A Deductive Approach towards Controller Synthesis[1]

Zohar Manna and Henny B. Sipma
Computer Science Department, Stanford University
Stanford, CA 94305
e-mail: zm@cs.stanford.edu, sipma@cs.stanford.edu

Abstract

This paper proposes a deductive approach towards controller synthesis. The system to be controlled and the environment are modeled as *phase transition systems* and the specification of the entire system is expressed by a *hybrid temporal logic formula*. Verification rules are used to determine whether the system in conjunction with a *control strategy* meets the specification. The control strategy is refined until it meets the given specification.

1. Introduction

Analytic synthesis of digital control programs for continuous systems is well established (see e.g. [1]). However, synthesis of programs to control hybrid systems, i.e., systems with a nontrivial mixture of continuous and discrete components, is still a relatively new area of research, approached both from control theory and from computer science. Computer scientists distinguish between automata-theoretic and logical-deductive approaches; our approach fits in the latter category. Since we are dealing with continuous dynamics we necessarily have to use analytic methods as well, however they are subordinate to the logical-deductive methods.

Our approach is based on the framework for specification and verification of reactive systems as presented in [4]. For each type of system, i.e., qualitative reactive systems, real-time and hybrid systems, this framework provides

- A *computational model* defining the set of system behaviors (computations);
- A *requirement specification* language for specifying properties of systems;
- A *system description* language;
- A set of *verification rules* by which valid properties of systems can be verified.

Each of these components have been adapted slightly for our purposes. In particular, the computational model and the system description language have been modified to cater especially to the modeling and description of controlled systems. We use the *actor phase transition system* APTS [6] as a computational model. It is based on the abstract phase transition system presented in [2] and augmented with the notion of actors and their ability to observe and manipulate variables. As a requirement language we use *hybrid temporal logic* HTL [2] and as a system description language we use the graphical language of *hierarchical actor activity graphs* HAAG [6].

To make this paper self-contained we give a brief explanation of these components in the next section. In Section 3 we present the steps in the controller design method. Section 4 illustrates the approach with an engineering example and Section 5 concludes the paper with a brief discussion.

2. Preliminaries

2.1. Computational Model

As computational model for controlled systems we use the Actor Phase Transition System (APTS). An APTS Φ has the following components:

- V: a finite set of *state variables*, partitioned into V_d, the set of *discrete variables*, V_c, the set of *continuous variables*, (i.e., variables that are continuous throughout a behavior) and V_h, the set of *hybrid variables*. Hybrid variables can both vary continuously and undergo discontinuous changes, but they are piece-wise continuous. A *state* is an interpretation of V; the set of all states is denoted by Σ.
- Θ: the *initial condition*; an assertion which must be satisfied by the initial state.

[1]This research was supported in part by the National Science Foundation under grant CCR-92-23226, by the Defense Advanced Research Projects Agency under NASA grant NAG2-892, by the United States Air Force Office of Scientific Research under grant F49620-93-1-0139, and by Department of the Army under grant DAAH04-95-1-0317.

- $\mathcal{T_S}$: a finite set of *synchronous transitions*. Each transition $\tau \in \mathcal{T_S}$ is a function $\tau : \Sigma \mapsto 2^{\Sigma}$ defined by a transition relation $\rho_\tau(V, V')$. A synchronous transition can modify only discrete or hybrid variables. Synchronous transitions correspond to the interactions between plant and controller; they are also referred to as *interaction instants*, since they do not advance time.
- $\mathcal{P}$: a set of *phases* over V. Each phase is a pair (b, f), where $b \in \Re^+$ is the length of an interval $[0, b)$, and $f = \{f^x | x \in V\}$ is a family of continuous functions defined on the interval $[0, b)$. A phase invariant is represented by an assertion of the form $\rho_\phi(V, \dot{V})$, referring to the state variables and their derivatives.
- $\mathcal{T_P}$: a set of *phase transitions*. Each transition $\tau \in \mathcal{T_P}$ is a binary relation on $\mathcal{P}$. We require for a pair of phases $(P_1, P_2) \in \tau \in \mathcal{T_P}$ with $P_1 = (b_1, f_1)$ and $P_2 = (b_2, f_2)$ that for all $x \in V_c$ $\lim_{t \uparrow b_1} f_1^x(t) = \lim_{t \downarrow 0} f_2^x(t)$
- $\mathcal{K}$: a finite set of actors. For controlled systems we consider three actors: *environment*, *plant*, and *controller*. Each of these can consist of a set of subordinate actors.
- $\mathcal{M} : V \mapsto \mathcal{K}$: the *modifiability* function. Each variable can be modified by exactly one actor. We also say the variable is owned by the actor. $\mathcal{M}$ induces a partition on V. The variables modifiable by the controller are denoted by $\mathcal{M}_c$.
- $\mathcal{O} : V \mapsto 2^{\mathcal{K}}$: the *observability* function. It indicates which variables are visible to which actors. We require that all variables modifiable by an actor are observable to that actor. The variables observable to the controller are denoted by $\mathcal{O}_c$.
- $\mathcal{B} : (\mathcal{T_S} \cup \mathcal{T_P}) \mapsto \mathcal{K}$: ownership of transitions. A transition is owned by exactly one actor. A transition owned by $k \in \mathcal{K}$ can only modify variables owned by k, and its enabling condition can only contain variables observable to k.

2.2. Behavior

A *behavior* of an APTS Φ is an infinite sequence

$$\sigma : \langle s_0, t_0 \rangle \overset{\alpha_1}{\to} \langle s_1, t_1 \rangle \overset{\alpha_2}{\to} \langle s_2, t_2 \rangle \overset{\alpha_3}{\to} \ldots$$

where

$$\begin{aligned} s_i &\in \Sigma \\ \alpha_i &\in \mathcal{K} \cup \{cont\} \\ t_i &\in \Re^+ \end{aligned}$$

that satisfies

Initiation: $s_0 \models \Theta,\ t_0 = 0$

Consecution: for each $j = 0, 1, 2, \ldots$ the step $\langle s_j, t_j \rangle \overset{\alpha_{j+1}}{\to} \langle s_{j+1}, t_{j+1} \rangle$ is either

- a synchronous transition: $t_j = t_{j+1}$ and $s_{j+1} \in \tau(s_j)$ for some $\tau \in \mathcal{T_S}$ and $\alpha_{j+1} = \mathcal{B}(\tau)$, or
- a continuous step: $t_{j+1} > t_j$ and $\langle s_{j+1}, t_{j+1} \rangle$ is a phase successor (defined below) of $\langle s_j, t_j \rangle$ and $\alpha_{j+1} = cont$.

Urgency No continuous step may disable a synchronous transition.

Time divergence Time grows beyond any bound.

A *phase successor* is defined as follows: $\langle s_{j+1}, t_{j+1} \rangle$ is a phase successor of $\langle s_j, t_j \rangle$ if there exists a finite phase sequence $P_0 \ldots P_n$, where each $P_i = (b_i, f_i) \in \mathcal{P}$ such that

$$\begin{aligned} s_j &= \lim_{t \to 0} f_0(t) \\ s_{j+1} &= \lim_{t \to b_n} f_n(t) \end{aligned}$$

$$\left(\sum_{i=0}^{n} b_i\right) = t_{j+1} - t_j$$

and for all i, $0 \le i < n$,

$$(P_i, P_{i+1}) \in \tau \text{ for some } \tau \in \mathcal{T_P}$$

Similar to the semantic model of [4] and unlike the model of [2], our model is not a function from time to states: the instants of plant-controller interaction are multi-valued, since time is not advanced by synchronous transitions.

2.3. Specification language

We use Hybrid Temporal Logic HTL [2] as our requirements specification language. Our semantics of this logic are given in [6]; in that paper we introduced a sampling semantics based on controller-plant interaction instants.

HTL has a number of temporal operators whose meanings can be defined inductively. To understand the remainder of this paper it is sufficient to know the meaning of the "always" operator, $\Box$, which is a derived operator in the logic. It was shown in [3] that for any state formula φ (i.e., φ does not contain any temporal operators) and phase $P = (b, f)$

$$P \models \Box\varphi \quad \text{iff} \quad \text{for all } t \in [0, b)\ \varphi \text{ holds at } t.$$

2.4. System Description Language

We use Hierarchical Actor Activity Graphs (HAAG), introduced in [6], as our system description language. HAAGs allow a graphical description of a system that we believe is readily accessible to the control engineer, yet it has a well-defined semantics. An algorithm to convert a HAAG into an APTS is given in [6]. An example of a HAAG is shown in Figure 4.

2.5. Control Strategy

Our concept of control strategy was introduced in [6]. A *control strategy* is a rule that tells the controller what action to impose on the plant based on what it has observed so far. A *controlled behavior* results when applying a control strategy. Formally, a *control strategy g* is a function from the set of finite behavior prefixes projected onto O_c, to Σ, where O_c is the set of variables that can be observed by the controller, i.e., $O_c = \{v \in V \mid \mathcal{O}(v) \in controller\}$. A sequence σ is a *g-controlled behavior* of a phase transition system Φ if

- σ is a behavior of Φ, and
- for all $m > 0$ if $\alpha_m \in controller$ then
  $g(\sigma|_m) = s_m$

where $\sigma|_m$ denotes the prefix of σ of length m. $\mathcal{C}_\Phi(g)$ is the set of all g-controlled behaviors of Φ with control strategy g.

2.6. Verification Rules

A verification rule allows us to prove a property of a system by proving simpler properties. As an example consider the rule INV [5] for qualitative reactive systems modeled by a transition system

$$
\begin{array}{ll}
S1. & \Theta \rightarrow \varphi \\
S2. & \varphi \wedge \rho_\tau \rightarrow \varphi' \quad \text{for all transitions } \tau \\
S3. & \varphi \rightarrow p \\
\hline
 & \Box p
\end{array}
$$

This rule states that if we can prove for a system Φ that the initial condition implies φ (S1), and that φ is preserved by all transitions (S2), then we may conclude that φ and, by (S3) p is true in all states of all behaviors of Φ. We call S1-S3 the premises of the rule, and $\Box p$ the conclusion of the rule.

Due to the interleaving of discrete transitions and continuous time steps verification rules for proving properties of hybrid systems are more complex. However, the basic idea is the same. In the control design procedure described in the next section we use a simplified version of the rule P-INV given in [3]. The rule presented here is applicable only to formulas $\psi(V)$ such that $V \subseteq V_c$. Its conclusion is the invariance of a point-based state formula.

$$
\begin{array}{lll}
P1. & \Theta \rightarrow \varphi(V) & \\
P2. & \varphi(V) \rightarrow \psi(V) & \\
P3. & \rho_\tau(V, V') \wedge \varphi(V) \rightarrow \varphi(V') & \forall \tau \in \mathcal{T} \\
P4. & \Box \rho_\phi(V, \dot{V}) \wedge \varphi(V) \Rightarrow_f \varphi(\overrightarrow{V}) & \forall \phi \in \Phi \\
\hline
 & \Box \psi(V) &
\end{array}
$$

P1 requires the initial condition to imply $\varphi(V)$. P3 states that all discrete transitions preserve $\varphi(V)$, and P4 states that if $\varphi(V)$ holds at the beginning of an interval and the phase invariant $\rho_\phi(V, \dot{V})$ holds throughout the interval then $\varphi(V)$ must hold at the end of that (finite) interval. If premises P1, P3 and P4 can be proved then we can conclude that φ holds in all states (i.e. at all timepoints) of a behavior, and if in addition we can prove P2, we can conclude that $\psi(V)$ holds at all timepoints.

3. Controller Synthesis

3.1. Motivation

The goal of controller synthesis is to obtain a control strategy g that, when imposed on the plant, results in the plant's meeting its specification. In other words, the control strategy should remove undesirable behaviors from the set of plant behaviors. Our design procedure is led by the following two considerations:

1. the control strategy should be as weak (i.e., unconstraining) as possible, to allow maximum flexibility for plant optimization;
2. the system description and environment assumptions should be constrained where possible.

This last consideration may seem to contradict the usual approach in controller design of considering the system and environment as a given, which cannot be changed. However, in practical design situations the system, and especially the environment assumptions, are often underspecified, which either goes unnoticed or additional assumptions are implicitly used in the controller design.

Consider for example a cooling water system for a rundown stream. A typical system description contains the range of the inlet temperatures of the stream. The specification states the desired range of outlet temperatures of the stream, and the environment assumption tells us what temperature range we may assume of the cooling water inlet. This is a common problem, and a controller can readily be designed that works well in practice. However when we apply formal verification to this system we discover that

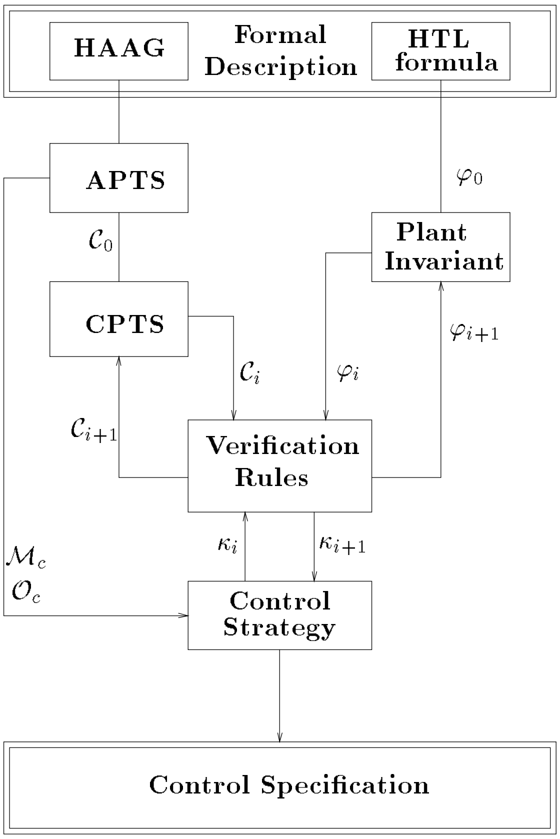

Figure 1: Steps in incremental controller design

in order to meet the specification with a reasonable control strategy we need a bound on the derivative of the temperature of the inlet cooling water. The temperature of a large body of water does not change rapidly, so this assumption is reasonable in practice. One could say that in the process of controller design we refine our assumptions as well.

3.2. Design procedure

The steps in the design procedure are shown schematically in Figure 1. We start with a formal description of the system and its environment in the form of a HAAG and a specification expressed as an HTL formula. The HAAG is converted into an APTS, which in turn is converted into a CPTS $\mathcal{C}_0$[1]. We take the specification, φ, as our initial plant invariant φ_0. Since it is our objective to find the weakest control strategy that makes the plant meet its specification, φ, we take the initial control strategy, κ_0, to be *true* (i.e., there are no constraints on the values of the manipulated variables).

After the initialization steps we iteratively refine the control strategy, the CPTS and the plant invariant as follows. We apply the verification rule P-INV in an attempt to establish

$$\mathcal{C}_i, \kappa_i \overset{?}{\models} \varphi_i.$$

If the proof does not go through there must be some premise of the rule that we cannot prove. We analyze this premise and refine the system in one of the following ways:

1. constrain the environment:
   $\mathcal{C}_{i+1} := \mathcal{C}_i \wedge \Gamma_i$
2. add an additional state to the system:
   $\mathcal{C}_{i+1} := \mathcal{C}_i \vee C.$
   Application of this procedure is usually accompanied by constraining the control strategy. Combined, they enable forcing the system outside of $\mathcal{C}_i$ before it violates φ_i.
3. constrain the control strategy:
   $\kappa_{i+1} := \kappa_i \wedge \gamma_i$
   such that $\Theta \rightarrow \mathcal{C}_i \wedge \kappa_{i+1}$ still holds to ensure that the system can start in its initial state.
4. strengthen the invariant:
   $\varphi_{i+1} := strengthen(\varphi_i).$
   Strictly speaking, this procedure does not refine the system; however, it is included, because it often facilitates subsequent refinement steps.

When the proof does go through for some $\mathcal{C}_n, \kappa_n(V')$, and φ_n, where V' denotes the variables that the control strategy depends on, the refined system with control strategy $\kappa_n(V')$ meets the specification. It remains to determine a control strategy $\kappa_n(\mathcal{O}_c)$ that only refers to the variables observable to the controller, such that $\forall V : \kappa_n(\mathcal{O}_c) \rightarrow \kappa_n(V')$. Depending on the plant and environment phase relations, $\kappa_n(\mathcal{O}_c)$ can be determined analytically, or alternatively may be able to be approximated. It is also possible that such $\kappa_n(\mathcal{O}_c)$ does not exist, in which case we have to return to the iteration step, possibly undo some refinements and explore a different decision path.

3.3. Dependency Graph

An attractive byproduct of this synthesis procedure is a dependency structure between properties of the system on one side and system design parameters and environment assumptions on the other. Knowledge of such dependencies is often useful both in design (in particular in redesign and debottlenecking studies) and in operation and maintenance. During operation it is not unusual that environment assumptions used in the design of the system temporarily cease to hold. Knowledge of which properties may be compromised by these violations can aid in the decision

[1]A CPTS (Concrete Phase Transition System) [3] is essentially the cross product of an APTS over all actors.

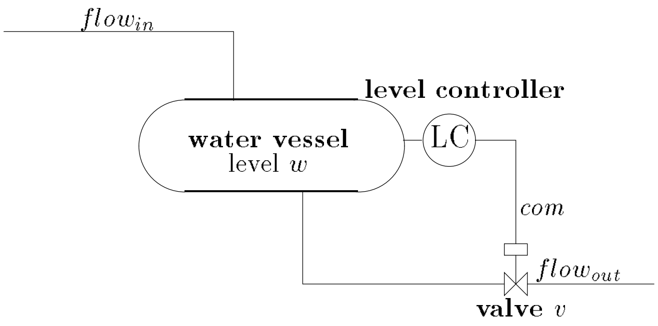

Figure 2: Water-level system

whether to shut down a system under such circumstances. E.g., one may choose to live with a reduction in yield, but cannot accept compromising a safety-related property.

4. Example: Water-level control system

4.1. Engineering Description

We illustrate our design and verification methodology with a generalized version of the water-level system given in [2]. The system, shown in Figure 2, consists of a vessel with cross section A, one incoming line and one outgoing line. Water is coming in with an unknown rate $flow_{in}$ between 0 and M units/min. The water level w in the vessel is regulated by a controller that manipulates a valve of size c_v (with valve opening v) in the outgoing line; the valve determines the outgoing flow $flow_{out}$. The valve accepts two commands: $com = on$ and $com = off$, and can be in any of four modes: opening, closing, open, and closed. The requirements state that the water level should stay between H and L.

4.2. Formal Description

A formal description of the system is shown in Figure 4. The nodes drawn with double lines represent the actors, the nodes drawn with single lines represent phases. The set of variables shown for each actor node denotes the variables that can be modified by that actor. Edges between actors are labeled by variables that are observable to the target of the edge. Edges between phase nodes are labeled by guarded commands, i.e., the system can move from the source to the target node if the guard (enabling condition) is true.

The requirements are expressed by the HTL-formula

$$\Box(w \geq L \ \wedge \ w \leq H)$$

$$
\begin{array}{rcl}
\textbf{Vessel} & \stackrel{\text{def}}{=} & \dot{w} = \frac{1}{A}(flow_{in} - c_v \cdot v) \\
\textbf{Valve} & \stackrel{\text{def}}{=} & (v \geq 0) \wedge (v \leq 1) \\
\textbf{FlowH} & \stackrel{\text{def}}{=} & flow_{in} \leq M \\
\textbf{FlowL} & \stackrel{\text{def}}{=} & flow_{in} \geq 0 \\
\textbf{HighInv} & \stackrel{\text{def}}{=} & w \leq H \\
\textbf{LowInv} & \stackrel{\text{def}}{=} & w \geq L \\
\textbf{Closed} & \stackrel{\text{def}}{=} & v = 0 \\
\textbf{Open} & \stackrel{\text{def}}{=} & v = 1 \\
\textbf{Opening} & \stackrel{\text{def}}{=} & \dot{v} = 1 \\
\textbf{Closing} & \stackrel{\text{def}}{=} & \dot{v} = -1
\end{array}
$$

Figure 3: Propositions used in the definition of the water-level system

4.3. Design Procedure

First, we have to obtain $\mathcal{C}_0$, φ_0 and κ_0. Since $\mathcal{C}_0$ is the actor product of the APTS, the phase relations of $\mathcal{C}_0$ are a conjunction of the phase relations of the plant and the environment assumptions. To facilitate the presentation of the proof obligations we use the propositions defined in Figure 3 to represent the various conjuncts of the phase relations.

Thus we have

$$\mathcal{C}_0 : \Box\textbf{Vessel} \wedge \Box\textbf{Valve} \wedge \Box\textbf{FlowL} \wedge \Box\textbf{FlowH} \wedge \Box\textbf{Closed}$$

$$\kappa_0 : \text{true}$$

$$\varphi_0 : \Box(\textbf{LowInv} \ \wedge \ \textbf{HighInv})$$

Application of the verification rule P-INV [3] to establish

$$\mathcal{C}_0, \kappa_0 \stackrel{?}{\models} \varphi_0$$

results in the following two proof obligations for an arbitrary interval:

$$\Box\textbf{Vessel} \wedge \Box\textbf{Valve} \wedge \Box\textbf{FlowL} \wedge \Box\textbf{Closed} \wedge \textbf{LowInv} \ \rightarrow \ \overrightarrow{\textbf{LowInv}}$$

$$\Box\textbf{Vessel} \wedge \Box\textbf{Valve} \wedge \Box\textbf{FlowH} \wedge \Box\textbf{Closed} \wedge \textbf{HighInv} \ \rightarrow \ \overrightarrow{\textbf{HighInv}}$$

where $\Box\varphi$, φ, and $\overrightarrow{\varphi}$ mean that φ holds throughout, at the beginning, and at the end of the interval respectively. The first of these formulas is valid. Intuitively, if we keep the valve of the outgoing line closed, the level is strictly non-decreasing. The second formula is not valid. However, the following formula is valid, where M denotes the maximum inletflow. This equation expresses the fact that the invariant is preserved if no flow enters the vessel.

$$\begin{aligned}&\Box\textbf{Vessel} \wedge \Box\textbf{Valve} \wedge \Box\textbf{FlowH} \wedge \Box\textbf{Closed}\\ &\quad \wedge \textbf{HighInv} \;\rightarrow\; \overrightarrow{\textbf{HighInv}} \vee \neg(M=0)\end{aligned}$$

We now have two options. We can either accept the condition $M = 0$, i.e., the maximum inletflow is zero, or we can add control states to the system.

Since it is unlikely that the condition $M = 0$ is acceptable in practice, we add control states and force the system out of the offending behavior by adding a conjunct to the control strategy as follows:

$$\begin{aligned}\mathcal{C}_1 :\; &\Box\textbf{Vessel} \wedge \Box\textbf{Valve} \wedge \Box\textbf{FlowL} \wedge \Box\textbf{FlowH} \wedge\\ &\Box(\textbf{Closed} \vee \textbf{Closing} \vee \textbf{Opening} \vee \textbf{Open})\end{aligned}$$

$$\kappa_1 :\; w \geq H \;\rightarrow\; com = on$$

$$\varphi_1 :\; \Box(\textbf{LowInv} \wedge \textbf{HighInv})$$

Application of the verification rule P-INV to this new system gives us similar proof obligations, and leaves us with the following options:

1. Accept $(flow_{in} \geq c_v \cdot v)$
2. Add $(w \leq L \;\rightarrow\; com = \mathit{off})$ to κ_1

If we choose option 2, four more refinement steps result in the following:

$$\begin{aligned}\mathcal{C}_6 :\; &\Box\textbf{Vessel} \wedge \Box\textbf{Valve} \wedge \Box\textbf{FlowL} \wedge \Box\textbf{FlowH} \wedge\\ &\Box(\textbf{Closed} \vee \textbf{Closing} \vee \textbf{Opening} \vee \textbf{Open})\end{aligned}$$

$$\begin{aligned}\kappa_6 :\; &w - \tfrac{1}{2}\tfrac{c_v}{A}v^2 \leq L \;\rightarrow\; com = \mathit{off}\\ &\wedge\\ &w - \tfrac{M}{A}v + \tfrac{1}{2}\tfrac{c_v}{A}v^2 \geq H \;\rightarrow\; com = on\end{aligned}$$

$$\varphi_6 :\; \Box(w - \tfrac{1}{2}\tfrac{c_v}{A}v^2 \geq L \wedge w - \tfrac{M}{A}v + \tfrac{1}{2}\tfrac{c_v}{A}v^2 \leq H)$$

When the verification rule is applied to establish

$$\mathcal{C}_6, \kappa_6 \overset{?}{\models} \varphi_6$$

all premises are satisfied, indicating that this system satisfies the original specification.

Finally we have to determine $\kappa_6(\mathcal{O}_c)$ with $\mathcal{O}_c = \{w\}$ such that $\kappa_6(\{w\}) \rightarrow \kappa_6(\{w, v\})$. A control strategy that satisfies this condition is the following:

$$\kappa_6(\{w\}) = \begin{bmatrix} w \leq L + \frac{1}{2}\frac{c_v}{A} & \rightarrow & com = \mathit{off}\\ & \wedge & \\ w \geq H - \frac{1}{2}\frac{M^2}{c_v A} & \rightarrow & com = on\end{bmatrix}$$

Thus we have obtained a control strategy that makes the water-level system meet its specification.

5. Discussion

We have presented a deductive procedure for controller synthesis. Our methodology offers several advantages. It provides an integrated framework for designing and verifying an entire control system that operates across all operating regions, rather than a piece-wise design and validation as is common practice today. It is our objective to automate this procedure as much as possible; however at this point it still relies on insight and judgement from the control engineer.

Acknowledgements
We thank Arjun Kapur who suggested turning the design procedure into an algorithm and proposed the diagram of Figure 1. We thank Anca Browne and Tomás Uribe for their careful reading of the paper and their many helpful suggestions.

References

[1] G.F. Franklin, J.D. Powell, and M.L. Workman. *Digital Control of Dynamic Systems.* Addison Wesley, second edition, 1992.

[2] T.A. Henzinger, Z. Manna, and A. Pnueli. Towards refining temporal specifications into hybrid systems. In R.L. Grossman, A. Nerode, A. Ravn, and H. Rischel, editors, *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 60–76. Springer-Verlag, 1993.

[3] A. Kapur, T.A. Henzinger, Z. Manna, and A. Pnueli. Proving properties of hybrid systems. In *Proc. Third Int. Symposium on Formal Techniques in Real Time and Fault Tolerant Systems*, volume 863 of *Lecture Notes in Computer Science*, pages 431–454, Lübeck, Germany, September 1994. Springer-Verlag.

[4] Z. Manna and A. Pnueli. Models for reactivity. *Acta Informatica*, 30:609–678, 1993.

[5] Z. Manna and A. Pnueli. *Temporal Verification of Reactive Systems: Safety.* Springer-Verlag, New York, 1995.

[6] H.B. Sipma and Z. Manna. Specification and verification of controlled systems. In *Proc. Third Int. Symposium on Formal Techniques in Real Time and Fault Tolerant Systems*, volume 863 of *Lecture Notes in Computer Science*, pages 641–659, Lübeck, Germany, September 1994.

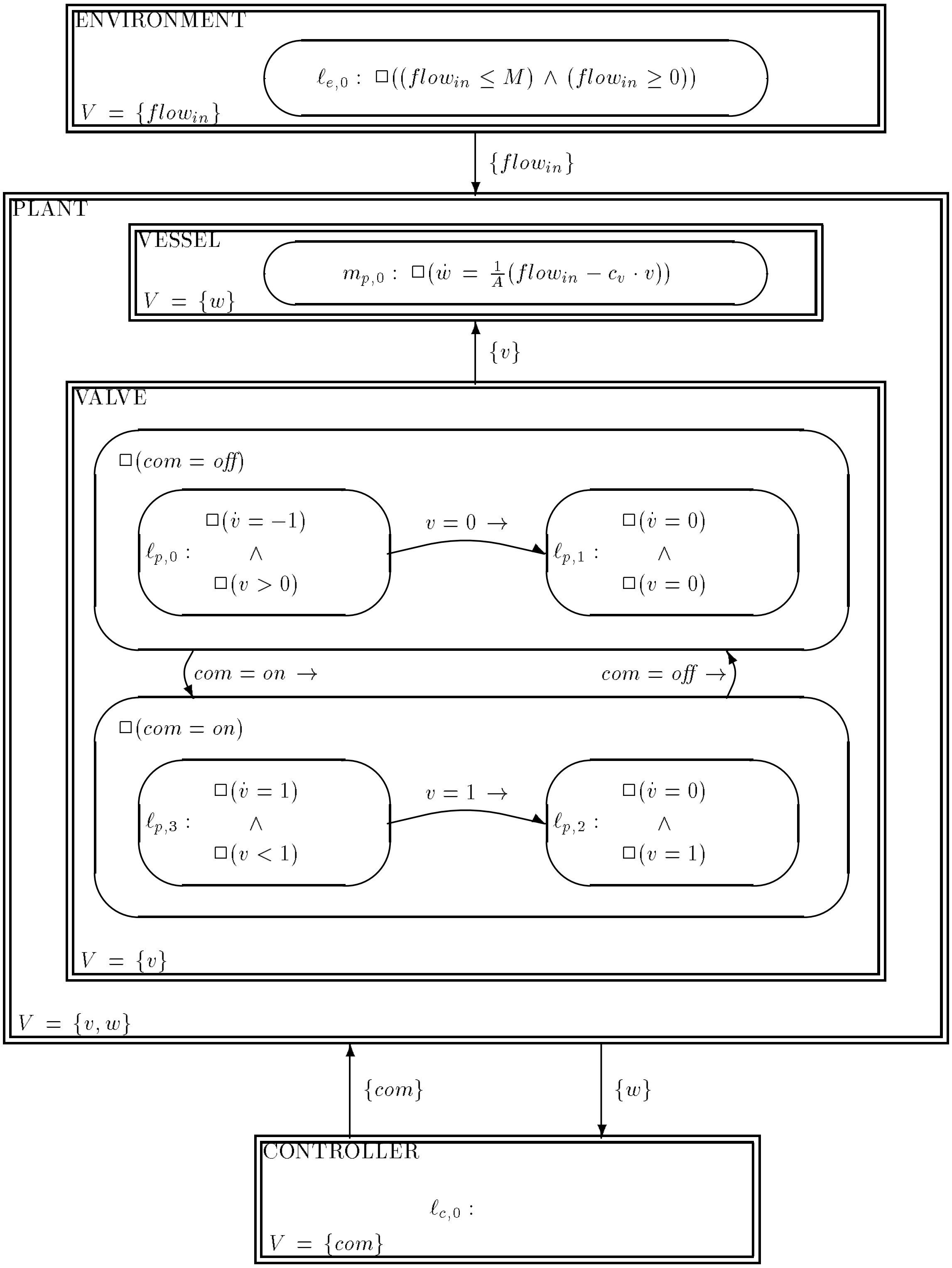

Figure 4. HAAG of water-level system