{\lceil \frac{s.rt}{\Delta} \rceil + 1}(s)$ holds, depending on the location of s with respect to I^+ and I^-.

Invariant 5.8 *In any reachable state s, such that $s.mode = \mathtt{sup}$:*

1. *If $s.\theta_s^1 > I^+(s.\theta_s^0)$ then*
   (a) $\mathcal{Q}^+_{\lceil \frac{s.rt}{\Delta} \rceil}(s)$,
   (b) *If $ready_d$ then $\mathcal{Q}^+_{\lceil \frac{s.rt}{\Delta} \rceil + 1}(s)$, and*
2. *If $s.\theta_s^1 < I^-(s.\theta_s^0)$ then*
   (a) $\mathcal{Q}^-_{\lceil \frac{s.rt}{\Delta} \rceil}(s)$,
   (b) *If $ready_d$ then $\mathcal{Q}^-_{\lceil \frac{s.rt}{\Delta} \rceil + 1}(s)$, and*

Proof: We shall prove part 1 of the invariant. The proof for part 2 is similar to that of part 1. The base case holds trivially because $s.mode = \mathtt{usr}$. We consider the discrete steps $s \xrightarrow{\pi} s'$ with $s'.mode = \mathtt{sup}$ and $s'.\theta_s^1 > I^+(s'.\theta_s^0)$.

Case 1: $\pi = sample$. Since $s.ready = \mathsf{false}$ and $s.mode = \mathtt{sup}$, it follows from the contrapositive of Invariant 5.5 that $s.\theta_s^1 > I^+(s.\theta_s^0)$ or $s.\theta_s^1 < I^-(s.\theta_s^0)$. According to Assumption 3, $s.\theta_s^1 \geq I^-(s.\theta_s^0)$, therefore $s.\theta_s^1 > I^+(s.\theta_s^0)$. Part 1(a): From part 1(a) of the inductive hypothesis it follows that $\mathcal{Q}^+_{\lceil \frac{s.rt}{\Delta} \rceil}(s)$ holds. Since *buffer* is not changed by π therefore $\mathcal{Q}^+_{\lceil \frac{s'.rt}{\Delta} \rceil}(s')$ holds.

Part 1(b): Assume $s'.ready_d = \mathtt{true}$. Since *sample* does not change $ready_d$, it follows that $s.ready_d = \mathtt{true}$. Therefore from the inductive hypothesis it follows that $\mathcal{Q}^+_{\lceil \frac{s.rt}{\Delta} \rceil + 1}(s)$ holds. Since *buffer* is not changed by π therefore $\mathcal{Q}^+_{\lceil \frac{s'.rt}{\Delta} \rceil + 1}(s')$ holds.

Case 2: $\pi = control$. If $s.mode = \mathtt{sup}$. The invariant is preserved since π does not change any of the variables involved other than *mode*. If $s.mode = \mathtt{usr}$ then $s.rt = 0 = s'.rt$. The invariant is satisfied because $\mathcal{Q}_0^+$ is trivially $\mathtt{true}$.

Case 3: $\pi = command$. Part 1(b): From the code it follows that $s.mode = \texttt{sup}$ and $s.\theta_s^1 > I^+(s.\theta_s^0)$. Therefore it follows from Invariant 5.6 that $s.U_{sup} = U_{min}$. Since $s'.buffer = s.buffer + (s.U_{sup}, s.now + \tau_{\mathsf{act}})$, and $\mathcal{Q}^+_{\lceil \frac{s.rt}{\Delta} \rceil}(s)$ holds from the inductive hypothesis, therefore it follows that $\mathcal{Q}^+_{\lceil \frac{s'.rt}{\Delta} \rceil + 1}(s')$ holds.

Part 1(a) follows from the above because $\mathcal{Q}^+_{\lceil \frac{s'.rt}{\Delta} \rceil + 1}(s')$ implies that $\mathcal{Q}^+_{\lceil \frac{s'.rt}{\Delta} \rceil}(s')$ holds.

Case 4: $\pi = dequeue$. From the code it follows that $s.mode = \texttt{sup}$, $s.\theta_s^1 > I^+(s.\theta_s^0)$, $s'.buffer = s.buffer.\texttt{tail}$, and that $s.ready_d = \mathsf{true}$. Part 1(b): From the inductive hypothesis it follows that $\mathcal{Q}^+_{\lceil \frac{s.rt}{\Delta} \rceil + 1}(s)$ holds, which implies that $\mathcal{Q}^+_{\lceil \frac{s'.rt}{\Delta} \rceil}(s')$ holds.

Part 1(b): From the code it follows that $s'.ready = \mathsf{false}$ therefore the invariant holds trivially.

For the continuous part, consider a closed trajectory τ, with $t' = \tau.ltime$, $s'.mode = \texttt{sup}$ and $s'.\theta_s^1 > I^+(s'.\theta_s^0)$. From the code it follows that $s'.buffer = s.buffer$, $s.\theta_s^1 > I^+(s.\theta_s^0)$ and $s'.rt = s.rt + t'$. Using Invariant 5.7 $s.rt$ can be written as $s.rt = n\Delta - s.time\_left$ for some $n \geq 1$; fix n. Therefore $s'.rt = n\Delta - s.time\_left + t' = n\Delta - s'.time\_left$. Since $0 \leq s.time\_left \leq \Delta$ and $0 \leq s'.time\_left \leq \Delta$, therefore $\lceil \frac{s.rt}{\Delta} \rceil = \lceil \frac{s'.rt}{\Delta} \rceil = n$

Part 1(a): From part 1(a) of the inductive hypothesis it follows that $\mathcal{Q}^+_n(s)$ holds. Since *buffer* is not changed over τ it follows that $\mathcal{Q}^+_n(s')$ holds.

Part 1(b): Assume $s'.ready_d = \mathsf{true}$. Therefore $s.ready_d = \mathsf{true}$. From part 1(b) of the inductive hypothesis it follows that $\mathcal{Q}^+_{n+1}(s)$ holds and since *buffer* is not changed over τ it follows that $\mathcal{Q}^+_{n+1}(s')$ holds. $\Box$

The next invariant formalizes the notion that after a certain τ_{act} period of time in the supervisor mode the input to the plant is correct.

Invariant 5.9 *In any reachable state s with $s.mode = \texttt{sup} \wedge s.rt \geq \tau_{\mathsf{act}}$*

1. *If $s.\theta_s^1 > I^+(s.\theta_s^0)$ then s.buffer$.\texttt{head.u} = U_{min}$, and*
2. *If $s.\theta_s^1 < I^+(s.\theta_s^0)$ then s.buffer$.\texttt{head.u} = U_{max}$.*

Proof: We shall prove part 1 of the invariant. Consider a reachable state s and assume that $s.mode = \texttt{sup}$, $s.rt > \tau_{\mathsf{act}}$ and $s.\theta_s^1 > I^+(s.\theta_s^0)$. From part 1 of Invariant 5.8 it follows that $\mathcal{Q}^+_{\lceil \frac{\tau_{\mathsf{act}}}{\Delta} \rceil}(s)$ holds. From Invariant 5.2 it is known that the maximum size of *buffer* is $\lceil \frac{\tau_{\mathsf{act}}}{\Delta} \rceil$. Therefore it follows from the definition of $\mathcal{Q}^+$ that $s.buffer.\texttt{head} = U_{min}$. $\Box$

Invariant 5.10 *In any reachable state s, such that $s.mode = \texttt{sup}$ and $s.rt > \tau_{\mathsf{act}}$*
1. If $s.\theta_s^1 > I^+(s.\theta_s^0)$ then $s.U = U_{min}$, and
2. If $s.\theta_s^1 < I^-(s.\theta_s^0)$ then $s.U = U_{max}$,

Proof: We shall prove part 1 of the invariant. The proof of part 2 is similar to that of part 1. The base case is trivially true because $s.mode = \texttt{usr}$. Consider discrete transitions $s \xrightarrow{\pi} s'$ with $s'.mode = \texttt{sup}$, $s'.rt > \tau_{\mathsf{act}}$, and $s.\theta_s^1 > I^+(s.\theta_s^0)$. Since none of the discrete steps change rt, it follows that $s.rt > \tau_{\mathsf{act}}$.

Case 1: $\pi = sample$. Since $s.ready$ is false and $s.mode = \mathtt{sup}$, it follows from the contrapositive of Invariant 5.5 that $s.\theta_s^1 > I^+(s.\theta_s^0)$ or $s.\theta_s^1 < I^-(s.\theta_s^0)$. According to Assumption 3, $s.\theta_s^1 \geq I^-(s.\theta_s^0)$, therefore $s.\theta_s^1 > I^+(s.\theta_s^0)$. From part 1 of the inductive hypothesis it follows that $s.U = U_{min}$. Since U is not changed by π, therefore $s'.U = U_{min}$.

Case 2: $\pi = control$. We claim that $s.mode = \mathtt{sup}$. The invariant is preserved since π does not change any of the variables involved other than $mode$. If $s.mode = \mathtt{usr}$ then $s.rt = 0 = s'.rt$, which contradicts our assumption that $s'.rt > \tau_{\mathtt{act}}$.

Case 3: $\pi = command$. From the code it follows that $s.mode = \mathtt{sup}, s'.U = s.U$ and $s.\theta_s^1 > I^+(s.\theta_s^0)$. Therefore From part 1 of the inductive hypothesis it follows that $s'.U = s.U = U_{min}$.

Case 4: $\pi = dequeue$. From the code it follows that $s.mode = \mathtt{sup}$ and $s.\theta_s^1 > I^+(s.\theta_s^0)$. From part 1 of Invariant 5.8 it follows that $Q^+_{\lceil \frac{s.rt}{\Delta} \rceil}$ holds. Since $s.rt > \tau_{\mathtt{act}}$, therefore $s.buffer.\mathtt{head.u}= U_{min}$, by Invariant 5.9. It follows from the code that $s'.U = U_{min}$.

For the continuous part of the induction consider a closed trajectory τ with $\tau.ltime = t'$. Assume $s'.mode = \mathtt{sup}$, $s'.rt \geq \tau_{\mathtt{act}}$ and $s'.\theta_s^1 > I^+(s'.\theta_s^0)$. We claim that $s.rt \geq \tau_{\mathtt{act}}$. Since U, $mode$, θ_s^0 and θ_s^1 do not change over τ, therefore it follows from the inductive hypothesis that $s'.U = U_{min}$.

Contrary to our claim, if $s.rt < \tau_{act}$, then there exists a $t'' \in \tau.dom$, such that $t'' < t'$ and $\tau(t'').rt = \tau_{\mathtt{act}}$. From Lemma 5.2 it follows that such a t'' would have to be equal to $\tau.ltime$ because the stopping condition of activity $d2a$ would be enabled at $\tau(t'')$. This contradicts our assumption $\tau.ltime = t'$. □

We split the execution in the supervisor mode into (a) a *settling phase* of length $\tau_{\mathtt{act}}$ in which the input U to the plant is arbitrary, and (b) a variable length *recovery phase* during which $rt > \tau_{\mathtt{act}}$ and the input to the plant is correct, that is, in accordance with Invariant 5.10.

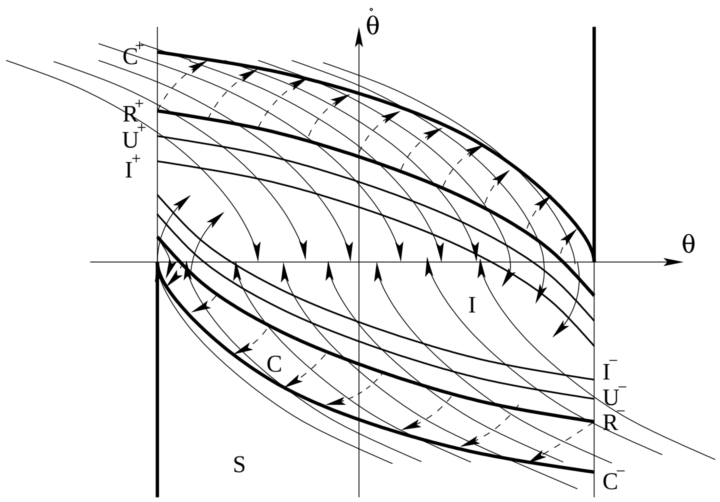

Figure 9: Trajectories in the settling (dashed lines) and recovery(solid lines) periods.

5.4 Settling Phase

We define a set of regions for $0 \leq t \leq \tau_{\mathtt{act}}$:

$$\mathbf{B}_t \triangleq \{s \mid \theta_{min} \leq s.\theta_p^0 \leq \theta_{max} \ \wedge \ \Gamma^-(s.\theta_p^0, \ \tau_{\mathtt{act}} - t) \leq s.\theta_p^1 \leq \Gamma^+(s.\theta_p^0, \ \tau_{\mathtt{act}} - t)\}. \tag{21}$$

Lemma 5.5 *The region* $\mathbf{B}_t$ *satisfies the following:*

1. $\mathbf{B}_0 = \mathbf{R}$,
2. $\mathbf{B}_{\tau_{\mathtt{act}}} = \mathbf{C}$,
3. *If* $0 \leq t \leq t' \leq \tau_{\mathtt{act}}$ *then* $\mathbf{B}_t \subseteq \mathbf{B}_{t'}$.

Proof: Parts 1 and 2 are proved by setting $t = 0$, and $t = \tau_{\mathtt{act}}$ in equation (21) respectively. Since $t \leq t'$, part 3 follows from Property 1.

Invariant 5.11 *For any reachable state* s*, if* $s.mode = \mathtt{sup} \wedge s.rt \leq \tau_{\mathtt{act}}$ *then* $s \in \mathbf{B}_{s.rt}$.

Proof: The base case holds trivially because $s.mode = \mathtt{usr}$. For the discrete part, consider discrete transitions $s \xrightarrow{\pi} s'$ with $s'.mode = \mathtt{sup}$. If $\pi = control$ there are two subcases: if $s.mode = \mathtt{sup}$ then from the induction hypothesis it follows that $s' \in \mathbf{B}_{s'.rt}$. Otherwise $s.mode = \mathtt{usr}$, and $s \in \mathbf{R}$ by Invariant 5.3. From Property 4 it follows that $s' \in \mathbf{R}$. Since $\mathbf{R} = \mathbf{B}_0 \subseteq \mathbf{B}_{s'.rt}$ for any $s'.rt \geq 0$ therefore the invariant holds at s'.

Consider a closed trajectory τ with $t' = \tau.ltime$. Assume $s'.mode = \mathtt{sup}$ and $s'.rt \leq \tau_{\mathtt{act}}$. From Property 3 it follows that $s.mode = \mathtt{sup}$ and $s.rt \leq \tau_{\mathtt{act}}$. From the induction hypothesis it follows that $s \in \mathbf{B}_{s.rt}$, that is $\theta_{min} \leq s.\theta_p^0 \leq \theta_{max}$ and $\Gamma^-(s.\theta_p^0, \ \tau_{\mathtt{act}} - s.rt) \leq s.\theta_p^1 \leq \Gamma^+(s.\theta_p^0, \ \tau_{\mathtt{act}} - s.rt)$. For all intermediate states between s and s' the input U to `Plant` is arbitrary. Using the maximum value U_{max} and integrating over $\tau.dom$ the same upper bounds on $s'.\theta_p^1$ and $s'.\theta_p^0$ are obtained as expressed by equations (15) and (16). Simplifying:

$$\begin{aligned} s'.\theta_p^0 &\leq \theta_{max}, \ \textit{and} && (22)\\ s'.\theta_p^1 &\leq \Gamma^+(s'.\theta_p^0, \ \tau_{\mathtt{act}} - s.rt - t'), && (23) \end{aligned}$$

Similarly using the lower bound on U, we get

$$\begin{aligned} s'.\theta_p^0 &\geq \theta_{min}, \quad \textit{and} && (24)\\ s'.\theta_p^1 &\geq \Gamma^-(s'.\theta_p^0, \ \tau_{\mathtt{act}} - s.rt - t'). && (25) \end{aligned}$$

Combining equations (22) (23) (24) and (25) we have $s' \in \mathbf{B}_{s.rt+t'} = \mathbf{B}_{s'.rt}$ $\Box$.

5.5 Recovery Phase

We introduce a few notations before moving on to prove the safety of the system in the recovery phase. In the context of a particular trajectory τ, we abbreviate $\tau \downarrow x(t)$ as simply $x(t)$. The tangent and the normal vectors to a curve at the point (x, y) are denoted by $\mathbf{n}(x, y)$ and $d(x, y)$ respectively.

Invariant 5.12 *In any reachable states* s*, if* $s.mode = \mathtt{sup}$ *and* $s.rt \geq \tau_{\mathtt{act}}$ *then* $s \in \mathbf{C}$.

Proof: The base case is trivially satisfied because $s.mode = \mathtt{usr}$. For the discrete part, consider discrete transitions $s \xrightarrow{\pi} s'$ with $s'.mode = \mathtt{sup}$. If $\pi = control$ there are two subcases: if $s.mode = \mathtt{sup}$ then from the inductive hypothesis $s \in \mathbf{C}$. Therefore using Property 3 it follows that $s' \in \mathbf{C}$. Otherwise $s.mode = \mathtt{usr}$ and $s'.rt = 0$ and the invariant holds vacuously. For all other discrete actions the invariant is preserved because none of the variables involved are altered.

For the continuous part of the induction, consider closed trajectory τ with $s'.mode = \mathtt{sup}$ and $s'.rt \geq \tau_{\mathtt{act}}$. We claim that $s \in \mathbf{C}$. From Property 3 it is known that $s.mode = \mathtt{sup}$, Consider two possible cases: (1) If $s.rt < \tau_{\mathtt{act}}$ then from Invariant 5.11 it follows that $s \in \mathbf{C}$. Otherwise (2) $s.rt \geq \tau_{\mathtt{act}}$ and from the inductive hypothesis it follows that $s \in \mathbf{C}$.

If $s \in \mathbf{U}$, then from Lemma 5.4 it follows that s' is in $\mathbf{R}$ and therefore in $\mathbf{C}$. So it remains to show that if $s \in \mathbf{C} \setminus \mathbf{U}$ then $s' \in \mathbf{C}$. We shall prove this by contradiction. Since $s.\theta_s^1 > I^+(s.\theta_S^0)$ or $s.\theta_s^1 < I^+(s.\theta_S^0)$ it follows from Invariant 5.10 that $s.U = U_{min}$ or U_{max} respectively. Now, suppose $s' \notin \mathbf{C}$, then there must exist $t' \in \tau.dom$ such that τ leaves the $\mathbf{C}$ at $\tau(t')$. At the boundary of $\mathbf{C}$ it must be the case that $d(\theta_p^0(t'), \theta_p^1(t')) \cdot \mathbf{n}(\theta_p^0(t'), \theta_p^1(t')) \geq 0$, where $\cdot$ denotes the inner product between the two vectors. We reach a contradiction by showing that at each point s'' on the boundary of $\mathbf{C}$, $d(s''.\theta_p^0, s''.\theta_p^1) \cdot \mathbf{n}(s''.\theta_p^0, s''.\theta_p^1) < 0$ Now onwards we shall write x instead of $s''.x$ where it is understood that x is the state component of a point in the state space which is on the boundary of $\mathbf{C}$. We consider the curves defining the boundary of $\mathbf{C}$(Figure 7).

Case 1: The upper boundary $\Gamma^+(\theta_p^0, 0)$ can be written as:

$$\mathbf{C}^+ = \left\{ d(\theta_p^0, \theta_p^1) \mid \theta_{min} \leq \theta_p^0 \leq \theta_{max} \wedge \theta_p^1 \geq 0 \wedge V_1(\theta_p^0, \theta_p^1) = \left(-U_{min} + \Omega^2 \cos\theta_{max}\right)\theta_{max} \right\},$$

where $V_1(\theta_p^0, \theta_p^1) = \frac{1}{2}{\theta_p^1}^2 + (-U_{min} + \Omega^2 \cos\theta_{max})\,\theta_p^0$. So the outer normal of $\mathbf{C}^+$ is given by

$$\mathbf{n}(\theta_p^0, \theta_p^1) = \nabla V_1 := \left(\frac{\partial V_1}{\partial \theta_p^0}, \frac{\partial V_1}{\partial \theta_p^1}\right) = (-U_{min} + \Omega^2 \cos\theta_{max}, \theta_p^1),$$

where ∇ is the gradient operator. Since $\theta_s^1 \geq I^+(\theta_s^0)$ and $rt > \tau_{\mathtt{act}}$ therefore $U = U_{min}$ by Invariant 5.10. The plant equations are given by: $d(\theta_p^0) = \theta_p^1$, and $d(\theta_p^1) = -\Omega^2\cos\theta_p^0 + U_{min}$. So we have

$$\begin{aligned} \mathbf{n}(\theta_p^0, \theta_p^1) \cdot d(\theta_p^0, \theta_p^1) &= (-U_{min} + \Omega^2\cos\theta_{max}, \theta_p^1) \cdot (\theta_p^1, -\Omega^2\cos\theta_p^0 + U_{min}) \\ &= \Omega^2(\cos\theta_{max} - \cos\theta_p^0)\theta_p^1 \leq 0, \end{aligned}$$

for $(\theta_p^0, \theta_p^1) \in \mathbf{C}^+$. The equal sign is valid iff $(\theta_p^0, \theta_p^1) = (\theta_{max}, 0)$. So the point $(\theta_p^0, \theta_p^1) = (\theta_{max}, 0)$ needs special treatment. Integrating for initial condition $(\theta_{max}, 0)$, we get

$$\sin\theta_p^0 = \sin\theta_{max} + \frac{1}{\Omega^2}\left[U_{min}(\theta_p^0 - \theta_{max}) - \frac{1}{2}{\theta_p^1}^2\right]. \tag{26}$$

This function defines an integral curve $\theta_p^0 = F_1(\theta_p^1)$. Differentiating (26) with respect to θ_p^1,

$$\frac{d\theta_p^0}{d\theta_p^1} = \frac{\theta_p^1}{U_{min} - \Omega^2\cos\theta_p^0}, \quad and \quad \frac{d^2\theta_p^0}{d{\theta_p^1}^2} = \frac{1}{U_{min} - \Omega^2\cos\theta_p^0} - \frac{\theta_p^1\sin\theta_p^0}{(U_{min} - \Omega^2\cos\theta_p^0)^3}.$$

By evaluating the above derivatives at $(\theta_{max}, 0)$, we have

$$\frac{d\theta_p^0}{d\theta_p^1}(\theta_{max}, 0) = 0, \qquad \frac{d^2\theta_p^0}{d{\theta_p^1}^2}(\theta_{max}, 0) = \frac{1}{U_{min} - \Omega^2\cos\theta_{max}} < 0.$$

The inequality holds because $U_{min} \leq 0$ and $-\frac{\pi}{2} < \theta_p^0 < \frac{\pi}{2}$. So the integral curve $\theta_p^0 = F_1(\theta_p^1)$ achieves a maximum at $(\theta_{max}, 0)$, which implies the trajectory goes inside $\mathbf{C}$.

Case 2: The left boundary of $\mathbf{C}$ is given by $\mathbf{C}^l = \left\{d(\theta_p^0, \theta_p^1) | \theta = \theta_{min} \wedge 0 < \theta_p^1 < \Theta^+\right\}$, where $\Theta^+ \triangleq \sqrt{2\left(-U_{min} + \Omega^2 \cos\theta_{max}\right)\left(\theta_{max} - \theta_{min}\right)}$. The outer normal of $\mathbf{C}^l$ is given by $\mathbf{n} = (-1, 0)$, and we have $\mathbf{n}(\theta_p^0, \theta_p^1) \cdot d(\theta_p^0, \theta_p^1) = (-1, 0) \cdot (d\theta_p^0, d\theta_p^1) = -d\theta_p^0 = -\theta_p^1 < 0$, for $(\theta_p^0, \theta_p^1) \in \mathbf{C}^l$, which implies the trajectory will not leave $\mathbf{C}$ through $\mathbf{C}^l$.

The proof for the lower and the right boundary are symmetrical to that of Case 1 and Case 2 respectively. By combining all the cases, we have shown that for any $t'' \in \tau.dom$, at any point on the boundary of $\mathbf{C}$ $d(\theta_p^0(t'), \theta_p^1(t')) \cdot \mathbf{n}(\theta_p^0(t'), \theta_p^1(t')) < 0$. Therefore s' is in $\mathbf{C}$.□

5.6 Safety

Combining the above invariants the safety of the composed system is established.

Theorem 1 *All reachable states of $\mathcal{A}$ are contained in* $\mathbf{C}$.

Proof: For any reachable state s, if $s.mode = \mathtt{usr}$ then $s \in \mathbf{R} \subseteq \mathbf{C}$ by Invariant 5.4. Otherwise $s.mode = \mathtt{sup}$, and there are two possibilities: if $s.rt < \tau_{\mathtt{act}}$ then, by Invariant 5.11, $s \in \mathbf{B}_{s.rt} \subseteq \mathbf{C}$. Else $s.rt \geq \tau_{\mathtt{act}}$ and it follows from Invariant 5.12 that $s \in \mathbf{C}$. □

6 Conclusions

In this paper we have presented an advanced application of the HIOA framework for verifying hybrid systems. The safety of the designed supervisory controller was established by proving a set of invariants. The proof techniques demonstrate two properties that are important for reasoning about complex hybrid systems: (1) the proofs are decomposed into discrete and continuous parts, which are independent of each other, and (2) the reasoning style is purely assertional, that is, based on the current state of the system, rather than complete executions.

The design of the supervisory controller uses a safe operating region of the plant, beyond which it overrides the user controller, performs appropriate recovery, and returns control to the user. The duration of the recovery period has not been discussed here, but it has been shown to be bounded in [13]. The size of the safe operating region, depends on the plant dynamics, sensor errors, sampling period, actuator bandwidth and saturation. An implementation of the supervisory controller in the actual system is in progress. We also intend to design and verify a class of supervisory controllers that reduce unnecessary interferences by utilizing additional information about particular user controllers.

The specification language used in this paper is based on the hybrid I/O automaton model. Certain extra structures have been added to the HIOA model of [9] in order to specify the trajectories using activities. We intend to incorporate the language extensions into a toolset for automatically checking HIOA programs. At present we are also working on building a theorem prover interface for HIOA which would allow us to partially automate the verification process.

References

[1] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34, 1995.

[2] Stephen Garland, Nancy Lynch, and Mandana Vaziri. IOA: A language for specifying, programming and validating distributed systems. Technical report, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA, October 1999.

[3] Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. Hytech: A model checker for hybrid systems. In *Computer Aided Verification (CAV '97)*, volume 1254 of *Lecture Notes in Computer Science*, pages 460–483, 1997.

[4] Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. What's decidable about hybrid automata? In *ACM Symposium on Theory of Computing*, pages 373–382, 1995.

[5] http://www.quanser.com/english/html/products/fs_product_challenge.asp?lang_code=english&pcat_code=ex spe&prod_code=S1-3dofheli.

[6] Carolos Livadas, John Lygeros, and Nancy A. Lynch. High-level modeling and analysis of TCAS. In *Proceedings of the 20th IEEE Real-Time Systems Symposium (RTSS'99),Phoenix, Arizona*, pages 115–125, December 1999.

[7] David G. Luenberger. *Introduction to Dynamic Systems: Theory, Models, and Applications.* John Wiley and Sons, Inc., New York, 1979.

[8] Nancy Lynch. A three-level analysis of a simple acceleration maneuver, with uncertainties. In *Proceedings of the Third AMAST Workshop on Real-Time Systems*, pages 1–22, Salt Lake City, Utah, March 1996. World Scientific Publishing Company.

[9] Nancy Lynch, Roberto Segala, and Frits Vaandraager. Hybrid I/O automata. Technical Report MIT-LCS-TR-827b, MIT Laboratory for Computer Science, Technical Report, Cambridge, MA 02139, February 2002. To appear in Information and Computation, http://theory.lcs.mit.edu/tds/papers/Lynch/HIOA-final.ps.

[10] Nancy Lynch, Roberto Segala, Frits Vaandrager, and H. B. Weinberg. Hybrid I/O automata. In T. Henzinger R. Alur and E. Sontag, editors, *Hybrid Systems III*, volume 1066 of *Lecture Notes in Computer Science*, New Brunswick, New Jersey, October 1995. Springer-Verlag.

[11] Nancy A. Lynch, Roberto Segala, and Frits W. Vaandrager. Hybrid I/O automata revisited. In M.D. Di Benedetto and A.L. Sangiovanni-Vincentelli, editors, *Proceedings Fourth International Workshop on Hybrid Systems: Computation and Control (HSCC'01),* Rome, Italy, volume 2034 of *lncs*. springer, March 2001.

[12] Sayan Mitra. Language for Hybrid Input/Output Automata, 2002. Work in progress. http://theory.lcs.mit.edu/mitras/research/composing_activities.ps.

[13] Yong Wang, Masha Ishutkina, Sayan Mitra, Carolos Livadas, Nancy A. Lynch, and Eric Feron. Design of Supervisory Safety Control for 3DOF Helicopter using Hybrid I/O Automata, 2002. pre-print http://gewurtz.mit.edu/ishut/darpa_sec_mit/papers/quanser.ps.

[14] H. B. Weinberg, Nancy Lynch, and Norman Delisle. Verification of automated vehicle protection systems. In T. Henzinger R. Alur and E. Sontag, editors, *Hybrid Systems III: Verification and Control (DIMACS/SYCON Workshop on Verification and Control of Hybrid Systems)*, volume 1066 of *Lecture Notes in Computer Science*, pages 101–113, New Brunswick, New Jersey, October 1995. Springer-Verlag.