Discrete Conservative Approximations of Hybrid Systems*

Andrew K. Martin
Carl-Johan H. Seger
Integrated Systems Design Laboratory
Department of Computer Science
University of British Columbia
Vancouver, B.C. V6T 1Z4 Canada

Abstract

Systems that are modeled using both continuous and discrete mathematics are commonly called hybrid systems. Although much work has been done to develop frameworks in which both types of systems can be handled at the same time, this is often a very difficult task. Verifying that desired properties hold in such hybrid models is even more daunting. In this paper we attack the problem from a different direction. First we make a distinction between two models of the system. A detailed model is developed as accurately as possible. Ultimately, one must trust in its correctness. An abstract model, which is typically less detailed, is actually used to verify properties of the system. The detailed model is typically defined in terms of both continuous and discrete mathematics, whereas the abstract one is typically discrete. We formally define the concept of conservative approximation, a relationship between models, that holds with respect to a translation between the questions that can be asked of them. We then progress by developing a theory that allows us to build a complicated detailed model by combining simple primitives, while simultaneously, building a conservative approximation, by similarly combining pre-defined parameterised approximations of those primitives.

1 Introduction

In our research laboratory we have a computer controlled model train set consisting of about 35 feet of track, 13 computer controlled switches, three remotely controlled trains and approximately 60 position sensors. Although it was built to provide a test bed for designing mixed hardware/software systems, the train set has become a source of many challenging research questions related to hybrid systems. While some of the problems encountered are specific to our particular model train set, it is clear that others, such as old sensor data, noise, unreliable sensors and actuators are typical of other real systems as well.

*This research was supported by a Killam predoctoral fellowship, by a postgraduate scholarship and by operating grant OGPO 109688 from the Natural Sciences and Engineering Research Council of Canada and by a fellowship from the B. C. Advanced Systems Institute.

One particularly challenging problem has been the development of a control system that guarantees freedom from collisions. Of course, there is a trivial solution to this task: never move any trains! The challenge, however, is to guarantee collision freedom *while at the same time* keeping the degree of utilisation as high as possible. While working on this problem we encountered two difficulties. The first was formalising the above problem statement. The second was verifying that a proposed solution was indeed correct. Partly because of our circuit verification background, and partly because of the nature of the specification, a model-checking verification approach seemed natural. However, such an approach would require a discrete state-machine model of the system.

Although, we could imagine various discrete state machine models of the train system, it was hard to be convinced that such models were accurate reflections of the physical reality. Moreover, we needed some way to rephrase the question "can two trains collide" into a verifiable property of such a state machine. Many aspects of the system, including what it means for two trains to collide are much easier to model using continuous mathematics. For example, the current position of an engine is, as a first approximation, a linear function of the speed of the train, its last known location, and the time that has elapsed since its location was known. On the other hand, any model of the system must also include the hard-wired digital control logic that is used to interface the computer with the train set. This control logic is modelled more naturally using discrete structures. While we needed to model the physical system using a combination of continuous and discrete mathematics, we had efficient verification procedures only for discrete state systems.

To capitalise on the strengths of both forms of representation, we have developed a theory of "conservative approximation". Loosely speaking, a discrete state model is a conservative approximation of a hybrid system if any verifiable safety properties of the discrete state model also hold in the hybrid system. One observation we made early in our effort to formalise this idea was that it is not enough merely to establish a correspondence between the two systems. Of equal importance is the establishment of a correspondence between the questions we can ask of one system and the questions we can ask of the other. Intuitively, we need some way of translating our "two trains do not occupy the same physical space" (which is a more precise statement of no-collision) question, to a question about the discrete state model of the system. Thus a conservative approximation must be defined in terms of a relation between the questions, as well as a relation between the models. It is the formalisation of these concepts that is the topic of the first part of this paper.

To prove that one model is a conservative approximation of another with respect to some question-translation, can be very difficult and time consuming. If we had to establish conservative approximation from the basic definitions every time we wanted to develop a model of a system, this approach would be completely impractical. To address this difficulty, we develop a calculus that allows us to build up conservative approximations of complex systems from many simpler approximations.

Section 3 develops simple discrete and continuous modeling frameworks. We illustrate these frameworks using a very simple water-tank example system. We show how to model the continuous tank, its event-driven valve actuator, and the discrete controller within the continuous framework. This section also contains the definitions of parallel composition and communication operations, with which the components are connected to build a complete model of the water-tank system. Section 4 gives a very general definition of conservative approximation. In Section 5, 3.3 and 3.4, we show how to construct conservative approximations in our discrete modeling framework, for the hybrid models developed in

Section 3. Finally, in Section 8, we show how to use the discrete conservative approximation, and conventional model-checking to verify that the controller will not over-fill the water-tank.

This paper should be seen as the first step towards building a practical system that would allow the user to construct natural and accurate models of physical plants by combining simple, well understood, components. At the same time as the model is constructed, the system would automatically construct a parameterised finite state conservative approximation automatically. Desired properties of the physical system could then be verified against this approximation.

2 Related Work

The idea of conservative approximation has been presented in other contexts by several other researchers. Burch [3] gives a very general notion of conservative approximation based on trace algebras. The theory appears particularly well suited to verifying the real-time behaviour of discrete event systems. Examples given are mostly from the domain of circuit design.

In this framework, a trace algebra is defined as an algebra with composition and projection operators that satisfies a set of eight axioms. A trace is an element in the domain of such a trace algebra. Both implementation and specification are represented by trace structures, themselves the domain of a trace structure algebra. Each trace structure includes a set of traces. Each trace in the set represents a possible behaviour of the agent being modeled. Verification amounts to showing that the trace-set of the implementation is contained in the trace-set of the specification.

A conservative approximation is a function from verification problems expressed in one trace structure algebra, to problems in another. It consists of a pair of functions, Ψ_u and Ψ_l, each of which take a trace-structure in the source algebra, and return a trace-structure in the destination algebra. Let E be any expressions in the trace structure algebra. Let E' be the expression derived from E by replacing each occurrence of a trace-structure T with $\Psi_(T)$. The pair $\Psi = (\Psi_u, \Psi_l)$ is a conservative approximation if $E' \subseteq \Psi_l(T_s)$, implies that $E \subseteq T_s$, for any such expression E and trace T_s.

The theory is applied to a variety of trace structures all of which associate varying amounts of timing information with discrete events. Burch gives a general method for constructing conservative approximations based on homomorphisms between trace-algebras. This method allows the construction of conservative approximations from traces in which time is represented by real numbers, to traces in which time has discrete values. A second-method, based on power-set algebras over trace algebras, is used to construct conservative approximations from discrete time traces with explicit simultaneity, to traces with only interleaving semantics.

The theory given is very general in its formation, and appears particularly well suited to verifying the real-time behaviour of discrete event systems. It is not clear how to instantiate the theory for traces that are essentially real-valued functions over time, rather than sequences of discrete events.

Clarke *et al.* [4] describe an approach to approximate abstraction in the context of finite-state transition systems. Programs and their abstractions are modelled as finite state transition systems. An abstraction is defined by a surjection from the concrete state-space to the abstract. Specifications are given in subsets of the temporal logic CTL. Atomic state formulae in the logic refer only to the abstract state. The abstraction surjection is used to provide a natural interpretation of such formulae in the concrete domain. Thus the surjection provides both a translation for models, and a translation for specification. The authors show that such abstractions are conservative when the specification language is limited to $\forall CTL^*$, a subset of full CTL with only universal path quantification, and restricted temporal operators. A class of mappings are identified that define exact abstractions for full CTL^*. A similar, but slightly more general approach to state abstraction is also described in [6]. A related approach based on ω-language containment is presented in [5]

Recently, there has been in increased interest in formalisms for describing the behaviour of hybrid systems, for which the verification task is tractable. Raven *et al.* model hybrid systems using a real-time interval temporal logic [7]. Both specifications and their refinements are given as formulae in the logic. Verification amounts to showing that the refinement implies its specification. The logic is defined over interpretations in which states are viewed as functions over real-time. The technique is particularly aimed at expressing duration properties such as "Within any time period of length T, state S may occur at most c per cent of the time". The logic is inherently undecidable, but some sound deduction rules are given

Alur *et al.* [1] use *hybrid automata* to model system behaviour. The state of a hybrid automata consists of a location counter, drawn from a finite set of locations, and a set of real valued variables. The program counter defines the state of a finite state machine. Associated with each state is a set of differential equations, which govern the behaviour of the real-valued variables while the finite-state machine is in that state. Also associated with each state is a set of exceptions, predicates over the real-valued variables. Progress can be made by such an automaton in two ways. As time passes, the real-valued variables change their value according to the active set of differential equations. At any time, the entire state of the machine may change instantly according to a transition relation over both the location counter and the real-valued variables. To ensure progress, the automaton must make such an instantaneous transition before the elapse of sufficient time to satisfy one of the exception predicates.

The authors give a semi-decision procedure for proving that members of a restricted class of hybrid automata satisfy linear invariants over the real-valued variables. The procedures, based on computing and minimising fixed points, are guaranteed to give correct results if they terminate. Several examples are given for which the procedures do indeed terminate.

A completely different approach to hybrid system specification [9] and verification [10] is given by Zhang and Mackworth. They use a formalism called constraint-nets to represent hybrid systems. Essentially, a constraint net represents the evolution of a system state as a set of mappings from algebraically defined time-structures to variable domains, which must have certain algebraic properties. These mappings represent the shared inputs and outputs of a set of transductions, and must be causally related with respect to the time structures. Semantically, a constraint net is denoted by the least fixed point of a set of equations. The existence of such a fixed-point is guaranteed by the algebraic properties of the time-structures and variable domains.

The approach that we advocate in this paper is to use hybrid models to model the system under examination initially. We then construct a discrete conservative approximation of this model for the purpose of verification. To illustrate this principle we have purposely chosen extremely simple frameworks with which to model our examples, and express our specifications.

3 System Domains

Mathematical models, have long been used to describe the behaviour of physical systems. Such models are useful, in as much as questions about the systems that they represent, can be rephrased as questions about the models. For example, the position of a train, traveling at a constant velocity of five miles per hour, can be modeled by the following linear equation.

$$x = 5t$$

This equation is a model of the train in two, related, respects. In one sense, the structure of the equation is intended to represent the structure of train behaviour. The 5 in the equation represents the speed of the train. The t in the equation represents the passage of time. The x represents the trains relative position. More fundamentally, however, certain questions about the train can be phrased in terms of questions about the equation. For example, the question "how many miles will the train cover in two hours," can be rephrased as "what is the solution to $x = 5t$ when 2 is substituted for t.

It is this latter property that makes the equation a useful model, for it enables us to rephrase questions about the train in terms of questions about the equation. While the structural relationship between the equation and the train might seem important, it is actually the relationship between questions that is essential, for without such a relationship the model would be useless. Indeed, the questions that one can ask of a model are, in a sense, its *raison d'être*. In recognition of this, we view a modeling framework as the combination of a set of models, with *a set of such questions*.

Since we are primarily motivated by the goal of formal verification, we confine our attention to decision questions. We view each such predicate as a formal specification that is satisfied by the models for which it holds. Thus the question "is $x = 10$ a solution when $t = 2$?" would be a legal question to ask of our train model, $x = 5t$. It would also constitute a specification, which this model does indeed satisfy.

More formally we view a *system domain* as a family (or language) of questions, together with a set of models. Each model provides a context in which the questions can be answered. Thus a system domain, γ, may be represented as a pair, (M, Q), in which M is a set of models, and Q is a set of predicates over M. We view the predicates in Q as *specifications*, saying that a model $m \in M$ *satisfies* the specification $q \in Q$ precisely when $q(m)$ holds.

3.1 Discrete Safety Automata

One of the main motivations for this work, is the need to verify that hybrid systems, such as our train-set, maintain certain safety invariants, such as remaining free from collisions. The system domain *discrete safety automata* is a simple modeling and specification framework in which such properties can be easily expressed. The models of discrete safety automata are (non-deterministic) finite state automata that are parameterised by their initial and final (accepting) state-sets. The automaton m may be asked questions (satisfies specifications) of the following form: "for initial state-set I and final state-set F, does $m(I, F)$ *reject* the trace w". Note the somewhat unconventional parity of the questions. It may seem more natural to ask which strings an automaton accepts, rather than which strings it rejects. The justification for our choice, which will be developed more formally in later sections, is that the final (accepting) state-set represents hazards to be avoided.

A safe system avoids these hazards, while its model avoids the corresponding accepting states.

Before proceeding further, let us define this notion of automata more precisely. The definition here is essentially the same as that given by Wood in [8]. If Z is any finite set, we denote the set of finite sequences over Z, including the empty sequence, ϵ, by Z^*. We denote sequence concatenation by juxtaposition. We use angle braces, $\langle\rangle$, to delimit the values of a sequence, Thus $\langle a, b, c, d\rangle$ denotes the of length 4, whose first element is a, and whose last element is d.

A lazy nondeterministic automaton, m, is a triple, (Σ, S, Δ), where Σ, the *input-set* can be any finite set, S is a (finite) set of states, and Δ is a finite state-transition relation.

$$\Delta \subseteq \Sigma^* \times S \times S$$

Notice that, somewhat non-traditionally, the state transition relation relates finite *sequences* of input symbols, rather than merely individuals, to pairs of states. Thus, the automaton may be able to read more than one input symbol in a single transition, inspiring the name *lazy* automata.

Given a set of initial states, I, and a set of final states, F, we say that the sequence w is in $L(m, I, F)$—the *language accepted* by m—if and only if there is a natural number z, a sequence of traces $w_1, w_2, ..., w_z$, and a sequence of states $s_0, s_1, ..., s_z$ such that w is the concatenation $w_1 w_2 \cdots w_z$, s_0 is in I, s_z is in F, and (w_k, s_{k-1}, s_k) is in Δ for each k in the set $\{1, 2, ..., z\}$.

We have chosen to base our models on "lazy" automata that can consume multiple elements from the input-set in a single step instead of the more conventional variety that accept symbols one at at time. This choice was made largely to simplify the generalisation of this formalism in Section 3.2 to continuous systems. It does, however, have a mildly unfortunate consequence; the role played by the state of the automaton, may differ from the role one expects state to play in a physical system.

One expects the "state" of a real system to capture all of the information about the history of the system, that could help one predict its future behaviour. Naturally, we would be pleased if this intuition was reflected in the model. For this reason, one would like the model to be capable of accepting symbols one at a time, keeping track of all the historical data relevant to subsequent decisions in its state. Clearly, lazy automata need not behave in this way. For example, the transition relation Δ may include the transition (ab, x, y) and yet have no transitions from x on the sequence a. To decide whether to consume the symbol a, the automaton must "know" whether the symbol b is forthcoming.

We introduce the term "eager" to describe automata that do not behave in this counter-intuitive way. Formally, the automaton $m = (\Sigma, S, \Delta)$ is *eager* if and only if the following condition holds for all strings u and v in Σ^* and for all states a and b in S.

$$(uv, a, b) \in \Delta \Longrightarrow \exists c \in S \cdot (u, a, c) \in \Delta \ \wedge \ (v, c, b) \in \Delta$$

As a result, for any step in which the eager automaton consumes several symbols, there must be a corresponding sequence of steps, each accepting only one symbol, that ultimately consumes the same sequence. Thus, an eager automaton is capable of accepting its input symbols one by one, storing enough information in its state to enable future decisions.

We can now define the system-domain *discrete safety automata* as the pair (M, Q) in which the set of models, M, is the set of eager, non-deterministic finite state automata, defined above. A predicate $q \in Q$ is a triple, (w, I, F), where I and F are finite sets of

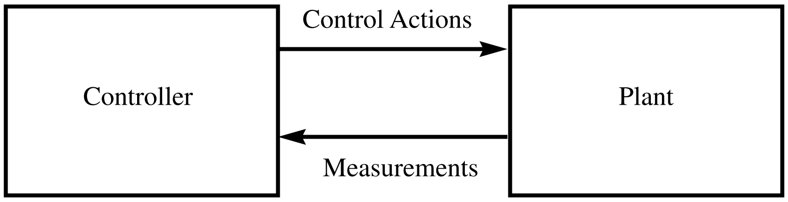

Figure 1: Measurements are generated by the plant.

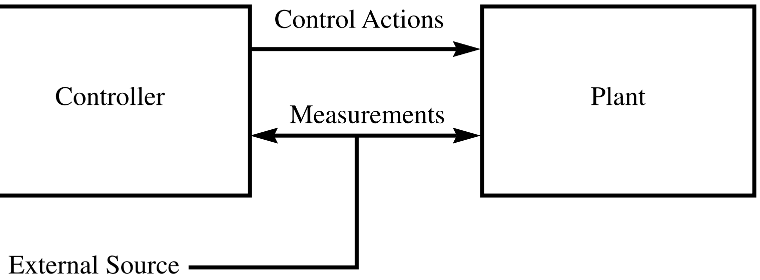

Figure 2: Measurements are provided by an external source.

states, and w is a trace over some input-set, $\widehat{\Sigma}$. We say that q holds for $m = (\Sigma, S, \Delta)$ if and only if $I \subseteq S$, $F \subseteq S$, $w \in \Sigma^*$, and $w \notin L(m, I, F)$.

The theory that we are developing is motivated by the problem of controller verification. We are interested in modeling the behaviour of plants from the point of view of their controllers. Figure 1 shows the configuration that we have in mind. The controller and the plant are two separate entities. Communication takes place in two directions. Measurements are sent from the plant to the controller, while control actions are sent from the controller to the plant. Thus, from the point of view of the controller, the plant appears to be a box, with an input channel, on which it receives control signals, and an output channel on which it produces measurements. Rather than directly modeling the output from a non-deterministic automata, we represent the measurement signal as an input to the system, common to both the controller and the plant. This new configuration is represented graphically in Figure 2. Whereas before, the plant simply did not produce impossible measurements, the plant must now fail to accept such signals. This can be accomplished in practise by restricting state transitions to those whose input sequences correspond to physical possibilities.

By way of example, we consider modeling the following simple plant, with an eye towards verifying its controller. The plant consists of a water storage tank, to which an inflow pipe is connected. Water can flow into the tank via the pipe. We are to design a controller that will fill, but not overflow, the tank. The tank is equipped with a sensor with which the controller can measure the height of the water.

Throughout our discussion of the problem, we simplify the exposition by choosing suitable units of measure, so that continuous references to scalar constants are avoided. Thus we assume that an inflow of one, maintained for one unit of time will cause the

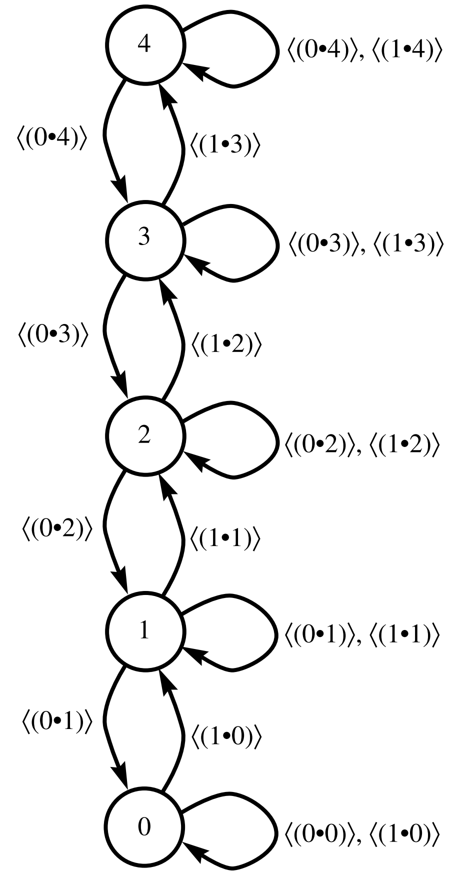

Figure 3: A simple discrete automaton

height of the water in the tank to increase by one unit.

In our simple discrete plant model, we represent the height of the water, by the automata state, drawn from the integers between 0 and 4. Thus the state-set, S, is given by the following expression.

$$S = \{0, 1, 2, 3, 4\}$$

We represent the control input and the measurement output by components of a discrete vector-valued input-set. Let $I = \{0, 1\}$ be the set of control inputs, and $O = \{0, 1, 2, 3, 4\}$ be the set of possible measurements. The input-set Σ is their cross-product.

$$\Sigma = I \times O$$

We define the state transition relation as follows.

$$\Delta(\langle i, o\rangle, h, h') \stackrel{\text{def}}{\equiv} (h = o) \;\wedge\; (h + i - 1) \leq h' \leq h + i)$$

To simplify the presentation of safety automata, we have adopted a standard format with which to describe them. Table 1 is an example. The first box defines the input-set, Σ. The second defines the state-set S. The remaining box defines the state transition

Table 1: Simple water tank model

$\Sigma = \{(cin \cdot mout) \mid cin \in \{0,1\} \;\wedge\; mout \in \{0,1,2\}\}$			
$S = \{0,1,2,3,4\}$			
$\Delta(\langle(cin \cdot mout)\rangle, s, s') \equiv$			
$s \in$	$cin \in$	$mout =$	$s' =$
$\{0,1,2,3\}$	$\{1\}$	s	$s+1$
$\{0,1,2,3,4\}$	$\{0,1\}$	s	s
$\{1,2,3,4\}$	$\{0\}$	s	$s-1$

relation, Δ, which relates an input trace to a starting state and an ending state. The box consists of two parts. The upper part is used to bind symbols representing the input trace, the starting state, and the ending state. In Table 1 for example, the transition relation relates traces consisting of a single symbol. The symbol itself is an ordered pair, the components of which are named *cin* and *mout*. That is, the input trace consumed by a single transition must consist of a single symbol, $(cin \cdot mout)$. The name s is introduced for the starting state, and the name s' is introduced for the ending state. In addition, other symbols may be quantified, and additional restrictions imposed here, although neither are required in this example.

The lower part is divided into columns. The first row of this lower part gives each column a heading. Each remaining row—there are three in Table 1—when taken with the headings, specifies a predicate over the variables introduced in the upper part. For example, the first row of this section in Table 1 specifies that the starting state, s, must be drawn from the set $\{0,1,2,3\}$, *cin* must have the value 1, *mout* must be equal to the starting state, s, and the ending state, s', must be one greater than the starting state. The table is to be interpreted as the conjunction of the upper part of the specification of Δ, with the disjunction of the predicates specified by each row of the lower section. The scope of any quantifications introduced in the upper parts, is understood to extend to the end of the table, unless specifically restricted with parentheses. For this simple machine, the graphic representation of Figure 3 is much easier to understand. However, as the machines become more complicated, the value of this tabular representation will become clear.

As stated previously, we limit our attention to questions of the form "would the plant model fail to accept the input sequence w if it was started in the initial state-set I with final (accepting) state-set F". To help develop the reader's intuition, we consider several concrete examples. Suppose that we wish to avoid filling the tank to the brim, so that any further addition would result in a spill. To model this specification, we let the set of final, or hazardous states be $F = \{4\}$, the set consisting only of the "full" state. We start the system in any non-full state. That is, $I = \{0,1,2,3\}$. We will thus be asking questions questions of the form $w \notin L(M, I, F)$, in which the trace w represents a sequence of actions taken and measurements made by a controller — a possible controller behaviour . Each such question asks whether a controller that behaves in this way could allow the plant to enter an unsafe state.

Consider the answer to this question for the following traces.

1. $w = \langle(1 \cdot 2), (1 \cdot 2)\rangle$

2. $w = \langle(1{\cdot}2), (1{\cdot}3)\rangle$

3. $w = \langle(0{\cdot}2), (1{\cdot}3)\rangle$

4. $w = \langle(1{\cdot}2), (1{\cdot}3), (0{\cdot}3)\rangle$

In the first sequence, $\langle(1{\cdot}2), (1{\cdot}2)\rangle$, the controller measures the water height as 2, and starts the water flowing in at rate 1. This results in a reachable state-set of $\{2, 3\}$. The controller then measures the water height as 2 and sets the input flow rate to 1. Once again, this results in a reachable state-set of $\{2, 3\}$. Since this does not intersect the final (accepting) state-set, $\{4\}$, we conclude that this sequence of events is "safe".

The second sequence, $\langle(1{\cdot}2), (1{\cdot}3)\rangle$, consists of the same control actions as the first. A different sequence of measurements, however, allows the final (accepting) state 4 to be reached. Thus, according to the model, a controller that behaves this way could permit the tank to overflow.

The third sequence, $\langle(0{\cdot}2), (1{\cdot}3)\rangle$, is safe by virtue of being unachievable. After the first measurement and control action, the model cannot be in state 3. Hence, the second measurement, 3, is not possible. A controller that would set the input flow rate to 1 in this circumstance would do no harm, since the circumstance can never arrise.

The final sequence is technically "safe" even though, as we have already shown, it has an "unsafe" prefix. To verify the safety of a controller with respect to this particular string, one must check, not only the given sequence, but also all of its prefixes.

3.2 Continuous Safety Automata

The system domain *continuous safety automata* is a generalisation of its discrete counterpart, discrete safety automata. The models, *continuous automata*, retain much of the notation of finite state automata, but have a continuous notion of time, and permit infinite, continuous state-spaces and input-sets. They are neither intended to be executable, nor to be feasible to simulate. The purpose of describing a continuous automata is not to simulate its execution but rather to demonstrate that the "languages" that it can accept have certain properties.

Fundamental to the earlier presentation of automata, is the notion of a sequence as a mathematical structure. We begin by defining sequences in terms of partial functions from subsets of the natural numbers to some finite sets. We then generalise this definition to arrive at *real-traces*, which are partial functions from real-intervals to infinite sets.

To define sequences so as to permit the generalisation that we have in mind, it will help to introduce a notation for partially open subsets. We denote the set of naturals less than n, but greater than or equal to m by $\mathcal{N}[m, n)$.

$$\mathcal{N}[m, n) \stackrel{\text{def}}{=} \{i \in \mathcal{N} \mid m \leq i < n\}$$

Similarly, we denote the left closed interval of the reals less than n, but greater than or equal to m by $\mathcal{R}[m, n)$

Let Σ be any set. We view sequences over Σ as partial functions from the natural numbers, $\mathcal{N}$, to Σ. For example, if $\Sigma = \{a, b, c, d\}$, the sequence $\langle acbd\rangle$ maps the number 0 to a, 1 to c, 2 to b and 3 to d.

More precisely, a *sequence* σ of n symbols from Σ is a function from the naturals less than n to Σ.

$$\sigma : \mathcal{N}[0, n) \mapsto \Sigma$$

We denote the set of such sequences of length n by $\Sigma^n[\mathcal{N}]$.

$$\Sigma^n[\mathcal{N}] \stackrel{\text{def}}{=} \{\sigma : \mathcal{N}[0,n) \mapsto \Sigma\}$$

So far, we have used the term "length" somewhat loosely. We now give it a precise definition. If the sequence σ is drawn from $\Sigma^k[\mathcal{N}]$ then we say that its length, denoted $|\sigma|$ is k. We use ϵ to denote the single element of $\Sigma^0[\mathcal{N}]$. We use the notation, $\Sigma^*[\mathcal{N}]$, to refer to the set of all finite length sequences over Σ.

$$\Sigma^*[\mathcal{N}] \stackrel{\text{def}}{=} \bigcup_{n \in \mathcal{N}} \Sigma^n[\mathcal{N}]$$

Concatenation of sequences, is denoted by juxtaposition. If $v \in \Sigma^a[\mathcal{N}]$, and $w \in \Sigma^b[\mathcal{N}]$ then their concatenation, $vw \in \Sigma^{a+b}[\mathcal{N}]$ is the following partial function, illustrated here using lambda notation to bind the formal parameter.

$$vw \stackrel{\text{def}}{=} \lambda x \cdot \begin{cases} v(x) & \text{if } x \in \mathcal{N}[0,a) \\ w(x-a) & \text{if } x \in \mathcal{N}[a,a+b) \end{cases}$$

Having defined *sequences* in this way, it is quite natural to extend the definition by replacing the natural numbers, $\mathcal{N}$, with the reals, $\mathcal{R}$. Once again, let Σ be any set. We define $\Sigma^0[\mathcal{R}]$ as the singleton set $\{\epsilon\}$. The set of *real traces* of length i, $\Sigma^i[\mathcal{R}]$, is defined as follows for all positive reals, i.

$$\Sigma^i[\mathcal{R}] \stackrel{\text{def}}{=} \{\sigma : \mathcal{R}[0,i) \mapsto \Sigma\}$$

Finally, we define $\Sigma^*[\mathcal{R}]$, the set of all real-traces of Σ, where $\mathcal{R}^+$ denotes the non-negative real numbers.

$$\Sigma^*[\mathcal{R}] \stackrel{\text{def}}{=} \bigcup_{i \in \mathcal{R}^+} \Sigma^i[\mathcal{R}]$$

In the interest of practicality, we rule out certain poorly behaved functions from the set of traces $\Sigma^*[\mathcal{R}]$. By definition, every trace $w \in \Sigma^*[\mathcal{R}]$ has a finite length. Moreover, for any set Σ, we restrict $\Sigma^*[\mathcal{R}]$ so that every trace $w \in \Sigma^*[\mathcal{R}]$ is infinitely differentiable at all but a finite set of points. That is, it must have at most a finite number of discontinuities. Furthermore, these discontinuities must partition the trace into a set of intervals that are closed from the left. That is, if a trace w has a discontinuity at time t, and $t < |w|$ there must be a time t^+ such that $t < t^+ < |w|$, and w is continuous on the closed interval $[t, t^+]$. Additionally we restrict $\Sigma^*[\mathcal{R}]$ so that each trace $w(t) \in \Sigma^*[\mathcal{R}]$ approaches a well-defined limit at $t = |w|$. If $l = |w|$, we use the notation $w(l)$ to denote this limit. Moreover, we require that this limit exist for every i^{th} derivative, and use the notation $w^i(l)$ to denote it.

From time to time, we will need to refer to the integral of a real or integer valued trace. We use the notation $\int w$ as a short-hand for the limit $\lim_{x \to |w|} \int_0^x w(t)dt$. The existence of this limit is assured since the limit $w(|w|)$ exists, and the trace can have only a finite number of discontinuities.

To complete the notation, we introduce a notion of parallel composition. The notion applies both to pairs of equal-length sequences, and to pairs of equal-length traces. Let D stand for either of the domains $\mathcal{N}$ or $\mathcal{R}$ If $w_1 \in \Sigma_1^k[D]$ and $w_2 \in \Sigma_2^k[D]$ then $(w_1 \,\|\, w_2) \in (\Sigma_1 \times \Sigma_2)^k[D]$ is the trace, $\lambda t : D[0,t) \cdot (w_1(t), w_2(t))$. We explicitly define $\epsilon \,\|\, \epsilon = \epsilon$. The following technical lemma, which follows directly from the definitions, of concatenation and parallel composition, will required later in the paper.

Lemma 3.1 *For all u, v, w_1, and w_2, $w_1 w_2 = u \| v$ if and only if there exist traces u_1, u_2, v_1 and v_2 such that $u = u_1 u_2$, $v = v_1 v_2$, $w_1 = u_1 \| v_1$ and $w_2 = u_2 \| v_2$.*

Proof: Two traces (partial functions) are equal, if they have the same domain and they define the same mapping. We prove the "if" part first. Let u, v, w_1, w_2, u_1, u_2, v_1, and v_2 be any traces such that $u = u_1 u_2$, $v = v_1 v_2$, $w_1 = u_1 \| v_1$ and $w_2 = u_2 \| v_2$. Let $l_1 = |w_1|$ and $l_2 = |w_2|$. Since $w_1 = u_1 \| v_1$ and $w_2 = u_2 \| v_2$, we must have $|u_1| = |v_1| = l_1$ and $|u_2| = |v_2| = l_2$. Clearly $|w_1 w_2| = |u \| v| = l_1 + l_2$. Let t be any value in $[0, l_1 + l_2)$. either $(t < l_1)$ or $t \geq l_1$. Suppose $t < l_1$. Then the following equalities hold.

$$\begin{aligned}
&(w_1 w_2)(t) \\
&\quad = ((u_1 \| v_1)(u_2 \| v_2))(t) \\
&\quad = (u_1 \| v_1)(t) \\
&\quad = (u_1(t), v_1(t)) \\
&\quad = ((u_1 u_2)(t) \cdot (v_1 v_2)(t)) \\
&\quad = ((u_1 u_2) \| (v_1 v_2))(t) \\
&\quad = (u \| v)(t)
\end{aligned}$$

On the other hand, if $t \geq l_1$ then

$$\begin{aligned}
&(w_1 w_2)(t) \\
&\quad = ((u_1 \| v_1)(u_2 \| v_2))(t) \\
&\quad = (u_2 \| v_2)(t - l_1) \\
&\quad = (u_2(t - l_1) \cdot v_2(t - l_1)) \\
&\quad = ((u_1 u_2)(t) \cdot (v_1 v_2)(t)) \\
&\quad = ((u_1 u_2) \| (v_1 v_2))(t) \\
&\quad = (u \| v)(t)
\end{aligned}$$

To prove the "only if" part, let w_1, w_2, u and v be arbitrary traces such that $w_1 w_2 = u \| v$. Once again, let $l_1 = |w_1|$ and let $l_2 = |w_2|$. Let $u_1 = \lambda t \in [0, l_1) \cdot u(t)$, $u_2 = \lambda t \in [l_1, l_1 + l_2) \cdot u(t + l_1)$, $v_1 = \lambda t \in [0, l_1) \cdot v(t)$, $v_2 = \lambda t \in [l_1, l_1 + l_2) \cdot v(t + l_1)$. Clearly $u = u_1 u_2$, $v = v_1 v_2$, $w_1 = u_1 \| v_1$ and $w_2 = u_2 \| v_2$. $\square$

As with finite state automata, a *continuous automata*, m, is a triple

$$m = (\Sigma, S, \Delta)$$

consisting of an input-set, Σ, a set of states, S, and a transition relation, Δ. This time, however, we allow Σ and Δ to be infinite. Whereas the transition relation for a finite state automata, relates pairs of states to sequences, the transition relation Δ relates pairs of states to *real-traces* over the input-set.

$$\Delta \subseteq \Sigma^*[\mathcal{R}] \times S \times S$$

The language accepted by a continuous automaton is defined in exactly the same way as it was for finite-state automata. Given a set of initial states, I, and a set of final states, F, we say that a real-trace w is in $L(m, I, F)$—the *language accepted* by m—if and only

Table 2: Continuous Integrator from s_{min} to s_{max}

$$M_I(s_{min}, s_{max}, i_{max}) = (\Sigma_I, S_I, \Delta_I)$$

$\Sigma_I = \{(i \cdot o) \mid i \in \mathcal{R}[-i_{max}, i_{max}] \ \wedge \ o \in \mathcal{R}[s_{min}, s_{max}]\}$
$S_I = \mathcal{R}[s_{min}, s_{max}]$
$\Delta_I((i \| o), s, \hat{s}) \stackrel{\text{def}}{\equiv}$ $\exists l \cdot l = \lvert (i \| o) \rvert \wedge$ $s' = s + \int_0^l i(x)dx \wedge$ $\forall t \in \mathcal{R}[0, l] \cdot$ $s + \int_0^t i(x)dx \in S \wedge$ $o(t) = s + \int_0^t i(x)dx$

if there is a natural number z, a sequence of traces $w_1, w_2, ..., w_z$, and sequence of states $s_0, s_1, ..., s_z$ such that the trace in question, w, is the concatenation $w_1 w_2 \cdots w_z$, $s_0 \in I$, $s_z \in F$, and $(w_k, s_{k-1}, s_k) \in \Delta$ for each $k \in \{1, 2, ..., z\}$.

As with their discrete counter-parts, the continuous automata need not necessarily reflect our intuition regarding their use of state. Here, of course, we cannot hope to have them consume their inputs one symbol at at time, since inputs are presented as a continuous real-trace. One can ask, however, that at all times, one need only consult the automaton's state, and never prior or subsequent input to predict its behaviour. The following definition of "eager" assures this intuitive behaviour, and differs from the definition presented in Section 3.1 only by the substitution of $\Sigma^*[\mathcal{R}]$ for $\Sigma^*[\mathcal{N}]$. A continuous automata $m = (\Sigma, S, \Delta)$ is *eager* if and only if the following condition holds for all real-traces u and v in $\Sigma^*[\mathcal{R}]$ and all states a and a'.

$$(uv, a, a') \in \Delta \Longrightarrow \exists a'' \in S \cdot (u, a, a'') \in \Delta \ \wedge \ (u, a'', a') \in \Delta$$

In addition, we impose the following restrictions on state transition relations, Δ. For every state $s \in S$, Δ must include the transition (ϵ, s, s). Furthermore, Δ may not include any transition (ϵ, s, t) where $s \neq t$. Informally, these restrictions say that if no time passes, every automaton can and must do nothing.

The system-domain, *continuous safety automata* consists of a set of models, M, and a set of predicates, Q. The models are eager continuous automata described above. A predicate $q \in Q$ consists of a triple (I, F, W) where I and F are state-sets and W is a real-trace. We say that q holds for automaton $m = (\Sigma, S, \Delta)$ if and only if $I \subseteq S$, $F \subseteq S$, $w \in \Sigma^*[\mathcal{R}]$ and $w \notin L(m, I, F)$.

To illustrate the idea of continuous safety automata we use a continuous automaton to represent an integrator. Integration is an important component of many physical systems. For example, one can view the height of the water in the water-tank from Section 3.1 as the integration over time of the rate at which water flows in.

The family of continuous automata, $M_I(i_{max}, s_{min}, s_{max})$, is summarised in Table 2. The parameter $i_{max} \in \mathcal{R}$ is an upper bound on the magnitude of the integrand. The parameters $s_{min} \in \mathcal{R}$ and $s_{max} \in \mathcal{R}$ respectively represent minimum and maximum values of the integral. Although continuous automaton do not differentiate between notions of "input" and "output", it is nontheless convenient to view the integrand as "input", and the integral as "output", hence our choice of the symbols i and o to represent these values

Table 3: Actuator model

$$M_A = (\Sigma_A, S_A, \Delta_A)$$

$\Sigma_A = \{(cin \cdot wout) \mid cin \in \{L, H\} \ \wedge \ wout \in \mathcal{R}[0,1]\}$					
$S_A = \{(s \cdot c) \mid s \in \{L, H\} \ \wedge \ c \in \mathcal{R}[0, 0.5]\}$					
$\Delta_A(cin \,\|\, wout, (s \cdot c), (s' \cdot c')) \equiv$ $\exists l \in \mathcal{R} \cdot l = \|(cin \,\|\, wout)\|$ $\forall t \in \mathcal{R}[0, l] \cdot$					
$s =$	$c \in$	$cin(t) =$	$wout(t) \in$	$s' =$	$c' =$
L	$[0,0]$	L	$[0,0]$	L	0
L	$[0,0.5]$	H	$[0,1]$	H	$0.5 - l$
L	$[l,0.5]$	L	$[0,1]$	L	$c - l$
H	$[0,0]$	H	$[1,1]$	H	0
H	$[0,0.5]$	L	$[0,1]$	L	$0.5 - l$
H	$[l,0.5]$	H	$[0,1]$	H	$c - l$

in the table. The automaton can go from some state $s \in S_I$ to state $s' \in S_I$ by consuming the trace $i \,\|\, o$ if and only if the trace o is the integration of i with respect to time, the value $o(0)$ equals s, and the limit $o(l)$ equals s', where l is the length of the input trace. Moreover, at all times, the input i must remain in the range $[-i_{max}, i_{max}]$, and the output o must remain in the range $[s_{min}, s_{max}]$. Notice that M_I has the unusual property that a trace w is in $L(M_I, \{i\}, \{f\})$ if and only if $(w, i, f) \in \Delta_I$. That is, if the automaton can consume an an input trace w, it can do so in one step.

Continuous automata use sequences of discrete transitions between states to model continuous processes. Informally, one can view the states as being instantaneous, whereas time passes during the transitions. Of course if the automaton is eager, as all the automata discussed in this paper will be, then the transitions may be arbitrarily short. Nevertheless, except for transitions taken on the empty-trace, ϵ, they each take a finite, non-zero amount of time.

This seems natural enough when modeling continuous physical systems. After all, if the state of a system is supposed to change continuously, the system may past through any given state instantaneously. Moreover, the very nature of continuity denys the instantaneous passage from one distinct state to another. Remarkably, however, the same formalism, continuous safety automata, can also model systems that respond instantly to discrete events. We illustrate this phenomenon by considering the valve mechanism that controls the water flowing into the water-tank.

The valve mechanism, or actuator, reads a discrete, digital signal, which can take on either the value L or the value H, from the controller. When the signal changes to H the actuator opens the source valve. When the signal changes to L, the actuator closes the source valve. Of course there will be some delay, between the time at which the control input changes, and the time at which the actuator has fully opened or closed the valve. Suppose, however, that we have observed that, regardless of its starting configuration, the actuator is always able to open or close the valve fully in response to an input event within 0.5 time units.

Our model will be the continuous automaton summarised in Table 3. Its input-set will consist of two components. The first, cin, drawn from the set $\{L, H\}$ will represent

Table 4: Controller model

$$M_C = (\Sigma_C, S_C, \Delta_C)$$

$$\Sigma_C = \{(sin \cdot cout) \mid sin \in \mathcal{R}[0,4] \ \wedge \ cout \in \{L, H\}\}$$

$$S_C = \{(s \cdot c) \mid s \in \{L, H\} \ \wedge \ c \in \mathcal{R}[0,1]\}$$

$$\begin{aligned} &\Delta_C(sin \,\|\, cout, (s \cdot c), (s' \cdot c')) \equiv \\ &\qquad \exists l \in \mathcal{R} \cdot l = |(min \,\|\, cout)| \\ &\qquad \forall t \in \mathcal{R}[0, l) \cdot \end{aligned}$$

$s =$	$c \in$	$min(0) =$	$cout(t) =$	$s' =$	$c' =$
$\{L, H\}$	$[0,0]$	$(2,4]$	L	L	$1-l$
$\{L, H\}$	$[0,0]$	$[0,2]$	H	H	$1-l$
$\{L, H\}$	$[l,1]$	$[0,4]$	s	s	$c-l$

the discrete control input signal. The second, $wout$, drawn from the interval $\mathcal{R}[0,1]$ will represent the flow of water. The model's state will have two components. The first, s, drawn from $\{L, H\}$ will behave like a flip-flop. We will use it to store the most recent control input event. The second, c, drawn from $\mathcal{R}[0, 0.5]$, will act as a real-valued counter to model the introduced bounded delays.

Informally, we view the first component, cin, of the input-set as the automaton's "input", and the second, $wout$, as its "output". With this interpretation its behaviour can be understood as follows. During the first 0.5 time units, after an event has occurred on the input cin, the automaton's output can randomly fluctuate between 0 and 1. However, after the control input cin has been stable at L or H for 0.5 time units, the output must become stable at 0 or 1 respectively.

To see that the automaton truly is event driven, consider its behaviour when the control signal goes from L to H. According to Table 3, the automaton must be in some state (L, c_0) just prior to the input event, since it will have consumed some trace $cin_1 \,\|\, wout_1$, in which cin_1 has the constant value L. After consuming any arbitrarily short prefix of the remaining input, however, the automaton must be in some state (H, c_1). That is, an arbitrarily short time after the "input", event, we can show that the automaton has made a transition from from some state (L, c_0) to some state (H, c_1).

Ultimately, the plant will be controlled by a controller. The controller is equipped with a sensor input, and a digital output. At any time the sensor input can take on any real value between 0 and 4. The output can take on the discrete values, L, and H. At regular time intervals of length 1, the controller samples the value on its input, and adjusts the value on its output. We view these two actions as occurring simultaneously at the beginning of each time unit. If the value sensed is greater than 2, the output is set to L for the remainder of the time interval. if the value sensed is less than or equal to 2, the output is set to H. This controller is described formally in Table 4. As with the actuator, we used a real-valued component of the controller state to keep track of the passage of time.

In summary, we have seen two kinds of continuous automata. The water-tank plant is modeled by an automaton with continuous state and input sets. Its behaviour is essentially continuous. The state-spaces of the actuator and the controller are continuous in one dimension, and discrete in the other. They both have discrete input-sets, but a continuous notion of time. Their behaviours are driven by discrete events.

3.3 Composition of automata

Ultimately, we would like to represent the behaviour of real physical devices using continuous safety automata. The real devices that we will want to model, will undoubtably have fairly complex behaviours. Representing these behaviours by describing their state-transition relations directly would be a daunting task. What is needed is a way to construct complex automata out of simpler parts. That is we need notions of composition and communication.

The notions of composition presented here are applicable both to discrete and to continuous automata. To avoid duplication, we parameterise the definitions by the "time-domain" D. To instantiate the definitions for finite-state automata, the naturals $\mathcal{N}$ should be substituted for D. To instantiate for continuous automata, the reals $\mathcal{R}$ should be substituted instead.

We separate the issues of composition and communication. We describe a notion of composition, in which composed automata operate in parallel without communication. We then show how communication can be achieved by means of a filter attached to the input of the composite automaton.

When two automata are composed in parallel, they run side by side, with no communication between them. The state-space of the composite automaton is the cross product of the state-spaces of the components. The composite input-set consists of ordered pairs, containing one element from the input-sets of each of the component automata. The composite takes a transition, when both of its component automata would.

More formally, suppose that two automata, $m_1 = (\Sigma_1, S_1, \Delta_1)$ and $m_2 = (\Sigma_2, S_2, \Delta_2)$, are given. We define their parallel composition $m_1 \| m_2 \stackrel{\text{def}}{=} (\Sigma_1 \times \Sigma_2, S_1 \times S_2, \Delta)$, where Δ is defined as follows.

$$(w_1 \| w_2, (s_1, s_2), (s_1', s_2')) \in \Delta \stackrel{\text{def}}{\equiv} (w_1, s_1, s_1') \in \Delta_1 \ \wedge \ (w_2, s_2, s_2') \in \Delta_2$$

The following proposition follows directly from this definition.

Proposition 3.2 *If m and n are eager automata, then so is $m \| n$.*

To illustrate composition, consider the water-tank from Section 3.1. One can view this system as the interaction of two state-holding components. One component is the water in the tank. Its state is the height of the water. As we have previously noted, one can view the water height as the integration over time of the net water flow. Thus we propose to use the integrator model developed earlier as the heart of our model of the tank. The actuator also has state, namely the position of the valve. We can take a step towards modeling the complete water tank plant, by composing an integrator model with the actuator and the controller. The composite automaton, $(M_C \| M_A \| M_I)$ is described in Table 5.

3.4 Communication in composite automata

The reader will recall that safety automata have no notion of output. More precisely, safety automata do not differentiate between inputs and outputs. If a real device is guaranteed to produce an output x, we model this by requiring the presence of x in the automaton's input trace, in order for a transition to occur.

To model communication formally, we introduce the term *transliteration* to refer to a binary relation over input-sets. Transliterations can be used to modify the input-set of

Table 5: Composition of the actuator, and the integrator.

$$M_A \,\|\, M_I = (\Sigma_{A\|I}, S_{A\|I}, \Delta_{A\|I})$$

$$\begin{aligned}\Sigma_{A\|I} = \{&(cin\cdot wout)\cdot(win\cdot sout) \mid \\ &(cin\cdot wout) \in \Sigma_A \ \wedge\ (win\cdot sout) \in \Sigma_I\}\end{aligned}$$

$$S_{A\|I} = \{(a\cdot i)\cdot a \in S_A \ \wedge\ i \in S_I\}$$

$$\begin{aligned}\Delta_{A\|I}(((cin \,\|\, wout) \,\|\, (win \,\|\, sout)), (a\cdot i), (a'\cdot i')) \equiv \\ ((cin \,\|\, wout), a, a') \in \Delta_A \wedge \\ ((win \,\|\, sout), i, i') \in \Delta_I\end{aligned}$$

a safety automaton, and hence to introduce constraints on the traces that it is able to accept.

Suppose that δ is a transliteration between two sets: $\delta \subseteq \Sigma_1 \times \Sigma_2$. We extend δ pointwise to traces of equal length over the two sets. That is if $w_1 \in \Sigma_1^*[D]$ and $w_2 \in \Sigma_2^*[D]$, then $(w_1 \delta w_2)$ is defined as follows.

$$(w_1 \delta w_2) \stackrel{\text{def}}{\equiv} (|w_1| = |w_2|) \ \wedge\ (\forall t \in D[0, |w_1|) \cdot (w_1(t) \delta w_2(t)))$$

We will often need to refer to the pre and post-images of a set with respect to a relation such as δ. We use the notation $\mathrm{pre}(\delta; S)$ and $\mathrm{post}(\delta; S)$ to denote these sets. In other words, if $\delta \subseteq \Sigma_1 \times \Sigma_2$, and if $S \subset \Sigma_2$, then $\mathrm{pre}(\delta; S) \stackrel{\text{def}}{=} \{s_1 \in \Sigma_1 \mid \exists s_2 \in \Sigma_2 \cdot s_1 \delta s_2 \ \wedge\ s_2 \in S\}$. Similarly, if $S \subseteq \Sigma_1$, then $\mathrm{post}(\delta; S) \stackrel{\text{def}}{=} \{s_2 \in \Sigma_2 \mid \exists s_1 \in \Sigma_1 \cdot s_1 \delta s_2 \ \wedge\ s_1 \in S\}$.

The composition, denoted δm, of an automaton $m = (\Sigma, S, \Delta)$ with a transliteration δ is given by $\delta m = (\Sigma', S, \Delta')$ where $\Sigma' = \mathrm{pre}(\delta; \Sigma)$, and $\Delta' \subseteq \Sigma' \times S \times S$ has the following definition.

$$(w', i, f) \in \Delta' \stackrel{\text{def}}{\equiv} \exists w \in \mathrm{post}(\delta; \{w'\}\,) \cdot (w, i, f) \in \Delta$$

In other words, the automata δm takes a transition on the trace w' if and only if there is a trace w upon which m would take a transition, that is related point-wise to w' by the transliteration δ. To model the connection of an "output" of one real device, to the "input" of another, we use an appropriately constructed relation δ, to ensure that the "connected" inputs of their models agree at all times.

The following proposition follows directly from the definitions of composition with a transliteration and eagerness.

Proposition 3.3 *If m is eager automaton, and δ is a transliteration, then δm is an eager automaton.*

Returning to the water-tank example, we use a transliteration to "connect" the "input" of the water-tank integrator, to the "output" of the actuator. Moreover, we "hide" this connection from the outside world, so that all that is visible is the control input, and the sensor output. In Table 6, such a transliteration, δ_P, is defined. Finally, we arrive at the complete water-tank plant model, by composing the transliteration with $M_A \,\|\, M_I$. This composition is illustrated in Figure 4.

$$M_P = \delta_P(M_A \,\|\, M_I)$$

Table 6: Transliteration connecting the actuator output with the integrator input

$\delta_P \subseteq \Sigma' \times \Sigma$
$\Sigma' = \{(cin' \cdot sout') \cdot cin' \in \{L, H\} \ \wedge \ sout' \in \mathcal{R}\}$
$\Sigma = \{(cin \cdot wout) \cdot (win \cdot sout) \cdot$ $cin \in \{L, H\} \ \wedge \ wout \in \mathcal{R} \ \wedge$ $win \in \mathcal{R} \ \wedge \ sout \in \mathcal{R}\}$
$\delta_P((cin' \cdot sout'), ((cin \cdot wout) \cdot (win \cdot sout))) \stackrel{\text{def}}{\equiv}$ $cin = cin' \ \wedge$ $sout = sout' \ \wedge$ $wout = win$

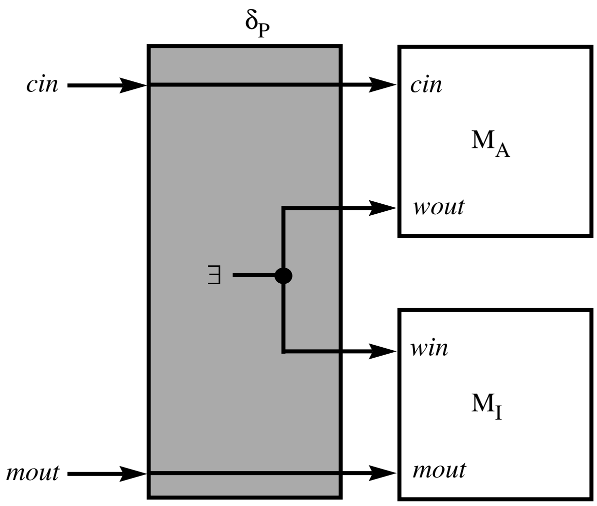

Figure 4: Transliteration connecting M_A with M_I

4 Conservative Approximation

Suppose we are given two system domains, $\gamma = (M, Q)$ and $\widehat{\gamma} = (\widehat{M}, \widehat{Q})$. We wish to compare models in M and $\widehat{M}$ on the basis of the answers each gives to the questions in Q and $\widehat{Q}$ respectively. For example, in earlier sections, we have proposed two models of a water-tank. One model was given in the domain of discrete safety automata, while the other was given in the analogous continuous domain. We would like to be able to claim that any specifications that are satisfied by the discrete model are also satisfied by the continuous one.

Such a claim, however, is clearly false, since the two models being compared do not satisfy specifications from the same language. For example, let the system domain continuous safety automata be (Q, M), and the system domain, discrete safety automata be $(\widehat{Q}, \widehat{M})$. The continuous model, $m \in M$, answers questions $(w, I, F) \in Q$ where w is a real-trace and I and F subsets of a continuous state-space. In contrast, the discrete model, $\widehat{m} \in \widehat{Q}$ answers questions $(\widehat{w}, \widehat{I}, \widehat{F}) \in \widehat{Q}$, where $\widehat{w}$ is a finite sequence, and $\widehat{I}$ and $\widehat{F}$ are subsets of a discrete state-set. Clearly, to make the above claim, we need a way to translate between these two specification languages.

Suppose we are given a relation, $R \subseteq Q \times \widehat{Q}$. We can now define conservative approximation with respect to this translation relation R as follows.

Definition 4.1 *Conservative Approximation: Given two system domains, $\gamma = (M, Q)$ and $\widehat{\gamma} = (\widehat{M}, \widehat{Q})$ and a relation $R \subseteq Q \times \widehat{Q}$ we say that $\widehat{m} \in \widehat{M}$ is a conservative approximation of $m \in M$ with respect to R if and only if the following implication holds for all $q \in Q$ and for all $\widehat{q} \in \widehat{Q}$.*

$$(q, \widehat{q}) \in R \ \wedge \ \widehat{q}(\widehat{m}) \Longrightarrow q(m)$$

If a model $\widehat{m}$ is known to be a conservative approximation of another model m with respect to relation R, and if $\widehat{m}$ is known to satisfy a specification $\widehat{q}$, then every specification q that is related to $\widehat{q}$ by R must be satisfied by m. Thus, if one wished to demonstrate that m satisfied a particular specification q, it would suffice to find a specification $\widehat{q}$ that is both related to q by R, and satisfied by $\widehat{m}$. The following proposition is an obvious consequence of the definition of conservative approximation.

Proposition 4.2 *If $\widehat{m}$ is a conservative approximation of m under R, and $R' \subseteq R$, then $\widehat{m}$ is a conservative approximation of m under R'.*

5 Conservative Approximations Between Automata

We have introduced the notion of conservative approximation with respect to a question translator R. One automaton, $\widehat{m}$, is a conservative approximation of another, m, if and only if m satisfies every specification q that is related by R to a specification $\widehat{q}$ satisfied by $\widehat{m}$. In this section, we present a practical method for constructing such a relation R, for specifications from the system domains, continuous safety automata and discrete safety automata. The method is based on relations between the input traces, and the states of the system domains being compared.

Suppose that we are comparing automata $m = (\Sigma, S, \Delta)$ and $\widehat{m} = (\widehat{\Sigma}, \widehat{S}, \widehat{\Delta})$ drawn from the two system domains, $\gamma = (Q, M)$ and $\widehat{\gamma} = (\widehat{Q}, \widehat{M})$ respectively. We are given relations $\rho \subseteq S \times \widehat{S}$ and $\sigma \subseteq \Sigma^* \times \widehat{\Sigma}^*$. We construct the relation $R(\sigma, \rho)$ between questions for the two automata in the following way. Recall that $\text{post}(\rho; I)$ denotes the post-image of I under ρ.

$$\begin{aligned}&(w, I, F) R(\sigma, \rho)(\widehat{w}, \widehat{I}, \widehat{F}) \stackrel{\text{def}}{\equiv} \\ &\quad w \sigma \widehat{w} \ \wedge \\ &\quad \text{post}(\rho; I) \subseteq \widehat{I} \ \wedge \\ &\quad \text{post}(\rho; F) \subseteq \widehat{F}\end{aligned}$$

That is to say that the question $(\widehat{w}, \widehat{I}, \widehat{F})$ is related to the question (w, I, F) if and only if w and $\widehat{w}$ are related by σ and the set of initial states $\widehat{I}$ (resp. final states $\widehat{F}$) is a superset of the post-image of I (resp. F) under ρ. The following propositions are obvious consequences of the preceding definition, and Proposition 4.2

Proposition 5.1 *If $\widehat{m}$ is a conservative approximation of m under $R(\sigma, \rho)$ and $\sigma' \subseteq \sigma$ then $\widehat{m}$ is a conservative approximation of m under σ'.*

Proposition 5.2 *If $\widehat{m}$ is a conservative approximation of m under $R(\sigma, \rho)$ and $\rho \subseteq \rho'$ then $\widehat{m}$ is a conservative approximation of m under $qmap \sigma \rho'$.*

Before introducing some examples, we prove some lemmas about conservative approximation, as it applies to safety automata. Lemma 5.3 allows us to restrict our attention to singleton initial and final state-sets. Lemma 5.4 exploits the structure of the relation $R(\sigma, \rho)$ to simplify the statement of conservative approximation.

Lemma 5.3 *The trace w is in $L(m, I, F)$ if and only if there exists states $i \in I$ and $f \in F$ such that $w \in L(m, \{i\}, \{f\})$.*

Proof: To prove the "if" part of the lemma, suppose that $i \in I$, $f \in F$, and $w \in L(m, \{i\}, \{f\})$. Then by definition, there exists a natural number z, and sequences $w_1, w_2, \dots, w_z$ and s_0, s_1, s_z such that $w = w_1 w_2 \cdots w_z$, $s_0 = i$, $s_z = f$, and $(w_k, s_{k-1}, s_k) \in \Delta$ for each $k \in \{1, 2, \dots, z\}$. Since $i \in I$, and $f \in F$, the same sequences witness the fact that $w \in L(m, I, F)$.

To prove the "only if" part, we observe that if $w \in L(m, I, F)$, then there must exist a natural number, z, and sequences $w_1, w_2, \dots, w_z$ and s_0, s_1, s_z such that $w = w_1 w_2 \cdots w_z$, $s_0 \in I$, $s_z \in F$, and $(w_k, s_{k-1}, s_k) \in \Delta$ for each $k \in \{1, 2, \dots, z\}$. Letting $i = s_0$ and $f = s_k$ allows the same sequences to witness the fact that $w \in L(m, \{i\}, \{f\})$. □

Lemma 5.4 *The automaton $\widehat{m}$ is a conservative approximation of m under the relation $R(\sigma, \rho)$ if and only if the following predicate holds for all input traces w and $\widehat{w}$ and all states i, f.*

$$(w \sigma \widehat{w} \ \wedge \ w \in L(m, \{i\} \{f\})) \Longrightarrow \exists \widehat{\imath}, \widehat{f} \cdot i \rho \widehat{\imath} \ \wedge \ f \rho \widehat{f} \ \wedge \ \widehat{w} \in L(\widehat{m}, \{\widehat{\imath}\}, \left\{\widehat{f}\right\})$$

Proof: To prove the "if" part, we suppose that $\widehat{m}$ is not a conservative approximation of m, and show the predicate to be false. Since m is not a conservative approximation of $\widehat{m}$, there must exist traces w and $\widehat{w}$, and state-sets I, F, $\widehat{I}$, $\widehat{F}$ such that $(w, I, F) R(\sigma, \rho)(\widehat{w}, \widehat{I}, \widehat{F})$ and $w \in L(m, I, F)$, yet $\widehat{w}$ is not in $L(\widehat{m}, \widehat{I}, \widehat{F})$. From $w \in L(m, I, F)$ and Lemma 5.3 we can conclude that there exists a particular $i \in I$ and a particular $f \in F$ such that $w \in L(m, \{i\}, \{f\})$. Since $(w, I, F)R(\sigma, \rho)(\widehat{w}, \widehat{I}, \widehat{F})$, we conclude that $w\sigma\widehat{w}$, $\{\widehat{\imath} \mid \exists i \in I \cdot i\rho\widehat{\imath}\} \subseteq \widehat{I}$, and $\left\{\widehat{f} \mid \exists f \in F \cdot f\rho\widehat{f}\right\} \subseteq \widehat{F}$.

Recall that we wish to show under these assumptions, that the predicate is false. Suppose with an eye towards contradiction that it is true. We have already established that $w\sigma\widehat{w}$, and that $w \in L(m, \{i\}, \{f\})$. The we can conclude that there exists $\widehat{\imath}$ and $\widehat{f}$ such that $i\rho\widehat{\imath}$, $f\rho\widehat{f}$ and $\widehat{w} \in L(\widehat{m}, \{\widehat{\imath}\}, \left\{\widehat{f}\right\})$. Since $\{\widehat{\imath} \mid \exists i \cdot i\rho\widehat{\imath}\} \subseteq \widehat{I}$ and $\left\{\widehat{f} \mid \exists f \cdot f\rho\widehat{f}\right\} \subseteq \widehat{F}$, Lemma 5.3 allows us to conclude $\widehat{w} \in L(\widehat{m}, \widehat{I}, \widehat{F})$ establishing the sought contraction, and completing the proof.

To prove the "only if" part, we suppose that the predicate is false, and show that $\widehat{m}$ is not a conservative approximation of m. Since the predicate is false, there must exist traces w and $\widehat{w}$ and states i and f such that $w\sigma\widehat{w}$ and $w \in L(m, \{i\}, \{f\})$, yet there are no states $\widehat{\imath}$ and $\widehat{f}$ satisfying $i\rho\widehat{\imath}$, $f\rho\widehat{f}$ and $\widehat{w} \in L(\widehat{m}, \{\widehat{\imath}\}, \left\{\widehat{f}\right\})$.

Suppose, with an eye towards contradiction, that $\widehat{m}$ is a conservative approximation of m under $R(\sigma, \rho)$. Then the following predicate must hold for all I, F, $\widehat{I}$, $\widehat{F}$, w, and $\widehat{w}$.

$$(w, I, F)R(\sigma, \rho)(\widehat{w}, \widehat{I}, \widehat{F}) \ \wedge \ w \in L(m, I, F) \Longrightarrow w \in L(\widehat{m}, \widehat{I}, \widehat{F})$$

In particular let $I = \{i\}$, $F = \{f\}$, $\widehat{I} = \{\widehat{\imath} \mid i\rho\widehat{\imath}\}$ and $\widehat{F} = \left\{\widehat{f} \mid f\rho\widehat{f}\right\}$. Clearly, the question $(w, \{i\}, \{f\})$ is related to $(\widehat{w}, \{\widehat{\imath} \mid i\sigma\widehat{\imath}\}, \left\{\widehat{f} \mid f\sigma\widehat{f}\right\})$ by $R(\sigma, \rho)$. We have already established that $w \in L(m, \{i\}, \{f\})$. Thus, we must conclude that $\widehat{w} \in L(\widehat{m}, \{\widehat{\imath} \mid i\rho\widehat{\imath}\}, \left\{\widehat{f} \mid f\rho\widehat{f}\right\})$. Finally, from Lemma 5.3 we obtain $\exists\widehat{\imath}, \widehat{f} \cdot i\rho\widehat{\imath} \ \wedge \ f\rho\widehat{f} \ \wedge \ \widehat{w} \in L(\widehat{m}, \{\widehat{\imath}\}, \left\{\widehat{f}\right\})$ establishing the required contradiction. □

5.1 A discrete conservative approximation of integration

To illustrate these ideas, we build a conservative approximations of the integrator and actuator developed in Section 3.2. We begin, by establishing some standard discretisations of real-traces. These discretisations will form the basis of the approximations. For example, suppose that $w \in \mathcal{R}^*[\mathcal{R}]$ is a real-trace over the reals. We will approximate w by a sequence $\widehat{w}$ of integers, as illustrated in Figure 5. Each position in the sequence will represent an interval of length 1 in the real-trace. Thus, the first integer, $\widehat{w}(0)$, will summarise the behaviour of the real-trace $w(t)$ over the interval $t \in \mathcal{R}[0, 1)$. The second, $\widehat{w}(1)$, will summarise the interval $t \in \mathcal{R}[1, 2)$. In general, suppose we are given a relation $\widetilde{\sigma}$ between real-traces of length not greater than 1 over some set Σ, and sequences of length one of elements from an approximation $\widehat{\Sigma}$. Informally, the relation $\widetilde{\sigma}$ defines what it means for a single element of $\widehat{\Sigma}$ to summarise a short trace over Σ.

$$\widetilde{\sigma} \subseteq \Sigma^*[\mathcal{R}] \times \widehat{\Sigma}^1[\mathcal{N}]$$

We define an operator Γ which takes such a relation, and extends it over traces of arbitrary length, as illustrated in the figure.

$$\begin{aligned}
&\Gamma(\widetilde{\sigma})(w, \widehat{w}) \equiv \\
&\qquad \exists w_1, ..., w_k \cdot \\
&\qquad \exists \widehat{w}_1, ..., \widehat{w}_k \cdot
\end{aligned}$$

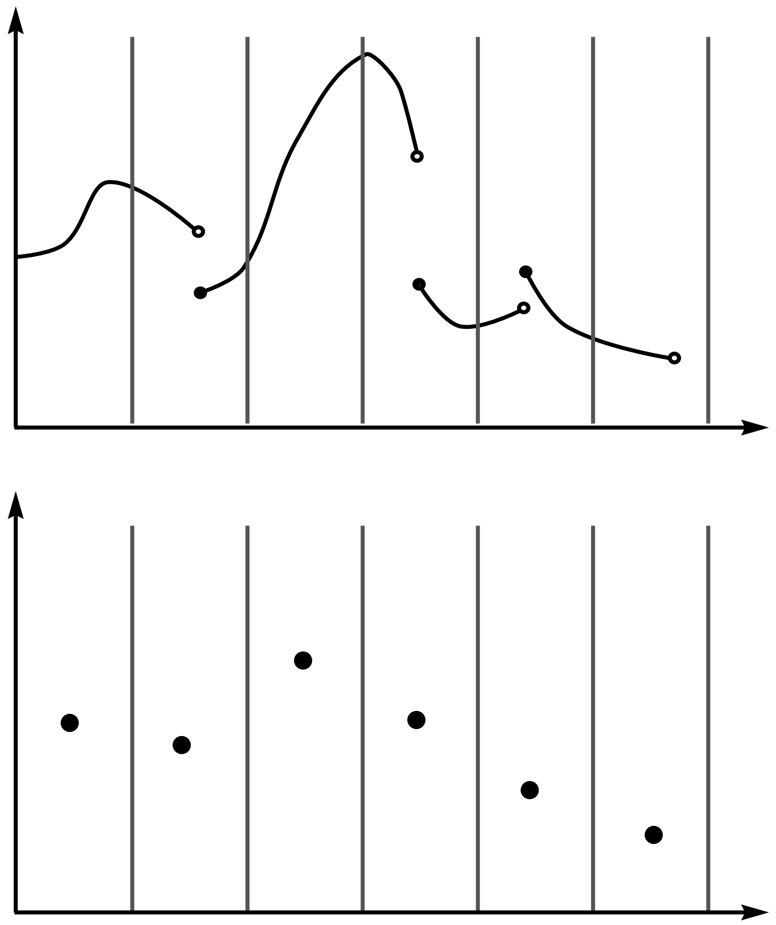

Figure 5: Discretisation of a real-trace

$$
\begin{aligned}
&w = w_1 w_2 \cdots w_k \ \wedge \\
&\widehat{w} = \widehat{w}_1 \widehat{w}_2 \cdots \widehat{w}_k \ \wedge \\
&\forall i \in \{1, ..., k-1\} \cdot |w_i| = 1 \ \wedge \ |\widehat{w}_i| = 1 \ \wedge \\
&|w_k| \leq 1 \ \wedge \ |\widehat{w}_k| = 1 \\
&\forall i \in \{1, .., k\} \cdot (w_i, \widehat{w}_i) \in \widetilde{\sigma}
\end{aligned}
$$

That is, a trace w is related to the sequence $\widehat{w}$ by $\Gamma(\widetilde{\sigma})$ if and only if each can be partitioned in to traces, such that all but the last is of length 1, the last is of length less than or equal to 1, and corresponding partitions are related by $\widetilde{\sigma}$.

For example, consider an input trace $w \in \Sigma_I^*$ to the integrator defined in Table 2. Since elements of Σ_I are ordered pairs, we can view $w = i \| o$ as the parallel combination of two traces, the integrand, i, and the integral, o. We approximate each trace i of length not greater than 1 by the ceiling of its integral. That is, define $\widetilde{\sigma}_i \subseteq \mathcal{R}^*[\mathcal{R}] \times \mathcal{I}^1[\mathcal{N}]$ as follows.

$$
\widetilde{\sigma}_i(i, \widehat{\imath}) \overset{\text{def}}{\equiv} \left(\widehat{\imath} = \left\langle \left\lceil \int i \right\rceil \right\rangle\right) \ \wedge \ |i| \leq 1
$$

Each trace o is to be used as measurement output. In Section 8 we will want to argue that the discretisation actually corresponds to the sampling behaviour of a real digital control system. The control system will take a measurement once during each unit interval of time. However, the precise time within this interval at which the measurement will be taken is unknown. For this reason, we approximate the integral trace as follows.

$$
\widetilde{\sigma}_o(o, \widehat{o}) \overset{\text{def}}{\equiv} \exists t \in \mathcal{R}[0,1) \cdot \widehat{o} = \langle \lceil o(t) \rceil \rangle
$$

That is to say, we relate a trace o to the trace $\widehat{o}$ if o takes on a value $o(t)$ such that $\widehat{o} = \langle \lceil o(t) \rceil \rangle$ at any arbitrary time t in the interval $\mathcal{R}[0,1)$. We use the notation $\widetilde{\sigma}_i \| \widetilde{\sigma}_o$ to represent the product relation.

$$
((i \| o), (\widehat{\imath} \| \widehat{o})) \in (\widetilde{\sigma}_i \| \widetilde{\sigma}_o) \overset{\text{def}}{\equiv} (i, \widehat{\imath}) \in \widetilde{\sigma}_i \ \wedge \ (o, \widehat{o}) \in \widetilde{\sigma}_o
$$

Finally, we arrive at a trace-mapping relation σ_{io}, by using the gamma operator to extend $\widetilde{\sigma}_i \| \widetilde{\sigma}_o$ over traces of arbitrary length.

$$
\sigma_{io} \overset{\text{def}}{=} \Gamma(\widetilde{\sigma}_i \| \widetilde{\sigma}_o)
$$

To build a discrete approximation of the integrator, we also need to define a mapping between the state-space of the integrator, and that of the discrete approximation. For this, we will simply use the ceiling relation. That is each state $s \in \mathcal{R}$ will be related to the state $\lceil s \rceil \in \mathcal{I}$.

$$
\rho_I(s, \widehat{s}) \overset{\text{def}}{\equiv} \widehat{s} = \lceil s \rceil
$$

The complete discrete integrator for input ranging between $-i_{max}$ and i_{max}, and output between s_{min} and s_{max} is presented in Table 7. We claim that for any positive real-valued i_{max}, and any real-valued s_{min}, and s_{max}, the automaton $\widehat{M_I}(i_{max}, s_{min}, s_{max})$ is a conservative approximation of the automaton $M_I(i_{max}, s_{min}, s_{max})$ under the translation $R(\Gamma(\widetilde{\sigma}_i \| \widetilde{\sigma}_o), \rho_I)$. The interested reader may examine the proof in Appendix A.1.

Table 7: Discrete Integrator

$$\widehat{M}_I(s_{min}, s_{max}, i_{max}) = (\widehat{\Sigma}_I, \widehat{S}_I, \widehat{\Delta}_I)$$

$$\begin{aligned} \widehat{\Sigma}_i = \{ & (\widehat{w}_i \cdot \widehat{w}_o) \mid \\ & \widehat{w}_i \in \{\lceil -i_{max} \rceil, \lceil 1 - i_{max} \rceil, ..., \lceil i_{max} \rceil\} \wedge \\ & \widehat{w}_o \in \{\lceil s_{min} \rceil, \lceil s_{min} + 1 \rceil, ..., \lceil s_{max} \rceil\}\} \end{aligned}$$

$$\widehat{S}_I = \{\lceil s_{min} \rceil, \lceil s_{min} + 1 \rceil, ..., \lceil s_{max} \rceil\}$$

$$\begin{aligned} & \widehat{\Delta}_i(\langle (\widehat{w}_i \cdot \widehat{w}_o) \rangle, s, s') \equiv \\ & \qquad (\widehat{w}_i \cdot \widehat{w}_o) \in \widehat{\Sigma} \ \wedge \ s \in \widehat{S} \ \wedge \ s' \in \widehat{S} \ \wedge \end{aligned}$$

$s =$	$\widehat{w}_i =$	$\widehat{w}_o \in$	$s' \in$
any	any	$\left[\frac{s+s'-\lceil i_{max} \rceil}{2}, \frac{s+s'+\lceil i_{max} \rceil}{2}\right]$	$[s + \widehat{w}_i - 1, s + \widehat{w}_i]$

5.2 A discrete conservative approximation of the actuator

In the same way that we approximated the integrator, we give a discrete conservative approximation, $\widehat{M}_A$, for the continuous event-driven model of the actuator, M_A, that was described in Table 3 on page 14.

The input-set, $\widehat{\Sigma}_A$, has two components. The first, drawn from $\{L, X, H\}$ represents the control input. The second, drawn from $\{0, 1\}$ represents the controlled output. As with the integrator, we use a summarising relation to relate continuous input traces from Σ_A^1 to sequences from $\widehat{\Sigma}_A^1$. Let the relation $\widetilde{\sigma}_c \subseteq \{L, H\}^* [\mathcal{R}] \times \{L, X, H\}^1 [\mathcal{N}]$ be defined as follows.

$$\begin{aligned} & \widetilde{\sigma}_c(cin, \widehat{cin}) \stackrel{\text{def}}{\equiv} \\ & \quad \exists l \in \mathcal{R}[0,1] \cdot l = |cin| \ \wedge \\ & \quad \widehat{cin} = \langle L \rangle \ \wedge \ \forall t \in \mathcal{R}[0, l) \cdot cin(t) = L \ \vee \\ & \quad \widehat{cin} = \langle H \rangle \ \wedge \ \forall t \in \mathcal{R}[0, l) \cdot cin(t) = H \ \vee \\ & \quad \widehat{cin} = \langle X \rangle \ \wedge \ \exists t_1, t_2 \in \mathcal{R}[0, l) \cdot cin(t_1) = L \ \wedge \ cin(t_2) = H \end{aligned}$$

That is, a short trace cin is related to the sequence $\langle L \rangle$, if it has the constant value L. It is related to the sequence $\langle H \rangle$, if it has the constant value H. It is related to the sequence $\langle X \rangle$ if it has the value L at one time, and H at another. We use the relation $\widetilde{\sigma}_i$, developed for the integrator input, to relate the continuous and discrete versions of the actuator output.

Recall that states of the continuous model, M_A have two components. The first takes on one of the values H or L according to the current value of the control input. The second is a real number between 0 and 0.5, which represents 0.5 minus the time since the control input last changed, down to a minimum of zero. Thus, if the state is $(L \cdot 0)$, then the valve is fully closed, since the control input has been stable at L for at least 0.5 time units. Similarly, if the state is $(H \cdot 0)$ then the valve is fully open.

The discrete model has three states, named L, H, and X. Informally, we imagine that the discrete state L represents the continuous actuator state $(L \cdot 0)$, in which the valve is definitely completely closed. We imagine that the discrete state H represents the continuous actuator state $(H \cdot 0)$, in which the valve is definitely fully open. Any state—

Table 8: Discrete Actuator

$$\widehat{M}_A = (\widehat{\Sigma}_A, \widehat{S}_A, \widehat{\Delta}_A)$$

$\widehat{\Sigma}_A = \{(i{\cdot}o) \mid i \in \{L, X, H\} \ \wedge \ o \in \{0,1\}\}$

$\widehat{S}_A = \{L, X, H\}$

$\widehat{\Delta}_A(\langle(i{\cdot}o)\rangle, s, s') \stackrel{\text{def}}{\equiv}$

$s =$	$i =$	$o \in$	$s' \in$
L	L	$\{0\}$	$\{L\}$
L	X	$\{0,1\}$	$\{L, X, H\}$
L	H	$\{0,1\}$	$\{H\}$
X	L	$\{0,1\}$	$\{L\}$
X	X	$\{0,1\}$	$\{L, X, H\}$
X	H	$\{0,1\}$	$\{H\}$
H	L	$\{0,1\}$	$\{L\}$
H	X	$\{0,1\}$	$\{L, X, H\}$
H	H	$\{1\}$	$\{H\}$

(L, t) or (H, t) where $t > 0$—in which the valve may be neither fully open nor fully closed is represented in the discrete model by X.

It is not hard to convince oneself that the discrete model does behave as the continuous model does, at least for traces with an integral length. For example, suppose that the continuous automaton starts in state $(H, 0)$, and receives an input trace, $(cin \,\|\, cout)$, of length 1, where cin has the constant value L, and $cout$ has the constant value 0. The trace will be consumed in a minimum of two transitions, leaving the automaton in the state $(L, 0)$. Referring to Table 8, we can see that. if the discrete model is started in state H, it is able to consume the sequence $\langle L{\cdot}0\rangle$, and move to the state L. Indeed, we show in Appendix A.2 that $\widehat{M}_I$ is a conservative approximation of M_I under $\Gamma(\widetilde{\sigma}_c \,\|\, \widetilde{\sigma}_i)$ and this state mapping, provided that $\widetilde{\sigma}_i$ is restricted to traces of length 1.

Consider, however, the same trace, but with its length reduced to $\frac{1}{4}$. Once again, we start the continuous automaton in state $(H{\cdot}0)$. This time, however, because the sequence is short, it can only move to state $(L{\cdot}0.25)$. The sequence has the same translation, however, and so the discrete automaton must be able to move from H to X, by consuming $\langle(L{\cdot}0)\rangle$, which it cannot do. One might be tempted to simply add transitions to the discrete automaton until it works. But if one did, the result would be excessively conservative.

Instead, we simply enlarge the state-mapping relation. In practise, we do not care about the actuator state. To verify that the tank will not overfill, we will only need to ask questions about the height of the water. The position of the valve will be immaterial. Thus, we simply relate every state of the continuous model, to every state in the discrete model.

$$(s, \widehat{s}) \in \rho_A \stackrel{\text{def}}{\equiv} s \in S_A \ \wedge \ \widehat{s} \in \widehat{S}_A$$

Under this relation, we will no longer be able to ask questions about the actuator state. However, as we prove in Appendix A.2, $\widehat{M}_A$ is a conservative approximation of M_A under $R(\Gamma(\widetilde{\sigma}_c \,\|\, \widetilde{\sigma}_i), \rho_A)$.

Table 9: Discrete Controller

$$\widehat{M}_C = (\widehat{\Sigma}_C, \widehat{S}_C, \widehat{\Delta}_C)$$

$\widehat{\Sigma}_C = \{(sin{\cdot}cout) \mid sin \in \{0,1,2,3,4\} \wedge cout \in \{L,X,H\}\}$			
$\widehat{S}_C = \{L, H\}$			
$\widehat{\Delta}_C(\langle(sin{\cdot}cout)\rangle, s, s') \stackrel{\text{def}}{\equiv}$			
$s =$	$sin \in$	$cout =$	$s' =$
L	$\{0,1,2\}$	X	H
L	$\{3,4\}$	L	L
H	$\{0,1,2\}$	H	H
H	$\{3,4\}$	X	L

5.3 A discrete conservative approximation of the controller

The approximation of the controller, $\widehat{M}_C$ is presented in Table 9. The model is a simple two-state automaton. Let ρ_P be the state translation relation that maps every state of M_C to every state of $\widehat{M}_C$. It is a simple matter to verify that $\widehat{M}_C$ is a conservative approximation of M_C under the relation $R(\Gamma(\widetilde{\sigma}_o \,\|\, \widetilde{\sigma}_c), \rho_P)$.

6 Composition of Conservative Approximations

In Section 3.3, we defined the parallel composition of two automata, and used this definition to compose the automata M_I and M_A. We now have in hand $\widehat{M}_I$ and $\widehat{M}_A$, which we have proved to be conservative approximations of M_I and M_A under the mappings $R(\Gamma(\widetilde{\sigma}_i \,\|\, \widetilde{\sigma}_o), \rho_I)$ and $R(\Gamma(\widetilde{\sigma}_c \,\|\, \widetilde{\sigma}_i), \rho_A)$ respectively. In this section we show that $\widehat{M}_A \,\|\, \widehat{M}_I$ is a conservative approximation of $M_A \parallel M_I$, under the relation $R(\Gamma((\widetilde{\sigma}_c \parallel \widetilde{\sigma}_i) \parallel (\widetilde{\sigma}_i \parallel \widetilde{\sigma}_o)), \rho_A{\cdot}\rho_I)$.

We begin by specialising Lemma 5.3 for composite automata.

Lemma 6.1 *$w \in L(m \,\|\, n, I, F)$ if and only if there exist states i, j, f and g, and traces u and v satisfying $(i,j) \in I$, $(f,g) \in F$, $w = (u \parallel v)$, $u \in L(m, \{i\}, \{f\})$ and $v \in L(n, \{j\}, \{g\})$.*

Proof: Let Δ_m, Δ_n and $\Delta_{m\|n}$ be the state transition relations for m, n and $m \,\|\, n$ respectively. We prove the "only if" part of the lemma first. Since $w \in L(m \,\|\, n, I, F)$ we know that there exists an integer, z, a sequence $w_1, w_2, ..., w_z$ and a sequence $s_0, s_1, ..., s_z$ such that $(w_k, s_{k-i}, s_k) \in \Delta_{m\|n}$ for each $k \in \{1, ..., z\}$. From the definition of parallel composition of automata, we know that each w_k can be expressed as an ordered pair, (u_k, v_k), and each state s_k can be expressed as an ordered pair, (a_k, b_k), so that $(u_k, a_{k-1}, a_k) \in \Delta_m$, and $v_k, b_{k-1}, b_k \in \Delta_n$ for each $k \in \{1, ..., z\}$. Thus, we can conclude that $(u_1 u_2 \cdots u_z) \in L(m, \{a_0\}, \{a_z\})$ and that $(v_1 v_2 \cdots v_z) \in L(n, \{b_0\}, \{b_z\})$. On the other hand, from repeated application of Lemma 3.1 we know that $(u_1 u_2 \cdots u_z) \parallel (v_1 v_2 \cdots v_z) = (u_1 \parallel v_1)(u_2 \parallel v_2) \cdots (u_z \parallel v_z)$ and hence $(u_1 u_2 \cdots u_z) \,\|\, (v_1 v_2 \cdots v_z) = w$ as required.

To prove the "if' part, suppose that $u \in L(m, \{i\}, \{f\})$ and $v \in L(m, \{j\}, \{g\})$. Then there exist sequences $(u_1, ..., u_x)$ and $(a_0, ..., a_x)$ such that $u = u_1 u_2 \cdots u_x$, $a_0 = i$, $a_k = f$

and $(u_k, a_{k-1}, a_k) \in \Delta_m$ for each $k \in \{1, 2, ..., x\}$. Similarly, there exist sequences $v_1, ..., v_y$ and $b_0, ..., b_y$ such that $v = v_1 v_2 \cdots v_x$, $b_0 = j$, $b_k = g$ and $(v_k, b_{k-1}, b_k) \in \Delta_n$ for each $k \in \{1, 2, ..., y\}$. The problem is that the sequences will generally be of different lengths, as will each trace u_k and v_k. However, since u and v have the same length, and both automata are eager, we can always further subdivide the traces in each sequence as necessary, so that corresponding traces from each automaton have the same length. Thus, there exists an integer z and sequences $\widehat{v}_1, ..., \widehat{v}_z$, $\widehat{u}_1, ..., \widehat{u}_z$, $\widehat{a}_0, ..., \widehat{a}_z$ and $\widehat{b}_0, ..., \widehat{b}_z$ such that $\widehat{a}_0 = i$, $\widehat{a}_z = f$, $\widehat{b}_0 = j$, $\widehat{b}_z = g$, $v = \widehat{v}_1 \widehat{v}_2 \cdots \widehat{v}_z$, $u = \widehat{u}_1 \widehat{u}_2 \cdots \widehat{u}_z$, $(\widehat{u}_k, \widehat{a}_{k-1}, \widehat{a}_k) \in \Delta_m$, $(\widehat{v}_v, \widehat{b}_{k-1}, \widehat{b}_k) \in \Delta_n$, and $|\widehat{u}_k| = |\widehat{v}_k|$ for each $k \in 1, 2, ..., z$. Now, from the definition of parallel composition for automata, we can conclude that $(u_k \,\|\, v_k, (a_{k-1}, b_{k-1}), (a_k, b_k)) \in \Delta_{m\|n}$ for each $k \in \{1, 2, ..., z\}$. Thus $(u_1 \,\|\, v_1)(u_2 \,\|\, v_2) \cdots (u_k \,\|\, v_k) \in L(m \,\|\, n, \{(i,j)\}, \{(f,g)\})$. From repeated application of Lemma 3.1 we obtain $((u_1 u_2 \cdots u_z) \,\|\, (v_1 v_2 \cdots v_z)) = (u_1 \,\|\, v_1)(u_2 \,\|\, v_2) \cdots (u_k \,\|\, v_k)$. Hence $((u_1 u_2 \cdots u_z) \,\|\, (v_1 v_2 \cdots v_z)) \in L(m \,\|\, n, \{(i,j)\}, \{(f,g)\})$. Since $u = u_1 u_2 \cdots u_z$, $v = v_1 v_2 \cdots v_z$, $\{(i,j)\} \subseteq I$, and $\{(f,g)\} \subseteq F$ we can conclude that $(u \,\|\, v) \in L(m \,\|\, n, I, F)$ as required. □

Before considering the specific case of M_I and M_A, we look at the more general situation. Suppose that we are given automata m_1, $\widehat{m}_1$, m_2, and $\widehat{m}_2$, along with relations $R_1 = R(\sigma_1, \rho_1)$ and $R_2 = R(\sigma_2, \rho_2)$. Further, suppose that we have established that $\widehat{m}_1$ is a conservative approximation of m_1 under the relation R_1, and that $\widehat{m}_2$ is a conservative approximation of m_2 under the relation R_2. Theorem 6.2 states that $\widehat{m}_1 \,\|\, \widehat{m}_2$ is a conservative approximation of $m_1 \,\|\, m_2$ under the relation $R((\sigma_1 \,\|\, \sigma_2), (\rho_1 \cdot \rho_2))$.

Theorem 6.2 *Parallel Composition: Let $m_1 = (\Sigma_1, S_1, \Delta_1)$, $m_2 = (\Sigma_2, S_2, \Delta_2)$, $\widehat{m}_1 = (\widehat{\Sigma}_1, \widehat{S}_1, \widehat{\Delta}_1)$ and $\widehat{m}_2 = (\widehat{\Sigma}_2, \widehat{S}_2, \widehat{\Delta}_2)$ be safety automata. Let $\rho_1 \subseteq S_1 \times \widehat{S}_1$ and $\rho_2 \subseteq S_2 \times \widehat{S}_2$ be state translation relations. Let $\sigma_1 \subseteq \Sigma_1^* \times \widehat{\Sigma}_1^*$ and $\sigma_2 \subseteq \Sigma_2^* \times \widehat{\Sigma}_2^*$ be input translation relations.*

If $\widehat{m}_1$ (resp. $\widehat{m}_2$) is a conservative approximation of m_1 (resp. m_2) under $R(\sigma_1, \rho_1)$ (resp. $R(\sigma_2, \rho_2)$) then $\widehat{m}_1 \,\|\, \widehat{m}_2$ is a conservative approximation of $m_1 \,\|\, m_2$ under $R((\sigma_1 \,\|\, \sigma_2), (\rho_1 \cdot \rho_2))$.

Proof: To show that $\widehat{m}_1 \,\|\, \widehat{m}_2$ is a conservative approximation of $m_1 \,\|\, m_2$, we must show that the following implication holds for any pair of questions $q = (w_1 \,\|\, w_2, I, F)$ and $\widehat{q} = (\widehat{w}_1 \,\|\, \widehat{w}_2, \widehat{I}, \widehat{F})$.

$$(q, \widehat{q}) \in R((\sigma_1 \,\|\, \sigma_2), (\rho_1 \cdot \rho_2)) \;\wedge\; (w_1 \,\|\, w_2) \in L(m_1 \,\|\, m_2, I, F) \Longrightarrow (\widehat{w}_1 \,\|\, \widehat{w}_2) \in L(\widehat{m}_1 \,\|\, \widehat{m}_2, \widehat{I}, \widehat{F})$$

Let $q = (w_1 \,\|\, w_2, I, F)$ and $\widehat{q} = (\widehat{w}_1 \,\|\, \widehat{w}_2, \widehat{I}, \widehat{F})$ be such an arbitrary pair of questions. If $(q, \widehat{q}) \notin R((\sigma_1 \,\|\, \sigma_2), (\rho_1 \cdot \rho_2))$ or if $(w_1 \,\|\, w_2) \notin L(m_1 \,\|\, m_2, I, F)$, then the implication is satisfied trivially. Assume that

$$((w_1 \,\|\, w_2, I, F), (\widehat{w}_1 \,\|\, \widehat{w}_2, \widehat{I}, \widehat{F})) \in R(\sigma_1 \,\|\, \sigma_2), \rho_1 \cdot \rho_2)$$

and

$$(w_1 \,\|\, w_2) \in L(m_1 \,\|\, m_2, I, F).$$

Since $(w_1 \,\|\, w_2) \in L(m_1 \,\|\, m_2, I, F)$ we can conclude from Lemma 6.1 that there exist states i_1, i_2, f_1 and f_2, satisfying $(i_1 \cdot i_2) \in I$, $(f_1 \cdot f_2) \in F$, $w_1 \in L(m_1, \{i_1\}, \{f_1\})$ and $w_2 \in L(m_2, \{i_2\}, \{f_2\})$. The automaton $\widehat{m}_1$ is a conservative approximation of m_1 under

$R(\sigma_1, \rho_1)$. From Lemma 5.4 we can conclude that there exist states $\widehat{\imath}_1$ and $\widehat{f}_1$ such that $(i_1, \widehat{\imath}_1) \in \rho_1$, $(f_1, \widehat{f}_1) \in \rho_1$ and $\widehat{w}_1 \in L(m_1, \{\widehat{\imath}_1\}, \{\widehat{f}_1\})$. Similarly, there exist states $\widehat{\imath}_2$ and $\widehat{f}_2$ such that $(i_2, \widehat{\imath}_2) \in \rho_2$, $(f_2, \widehat{f}_2) \in \rho_2$ and $\widehat{w}_2 \in L(m_2, \{\widehat{\imath}_2\}, \{\widehat{f}_2\})$. From the definition of $R((\sigma_1 \| \sigma_2), (\rho_1 \cdot \rho_2))$ we can conclude that $(\widehat{\imath}_1 \cdot \widehat{\imath}_2) \in \widehat{I}$, and that $(\widehat{f}_1 \cdot \widehat{f}_2) \in \widehat{F}$. Lemma 6.1 allows us to conclude that $\widehat{w}_1 \| \widehat{w}_2 \in L(m_1 \| m_2, \widehat{I}, \widehat{F})$ as required. $\square$

Returning to $\widehat{M_I}$ and $\widehat{M_A}$ from Sections 5.1 and 5.2, it is an immediate consequence of Theorem 6.2 that $\widehat{M_A} \parallel \widehat{M_I}$ is a conservative approximation of $M_A \parallel M_I$ under the translation $R(\Gamma(\widetilde{\sigma}_c \parallel \widetilde{\sigma}_i) \parallel \Gamma(\widetilde{\sigma}_i \parallel \widetilde{\sigma}_o), \rho_A \cdot \rho_I)$. Lemma 6.3, which follows, allows us to conclude that $\widehat{M_A} \| \widehat{M_I}$ is a conservative approximation of $M_A \| M_I$ under the translation $R(\Gamma((\widetilde{\sigma}_c \| \widetilde{\sigma}_i) \| (\widetilde{\sigma}_i \| \widetilde{\sigma}_o)), \rho_A \cdot \rho_I)$.

Lemma 6.3 *For all binary relations f and g the following equivalence holds.*

$$\Gamma(f \| g) \equiv \Gamma(f) \| \Gamma(g)$$

Proof: Let (u, v) be a pair in $\Gamma(f \| g)$. There exist traces $u_1, ..., u_k$ and $v_1, ..., v_k$, each of length 1 such that $u = u_1 \cdots u_k$ and $v = v_1 \cdots v_k$, and such that $(u_i, v_i) \in f \| g$ for each $i \in \{1, \cdots, k\}$. Since $u_i, v_i \in f \| g$, there must exist u_i^f, u_i^g, v_i^f and v_i^g such that $(u_i^f, v_i^f) \in f$ and $(u_i^g, v_i^g) \in g$. Clearly $((u_0^f \cdots u_k^f), (v_0^f \cdots v_k^f)) \in \Gamma f$ and $((u_0^g \cdots u_k^g), (v_0^g \cdots v_0^g)) \in \Gamma(g)$. Hence $(((u_0^f \cdots u_k^f) \| (u_0^g \cdots u_k^g)), ((v_0^f \cdots v_k^f) \| (v_0^g \cdots v_0^g))) \in \Gamma(f) \| \Gamma(g)$. Repeated application of Lemma 3.1 allows us to conclude that $(((u_0^f \cdots u_k^f) \| (u_0^g \cdots u_k^g)) = u$ and $(((v_0^f \cdots v_k^f) \| (v_0^g \cdots v_k^g)) = v$. The reverse construction shows that if $(u, v) \in \Gamma(f) \| \Gamma(g)$ then $(u, v) \in \Gamma(f \| g)$. $\square$

7 Communication and Conservative Approximations

Suppose we have constructed a conservative approximation $\widehat{m}$ of an automaton m, under some mapping relation $R(\sigma, \rho)$. Moreover, suppose that we have constructed transliterations δ and $\widehat{\delta}$ which we compose with m and $\widehat{m}$ respectively. We now wish to construct a relation σ', so that $\widehat{\delta}\widehat{m}$ is a conservative approximation of δm under the relation $R(\sigma', \rho)$. The reader is referred to Figure 7 for a graphic representation of the relationship between the relations σ and σ', and the transliterations δ and $\widehat{\delta}$.

Theorem 7.1 *Serial Composition: Let $m = (\Sigma, S, \Delta)$ and $\widehat{m} = (\widehat{\Sigma}, \widehat{S}, \widehat{\Delta})$ be arbitrary safety automata. Let $\delta \subseteq \Sigma' \times \Sigma$ and $\widehat{\delta} \subseteq \widehat{\Sigma}' \times \widehat{\Sigma}$ be transliterations. Let $\rho \subseteq S \times \widehat{S}$, be a state translation relation, and let $\sigma \subseteq \Sigma^* \times \widehat{\Sigma}^*$ be an input translation relation.*

If $\widehat{m}$ is a conservative approximation of m under $R(\sigma, \rho)$ then $\widehat{\delta}\widehat{m}$ is a conservative approximation of δm under $R(\sigma', \rho)$, provided that every pair of traces $(w', \widehat{w}') \in \sigma'$ satisfies the following condition.

$$\forall w \in \mathrm{post}(\delta; \{w'\}) \cdot \exists \widehat{w} \in \mathrm{post}(\widehat{\delta}; \{\widehat{w}'\}) \cdot (w \sigma \widehat{w})$$

Proof: Suppose that (w', I, F) and $(\widehat{w}', \widehat{I}, \widehat{F})$ are questions such that $w' \in L(\delta m, I, F)$ and $((w', I, F), (\widehat{w}', \widehat{I}, \widehat{F})) \in R(\sigma', \rho)$. We must show that $\widehat{w}' \in L(\widehat{m}, \widehat{I}, \widehat{F})$. Since $w' \in L(\delta m, I, F)$, there must exist a trace $w \in \mathrm{post}(\delta; \{w'\})$ such that $w \in L(m, I, F)$. Let

$$\boxed{\begin{array}{ccccc} \forall w & \leftarrow & \sigma & \rightarrow & \exists \widehat{w} \\ \uparrow & & & & \uparrow \\ \delta & & & & \widehat{\delta} \\ \downarrow & & & & \downarrow \\ w' & \leftarrow & \sigma' & \rightarrow & \widehat{w}' \end{array}}$$

Figure 6: The relationship between transliterated traces and their approximations

w be such a trace. By assumption, $((w', I, F), (\widehat{w}', \widehat{I}, \widehat{F})) \in R(\sigma', \rho)$, hence $(w', \widehat{w}') \in \sigma'$. According to the definition of σ' given in the theorem, there exists a trace $\widehat{w} \in \text{post}(\widehat{\delta}; \{\widehat{w}'\})$ such that $(w\sigma\widehat{w})$. Let $\widehat{w}$ be such a trace. Since $((w', I, F), (\widehat{w}', \widehat{I}, \widehat{F})) \in R(\sigma', \rho)$, and $w\sigma\widehat{w}$, we must have $((w, I, F), (\widehat{w}, \widehat{I}, \widehat{F})) \in R(\sigma, \rho)$. By assumption, $\widehat{m}$ is a conservative approximation of m under $R(\sigma, \rho)$, so we can conclude that $\widehat{w} \in L(\widehat{m}, \widehat{I}, \widehat{F})$. Since $\widehat{w}'\widehat{\delta}\widehat{w}$, we can conclude that $\widehat{w}' \in L(\widehat{\delta}\widehat{m}, \widehat{I}, \widehat{F})$ as required. □

Theorem 7.2 *If $\widehat{m}$ is a conservative approximation of m under the relation $R(\Gamma(\widetilde{\sigma}), \rho)$, then $\widehat{\delta}\widehat{m}$ is a conservative approximation of δm under the relation $R(\Gamma(\widetilde{\sigma}'), \rho)$ provided that every pair $(w', \widehat{w}') \in \widetilde{\sigma}'$ satisfies the following condition.*

$$\forall w \in \text{post}(\delta; \{w'\}) \cdot \exists \widehat{w} \in \text{post}(\widehat{\delta}; \{\widehat{w}'\}) \cdot (w\widetilde{\sigma}\widehat{w}) \tag{1}$$

Proof: We prove the theorem by showing that if every pair of traces $(w', \widehat{w}') \in \widetilde{\sigma}'$ satisfies condition 1, then every pair $(w', \widehat{w}') \in \Gamma(\widetilde{\sigma}')$ must satisfy the following condition.

$$\forall w \in \text{post}(\delta; \{w'\}) \cdot \exists \widehat{w} \in \text{post}(\widehat{\delta}; \{\widehat{w}'\}) \cdot (w, \widehat{w}) \in \Gamma(\widetilde{\sigma}) \tag{2}$$

The theorem then follows as a direct consequence of Theorem 7.1. Assume that every pair $(w', \widehat{w}') \in \widetilde{\sigma}'$ satisfies condition 1. Let $(w', \widehat{w}')$ be an arbitrary pair of traces in $\Gamma(\widetilde{\sigma}')$. By the definition of Γ there must exists traces $w'_1, ..., w'_k$, and $\widehat{w}'_1, ..., \widehat{w}'_k$, each of length 1, such that $w' = w'_1 \cdots w'_k$, $\widehat{w}' = \widehat{w}'_1 \cdots \widehat{w}'_k$, and such that $(w'_i, \widehat{w}'_i) \in \widetilde{\sigma}'$ for each $i \in \{1, .., k\}$.

Let w be a trace in $\text{post}(\delta; \{w'\})$. Since δ is a transliteration, there must be traces $w_1, ..., w_k$, each of length 1, such that $w = w_1 \cdots w_k$, and such that $w_i \in \text{post}(\delta; \{w'_i\})$ for each $i \in \{1, ..., k\}$. By assumption, there exists $\widehat{w}_i \in \text{post}(\widehat{\delta}; \{\widehat{w}'_i\})$ such that $w_i\widetilde{\sigma}\widehat{w}_i$. Let $\widehat{w} = \widehat{w}_0 \cdots \widehat{w}_i$. By the definition of Γ, $(w, \widehat{w}) \in \Gamma(\widetilde{\sigma})$. Thus we have shown that every pair of traces $w', \widehat{w}' \in \Gamma(\widetilde{\sigma}')$ satisfies condition 2 as required. □

To conclude this section, we complete the conservative approximation of the water tank plant-model. So far, we have established that $\widehat{M_A} \parallel \widehat{M_I}$ is a conservative approximation of $M_A \parallel M_I$ under the translation $R(\Gamma((\widetilde{\sigma}_c \parallel \widetilde{\sigma}_i) \parallel (\widetilde{\sigma}_i \parallel \widetilde{\sigma}_o)), \rho_A \cdot \rho_I)$. Ultimately, we wish to provide a conservative approximation for the plant-model $\delta_P(M_A \parallel M_I)$. Let $\widehat{\delta}_P$ be the following transliteration.

$$\begin{aligned} &\widehat{\delta}_P((cin' \cdot mout'), ((cin \cdot cout) \cdot (win \cdot mout))) \stackrel{\text{def}}{\equiv} \\ &\qquad cin' = cin \land \\ &\qquad mout' = mout \land \\ &\qquad cout = win \end{aligned}$$

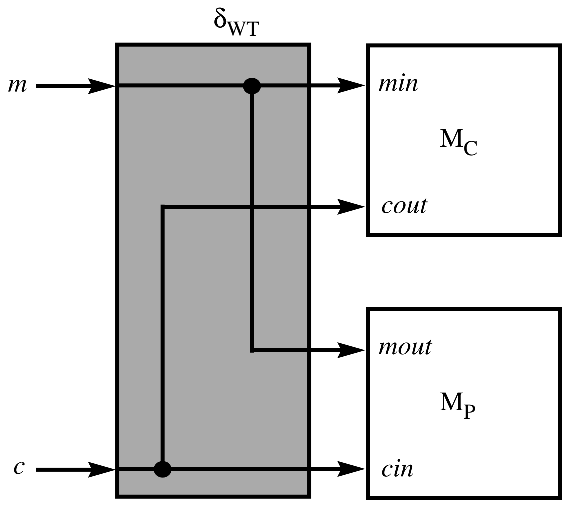

Figure 7: Connecting the controller with the plant

8 Verification Using Conservative Approximations

Throughout this paper, we have focused on the task of building conservative models. Ultimately, we would like to use these models to verify that the system has certain properties. For example, we show how to verify that the controller modeled in Section 3.2 will not allow the water tank to over-fill.

We have developed a hybrid plant model $M_P = \delta_P(M_A \,\|\, M_I)$, by composing and connecting two continuous safety automata. We have developed a discrete approximation of this model, $\widehat{M}_P = \widehat{\delta}_P(\widehat{M}_A \,\|\, \widehat{M}_I)$, by composing and connecting two finite-state automata. Moreover, we showed that $\widehat{M}_P$ was a conservative approximation of M_P under the relation $R(\Gamma(\widetilde{\sigma}_c \,\|\, \widetilde{\sigma}_o), \rho_A \cdot \rho_I)$, by virtue of its construction.

We have also developed a model M_C of the proposed controller for this plant, and a discrete conservative approximation of it, $\widehat{M}_C$. We model the behaviour of the combined plant and controller by composing and connecting M_C with M_P. Let δ_{WT} be the following transliteration.

$$\begin{aligned}\delta_{WT}((s \cdot c), ((sin \cdot cout) \cdot (cin \cdot sout))) &\stackrel{\text{def}}{\equiv} \\ s = sin = sout \ \wedge& \\ c = cin = cout&\end{aligned}$$

The combination of controller and plant, as illustrated in Figure 7, is modeled by the automaton, $M_{WT} = \delta_{WT}(M_C \,\|\, M_P)$. The automaton has states $(c \cdot p)$, consisting of two components, the controller state c and the plant state p. Its input-set combines sensor and control signals. The automaton is able to make a transition from one state $(c \cdot p)$ to another $(c' \cdot p')$ if and only if the controller model M_C can make the transition from c to c', and the plant model M_P can make the transition from p to p'.

For convenience, let $\sigma_{WT} = \Gamma(\widetilde{\sigma}_o \,\|\, \widetilde{\sigma}_c)$, and let $\rho_{WT} = \rho_c \cdot (\rho_A \cdot \rho_I)$. It is a straightforward application of the theory developed in Sections 6 and 7 to show that $\widehat{M}_{WT} = \delta_{WT}(\widehat{M}_C \,\|\, \widehat{M}_P)$ is a conservative approximation of M_{WT} under the relation $R(\sigma_{WT}, \rho_{WT})$.

Suppose we wish to show that, provided the system is started with the water-tank empty, the controller will never fill the water-tank to the brim. We first translate this question, in to questions about the continuous model M_{WT}. Let I be the set of system states in which the tank is empty.

$$I = \{(c \cdot (a \cdot i)) \in S_{WT} \mid i = 0\}$$

Let F be the set of system states in which the tank is full.

$$F = \{(c \cdot (a \cdot i)) \in S_{WT} \mid i = 4\}$$

We wish to show that for all traces $w \in \Sigma_{WT}^*[\mathcal{R}]$, $w \notin L(M_{WT}, I, F)$. That is, we wish to show that $L(M_{WT}, I, F)$ is the empty set. Of course, it may well be possible to show this directly, from analysis of the continuous model M_{WT}. In general, however, such analysis will be too complex to be tractable. Instead, we will use the discrete conservative approximation $\widehat{M}_{WT}$. We need to show that for all $w \in \Sigma_{WT}$, there exists $\widehat{w}$, $\widehat{I}$, and $\widehat{F}$ such that $((w, I, F), (\widehat{w}, \widehat{I}, \widehat{F})) \in R(\sigma_{WT}, \rho_{WT})$ and such that $\widehat{w} \notin L(\widehat{M}_{WT}, \widehat{I}, \widehat{F})$.

Expanding the definition of $R(\sigma, \rho)$ shows that we need to find $\widehat{w}$, $\widehat{I}$, and $\widehat{F}$ such that $(w, \widehat{w}) \in \sigma_{WT}$, $\mathrm{post}(\rho_{WT}; I) \subseteq \widehat{I}$, and $\mathrm{post}(\rho_{WT}; F) \subseteq \widehat{F}$. It is straightforward to show that $\widehat{I} = \left\{\widehat{c} \cdot (\widehat{s} \cdot \widehat{\imath}) \in \widehat{S}_{WT} \mid \widehat{\imath} = 0\right\}$, and $\widehat{F} = \left\{\widehat{c} \cdot (\widehat{s} \cdot \widehat{\imath}) \in \widehat{S}_{WT} \mid \widehat{\imath} = 4\right\}$ will suffice for $\widehat{I}$ and $\widehat{F}$. To complete the proof, we must show that for every trace w in which we are interested, there is a related trace $\widehat{w}$, with which we can form a question to pose to the approximation. Recall that $\sigma_{WT} = \Gamma(\widetilde{\sigma}_o \cdot \widetilde{\sigma}_c)$. Clearly, for every trace $w \in \Sigma_{WT}^*$ of length less than 1, there is a related trace $\widehat{w} \in \widehat{\Sigma}^1$ such that $(w, \widehat{w}) \in (\widetilde{\sigma}_o \parallel \widetilde{\sigma}_c)$. The operator Γ partions longer traces into a sequence traces of length not greater than 1, to which $(\widetilde{\sigma}_o \parallel \widetilde{\sigma}_c)$ can be applied.

We wish to show that no traces are in $L(M_{WT}, I, F)$. Since every possible trace w has a translation $\widehat{w}$, and since $\widehat{M}_{WT}$ is a conservative approximation of M_{WT}, we need only show that there are no traces in $L(\widehat{M}_{WT}, \widehat{I}, \widehat{F})$. This can easily be confirmed by conventional model-checking techniques, such as those described by Burch *et al.* in [2].

9 Conclusions and Future Work

In this paper, we have presented an approach to the problem of verifying hybrid systems. We model the system initially using a combination of both continuous and discrete mathematics. Subsequently, a discrete conservative approximation of this model is build. Finally, the discrete model is verified against a translation of the original specification using conventional techniques.

To illustrate this approach, we have verified that a simple control mechanism does not overflow the water-tank to which it is attached. The example system, a water tank, is composed of several parts, each with different behaviours. The tank itself behaves continuously according to the laws of physics. The valve mechanism is event driven, while the controller is essentially discrete. Each of these parts is modeled independently by a *continuous automata*. The models of the parts are then composed and connected, to form a model of the complete system.

We then developed a discrete finite-state approximation of this continuous model. We began by developing approximations of the components — the tank, the actuator, and the controller. Each of these primitive approximations was demonstrated to be a conservative approximation of the corresponding continuous model. An approximation of the entire system was developed by performing the same composition and connection operations on these primitive approximations that were performed on the originals. The composition and connection operations that were used preserve conservative approximation for our specification language. Thus, we were able to conclude that the resulting discrete finite-state automata was a conservative approximation of the original continuous automata. Finally, we argued that conventional model-checking techniques could be used to show that the discrete model satisfied a related specification, and hence that the original specification was satisfied by the continuous system.

This research was originally motivated by the desire to design and verify a control system for our model train set. Although this goal remains unfulfilled, the theory developed here represents a significant step in this direction. In principle, the process of composing and connecting the primitive approximations could have been accomplished automatically. We envision a tool, with a built-in set of primitive continuous models and their (parameterised) approximations. The user would construct a model of the system under investigation, by composing and connecting these built-in primitives. The tool would automatically perform the same compositions and connections on the primitive approximations, producing a conservative approximation of the entire system, suitable for verification. Identifying a useful set of primitives, and appropriate approximations of them remains a matter for future research.

For the purpose of illustration, we have limited the expressiveness of our specification language. The modeling frameworks, continuous and discrete automata, are likewise very simple. In spite of this simplicity, we have been surprised by their expressiveness. Nontheless, these limitations are more historical than fundamental to the approach. Developing approximations for other types of specifications seems like a natural way to extend the work. Here it seems likely that one will wish to develop different approximations for different kinds of questions. For example it may well be best to develop one approximation that is conservative with respect to safety questions, and another that is conservative with respect to liveness.

We are suggesting an approach to hybrid system verification, which essentially reduces the problem to that of verifying discrete systems. The latter, has been extensively studied, and considerable progress has recently been made. One possible objection to this approach is its failure to exploit the underlying continuity of the physical systems being modelled. As it stands, this continuity is exploited, if at all, only in the development of primitive approximations. Whether the continuity of an underlying system can be exploited to simplify the verification of its discrete approximation, remains an intriguing question for future research.

A Proofs of conservative approximations

A.1 Integrator

In Section 5.1 we claimed that the discrete model $\widehat{M}_I$ was a conservative approximation of the integrator model M_I under the relation $R(\Gamma(\widetilde{\sigma}_i \,\|\, \widetilde{\sigma}_o), \rho_I)$. We offer the following proof.

We begin by providing a simplified way to prove that conservative approximation holds in the special case that the state-mapping relation σ actually describes a function from S to $\widehat{S}$.

Lemma A.1 *If ρ is a function, that is, for all $s, \widehat{s}_1, \widehat{s}_2$, $((s, \widehat{s}_1) \in \rho \ \wedge\ (s, \widehat{s}_2) \in \rho) \Longrightarrow \widehat{s}_1 = \widehat{s}_2$, and if $\widehat{m}$ is a discrete conservative approximation of m under $R(\widetilde{\sigma}, \rho)$, then $\widehat{m}$ is* *a conservative approximation of m under $R(\Gamma(\tilde{\sigma}), \rho)$.*

Proof: Let Σ and S be the input-set and state-set of m, and let $\widehat{\Sigma}$ and $\widehat{S}$ be the input-set and state-set of $\widehat{m}$. Lemma 5.3 says that we can prove that $\widehat{m}$ is a conservative approximation of m under $R(\Gamma(\tilde{\sigma}), \rho)$ by showing that the following implication holds for all traces w and $\widehat{w}$, and for all states i and f.

$$\begin{aligned}&(w, \widehat{w}) \in \Gamma(\tilde{\sigma}) \wedge w \in L(m, \{i\}, \{f\}) \Longrightarrow \\ &\qquad \exists \widehat{\imath}, \widehat{f} \in \widehat{S} \cdot (i \rho \widehat{\imath}) \wedge (f \rho \widehat{f}) \wedge \widehat{w} \in L(\widehat{m}, \{\widehat{\imath}\}, \left\{\widehat{f}\right\})\end{aligned}$$

Let w be an arbitrary trace in $\Sigma^*[\mathcal{R}]$ and let i and f be arbitrary states in S. Also, let $\widehat{w}$ be an arbitrary trace in $\widehat{\Sigma}^*[\mathcal{N}]$. If $(w, \widehat{w}) \notin \Gamma(\tilde{\sigma})$ or if $w \notin L(m, \{i\}\{f\})$ then the implication is satisfied trivially. Assume that $(w, \widehat{w}) \in \Gamma(\tilde{\sigma})$ and that $w \in L(m, \{i\}, \{f\}$. From the definition of Γ, and the assumption that $(w, \widehat{w}) \in \Gamma(\tilde{\sigma})$, we can conclude that there exists a natural number k, traces $w_1, ..., w_k$, and traces $\widehat{w}_1, ..., \widehat{w}_k$ satisfying the following conditions.

$$\begin{aligned} w &= w_1 w_2 \cdots w_k \\ \widehat{w} &= \widehat{w}_1 \widehat{w}_2 \cdots \widehat{w}_k \\ \forall i \in \{1, 2, ..., k\} &\cdot (w_i, \widehat{w}_i) \in \tilde{\sigma}\end{aligned}$$

Since m is eager, and $w \in L(m, \{i\}, \{f\})$, there must exists states $s_0, ..., s_k$ satisfying the following conditions.

$$\begin{aligned} s_0 &= i \\ s_k &= f \\ \forall i \in \{1, ..., k\} &\cdot w_i \in L(m, \{s_{i-1}\}, \{s_i\})\end{aligned}$$

Moreover, since $\widehat{m}$ is a conservative approximation of m under $R(\tilde{\sigma}, \rho)$, we know that there exist states $\widehat{s}_0, ..., \widehat{s}_{k-1}$, and states $\widehat{s}'_1, ..., \widehat{s}'_k$ such that the following conditions hold for all $\forall i \in \{1, ..., k\}$.

$$\begin{aligned}(s_{i-1}, \widehat{s}_{i-1}) &\in \rho \ \wedge \\ (s_i, \widehat{s}'_i) &\in \rho\end{aligned}$$

Now, since ρ is known to be a function, we can conclude that each $\widehat{s}'_i = \widehat{s}_i = \tilde{\sigma}(s_i)$. Thus, by progressing through the sequence of states $\widehat{s}_0, \widehat{s}'_0, \widehat{s}'_1, ..., \widehat{s}'_{k-1}$ the automaton $\widehat{m}$ is able to accept the trace $\widehat{w}$. Thus, we have demonstrated that there exists initial state $\widehat{s}_0$ such that $(i, \widehat{s}_0) \in \tilde{\sigma}$, and a final state $\widehat{s}'_{k-1}$ such that $(f, \widehat{s}'_{k-1}) \in \tilde{\sigma}$ and such that $\widehat{w} \in L(\widehat{m}, \widehat{s}_0, \widehat{s}'_{k-1})$ as required. □

The relation ρ_I is a total function over the state-set S_I. Lemma A.1 says that we can prove that $\widehat{M}_I$ is a conservative approximation of M_I under $R(\Gamma(\tilde{\sigma}_i \| \tilde{\sigma}_o), \rho_I)$. by proving that for all traces $w \in \Sigma_I^*[\mathcal{R}]$, and $\widehat{w} \in \widehat{\Sigma}_I*[\mathcal{N}]$, and for all states i and f in S_I, the following implication holds.

$$\begin{aligned}&(w, \widehat{w}) \in (\tilde{\sigma}_i \| \tilde{\sigma}_o) \wedge w \in L(M_I, \{i\}, \{f\}) \Longrightarrow \\ &\qquad \exists \widehat{\imath}, \widehat{f} \in \widehat{S}_I \cdot (i \rho_I \widehat{\imath}) \wedge (f \rho_I \widehat{f}) \wedge \widehat{w} \in L(\widehat{M}_I, \{\widehat{\imath}\}, \left\{\widehat{f}\right\})\end{aligned}$$

Let w be an arbitrary trace in $\Sigma^*[\mathcal{R}]$ and let i and f be arbitrary states in S. Also, let $\widehat{w}$ be an arbitrary trace in $\widehat{\Sigma}^*[\mathcal{N}]$. If $(w, \widehat{w}) \notin (\tilde{\sigma}_i \| \tilde{\sigma}_o)$ or if $w \notin L(M_I, \{i\}\{f\})$ then the implication is satisfied trivially. Assume that $(w, \widehat{w}) \in (\tilde{\sigma}_i \| \tilde{\sigma}_o)$ and that $w \in L(M_I, \{i\}, \{f\})$. We will show that $\widehat{w} \in L(\widehat{M}_I, \{\rho_I(i)\}, \{\rho_I(f)\})$.

The trace w is expressible as the parallel combination of its two components. Let $w_i \in \mathcal{R}[-i_{max}, i_{max}]$ and $w_o \in \mathcal{R}[s_{min}, s_{max}]$ be the two components of w. That is, $w = w_i \parallel w_o$. The sequence $\widehat{w}$ consists of a single symbol, which has two components. Let these components be $\widehat{w}_i$ and $\widehat{w}_o$, so that $\widehat{w} = \langle(\widehat{w}_i \cdot \widehat{w}_o)\rangle$. To show that $\widehat{w} \in L(\widehat{M}_I, \{\rho_I(i)\}, \{\rho_I(f)\})$, we must show that $(\widehat{w}_i \cdot \widehat{w}_o) \in \widehat{\Sigma}_I$, that $\rho_I(i) \in \widehat{S}_I$, and that $\rho_I(f) \in \widehat{S}_I$. Additionally, we must show that there is a sequence of transitions in $\widehat{\Delta}_I$ that starts in state $\rho_I(i)$, consumes $\widehat{w}$, and ends in $\rho_I(f)$. We show that there is a sequence conisting of a single transition. From the last row of Table 7 we can see that we need to show that the following two conditions are satisfied

$$\rho_I(i) + \widehat{w}_i - 1 \leq \rho_I(f) \leq \rho_I(i) + \widehat{w}_i \tag{3}$$

$$\frac{\lceil \rho_I(i) \rceil + \lceil \rho_I(f) \rceil - \lceil i_{max} \rceil}{2} \leq \widehat{w}_o \leq \frac{\lceil \rho_I(i) \rceil + \lceil \rho_I(f) \rceil + \lceil i_{max} \rceil}{2} \tag{4}$$

The preliminary requirements are easily satisfied. The states i and f must lie in the range $[s_{min}, s_{max}]$, hence $\rho_I(i)$ and $\rho_I(f)$ must lie in the range $[\lceil s_{min} \rceil, \lceil s_{max} \rceil]$. Any real-valued function, g, must satisfy the following inequality for any $t \geq 0$.

$$\min_{0 \leq x \leq 1} tg(x) \leq \int_0^t g(x)dx \leq \max_{0 \leq x \leq 1} tg(x)$$

The value of $w_i(t)$ is restricted to the interval $[-i_{max}, i_{max}]$. Let $l = |w_i|$. Since $(w_i, \langle \widehat{w}_i \rangle) \in \widetilde{\sigma}_i$, $l \leq 1$. Now $\widehat{w}_i = \left\lceil \int_0^l w_i(t)dt \right\rceil$. Thus $\widehat{w}_i$ is restricted to the interval $[\lceil -i_{max} l \rceil, \lceil i_{max} l \rceil]$. The value of $w_o(t)$ must remain in the range $[s_{min}, s_{max}]$, and $\widehat{w}_o = \lceil w_o(t) \rceil$ for some time $t \in [0,1]$. Thus $\widehat{w}_o$ must lie in the interval $[\lceil s_{min} \rceil, \lceil s_{max} \rceil]$.

We are left with conditions 3 and 4. Let $k = \int w_i$. We know, from the definition of M_I, that $k = i - f$. Moreover, from the definitions of $\widetilde{\sigma}_i$ and ρ_I we know that $\widehat{w}_i = \lceil k \rceil$, $\rho_I(i) = \lceil i \rceil$, and $\rho_I f = \lceil f \rceil$. We begin with some simple observations of the function $g(x) = \lceil x \rceil$, which we simply state without proof.

Observaton A.2 $\forall x \in \mathcal{R} \cdot x \leq \lceil x \rceil < x + 1$

Observaton A.3 $\forall x, y \in \mathcal{R} \cdot \lceil x + y \rceil \leq \lceil x \rceil + \lceil y \rceil$

Substituting f for x in Observation A.2 we obtain

$$f \leq \lceil f \rceil < f + 1$$

Substituting $i + k$ for f yields

$$i + k \leq \lceil f \rceil < i + k + 1$$

The following sequence of inequalities follows trivially by repeated application of the Observations A.2 and A.3.

$$\begin{aligned}
\lceil k \rceil + \lceil i \rceil - 2 < \quad & \lceil f \rceil \quad < \lceil k + i \rceil + 1 \\
\lceil k \rceil + \lceil i \rceil - 2 < \quad & \lceil f \rceil \quad < \lceil k \rceil + \lceil i \rceil + 1 \\
\lceil k \rceil + \lceil i \rceil - 1 \leq \quad & \lceil f \rceil \quad \leq \lceil k \rceil + \lceil i \rceil
\end{aligned}$$

Replacing $\lceil k \rceil$ with $\widehat{w}_i$, $\lceil i \rceil$ with $\rho_I(i)$ and $\lceil f \rceil$ with $\rho_I(f)$ gives the following result, demonstrating that condition 3 is satisfied.

$$\widehat{w}_i + \rho_I i - 1 \leq \rho_I(f) \leq \widehat{w}_i + \rho_I(i)$$

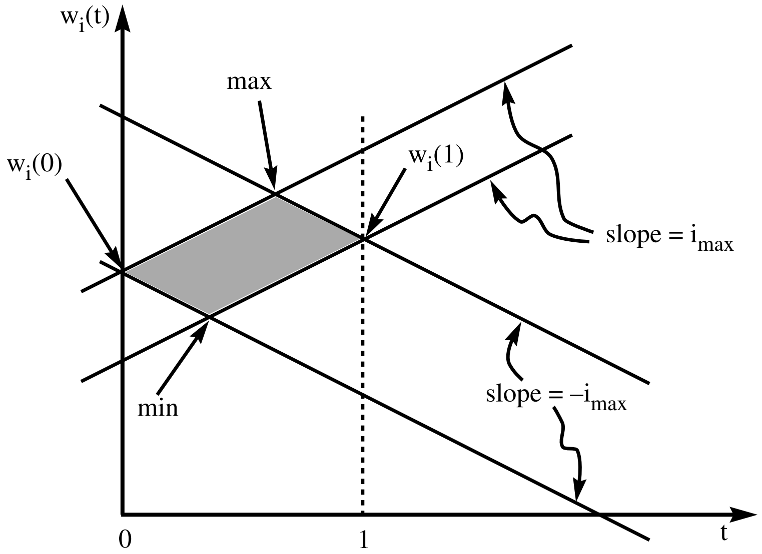

Figure 8: Constraints on integration output

The values of function w_i are restricted to the interval $[-i_{max}, +i_{max}]$. Since w_o is the integral of w_i, the slope of w_o is likewise constrained. Thus, for all times $t \in [0, 1]$, $w_o(t)$ must lie above the line with slope $-i_{max}$ that goes through the point $(0, w_i(0))$. Similarly, $w_i(t)$ must lie above the line with slope i_{max} that goes through the point $(1, w_i(1))$. As shown in Figure 8, the function $w_o(t)$ must remain above the point $(\frac{1}{2} + \frac{w_i(0)-w_i(1)}{2i_{max}}, \frac{w_i(0)+w_i(1)-i_{max}}{2})$, labelled min in the figure, at which these two lines intersect. Replacing $w_o(0)$ with i, and $w_o(1)$ with f gives a lower bound, $w(t) \geq \frac{i+f-i_{max}}{2}$. Similar analysis yields an upper bound, $w(t) \leq \frac{i+f-i_{max}}{2}$. From the definition of $\widetilde{\sigma}_o$, we can conclude that the following inequality holds.

$$\left\lceil \frac{i + f - i_{max}}{2} \right\rceil \leq \widehat{w}_o \leq \left\lceil \frac{i + f + i_{max}}{2} \right\rceil$$

Repeated application of observation A.2 and A.3 yields the following sequence of inequalities.

$$\begin{array}{rcl}
\left\lceil \frac{i + f - i_{max}}{2} \right\rceil \leq & \widehat{w}_o & \leq \left\lceil \frac{i + f + i_{max}}{2} \right\rceil \\
\frac{i + f - i_{max}}{2} \leq & \widehat{w}_o & < \frac{i + f + i_{max}}{2} + 1 \\
\frac{\lceil i \rceil + \lceil f \rceil - 2 - \lceil i_{max} \rceil}{2} < & \widehat{w}_o & < \frac{\lceil i \rceil + \lceil f \rceil + \lceil i_{max} \rceil}{2} + 1 \\
\frac{\lceil i \rceil + \lceil f \rceil - \lceil i_{max} \rceil}{2} - 1 < & \widehat{w}_o & < \frac{\lceil i \rceil + \lceil f \rceil + \lceil i_{max} \rceil}{2} + 1 \\
\frac{\lceil i \rceil + \lceil f \rceil - \lceil i_{max} \rceil}{2} \leq & \widehat{w}_o & \leq \frac{\lceil i \rceil + \lceil f \rceil + \lceil i_{max} \rceil}{2}
\end{array}$$

Replacing $\lceil i \rceil$ with $\rho_I(i)$, and $\lceil f \rceil$ with $\rho_I(f)$, yields the following final result, satisfying

condition 4
$$\frac{\rho_I(i)+\rho_I(f)-\lceil i_{max}\rceil}{2} < \widehat{w}_o \leq \frac{\rho_I(i)+\rho_I(f)+\lceil i_{max}\rceil}{2}$$

Thus all the conditions have been satisfied, and we have shown that $\widehat{w}$ is indeed a string in the language $L(\widehat{M}_I, \{\rho_I(i)\}, \{\rho_I(f)\})$, as required to complete the proof.

A.2 Actuator

We begin by extending concatenation from traces, to relations over traces. That is, if σ_1 and σ_2 are binary relations between traces, then we define their concatenation as follows:

$$\begin{aligned}&(u,v) \in \sigma_1\sigma_2 \stackrel{\text{def}}{\equiv}\\ &\quad \exists u_1, u_2, v_1 v_2 \cdot u = u_1 u_2 \ \wedge \ v = v_1 v_2 \ \wedge \ (u_1, v_1) \in \sigma_1 \ \wedge \ (u_2, v_2) \in \sigma_2\end{aligned}$$

Let $\widetilde{\sigma}$ be $\widetilde{\sigma}_c \,\|\, \widetilde{\sigma}_d$, restricted to traces of length 1.

$$(w,\widehat{w}) \in \widetilde{\sigma} \stackrel{\text{def}}{\equiv} |w| = 1 \ \wedge \ |\widehat{w}| = 1 \ \wedge \ (w,\widehat{w}) \in (\widetilde{\sigma}_c \,\|\, \widetilde{\sigma}_i)$$

Observe that $\Gamma(\widetilde{\sigma}_c \,\|\, \widetilde{\sigma}_i)$ is equal to the concatenation of $\Gamma(\widetilde{\sigma})$ and $(\widetilde{\sigma}_c \,\|\, \widetilde{\sigma}_i)$.

$$\Gamma(\widetilde{\sigma}_c \,\|\, \widetilde{\sigma}_i) = \Gamma(\widetilde{\sigma})(\widetilde{\sigma}_c \,\|\, \widetilde{\sigma}_i)$$

Let $\widetilde{\rho}$ be the the following function from S_A to $\widehat{S}_A$.

$$\begin{aligned}&((s{\cdot}c), \widehat{s}) \in \widetilde{\rho} \stackrel{\text{def}}{\equiv}\\ &\quad s = L \ \wedge \ c = 0 \ \wedge \ \widehat{s} = L \ \vee\\ &\quad s = H \ \wedge \ c = 0 \ \wedge \ \widehat{s} = L \ \vee\\ &\quad c > 0 \ \wedge \ \widehat{s} = X\end{aligned}$$

Recall that $\widetilde{\rho}_A$ relates every state in S_A to every state in $\widehat{S}_A$. Clearly, $\widetilde{\rho} \subseteq \widetilde{\rho}_A$.

We prove that $\widehat{M}_A$ is a conservative approximation of M_A under $R(\Gamma(\widetilde{\sigma}_c \,\|\, \widetilde{\sigma}_i), \rho_A)$ as follows. First, we show that $\widehat{M}_A$ is a conservative approximation of M_A under $R(\Gamma(\widetilde{\sigma}), \widetilde{\rho})$. We then show that for all w and $\widehat{w}$ such that $(w,\widehat{w}) \in (\widetilde{\sigma}_c \parallel \widetilde{\sigma}_i)$, and for all states i and f in S_A, if $w \in L(M_A, \{i\}, \{f\})$, then there exists a final state $\widehat{f} \in \widehat{S}_A$ such that $\widehat{w} \in L(\widehat{M}_I, \widetilde{\rho}(i), \widehat{f})$. It follows immediately that $\widehat{M}_A$ is a conservative approximation of M_A under the relation $R(\Gamma(\widetilde{\sigma})(\widetilde{\sigma}_c \,\|\, \widetilde{\sigma}_i), \rho_A)$.

Lemma 6.3 says that in order to show that $\widehat{M}_A$ is a conservative approximation of M_A under $R(\Gamma(\widetilde{\sigma}), \widetilde{\rho})$, it suffices to show that it is a conservative approximation under $R(\widetilde{\sigma}, \widetilde{\rho})$. We do this by simple case analysis. Let w be an arbitrary trace in Σ_A^*, for which there exists a sequence $\widehat{w} \in \widehat{\Sigma}_A^*$ such that $(w,\widehat{w}) \in \widetilde{\sigma}$. Moreover, suppose that i and f are arbitrary states, subject to the condition that $w \in L(M_A, \{i\}, \{f\})$. We must show that $\widehat{w} \in L(\widehat{M}_A, \{\widetilde{\rho}(i)\}, \{\widetilde{\rho}(f)\})$. Now, w can be expressed as $cin \,\|\, cout$, and $\widehat{w}$ can be expressed as $\left\langle (\widehat{cin} \cdot \widehat{cout}) \right\rangle$. Since $w, \widehat{w} \in \widetilde{\sigma}$ we know that $(cin, \left\langle \widehat{cin} \right\rangle) \in \widetilde{\sigma}_c$ and that $(cout, \left\langle \widehat{cout} \right\rangle) \in \widetilde{\sigma}_i$. We base our case analysis on the values of $\widetilde{\rho}(i)$ and $\widehat{cin}$.

Suppose that $\widetilde{\rho}(i) = L$, and hence $i = (L \cdot 0)$. Suppose that $\widehat{cin} = L$. Then cin must be the constant trace L. Clearly, the only transitions that M_A can take are those in row 1 of Table 3. Thus, the final state f must be $(L \cdot 0)$ as well, and $cout$ must be the constant trace 0. Since $cout$ is the constant trace 0, $\widehat{cout}$ must be $\langle 0 \rangle$. From row 1 of Table 8 we can see that indeed, the discrete model can start in state L, consume $\langle(L\cdot 0)\rangle$ and arrive in state L.

Suppose that $\widehat{cin} = X$. From row 2 of Table 8 we can see that the discrete can start in L, consume $\left\langle(X, \widehat{cout})\right\rangle$ for any symbol $cout$, and arrive in any other state. Thus, it will be able to arrive in state $\widetilde{\rho}(f)$ regardless of the value of $cout$.

Suppose that $\widehat{cin} = H$. Then cin must be the constant trace H, and must have a length of 1. The only transition that M_A can take on a trace of non-zero length is that given by row 2 of Table 3. However, this transition cannot consume the whole trace, since it must end in the state $(H \cdot (0.5 - l)$ where l is the length of the trace, and $0.5 - l$ must remain greater than *zero*. Suppose it makes such a transition, consuming a prefix of length μ_1. It must then be in the state $(H \cdot 0.5 - \mu_1)$. If μ_1 is less than 0.5, then the next transition must be from row 6 of the table. The trace consumed must be of length μ_2 such that $\mu_2 \leq 0.5 - \mu_1$. It arrives in the state $H{\cdot}0.5 - \mu_1 - \mu_2$. Clearly any sequence of k such transitions, whose length totals less than 0.5 is equivalent to a single transition of the same total length. Since, eventually, we must consume the whole trace of length 1, we must arrive at state $(H\cdot 0)$ after consuming a trace of length 0.5. Having arrived at state $(H\cdot 0)$, the only transition that can be taken is that given by row 4, and the state remains $(H\cdot 0)$ for the remainder of the trace. Thus, if M_A starts in state $(L\cdot 0)$ and consumes a trace of length 1 where the first component is the constant H, it must arrive at state $(H\cdot 0)$. From row 3 of Table 8 can make the corresponding transition, regardless of the value of $cout$.

Suppose on the other hand that $\widetilde{\rho}(i) = X$. If $\widehat{cin} = L$, then by the same argument as used above, for the case $\widetilde{\rho}(i) = L \wedge \widehat{cin} = H$, the final state f must be $(L{\cdot}0)$. From row 4 of Table 8, the discrete model can go from state X to state L by consuming the input trace, regarless of the value of $\widehat{cout}$. If $\widehat{cin} = X$, then the discrete model can go to any state by consuming the input trace. If $\widehat{cin} = H$, then by the same argument used for the case in which $\widehat{cin} = L$, the continuous model must arrive in state $(H\cdot 0)$. Row 5 of Table 8 shows that the discrete model can do the same.

Finally, if $\widetilde{\rho}(i) = H$, the arguments are exactly symmetric to those for the cases in which $\widetilde{\rho}(i) = L$.

Thus, we have succeeded in showing that $\widehat{M}_A$ is a conservative approximation of M under $R(\Gamma(\widetilde{\sigma}), \widetilde{\rho})$.

We now show that for arbitrary traces w and $\widehat{w}$ such that $(w, \widehat{w}) \in (\widetilde{\sigma}_c \parallel \widetilde{\sigma}_i)$, and arbitrary states i and f in S_A such that $w \in L(M_A, \{i\}, \{f\})$, there exists a state $f' \in \widehat{S}_A$ such that $\widehat{w} \in L(\widehat{M}_A, \{\widetilde{\rho}(i)\}, \left\{\widehat{f}\right\}$. Let w and $\widehat{w}$ be such arbitrary traces and let i and f be such states. Let cin and $cout$ be the components of w so that $w = cin \| cout$ and let $\widehat{cin}$ and $\widehat{cout}$ be the components of the single symbol of $\widehat{w}$, so that $\widehat{w} = \left\langle(\widehat{cin} \cdot \widehat{cout})\right\rangle$. Observe from Table 8, that, with the following exceptions, there is a transition from every possible starting state on every possible input trace of length 1. The exceptions are that the discrete model cannot make a transition from state L by consuming the trace $\langle(L \cdot 1)\rangle$, nor can it make a transition from state H by consuming the trace $\langle(H \cdot 0)\rangle$. We must therefor be assured that no related traces w can be consumed by the continuous model when started from a related state.

Suppose that M_A is started in $(L \cdot 0)$, the only state that is related by $\widetilde{\rho}$ to to L. If a trace $w = (cin \parallel cout)$ is related to the trace $\widehat{w} = \langle(L \cdot 1)\rangle$ by $(\widetilde{\sigma}_c \parallel \widetilde{\sigma}_i)$, then it must have length at most 1, cin must have the constant value L, and $cout$ must attain a value greater than 0 at some point during the trace. From Table 3, we can see that only traces for which *cout* has the constant value 0 can be accepted from a starting state of $(L \cdot 0)$ while *cin* has the constant value L.

Suppose that M_A is started in $(H \cdot 0)$, the only state related by $\widetilde{\rho}$ to H. If a trace $(cin \,\|\, cout)$ is related to $\widehat{w} = \langle (H \cdot 0) \rangle$ by $(\widetilde{\sigma}_c \,\|\, \widetilde{\sigma}_i)$ then it must have length at most 1, *cin* must have the constant value H, and *cout* must attain the value 0 at some point during the trace. From Table 3, we can see that only traces for which *cout* has the constant value 1 can be accepted from a starting state of $(H \cdot 0)$ while *cin* has the constant value H.

A.3 Transliteration

We wish to show that $\widehat{\delta}_P(\widehat{M_A} \,\|\, \widehat{M_I})$ is a conservative approximation of $\delta_P(M_A \,\|\, M_I)$ under the relation $R(\Gamma(\widetilde{\sigma}_c \,\|\, \widetilde{\sigma}_o), \rho_A{\cdot}\rho_I)$. According to Theorem 7.2, we must show that every pair of traces $(w', \widehat{w}') \in (\widetilde{\sigma}_c \,\|\, \widetilde{\sigma}_o)$ satisfies the following condition.

$$\forall w \in \text{post}(\delta_P; \{w'\}) \cdot \exists \widehat{w} \in \text{post}(\widehat{\delta}_P; \{\widehat{w}'\}) \cdot (w, \widehat{w}) \in ((\widetilde{\sigma}_c \,\|\, \widetilde{\sigma}_i) \,\|\, (\widetilde{\sigma}_i \,\|\, \widetilde{\sigma}_o)) \tag{5}$$

Let $(w', \widehat{w}')$ be an arbitrary pair of traces in $(\widetilde{\sigma}_c \,\|\, \widetilde{\sigma}_o)$. By definition there are traces w'_c, w'_o, $\widehat{w}'_c$ and $\widehat{w}'_o$ such that $w' = w'_c \,\|\, w'_o$, $\widehat{w}' = \widehat{w}'_c \,\|\, \widehat{w}'_o$, $w'_c \widetilde{\sigma}_c \widehat{w}'_c$, and $w'_o \widetilde{\sigma}_o \widehat{w}'_o$. Let w be a trace in $\text{post}(\delta_P; \{w'\})$. From the definition of δ_p we know that there exists a trace $w_i \in \mathcal{R}^1$ such that $w = (w'_c \,\|\, w_i) \,\|\, (w_i \,\|\, w'_m)$. From the definition of $\widetilde{\sigma}_i$ there exists $\widehat{w}_i = \langle \int w_i \rangle$, such that $w_i \widetilde{\sigma}_i \widehat{w}_i$. Let $\widehat{w}$ be the trace $(\widehat{w}'_c \,\|\, \widehat{w}_i) \,\|\, (\widehat{w}_i \,\|\, \widehat{w}'_m)$ Clearly $\widehat{w}' \in \text{post}(\widehat{\delta}_P; \{\widehat{w}'\})$. By construction, $(w, \widehat{w}) \in ((\widetilde{\sigma}_c \,\|\, \widetilde{\sigma}_i) \,\|\, (\widetilde{\sigma}_i \,\|\, \widetilde{\sigma}_m))$ as required.

References

[1] R. Alur, C. Courcoubetis, T. A. Henzinger, and P. Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In R. Grossman, A. Nerode, R. Ravn, and H. Rischel, editors, *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 209–229. Springer-Verlag, 1993.

[2] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98(2):142–170, June 1992.

[3] Jerry R. Burch. *Trace Algebra for Automatic Verification of Real-Time Concurrent Systems.* PhD thesis, Carnegie Mellon University, Pittsburgh, PA 15213, August 1992.

[4] Edmund M. Clarke, Orna Grumberg, and David E. Long. Model checking and abstraction. In *Procedings 19th Annual ACM Symposium on Principles of Programming Languages*, January 1992.

[5] R. P. Kurshan. Analysis of discrete event coordination. In J. W. de Bakker and W. -P. de Roever and G. Rozenberg, editor, *Stepwise Refinement of Distributed Systems*, volume 430 of *Lecture Notes in Computer Science*, pages 414–453. REX Project, Springer-Verlag, May 1989.

[6] Anthony McIsaac. A formalization of abstraction in lambda. In Carl Seger and Jeffrey Joyce, editor, *HUG '93 HOL User's Group Workshop*, pages 229–240. University of British Columbia, Department of Computer Science, August 1993.

[7] Anders P. Ravn, Hans Rischel, and Kirsten Mark Hansen. Specifying and verifying requirements of real-time systems. *IEEE Transatcions on Software Engineering*, 19(1):41–55, January 1993.

[8] Derick Wood. *Theory of Computation.* John Wiley & Sons, Inc., Toronto, 1987.

[9] Ying Zhang and Alan K. Mackworth. Constraint nets: A semantic model for real-time embedded systems. Technical Report 92-10, University of British Columbia, Vancouver, B.C., Canada, October 1992.

[10] Ying Zhang and Alan K. Mackworth. Will the robot do the right thing? Technical Report 92-31, Department of Computer Science, The University of British Columbia, November 1992.