Ann Math Artif Intell (2006) 47:397–425
DOI 10.1007/s10472-006-9035-0

A formal mathematical framework for modeling probabilistic hybrid systems

Robert St-Aubin · Joel Friedman · Alan K. Mackworth

Published online: 9 January 2007

Abstract The development of autonomous agents, such as mobile robots and software agents, has generated considerable research in recent years. Robotic systems, which are usually built from a mixture of continuous (analog) and discrete (digital) components, are often referred to as hybrid dynamical systems. Traditional approaches to real-time hybrid systems usually define behaviors purely in terms of determinism or sometimes non-determinism. However, this is insufficient as real-time dynamical systems very often exhibit uncertain behavior. To address this issue, we develop a semantic model, *Probabilistic Constraint Nets* (PCN), for probabilistic hybrid systems. PCN captures the most general structure of dynamic systems, allowing systems with discrete and continuous time/variables, synchronous as well as asynchronous event structures and uncertain dynamics to be modeled in a unitary framework. Based on a formal mathematical paradigm exploiting abstract algebra, topology and measure theory, PCN provides a rigorous formal programming semantics for the design of hybrid real-time embedded systems exhibiting uncertainty.

Keywords dynamical systems · constraint nets · probabilistic hybrid system · algebraic topology · programming semantics

Mathematics Subject Classification (2000) 68T37

R. St-Aubin (✉) · J. Friedman · A. K. Mackworth
Department of Computer Science, University of British Columbia, 201-2366 Main Mall, Vancouver, BC, Canada
e-mail: staubin@gmail.com

J. Friedman
e-mail: jf@cs.ubc.ca

A. K. Mackworth
e-mail: mack@cs.ubc.ca

1 Introduction

Dynamical systems are defined on time and domain structures. Both of which can be either discrete or continuous. A *hybrid* dynamical system is a dynamical system composed of a combination of discrete and continuous time and domain structures. A robotic system consisting of a computer-controlled robot coupled to a continuous environment is an example of a hybrid dynamical system. Hang and Tamworth proposed a formal framework for deterministic hybrid systems called *Constraint Nets* (CN) [1]. Although their paradigm allows for the modeling of non-deterministic systems through hidden inputs, it does not permit the specification of uncertainty in the system. However, real-time dynamical systems very often behave unpredictably and thus exhibit (structured) uncertainty. It is therefore important to be able to model and analyze real-time probabilistic systems.

In this paper we introduce a sound mathematical framework for the modeling of probabilistic hybrid dynamical systems that we call *Probabilistic Constraint Nets* (PCN). PCN provides a model that is formal and general, modular and composite, powerful and practical. Moreover, PCN has a graphical representation which simplifies the modeling task. Based on algebraic, topological and measure-theoretic structures of dynamics, the PCN framework extends the CN paradigm to allow the user to model uncertainty in the dynamics of the system and in the environment. We will introduce the syntax of the modeling language along with its semantics which leads to a midpoint in distribution.

However, before introducing the syntax of our framework, we present an example of a basic dynamical system that we will use throughout this paper. Consider the discrete time dynamical system corresponding to the following recursive function: $f(t+1) = 0.5\,f(t) + Y(\omega)$, $f(0) = 0$, where $Y(\omega) : \Omega \to \{1, 2\}$ is a random variable with a discrete uniform distribution over the set $\{1, 2\}$. The PCN graphical representation of this system is depicted in Fig. 1. This system will serve as a running example throughout this paper. Figure 1 will be explained when we introduce the PCN syntax in Section 3. We will also use this example to illustrate the semantics of the PCN framework in Section 4.

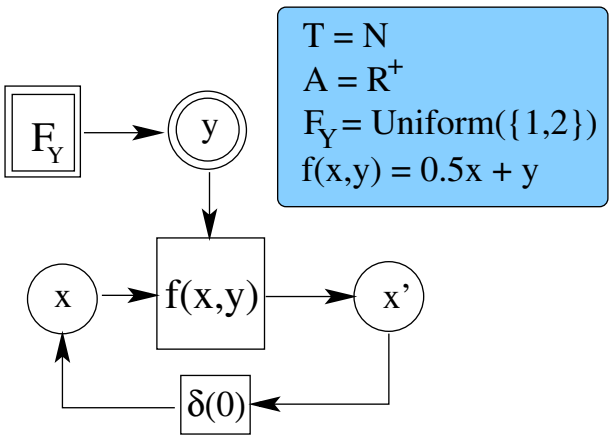

Fig. 1 Simple transduction with uncertainty

1.1 Practical application to dynamical systems: introduction to an elevator system

To further demonstrate the scope of the applicability of our approach, we will analyze an elevator system exhibiting uncertainty. Elevator systems constitute a useful tested for hybrid systems, as confirmed by the fact that they have been used as a benchmark for various approaches to real-time systems [2–5]. 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 [2] 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) and, in [6], provide a methodology for verifying the behavior of this system with respect to a behavioral constraint of real-time response. We refer the reader to [2] for a complete description and analysis of the deterministic hybrid version of this example. For another application of our framework, the reader should consult [7] where we modeled and analyzed a robotic museum surveillance system encompassing uncertainty, on which were imposed constraints on the quality of service.

In Fig. 2, we show the PCN model of the elevator body as represented by a second order stochastic differential equation, based on Newton's second law:

$$\ddot{h} = k(t)\dot{h} + F + w(t) \tag{1}$$

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 with 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.

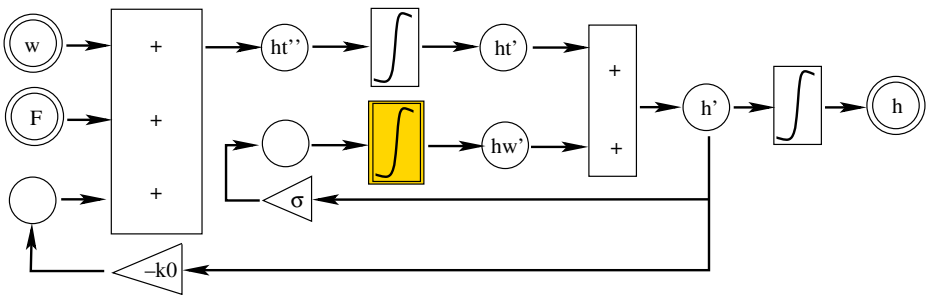

Fig. 2 The *BODY* module of the elevator system

2 Mathematical foundations

In this section, we present the essential mathematical concepts needed to understand the theoretical results for the semantics of the PCN framework. These concepts are based on general topology notions and probability and measure theory. General topology allows us to reason about convergence, connectivity and continuity while probability and measure theory are tools that enable us to reason about integration in arbitrary measure spaces. For a more comprehensive introduction to the mathematical foundations introduced here, the reader is referred to [8–13] while we suggest [14–17] for a more thorough training in measure and probability theory.

As we are interested in modeling dynamical systems, a model of time and its evolution is necessary. In fact, a clear notion of the concept of time is central to understanding dynamics. We formalize time using an abstract structure that captures its most important properties. In general, a time structure can be considered as a totally ordered set with an initial start time, an associated metric for "the distance between any two time points" and a measure for "the duration of an interval of time." Formally, we define the concept of time structure as follows.

Definition 2.1 (Time structure) A *time structure* is a triple $\langle \mathcal{T}, d, \mu \rangle$[1] where

- $\mathcal{T}$ is a linearly ordered set $\langle \mathcal{T}, \leq \rangle$ with $\mathbf{0}$ as the least element;
- $\langle \mathcal{T}, d \rangle$ forms a metric space with d as a metric satisfying: for all $t_0 \leq t_1 \leq t_2$,

$$d(t_0, t_2) = d(t_0, t_1) + d(t_1, t_2),$$

  $\{t | m(t) \leq \tau\}$ has a greatest element and $\{t | m(t) \geq \tau\}$ has a least element for all $0 \leq \tau < \sup\{m(t) | t \in \mathcal{T}\}$ where $m(t) = d(\mathbf{0}, t)$;
- $\langle \mathcal{T}, \sigma, \mu \rangle$ forms a measure space with σ as the Boreal set of topological space $\langle \mathcal{T}, d \rangle$ and μ as a Boreal measure satisfying $\mu([t_1, t_2)) \leq d(t_1, t_2)$ for all $t_1 \leq t_2$ where $[t_1, t_2) = \{t | t_1 \leq t < t_2\}$ and $\mu([t_1, t_2)) = \mu([\mathbf{0}, t_2)) - \mu([\mathbf{0}, t_1))$.

Note that the set of rational numbers $\mathbb{Q}$ with the metric d and the measure μ does not form a time structure. This can be proved using the fact that the set $\mathbb{Q}$ of rationales lacks the least upper bound property stating that if a set S has the property that every nonempty subset of S which has an upper bound also has a least upper bound.

As with time, we formalize domains as abstract structures so that discrete and continuous domains are defined uniformly. A domain can be either simple or composite. Simple domains denote simple data types, such as reals, integers, Boolean and characters; composite domains denote structured data types, such as arrays, vectors, strings, objects, structures and records.

[1] To abbreviate the notation, we will simply use $\mathcal{T}$ to refer to the time structure $\langle \mathcal{T}, d, \mu \rangle$ when no ambiguity arises.

Definition 2.2 (Simple domain) A *simple domain* is 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. Let $\overline{A} = A \cup \{\perp_A\}$. For simplicity, we will use $\overline{A}$ to refer to simple domain $\langle \overline{A}, d_A\rangle$ when no ambiguity arises. For example, let $\mathbb{R}$ be the set of real numbers, $\overline{\mathbb{R}}$ is a simple domain with a connected metric space; let $\mathbb{B} = \{0, 1\}$, $\overline{\mathbb{B}}$ is a simple domain with a discrete topology on $\mathbb{B}$.

Any simple domain $\overline{A}$ is associated with a partial order relation $\leq_{\overline{A}}$. $\langle \overline{A}, \leq_{\overline{A}}\rangle$ is a flat partial order with $\perp_A$ as the least element. In addition, $\overline{A}$ is associated with a *derived metric topology* $\tau = \tau_A \cup \{\overline{A}\}$ where τ_A is the metric topology on A derived from the metric d_A.

A domain is defined recursively based on simple domains.

Definition 2.3 (Domain) $\langle A, \leq_A, \tau\rangle$, with $\leq_A$ as the partial order relation and τ as the derived metric topology, is a *domain*[2] if:

- It is a simple domain; or
- It is a *composite* domain, i.e., it is the product of a family of domains $\{\langle A_i, \leq_{A_i}, \tau_i\rangle\}_{i\in I}$ such that $\langle A, \leq_A\rangle$ is the product partial order of the family of partial orders $\{\langle A_i, \leq_{A_i}\rangle\}_{i\in I}$ and $\langle A, \tau\rangle$ is the product space of the family of topological spaces $\{\langle A_i, \tau_i\rangle\}_{i\in I}$.

We take a signature as a syntactical structure of a class of multi-sorted domains with associated functions defined on these domains. Let $\Sigma = \langle S, F\rangle$ be a *signature* where S is a set of *sorts* and F is a set of *function symbols*. F is equipped with a *mapping type*: $F \to S^* \times S$ where S^* denotes the set of all finite tuples of S. For any $f \in F$, $type(f)$ is the type of f. We use $f : s^* \to s$ to denote $f \in F$ with $type(f) = \langle s^*, s\rangle$. For example, the signature of an algebra on the Naturals can be denoted by $\Sigma_{\mathbb{N}} = \langle \mathbb{N}, \{0, +, -, \times\}\rangle$. This signature has only one sort, $\mathbb{N}$, with 4 different function symbols.

A domain structure of a signature is defined as follows. Let $\Sigma = \langle S, F\rangle$ be a signature. A Σ-*domain structure* A is a pair $\langle \{A_s\}_{s\in S}, \{f^A\}_{f\in F}\rangle$ where for each $s \in S$, A_s is a domain of sort s, and for each $f : s^* \to s \in F$ with $s^* : I \to S$ and $s \in S$, $f^A : \times_I A_{s_i^*} \to A_s$ is a function denoted by f, which is continuous in the partial order topology. For example, $\langle \overline{\mathbb{N}}, \{0, +, -, \times\}\rangle$ is a $\Sigma_{\mathbb{N}}$ structure where $+$, $-$ and $\times$ are addition, subtraction and multiplication, respectively.

With any time structure and domain structure, we can define two basic elements in probabilistic dynamical systems: stochastic traces, which are functions of time and sample space Ω, and transduction, which are mappings from stochastic traces to stochastic traces.

Stochastic traces are a central notion in representing the dynamical behavior of the systems modeled within the PCN framework. A stochastic trace intuitively denotes the (random) changes of values over time. Formally, a *stochastic trace* is a mapping $v : \Omega \times \mathcal{T} \to A$ from sample space Ω and time domain $\mathcal{T}$ to value domain A. For a given $\omega \in \Omega$, the function $v_\omega : \mathcal{T} \to A$ is simply called a *trace*. In the literature, a

[2]For simplicity, we will use A to refer to domain $\langle A, \leq_A, \tau\rangle$ when no ambiguity arises.

trace is often referred to as a sample function, a realization, a trajectory or a path of the underlying stochastic process. We will use v to denote both the stochastic trace v or one of its realization traces v_ω when it is clear from the context and no ambiguity arises.

A stochastic trace v is *well-defined* if $v(\omega, t)$ is well-defined for all $(\omega, t) \in \Omega \times \mathcal{T}$. A stochastic trace v is *undefined* if $v(\omega, t)$ is undefined for any $(\omega, t) \in \Omega \times \mathcal{T}$. For example, denote a Browning motion process by $B_t(\omega)$ and $\mathcal{T} = \mathbb{R}^+$ and $A = \overline{\mathbb{R}}$. Then $v = \lambda\omega, t.B_t(\omega)$ is a well-defined stochastic trace. For a fixed ω in Ω, $v_\omega = \lambda t.B_t(\omega)$ represents a path of the Browning motion process. On the other hand, $v_1 = \lambda t.\cos(t)$ and $v_2 = \lambda t.e^{-t}$ are well-defined deterministic traces, i.e., stochastic traces for which $|\Omega| = 1$.

Due to the fact that physical systems encompass uncertainty, one is often more interested in the distribution of the set of all execution traces of system rather than in one specific execution trace.

One important feature of a trace is that it provides complete information about the current execution of the system of interest at every time point. In the presence of uncertainty, the limiting value of a specific execution trace v_ω is of little interest since the measure of that trace is typically zero. The distribution of a stochastic trace, on the other hand, provides complete information about the probability of the state of the system at every finite time point.

Although trace distribution values at infinite time points are not represented explicitly, they can be derived when limits (in distributions) are introduced. The limiting distribution of a stochastic trace can provide useful information when assessing the behavior of the system in the long run. For example, consider the stochastic trace associated to the system denoted by $f : \Omega \times \mathbb{R}^+ \to \mathbb{R}^+$, where $f(\omega, t) = 1 + B_t(\omega)e^{-t}$, with $B_t(\omega)$ a Browning motion process. For each value of t, one can easily show that f follows a Gaussian distribution with mean 1 and variance te^{-2t} ($F_f = \mathcal{N}(1, te^{-2t})$). The limiting distribution is hence $lim_{t\to\infty}\mathcal{N}(1, te^{-2t}) = \mathcal{N}(1, 0)$, which indicates that in the long run, the system will converge to value 1 and will not fluctuate away from it, despite being influenced by a Browning motion with increasing variance.

A transduction is a mapping from input stochastic traces to output stochastic traces that satisfies the causal relationship between its inputs and outputs, i.e., the output value at any time depends only on inputs up to and including that time. The causal relationship stipulates that the evolution of the system cannot be dictated by the future state of the system, but only by past and present values. Formally, causality can be defined as follows

Definition 2.4 (Causality via $\mathcal{F}_t$ -advisedness) Assume $\{\mathcal{F}_t\}_{t\geq 0}$ to be an increasing family of σ-algebra of subsets of $A^{\Omega\times\mathcal{T}}$. A mapping $F(v)(\omega, t) : A^{\Omega\times\mathcal{T}} \to A'^{\Omega\times\mathcal{T}'}$ is *causal* if $F(v)(\omega, t)$ is $\mathcal{F}_t$-adapted. A causal mapping on stochastic trace spaces is called a *transduction*.

Primitive transduction are defined on a generic time structure $\mathcal{T}$ and are functional compositions of three types of basic transduction: *generators*, *transliterations* and *delays*. We differentiate between *deterministic* and *probabilistic* transduction depending on whether or not they encompass a generator. Compound transduction of each type are built by combining simple transduction of the same type with transliterations and delays. Figure 3 shows the hierarchy of transduction within the PCN framework.

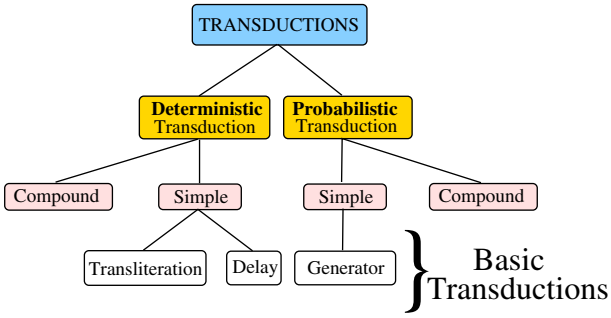

Fig. 3 Transduction type hierarchy

Definition 2.5 (Generator) Let A be a domain, Ω be a sample space and $\mathcal{T}$ a time structure. Moreover, let $F_{X|A}$ denote the (potentially conditional) cumulative distribution function for the random variable X. A *generator* $\mathcal{G}_{\mathcal{T}}^{A}(v_0) : \Omega \times \mathcal{T} \times A \to A$ is a basic transduction defined as

$$\mathcal{G}_{\mathcal{T}}^{A}(v_0, F_X)(v) = \lambda\omega, t. \begin{cases} v_0 & \text{if } t = 0 \\ rand(F_{X|v(\omega,t)}(t), \omega) & \text{else} \end{cases}$$

where $rand(F_{X|A}, \cdot)$ is a random number generator associated with $F_{X|A}$.

We allow the distribution function $F_{X|A}$ to be conditioned on t and values of the systems to produce a general model of uncertainty. This enables the user to model systems where the uncertainty component is non-stationary and conditioned on the state of the system. Also note that in this paper, we are not interested in the simulation of random variables per se, but rather in the analysis of the resulting models. Hence, we will assume that we are given, for each generator included in the model, appropriate random number generators [18–21].

Definition 2.6 (Transliteration) A *transliteration* is a pointiest extension of a function. Formally, let $f : \Omega \times A \to A'$ be a function and $\mathcal{T}$ be a time structure. The pointiest extension of f onto $\mathcal{T}$ is a mapping $f_{\mathcal{T}} : A^{\Omega \times \mathcal{T}} \to A'^{\Omega \times \mathcal{T}}$ satisfying $f_{\mathcal{T}}(v) = \lambda\omega, t. f(v(\omega, t))$.

By this definition, $(f \circ g)_{\mathcal{T}} = f_{\mathcal{T}} \circ g_{\mathcal{T}}$. We will also use f to denote transliteration $f_{\mathcal{T}}$ if no ambiguity arises.

Intuitively, a transliteration is a transformational process without memory or internal state, such as a combinational circuit. Note that in the absence of any random variable within the transliteration, the transformational process is simply a deterministic function of the current input.

Now let us present the last type of basic transduction: delays. There are two types of delay: unit delays and transport delays.

For a given trace, a unit delay $\delta_{\mathcal{T}}^{A}(\omega, v_0)$ acts as a unit memory for data in domain A, given a discrete time structure. We will use $\delta(v_0)$ to denote unit delay $\delta_{\mathcal{T}}^{A}(\omega, v_0)$ if no ambiguity arises.

Definition 2.7 (Unit delay) Let A be a domain, v_0 a well-defined value in A, and $\mathcal{T}$ a discrete time structure. A *unit delay* $\delta^A_{\mathcal{T}}(\omega, v_0) : A^{\Omega\times\mathcal{T}} \to A^{\Omega\times\mathcal{T}}$ is a transduction defined as

$$\delta^A_{\mathcal{T}}(\omega, v_0)(v) = \lambda t. \begin{cases} v_0 & \text{if } t = \mathbf{0} \\ v(\omega, pre(t)) & \text{otherwise} \end{cases}$$

where v_0 is called the *initial output value* of the unit delay.

However, in the presence of non-discrete time structures, unit delays may not be meaningful. Hence we need a transduction that is suitable for more general time structures.

Definition 2.8 (Transport delay) Let A be a domain, v_0 a well-defined value in A, $\mathcal{T}$ a time structure and $\tau > 0$. A *transport delay* $\Delta^A_{\mathcal{T}}(\tau)(\omega, v_0) : A^{\Omega\times\mathcal{T}} \to A^{\Omega\times\mathcal{T}}$ is a transduction defined as

$$\Delta^A_{\mathcal{T}}(\tau)(\omega, v_0)(v) = \lambda t. \begin{cases} v_0 & \text{if } m(t) < \tau \\ v(\omega, t-\tau) & \text{otherwise} \end{cases}$$

where v_0 is called the *initial output value* of the transport delay and τ is called the *time delay*. We will use $\Delta(\tau)(v_0)$ to denote transport delay $\Delta^A_{\mathcal{T}}(\tau)(\omega, v_0)$ if no ambiguity arises. Transport delays are essential for modeling sequential behaviors in dynamical systems.

With preliminaries established, we define an abstract structure of dynamics.

Definition 2.9 (Σ-dynamics structure) Let $\Sigma = \langle S, F\rangle$ be a signature. Given a Σ-domain structure A and a time structure $\mathcal{T}$, a Σ-*dynamics structure* $\mathcal{D}(\mathcal{T}, A)$ is a pair $\langle \mathcal{V}, \mathcal{F}\rangle$ such that

- $\mathcal{V} = \{A^{\Omega\times\mathcal{T}}_s\}_{s\in S} \cup \mathcal{E}^{\Omega\times\mathcal{T}}$ where $A^{\Omega\times\mathcal{T}}_s$ is a stochastic trace space of sort s and $\mathcal{E}^{\Omega\times\mathcal{T}}$ is the stochastic event space;
- $\mathcal{F} = \mathcal{F}_{\mathcal{T}} \cup \mathcal{F}^\circ_{\mathcal{T}}$ where $\mathcal{F}_{\mathcal{T}}$ is the set of basic transduction, including the set of transliterations $\{f^A_{\mathcal{T}}\}_{f\in F}$, the set of unit delays $\{\delta^{A_s}_{\mathcal{T}}(v_s)\}_{s\in S, v_s\in A_s}$, the set of transport delays $\{\Delta^{A_s}_{\mathcal{T}}(\tau)(v_s)\}_{s\in S,\tau>0,v_s\in A_s}$, and the set of generators $\{\mathcal{G}^{A_s}_{\mathcal{T}}\}_{s\in S}$; $\mathcal{F}^\circ_{\mathcal{T}}$ is the set of event-driven transduction derived from the set of basic transduction, i.e., $\{F^\circ | F \in \mathcal{F}_{\mathcal{T}}\}$.

In this body of work, we are interested in modeling the larger class of hybrid probabilistic dynamical systems, that is, systems encompassing components of more than one basic type. Within the PCN paradigm, a probabilistic hybrid dynamical system consists of modules with different time structures, with its domain structure multi-sorted and with a set of probabilistic generators, as basic transduction, which allows for the modeling of the uncertain components of these modules.

To model systems with modules that are associated with different clocks we introduce the notion of *event-driven* transduction. In order to properly introduce the notion of event-driven transduction, we need to define the concept of sample and extension traces. Let $\mathcal{T}_r$ be a reference time of $\mathcal{T}$ with a reference time

mapping h. The *sample stochastic trace* of $v : \Omega \times \mathcal{T}_r \to A$ onto $\mathcal{T}$ is a stochastic trace $\underline{v} : \Omega \times \mathcal{T} \to A$ satisfying $\underline{v} = \lambda\omega, t.v(\omega, h(t))$. The *extension stochastic trace* of $v : \Omega \times \mathcal{T} \to A$ onto $\mathcal{T}_r$ is a stochastic trace $\overline{v} : \Omega \times \mathcal{T}_r \to A$ satisfying

$$\overline{v} = \lambda\omega, t_r. \begin{cases} v(\omega, h^{-1}(t_r)) & \text{if cond} \\ \perp_A & \text{otherwise} \end{cases}$$

where contd $= \exists t \in \mathcal{T}, \mu_r([\mathbf{0}_r, t_r)) \leq \mu([\mathbf{0}, t))$ or $\mu_r([\mathbf{0}_r, t_r)) < \mu(\mathcal{T})$ and $h^{-1}(t_r) = \{t | h(t) \leq_r t_r\} \in \mathcal{T}^\infty$.

Both sampling and extension can be seen as transformational processes on traces, hence they are transduction. *Sampling* is a transduction whose output is a sample trace of its input. *Extending* is a transduction whose output is an extension trace of its input.

An event-driven transduction is a primitive transduction augmented with an extra input which is an event trace; it operates at each event point and the output value holds between two events. This additional event trace input of an event-driven transduction is called the *clock* of the transduction. Intuitively, an event-driven transduction works as follows. First, the input trace with the reference time $\mathcal{T}$ is sampled onto the sample time $\mathcal{T}_e$ generated by the event trace e. Then, the primitive transduction is performed on $\mathcal{T}_e$. Finally, the output trace is extended from $\mathcal{T}_e$ back to $\mathcal{T}$.

Definition 2.10 (Event-driven transduction) Let $\mathcal{T}$ be a time structure and let the mapping $F_{\mathcal{T}} : A^{\Omega \times \mathcal{T}} \to A'^{\Omega \times \mathcal{T}}$ a primitive transduction. Let $\mathcal{E}^{\Omega \times \mathcal{T}}$ be the set of all stochastic event traces on time structure $\mathcal{T}$. The *event-driven transduction* of F is a mapping $F^\circ_{\mathcal{T}} : \mathcal{E}^{\Omega \times \mathcal{T}} \times A^{\Omega \times \mathcal{T}} \to A'^{\Omega \times \mathcal{T}}$ satisfying:

$$F^\circ_{\mathcal{T}}(e, v) = \begin{cases} \lambda t. \perp_{A'} \text{ if } e = \lambda t. \perp_{\mathcal{B}} \\ \\ \overline{F_{\mathcal{T}_e}(\underline{v})} \quad \text{otherwise.} \end{cases}$$

We will use F° to denote event-driven transduction $F^\circ_{\mathcal{T}}$ if no ambiguity arises.

Hence, we can unify, within the same model, modules with different sample time structures generated by event traces. There are two ways in which an event trace can be generated: either with a fixed sampling rate, or by an event generator that reacts to changes in its inputs. Moreover, we can also combine multiple event traces, yielding new event traces. Typically, event traces are combined using event logic which allow various asynchronous components within a given set of modules to be coordinated. Common logical interactions are "event or," "event and," and "event select." With event logic modules, asynchronous components can be coordinated.

We have modeled and analyzed several real world applications within the PCN framework. Such applications include an elevator system with uncertain passenger arrivals, a museum surveillance robot and a package delivery robot. The model for the elevator system is presented in Section 5 while the other models can be found in [6, 7, 22].

3 Syntax of PCN

A probabilistic constraint net consists of a finite set of locations, a finite set of transduction and a finite set of connections. However, in order to be able to handle the uncertainty in the systems that we model, we add an essential component: the *generator*. A generator acts as a random number generator, following a given probability distribution and inducing a random location as its output. Thus, in practice, generators can be represented as discrete (e.g. Poisson, uniform) or continuous (Gaussian, exponential) probability distributions although we will use a general (and formal) measure theoretic definition.

Definition 3.1 (Probabilistic constraint nets) A probabilistic constraint net is a tuple $PCN = \langle Lc, Td, Cn \rangle$, where Lc is a finite set of locations, each associated with a sort; Td is a finite set of labels of transduction (either deterministic or probabilistic), each with an output port and a set of input ports, and each port is associated with a sort; 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.

Intuitively, each location is of fixed sort; a location's value typically changes over time. A location can be regarded as a wire, a channel, a variable, or a memory cell. An output location of a generator will be viewed as a *random* variable.

Each transduction is a causal mapping from inputs to output over time, operating according to a certain reference time or activated by external events. Note that probabilistic transduction are built of at least one basic generator transduction. Every generator 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.

Connections link locations with ports of transduction. A clock is a special kind of location connected to the input event port of event-driven transduction.

A location l is called an *output* location of a PCN if l connects to the output port of a transduction in Td; otherwise, since isolated locations are not allowed it is an *input* location. We will use the notation $I(PCN)$ and $O(PCN)$ to denote the set of input and output locations of a probabilistic constraint net PCN. A probabilistic constraint net is *open* if there exists at least one input location, otherwise it is said to be *closed*.

Another feature of our framework is its graphical representation. A PCN can be represented by a bipartite graph where locations are depicted by circles, transduction by boxes, generators by double boxes and connections by arcs. To differentiate them from deterministic locations, we depict random locations with double circles.

Most commonly used families of probability distributions are parameterized, i.e., one can fully specify a probability distribution by giving values to the parameters of the family. The ability of generators to be dependent on certain locations of the model also greatly simplifies the design task when modeling a complex system for which the various uncertain inputs are not fully known. Indeed, specifying the parameters of a probability distribution is often hard and counter-intuitive. Therefore, a designer could set the parameters of the distribution to some *default*

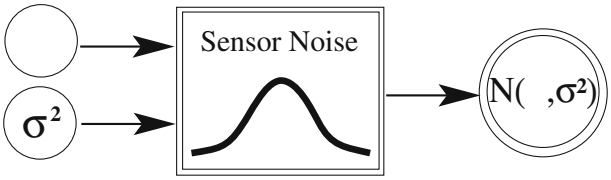

Fig. 4 Gaussian probability distribution as a generator and random location

location value, and then, as the system evolves, learn the values of the parameters of the distribution, thus updating their values as a better estimate is being learned. For example, to model sensor noise with a PCN generator following a Gaussian probability distribution on the discrete time structure $\mathcal{T} = \mathbb{N}$, one would simply need to connect the inputs of the generator to the locations holding the static values of the mean μ and the variance σ^2 to generate samples from the Gaussian distribution at every time point in $\mathcal{T}$ (see Fig. 4).

To exemplify the graphical syntax of PCN further, let us return to the PCN of Fig. 1. In this PCN model, there are three locations (x', x and y), one transduction, one generator and one unit delay. The transduction $f(x, y)$ is a transliteration with two inputs, namely x and y. The unit delay $\delta(0)$ is introduced to eliminate an algebraic loop and the generator F_y follows a discrete uniform distribution over the set $\{1, 2\}$. Hence, the output of the transduction would be a random sequence of values where the value at time $t + 1$ would be half of the value at time t added to either 1 or 2, with equal probability. A possible execution trace resulting from this transduction on $\mathcal{T} = \mathbb{N}$ is $\{0, 1, 2.5, 3.25, 2.625, \ldots\}$. This trace has a measure of 0.0625.

4 Semantics of PCN

We have briefly introduced the syntax of the probabilistic constraint nets model, which has the useful properties of being graphical and modular. However, the syntax does not provide a meaning for the model. Indeed, there are multiple models with similar syntax to probabilistic constraint nets (Petri Nets [23] and their generalization Colored Petri Nets [24] for example) that have completely different interpretations. Therefore, it is necessary to have a formal semantics of probabilistic constraint nets in order to correctly interpret models of complex physical systems.

The midpoint theory of partial order has been used as a semantical model for programming languages and models [9]: in this case, a program (or a model) defines a function f and its semantics are defined to be the least solution of $x = f(x)$, or the least midpoint of f. A similar approach was developed to provide a midpoint semantics for the Constraint Net model [1]. However, even though our framework is similar to that of Constraint Nets, the semantics of PCN differ significantly from that of CN, because we have introduced uncertainty into the set of equations induced by the PCN model. Hence, a probabilistic constraint net is a set of equations with locations serving as variables. Some of the variables (locations) in the equations, those that are outputs of generators, are in fact random variables, obeying some probability distribution, which in turn affect the value of the transduction for which they are inputs. Transduction play the role of functions and the connections between locations and transduction generate a set of equations. Obviously, the semantics of a PCN should be a solution to this set of equations containing random variables.

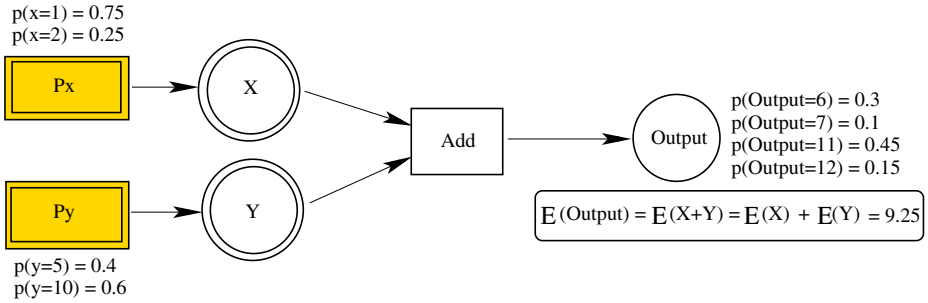

Fig. 5 Simple PCN for a probabilistic sum

Figure 5 demonstrates the effect of random locations on the transduction. Transduction *Add* is a very simple transliteration representing the sum of two (probabilistic) inputs X and Y. It is easy to notice that the output value for this transliteration also follows a probability distribution. In this case, there are four possible values which each have different likelihood of occurrence. One should note that although the distribution of a random variable is helpful in reasoning about its behavior, one can reason about statistics such as the expected value, that is, one can redefine the notion of behavior in terms of *average* behavior for the system. In our simple example, we can see that the average output value of the system is 9.25.

Since the equations in a PCN model do not converge to a midpoint but rather to a stationary distribution, the midpoint theory of partial order cannot be utilized directly to provide a denotational semantics for PCN. In fact, in the presence of uncertainty in the system, the least solution of an equation with random variables is a Marked stochastic process [25].

To further illustrate the difference between the semantics of a deterministic system (CN) and one encompassing uncertainty (PCN), let us 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),[3] at 2 and at 0, as shown in Fig. 6a. The behavior of this system is fully determined by its initial value and it reaches one of the two stable midpoints based on this initial value.

The second system, which cannot be modeled with a constraint net, is stochastically affected by a simple Browning motion process. A sample path for this system, for an initial value of $X_0 = -2$, is shown in Fig. 6b. For this specific realization, the system is initially attracted toward the closest equilibrium which is at $X = 0$. The system then fluctuates around this attractor, reacting under the influence of the Browning 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

[3]There are in fact three different equilibria, at 0, 1 and 2, respectively. However, the equilibrium at 1 is unstable. Any shift in value will cause the system to move away from this unstable equilibrium and move towards one of the other two stable equilibria.

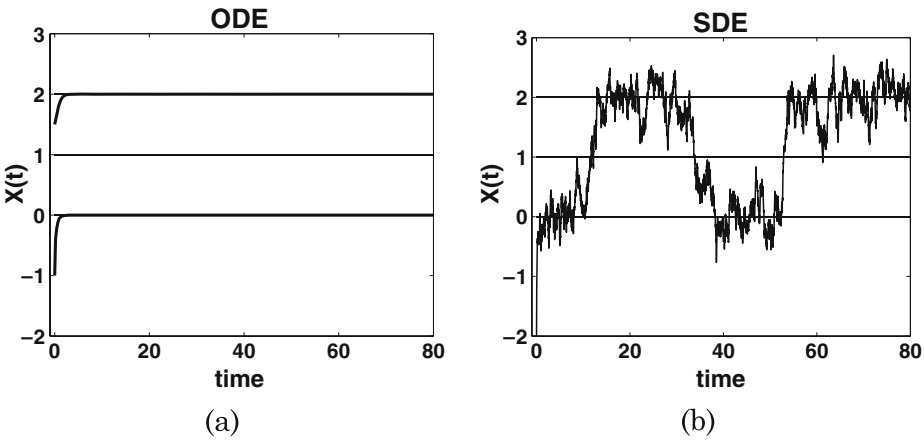

Fig. 6 **a** ODE: $\dot{X}_t = -X_t(X_t - 1)(X_t - 2)$; $X_0 = -1$ and $X_0 = 1.5$; **b** SDE $\dot{X}_t = -X_t(X_t - 1)(X_t - 2) + N_t$; $X_0 = -2$

toward the other equilibrium, at $X = 2$. Another spike 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 its behavior.

In this case, there is no midpoint for this realization nor for the full system. For a set of sample paths with non-zero measure, the system will keep moving back and forth between the two stable equilibria as it is affected by the noise introduced by the Browning motion component of the equation. However, the system will reach a stationary distribution. That is, in the long run, the probability distribution of the system will remain unchanged, independent of time.

The corresponding density function for this distribution is shown in Fig. 7. One can clearly observe that the system is symmetrically distributed with higher weight around the two stable equilibria located at $X = 0$ and $X = 2$. One should note that if the effect of the Browning noise is diminished, the peaks at $X = 0$ and $X = 2$

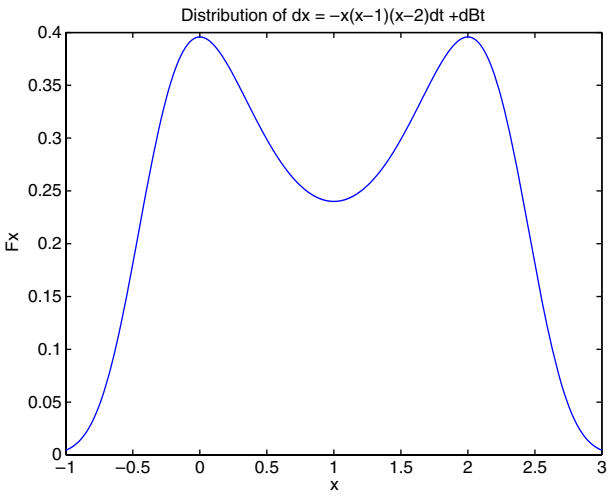

Fig. 7 Density of $dX = -X(X-1)(X-2)dt + dB_t$

rise or fall (depending on the starting value) as the noise is less likely to cause a jump large enough to cause the other equilibrium to become the main attractor. Letting the effect of the noise converge to zero would lead to the deterministic case as presented in Fig. 6a, that is, the stationary distribution would be degenerate everywhere except at the equilibrium corresponding to the initial value of the system. Hence a deterministic system is in fact a special case of the more general stochastic system.

We define the semantics for the Probabilistic Constraint Net model to be the least midpoint of the distribution of the solution to the set of equations of the PCN model. These semantics are, as it was mentioned in the previous paragraph, applicable to any system, whether it be stochastic or deterministic.

4.1 Midpoint in distribution of partial orders

The midpoint theorems used here are for *complete partial orders* (cops). *Continuous* functions are functions which are continuous in partial order topologies. A midpoint in the distribution of a function f can be considered as a solution of the equation $x = f(x)$, where $f(\cdot)$ is an stochastic function. The least midpoint is the least element in the midpoint set.

Definition 4.1 (Midpoint in distribution and Least midpoint) Let $f : \Omega \times A \to A$ be a function on a sample space Ω and a partial order A. A function $g : \Omega \times A \to A$ is a *midpoint in distribution* of f if the distribution of g is a stationary distribution for f. It is the *least midpoint* in distribution of f if, in addition, $F_g \leq F_{g'}$ for every other function g' which is a midpoint in distribution of f. Least midpoints in distribution, if they exist, are unique. The least midpoint in distribution of f will be denoted by $\mu.F_f$.

Based on the above definition, we can state our first midpoint in distribution theorem as follows. The proofs of the following two theorems are shown in the Appendix A.

Theorem 4.1 (Midpoint Theorem I) *Let A be a cpo and assume that either A is also a* total order *or that the set of distributions over A is a cpo and the function over distributions is continuous. Then, every continuous function $f : \Omega \times A \to A$ or g continuous function $f_\omega : A \to A$ (for a fixed $\omega \in \Omega$) has a least midpoint in distribution.*

We now present our second midpoint in distribution theorem which is applicable to a function of two arguments.

Theorem 4.2 (Midpoint Theorem II) *Let A and A' be two cpos and assume that either A, A' are also* total orders *or that the set of distributions over A' is a cpo and the function over distributions is continuous. If $f : \Omega \times A \times A' \to A'$ is a continuous function, then there exists a unique continuous function $\mu.f : \Omega \times A \to A'$, such that for all $a \in A$, the distribution of $(\mu.f)(a)$ is the least midpoint in distribution of $\lambda\omega, x.\, f_\omega(a, x)$.*

Formally, a set of equations can also be written as $\vec{o} = \vec{f}(\vec{\omega}, \vec{i}, \vec{o})$ where $\vec{i}$ is a tuple of input variables and $\vec{o}$ is a tuple of output variables. Based on our previous results, if $\vec{f}$ is continuous, then its least midpoint in distribution is a continuous function, denoted $\mu.\vec{f}$.

4.2 Semantics of probabilistic constraint nets

In this section, we define the midpoint in distribution semantics of probabilistic constraint nets. Let $\Sigma = \langle S, F\rangle$ be a signature and $c \in S$ be a special sort for clocks. A probabilistic constraint net with signature Σ is a tuple $PCN_\Sigma = \langle Lc, Td, Cn\rangle$ where

- Each location $l \in Lc$ is associated with a sort $s \in S$, the sort of location l is written as s_l;
- Each transduction $F \in Td$ is a basic transduction or an event-driven transduction, the sorts of the input and output ports of F are as follows:
  1. If F is a transliteration of a function $f : s^* \to s \in F$, the sort of the output port is s and the sort of the input port i is $s^*(i)$;
  2. If F is a unit delay δ^s or a transport delay Δ^s, the sort of both input and output ports is s;
  3. If F is an event-driven transduction, the sort of the event input port is c, the sorts of the other ports are the same as its primitive transduction;

Let $\mathcal{D}(\mathcal{T}, A) = \langle \mathcal{V}, \mathcal{F}\rangle$ be a Σ-dynamics structure. PCN_Σ on $\langle \mathcal{V}, \mathcal{F}\rangle$ denotes a set of equations $\{o = F_o(\vec{x})\}_{o\in O(PCN)}$, such that for any output location $o \in O(PCN)$,

- F_o is a continuous or g continuous transduction in $\mathcal{F}$ whose output port connects to o,
- $\vec{x}$ is the tuple of input locations of F_o, i.e., the input port i of F_o connects to location $\vec{x}(i)$.

The semantics of a probabilistic constraint net is defined as follows.

Definition 4.2 (Semantics) The *semantics* of a probabilistic constraint net PCN on a dynamics structure $\langle \mathcal{V}, \mathcal{F}\rangle$, denoted $[\![PCN]\!]$, is the least stationary distribution of the set of equations $\{o = F_o(\vec{x})\}_{o\in O(PCN)}$, given that F_o is a continuous or g continuous transduction in $\mathcal{F}$ for all $o \in O(PCN)$; it is a continuous or g continuous transduction from the input trace space to the output trace space, i.e., $[\![PCN]\!] : \times_{I(PCN)} A_{s_i}^{\Omega\times\mathcal{T}} \to \times_{O(PCN)} A_{s_o}^{\Omega\times\mathcal{T}}$.

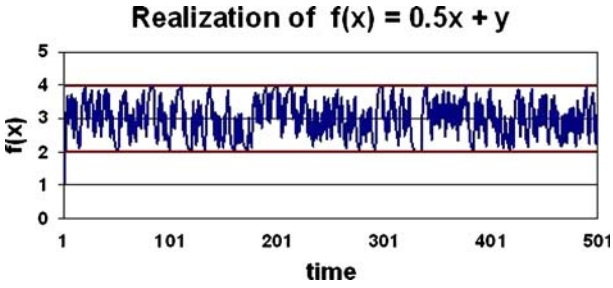

Fig. 8 Sample path of the system $f(x) = 0.5x + y$

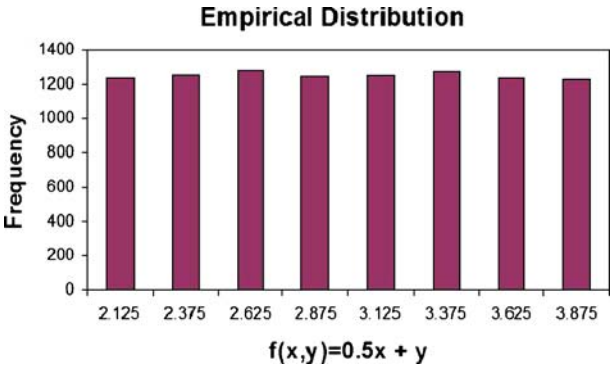

Fig. 9 Empirical Distribution of $f(x) = 0.5x + y$ after 10, 000 time steps

Given any set of output locations O, the restriction of $[\![PCN]\!]$ onto O, denoted $[\![PCN]\!]_{|O} : \times_{I(PCN)} A_{s_i}^{\mathcal{T}} \to \times_O A_{s_o}^{\mathcal{T}}$, is called *the semantics of PCN for O*. For example, consider the probabilistic constraint net denoted by equations $x' = f(x, \omega) = 0.5x + y(\omega)$ and $x = \delta(0)(x)$ with $F_Y = Uniform(\{1, 2\})$ and $\Omega = \{\omega_1, \omega_2\}$ as described in Fig. 1. Given a discrete time structure $\mathbb{N}$, a domain $\overline{\mathcal{I}} = \{1, 2\}$ for inputs and a domain $\overline{\mathcal{O}} = \overline{\mathbb{R}}$ for output, the semantics for x is $F : \overline{\mathcal{I}}^{\Omega \times \mathbb{N}} \to \overline{\mathbb{R}}^{\Omega \times \mathbb{N}}$ such that $F(v)(0) = 0$ and $F(v)(n) = f(F(v)(n-1), v(n-1))$ where the limiting distribution for F is stationary.

Let us show the derivation of the semantics of this model. In Fig. 8, we plot a realization trace of the system, while in Fig. 9 we can see the empirical distribution of the system after 10,000 time steps. The least midpoint distribution follows a uniform distribution over the range $[2, 4]$. The evolution of the distributions is presented in Fig. 10. One can see that the system's distribution starts as uniform over the range $\{1, 2\}$ and the distribution gradually increases to reach a stationary distribution which follows a uniform distribution over $[2, 4]$.

5 Practical application to dynamical systems: augmented model of an elevator system

We augment the elevator system introduced in Section 1.1 with probabilistic passenger arrivals, which are modeled as a PCN transduction of a Poisson process.

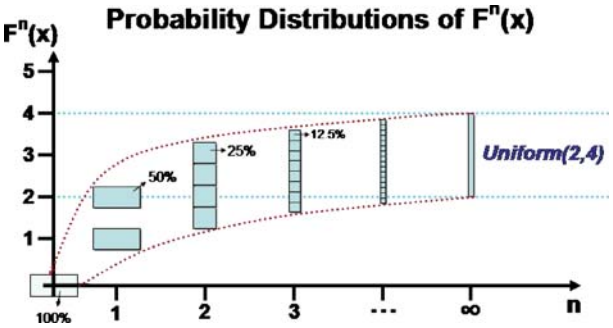

Fig. 10 Evolution of the distributions of f(x)

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.

In the dissertation from which this work originated, we developed sets of automatic and semi-automatic rules to verify behavioral constraints on dynamical systems. As an example, we have verified the non-trivial behavioral 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. These verification rules also allow us to obtain a probability bound on the time that a request could take to be satisfied. Before we briefly present the results from the application of our behavioral constraint verification method, let us describe the elevator model in more detail.

5.1 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:

$$f = [h/H] + 1 \tag{2}$$

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:

$$e_h : |d_s| \leq \epsilon. \tag{3}$$

In Fig. 11 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,

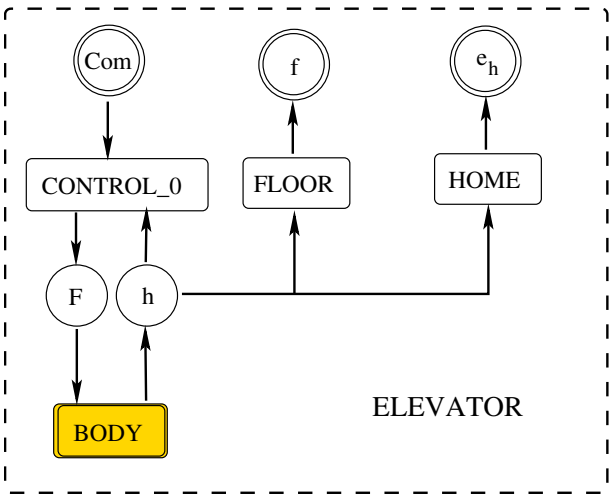

Fig. 11 **a** The *Elevator* module: continuous components of the elevator system; **b** the hybrid model of the elevator

respectively denoting **up**, **down** and **stop**. *CONTROL_0* is an analog controller which determines the force that drives the elevator body *BODY* (see Fig. 2). 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 (2) and (3), respectively.

5.2 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: $b'_s = b_i \vee (\neg b_r \wedge b_s)$.

5.3 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 Fig. 12.

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.

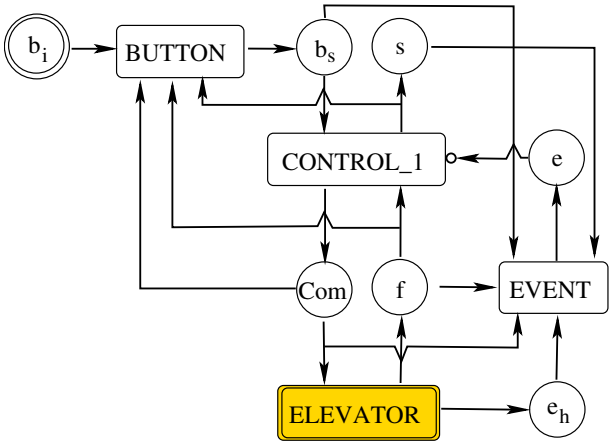

Fig. 12 The hybrid model: combining continuous and discrete components of the elevator system

5.4 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.

5.4.1 H^∞ control design

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

$$F = \begin{cases} F_0 & \text{if } \mathbf{Com} = 1 \\ -F_0 & \text{if } \mathbf{Com} = -1 \\ -K_p d_s - K_v \dot{d}_s & \text{if } \mathbf{Com} = 0 \end{cases} \tag{4}$$

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 [26, 27]. We can rewrite (1) in a mathematically sound fashion by replacing the Gaussian white noise term with Browning motion and by using the Itô stochastic differential equation:

$$\begin{aligned} dx &= (Ax + B_1 u + B_2 w)dt + HxdW(t) \\ z &= Cx + Du \end{aligned} \tag{5}$$

where $x = [h \;\; \dot{h}]' \in \mathbb{R}^2$, z is called the uncertainty output [27]and $W(t)$ is a scalar Browning motion process with identity covariance . Furthermore from our elevator model we get

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

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

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

We now apply Theorem 8.2.2 from [27] 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 Privati equations which can be solved by homotype method [28] or by a version of Newton's method introduced in [29–32]. We used the latter for this particular example. We obtained a stabilizing controller of the form (4) 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 (1) with (4), we have

$$\ddot{d}_s + (k(t) + K_v)\dot{d}_s + K_p d_s = 0 \tag{6}$$

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 (6) can be *critically damped*, *undernamed* or *overlapped* 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 (6) for those three cases separately. Nevertheless, we showed that for each of these three cases, we obtain hybrid consistency.

5.4.2 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: **Unrequested, Consequent** and **Storehouses**, which we define as follows:

$$Ur = UpRequest = Urb(f) \bigvee_{n \geq k > f} (Urb(k) \vee Db(k) \vee Fib(k))$$

$$Dr = DownRequest = Db(f) \bigvee_{1 \leq k < f} (Urb(k) \vee Db(k) \vee Fib(k))$$

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

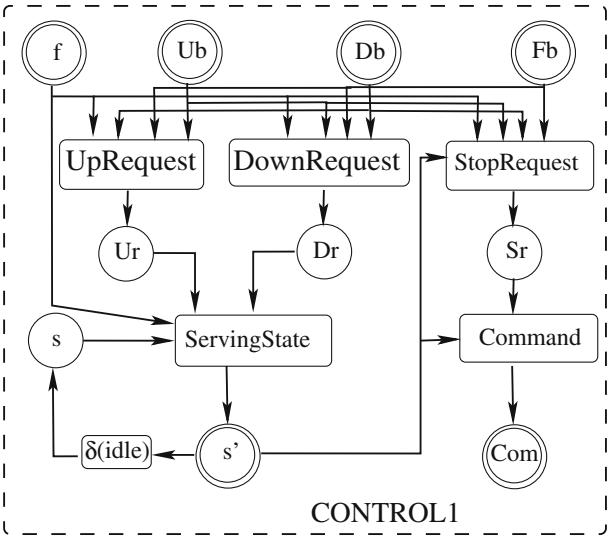

Fig. 13 The module of the discrete controller: Control_1

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

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

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

We show in Fig. 13 the PCN model of the discrete controller of the elevator, obtained by combining the logical expressions above.

5.5 Example of behavioral constraint verification

Given a simple three-floor elevator modeled as in Fig. 11b, 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. Even though the dynamics of the elevator are continuous, the specification of the behavioral constraints are such that combined with our system's model, we can associate it to the stochastic transition system of Fig. 14. The corresponding state space $\mathcal{S}$ 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

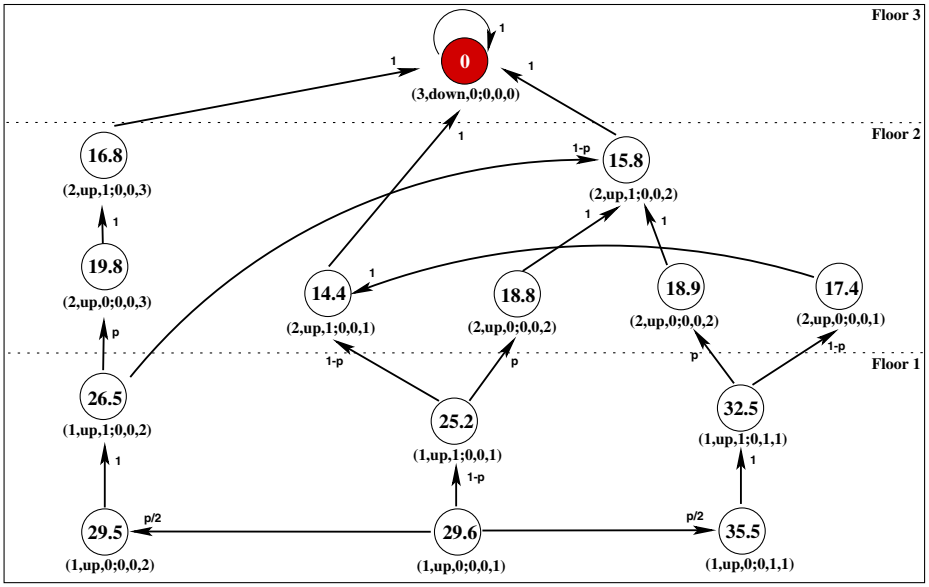

Fig. 14 Stochastic state transition system of the elevator behavior

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 U; and (3) the time to close the elevator doors is 3 U. 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 Fig. 14.

We have showed, in [6], 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 behavior 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 $Pr(\text{time of service} > 90 \text{ time units}) \leq 0.12$.

6 Related work and conclusion

The motivation for developing the PCN framework was to be able to model hybrid dynamical systems while considering the underlying uncertainty in the system. Uncertainty is inherent in any physical system, hence modeling its effects and considering its impact on system behavior is essential. While development of models for hybrids systems has been very active in the last few years [33, 34], there also exist a multitude of paradigms that allow the modeling of uncertain system. Such paradigms include Marked Processes [25], Marked Decision Processes [35] and Dynamic Batesian Networks [36]. However, in most cases, these models are either hybrid and deterministic, or stochastic and restricted to a single time structure (either discrete or continuous). We have shown, in the dissertation associated with this work [6], that PCN subsumes most existing computational models handling uncertainty, and that hybrid, sequential and analog computations can be modeled effectively. The advantages of the subsumption offered by PCN, other than the obvious advantage of parsimony, are many. They include ease of implementation, absence of redundancy while avoiding the system designer having to learn and master multiple paradigms.

Some recent work is very relevant to the PCN approach. In [37, 38] methods are described for efficiently estimating the state of large hybrid systems with, typically, a very large number of discrete modes. This is an important problem, with application to fault diagnosis and repair, that we have not addressed here. Their methods could be fruitfully adopted within the PCN framework. Also relevant is recent work on reactive robot planning as exemplified by [39]. The authors present Probabilistic Hybrid Action Models (PHAMs) as a model of the behaviors generated by reactive plans. They go on to show how PHAMs can be used for online robot planning. Again, online planning is not a focus of our current work. Future work could consider ways these insights could be incorporated into the PCN framework.

In conclusion, we have developed a semantic model for uncertain hybrid dynamical systems, that we call Probabilistic Constraint Nets (PCN). Based on abstract algebra, topology and measure theory, we have represented both time and domains in abstract forms, and uniformly formalized basic elements of dynamical systems in terms of traces, transduction and probabilistic transduction. Furthermore, we have also studied both primitive and event-driven transduction which are important elements of dynamical systems, with or without uncertainty.

Since PCN is an abstraction and generalization of data-flow networks, with the addition that we explicitly handle the uncertain components of the system. Within this framework, the behavior of a system (the semantics of a PCN model) is formally obtained using both the theory of continuous algebra and stochastic systems. Specifically, a probabilistic constraint net models an uncertain dynamical system as a set of interconnected transduction, while the behavior of the system is the set of input/output traces of the system satisfying all the relationships (constraints on the dynamics) imposed by the transduction. PCN models a hybrid system using event-driven transduction, while the events are generated and synchronized within the system.

Complementary work on PCN was performed and led to the development of language specification for behavioral constraints on the dynamics of the systems. Moreover, verification techniques were also developed to allow for the probabilistic verification of the behavioral constraints [7]. A control synthesis approach was also

developed which enables the system designer to synthesize the controller component of a PC model, hence simplifying the modeling task greatly [6].

Acknowledgements The authors gratefully acknowledge discussions with Yin Hang. This work is supported by the Natural Sciences and Engineering Research Council and Precarn Inc.

Appendix A: Proofs of theorems

In order to prove the midpoint theorems introduced in this paper, we need to present the following two propositions:

Proposition A.1 *Let $I \subseteq J$ be an index set. If $f : \Omega \times (\times_I A_i) \to A$ is a continuous or pathwise continuous function, then the* extension *of f, $f' : \Omega \times (\times_J A_J) \to A$ satisfying $f'(\omega, a) = f(\omega, a_{|I})$, is a continuous or pathwise continuous function.*

Proof According to the definitions of continuous functions and product topologies. □

Proposition A.2 *Any continuous* (*or pathwise continuous*) *function is monotonic, i.e., if $f : \Omega \times A \to A'$ ($f_\omega : A \to A'$) is continuous* (*pathwise continuous*)*, then $(\omega_1, a_1) \leq_{\Omega \times A} (\omega_2, a_2)$ $(a_1 \leq_A a_2)$ implies $f(\omega_1, a_1) \leq_{A'} f(\omega_2, a_2)$ $(f_\omega(a_1) \leq_{A'} f_\omega(a_2))$.*

Proof We prove this result for pathwise continuous functions. The result extends easily to continuous functions. Suppose $f_\omega(a_1) \not\leq_{A'} f_\omega(a_2)$, then according to the definition of partial order topology, there is an open set $S \subseteq A'$ including $f_\omega(a_1)$ but not $f_\omega(a_2)$. Therefore, $f_\omega^{-1}(S) \subseteq A$ is an open set including a_1 but not a_2. So $a_1 \not\leq_A a_2$. □

We are now ready to prove both our midpoint theorems.

Theorem 4. 1 *Let A be a cpo and assume that either A is also a* total order *or that the set of distributions over A is a cpo and the function over distributions is continuous. Then, every continuous function $f : \Omega \times A \to A$ or g continuous function $f_\omega : A \to A$ (for a fixed $\omega \in \Omega$) has a least midpoint in distribution.*

Proof

When A is a *total order*

To prove this results, we will use the classic *Tarksi's midpoint theorem* (Lattice-theoretical midpoint theorem) [40]. Let us first introduce the theorem and then show how to use the result to prove our Midpoint Theorem. □

Theorem A.1 (Tarksi's Midpoint Theorem) *Let*

1. *$\mathscr{U} = \langle A, \leq \rangle$ be a complete lattice,*
2. *f be a monotonically increasing function on A to A,*
3. *P be the set of all midpoints of f.*

Then the set P is not empty and the system $\langle P, \leq \rangle$ is a complete lattice.

Proof (Theorem A.1) For the proof of this well-known result the reader is referred to the original work from Tarksi [40]. □

In order to be able to use Tarksi results, we need to show that the set of distributions and its partial order define a complete lattice. Moreover, we also need to show that the function f on the set of distribution is monotonically increasing.

First, denote the set of all distributions on A by $\mathscr{D}$. We formally define a partial order on the set of distributions $\mathscr{D}$. The binary relation $\leq_D$ on $\mathscr{D}$ is defined as follow. Let F_{X_1} and F_{X_2} be distributions of two random variables, namely X_1 and X_2. We write $F_{X_1} \leq_D F_{X_2}$, if $\forall a \in A$, $Pr(X_1 \leq a) \geq Pr(X_2 \leq a)$. It is easy to show that $\leq_D$ induces a partial order on $\mathscr{D}$.

Second, we need to show that for any two distributions $F_1, F_2 \in \mathscr{D}$, there exists a least upper bound and a greatest lower bound. To prove this, let us look at the *cumulative distribution function* (CDs) of each distributions. Here we reproduce Theorem 1.5.1 of [41] and refer to this reference for the proof.

Theorem A.2 (*Theorem 1.5.1 of Caseosa and Buerger*)
The function $F(x)$ is a CDs if and only if the following three conditions hold:

1. $lim_{x \to -\infty} F(x) = 0$ *and* $lim_{x \to \infty} F(x) = 1$.
2. $F(x)$ *is a nondecreasing function of* x.
3. $F(x)$ *is right-continuous. That is, for every number* x_0, $lim_{x \downarrow x_0} F(x) = F(x_0)$.

Proof (Theorem A.2) See p. 30 of Section 1.5 from [41]. □

Since A is a total order, for each $F \in \mathscr{D}$, we have a well defined CDs. Hence every $F \in \mathscr{D}$ possess the three properties of a formal CDs. Based on these properties, it is easy to show that the upper envelope of any set of CDs is a least upper bound (LUG) while the lower envelope of the CDs is the greatest lower bound (GLIB). Moreover, both the LUG and GLIB can be showed to be cumulative distribution functions since they are nondecreasing, right-continuous and converge to 0 and 1 as $x \downarrow -\infty$ and $x \uparrow \infty$, respectively. This demonstrate that we have a complete lattice.

Now let us show that f applied recursively generates a sequence of monotonically increasing distributions. First assume, without loss of generality, that f is Harrovian. Moreover, let us assume that at each transition, the events $\{\omega_1, \cdots, \omega_n\}$ are independent and chosen from the sample space Ω. Order the events $\{\omega_1, \cdots, \omega_n\}$ such that $f_{\omega_1}(a) \leq_A f_{\omega_2}(a) \leq_A \cdots \leq_A f_{\omega_n}(a)$ for any $a \in A$. Let $\perp_A$ denote the least element of A and let F_X denote the distribution of the random variable X.

Now we want to prove that $F_{f^n(\perp_A)} \leq_D F_{f^{n+1}(\perp_A)}$.

Proof by Induction on n:

- For $n = 0$: We have that $F_{f^0(\perp_A)} = \perp_A$
- For $n = 1$: From Proposition A.2, since f_ω is continuous, we have that f_ω is also monotonic, $\forall \omega \in \Omega$. Hence we have $f_\omega(\perp_A) \geq_A \perp_A, \forall \omega \in \Omega$. Therefore, it is trivial to prove that $F_{f^0(\perp_A)} = F_{\perp_A} \leq_D F_{f^1(\perp_A)}$.
- Induction Hypothesis: Assume that for an arbitrary chosen $n \in \mathbb{N}$, $F_{f^n(\perp_A)}$ exists and is well-defined. We now need to show that $F_{f^n(\perp_a)} \leq_D F_{f^{n+1}(\perp_A)}$.

Let $M_1 = max\{f^n(\bot_A)\} = \underbrace{f_{\omega_n} \circ \cdots \circ f_{\omega_n}}_{\text{n times}}(\bot_A)$ Based on this definition, we have $P(f^n(\bot_A) \leq M_1) = 1$.

It is easy to show that $\underbrace{f_{\omega_n} \circ \cdots \circ f_{\omega_n}}_{\text{n times}} \circ f_{\omega_i} \geq M_1$ since $f_{\omega_n} \circ \cdots \circ f_{\omega_n}$ is monotonic.

Hence, we get the following result:

$$\begin{aligned} P(f^{n+1}(\bot_A) \leq M_1) &\leq 1 - \textstyle\sum_{i=1}^{n} P(\text{event } \omega_n \cdots \omega_n \omega_i) \\ &\leq 1 - \underbrace{P(\text{event } \omega_n \cdots \omega_n \omega_i)}_{\text{finite and} >0} \text{ since } \textstyle\sum_{i=1}^{n} P(\omega_i) = 1 \\ &\leq 1 - P^n(\omega_n) \\ &< 1. \end{aligned} \tag{7}$$

Let $M_2 = max\{\{f^n(\bot_A)\} - M_1\}$ be the second highest value after n iterations.

Say that M_2 arose from $f_{\omega^*}(\bot_A)$ where $\omega^* \in \omega_{i_1}\omega_{i_2} \cdots \omega_{i_n}$ with $\omega_i \in \Omega$. Then,

$$\begin{aligned} P(f^n(\bot_A) \leq M_2) &\leq 1 - P(\omega_n \cdots \omega_n) \\ &= 1 - P^n(\omega_n) \end{aligned} \tag{8}$$

Based on the same reasoning as above, we know that $f_{\omega^*} \circ f_{\omega_i}(\bot_A) \geq f_{\omega^*}(\bot_A)$. Therefore, we have

$$\begin{aligned} P(f^{n+1}(\bot_A) \leq M_2) &\leq 1 - \underbrace{P^n(\omega_n)}_{>0} - \underbrace{P(\omega^*)}_{>0} \\ &< 1 - P^n(\omega_n) \end{aligned} \tag{9}$$

By applying this reasoning until $M_n = min\{f^n(\bot_A)\} = \underbrace{f_{\omega_1} \circ \cdots \circ f_{\omega_1}}_{\text{n times}}(\bot_A)$, we get $P(f^n(\bot_A) \leq M_n) = P^n(\omega_1)$.

We know that $\underbrace{f_{\omega_1} \circ \cdots \circ f_{\omega_1}}_{\text{n times}} \circ f_{\omega_i}(\bot_A) \geq \underbrace{f_{\omega_1} \circ \cdots \circ f_{\omega_1}}_{\text{n times}}(\bot_A) = M_n$, which means that $P(f^{n+1}(\bot_A) \leq M_n) = 0$. We have proven that $F_{f^n(\bot_A)} \leq F_{f^{n+1}(\bot_A)}$ for any value of $n \in \mathbb{N}$. Hence, the transformational process on the distributions is a monotonically increasing one.

By applying Tarksi's theorem, we have that the set of midpoints of the distributions is non-empty and is a complete lattice. Therefore, there exist a least midpoint in distribution and it concludes the proof under the assumption that A is a total order. □

When the set of distributions over A, $\mathscr{D}$, is a *cpo* and the function over $\mathscr{D}$ is continuous

To prove this result, we can simply claim the following midpoint theorem on cops:

Theorem A.3 *Let $\langle A, \leq, \bot_A\rangle$ be a cop with least element $\bot_A$. Let $f : \langle A, \leq, \bot_A\rangle \to \langle A, \leq, \bot_A\rangle$ be a continuous function and let $\mu.f$ be the least upper bound of the chain $\{f^n(\bot_A) | n \in \mathbb{N}\}$. Then $\mu.f$ is the* least midpoint *of f.*

Proof (Theorem A.3) The proof can be found in any elementary algebraic theory textbooks such as [9]. □

This concludes the proof of the Midpoint Theorem under the assumption that the set of distributions over A, $\mathscr{D}$, is a *cpo* and the function over $\mathscr{D}$ is continuous since we have satisfied all the necessary assumptions of the midpoint theorem on a *cpo*.

Theorem 4.2 Let A and A' be two *cpo*s and assume that either A, A' are also *total orders* or that the set of distributions over A' is a *cpo* and the function over distributions is continuous. If $f : \Omega \times A \times A' \to A'$ is a continuous function, then there exists a unique continuous function $\mu.f : \Omega \times A \to A'$, such that for all $a \in A$, the distribution of $(\mu.f)(a)$ is the least midpoint in distribution of $\lambda\omega, x.\, f_\omega(a, x)$.

Proof Let $F^0(a) = f(\omega, a, \bot_{A'})$ and $F^{k+1}(a) = f(\omega, a, F^k(a))$. Since f is continuous, it is continuous w.r.t. the third argument. Moreover, a continuous function in any partial order is also monotonic. Therefore, for every a,

$$F^0(a) \leq_{A'} F^1(a) \leq_{A'} F^2(a) \cdots \leq_{A'} F^k(a) \leq \ldots.$$

The proof of the existence of the least midpoint $\mu.f$ is left to the reader as it is very similar to the proof of Theorem 4.1.

Next, we prove that $\mu.f$ is continuous.

Clearly for every k, F^k is continuous since f is continuous and continuity is closed under functional composition. Therefore, for any directed subset D of A,

$$\begin{aligned}
\mu.f\left(\bigvee_A D\right) &= \bigvee_{A'}\left\{F^k\left(\bigvee_A D\right)|k \geq 0\right\} \\
&= \bigvee_{A'}\left\{\bigvee_{A'}\left\{F^k(D)\right\}|k \geq 0\right\} \\
&= \bigvee_{A'}\left\{\bigvee_{A'}\left\{F^k(a)|k \geq 0\right\}|a \in D\right\} \\
&= \bigvee_{A'} \mu.f(D).
\end{aligned}$$

□

Proposition A.1 *Let $I \subseteq J$ be an index set. If $f : \Omega \times (\times_I A_i) \to A$ is a continuous or g continuous function, then the* extension *of f, $f' : \Omega \times (\times_J A_J) \to A$ satisfying $f'(\omega, a) = f(\omega, a_{|I})$, is a continuous or g continuous function.*

Proof According to the definitions of continuous functions and product topologies. □

References

1. Zhang, Y., Mackworth, A.K.: Constraint nets: a semantic model for hybrid systems. Theor. Comput. Sci. **138**(1), 211–239 (1995)

2. 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, pp. 370–396. Springer, Berlin Heidelberg New York (1999)
3. Barringer, H.: Up and down the temporal way. Tech. rep., Computer Science, University of Manchester, England (September 1985)
4. Dyck, D., Caines, P.: The logical control of an elevator. IEEE Trans. Automat. Contr. **3**, 480–486 (March 1995)
5. Sanden, B.: An entity-life modeling approach to the design of concurrent software. Commun. ACM **32**, 230–243 (March 1989)
6. St-Aubin, R.: Probabilistic Constraint Nets: A Framework for the Modeling and Verification of Probabilistic Hybrid Systems. PhD thesis, University of British Columbia, Department of Computer Science, www.cs.ubc.ca/spider/staubin/Papers/StAubin.pdf (June 2005)
7. St-Aubin, R., Mackworth, A.K.: Modeling uncertain dynamical systems and solving behavioural constraints using probabilistic constraint nets. In: ECAI: Workshop on Modeling and Solving Problems with Constraints, pp. 70–85, Valencia, Spain (August 2004)
8. Gemignani, M.C.: Elementary Topology. Addison-Wesley, Reading, MA (1967)
9. Hennessy, M.: Algebraic Theory of Processes. MIT, Cambridge, MA (1988)
10. Vickers, S.: Topology via Logic. Cambridge University Press, Cambridge, UK (1989)
11. Manes, E.G., Arbib, M.A.: Algebraic Approaches to Program Semantics. Springer, Berlin Heidelberg New York (1986)
12. Warga, J.: Optimal Control of Differential and Functional Equations. Academic Press, New York (1972)
13. Royden, H.L.: Real Analysis, 3rd edn. Macmillan, New York (1988)
14. Billingsley, P.: Probability and Measure, Wiley series in probability and mathematical statistics. Wiley, New York (1986)
15. Breiman, L.: Probability. Addison-Wesley, Reading, MA (1968)
16. Williams, D.: Probability with Martingales. Cambridge Mathematical Textbooks, Cambridge (1991)
17. Rudin, W.: Real and Complex Analysis. McGraw-Hill, New York (1966)
18. Bratley, P., Fox, B., Schrage, L.: A guide to simulation, 2nd edn. Springer, Berlin Heidelberg New York (1987)
19. Gentle, J.: Random number generation and Monte Carlo methods. Springer, Berlin Heidelberg New York (1998)
20. Law, A., Kelton, W.: Simulation Modeling and Analysis, 3rd edn. McGRaw-Hill, New York (2000)
21. L'Ecuyer, P.: Handbook of Simulation, ch. 4: Random Number Generation, pp. 93–137. Wiley, New York (1998)
22. St-Aubin, R., Mackworth, A.K.: Constraint-based approach to modeling and verification of probabilistic hybrid systems. Tech. rep. TR-2004-05, University of British Columbia, www.cs.ubc.ca/spider/staubin/Papers/TR-04-05.pdf (April 2004)
23. Peterson, J.L.: Petri Net Theory and the Modeling of Systems. Prentice -Hall, Englewood Cliffs, NJ (1981)
24. Jensen, K.: Coloured petri nets and the invariant-method. Theor. Comp. Sci. **14**, 317–336 (1981)
25. Maruyama, G.: Continuous markov processes and stochastic equations. Rend. Circ. Mat. Palermo **4**, 48–90 (1955)
26. Zames, G.: Feedback and optimal sensitivity: Model reference transformations, multiplicative seminorms, and approximate inverses. IEEE Trans. Automat. Contr. **26**, 301–320 (1981)
27. Petersen, I.R., Ugrinovskii, V.A., Savkin, A.V.: Robust Control Design Using H^{∞} Methods. Springer, Berlin Heidelberg New York (2000)
28. Richter, S., Hodel, A., Pruett, P.: Homotopy methods for the solution of general modified riccati equations. IEE. Proceedings.-D, Control Theory Appl. **160**(6), 449–454 (1993)
29. Ugrinovskii, V.A.: Robust H^{∞} control in the presence of stochastic uncertainty. Int. J. Control **71**(2), 219–237 (1998)
30. Damm, T., Hinrichsen, D.: Newton's method for a rational matrix equation occurring in stochastic control. Linear Algebra Appl. **332/334**, pp. 81–109 (2001)
31. Wonham, W.: On a matrix Riccati equation of stochastic control. SIAM J. Control **6**(4), 681–697 (1968)
32. Guo, C.-H.: Iterative solution of a matrix riccati equation arising in stochastic control. Oper. Theory Adv. Appl. **130**, 209–221 (2001)

33. Maler, O., Manna, Z., Pnueli, A.: From timed to hybrid systems. Real-time: theory in practice in lecture notes in computer science, pp. 448–484 (1992)
34. Nerode, A., Kohn, W.: Models for hybrid systems: Automata, topologies, controllability, observability. In: Grossman, R.L., Nerode, A., Ravn, A.P., Rischel, H. (eds.) Hybrid Systems. Lecture Notes on Computer Science (736), pp. 317–356. Springer, Berlin Heidelberg New York (1993)
35. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley, New York (1994)
36. Murphy, K.: Dynamic Bayesian Networks: Representation, Inference and Learning. PhD thesis, UC Berkeley, Computer Science Division (July 2002)
37. Hofbaur, M., Williams, B.: Hybrid estimation of complex systems. In: IEEE Transactions on Man and Cybernetics, Part B: Cybern. **34**(5), 2178–2191 (2004)
38. Stanislav Funiak, L.J.B., Williams, B.C.: Gaussian particle filtering for concurrent hybrid models with autonomous transitions. J. Artif. Intell. Res. (2007)
39. Beetz, M., Grosskreutz, H.: Probabilistic hybrid action models for predicting concurrent percept-driven robot behavior. In: Artificial Intelligence Planning Systems, pp. 42–61 (2000)
40. Tarksi, A.: A lattice theoretical fixpoint theorem and its applications. Pac. J. Math. **5**, 285–309 (1955)
41. Cassela, G., Berger, R.: Statistical Inference. Wadsworth and Brooks/Cole, Belmont, California (1990)