Probabilistic Constraint Nets: Constraint-Based Approach to Modeling and Verification of Hybrid Systems with Uncertainty.

Robert St-Aubin   Alan K. Mackworth

TR-2004-05

Department of Computer Science
University of British Columbia
Vancouver, BC, V6T 1Z4, CANADA
{staubin,mack}@cs.ubc.ca

April 21, 2004

Abstract

Due to the recent technological advances, real-time hybrid dynamical systems are becoming ubiquitous. Most of these systems behave unpredictably, and thus, exhibit uncertainty. Hence, a formal framework to model systems with unpredictable behaviours is needed. We develop Probabilistic Constraint Nets (PCN), a new framework that can handle a wide range of uncertainty, whether it be probabilistic, stochastic or non-deterministic. In PCN, we view probabilistic dynamical systems as online constraint-solvers for dynamic probabilistic constraints and requirements specification as global behavioural constraints on the systems. We demonstrate the power of PCN by applying it to a fully hybrid model of an elevator system which encompasses several different types of uncertainty. We present verification rules, which have been fully implemented, to perform automatic behavioural constraint verification.

1 Introduction and Motivation

The notions of constraint programming and of Constraint Satisfaction Problem (CSP) are important paradigms that have been studied extensively. Typically, they provide problem solvers, which are, more often then not, off-line solvers. Despite the advances in the field of CSP, one area that has yet to be explored thoroughly is the problem of designing dynamical systems in a constraint-based fashion. Dynamical systems cannot be handled in an off-line approach but rather should be seen as online constraint satisfying systems. One proposed solution, Constraint Nets (CN), was developed for building deterministic hybrid intelligent systems as situated agents [1]. In Cn, a deterministic dynamical system is modeled, in a unitary way, by the coupling of an agent with its environment.

Real-time dynamical systems, however, commonly behave somewhat unpredictably and thus often exhibit structured uncertainty. We have elaborated the CN framework so that we can reason about stochastic constraint-based hybrid dynamical systems. We call this new framework the *Probabilistic Constraint Nets* (PCN) model. The corresponding coupled relationship between an agent and its (uncertain) environment is shown in details in Figure 1. From this figure, one can see how the coupled agents act on, and react to, each other in a closed-loop system evolving over time. It is important to notice how the system is affected at different levels by the various types of uncertainty.

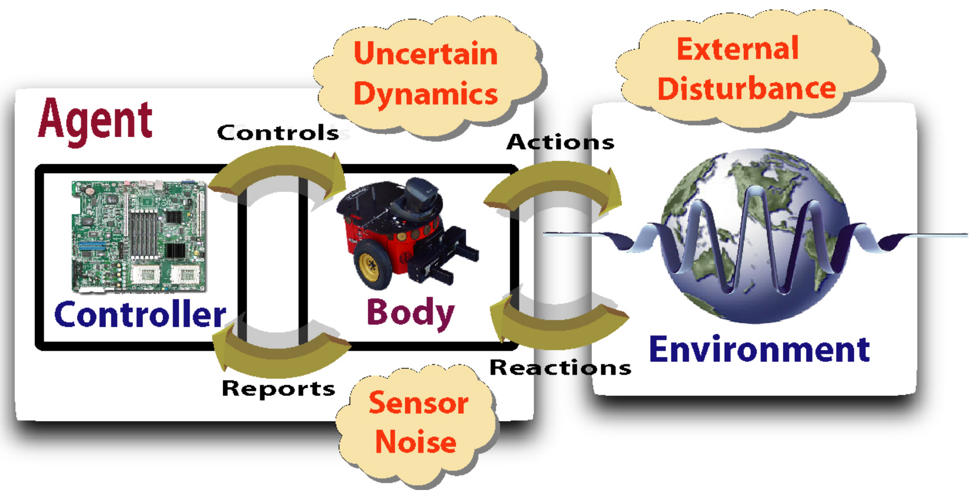


Figure 1: The structure of a constraint-based agent system

We model an agent as composed of two distinct constraint-based modules: a *body* which usually encompassing the various sensors and actuators, and a *controller*, which is the software portion that controls the behaviour of the agent. With its sensors, the agent's body senses the uncertain environment, and reports to the controller on the perceived state of the environment. In turn, the controller, now equipped with an updated status of the state of the environment, sends appropriate control signals to the actuators of the body to perform the required actions. These actions affect the state of the world, hence changing the agent's environment.

In order to handle embedded systems in an efficient way, we must move beyond the typical offline constraint satisfaction model and adopt a model where the solution to a constraint problem is a temporal trace of values, obtained via a non-anticipative transformational process, a *transduction*, of the input trace over time. Furthermore, the hybrid nature of most embedded systems necessitates that computations be performed on various time structures such as discrete, continuous or event-based. In this paper, we show how the PCN framework is suited for modeling and reasoning about such systems. Moreover, we introduce a formal behavioural constraint language, average-timed $\forall$-automata, and a set of verification rules which enable us to perform behaviour constraint satisfaction.

1.1 Modeling Uncertainty

So far, we have not directly addressed the importance of modeling a systems's uncertainty with a constraint-based approach. In fact, one could go as far as asking whether one should even worry about uncertainty within systems. Since no matter how well uncertainty is modeled, one will never capture its exact nature, and thus, one should simply avoid modeling it all together. However, many factors, such as industrial production variations, incomplete information or even numerical precision errors lead, to the conclusion that uncertainty is inherently present in any *real* physical system. Therefore, avoiding to model the effects of the various sources of uncertainty can have a drastic influence on the system's behaviour. Although the uncertainty model might not reflect the *exact* nature of the system under study, completely ignoring it from the modeling task can be even more disastrous. For instance, when designing an airplane controller, it is crucial to take the uncertainty of the wind speed into account as a given controller might be optimal for a certain wind condition but might behave dangerously for other wind conditions.

As another example, consider a simulation of the behaviour of an autonomous robotic system traveling to a series of goal locations, as shown in Figure 2. In the situation where the uncertainty in the odometry of the robot is ignored, one can see that whether or not a "reset" function of the belief state is used to relocate the robot when it has reached an intermediate goal, the robot's belief of its location will diverge from its true position as time goes by, as it is increasingly affected by the odometric uncertainty. It is clear from this example that for the robot to "know" where it is and eventually localize itself, a model of uncertainty is needed.

As shown in Figure 1, we consider three common ways in which the uncertainty can enter a system:

1. **Uncertainty in the Dynamics:** Uncertainty can enter a system through the actual model of the physical system. High-order and nonlinear dynamics can be unknown or deliberately ignored in order to keep the model simple and tractable. This lack of knowledge of the *true* underlying dynamics introduces some uncertainty in the model, thus rendering the behaviour somewhat unpredictable.

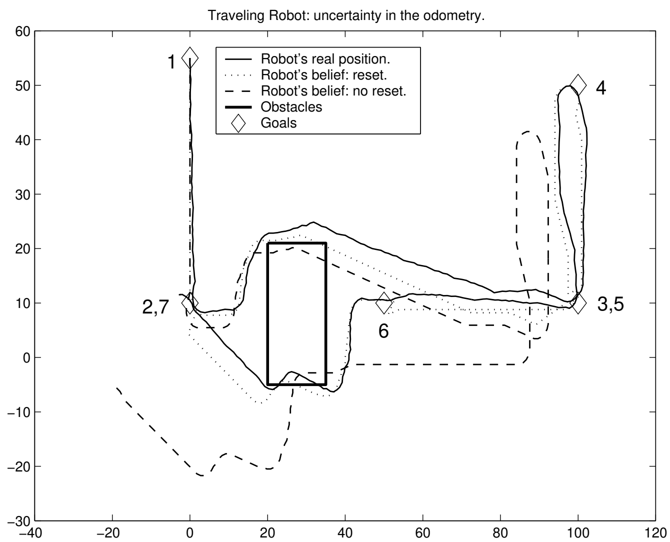


Figure 2: Autonomous robot chasing series of targets. Effect of uncertainty in the odometry.

Let us revisit the previous mobile robot example of Figure 2. Consider for instance that some nonlinear effects in the actuators of the robot need to be modeled in order to precisely capture their dynamics. However, to simplify the model, one might want to omit these effects. Yet, omitting to account for the uncertainty introduced in the dynamics by the *simpler* model might lead to imperfect movement of the robot. As a result, the robot might have a false belief of where it is, even though it reaches the intermediate goals.

2. **External Disturbances:** Disturbances (not attributed to the inaccuracy of the model of the agent itself), can also enter in the dynamics of the plant. Such uncertainty is caused by external disturbances that are outside of the control of the system. E.g., barometric pressure or wind speed are elements outside of the plant's dynamics which can influence the behaviour of an airplane system.

3. **Sensor Noise:** Referred to as "noise" since it is usually modeled via Gaussian white noise (Brownian motion), this type of uncertainty enters the system through imprecise sensor measurements. For example, sensor noise could come from the sonar readings of a mobile robot.

2 Introduction of the PCN Framework

2.1 Overview of Constraint Nets

Before going into the description of the syntax and semantics of the Probabilistic Constraint Net model, we wish to provide an introduction to the framework on which PCN is based: Constraint Nets (CN). Developed by Zhang and Mackworth [1], the CN modeling framework is built on a topological view of time and domain structures along with notions of traces, events and transductions. These definitions, which we present below for sake of clarity, represent the foundation for the notion of time and domain in the Constraint Net framework. We retain these concepts and extend the CN framework to build the Probabilistic Constraint Net model. We refer the reader to Chapter 3: *Topological Structure of Dynamics*, from [2], for a thorough introduction of those concepts and more concerning the CN modeling framework.

Within (P)CN, we view time and domains as fully abstract concepts, defined as follows:

- *Time* is presented as a linearly ordered set $\langle \mathcal{T}, \leq \rangle$ with a least element t_0, and a metric d on the set $\mathcal{T}$. With this abstract representation of time, one can model discrete, continuous or event-based systems.
- Simple and composite domains denote, respectively, simple data types (e.g. reals, integers, Boolean and characters) and structured data types (e.g. arrays, vectors or objects). A *simple* domain is represented as a pair $\langle A \cup \{\perp_A\}, d_A \rangle$ where A is a set, $\perp_A \notin A$ means undefined in A, and d_A is a metric on A. *Composite* domains are obtained by the product of simple domains. Obviously, with this abstract notion, domains can be numerical or symbolic, discrete or continuous.

We also view the history of a dynamical system as a set of traces, where the trace of each variable[1] is defined as follow:

- *Traces* can intuitively be perceived as changes of values over time. Formally, a mapping $v : \mathcal{T} \to A$ from time $\mathcal{T}$ to domain A is called a *trace*. An *event* trace is a trace with a Boolean domain. An event in an event trace is a transition from 0 to 1 or from 1 to 0.

It is important to note that for a deterministic system, the initial value of a trace completely defines the whole trace. Indeed, given the (deterministic) dynamics of a system, with the initial value of each variable, one can completely predict the value of each variable over the time history of the system. In the presence of uncertainty, however, this fails to be true. Given the model of the uncertain dynamics of a system and its initial values, one can only predict the possible trajectories, i.e., the set of all traces that the system can take. Therefore, one has to be extremely careful when designing systems exhibiting uncertainty as although some trajectories might be acceptable for a system, chances are that some undesired trajectory can be achieved with the current

[1] From now on, we will refer to variables and *locations* interchangeably.

conditions. Thus, controller design and behaviour verification becomes not only very important but also much more complicated. We will return to these issues in the later sections of this paper.

In the CN framework, functions are referred to as transductions. Formally, these are defined as:

- *Transductions* are causal mappings from inputs to outputs over time, either operating according to a certain reference time or activated by external events. Note that in PCN, the notion of *causal* mapping will be replaced by *adapted* mapping, its measure theoretic equivalent. However, both concepts have a similar intuitive meaning. The class of simple transductions contains *transliterations* and *delays*. A transliteration is a pointwise extension of a function. It can be seen as a transformational process without internal state (memory). A (unit) delay is a process where the output value at time t is the input value at time $t-1$. Intuitively, (unit) delays can be seen as a unit memory.

2.2 Probabilistic Constraint Nets

We developed a model for building probabilistic hybrid systems as situated agents. As our framework extend the CN framework, we refer the reader to [1] for a thorough introduction to the basic definitions and concepts which we will avoid here for economy of space.

Definition 2.1 (Probabilistic Constraint Nets) *A probabilistic constraint net $\mathcal{P}$ is a tuple $\langle Lc, Tp, Td, Cn \rangle$, where Lc is a finite set of locations, each associated with a sort; Td is a finite set of labels of transductions, each with an output port and a set of input ports, and each port is associated with a sort; Tp is a finite set of labels of probabilistic transductions, each with an output port and a set of input ports (parameters for standard probability distributions), and each port is associated with a sort. Every probabilistic transduction is associated with a given probability distribution, either discrete or continuous, thus the sort of the output of a probabilistic transduction is the sort of its probability distribution. Cn is a set of connections between locations and ports of the same sort, with the restrictions that (1) no location is isolated, (2) there is at most one output port connected to each location, (3) each port of a transduction connects to a unique location.*

To explain the notion of transductions, we need to introduce the concept of a *trace*, which intuitively, denotes changes of values over time. Formally, a trace is a mapping $v : \mathcal{T} \times \Omega \to A$ from time $\mathcal{T}$ and event space Ω to a domain A. In practice, one does not need to take into account the event space when observing a trace. Therefore, when no ambiguity arises, we will shorten our definition of a trace to $v : \mathcal{T} \to X$. A transduction is a mapping from input traces to an output trace that satisfies the non-anticipative causal relationship between its inputs and output.

Probabilistic transductions act in some ways like random number generators, obeying a given probability distribution. Thus, in practice, probabilistic transductions can be represented as discrete (e.g. Poisson or uniform) or continuous (Gaussian or exponential) probability distributions.

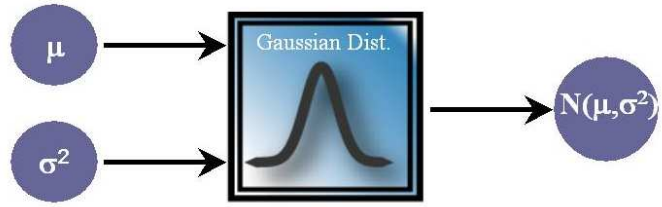


Figure 3: Gaussian probability distribution as a Probabilistic Transduction.

It can easily be seen that a probabilistic transduction induces a *random* variable as its output location. Therefore, one might wonder if there is a difference between a location which is the output of a deterministic transduction and one that is the output of a probabilistic transduction. The difference is subtle as it can be seen that, once feedback is introduced, every location is in fact a random variable and thus should be seen as a *probabilistic* location. We will, however, not make the distinction between probabilistic and deterministic locations but rather keep the distinction at the transduction level.

The Probabilistic Constraint Net framework is developed to model probabilistic systems which cannot adequately be described by deterministic frameworks. Moreover, most existing probabilistic modeling paradigms can only efficiently model a small subset of the types of uncertain systems that arise in practice. The PCN framework allows for a complete hybrid modeling approach, where one can model time and domains as either discrete, continuous or both, and uncertainty as taking shape in many different ways: probabilistic like in a Markov Chain, or stochastic as with Brownian Motion and stochastic differential equations. The flexibility of our framework is a great asset as it allows a designer to model a complex system under the umbrella of a single modeling language. Another advantage of our framework is its graphical representation. A PCN can be represented by a bipartite graph where locations are depicted by circles,transductions by boxes and connections by arcs. In addition, we depict probabilistic transductions with double boxes to differentiate them from deterministic transductions. Consider for example the task of modeling a Gaussian probability distribution in PCN. To do so, one would need to specify the value of μ (the mean) and σ^2 (the variance), the two inputs of the Probabilistic Transduction as shown in Figure 3. Moreover, Figure 4 displays a closed probabilistic constraint net representing the correct mathematical interpretation of a generic stochastic differential equation (also known as an *Itô process*):

$$X_t = X_{t_0} + \int_{t_0}^{t} f(s, X_s)ds + \oint_{t_0}^{t} g(s, X_s)dW_s \tag{1}$$

Note that the second integral $\oint$ in this equation is the *Itô Stochastic integral* which differs from the usual *Riemann* integral. Semantically, a probabilistic constraint net $\mathcal{P}$ denotes a set of equations $\mathbf{o} = \mathbf{F}(\mathbf{i}, \mathbf{o})$ where each function $\mathbf{F}$ corresponds to a transduction in $\mathcal{P}$. The semantics of $\mathcal{P}$ is a "solution" of the set of equations. Formally, we

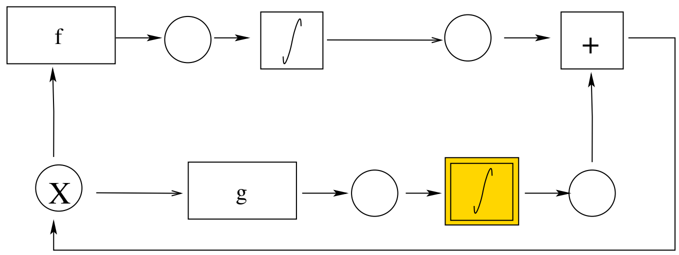


Figure 4: PCN model of an Itô process.

can show that the solutions are stochastic processes, usually Markov processes, and in many cases, diffusion processes. We view the constraint satisfaction problem as a dynamic process that approaches the solution set of the given constraints asymptotically (in the stochastic sense). The semantics of PCN differ significantly from CN, which is based on the fixed point solution of a set of equations. See [3] for the relation between stable attractors and solutions of constraints.

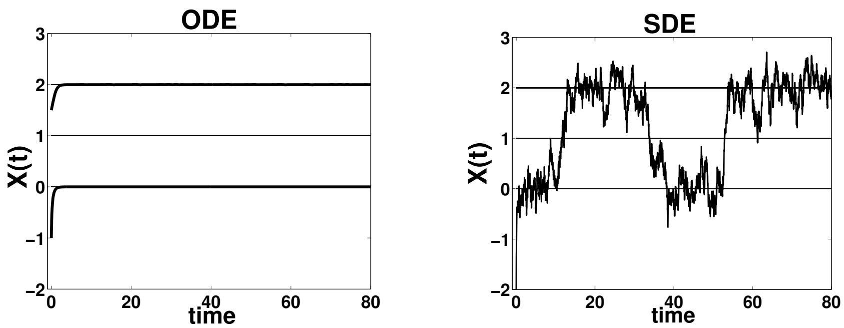
To illustrate this, in Figure 5 we compare, two dynamical systems with nominal component

$$\dot{X}_t = -X_t * (X_t - 1) * (X_t - 2).$$

The first one is deterministic and has two distinct stable attractors (equilibria), at 2 and at 0, as shown in Figure 5(a). The second is affected by a Brownian Motion process. For the this system, as shown in Figure 5(b), a starting value at $X_0 = -2$, would cause the system to be attracted toward the closest equilibrium which is at $X = 0$. The system fluctuates around this attractor, reacting under the influence of the Brownian motion component and, around time $t = 12$, a large enough noise disturbance pushes the system over the value of 1, causing the system to be attracted toward the other equilibrium, at $X = 2$. Another influx of noise flips the system back to the lower equilibrium at $t = 35$ and so on. This example shows the effect of uncertainty on the system and on the solution of a set of constraints.

3 The Dynamics of An Elevator System with Uncertainty

To demonstrate the scope of our approach, we will analyze an elevator system exhibiting uncertainty. Elevator systems constitute a useful testbed for hybrid systems, as confirmed by the fact that they have been used as a benchmark for various approaches to real-time systems [4, 5, 6, 7]. Nevertheless, most previous approaches focus on discrete deterministic dynamics. Combining continuous Newtonian dynamics and discrete control from users' requests, we extend the benchmark example in [4] to account for the different types of uncertainty that can arise in a real physical elevator system. We model the different levels of the system (continuous motion, discrete controller)



(a) (b)

Figure 5: (a) ODE: $\dot{X}_t = -X_t(X_t - 1) \quad (X_t - 2)$; $X_0 = -1$ and $X_0 = 1.5$
(b) SDE: $\dot{X}_t = -X_t(X_t - 1) \quad (X_t - 2) + N_t$; $X_0 = -2$

and provide a methodology for verifying the behaviour of the system with respect to a behavioural constraint of real-time response. We refer the reader to [4] for a complete description and analysis of the deterministic hybrid version of this example.

In Figure 6, we show the PCN model of the elevator body as represented by a second order stochastic differential equation, based on Newton's second law:

$$\ddot{h} = k(t)\dot{h} + F + w(t) \tag{2}$$

where F is the motor force (control input), $k(t)$ is the coefficient of friction and h is the height of the elevator. Moreover, the model is augmented two types of uncertainty: (1) uncertainty in the dynamics through variation of the friction coefficient $k(t)$, and (2) a time-varying external disturbance force $w(t)$ acting on the elevator. We assume that $k(t)$ has a nominal value of $k_0 = 1.05$ and that it can vary with $k(t) \in [0.70, 1.40]$, modeled as a Gaussian White Noise process with zero mean and standard deviation $\sigma = 0.15$. With these parameters, we have $Pr(|k(t) - k_0| \leq 0.35) \geq 0.9804$; that is, the value of the friction coefficient may exceed the presumed bounds on the uncertainty, but with a small non-zero probability. This phenomenon indicates a "soft" norm constraint on the uncertainty.

We will later augment the system with probabilistic passenger arrivals, and will then show that the system satisfies the constraint of serving passengers within fixed bounded time on average.

4 Behavioural Constraints

The online satisfaction of the constraints of the dynamics of a the system ensure that its behaviour will be according to the stochastic process that corresponds to its solution. However, such constraint satisfaction does not guarantee that the behaviour of the system will satisfy global behavioural constraints. For example, consider the dynamics of our elevator system. Equation 2 represents the constraints on the dynamics, essentially

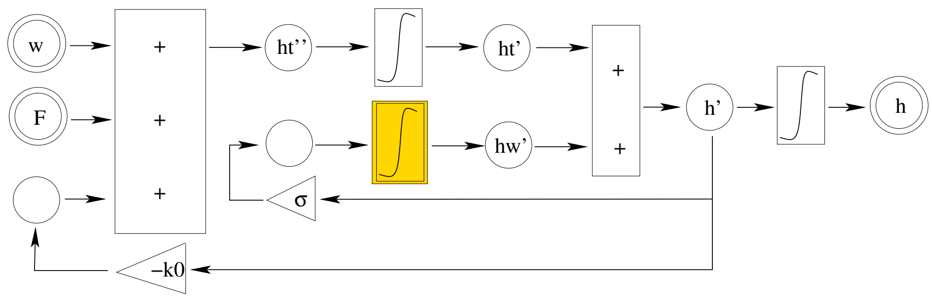


Figure 6: The $BODY$ module of the elevator system

stating that the elevator must move obeying laws of Newtonian motion. Although this equation can represent very accurately the local behaviour of the system, it does not preclude the elevator from stopping halfway down a floor nor does it guarantee that a service request will always be successful in timely manner. Such restrictions are global constraints on the behaviour of the system and cannot easily be represented within the PCN framework. We will thus introduce a specification language, *average-timed* $\forall$-*automata*, that will allow us to represent behavioural constraints of a given system. Together, these two languages will specify the whole set of constraints on a stochastic dynamical system.

For the scope of this paper, we will focus on a class of dynamical systems that we call *stochastic dynamical systems* (SDS).

4.1 Stochastic Dynamical Systems

Definition 4.1 (Stochastic Dynamical System (SDS)) *A stochastic dynamical system over the metric space* (X, d) *and defined on a complete probability space* $(\Omega, \mathcal{F}, \mathcal{F}_t, \mathbb{P})$*, is a mapping*

$$\phi : \mathcal{T} \times \Omega \times X \to X,$$

with the following properties:

1. $\boldsymbol{\mathcal{F}_t}$ **-adapted:**[2] *if* $\phi(t, \omega)$ *is* $\mathcal{F}_t$*-measurable* $\forall t \in \mathcal{T}$.
2. **Measurability:** ϕ *is* $\mathcal{B}(\mathcal{T}) \otimes \mathcal{F} \otimes \mathcal{B}$, $\mathcal{B}$*-measurable.*
3. **Cocycle property (Chapman-Kolmogorov equations):**

$$\phi(0, \omega, x) = x, \forall x \in X$$

$$\phi(t + s, \omega, x) = \phi(t, \omega, \phi(s, \omega, x)), \forall s, t \in \mathcal{T}, \omega \in \Omega.$$

With every stochastic dynamical system ϕ, we can associate a probability distribution which we will call an *evolution kernel*.

[2]We will also refer to this property as non-anticipative

Definition 4.2 (Discrete-time Evolution Kernel) *An evolution kernel on a discrete time structure $\mathcal{T}$ is a function $K : \mathcal{T} \times X^k \times \mathcal{B}(X)$ such that:*

- $\forall x \in X^k, K(\cdot, x, \cdot)$ *is a probability measure*
- $\forall B \in \mathcal{B}(X), K(\cdot, \cdot, B)$ *is a measurable function.*

In the remainder of this paper we will focus our attention toward time-invariant and Markovian systems. We briefly introduce the definition of these concepts.

Proposition 4.1 (Time-Invariance) *The behaviour of a SDS is time invariant if its associated evolution kernel K is stationary, that is, if the condition*

$$K(t + u, x, B) = K(t, x, B), k \leq t$$

is satisfied.

Proof (Time-Invariance): We can assume $k = 1$ without loss of generality. This equality implies that $Pr(X_{t+u+1} \in B|X_{t+u} = x) = Pr(X_{t+1} \in B|X_t = x)$ for $t_0 \leq t < t + 1 < \infty$ and $t_0 \leq t + u < t + 1 + u < \infty$. In this case, the evolution kernel is a function only of x, and B (and 1). Hence we can write it in the form

$$K(x, B) = Pr(X_{t+1} \in B|X_t = x), 0 \leq t < \infty$$

which is equivalent to the probability of evolution from state x to B regardless of the position of the system in time. $\Box$

Proposition 4.2 (Markovian Property) *The Markovian property of order* 1 *is also equivalent to the evolution kernel being a conditional distribution of only the previous state, i.e., $K : \mathcal{T} \times X^k \times \mathcal{B}(X)$, with $k = 1$.*

Proof (Markovian Property): Trivial. $\Box$

Before formally discussing the notion of behavioural constraint verification, it is necessary to discuss the relationship between stochastic dynamical systems and their behaviours. Intuitively, the behaviour of a stochastic dynamical system is the set of observable input/output traces of a given system. Let $\mathcal{P}(I, O)$ be a PCN module, where (I, O) is the tuple of input and output locations of the module. Formally, an input/output pair (i, o) is an *observable trace* of $\mathcal{P}(I, O)$ iff $\exists F \in [\![\mathcal{P}(I, O)]\!]$ such that $o = F(i)$. The reader should note that, within the PCN framework, the function F can be deterministic, non-deterministic or probabilistic, depending on what type of transductions (deterministic or probabilistic) and locations (hidden or not) are used to model the system. We define the *behaviour* of $\mathcal{P}(I, O)$ as the set of all observable traces and we denote it as $[\![\mathcal{P}(I, O)]\!]$. We will abbreviate $[\![\mathcal{P}(I, O)]\!]$ to $[\![\mathcal{P}]\!]$ if $I = I(\mathcal{P})$, $O = O(\mathcal{P})$ and no ambiguity arises.

The notion of equivalency of PCN modules stems directly from their behaviour. Two PCN modules, $\mathcal{P}_1$ and $\mathcal{P}_2$, are *equivalent*, denoted $\mathcal{P}_1 \simeq \mathcal{P}_2$, iff they exhibit the same behaviour: $[\![\mathcal{P}_1]\!] = [\![\mathcal{P}_2]\!]$.

Now that we have defined formally what we mean by the behaviour of a system, we introduce the definitions of time-invariant and Markovian behaviours.

Definition 4.3 (Time-Invariant) *Let $\mathcal{B} = \{v | v : \mathcal{T} \to A\}$ be a behaviour. $\mathcal{B}$ is a* time-invariant *behaviour*[3] *if for any $a_1, a_2 \in A$, $\forall v \in \mathcal{B}$ such that $v(t_1) = a_1$ and $v(t_1 + s) = a_1$ then $Pr(v(t_2) = a_2) = Pr(v(t_2 + s) = a_2)$ for $t_1 < t_2 \in \mathcal{T}$.*

Definition 4.4 (Markov Property) *Let $\mathcal{B}$ be a behaviour. Given any time point $t \in \mathcal{T}$, $v \in \mathcal{B}$ and $a \in A$, such that $v(t) = a$, the behaviour is called Markovian (of order 1) if the probability distribution of the next state, $v(t+1)$, is independent of the past history of the system except for the state $v(t) = a$. A behaviour is* Markovian *of order n if the probability of the next state depends only on the n most recent states.*

The class of discrete time stochastic dynamical systems exhibiting a time-invariant Markovian behaviour is very vast. In fact, any discrete-time, time invariant Markovian stochastic dynamical system corresponds to what we call a stochastic transition system.

A *stochastic state transition* system is a tuple $\langle \mathcal{S}, \mathbb{P}, \Theta \rangle$ where $\mathcal{S}$ is a set of states, $\mathbb{P} : \mathcal{S} \times \mathcal{S}$ is an evolution kernel representing the transition probability distribution between two states, i.e., $\mathbb{P}(s_1, s_2)$ is the probability of a transition occurring between $s_1, s_2 \in \mathcal{S}$ and Θ represents the distribution of the initial state of the system. Notice that due to the time invariance and Markovian properties, $\mathbb{P}$ is independent of the time parameter and transitions only depend on the current state. For any discrete time $\mathcal{T}$, $v : \mathcal{T} \to \mathcal{S}$ is a trace of $\langle \mathcal{S}, \mathbb{P}, \Theta \rangle$ iff $\forall t > 0, \mathbb{P}(v(pre(t)), v(t)) > 0$, where $pre(t)$ represents the time value preceding t. We will denote an allowed transition from $v(pre(t))$ to $v(t)$ by $v(pre(t)) \rightsquigarrow v(t)$. A behaviour $\mathcal{B}$ corresponds to a stochastic state transition system $\langle \mathcal{S}, \mathbb{P}, \Theta \rangle$ iff $\mathcal{B}$ is equal to the set of all traces of $\langle \mathcal{S}, \mathbb{P}, \Theta \rangle$. Stochastic state transition systems constitute a compact representation of time invariant Markovian behaviours.

Let us now introduce the notion of *probability measure* on a behaviour. To do so, we introduce the notion of a Borel space on traces of a system, which follows [8, 9]. Let $\mathcal{B}$ be the behaviour of a stochastic state transition system $STS = \langle \mathcal{S}, \mathbb{P}, \Theta \rangle$. A distribution Θ on the initial state of STS induces a probability measure μ_Θ on its traces in the following way. First let $s_0, s_1, \ldots, s_n \in \mathcal{S}$ with $s_i \rightsquigarrow s_{i+1}, (0 \leq i < n)$. Then denote $C(s_0, s_1, \ldots, s_n)$ by the *cylinder* set of all traces $v \in \mathcal{B}$ such that $v(0) = s_0, \ldots, v(n) = s_n$. Define $\mathcal{F}(\mathcal{B})$ to be the smallest σ-algebra on the behaviour $\mathcal{B}$ which contains all cylinder sets $C(s_0, \ldots, s_n)$. The probability measure μ_Θ on $\mathcal{F}(\mathcal{B})$ is the unique measure defined by induction on n with base case $n = 0 : \mu_\Theta(C(s_0)) = \Theta(s_0)$ and induction hypothesis for $n \geq 0$:

$$\mu_\Theta(C(s_0, \ldots, s_n, s_{n+1})) = \mu_\Theta(C(s_0, \ldots, s_n)) \cdot \mathbb{P}(s_n, s_{n+1})$$

We define a behavioural constraint (or requirements specification) $\mathcal{B}_\mathcal{C}$ for a stochastic system $STS = \langle \mathcal{S}, \mathbb{P}, \Theta \rangle$ as a set of allowable input/output traces of the system, i.e., $\mathcal{B}_C \subset \times_{I \cup O} A_l^{\mathcal{T}}$. Let $\mathcal{B} = [\![STS]\!]$ be the behaviour of STS. We say that STS satisfies the behavioural constraints $\mathcal{B}_C$ at a level α, denoted by $\mathcal{B} \models_\alpha \mathcal{B}_C$ iff $\mu(\mathcal{B} \cap \mathcal{B}_C) = \mu(\{v \in \mathcal{B} | v \models \mathcal{B}_C\}) \geq \alpha$, where $v \models \mathcal{B}_C$ is the predicate for v satisfying the behavioural constraint $\mathcal{B}_C$. Perfect satisfaction of behavioural constraints is indicated by $\alpha = 1$, which means that **all** traces of $\mathcal{B}$ are allowable traces. Satisfaction at a level of $\alpha < 1$ results in a subset of the traces of $\mathcal{B}$ being undesirable.

[3]This is sometimes also referred to as time homogeneous behaviour.

Obviously, a satisfaction at level $\alpha = 0$ is equivalent to a total absence of satisfaction, i.e., **none** of the possible traces of the system will ever satisfy the constraint $\mathcal{B}_C$ ($\mu = 0 \Rightarrow (\mathcal{B} \cap \mathcal{B}_C) = \emptyset$). Requirements satisfaction of deterministic system is equivalent to requirement satisfaction at level $\alpha = 1$. However, in the presence of probabilistic behaviour, requiring satisfaction at level $\alpha = 1$ might be too strict as one might be willing to accept a small risk of not satisfying the requirement (e.g., $\alpha = 0.95$), rather then flat out rejecting the system. Note that for the scope of this paper, we will elaborate a method which performs behavioural constraint satisfaction at level $\alpha = 1$ for untimed constraints while temporal behavioural constraints will be satisfied on average. We are currently working on methods for behavioural specification at any level $\alpha < 1$.

Now that we formally defined the notion of requirements specification at level α, we can now introduce the notion of robustness and complexity of systems. The *robustness* of a system is a notion defined on parameterized probabilistic constraint nets. We say that a parameterized system $\mathcal{P}_1^P$ is *less robust* than a second system $\mathcal{P}_2^P$ with respect to a behavioural constraint $\mathcal{B}_C$, denoted by $\mathcal{P}_1^P \preceq_{\mathcal{B}_C} \mathcal{P}_2^P$, iff $\forall p \in \times_P D_P, [\![\mathcal{P}_1^P]\!](p) \models_\alpha \mathcal{B}_C \Rightarrow [\![\mathcal{P}_2^P]\!](p) \models_\beta \mathcal{B}_C$, for $\alpha \leq \beta$. The two systems above are equivalent w.r.t. $\mathcal{B}_C$, written by $\mathcal{P}_1^P \simeq_{\mathcal{B}_C} \mathcal{P}_2^P$, iff $\mathcal{P}_1^P \preceq_{\mathcal{B}_C} \mathcal{P}_2^P$ and $\mathcal{P}_2^P \preceq_{\mathcal{B}_C} \mathcal{P}_1^P$. In this case, both systems would satisfy the behavioural constraint at level $\alpha = \beta$. Note that this definition of equivalence is somewhat more subtle than for deterministic systems. Equivalence of deterministic systems requires that the behaviours of the two systems be the same. This implies that the traces of the two systems are exactly the same. For probabilistic systems, we relax this assumptions by only requiring that the measure over allowable traces be the same for both systems. However, it is easy to construct two equivalent systems for which their respective set of allowable traces are different, even though both sets have equal measure.

Let us now define the complexity of a behaviour. Behavioural complexity is defined with respect to a given measurement on the size of a stochastic dynamical system. This measurement could be the number of transductions, the number of delays or the maximum number of delay element in any path (which is equivalent to the order of the Markovian property of the system). Therefore, given a measurement κ, denote $|\mathcal{P}|_\kappa$ as the size of the system $\mathcal{P}$ with respect to κ. We then define the complexity of the behaviours satisfying the requirements specification $\mathcal{B}_C$, w.r.t. κ and level α, written $|\mathcal{B}_C|_\kappa^\alpha$, to be the smallest stochastic dynamical system which respects $\mathcal{B}_C$. That is, $|\mathcal{B}_C|_\kappa^\alpha = min\{|\mathcal{P}|_\kappa\}_{[\![\mathcal{P}]\!] \models_\alpha \mathcal{B}_C}$.

4.2 $\forall$-Automata

A popular method for representing behavioural constraints of systems is automata. Such method is also very well suited for the PCN framework as we can view traces as a generalization of infinite sequences. A desired property of the systems (hence the traces) can be specified by an automaton. That is, a trace of a system would satisfy the behavioural constraints iff the associated automaton accepts the trace.

Manna and Pnueli [10] first proposed $\forall$-automata and applied it to the specification and verification of concurrent programs. An extension to $\forall$-automata, *timed* $\forall$-automata, was proposed in [2] and applied in the context of behaviour verification of dynamical hybrid systems. We augment the notion of $\forall$-automata behaviour verification to *average-timed* $\forall$-automata for stochastic dynamical systems.

Definition 4.5 (Syntax of $\forall$-automata) *A* $\forall$-automaton $\mathcal{A}$ *is a quintuple* $\langle Q, R, S, e, c\rangle$ *where* Q *is a finite set of* automaton states*,* $R \subseteq Q$ *is a set of* recurrent states *and* $S \subseteq Q$ *is a set of* stable states*. With each* $q \in Q$*, we associate a state proposition* $e(q)$*, which characterizes the* entry condition *under which the automaton may start its activity in* q*. With each pair* $q, q' \in Q$*, we associate a state proposition* $c(q, q')$*, which characterizes the* transition condition *under which the automaton may move from* q *to* q'*.*

R and S are the generalization of *accepting* states to the case of infinite inputs. We denote by $B = Q - (R \cup S)$ the set of *non-accepting (bad)* states.

Let $\mathcal{T}$ be a discrete time structure, A be a domain and $v : \mathcal{T} \rightarrow A$ be a trace. A *run* of $\mathcal{A}$ over v is a mapping $r : \mathcal{T} \rightarrow Q$ such that (1) $v(\mathbf{0}) \models e(r(\mathbf{0}))$; and (2) for all $t > \mathbf{0}$, $v(t) \models c(r(pre(t)), r(t))$. Obviously, any complete automaton guarantees that any discrete time trace has a run over it.

If r is a run then let $Inf(r)$ denote the set of automaton states which appears infinitely often in r. That is, $Inf(r) = \{q | \forall t, \exists t_0 \geq t, r(t_q 0) = q\}$. If $\mathcal{T}$ has a greatest element t_0 then we define $Inf(r) = \{r(t_0)\}$. Therefore, $Inf(r)$ can be seen as a generalization of the "final value" of a system.

Let $\mathcal{A}$ be a $\forall$-automaton. A run r over $\mathcal{A}$ is defined to be *accepting* iff it satisfies one of the two conditions:

1. $Inf(r) \cap R \neq \emptyset$, i.e., *some* of the states appearing infinitely many times in r belong to R, or
2. $Inf(r) \subseteq S$, i.e., *all* the states appearing infinitely many times in r belong to S.

Essentially, the notion of acceptance of traces states that in the long run, the system either always returns to the set of recurrent states R or it will remain forever within the stable set S. Systems which cannot guarantee this are deemed unsatisfactory for the requirements specification. Based on this requirement, the semantics of $\forall$-automata follow:

Definition 4.6 (Semantics of $\forall$-automata) *A* $\forall$*-automaton* $\mathcal{A}$ accepts *a trace* v*, written* $v \models \mathcal{A}$*, iff* all *possible runs of* $\mathcal{A}$ *over* v *are accepting.*

One should note that these semantics differ in the way it handles non-determinism from the semantics of conventional automata, with which the reader might be more accustomed to. A conventional automata $\mathcal{C}$, which could in this context also be called a $\exists$-automata, accepts a language if there exists at least one run over $\mathcal{C}$ which is accepting. However, in the context of behaviour verification, having at least one trace satisfying the requirements is obviously not a strong enough statement as the accepted system could have one and only one trace which satisfies the requirements. In the case of a safety requirement, this is generally not what we define as a *safe* system.

As previously mentioned, we will, in this work, focus solely on requirements at level $\alpha = 1$ for untimed behaviour while we will introduce the notion of "average" satisfaction for behavioural constraints containing a temporal component. Intuitively,

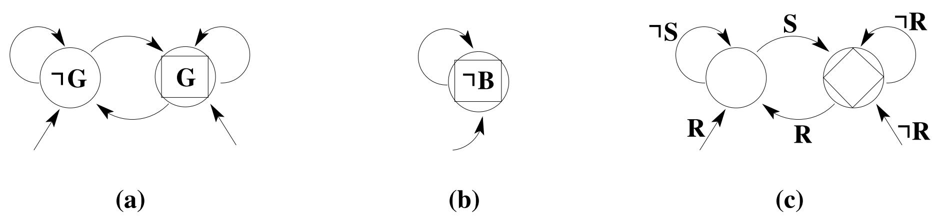


Figure 7: ∀-Automaton Specifications (a) goal achievement (b) safety (c) bounded response

satisfaction at level $\alpha = 1$ of a behavioural constraint amounts to the system not possessing any *absorbing* bad states. In practice, for a system to be without any absorbing bad states requires that for any state of the stochastic dynamical system associated with a bad automaton state, there must exist a path with positive probability which leads to an accepting state (associated to either R or S). We will return to this observation when introducing the verification rules later in this section.

For economy of space, we leave the remaining details of ∀-automata for the reader to look up. We refer the reader to Chapter 10 of [2] for a comprehensive introduction to behaviour verification with ∀-automata and .

Let us now present some common requirements specification of dynamical systems. Figure 7(a) represents a constraint which is satisfied if the traces of a systems which eventually will always satisfy the goal condition G are all accepted. Figure 7(b) is a safety constraint which states that an acceptable system should never satisfy the *BAD* condition B. Finally, Figure 7(c) is a bounded response constraint. It states that whenever event R occurs, the response S will occur in bounded time.

4.3 Average-timed automata

Meaningful behavioural constraints often encompass temporal constraints. Consider Figure 7(c), where one might be interested in a system satisfying a bounded response specification where the time bound is a known finite constant. In order to represent timeliness of behavioural constraints, timed ∀-automata was proposed [11]. Timed ∀-automata augments basic ∀-automata with timed automaton states and time bounds. This approach, however, is ill-suited for stochastic dynamical systems. Indeed, since we are interested in solving behavioural constraints on stochastic systems, we cannot talk about satisfying a given time constraint in an absolute way but rather we need to reason about satisfying that time constraint on *average*. We mentioned earlier that in order for a system to be accepting by a ∀-automata specification, it needs not have any absorbing bad states. This is characterized by the limiting behaviour: $lim_{t\to\infty} Pr(X_t \in S \cup R | X_0 \in B) = 1$. Although this requirement is sufficient to guarantee that for any run r, $Inf(r) \cap R \neq \emptyset$ or that $Inf(r) \subseteq S$, it does not guarantee that it will happen in a finite time for every run. However, for systems without any absorbing bad state, we are assured that the *average* time will be bounded, as stated in the proposition below. We prove the result for finite state space, but the result can be extended to countable state space at the price of a more complicated proof.

Proposition 4.3 *Assume a finite state space $\mathcal{S}$, and assume that, for all the bad states B, there is a positive probability of moving toward an accepting state R or S, i.e., the set of bad states is irreducible. Define $\zeta_s^{s'}$ as the time needed to reach state s' from state s. Then, $\mathbb{E}(\zeta_b^s | b \in B, s \in R \cup S) < \infty$.*

Proof : Let $p > 0$ be the smallest probability with which a bad state $b \in B$ moves toward an accepting state. Due to the irreducibility of B, we are guaranteed that $p > 0$ exists. We can show that the measure of paths starting in a bad state and never reaching the set of accepting states is $lim_{t\to\infty}(1-p)^t = 0$. Hence, any path which never reaches the set of accepting states has measure 0. Let us now prove that $\mathbb{E}(\zeta_s^{s'} | s \in B, s' \in R \cup S) < \infty$. Assume that state s is located at n transitions away from s', the closest $R \cup S$-state. Hence,

$$\begin{array}{ll}
\mathbb{E}(\zeta_s^{s'}) & = \sum_{i=n}^{\infty} ip^n(1-p)^{i-n} \\
& = \frac{p^n}{(1-p)^{n-1}} \sum_{i=n}^{\infty} i(1-p)^{i-1} \\
& = \frac{p^n}{(1-p)^{n-1}} \frac{\partial}{\partial p}(\sum_{i=n}^{\infty}(1-p)^i) \\
& \leq \frac{p^n}{(1-p)^{n-1}} \frac{\partial}{\partial p}(\sum_{i=0}^{\infty}(1-p)^i) \\
& = \frac{p^n}{(1-p)^{n-1}} \frac{\partial}{\partial p}(\frac{1}{1-(1-p)}) \\
& = \frac{p^n}{(1-p)^{n-1}} \frac{1}{(1-(1-p))^2} \\
& < \infty
\end{array}$$

□

A logical extension of time constraints is *average* time constraints. The idea behind average time constraints is that although we cannot prove that a stochastic dynamical system can *always* satisfy some given time constraint, we can show that the average behaviour of the system does satisfy the constraints. This is similar to the well-known concept of sample paths and expected sample paths of stochastic analysis. For completeness of this discussion, we present the notion of timed-∀-automata prior to introducing the definitions for what we call average-timed ∀-automata.

Definition 4.7 (Syntax of timed ∀-automata) *A* timed ∀-automaton $\mathcal{T}A$ *is a triple* $\langle \mathcal{A}, T, \tau \rangle$ *where* $\mathcal{A} = \langle Q, R, S, e, c \rangle$ *is a* ∀*-automaton,* $T \subseteq Q$ *is a set of* timed *automaton states and* $\tau : T \cup \{bad\} \to \mathbb{R}^+ \cup \{\infty\}$ *is a* time function.

It is easy to show that any ∀-automaton is equivalent to a special timed ∀-automaton with $T = \emptyset$ and $\tau(bad) = \infty$. Graphically, a T-state is denoted by a nonnegative real number indicating its time bound. The conventions for complete ∀-automata (which were omitted in the last section for lack of space) are adopted for timed ∀-automata.

Let $v : \mathcal{T} \to A$ be a trace. We define a *run* r of $\mathcal{T}A$ over v has being a run of $\mathcal{A}$ over v; r is *accepting* for $\mathcal{T}A$ iff

1. r is accepting for $\mathcal{A}$ and

2. r satisfies the time constraints. If $I \subseteq \mathcal{T}$ is an interval of $\mathcal{T}$ and $q^* : I \to Q$ is a segment of run r, i.e., $q^* = r_{|I}$, let $\mu(q^*)$ denote the measure of q^*, i.e., $\mu(q^*) = \mu(I) = \Sigma_{t \in I} \mu(t)$ since I is discrete. Furthermore, let $\mu_B(q^*)$ denote the measure of bad automaton states in q^*, i.e., $\mu_B(q^*) = \Sigma_{t \in I, q^*(t) \in B} \mu(t)$. Let $Sg(q)$ be the set of segments of consecutive q's in r, i.e., $q^* \in Sg(q)$ implies $\forall t \in I, q^*(t) = q$. Let BS be the set of segments of consecutive B and S-states in r, i.e., $q^* \in BS$ implies $\forall t \in I, q^*(t) \in B \cup S$. The run r satisfies the time condition iff

   (a) (local time constraint) $\forall q \in T, q^* \in Sg(q),\ \mu(q^*) \leq \tau(q)$ and

   (b) (global time constraint) $\forall q^* \in BS,\ \mu_B(q^*) \leq \tau(bad)$.

The first condition stipulates that for a local time constraint, the system will not stay continuously in a given state $q \in T$ for longer than its local time bound $\tau(q)$. The second condition requires the system to leave the set of bad states within $\tau(bad)$ time units.

Definition 4.8 (Semantics of timed $\forall$-automata) *A timed $\forall$-automaton $\mathcal{T}A$* accepts *a trace v, written $v \models \mathcal{T}A$, iff* all *possible runs of $\mathcal{T}A$ over v are accepting.*

We now present the syntax and semantics of average-timed $\forall$-automata, which extend the definitions for timed $\forall$-automata presented above. Average-timed $\forall$-automate allow for the verification of behavioural constraints for systems exhibiting uncertainty.

Definition 4.9 (Syntax of average-timed $\forall$-automata) *An average-timed $\forall$-automaton $\mathcal{ATA}$ is a triple $\langle \mathcal{A}, T, \tau \rangle$ where $\mathcal{A} = \langle Q, R, S, e, c \rangle$ is a $\forall$-automaton, $T \subseteq Q$ is a set of* average-timed *automaton states and $\tau : T \cup \{bad\} \to \mathbb{R}^+ \cup \{\infty\}$ is an* average-timing function*.*

We can easily show that any $\forall$-automaton is equivalent to a special average-timed $\forall$-automaton with $T = \emptyset$ and $\tau(bad) = \infty$. A T-state is denoted by a nonnegative real number indicating its average-time bound. However, unlike with typical $\forall$-automata, or even timed-$\forall$-automata, we cannot define the acceptance of a single trace by an average-timed $\forall$-automata. In fact, due to the stochastic nature of the systems of interest, we are no longer interested in the behaviour exhibited by individual traces but rather in the behaviour of a set of traces. Expected time constraints should be satisfied by the average behaviours of systems, hence we need to look at the ensemble of traces induced by those systems.

Let $v : \mathcal{T} \to A$ be a trace from the behaviour $\mathcal{B}$ of a system. We define a *run* r of $\mathcal{ATA}$ over v has being a run of $\mathcal{A}$ over v. A run r is *accepting* for $\mathcal{ATA}$ iff

1. r is accepting for $\mathcal{A}$ and

2. r satisfies the expected time constraints. If $I \subseteq \mathcal{T}$ is an interval of $\mathcal{T}$ and $q^* : I \to Q$ is a segment of run r, i.e., $q^* = r_{|I}$, let $\mu(q^*)$ denote the measure of q^*, i.e., $\mu(q^*) = \mu(I) = \Sigma_{t \in I} \mu(t)$ since I is discrete. Furthermore, let $\mu_B(q^*)$ denote the measure of bad automaton states in q^*, i.e., $\mu_B(q^*) = \Sigma_{t \in I, q^*(t) \in B} \mu(t)$.

Let $Sg(q)$ be the set of segments of consecutive q's in r, i.e., $q^* \in Sg(q)$ implies $\forall t \in I, q^*(t) = q$. Let BS be the set of segments of consecutive B and S-states in r, i.e., $q^* \in BS$ implies $\forall t \in I, q^*(t) \in B \cup S$. The run r satisfies the time condition iff

(a) (local time constraint) $\forall q \in T, q^* \in Sg(q),\ \mathbb{E}(\mu(q^*)) \leq \tau(q)$ and

(b) (global time constraint) $\forall q^* \in BS,\ \mathbb{E}(\mu_B(q^*)) \leq \tau(bad)$.

Definition 4.10 (Semantics of average-timed $\forall$-automata) *An average-timed $\forall$-automaton $\mathcal{ATA}$* accepts *a trace v, written $v \models \mathcal{ATA}$, iff all possible expected runs of $\mathcal{ATA}$ over v are accepting.*

As an example, Figure 8 depicts the real-time response constraint which states that R will be reached within 40 time units of B, where the time in S is not accounted for.

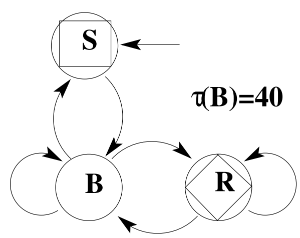


Figure 8: Timed $\forall$-Automaton Specifications: real-time bounded response

5 Model-Checking Approach to Constraint-Based Behaviour Verification

We have defined above the concepts of the behaviour of a system and of behavioural constraints. Given the behaviour $\mathcal{B}$ of a stochastic dynamical system and a behavioural constraint $\mathcal{B}_C$, the behaviour satisfies the requirements at level α, written as $\mathcal{B} \models_\alpha \mathcal{RS}$, iff $\mu(\{v \in \mathcal{B} | v \models \mathcal{RS}\}) = \alpha$. As mentioned before, we will restrict ourselves to the satisfaction of the behavioural constraints of a system at "average" level only. Therefore, given the probabilistic constraint net model of a system and an average-timed $\forall$-automaton specification, we say that the behaviour of the system satisfies the requirement specification if and only if the all traces of the system are accepting for the average-timed $\forall$-automaton.

The formal behaviour verification method consists of a set of model-checking rules. The rules are a generalization of the rules for dynamical systems developed by [2], which themselves extended the rules developed for concurrent programs [10].

5.1 Behavioural Constraint Verification Rules for Discrete-time

For sake of simplicity and since it applies directly to the elevator example presented earlier, we will introduce the verification rules for the simplest possible situation: discrete time and discrete domain. A generalization of the rules for arbitrary time and domain will follow shortly.

As we mentioned earlier, any time-invariant Markovian behaviour $\mathcal{B}$ in discrete time corresponds to a stochastic state transition system $\langle \mathcal{S}_{\mathcal{B}}, \mathbb{P}, \Theta \rangle$ for which we denoted an allowed transition from state s to state s' by $s \rightsquigarrow s'$. We also write $\{\varphi\}\mathcal{B}\{\psi\}$ iff the consecutive condition: $\varphi(s) \wedge (s \rightsquigarrow s') \rightarrow \psi(s')$ is valid. This relation is different from the one defined for deterministic systems in that it is valid not if there is a transition from s to s' but more generally if there is an allowed transition (non-zero probability of transition) from s to s'.

Our verification method is composed of three types of rules: Invariance rules (I), Stability (Lyapunov-based) rules (S) and Average Timeliness rules (AT). Assume $\mathcal{ATA}$ is a $\forall$-automaton $\langle \mathcal{A}, T, \tau \rangle$ representing the behavioural constraints for the stochastic dynamical system: $\langle \mathcal{S}, \mathbb{P}, \Theta \rangle$.

5.1.1 (I) Invariance Rules

We define a set of propositions $\{\alpha_q\}_{q \in Q}$ as a set of *invariants* for the behaviour $\mathcal{B}$ and specification $\mathcal{A}$ iff

1. *Initiality*: $\forall q \in Q, \Theta \wedge e(q) \rightarrow \alpha_q$, and
2. *Consecution*: $\forall q, q' \in Q, \{\alpha_q\}\mathcal{B}\{c(q, q') \rightarrow \alpha_{q'}\}$.

Proposition 5.1 *Let $\{\alpha_q\}_{q \in Q}$ be invariants for $\mathcal{B}$ and $\mathcal{A}$. If r is a run of $\mathcal{A}$ over a trace $v \in \mathcal{B}$, then $\forall t \in \mathcal{T}, v(t) \models \alpha_{r(t)}$.*

Proof : For any trace v, $v(\mathbf{0})$ is an initial state, therefore, $v(\mathbf{0}) \models \Theta$. Furthermore, r is a run, $v(\mathbf{0}) \models e(r(\mathbf{0}))$. Therefore, $v(\mathbf{0}) \models e(r(\mathbf{0})) \wedge \Theta$. Since $e(r(\mathbf{0})) \wedge \Theta \rightarrow \alpha_{r(\mathbf{0})}$, we have $v(\mathbf{0}) \models \alpha_{r(\mathbf{0})}$.

Assume that $v(pre(t)) \models \alpha_{r(pre(t))}$. Therefore, $v(t) \models c(r(pre(t)), r(t)) \rightarrow \alpha_{r(t)}$ since $v(pre(t)) \rightsquigarrow v(t)$. In addition, $v(t) \models c(r(pre(t)), r(t))$. Therefore, $v(t) \models \alpha_{r(t)}$.

A set equipped with a well-founded relation is said to be a well-founded set. A well-founded set is a partially ordered set which contains no infinite descending chains, or equivalently, a partially ordered set in which every non-empty subset has a minimal element. If the order is a total order then the set is called a well-ordered set. Since by definition, time is a linearly ordered set with **0** as a least element, $\mathcal{T}$ is then a well-founded set. Using the technique of mathematical induction for well-founded sets, we can conclude that $v(t) \models \alpha_{r(t)}$ for all t. $\Box$

Note that this proposition stipulates that no matter which (uncertain) transition occurs, the destination state must always satisfy the invariant condition. This is consistent with the notion of invariants, regardless of whether the dynamics of the underlying systems are deterministic or not.

5.1.2 (S) Stability Rules

Let $\{\alpha_q\}_{q\in Q}$ be a set of invariants for $\mathcal{B}$ and $\mathcal{A}$ as defined above. A set of partial functions $\{\rho_q\}_{q\in Q}$ is called a set of *Lyapunov functions* for $\mathcal{B}$ and $\mathcal{A}$ iff $\rho_q : \mathcal{S}_{\mathcal{B}} \to \mathbb{R}^+$ satisfies the following conditions:

1. *Definedness*: $\forall q \in Q, \alpha_q \to \exists w \in \mathbb{R}^+, \rho_q = w$.
2. *Non-increase*: $\forall q \in S, q' \in Q, \{\alpha_q \wedge \rho_q = w\}\mathcal{B}\{c(q,q') \to \mathbb{E}(\rho_{q'}) \leq w\}$.
3. *Decrease*: $\exists \epsilon > 0, \forall q \in B, \exists q' \in Q, \{\alpha_q \wedge \rho_q = w\}\mathcal{B}\{c(q,q') \to \rho_{q'} - w \leq -\epsilon.\}$

Those three conditions are derived from [11]. However, the last two have been adapted for stochastic dynamical systems. Condition (S2) requires that for each stable state $q \in S$, the transitions from q lead on average to a state for which the value of the Lyapunov function is less than or equal to the current value. Condition (S3) is similar in that it requires that for each bad state $q \in B$, there exists at least one allowed transition (i.e., with positive probability) leading to a state with strictly smaller Lyapunov value. This is a formal requirement that can only be satisfied if there are no absorbing bad states in the system under study, as discussed previously.

Proposition 5.2 *Let $\{\alpha_q\}_{q\in Q}$ be a set of invariants for $\mathcal{B}$ and $\mathcal{A}$. Let r be a run of $\mathcal{A}$ over a trace $v \in \mathcal{B}$. Also, let $\mathcal{V}_{\mathcal{B}} = \{r | r \text{ is a run of } \mathcal{A} \text{ over } v \in \mathcal{B}\}$ be the set of runs induced by $\mathcal{B}$. If $\{\rho_q\}_{q\in Q}$ is a set of Lyapunov functions for $\mathcal{B}$ and $\mathcal{A}$, then*

- $\forall t \in \mathcal{T}, \mathbb{E}_{r^*,v^*}(\rho_{r^*(t)}(v^*(t))) \leq \rho_{r(pre(t))}(v(pre(t)))$*, where r^* and v^* denote all $r' \in \mathcal{V}_{\mathcal{B}}$ and $v' \in \mathcal{B}$ such that $v(pre(t)) \rightsquigarrow v'(t)$ and $c(r(pre(t)), r'(t))$, when $r(pre(t)) \in S$;*
- $\exists \epsilon > 0, \forall t \in \mathcal{T}, \exists v' \in \mathcal{B}, \exists r' \in \mathcal{V}_{\mathcal{B}}, ([\rho_{r'(t)}(v'(t)) - \rho_{r(pre(t))}(v(pre(t))) \leq -\epsilon] \wedge [(v(pre(t)) \rightsquigarrow v'(t)) \wedge c(r(pre(t)), r'(t))])$ *when* $r(pre(t)) \in B$.

Proof : Obtained from the conditions of the Lyapunov functions. □

Theorem 1 *Let $\{\alpha_q\}_{q\in Q}$ be a set of invariants for $\mathcal{B}$ and $\mathcal{A}$. Let r be a run of $\mathcal{A}$ over a trace $v \in \mathcal{B}$. If $\{\rho_q\}_{q\in Q}$ is a set of Lyapunov functions for $\mathcal{B}$ and $\mathcal{A}$, then*

- *if BS is the set of segments of consecutive B-states and S-states in r, then $\forall q^* \in BS, q^*$ has a finite number of B-states;*

Proof : From the conditions of the Lyapunov functions, we are assured that a transition from any B-states will eventually lead to a state with smaller Lyapunov value. Moreover, from these conditions and due to the fact that the Lyapunov functions are defined on a well-founded set, we are also assured that any segments containing only B-states and S-states, the system will eventually finish in an absorbing S or will leave to an R-state, both cases being satisfactory, hence completing the proof. □

The first result of Proposition 5.2 is valid for every run of a system. However, the last two results need to be applied to a set of traces, and thus to a set of runs. In the context of stochastic dynamical systems, we cannot guarantee that for every trace a transition from a bad (or stable) state to any other state will yield an immediate decrease (or non-increase) in the value of the Lyapunov function; nevertheless, we can show that there is a positive probability of this happening at any time point. Hence, we know that at any time t there exists at least one trace whose transition from $v(t)$ to $v(t+1)$ will yield a decrease in the value of the Lyapunov function.

5.1.3 (AT) Average-Timeliness Rules

Let $\mathcal{ATA} = \langle \mathcal{A}, T, \tau \rangle$ be an average-timed $\forall$-automata. Assume, without loss of generality, that time is encoded in the stochastic state transition system. We assume that it is defined in a general sense as $\lambda : \mathcal{S}_\mathcal{B} \to \mathcal{T}$; i.e., as a function of time measure on states returning the time until the next transition. Note that for the special case of discrete time systems on $\mathbb{N}$, $\lambda \equiv 1$ uniformly. We now define two different types of timing functions, associated with the local and global average-time bounds respectively.

Once again, let $\{\alpha_q\}_{q \in Q}$ be a set of invariants for $\mathcal{B}$ and $\mathcal{A}$. A set of partial functions $\{\gamma_q\}_{q \in T}$ is called a set of *local timing functions* for $\mathcal{B}$ and $\mathcal{ATA}$ iff $\gamma_q : \mathcal{S}_\mathcal{B} \to \mathbb{R}^+$ satisfies the following conditions:

(L1) *Boundedness*: $\forall q \in T, \alpha_q \to \lambda \leq \gamma_q \leq \tau(q)$.

(L2) *Decrease*: $\forall q \in T, \{\alpha_q \wedge \gamma_q = w \wedge \mathbb{E}(\lambda) = l\}\mathcal{B}\{c(q,q) \to \mathbb{E}(\gamma_q) - w \leq -l\}$.

A set of partial functions $\{\eta_q\}_{q \in Q}$ is called a set of *global timing functions* for $\mathcal{B}$ and $\mathcal{ATA}$ iff $\eta_q : \mathcal{S}_\mathcal{B} \to \mathbb{R}^+$ satisfies the following conditions:

(G1) *Definedness*: $\forall q \in Q, \alpha_q \to \exists w \in \mathbb{R}^+, \eta_q = w$.

(G2) *Boundedness*: $\forall q \in B, \alpha_q \to \eta_q \leq \tau(bad)$.

(G3) *Non-increase*: $\forall q \in S, q' \in Q, \{\alpha_q \wedge \eta_q = w\}\mathcal{B}\{c(q,q') \to \mathbb{E}(\eta_{q'}) \leq w\}$.

(G4) *Decrease*: $\forall q \in B, q' \in Q, \{\alpha_q \wedge \eta_q = w \wedge \mathbb{E}(\lambda) = l\}\mathcal{B}\{c(q,q') \to \mathbb{E}(\eta_{q'}) - w \leq -l\}$.

Proposition 5.3 *Let $\{\alpha_q\}_{q \in Q}$ be a set of invariants for $\mathcal{B}$ and $\mathcal{A}$, and r be a run of $\mathcal{A}$ over a trace $v \in \mathcal{A}$. If there exist local timing functions, $\{\gamma_q\}_{q \in T}$, and global timing functions, $\{\eta_q\}_{q \in Q}$, for $\mathcal{B}$ and $\mathcal{ATA}$, then*

1. *if $Sg(q)$ is the set of segments of consecutive q's in r, then $\forall q \in T, q^* \in Sg(q)$, $\mathbb{E}(\mu(q^*)) \leq \tau(q)$, and*
2. *if BS is the set of segments of consecutive B and S-states in r, then $\forall q^* \in BS$, $\mathbb{E}(\mu_B(q^*)) \leq \tau(bad)$.*

Proof :

1. Let $\{s_i\}_{i=1}^n$ be a sequence of q-states.

   From condition (ATL1), we know that $\forall q \in T, \alpha_q \to \lambda \leq \gamma_q \leq \tau(q)$. Therefore we can also deduce that:

(1) $\mathbb{E}(\lambda(s_n)) \leq \mathbb{E}(\gamma_q(s_n))$ and
(2) $\mathbb{E}(\gamma_q(s_n)) \leq \mathbb{E}(\tau(q)) = \tau(q)$.

From condition (ETL2), we get:

$$\mathbb{E}(\gamma_q(s_2) - \gamma_q(s_1)) \leq -\mathbb{E}(\lambda(s_1))$$

$$\mathbb{E}(\gamma_q(s_3) - \gamma_q(s_2)) \leq -\mathbb{E}(\lambda(s_2))$$

$$\ldots$$

$$\mathbb{E}(\gamma_q(s_n) - \gamma_q(s_{n-1})) \leq -\mathbb{E}(\lambda(s_{n-1}))$$

adding these inequalities, we have

$$\mathbb{E}(\gamma_q(s_n)) - \mathbb{E}(\gamma_q(s_1)) \leq -\Sigma_{i=1}^{n-1}\mathbb{E}(\lambda(s_i)).$$

Using (1) and (2) above we get:

$$\Sigma_{i=1}^{n}\mathbb{E}(\lambda(s_i)) \leq \tau(q),$$

$$\mathbb{E}(\Sigma_{i=1}^{n}\lambda(s_i)) \leq \tau(q),$$

$$\mathbb{E}(\mu(q^*)) \leq \tau(q).$$

2. Let $\{s_i\}_{i=1}^{n}$ be a sub-sequence of B-states in a BS segment. Similarly to the proof above (using (ATG4) this time), we get:

$$\mathbb{E}(\eta_{q'_1}(s'_1) - \eta_{q_1}(s_1)) \leq -\mathbb{E}(\lambda(s_1))$$

$$\mathbb{E}(\eta_{q'_2}(s'_2) - \eta_{q_2}(s_2)) \leq -\mathbb{E}(\lambda(s_2))$$

$$\ldots$$

$$\mathbb{E}(\eta_{q'_n}(s'_n) - \eta_{q_n}(s_n)) \leq -\mathbb{E}(\lambda(s_n))$$

and

$$\mathbb{E}(\eta_{q'_i}(s'_i)) \geq \mathbb{E}(\eta_{q_{i+1}}(s_{i+1}))$$

we have

$$\mathbb{E}(\eta_{q'_n}(s'_n)) - \mathbb{E}(\eta_{q_1}(s_1)) \leq -\Sigma_{i=1}^{n}\mathbb{E}(\lambda(s_i)).$$

Finally, from (ATG2) we get that $\mathbb{E}(\eta_{q_1}(s_1)) \leq \tau(bad)$ and $\mathbb{E}(\eta_{q'_n}(s'_n)) \geq 0$. Therefore,

$$\Sigma_{i=1}^{n}\mathbb{E}(\lambda(s_i)) \leq \tau(bad)$$

$$\mathbb{E}(\Sigma_{i=1}^{n}\lambda(s_i)) \leq \tau(bad)$$

$$\mathbb{E}(\Sigma_{i=1}^{n}\mu(q^*)) \leq \tau(bad)$$

$\square$

Following is the set of *verification rules* for a behaviour $\mathcal{B}$ and an average-timed automaton $\mathcal{ATA} = \langle \mathcal{A}, T, \tau \rangle$:

(I) Associate with each automaton state $q \in Q$ a state formula α_q, such that $\{\alpha_q\}_{q \in Q}$ is a set of invariants for $\mathcal{B}$ and $\mathcal{A}$.

(S) Associate with each automaton state $q \in Q$ a partial function ρ_q, such that $\{\rho_q\}_{q \in Q}$ is a set of Lyapunov functions for $\mathcal{B}$ and $\mathcal{A}$.

(ET) Associate with each average-timed automaton state $q \in T$ a partial function γ_q, such that $\{\gamma_q\}_{q \in T}$ is a set of local timing functions for $\mathcal{B}$ and $\mathcal{ATA}$. Associate with each automaton state $q \in Q$ a partial function η_q, such that $\{\eta_q\}_{q \in Q}$ is a set of global timing functions for $\mathcal{B}$ and $\mathcal{ATA}$.

Let us now present the main result of this section. The following theorem stipulates that if we are equipped with a set of invariants, Lyapunov functions and local and global timing functions, then the behaviour verification is sound and complete.

Theorem 2 (Verification Rules) *For any state-based and time-invariant behaviour $\mathcal{B}$ with an infinite time structure and a complete average-timed $\forall$-automaton $\mathcal{ATA}$, the verification rules are sound and complete, i.e., $\mathcal{B} \models \mathcal{ATA}$ iff there exist a set of invariants, Lyapunov functions and timing functions.*

Proof (Verification Rules):

Soundness ($\Longleftarrow$)

The construction of these rules guarantees the soundness of the verification method. For any trace v, there is a run because $\mathcal{ATA}$ is complete. For any run r over v, if any automaton state in R appears infinitely many times in r, r is accepting. Otherwise, there is a time point $t_0 \in \mathcal{T}$, the sub-sequence r on $I = \{t \in \mathcal{T} | t \geq t_0\}$, denoted q^*, has only bad and stable automaton states. From the results of Proposition 5.2, if there exist a set of invariants and a set of Lyapunov functions, q^* has only a finite number of B-states. Since time is infinite, all the automaton states appearing infinitely many times in r belong to S; so r is accepting too. Therefore, every trace is accepting for the automaton. If there exists a set of local and global timing functions, every trace satisfies the timing constraints on average.

Completeness ($\Longrightarrow$)

On the other hand, if $\mathcal{ATA}$ is valid over $\mathcal{B}$, then to prove completeness, we need to show that there exist a set of invariants, a set of Lyapunov functions, and a set of local and global timing functions that satisfy the requirements. We will use a constructive proof which in turn will be used later when introducing the verification algorithm.

For any state $s \in \mathcal{S}$ and state proposition α, we write $\alpha(s)$ iff $s \models \alpha$. It is possible to construct the invariants by choosing them as the fix-point of the set of equations:

$$\alpha_{q'}(s') = (\exists q, s, \alpha_q(s) \wedge (s \rightsquigarrow s') \wedge c(q, q')(s')) \bigvee (\theta(s') \wedge e(q')(s')). \quad (3)$$

We can verify that $\{\alpha_q\}_{q\in Q}$ is a set of propositions over $S_{\mathcal{B}}$ and satisfies the requirements of initiality and consecution. The reader is referred to [12, 13] for a formal argument showing that the invariants can be obtained via the fixpoints of the above equations. Furthermore, $s \models \alpha_q$ iff $\langle q, s\rangle$ is a reachable pair for $\mathcal{ATA}$ and $\mathcal{B}$.

Given the constructed invariants $\{\alpha_q\}_{q\in Q}$, a set of Lyapunov functions $\{\rho_q\}_{q\in Q}$ and a set of global timing functions $\{\eta_q\}_{q\in Q}$ can be constructed as follows:

- $\forall q \in R, s \models \alpha_q$, let $\rho_q(s) = 0$ and $\eta_q(s) = 0$.
- $\forall q \notin R, s \models \alpha_q$, $\rho_q(s)$ and $\eta_q(s)$ are defined as follows. Construct a weighted directed graph $G = \langle V, E, W\rangle$, where W is the set of weights corresponding to the transition probabilities, such that $\langle q, s\rangle \in V$ iff $q \notin R, s \models \alpha_q$, and $\langle q, s\rangle \rightsquigarrow \langle q', s'\rangle$ in E iff $s \rightsquigarrow s' \wedge c(q, q')(s')$. For any $\langle q, s\rangle \in V$, let $\mathbb{E}(P_B)$ be the average number of B-states in the set of paths P starting from $\langle q, s\rangle$ and $\mathbb{E}(\mu_B(P))$ be the average measure of B-states in P. Let $\rho_q(s) = \mathbb{E}(P_B)$ and $\eta_q(s) = \mathbb{E}(\mu_B(P))$.

We can verify that $\{\rho_q\}_{q\in Q}$ is a set of Lyapunov functions, and that $\{\eta_q\}_{q\in Q}$ is a set of global timing functions.

Similarly, a set of local timing functions $\{\gamma_q\}_{q\in T}$ can be constructed as follows. For all $q \in T$, construct a weighted directed graph $G = \langle V, E, W\rangle$, such that $s \in V$ iff $s \models \alpha_q$, and $s \rightsquigarrow s'$ in E iff $s \rightsquigarrow s' \wedge c(q, q)(s')$. For paths P starting at s, let $\mathbb{E}(\mu(P))$ be the average measure of the path. Let $\gamma_q(s) = \mathbb{E}(\mu(P))$. We can verify that $\{\gamma_q\}_{q\in T}$ is a set of local timing functions. $\Box$

5.2 Automatic Behaviour Verification

The above rules do not guarantee the existence of an automatic verification method. However, for finite domain probabilistic constraint nets, we can fully automate the process in order to verify an average-timed $\forall$-automata constraint on the behaviour. We will briefly describe the algorithm and then will utilize it to verify the elevator system augmented with probabilistic passenger arrivals.

First, let us assume that $PCN = \langle Lc, Td, Tp, Cn\rangle$ is a probabilistic constraint net made solely of transliterations and unit delays. We denote an acceptable state by $PCN(s)$ iff for every equation of the form $l_0 = f(l_1, \cdots, l_n) \rightarrow Pr(s(l_0) = f(s(l_1), \cdots, s(l_n))) > 0$, and denote an acceptable transition by $PCN(s, s')$ if and only if $PCN(s)$ and $PCN(s')$, and if for every delay equation $l'_0 = l$, $s'(l_0) = s(l)$. Let us also denote a *reachable pair* (q, s) by $r(q, s)$ where $q \in Q$ and $s \in \times_{Lc} A_{s_l}$. Furthermore, let K be the evolution kernel associated with the set of reachable pairs for $q \in B \cup S$ and let T be the matrix summarizing the time for each transition within the set of reachable pairs with $q \in B \cup S$. In addition, let K_1 represent the evolution kernel for the set of reachable states with $q \in B \cup S$ which constitute the RS-boundary of the set $B \cup S$. This set is composed of all the bad and stable states which have a direct transition to a state $r \in R$ or to an absorbing S state. Proceed similarly to define T_1. Finally, let L and L_1 be the matrix T and T_1 respectively where the non-zero entries have been replaced by the value 1.

The algorithm follows the verification rules and has four steps which we describe below:

1. **Invariant Generation:** We can show that invariants can be constructed by finding the fix-point of the sets of Equation 3. This fix-point can be obtained with the following two steps:

   (a) **Initiality:** Generate $r(q, s)$ if $\Theta(s), e(q)(s), PCN(s)$.

   (b) **Consecution:** Generate $r(q', s')$ if $r(q, s), PCN(s, s'), c(q, q')(s')$.

2. **Non-Absorbness and Stability:**

   - Verify that the set of bad states is irreducible. That is, ensure that for every bad state $b \in B$ there is a path with non-zero measure leading to a R-state or an S-state.
   - For $q \in R$, let $\rho_q = 0$.
   - Solve the set of linear equations for the average number of transitions taken to enter the set of recurrent states or absorbing S-states. The solution is the set of Lyapunov functions, $\{\rho_q\}_{q \in B \cup S}$. Practically, to solve for $\{\rho_q\}_{q \in B \cup S}$, define $A = -K + I_n$ and $u = diag([T, T_1] * [K, K_1]')$, then solve $A\rho = u$. Here $diag$ denotes the diagonal operator, which returns the diagonal elements of a matrix.

3. **Global Average Timing:**

   - For $q \in R$, let $\eta_q = 0$.
   - Similarly to the method for stability, solve the set of linear equations for the average time measure to leave the set of bad and non-absorbing stable states, not accounting for time spent in an S-state. The solution is the set of Global timing functions $\{\eta_q\}_{q \in B \cup S}$. Verify that $\eta_q < \tau(bad), \forall q \in Q$.

4. **Local Average Timing:**

   - For each $q \in T$, solve the set of linear equations for the average time measure to leave q. This is similar to solving for the Lyapunov functions where we only consider states $q \in T$. This leads to the local timing functions $\{\gamma_q\}_{q \in T}$. Verify that $\gamma_q < \tau(q), \forall q \in T$.

It is possible, given this method, to obtain a bound on the probability that a certain time bound will be exceeded. The bound is obtained from the well known equation $Pr(X > \tau) \leq \mathbb{E}(X^2)/\tau^2$. We can calculate $\mathbb{E}(X^2)$, where X is the average time to reach reach a R-state or an absorbing S-state, as obtained in the global and local average timeliness rules. With the equation

$$u = 2[T, T1]. * [K, K1] * \eta + diag([T, T1]. * [T, T1] * [P, P1]',)$$

where $.*$ denote the element-wise matric multiplication, solve $AY = u$ for Y to calculate the value of the probability bound.

6 Generalizing the Approach

In this section, we generalize the concept of average-timed $\forall$-automata on discrete time structures to arbitrary time structures. This allows us to apply our method to behavioural constraint verification of stochastic hybrid dynamical systems, which generate traces on general time structures. Note that the common time structures of continuous and discrete time both act as special cases.

Essentially, the set of verification rules for general time structures follows closely that of discrete time systems, however, the definitions of invariants, Lyapunov functions and timing functions are generalized.

For any trace $v : \mathcal{T} \to A$, let $\{\varphi\}v\{\psi\}$ denote the validity of the following two consecutive conditions:

- $\{\varphi\}v^-\{\psi\}$: for all $t > \mathbf{0}$, $\exists t' < t$, $\forall t'', t' \le t'' < t, v(t'') \models \varphi$ implies $v(t) \models \psi$;
- $\{\varphi\}v^+\{\psi\}$: for all $t < \infty$, $v(t) \models \varphi$ implies $\exists t' > t, \forall t'', t < t'' < t', v(t'') \models \psi$.

If $\mathcal{T}$ is discrete, these two conditions are reduced to one, i.e., $\forall t > \mathbf{0}, v(pre(t)) \models \varphi$ implies $v(t) \models \psi$.

Given $\mathcal{B}$ as a behaviour, let $\Theta = \{v(\mathbf{0}) | v \in \mathcal{B}\}$ denote the set of initial values in $\mathcal{B}$. Let $\mathcal{A} = \langle Q, R, S, e, c\rangle$ be a $\forall$-automaton. A set of propositions $\{\alpha_q\}_{q\in Q}$ is called a set of *invariants* for $\mathcal{B}$ and $\mathcal{A}$ iff

- *Initiality*: $\forall q \in Q, \Theta \wedge e(q) \to \alpha_q$.
- *Consecution*: $\forall v \in \mathcal{B}, \forall q, q' \in Q, \{\alpha_q\}v\{c(q, q') \to \alpha_{q'}\}$.

Proposition 6.1 *Let $\{\alpha_q\}_{q\in Q}$ be invariants for $\mathcal{B}$ and $\mathcal{A}$. If r is a run of $\mathcal{A}$ over $v \in \mathcal{B}$, $\forall t \in \mathcal{T}, v(t) \models \alpha_{r(t)}$.*

Proof : In order to prove this proposition, we shall introduce a variation of the method of continuous induction [14]. A property Γ is *inductive* on a time structure $\mathcal{T}$ iff for all $t_0 \in \mathcal{T}$, Γ is satisfied at all $t < t_0$ implies that Γ is satisfied at t_0. Γ is *continuous* iff Γ is satisfied at a non-greatest element $t \in \mathcal{T}$ implies that $\exists t' > t, \forall t < t'' < t'$, Γ is satisfied at t''. Note that when $\mathcal{T}$ is discrete, any property is continuous. The theorem of continuous induction [14] says:

Theorem 3 (Continuous Induction) *If the property Γ is inductive and continuous on a time structure $\mathcal{T}$ and Γ is satisfied at $\mathbf{0}$, Γ is satisfied at all $t \in \mathcal{T}$.*

We prove that the property $v(t) \models \alpha_{r(t)}$ is satisfied at $\mathbf{0}$ and is both inductive and continuous on any time structure $\mathcal{T}$.

- Initiality: Since $v(\mathbf{0}) \models \Theta$ and $v(\mathbf{0}) \models e(r(\mathbf{0}))$, we have $v(\mathbf{0}) \models \Theta \wedge e(r(\mathbf{0}))$. According to the *Initiality* condition of invariants, we have $v(\mathbf{0}) \models \alpha_{r(\mathbf{0})}$.

- Inductivity: Suppose $v(t) \models \alpha_{r(t)}$ is satisfied at $\mathbf{0} \leq t < t_0$. Since r is a run over v, $\exists q \in Q$ and $t'_1 < t_0, \forall t, t'_1 \leq t < t_0, r(t) = q$ and $v(t_0) \models c(q, r(t_0))$. According to the *Consecution* condition of the invariants, $\exists t'_2 < t_0, \forall t, t'_2 \leq t < t_0$, $v(t) \models \alpha_q$ implies $v(t_0) \models c(q, r(t_0)) \to \alpha_{r(t_0)}$. Therefore, $\forall t, \max(t'_1, t'_2) \leq t < t_0$, $r(t) = q$, $v(t) \models \alpha_q$ (assumption), $v(t_0) \models c(q, r(t_0)) \to \alpha_{r(t_0)}$ and $v(t_0) \models c(q, r(t_0))$. Thus, $v(t_0) \models \alpha_{r(t_0)}$.

- Continuity: Suppose $v(t_0) \models \alpha_{r(t_0)}$. Since r is a run over v, $\exists q \in Q$ and $t'_1 > t_0, \forall t, t_0 < t < t'_1$, $r(t) = q$ and $v(t) \models c(r(t_0), q)$. According to the *Consecution* condition of the invariants, $\exists t'_2 > t_0, \forall t, t_0 < t < t'_2$, $v(t_0) \models \alpha_{r(t_0)}$ implies $v(t) \models c(r(t_0), q) \to \alpha_q$. Therefore, $\forall t, t_0 < t < \min(t'_1, t'_2)$, $r(t) = q$, $v(t_0) \models \alpha_{r(t_0)}$ (assumption), $v(t) \models c(r(t_0), q) \to \alpha_q$ and $v(t) \models c(r(t_0), q)$. Thus, $\forall t, t_0 < t < \min(t'_1, t'_2), v(t) \models \alpha_{r(t)}$.

$\Box$

Proof (Theorem 3: Continuous Induction): We call a time point $t \in \mathcal{T}$ *regular* iff Γ is satisfied at all t', $\mathbf{0} \leq t' \leq t$. Let T denote the set of all regular time points. T is not empty since Γ is satisfied at $\mathbf{0}$. We prove the theorem by contradiction, i.e., assume that Γ is not satisfied at all $t \in \mathcal{T}$. Therefore, $T \subset \mathcal{T}$ is bounded above; let $t_0 = \bigvee T \in \mathcal{T}$ be the least upper bound of T (t_0 exists according to Proposition 3.2.1). Since t_0 is the least upper bound, it follows that Γ is satisfied at all t, $\mathbf{0} \leq t < t_0$. Since Γ is inductive, it is satisfied at time t_0. Therefore, $t_0 \in T$.

Since $T \subset \mathcal{T}$, t_0 is not the greatest element in $\mathcal{T}$. Let $T' = \{t | t > t_0\}$. There are two cases: (1) if T' has a least element t', since Γ is inductive, $t' \in T$ is a regular time point. (2) otherwise, for any $t' \in T'$, $\{t | t_0 < t < t'\} \neq \emptyset$. Since Γ is also continuous, we can find a $t' \in T'$ such that Γ is satisfied at all $T'' = \{t | t_0 < t < t'\}$. Therefore, t is a regular time point $\forall t \in T''$. Both cases contradict the fact that t_0 is the least upper bound of the set T. $\Box$

Without loss of generality, we assume that time is encoded in domain A by $\lambda : A \to \mathcal{T}$. Given that $\{\alpha_q\}_{q \in Q}$ is a set of invariants for $\mathcal{B}$ and $\mathcal{A}$, a set of partial functions $\{\rho_q\}_{q \in Q} : A \to \mathcal{R}^+$ is called a set of *Lyapunov functions* for $\mathcal{B}$ and $\mathcal{A}$ iff the following conditions are satisfied:

- *Definedness*: $\forall q \in Q, \ \alpha_q \to \exists w \in \mathbb{R}^+, \rho_q = w$.

- *Non-increase*: $\forall v \in \mathcal{B}, \forall q \in S, q' \in Q$,

$$\{\alpha_q \wedge \rho_q = w\} v^- \{c(q, q') \to \mathbb{E}(\rho_{q'}) \leq w\}$$

  and $\forall q \in Q, q' \in S$,

$$\{\alpha_q \wedge \rho_q = w\} v^+ \{c(q, q') \to \mathbb{E}(\rho_{q'}) \leq w\}.$$

- *Decrease*: $\forall v \in \mathcal{B}, \exists \epsilon > 0, \forall q \in B, q' \in Q$,

$$\{\alpha_q \wedge \rho_q = w \wedge \mathbb{E}(\lambda) = t\} v^- \{c(q, q') \to \frac{\mathbb{E}(\rho_{q'}) - w}{\mu([0, \mathbb{E}(\lambda)))} \leq -\epsilon\}$$

and $\forall q \in Q, q' \in B$,

$$\{\alpha_q \wedge \rho_q = w \wedge \mathbb{E}(\lambda) = t\}v^+\{c(q,q') \rightarrow \frac{\mathbb{E}(\rho_{q'}) - w}{\mu([0, \mathbb{E}(\lambda)))} \leq -\epsilon\}.$$

Proposition 6.2 *Let $\{\alpha_q\}_{q \in Q}$ be invariants for $\mathcal{B}$ and $\mathcal{A}$ and r be a run of $\mathcal{A}$ over a trace $v \in \mathcal{B}$. If $\{\rho_q\}_{q \in Q}$ is a set of Lyapunov functions for $\mathcal{B}$ and $\mathcal{A}$, then*

- $\mathbb{E}(\rho_{r(t_2)}(v(t_2))) \leq \rho_{r(t_1)}(v(t_1))$ *when* $\forall t_1 \leq t \leq t_2, r(t) \in B \cup S$,
- $\frac{\mathbb{E}(\rho_{r(t_2)}(v(t_2))) - \rho_{r(t_1)}(v(t_1))}{\mu([t_1, t_2))} \leq -\epsilon$ *when* $t_1 < t_2$ *and* $\forall t_1 \leq t \leq t_2, r(t) \in B$, *and*
- *if BS is the set of segments of consecutive B and S-states in r, then $\forall q^* \in BS, \mu_B(q^*)$ is finite.*

Proof : For any run r over v and for any segments q^* of r containing only bad states and stables states, $\mathbb{E}(\rho)$ on q^* is non increasing, i.e., let I be the time interval of q^*, for any $t_1 < t_2 \in I$, $\rho_{r(t_1)}(v(t_1)) \geq \mathbb{E}(\rho_{r(t_2)}(v(t_2))$, and the decreasing speed at the bad states is no less than ϵ. Let m be the upper bound on $\{\rho_{r(t)}(v(t)) | t \in I\}$. Since $\rho_q \geq 0$, $\mu_B(q^*) \leq m/(\epsilon) < \infty$. $\Box$

Let $\mathcal{TA} = \langle \mathcal{A}, T, \tau \rangle$. Corresponding to two types of time bound, we define two timing functions. Let $\{\alpha_q\}_{q \in Q}$ be invariants for $\mathcal{B}$ and $\mathcal{A}$. A set of partial functions $\{\gamma_q\}_{q \in T}$ is called a set of *local timing functions* for $\mathcal{B}$ and $\mathcal{TA}$ iff $\gamma_q : A \rightarrow \mathcal{R}^+$ satisfies the following conditions:

- *Boundedness*: $\forall v \in \mathcal{B}, \forall q \in Q, q' \in T$,

  $$\{\alpha_q\}v^-\{c(q,q') \rightarrow \gamma_{q'} \leq \tau(q')\}$$

  and $\forall q \in T, q' \in Q$,

  $$\{\alpha_q \wedge \mathbb{E}(\lambda) = t \wedge \gamma_q = w\}v^-\{c(q,q') \rightarrow w \geq \mu([0, \mathbb{E}(\lambda)))\}.$$

- *Decrease*: $\forall v \in \mathcal{B}, \forall q \in T, \{\alpha_q \wedge \gamma_q = w \wedge \mathbb{E}(\lambda) = t\}v\{c(q,q) \rightarrow \frac{\mathbb{E}(\gamma_q) - w}{\mu([0, \mathbb{E}(\lambda)))} \leq -1\}$.

A set of partial functions $\{\eta_q\}_{q \in Q}$ is called a set of *global timing functions* for $\mathcal{B}$ and $\mathcal{TA}$ iff $\eta_q : A \rightarrow \mathcal{R}^+$ satisfies the following conditions:

- *Definedness*: $\forall q \in Q, \alpha_q \rightarrow \exists w \in \mathbb{R}^+, \eta_q = w$.
- *Boundedness*: $\forall q \in B, \alpha_q \rightarrow \mathbb{E}(\eta_q) \leq \tau(bad)$.
- *Non-increase*: $\forall v \in \mathcal{B}, \forall q \in S, q' \in Q$,

  $$\{\alpha_q \wedge \eta_q = w\}v^-\{c(q,q') \rightarrow \mathbb{E}(\eta_{q'}) \leq w\}$$

  and $\forall q \in Q, q' \in S$,

  $$\{\alpha_q \wedge \eta_q = w\}v^+\{c(q,q') \rightarrow \mathbb{E}(\eta_{q'}) \leq w\}.$$

- *Decrease*: $\forall v \in \mathcal{B}, \forall q \in B, q' \in Q$,

$$\{\alpha_q \wedge \eta_q = w \wedge \mathbb{E}(\lambda) = t\}v^{-}\{c(q,q') \rightarrow \frac{\mathbb{E}(\eta_{q'}) - w}{\mu([0,\mathbb{E}(\lambda)))} \leq -1\}$$

  and $\forall q \in Q, q' \in B$,

$$\{\alpha_q \wedge \eta_q = w \wedge \mathbb{E}(\lambda) = t\}v^{+}\{c(q,q') \rightarrow \frac{\mathbb{E}(\eta_{q'}) - w}{\mu([0,\mathbb{E}(\lambda)))} \leq -1\}.$$

Proposition 6.3 *Let $\{\alpha_q\}_{q\in Q}$ be invariants for $\mathcal{B}$ and $\mathcal{A}$ and r be a run of $\mathcal{A}$ over a trace $v \in \mathcal{B}$. If there exist local and global timing functions for $\mathcal{B}$ and $\mathcal{TA}$, then*

- *if $Sg(q)$ is the set of segments of consecutive q's in r, then $\forall q \in T, q^* \in Sg(q)$, $\mathbb{E}(\mu(q^*)) \leq \tau(q)$, and*
- *if BS is the set of segments of consecutive B and S-states in r, then $\forall q^* \in BS$, $\mathbb{E}(\mu_B(q^*)) \leq \tau(bad)$.*

Proof : Similar to the proofs of Proposition 5.3 and Proposition 6.2. □

The following theorem is a generalization of the soundness and completeness of the set of verification rules.

Theorem 4 *The verification rules (I), (S) and (AT) are sound if the following conditions on $\mathcal{B}$ and $\mathcal{TA}$ are satisfied:*

- *$\mathcal{T}$ is an infinite time structure.*
- *All traces in $\mathcal{B}$ are specifiable by $\mathcal{TA}$.*

The verification rules are complete if the following conditions on $\mathcal{B}$ and $\mathcal{TA}$ are satisfied:

- *$\{\langle v, r\rangle | v \in \mathcal{B}, r$ is a run over $v\}$ is time-invariant.*
- *All transitions from R to non-R-states are left-closed, i.e., if r is a run, and there is a transition from a R-state to a B-state or a S-state at t, then $r(t) \in B \cup S$.*

Proof : Soundness is derived from Propositions 6.1, 6.2 and 6.3. For any trace v, there is a run since v is specifiable by $\mathcal{TA}$. For any run r over v, if any automaton-state in R appears infinitely many times in r, r is accepting. Otherwise there is a time point t_0, the sub-sequence r on $I = \{t \in \mathcal{T} | t \geq t_0\}$, denoted q^*, has only bad and stable automaton states. If there exist a set of invariants and a set of Lyapunov functions, $\mu_B(q^*)$ is finite. Since time is infinite, all the automaton states appearing infinitely many times in r belong to S; r is accepting too. If there exists a set of local and global timing functions, r satisfies the time constraints; r is accepting for $\mathcal{TA}$.

On the other hand, if $\mathcal{TA}$ is valid over $\mathcal{B}$, there exist a set of invariants, a set of Lyapunov functions, and a set of local and global timing functions that satisfy the requirements.

The set of invariants can be constructed as follows: $\forall s \forall q, s \models \alpha_q$ iff the pair $\langle q, s\rangle$ is reachable, i.e., $\exists r, v, t, r(t) = q \wedge v(t) = s$. We shall prove that $\{\alpha_q\}_{q\in Q}$ is a set of invariants.

- Initiality: if $\Theta(s) \wedge e(q)(s)$, $\exists r, v, r(\mathbf{0}) = q$ and $v(\mathbf{0}) = s$. Therefore, $s \models \alpha_q$.

- Inductivity: $\forall v, t$, if $\exists t' < t, \forall t' \leq t'' < t$, $\exists r, r(t'') = q$ ($v(t'') \models \alpha_q$), then $\exists r$, $\exists t'_0 < t, \forall t'_0 \leq t'' < t$, $r(t'') = q$. If $v(t) \models c(q, q')$, then $r(t) = q'$, i.e., $v(t) \models \alpha_{q'}$. Therefore, $v(t) \models c(q, q') \rightarrow \alpha_{q'}$.

- Continuity: $\forall v, t$, if $\exists r, r(t) = q$ ($v(t) \models \alpha_q$), and $\exists t' > t \exists q', \forall t' < t'' < t, v(t'') \models c(q, q')$, $\forall t' < t'' < t, r(t'') = q'$. Therefore, $\exists t' > t, \forall t' < t'' < t, c(q, q') \rightarrow \exists r, r(t'') = q'$.

Given the above constructed invariants, a set of Lyapunov functions can be constructed as follows:

- $\forall q \in R$ and $s \models \alpha_q$, let $\rho_q(s) = 0$.

- $\forall q \notin R$ and $s \models \alpha_q$, the Lyapunov function is defined as follows. For any r, v, t with $r(t) = q$ and $v(t) = s$, let q^* be a segment of r with only bad and stable states starting at q, and $\mu_B(q^*)$ be the measure of B-states in q^*. Let $\rho_q(s)$ be the average measure for all r, v, t with $r(t) = q$ and $v(t) = s$, i.e., $\rho_q(s) = \mathbb{E}(\mu_B(q^*))$.

We shall prove that $\{\rho_q\}_{q \in Q}$ is a set of Lyapunov functions and global timing functions. For $q, q' \notin R$, let $\langle q, s\rangle \prec \langle q', s'\rangle$ iff $\exists r, v, t < t', \forall t < t'' < t', r(t'') \notin R$, $r(t) = q, v(t) = s$ and $r(t') = q'$ and $v(t') = s'$. Since $\{\langle v, r\rangle\}$ is time-invariant, $\prec$ is transitive. Therefore, $\langle q, s\rangle \prec \langle q', s'\rangle$ implies $\rho_q(s) \geq \mathbb{E}(\rho_{q'}(s'))$.

- *Definedness*: $\forall q \in Q, s \models \alpha_q$, ρ_q is defined at s.

- *Non-increase*: $\forall v \in \mathcal{B}, \forall q \in S, q' \in R$,

  $$\{\alpha_q \wedge \rho_q = w\} v^- \{c(q, q') \rightarrow \mathbb{E}(\rho_{q'}) \leq w\}$$

  is trivially satisfied. $\forall q \in S, q' \in B \cup S$,

  $$\{\alpha_q \wedge \rho_q = w\} v^- \{c(q, q') \rightarrow \mathbb{E}(\rho_{q'}) \leq w\}$$

  is satisfied since $\langle q, s\rangle \prec \langle q', s'\rangle$.
  $\forall v \in \mathcal{B}, \forall q \in B \cup S, q' \in S$,

  $$\{\alpha_q \wedge \rho_q = w\} v^+ \{c(q, q') \rightarrow \mathbb{E}(\rho_{q'}) \leq w\}$$

  is satisfied since $\langle q, s\rangle \prec \langle q', s'\rangle$. $\forall q \in R, q' \in S$, $c(q, q')$ is $false$ since all transitions from R to non-R-states are left-closed.

- *Decrease*: $\forall v \in \mathcal{B}, \forall q \in B, q' \in Q$,

  $$\{\alpha_q \wedge \rho_q = w \wedge \mathbb{E}(\lambda) = t\} v^- \{c(q, q') \rightarrow \frac{\mathbb{E}(\rho_{q'}) - w}{\mu([0, \mathbb{E}(\lambda)))} \leq -1\}.$$

  $\forall q \in R, q' \in B$,

  $$\{\alpha_q \wedge \rho_q = w \wedge \mathbb{E}(\lambda) = t\} v^+ \{c(q, q') \rightarrow \frac{\mathbb{E}(\rho_{q'}) - w}{\mu([0, \mathbb{E}(\lambda)))} \leq -1\}$$

is trivially satisfied since $c(q, q')$ is $false$. $\forall q \in B \cup S, q' \in B$,

$$\{\alpha_q \wedge \rho_q = w \wedge \mathbb{E}(\lambda) = t\} v^+ \{c(q, q') \rightarrow \frac{\mathbb{E}(\rho_{q'}) - w}{\mu([0, \mathbb{E}(\lambda)))} \leq -1\}.$$

The local timing functions can be defined similarly. $\Box$

The conditions for the completeness of the rules are imposed so as to be able to define Lyapunov functions for a behaviour and an automaton, as long as the behaviour satisfies the automaton. The second condition for completeness is always satisfied for traces with discrete time structures. More generally, the following proposition may apply.

Proposition 6.4 *All transitions from R to non-R-states are left-closed, if the following conditions are satisfied:*

- *$\mathcal{TA}$ is open and complete.*
- *$\forall q \in R$, $q_1 \notin R$ and $q_2 \in R$, $c(q, q_1) \wedge c(q, q_2)$ is not satisfiable.*
- *All traces in $\mathcal{B}$ are right-continuous.*

Proof : Since $\mathcal{TA}$ is open, $\forall q \in Q, q' \in R$, $c(q, q')$ is open. Therefore, $\forall q \in Q, \bigvee_{q' \in R} c(q, q')$ is open. Since $\forall q \in R$, $q_1 \notin R$ and $q_2 \in R$, $c(q, q_1) \wedge c(q, q_2)$ is not satisfiable, $(\bigvee_{q' \in R} c(q, q')) \wedge (\bigvee_{q' \in B \cup S} c(q, q'))$ is not satisfiable. Since $\mathcal{TA}$ is complete, $\bigvee_{q' \in R} c(q, q')$ and $\bigvee_{q' \in B \cup S} c(q, q')$ are complementary. Therefore, we obtain $\bigvee_{q' \notin R} c(q, q')$ is closed. Since all traces in $\mathcal{B}$ are right-continuous, for all v, t, if t is a limit point to the right time points T, $v(t)$ is a point or a limit point of $v(T)$. If $\exists t' > t, \forall t < t'' < t', v(t'') \in \bigvee_{q' \notin R} c(q, q')$, $v(t) \in \bigvee_{q' \notin R} c(q, q')$. Therefore, all transitions from R to non-R-states are left-closed. $\Box$

These definitions are essential to provide understand of general behaviours of stochastic hybrid dynamical systems. At the present time, however, we have yet to develop an algorithm, either semi-automatic or automatic, based on these rules. Work in progress include the development of such algorithms along with the augmentation of the behavioural constraint verification technique to perform quantitative probabilistic verification.

7 Modeling and Verifying the Elevator

To demonstrate the power of our framework, we augment the elevator system introduced in Section 3 with probabilistic passenger arrivals, which are modeled as a PCN transduction of a Poisson process. Passengers can arrive at any floor to request the use of the elevator. These requests will be granted when they conform to the elevator serving state. Obviously, passenger arrivals have an effect on the current passengers by increasing the time needed for the elevator to service their request.

As an example, we verify the non-trivial behavioural constraint that a passenger request will be serviced on average within $\tau = 40$ time units, regardless of the incoming requests that can occur during that passenger's travel. Furthermore, we would like to

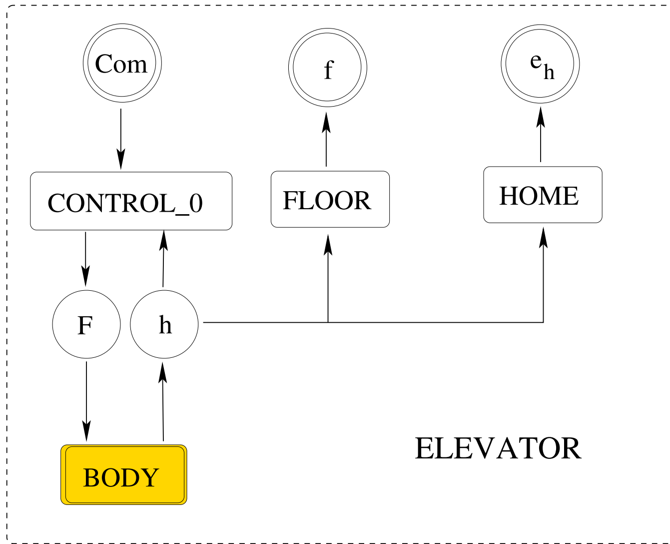


Figure 9: The $Elevator$ module: continuous components of the elevator system.

obtain a probability bound on the time that a request could take to be satisfied. Before we proceed to behavioural constraint verification, let us describe the elevator model in more detail.

7.1 Continuous model

Let us assume that floors are separated by H units. Using the continuous model of the dynamics presented earlier we calculate the current floor number with:

$$f = [h/H] + 1 \tag{4}$$

where $[x]$ denotes the integer value closest to x. Using this relationship, we can get the distance to the nearest floor from: $d_s = h - (f-1)H$. We also say that the elevator is at "home" position, for some $\epsilon > 0$, if:

$$e_h : |d_s| \leq \epsilon. \tag{5}$$

In Figure 9 we present the PCN module of the continuous component of the system. In this diagram, *Com* is a high level command that can take values $1, -1$ and 0, respectively denoting **up**, **down** and **stop**. *CONTROL_0* is an analog controller which determines the force that drives the elevator body *BODY* (see Figure 6). Since the dynamics of the elevator are uncertain, *CONTROL_0* needs to be optimal in some stochastic sense. Finally, the components *FLOOR* and *HOME* are represented by Equations 4 and 5 respectively.

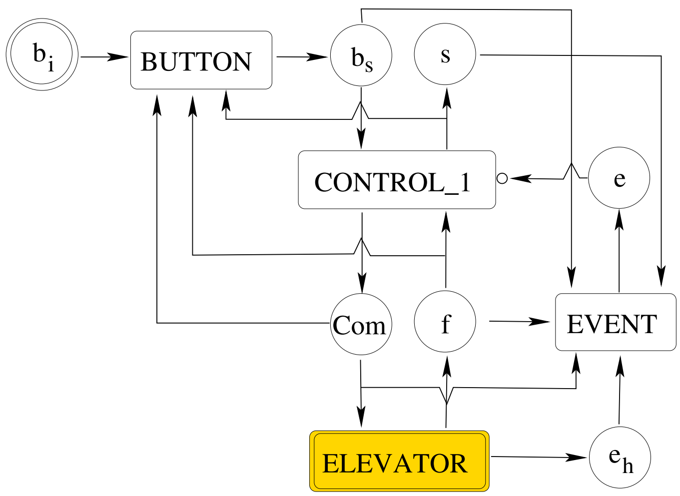


Figure 10: The hybrid model: combining continuous and discrete components of the elevator system.

7.2 Discrete model

An important discrete component of the system is the set of push buttons used by the users to issue requests. Each push button takes value 1 if pushed, and 0 otherwise. In our model, we will consider three different types of buttons: Ub, Db and Fb, which respectively denote up, down, and floor buttons. For an elevator consisting of n floors, we have Ub, Db, $Fb \in \{0,1\}^n$ with $Ub(n) = Db(1) = 0$. The state of a push button b_s is determined by the user's input b_i and the reset signal b_r issued when a request has been served. A floor button will be on until the elevator stops at that floor while a direction button will be on until the elevator stops at the floor and is heading in the corresponding direction. The next state of a push button can be represented as: $b'_s = b_i \vee (\neg b_r \wedge b_s)$.

7.3 Hybrid model

Equipped with a model for the continuous dynamics of the elevator and a model for the user's input, we now need to combine the two to form an hybrid model of the elevator system. A discrete event-driven controller *CONTROL_1* takes as input the current floor f and button states b_s, and outputs the command *Com* and a serving state as displayed in Figure 10.

The events driving the controller are the results of the union of three event spaces: (1) there is a user request when the elevator is idle at floor 1; (2) the elevator reached a home position ($|d_s| \leq \epsilon$); and (3) a request has been served for a given amount of time. Therefore, when any of these events occur, *CONTROL_1* proceeds to an update using the current values of its inputs.

7.4 Control Design

In the previous sections, we referred to *CONTROL_0*, an analog controller generating the force to drive the elevator's body, and to *CONTROL_1*, a discrete controller generating the high level command sent to the elevator. However, we did not define the controllers completely, instead using them as black boxes assumed to perform optimally in some sense. In general, the task of designing optimal controllers is complex and no automatic method exists. Furthermore, remember that we are dealing with uncertainty in the dynamics, which renders the design task harder.

7.4.1 H^∞ control design

Assume that we are interested in finding a simple linear proportional and derivative controller of the form:

$$F = \begin{cases} F_0 & \text{if } \textbf{Com} = 1 \\ -F_0 & \text{if } \textbf{Com} = -1 \\ -K_p d_s - K_v \dot{d}_s & \text{if } \textbf{Com} = 0 \end{cases} \tag{6}$$

where $F_0 > 0$ is a constant force, K_p is a proportional gain and K_v is the derivative gain.

For a controller to be acceptable for our system, it needs to possess continuous (and exponential) stability. Moreover, we require hybrid consistency. That is, the analog controller must interface with the discrete control in a consistent fashion: if a stop command is issued by *CONTROL_1*, then the elevator should continuously maintain $|d_s| \leq \epsilon$, for all time values.

Let us now show that we can design a stabilizing controller. To design the controller for the elevator's body, we chose to apply a robust control design using an H^∞ method [15, 16]. We can rewrite Equation 2 in a mathematically sound fashion by replacing the Gaussian white noise term with Brownian motion and by using the Itô stochastic differential equation:

$$\begin{aligned} dx &= (Ax + B_1 u + B_2 w)dt + HxdW(t) \\ z &= Cx + Du \end{aligned} \tag{7}$$

where $x = [h \ \ \dot{h}]' \in \mathbb{R}^2$, z is called the uncertainty output [16]and $W(t)$ is a scalar Brownian motion process with identity covariance . Furthermore from our elevator model we get

$$A = \begin{bmatrix} 0 & 1 \\ 0 & -1.15 \end{bmatrix}; \quad B_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}; \quad B_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix};$$

$$C = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}; \quad D = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}; \quad F = \begin{bmatrix} 0 \\ -1 \end{bmatrix}; \quad S = \begin{bmatrix} 0 & 1 \end{bmatrix}.$$

We also define $H = \sigma FS$, where $\sigma = 0.15$ as specified in Section 3.

We now apply Theorem 8.2.2 from [16] to obtain a stabilizing controller which solves the H^{∞} problem associated with our stochastic dynamical system. The process of designing a controller based on Theorem 8.2.2 involves solving a set of Riccati equations which can be solved by homotopy method [17] or by a version of Newton's method introduced in [18, 19, 20, 21]. We used the latter for this particular example. We obtained a stabilizing controller of the form (6) with $F_0 = 0.31$, $K_p = 1.1547$ and $K_v = 1.0691$.

To demonstrate that this controller guarantees hybrid consistency, we needed to show that, given the values of F_0, K_p and K_v obtained above, we have $max_t|d_s(t)| \leq \epsilon$ for every possible value of $k(t) \in [0.70, 1.40]$. From the relationship between the variables h and d_s,we can see that $\dot{h} = \dot{d}_s$. Therefore, for *Com*$= 0$, and by combining Equation 2 with Equation 6, we have

$$\ddot{d}_s + (k(t) + K_v)\dot{d}_s + K_p d_s = 0 \tag{8}$$

It is easy to deduce that the maximum distance to a floor D, once *Com*$= 0$, is attained when $\dot{d}_s = 0$. At this point, it is important to notice that Equation 8 can be *critically damped*, *underdamped* or *overdamped* given that $k(t)$ takes values 1.08, $[0.70, 1.08)$ and $(1.08, 1.40]$ respectively. Therefore, we need to analyze the solution of Equation 8 for those three cases separately. Nevertheless, we showed that for each of these three cases, we obtain hybrid consistency.

7.4.2 Discrete control design

At the discrete level, we adopt a control strategy that forces the elevator to move persistently in one direction until there are no more requests in that direction. This ensures that we avoid the presence of dead locks or live locks in the system. We define *CONTROL_0* as a controller which accepts the current request from the push buttons b_s along with the current floor f and state values s and determines: (1) the next command to issue to the elevator module; (2) the updated value of the serving state s. For our system, we assume three different serving states: **up**, **down** and **idle**. The elevator is only idle at the first floor. We consider also three distinct types of binary requests that can be sent to the elevator: **UpRequest, DownRequest** and **StopRequest**, which we define as follows:

$$Ur = UpRequest \quad = \quad Ub(f) \bigvee_{n \geq k > f} (Ub(k) \vee Db(k) \vee Fb(k))$$

$$Dr = DownRequest \quad = \quad Db(f) \bigvee_{1 \leq k < f} (Ub(k) \vee Db(k) \vee Fb(k))$$

$$Sr = StopRequest \quad = \quad \begin{cases} (Db(f) \vee Fb(f)) & \text{if } s = down \\ (Ub(f) \vee Fb(f)) & \text{otherwise} \end{cases}$$

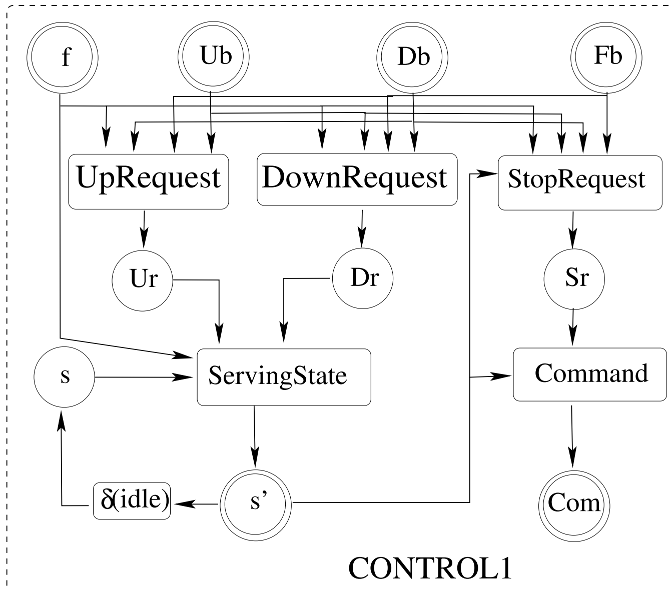


Figure 11: The module of the discrete controller: Control_1

Given these components, we can define the logical expressions for the transition functions for the serving state and the command to the elevator:

$$s' = ServingState(f, s, Ur, Dr) = \begin{cases} up & \text{if } Ur \wedge (s \neq down \vee \neg Dr) \\ down & \text{if } (\neg Ur \wedge (f > 1)) \vee \\ & \quad (Dr \wedge s = down) \\ idle & otherwise \end{cases}$$

$$Com = Command(Sr, s) = \begin{cases} 0 & \text{if } Sr \vee (s = idle) \\ 1 & \text{if } \neg Sr \wedge (s = up) \\ -1 & \text{otherwise} \end{cases}$$

We show in Figure 11 the PCN model of the discrete controller of the elevator, obtained by combining the logical expressions above.

7.5 Example of Behavioural Constraint Verification

Given a simple 3-floor elevator modeled as in Figure 9(b), we can verify whether or not a request to go up from floor 1 to floor 3 will be served within 40 units of the elevator's motion time. This constraint can be represented as in Figure 7(d).

Even though the dynamics of the elevator are continuous, the specification of the behavioural constraints are such that combined with our system's model, we can associate it to the stochastic transition system of Figure 12. The corresponding state space

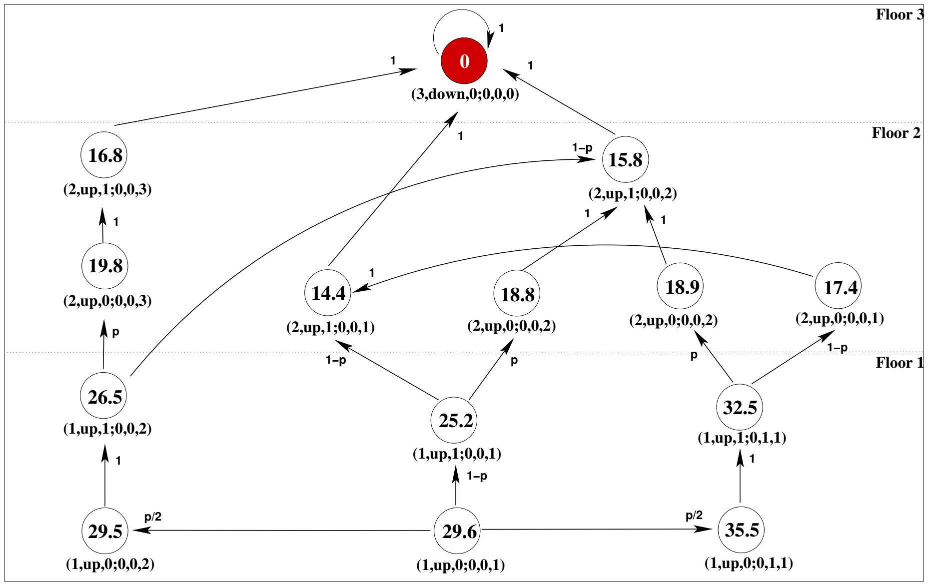


Figure 12: Stochastic State Transition System of the elevator behaviour

$\mathcal{S}$ is of the form $(f, s, Com, N1, N2, N3)$ where f, s and *Com* are the current floor, serving state and command of the elevator. $N1, N2$ and $N3$ denote the number of passengers currently in the elevator wanting to go to floors 1, 2 and 3, respectively. The initial state Θ is set to $(1, up, 0; 0, 0, 1)$. Furthermore, to keep it simple, we assume: (1) only one arrival can occur at any floor; (2) the average time required for someone to get in or out of the elevator is 5 units; and (3) the time to close the elevator doors is 3 units. By simple analysis of the passenger arrival probabilities, we obtain that the probability for the occurrence of a new request for the elevator is $p = 0.15$. Indeed, since we assume that the arrivals follow a Poisson process, and given the fact that the events triggering the discrete controller only happens once the elevator has reached a floor (there is no events possible while the elevator is traveling between floor), we can assume that the arrivals are concentrated at the time when the elevator reaches a new floor. Therefore, we can summarize the probability of arrivals with a single probability, p, obtained from solving $p = Pr(X(t_max) = 1)$, where $X(t)$ is the Poisson process modeling the arrivals and t_max is the maximum traversal time of the elevator from one floor to another. To calculate t_max, we perform a worst case analysis, for all possible values of $k(t)$, on $\dot{h} = (F_0/k(t))(1 - e^{-k(t)t})$ and $h(0) = 0$. we obtain that $h = (F_0/k(t))(t + (1/k(t))e^{-k(t)t}) - F_0/K^2)$. Assume that the distance between two floors if H. Then the time to traverse one level from stationary state will be $t \leq Hk(t)/F + 1/K$. If we assume $H = 2$, we get $t_max = 9.75$ time units.

The evolution kernel $\mathbb{P}$ is represented by the values at the head of the arrows in Figure 12.

To apply the verification rules, we need to:

I: Find a set of invariants for the average-timed $\forall$-automata in Figure 7(d) that satisfies the invariance rules. Let us define $I_B : Fb_s(3) = 1 \wedge \mathit{Com} \neq \text{down}$, $I_S : \mathit{Com} = \text{down}$, and $I_R : Fb_s(3) = 0$. It is easy to see that I_B, I_S and I_R are invariants for states B, S and R, respectively. Note that $R = S$ in our example. In Figure 12, bad states B are denoted by empty nodes while recurrent/stable states R and S are denoted by filled nodes.

S: Find a set of Lyapunov functions. We omit the details but, in conjunction with the invariants above, choosing a function ρ, whose value is the average number of transitions for reaching state R, satisfies the Stability rules.

AT: We can show that choosing a global timing function η which corresponds to the average time to reach R satisfies the average-timeliness rules. For values of η at each state, see the number in the state nodes in Figure 12. It is easy to see that these values satisfy the rules for average-timeliness. Indeed, every value is less then the global bound on bad states τbad and each transition from a state s to another state s' leads to a decrease in value for the global timing function.

We have showed that the average time for a passenger located at floor 1 to be taken to floor 3 is $29.6 < 40$ time units. Hence we have shown that the constraint on the elevator behaviour is satisfied. Note that this is not an absolute bound on the value. The completion time of an instance of the request may exceed 40 time units. With our method, we can automatically obtain probability bounds on the possible time of service. For the elevator example discussed here, we can show that $Pr(\text{time of service} > 90 \text{ time units}) \leq 0.12$.

8 Conclusions and Future Work

We provide a useful framework for constraint-based modeling of the dynamics and behaviours of systems exhibiting uncertainty. The framework abstracts the notions of time and domains for a general approach. We also model uncertainty of several different forms (stochastic, probabilistic, random) while providing a formal semantics. To allow the expression of constraints on both the dynamics and behaviours of systems, we provide two modeling languages: PCN and average-timed $\forall$-automata. PCN is based on a data-flow model and provides a graphical and modular representation which simplifies the task of modeling complex uncertain dynamical systems.

Finally, we provide a set of rules, which in some cases can be fully automated, to verify the satisfaction of global behavioural constraints. We demonstrate the use of the method on an hybrid elevator system exhibiting uncertainty.

References

[1] Zhang, Y., Mackworth, A.K.: Constraint nets: a semantic model for hybrid systems. Theoretical Computer Science **138** (1995) 211–239

[2] Zhang, Y.: A foundation for the design and analysis of robotic systems and behaviours. PhD thesis, University of British Columbia (1994)

[3] Zhang, Y., Mackworth, A.: Specification and verification of constraint-based dynamic systems. In Borning, A., ed.: Princ. and Pract. of Constr. Progr. Number 874 in LNCS. Springer-Verlag (1994) 229–242

[4] Zhang, Y., Mackworth, A.K.: Modeling and analysis of hybrid control systems: An elevator case study. In Levesque, H., Pirri, F., eds.: Logical Foundations for Cognitive Agents. Springer, Berlin (1999) 370–396

[5] Barringer, H.: Up and down the temporal way. Technical report, Computer Science, University of Manchester, England (1985)

[6] Dyck, D., Caines, P.: The logical control of an elevator. IEEE Trans. Automatic Control (1995) 480–486

[7] Sanden, B.: An entity-life modeling approach to the design of concurrent software. Communication of the ACM **32** (1989) 230 – 243

[8] Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. Formal aspects of computing **6** (1994) 512–535

[9] Baier, C., Haverkort, B., Hermanns, H., Katoen, J.P.: Model-checking algorithms for continuous-time markov chains. IEEE Transactions on Software Engineering **29** (2003)

[10] Manna, Z., Pnueli, A.: Specification and verification of concurrent programs by $\forall$-automata. In: 14th Ann. ACM Symp. on Princ. of Progr. Lang. (1987) 1–12

[11] Zhang, Y., Mackworth, A.K.: Specification and verification of hybrid dynamic systems by timed $\forall$-automata. In Alur, R., Henzinger, T., Sontag, E., eds.: Hybrid Systems III. Verification and Control. Number 1066 in LNCS. (1996)

[12] Apt, K., Plotkin, G.: Countable nondeterminism and random assignment. Journal of the Association for Computing Machinery **33** (1986)

[13] Stomp, F., DeRoever, W., Gerth, R.: The μ-calculus as an assertion language for fairness arguments. Technical Report 84-12, Utrecht (1984)

[14] Khilmi, G.: Qualitative Methods in the Many Body Problem. Science Publishers Inc., New York (1961)

[15] Zames, G.: Feedback and optimal sensitivity: Model reference transformations, multiplicative seminorms, and approximate inverses. IEEE Trans. Automat. Control **26** (1981) 301–320

[16] Petersen, I.R., Ugrinovskii, V.A., Savkin, A.V.: Robust Control Design Using H^∞ Methods. Springer (2000)

[17] Richter, S., Hodel, A., Pruett, P.: Homotopy methods for the solution of general modified riccati equations. IEE. Proceedings.-D, Control Theory and Applications **160** (1993) 449–454

[18] Ugrinovskii, V.A.: Robust H^{∞} contorl in the presence of stochastic uncertainty. Int. J. Control **71** (1998) 219–237

[19] Damm, T., Hinrichsen, D.: Newton's method for a rational matrix equation occurring in stochastic control. Linear Algebra Appl. **332/334** (2001) 81–109

[20] Wonham, W.: On a matrix riccati equation of stochastic control. SIAM J. Control **6** (1968) 681–697

[21] Guo, C.H.: Iterative solution of a matrix riccati equation arising in stochastic control. Oper. Theory Adv. Appl. **130** (2001) 209–221