Safety Verification of Model Helicopter Controller using Hybrid Input/Output Automata*

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

[1]MIT LCS,
Cambridge, MA 02139, USA
{mitras,lynch}@theory.lcs.mit.edu

[2]MIT LIDS,
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, 15, 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 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

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

constitutes a first step for automating the verification process using HIOA.

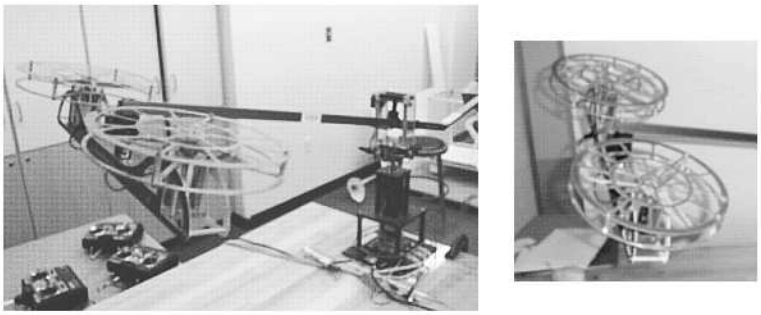

Figure 1: Helicopter model with three degrees of freedom.

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. Due to limited space we present brief proof sketches for the important invariants required for proving safety of the system in Section 5. The complete proofs are given in the Appendix and are also available in the form of a technical report[13]. Concluding remarks and future directions for research are discussed in Section 6.

2 Hybrid I/O Automata

A brief review of the HIOA model is presented in this section. For a complete discussion 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.

A hybrid I/O automaton $\mathcal{A}$ consists of : (1) a set V of variables, partitioned into *internal* X, *input* U, and *output variables* Y. The internal variables are also called state variables. $Z \triangleq X \cup Y$ is the set of *locally controlled or local variables*. (2) a set A of actions , partitioned into *internal* H, *input* I, and *output actions* O. (3) a set of states $Q \subseteq val(X)$, (4) a non-empty set of *start states* $\Theta \subseteq Q$, (5) a set of *discrete transitions* $\mathcal{D} \subseteq Q \times A \times Q$. A transition $(\mathbf{x}, a, \mathbf{x}') \in \mathcal{D}$ is written in short as $\mathbf{x} \xrightarrow{a}_{\mathcal{A}} \mathbf{x}'$. (6) a set of *trajectories* $\mathcal{T}$ 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 is denoted by $\tau.fstate$. 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 $\tau_i.lstate \stackrel{a_{i+1}}{\rightarrow} \tau_{i+1}.fstate$. 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.

An invariant $\mathcal{I}$ of $\mathcal{A}$ is either derived from other invariants or proved by induction on the length of a closed execution of $\mathcal{A}$. The induction consists of a base case, and an induction step. The base case tests that $\mathcal{I}(s)$ is satisfied for all $s \in \Theta$. The induction step consists of : (1) A discrete part—which tests that for every discrete step $s \stackrel{\pi}{\rightarrow} s' \in \mathcal{D}$, $\mathcal{I}(s)$ implies $\mathcal{I}(s')$. (2) A continuous part—which tests that for any closed trajectory $\tau \in \mathcal{T}$, with $\tau.fstate = s$ and $\tau.lstate = s'$, $\mathcal{I}(s)$ implies $\mathcal{I}(s')$. We shall use s and s' to denote the pre and the post states of discrete transitions, and also the $fstate$ and the $lstate$ of closed trajectories, as will be clear from the context.

2.2 New Addition to HIOA Structure: *Activities*

In the earlier works [6, 15, 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] used in this paper uses "state space" representation [7] of the trajectories. To make this representation work, the following extra structure has been introduced into the basic HIOA model of [9].

The time domain is assumed to be the set of reals 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 refer to the discrete and continuous subsets of S respectively.

Let e be a real valued algebraic expression involving the variables in $X \cup U$. For a given trajectory τ, $\tau.e$ denotes the function with domain $\tau.dom$ that gives the value of the expression e at all times over τ. Given that v is a local 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 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 independent 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 $X \cup U$. 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 over τ, all the variables in Z_a satisfy the differential and algebraic equations in E with $\tau(0)$ defining the initial valuations.

An *activity* α of HIOA $\mathcal{A}$ consists of four components: (1) a *starting condition* $P^- \subseteq Q$, (2) a *operating condition* $P \subseteq P^-$, (3) a *stopping condition* $P^+ \subseteq Q$, and (4) a state model E. 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. τ satisfies the state model E.
2. For all $t \in \tau.dom$, $(\tau \downarrow X)(t) \in P^-$.
3. For all $t \in \tau.dom - \{0\}$, $\tau \downarrow X(t) \in P$.
4. If $(\tau \downarrow X)(t) \in P^+$ for $t \in dom(\tau)$ then τ is closed and $t = \tau.ltime$.

We impose the following restrictions on hybrid I/O automata model in order to specify the trajectories of an automaton as the union of the sets of trajectories specified by its activities.

[1] By *specifies* we mean restricts rather than uniquely determines. Due to possible nondeterminism in the state model, unique determination might not be possible.

type RAD = Real **suchthat** $(i : \texttt{RAD},\ |i| \leq \Theta)$ % Θ max abs value for angles
type RADPS = Real **suchthat** $(i : \texttt{RADPS},\ |i| \leq \dot{\Theta})$ % $\dot{\Theta}$ max abs value of angular velocity
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 2: HIOA specification of the plant

R1 Every variable is either discrete or continuous.
R2 Local discrete variables are constant over trajectories, that is, $\forall \tau \in \mathcal{T}, \tau.lval\lceil Z_d = \tau.fval\lceil Z_d$.
R3 Starting condition of all the activities are disjoint, that is, $P_i^- \cap \mathcal{P}_j^- = \emptyset$ if $i \neq j$.

It can be shown that (Lemma 6.1 in the Appendix) a set of trajectories specified by a set of activities, satisfy the prefix, suffix, and concatenation closure properties.

2.3 Language Constructs

In the HIOA specification language variables are declared by their names, types, and optionally their initial valuations. Varibales declared with the **analog** keyword are continuous, else they are discrete. Actions are declared by their names, types, and optional list of parameters. Algebraic expressions are written using the operators $+, -, *$, and $\backslash$. A nondeterministic assignment, such as $v \in [e_1, e_2]$, is written as $v :=$ **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 $\alpha : (P^-, P, P^+, E)$ is written as:

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

For automata with a single activity, if the starting condition P^- is not specified explicitly, then it is assumed to be equal to the entire state space Q. If P and P^+ are omitted then they are assumed to be equal to P^-, and $\emptyset$ respectively.

3 Specification of System Components

This section describes the HIOA models for the components of the helicopter system, except for the supervisory controller, which is in Section 4. Discrete and continuous communication among the components are shown in Figure 3. In this paper we consider the pitch dynamics of the helicopter, which are critical for safety. A complete dynamical model of the helicopter with three degrees of rotational freedom can be found in [14]. In practice the roll and yaw effects are eliminated by making the initial conditions along these axes to be zero and giving identical input to the two rotors. The pitch dynamics is described by $\ddot{\theta} + \Omega^2 \cos\theta = U(t)$, where Ω is the characteristic frequency of the system and U is the net input for the pitch axis which can range between U_{min} and U_{max}. The `Plant` automaton (Figure 2) specifies the evolution of the pitch angle θ_p^0 and the velocity θ_p^1 with input U, according to the picth dynamics. The global types `RAD, RADPS` and `UTYPE` define the domains for variables representing angle, angular velocity and actuator output respectively. 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 (4). The `Plant` is *safe* at a given state s if the pitch angle $s.\theta_p^0$ is within the allowed limits θ_{min} and θ_{max}. Where θ_{min} corresponds to the helicopter hitting the table (or the ground). And θ_{max} corresponds to the helicopter hitting a very fragile mechanical stop. The set of safe states is defined as:

$$\mathbf{S} \triangleq \{s \mid \theta_{min} \leq s.\theta_p^0 \leq \theta_{max}\}. \tag{1}$$

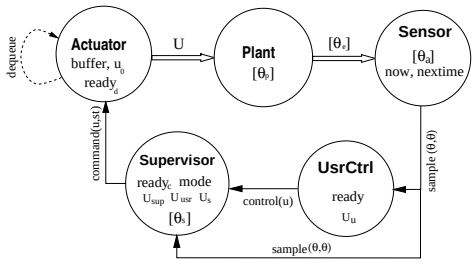

Figure 3: Components of Helicopter system. Continuous and discrete communication among components are shown by wide and thin arrows respectively. The internal variables are marked inside the circles and internal actions are shown by a dashed self loop.

---

hybridautomaton Sensor(ϵ_0,ϵ_1, Δ : Real)

actions

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

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 $now = next\_time$

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

derived variable
$s.time\_left \triangleq s.next\_time - s.now$.

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

The `Sensor` automaton (Figure 4) periodically conveys the state of `Plant` to the controllers as observed by the physical sensors. It is parameterized by the sampling period Δ, the sensor errors for pitch angle ϵ_0, and velocity ϵ_1. The variable *now* serves as a clock. 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 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 (Figure 5) models an arbitrary user controller. It reads the *sample* action and triggers an output *control(u_d)* action, which communicates the user's output U_u to the supervisor. The output U_u is modeled as a nondeterministic choice over the entire range U_{min} to U_{max}. This captures our assumption that the user is capable of issuing arbitrarily bad control inputs to the plant. The design of a safe supervisor for `UsrCtrl` ensures that the system would be safe for *any* user controller because every controller must implement this specification of `UsrCtrl`.

The `Actuator` automaton (Figure 6) 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 following invariant for `Actuator` can be derived from its specification. The proof follows from a simple induction and is given in the Appendix.

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

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

discrete transitions
    input sample ( θ_d^0 , θ_d^1 )
    eff θ_u^0 := θ_d^0; θ_u^1 = θ_d^1
        U_u := choose [U_min, U_max];
        ready := true

trajectories
    activity void
        evolve stop at ready

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

    output control ( u_d )
    pre (u_d = U_u) ∧ ready
    eff ready := false
```

Figure 5: Specification of User's Controller

```
type MODES = { usr, sup }
hybridautomaton Actuator(τ_act)
actions
    input command ( u : UTYPE )
    internal dequeue

discrete transitions
    input command ( u )
    eff buffer + := (u, now + τ_act);
        ready_d := true

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

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

    internal dequeue
    pre buffer.head.st = now ∧ ready_d
    eff u_o := buffer.head.v;
        buffer := buffer.tail;
        ready_d := false
```

Figure 6: Actuator and D/A conversion

4 Supervisory Controller

The first goal of the supervisory controller is to ensure safety. A second requirement is to interfere as little as possible with the user's controller. The design principle of the supervisor is simple: Allow the user to be in control in a *safe operating region* **U**, from where the supervisor is guaranteed to restore the plant to a safe state; Outside of **U**, block the user's controller, perform recovery, and return control to the user. In order to satisfy the second requirement it is also desirable to make **U** as large as possible. Therefore we have to find **U**, which is the largest set of states in which the user can be allowed to operate without threatening the safety of the plant.

4.1 Safe Operating Region

Clearly the **U** has to be a subset of **S**. Consider a region $\mathbf{C} \subseteq \mathbf{S}$, from which all the reachable states are contained in **S**, provided that the input U to the plant is *correct*. By correct we mean that the input to the plant is $U = U_{min}$ (or U_{max}) if the pitch θ_p^0 is in the danger of hitting θ_{min} (θ_{max} resp.). Since the supervisor cannot change the output of the actuator instantaneously, due to the $\tau_{\mathtt{act}}$ delay in `Actuator` *buffer*, therefore the region **C** is not a safe operating region for the user. The supervisor has to estimate if the plant under user's control would go outside of **C** in $\tau_{\mathtt{act}}$ time. We use a conservative estimate and define the region **R** as the set of states from which all reachable states over a period of $\tau_{\mathtt{act}}$ are within **C**. If the user is restricted to operate in **R** and if the supervisor can monitor the plant state accurately, then it can take a timely decision to take over. However, the supervisor cannot observe the plant state accurately, it relies on the periodic updates from the inaccurate sensors. Taking the errors and the delay into account we

define the region **U** as follows: An observed state s is in **U** if starting from any actual state corresponding to s all the reachable states over a Δ interval of time are in **R**. Switching back to the user's controller from the supervisor is delayed until the supervisor brings the plant state within an *inner* region $\mathbf{I} \subseteq \mathbf{U}$. This asymmetry in switching prevents high frequency chattering between the user and the supervisor.
The regions **C**, **R**, **U**, and **I** are defined as follows. $U_{mag} = U_{max} - U_{min}$.

$$\mathbf{C} \triangleq \{s \mid s.\theta_p^0 \in [\theta_{min}, \theta_{max}] \wedge s.\theta_p^1 \in [\Gamma^-(s.\theta_p^0, 0), \Gamma^+(s.\theta_p^0, 0)]\}, \tag{2}$$

$$\mathbf{R} \triangleq \{s \mid \theta_{min} \leq s.\theta_p^0 \leq \theta_{max} \wedge \Gamma^-(s.\theta_p^0, \tau_{\mathtt{act}}) \leq s.\theta_p^1 \leq \Gamma^+(s.\theta_p^0, \tau_{\mathtt{act}})\}, \tag{3}$$

$$\mathbf{U} \triangleq \{s \mid \theta_{min} + \epsilon_0 \leq s.\theta_s^0 \leq \theta_{max} - \epsilon_0 \ \wedge U^-(s.\theta_s^0) \leq s.\theta_s^1 \leq U^+(s.\theta_s^0)\}, \tag{4}$$

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

$$\Gamma^+(\theta, \mathcal{T}) = -U_{mag}\mathcal{T} + \sqrt{2(\Omega^2 \cos\theta_{max} - U_{min})(\theta_{max} - \theta + \frac{1}{2}U_{mag}\mathcal{T}^2)}, \tag{6}$$

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

$$U^+(\theta) = -\epsilon_1 + \Gamma^+(\theta + \epsilon_0, \tau_{\mathtt{act}} + \Delta) \qquad U^-(\theta) = +\epsilon_1 + \Gamma^-(\theta - \epsilon_0, \tau_{\mathtt{act}} + \Delta)$$

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

From the above definitions the following properties are derived.

Property 1 *Over the interval* $-\frac{\pi}{2} \leq \theta \leq \frac{\pi}{2}$ *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* $\mathcal{T} > 0$.

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

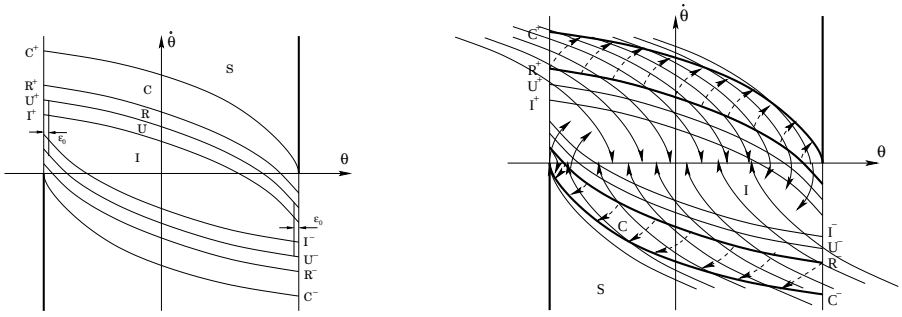

Figure 7: (a) Regions in the statespace. (b) Trajectories in the settling (dashed lines) and recovery(solid lines) periods.

4.2 Supervisor Automaton

The `Supervisor` automaton (Figure 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 **U** then *mode* remains unchanged and U_s is set to U_{usr}. If the present state is not in **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 **I**. The *control* action enables the *command* output action by setting the $ready_c$ flag.

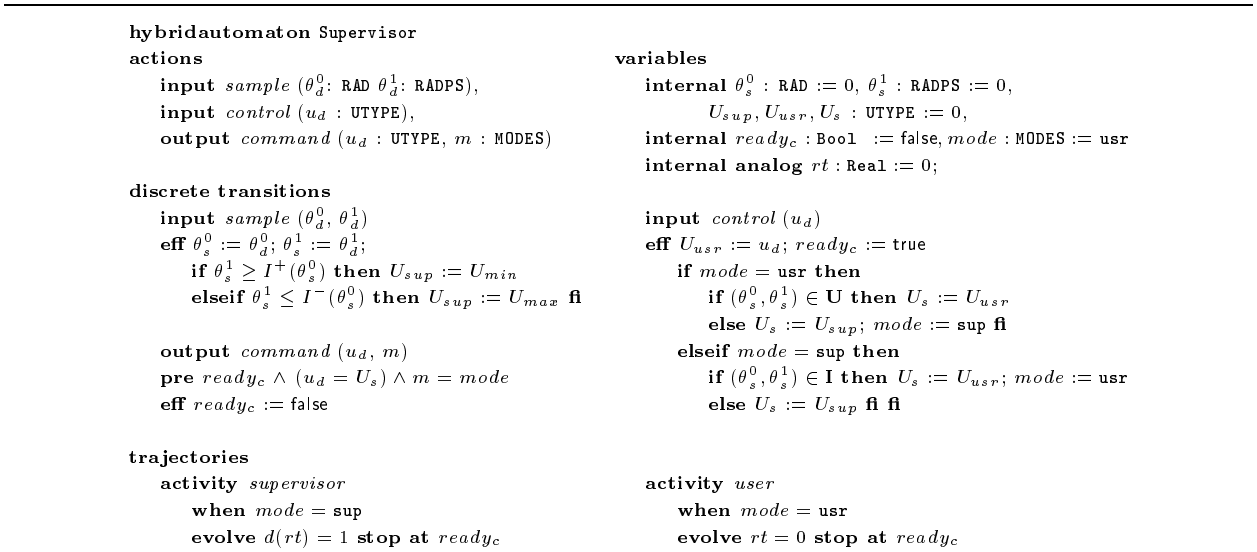

Figure 8: HIOA specification of supervisor automaton

5 Analysis of Helicopter System

In this section we present the safety verification of the composed system. Let $\mathcal{A}$ denote the composition of the `Plant`, `Sensor`, `UsrCtrl`, `Actuator`, and the `Supervisor` automata. Safety is preserved if all the reachable states of the $\mathcal{A}$ are contained within the region **S**. It is assumed that: (1) $\theta_{min} < 0 < |\theta_{min}| < \theta_{max}$, (2) $U_{max} > \Omega^2$, $U_{min} \leq 0$, and (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, $s'.\theta_s^1 \leq I^+(s'.\theta_s^0)$. Assumptions (1) and (2) are derived from the dimensions of the physical system. Assumption (3) is a requirement which limits the speed of the plant and the sampling period so that it is not possible for the plant to jump across **I** without the sensors detecting it.

In the next section we present some preliminary properties of $\mathcal{A}$, then we state the key invariants of $\mathcal{A}$ in the user and the supervisor modes along with their proofs. The details of all the invariant proofs are given in the Appendix and can also be found in [13].

5.1 Some Preliminary Properties of A

The specification of the components of $\mathcal{A}$ satisfy restrictions **R2**, **R2** and **R3** and the plant state variables θ_p^0 and θ_p^1 are not modified by any discrete action. The next two properties are consequences of these facts:

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

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

Invariant 5.1 follows from the code by a straightforward induction. Lemma 5.1 follows from Invariant 5.1 and indicates the times at which the different actions of $\mathcal{A}$ occur. Invariant 5.2 limits the size of the *buffer* and it is a consequence of Invariant 3.1 and Lemma 5.1.

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

Lemma 5.1 *In any execution of $\mathcal{A}$, sample, control, and command actions occur when $now = n\Delta$, and dequeue actions occurs when $timer = \tau_{\texttt{act}} + n\Delta$ for some integer $n > 0$.*

Invariant 5.2 *In every reachable state s, for all $0 \leq i < s.buffer.\texttt{size} - 1$, $s.buffer[\texttt{i+1}].\texttt{st} = s.buffer[\texttt{i}].\texttt{st} + \Delta$, and $s.buffer.\texttt{size} \leq \lceil \frac{\tau_{\texttt{act}}}{\Delta} \rceil$.*

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 $0 \leq t \leq \Delta$, $\mathbf{A}_t \triangleq \{s \mid s.\theta_p^0 \in [\theta_{min}, \theta_{max}] \wedge s.\theta_p^1 \in [\Gamma^-(s.\theta_p^0, \tau_{\texttt{eff}} + t), \Gamma^+(s.\theta_p^0, \tau_{\texttt{eff}} + t)\}$. Lemma 5.2 states the properties of the $\mathbf{A}_t$ regions.

Lemma 5.2 *The regions $\mathbf{A}_t$ satisfy:* 1. $\mathbf{A}_0 = \mathbf{R}$, 2. $\mathbf{U} \subseteq \mathbf{A}_\Delta$, *and* 3. *If $0 \leq t \leq t' \leq \Delta$ then $A_{t'} \subseteq A_t$.*

The next lemma bounds the reachable sates over a singe trajectory and is used to prove safety when a tarjectory starts from the safe operating region $\mathbf{U}$. Invariant 5.3 makes use of Lemma 5.3. The safety of the system in the user mode is established by Invariant 5.4.

Lemma 5.3 *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, $\theta_{min} \leq s.\theta_p^0 \leq \theta_{max}$ and $\Gamma^-(s.\theta_p^0,\ \tau_{\texttt{eff}} + t) \leq s.\theta_p^1 \leq \Gamma^+(s.\theta_p^0,\ \tau_{\texttt{eff}} + t)$. 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 : $d(s''.\theta_p^1) \leq -\Omega^2 \cos\theta_{max} + U_{max}$. Integrating from t to t', it follows that $s'.\theta_p^1 \leq (U_{max} - \Omega^2 \cos\theta_{max})t' + s.\theta_p^1$, and $s'.\theta_p^0 \leq \frac{1}{2}(U_{max} - \Omega^2 \cos\theta_{max})t'^2 + s.\theta_p^1 t' + s.\theta_p^0$. Simplifying and using the definition of Γ^+ we get the following bounds on $s'.\theta_p^0$ and $s'.\theta_p^1$: $s'.\theta_p^0 \leq \theta_{max}$, and $s'.\theta_p^1 \leq \Gamma^+(s'.\theta_p^0,\ \tau_{\texttt{eff}} + t - t')$. Similarly considering maximum negative output, $U = U_{min}$, we get the lower bounds on $s'.\theta_s^0$ and $s'.\theta_s^1$. $s'.\theta_p^0 \geq \theta_{min}$, and $s'.\theta_p^1 \geq \Gamma^-(s'.\theta_p^0,\ \tau_{\texttt{eff}} + t - t')$. Combining equations all the above bounds on s' it follows that $s' \in \mathbf{A}_{t-t'}$. □

Invariant 5.3 *In any reachable state s, if $s.mode = \texttt{usr}$ and $\neg s.ready$ then $s \in \mathbf{A}_{s.time\_left}$.*

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

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 : Settling Phase

For proving safety in the supervisor mode, we first state some of the simple invariants. Invariant 5.5 states that, in all reachable with *ready* set to false, if the sensed plant state is within I^+ and I^-, then the system is in the user mode. Invariant 5.6 follows from the code of the *sample* action. And Invariant 5.7 is proved by a simple induction.

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

Invariant 5.6 *In any reachable state s, else 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 $s.U_{sup} = U_{max}$.*

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

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 $k \geq 0$,

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

Clearly, for all $k > 0$, $\mathcal{Q}_k^+(s)$ implies $\mathcal{Q}_{k-1}^+(s)$, and therefore for any $k \geq s.buffer.\texttt{size}$, $\mathcal{Q}_k^+(s)$ implies that $\mathcal{Q}_j^+(s)$ holds for all $j < s.buffer.\texttt{size}$. 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 $\texttt{true}$, 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 $s.mode = \texttt{sup}$:*

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*

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

Invariant 5.9 *In any reachable state s, such that $s.mode = \texttt{sup}$ and $s.rt > \tau_{\texttt{act}}$*
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 $s.U = U_{max}$,

We split the execution of $\mathcal{A}$ in the supervisor mode (Figure 7(b)) into (a) a *settling phase* of length $\tau_{\texttt{act}}$ in which the input U to the plant is arbitrary, and (b) a variable length *recovery phase* during which $rt > \tau_{\texttt{act}}$ and the input to the plant is correct, that is, in accordance with Invariant 5.9.

Next we define a set of regions $\mathbf{B}_t$ which are analogous to the $\mathbf{A}_t$ regions. $\mathbf{B}_t \triangleq \{s \mid s.\theta_p^0 \in [\theta_{min}, \theta_{max}] \wedge\ s.\theta_p^1 \in [\Gamma^-(s.\theta_p^0,\ \tau_{\texttt{act}} - t), \Gamma^+(s.\theta_p^0,\ \tau_{\texttt{act}} - t)]\}$, for $0 \leq t \leq \tau_{\texttt{act}}$. Lemma 5.4 states the relationship between the $\mathbf{B}_t$ regions and its proof is similar to that of Lemma 5.2. Invariant 5.10 bounds the location of a state s in terms of the $\mathbf{B}_t$ regions, when $s.rt \leq \tau_{\texttt{act}}$. This implies the safety of the system in the settling phase.

Lemma 5.4 *The regions $\mathbf{B}_t$ satisfy:* 1. $\mathbf{B}_0 = \mathbf{R}$, 2. $\mathbf{B}_{\tau_{\texttt{act}}} = \mathbf{C}$, 3. *If* $0 \leq t \leq t' \leq \tau_{\texttt{act}}$ *then* $\mathbf{B}_t \subseteq \mathbf{B}_{t'}$.

Invariant 5.10 *For any reachable state s, if $s.mode = \texttt{sup} \wedge s.rt \leq \tau_{\texttt{act}}$ then $s \in \mathbf{B}_{s.rt}$.*

5.4 Supervisor Mode: Recovery Phase

Invariant 5.11 states that $\mathbf{C}$ is an invariant set for the system in the recovery phase. A sketch of the proof is given here, the complete proof is in the Appendix.

Invariant 5.11 *In any reachable states s, if $s.mode = \texttt{sup}$ and $s.rt \geq \tau_{\texttt{act}}$ then $s \in \mathbf{C}$.*

proof sketch: The base case is trivially satisfied. The discrete part of the induction is also straightforward, the *control* action alters the *mode*. If $s.mode = \texttt{sup}$ then using the inductive hypothesis, $s' \in \mathbf{C}$. Otherwise $s.mode = \texttt{usr}$ and $s'.rt = 0$ and the invariant holds vacuously. For all other discrete actions the invariant is preserved. For the continuous part of the induction, consider closed trajectory τ with $s'.mode = \texttt{sup}$ and $s'.rt \geq \tau_{\texttt{act}}$. We claim that $s \in \mathbf{C}$. From Property 3 it is known that $s.mode = \texttt{sup}$, (1) If $s.rt < \tau_{\texttt{act}}$ then from Invariant 5.10 it follows that $s \in \mathbf{C}$. Otherwise (2) $s.rt \geq \tau_{\texttt{act}}$ and from the inductive hypothesis it follows that $s \in \mathbf{C}$. If $s \in \mathbf{U}$, then from Lemma 5.3 it follows that $s' \in \mathbf{R} \subseteq \mathbf{C}$. So it remains to show that if $s \in \mathbf{C} \setminus \mathbf{U}$ then $s' \in \mathbf{C}$. This is proved by contradiction, suppose $s' \notin \mathbf{C}$, then there must exist $t' \in \tau.dom$ such that τ leaves the $\mathbf{C}$ at $\tau(t')$. Then it must be the case that the trajectory τ and the outer-normal of boundary of $\mathbf{C}$ should form an acute angle. It is known from Lemma 5.9 that at any intermediate state $\tau(t')$, the input U to the plant is correct. A contradiction is reached by showing that if $\tau(t')$ is on the boundary of $\mathbf{C}$, then the angle between the above-mentioned vectors is obtuse.

Finally, combining the Invariants proved above the safety property of the composed system can be proved.

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

Proof: For any reachable state s, if $s.mode = \texttt{usr}$ then $s \in \mathbf{R} \subseteq \mathbf{C}$ by Invariant 5.4. Otherwise $s.mode = \texttt{sup}$, and there are two possibilities: if $s.rt < \tau_{\texttt{act}}$ then, by Invariant 5.10, $s \in \mathbf{B}_{s.rt} \subseteq \mathbf{C}$. Else $s.rt \geq \tau_{\texttt{act}}$ and it follows from Invariant 5.11 that $s \in \mathbf{C}$.

6 Conclusions

In this paper we have presented and formally verified a supervisory controller for a model helicopter system. This supervisory controller allows the user to control in a safe operating region, beyond which it overrides the user, performs appropriate recovery, and returns control to the user. The duration of the recovery period has not been discussed here, it has been shown in [14] that this period is bounded. The size of the safe operating region, depends on the plant dynamics, sensor errors, sampling period, actuator bandwidth and saturation. An implementation of supervisory controller in the actual system is in progress.

The specification language used 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 aiding verification of hybrid systems. In verifying safety, all the invariants were proved in the assertional style. The proof techniques presented here demonstrate two properties which we believe are important for reasoning about complex systems: (1) the proofs are decomposed into discrete and continuous parts, which are independent of each other, and (2) the reasoning is based on current state of the system, rather than complete executions.

In the future we intend to design and verify a class supervisory controllers that reduce unnecessary interferences by utilizing additional information about the users controller. We also intend to evaluate the applicability of the proof methods, to hybrid systems with more complicated discrete behavior and dynamics, possibly using mechanical theorem provers.

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 *Coputer 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/htm/about/fs_about_splash.htm.

[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. submitted for journal publication, 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] Sayan Mitra, Yong Wang, Nancy Lynch, and Eric Feron. Application of Hybrid I/O Automata in safety verification of pitch controller for model helicopter system. Technical report, MIT Laboratory for Computer Science, Cambridge, MA 02139, October 2002. http://theory.lcs.mit.edu/mitras/research/QuanTR02.ps.

[14] Yong Wang, S. Mitra, C. Livadas, N. Lynch, and E. 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.

[15] 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.

Appendix

6.1 Activities satisfy trajectory closure properties

Lemma 6.1 *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 $\tau_i.lstate = \tau_{i+1}.fstate$. Let $\tau_i \in [\alpha_j]$, $\tau_{i+1} \in [\alpha_k]$, where $j, k \in \mathcal{I}$ and let us also assume for the sake of contradiction that $j \neq k$. From **R3**, $P_j^- \cap P_k^-$ must be empty. But it is known that $\tau_i.lstate \in P_j^- \cap P_k^-$, which contradicts the 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, $\tau_0 \frown \tau_1 \frown \tau_2 \frown \ldots \in \mathcal{T}$.□

6.2 Relation between C, R, U, and I

Property 1 *Over the interval $-\frac{\pi}{2} \leq \theta \leq \frac{\pi}{2}$ 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 $\mathcal{T} > 0$.*

Proof: Part (1): Using the definitions of Γ^+ and Γ^-.

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

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

The monotonicity properties can be shown by calculating the partial derivatives.

$$\frac{\partial\Gamma^+}{\partial\theta} = -\sqrt{\frac{2(U_{min} + \Omega^2\cos\theta_{max})}{(\theta_{max} - \theta + \frac{1}{2}U_{mag}\mathcal{T}^2)}} < 0 \quad and \quad \frac{\partial\Gamma^-}{\partial\theta} = -\sqrt{\frac{U_{max} + \Omega^2}{2(\theta - \theta_{min} + \frac{1}{2}U_{mag}\mathcal{T}^2)}} < 0$$

Part (2):

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

Using $U_{max} > \Omega^2$, $\theta \leq \theta_{max}$ and $U_{mag} = U_{min} + U_{max}$. We get

$$\frac{\partial\Gamma^+}{\partial\mathcal{T}} < -U_{mag} + U_{mag}\mathcal{T}\sqrt{\frac{U_{min} + U_{max}}{U_{mag}\mathcal{T}^2}} = 0 \quad i.e. \quad \frac{\partial\Gamma^+}{\partial\mathcal{T}} < 0.$$

Part (3)

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

Using $U_{max} > \Omega^2$, $\theta \geq \theta_{min}$ and $U_{mag} = U_{min} + U_{max} \geq U_{max}$. We get

$$\frac{\partial \Gamma^-}{\partial \mathcal{T}} < U_{mag} - U_{mag} \sqrt{\frac{U_{max} - \Omega^2}{U_{max} + U_{min}}} > 0 \quad i.e. \quad \frac{\partial \Gamma^-}{\partial \mathcal{T}} > 0.$$

Part (4) $\Gamma^+(\theta_{max}, \mathcal{T}) = -U_{mag}\mathcal{T} + \sqrt{(U_{min} + \Omega^2 \cos\theta_{max}) U_{mag} \mathcal{T}}$

$$\Gamma^+(\theta_{max}, \mathcal{T}) < -U_{mag}\mathcal{T} + U_{mag}\mathcal{T} = 0$$

and $\Gamma^-(\theta_{min}, \mathcal{T}) = -U_{mag}\mathcal{T} + \sqrt{(U_{max} - \Omega^2) U_{mag} \mathcal{T}}$

$$\Gamma^-(\theta_{max}, \mathcal{T}) > U_{mag}\mathcal{T} - U_{mag}\mathcal{T} = 0. \ \square$$

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

Proof: The set inclusions are trivial to prove by using the monotonicity properties of Property 1. $\mathbf{I} \subseteq \mathbf{U}$ and $\mathbf{R} \subseteq \mathbf{S}$ follow directly from the definition of these regions. To prove that $\mathbf{U} \subseteq \mathbf{R}$, let us consider a point $p = (\theta, \dot{\theta}) \in \mathbf{U}$. As $p \in \mathbf{U}$, it must satisfy $\theta_{min} + \epsilon_0 \leq \theta \leq \theta_{max} - \epsilon_0$. Since $\epsilon_0 \geq 0$ we have $\theta_{min} \leq \theta \leq \theta_{max}$. Also $\Gamma^-(\theta - \epsilon_0, \mathcal{T} + \Delta) + \epsilon_1 \leq \dot{\theta} \leq \Gamma^+(\theta + \epsilon_0, \mathcal{T} + \Delta) - \epsilon_1$. Since $\epsilon_1 y > 0$, implies $\Gamma^-(\theta - \epsilon_0, \mathcal{T} + \Delta) \leq \dot{\theta} \leq \Gamma^+(\theta + \epsilon_0, \mathcal{T} + \Delta)$. From Property 1 we know that Γ^+ and Γ^- decrease with respect to θ, so $\Gamma^-(\theta, \mathcal{T} + \Delta) \leq \dot{\theta} \leq \Gamma^+(\theta, \mathcal{T} + \Delta)$. Also from Property 1 Γ^+ and Γ^- respectively increase and decreases monotonically with respect to $\mathcal{T}$ and $\Delta > 0$ so $\Gamma^-(\theta, \mathcal{T}) \leq \dot{\theta} \leq \Gamma^+(\theta, \mathcal{T})$.i.e. $p \in \mathbf{R}$. $\square$

6.3 Preliminary Properties

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

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 $s'.buffer = s.buffer + (u, s.now + \tau_{\texttt{act}})$. Since $s'.now = s.now$, it follows from the inductive hypothesis that $s'.now \leq s'.buffer.\texttt{nexttolast.st} \leq s'.buffer.\texttt{last.st} \leq s'.now + \tau_{\texttt{act}}$. Therefore $s'.now \leq s'.buffer\texttt{[i].st} \leq s'.buffer\texttt{[i+1].st} \leq s'.now + \tau_{\texttt{act}}$, for all $0 \leq i < s'.buffer.\texttt{size} - 1$.

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

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 $s.now \leq s.buffer\texttt{[i].st} \leq s.buffer\texttt{[i+1].st} \leq s.now + \tau_{\texttt{act}}$, for all $0 \leq i < s.buffer.\texttt{size} - 1$. From the code it follows that $s'.now = s.now + t'$ and $s'.buffer = s.buffer$. We claim that $s'.now \leq s'.buffer.\texttt{head.st}$ and therefore the invarint holds at s'. Suppose this was not the case, that is $s'.now > s'.buffer.\texttt{head.st}$. Then there would exist $t'' \in \tau.dom$ such that $t'' < t'$ and $\tau(t'').now = s'.buffer.\texttt{head.st}$. Since $\tau(t'')$ satisfies the stopping condition for activity *d2a* therefore $\tau.ltime = t''$, which contradicts our assumption.$\square$

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 $0 \leq s'.time\_left \leq \Delta$. □

6.4 User mode

Lemma 5.2 *The regions $\mathbf{A}_t$ satisfy the following:*

1. $\mathbf{A}_0 = \mathbf{R}$, 2. $\mathbf{U} \subseteq \mathbf{A}_\Delta$, *and* 3. *If $0 \leq t \leq t' \leq \Delta$ then $A_{t'} \subseteq A_t$.*

Proof: For part 1, set $t = 0$ in the definition of $\mathbf{A}_t$. For part 2, $\theta_s^0 - \epsilon_0 \leq \theta_p^0 \leq \theta_s^0 + \epsilon_0$ and $\theta_s^1 - \epsilon_1 \leq \theta_p^1 \leq \theta_s^1 + \epsilon_1$. Setting $t = \Delta$ it follows that:

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

From Property 1, $\theta_p^0 \geq \theta_s^0 - \epsilon_0 \implies \Gamma^-(\theta_p^0, y) \leq \Gamma^-(\theta_s^0 - \epsilon_0, y)$ and $\theta_p^0 \leq \theta_s^0 + \epsilon_0 \implies \Gamma^+(\theta_p^0, y) \geq \Gamma^+(\theta_s^0 + \epsilon_0, y)$. Therefore,

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

The left hand side is equal to $\mathbf{U}$ as defined in equation (4). For part 3, we observe that in the definition of $\mathbf{A}_t$, Γ^+ and Γ^- are monotonically decreasing and monotonically increasing respectively with respect to t. Therefore if $0 \leq t \leq t' \leq \Delta$ then $\mathbf{A}_{t'} \subseteq \mathbf{A}_t$. □

Invariant 5.3 *In any reachable state s, if $s.mode = \texttt{usr}$ and $\neg s.ready$ then $s \in \mathbf{A}_{s.time\_left}$.*

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 $s'.ready = \texttt{true}$ 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 5.3, $s' \in \mathbf{A}_{s.time\_left} - t' = \mathbf{A}_{s'.time\_left}$. □

6.5 Supervisor mode

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

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 $s.mode = \texttt{usr}$. We consider the discrete steps $s \xrightarrow{\pi} s'$ with $s'.mode = \texttt{sup}$ and $s'.\theta_s^1 > I^+(s'.\theta_s^0)$.

> Case 1: $\pi = sample$. Since $s.ready = \mathsf{false}$ and $s.mode = \texttt{sup}$, 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 $s'.ready_d = \mathsf{true}$. Since *sample* does not change $ready_d$, it follows that

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

Case 3: π =*command*. Part 1(b): From the code it follows that $s.mode$ = sup and $s.\theta^1_s > I^+(s.\theta^0_s)$. Therefore it follows from Invariant 5.6 that $s.U_{sup} = U_{min}$. Since $s'.buffer = s.buffer + (s.U_{sup}, s.now + \tau_{\mathtt{act}})$, 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: π = *dequeue*. From the code it follows that $s.mode$ = sup, $s.\theta^1_s > I^+(s.\theta^0_s)$, $s'.buffer = s.buffer$.tail, and that $s.ready_d$ = 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 $s'.ready$ = false therefore the invariant holds trivially.

For the continuous part, consider a closed trajectory τ, with $t' = \tau.ltime$, $s'.mode$ = sup and $s'.\theta^1_s > I^+(s'.\theta^0_s)$. From the code it follows that $s'.buffer = s.buffer$, $s.\theta^1_s > I^+(s.\theta^0_s)$ 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 $s'.ready_d$ = true. Therefore $s.ready_d$ = 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. □

Invariant 6.5 *In any reachable state s with $s.mode$ = sup $\wedge$ $s.rt \geq \tau_{\mathtt{act}}$*

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

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

Invariant 5.9 *In any reachable state s, such that $s.mode$ = sup and $s.rt > \tau_{\mathtt{act}}$*
1. If $s.\theta^1_s > I^+(s.\theta^0_s)$ then $s.U = U_{min}$, and 2. If $s.\theta^1_s < I^-(s.\theta^0_s)$ 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 $s.mode$ = usr. Consider discrete transitions $s \xrightarrow{\pi} s'$ with $s'.mode$ = sup, $s'.rt > \tau_{\mathtt{act}}$, and $s.\theta^1_s > I^+(s.\theta^0_s)$. Since none of the discrete steps change rt, it follows that $s.rt > \tau_{\mathtt{act}}$.

Case 1: π = *sample*. Since $s.ready$ is false and $s.mode$ = sup, it follows from the contrapositive of Invariant 5.5 that $s.\theta^1_s > I^+(s.\theta^0_s)$ or $s.\theta^1_s < I^-(s.\theta^0_s)$. According to Assumption 3, $s.\theta^1_s \geq I^-(s.\theta^0_s)$, therefore $s.\theta^1_s > I^+(s.\theta^0_s)$. 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: π =*control*. We claim that $s.mode$ = sup. The invariant is preserved since π does not change any of the variables involved other than *mode*. If $s.mode$ = usr then $s.rt = 0 = s'.rt$, which contradicts our assumption that $s'.rt > \tau_{\mathtt{act}}$.

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

Case 4: π = *dequeue*. From the code it follows that $s.mode$ = sup and $s.\theta^1_s > I^+(s.\theta^0_s)$. 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 $s.buffer$.head.u$= U_{min}$, by Invariant 6.5. 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 $s'.mode = \texttt{sup}$, $s'.rt \geq \tau_{\texttt{act}}$ and $s'.\theta_s^1 > I^+(s'.\theta_s^0)$. We claim that $s.rt \geq \tau_{\texttt{act}}$. 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 $\tau(t'').rt = \tau_{\texttt{act}}$. From Lemma 5.1 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 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 normal and the tangent vectors to a curve at the point (x, y) are denoted by $\mathbf{n}(x, y)$ and $d(x, y)$ respectively.

Invariant 5.11 *In any reachable states s, if $s.mode = \texttt{sup}$ and $s.rt \geq \tau_{\texttt{act}}$ then $s \in \mathbf{C}$.*

Proof: The base case is trivially satisfied because $s.mode = \texttt{usr}$. For the discrete part, consider discrete transitions $s \xrightarrow{\pi} s'$ with $s'.mode = \texttt{sup}$. If $\pi =$*control* there are two subcases: if $s.mode = \texttt{sup}$ then from the inductive hypothesis $s \in \mathbf{C}$. Therefore using Property 3 it follows that $s' \in \mathbf{C}$. Otherwise $s.mode = \texttt{usr}$ 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 $s'.mode = \texttt{sup}$ and $s'.rt \geq \tau_{\texttt{act}}$. We claim that $s \in \mathbf{C}$. From Property 3 it is known that $s.mode = \texttt{sup}$, Consider two possible cases: (1) If $s.rt < \tau_{\texttt{act}}$ then from Invariant 5.10 it follows that $s \in \mathbf{C}$. Otherwise (2) $s.rt \geq \tau_{\texttt{act}}$ and from the inductive hypothesis it follows that $s \in \mathbf{C}$.

If $s \in \mathbf{U}$, then from Lemma 5.3 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 5.9 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 $V_1(\theta_p^0, \theta_p^1) = \frac{1}{2}{\theta_p^1}^2 + \left(-U_{min} + \Omega^2 \cos\theta_{max}\right) \theta_p^0$. 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 $rt > \tau_{\texttt{act}}$ therefore $U = U_{min}$ by Invariant 5.9. 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

$$\sin\theta_p^0 = \sin\theta_{max} + \frac{1}{\Omega^2} \left[U_{min} (\theta_p^0 - \theta_{max}) - \frac{1}{2} {\theta_p^1}^2 \right]. \tag{8}$$

This implicit function defines an integral curve $\theta_p^0 = F_1(\theta_p^1)$. Differentiating (8) with respect to θ_p^1, we get

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

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 $\mathbf{C}^l = \{d(\theta_p^0, \theta_p^1) | \theta = \theta_{min} \wedge 0 < \theta_p^1 < \Theta^+\}$,
where $\Theta^+ \triangleq \sqrt{2\left(-U_{min} + \Omega^2 \cos\theta_{max}\right)(\theta_{max} - \theta_{min})}$. 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 $\mathbf{C}$.□