Modeling Uncertain Dynamical Systems and Solving Behavioural Constraints using Probabilistic Constraint Nets

Robert St-Aubin and Alan K. Mackworth

University of British Columbia
{staubin,mack}@cs.ubc.ca

Abstract. Computer-controlled systems are becoming ubiquitous. Most of these systems exhibit uncertainty, rendering their behaviour somewhat unpredictable. For some systems, this unpredictability can be overlooked. However, for many systems, such as safety critical systems, the uncertainty arising can have dramatic effects. In order to handle such systems, we need a formal modeling framework and a methodology for analyzing them. In systems responding to users requests, for instance, an analysis could be performed to show that a time of service property is satisfied on average. We have developed 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 present verification rules, which have been fully implemented, to perform automatic behavioural constraint verification. Finally, we demonstrate the utility of our framework by applying it to a simple robotic surveillance system.

1 Introduction and Motivation

Using constraints to model systems is a widely studied approach. This approach has lead to the fruitful constraint programming paradigm. However, despite the advances in the Constraint Satisfaction Problem (CSP) framework, one area that still remains to be explored in depth is the constraint-based design of dynamical systems. Dynamical systems cannot be handled in an offline approach, as is often the case for CSP, but rather should be seen as online constraint satisfying systems. Therefore, to model dynamical systems, we must move beyond the typical offline constraint satisfaction model and develop a paradigm where constraints are solved dynamically as the system evolves in time. One proposed solution, Constraint Nets (CN), was developed by Zhang and Mackworth [1, 2]. The approach describes a model for building deterministic hybrid intelligent systems as situated agents. However, real-time dynamical systems commonly behave unpredictably and thus often exhibit (structured) uncertainty. We have augmented the CN framework so that we can model, and reason about, stochastic constraint-based hybrid dynamical systems. We call this new framework the *Probabilistic Constraint Nets* (PCN) model [3, 4]. We provide, within PCN, a

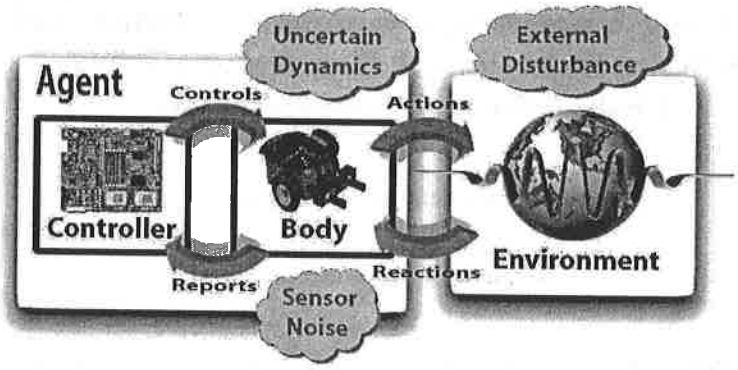


Fig. 1. The structure of a constraint-based agent system

modeling environment based on arbitrary time and domain structures, making it suitable for modeling hybrid systems. The coupled relationship between the constraints on an agent and its (uncertain) environment is shown in detail in Fig. 1. As shown here, we consider three common ways in which the uncertainty components can enter a system: uncertain dynamics, sensor noise and external disturbance.

We model an agent as composed of two distinct constraint-based modules: a *body* with its various sensors and actuators, and a *controller* which is the module that controls the behaviour of the agent. The body senses the uncertain environment and reports to the controller on the perceived state of the environment. In turn, the controller then sends appropriate control signals, based on the updates, 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. The constraints imposed on the system's dynamics represent local behavioural constraints. However, to ensure that a certain system obeys global behavioural constraints, we have also developed a formal behavioural constraint language, *average-timed* $\forall$*-automata*, and a set of verification rules which enable us to perform global behavioural constraint satisfaction for properties of systems, such as *safety*, *goal reachability* and *bounded time response* [3, 4].

In the remainder of this paper, we will show that the solutions to PCN models are, in general, Markov processes. Moreover, for a class of systems where the uncertainty diminishes as the system reaches the equilibrium, we will show that a stochastic equivalent to the property of *monotonicity* of the convergence to the equilibrium can be characterized by stochastic Lyapunov functions. Based on ideas from Lyapunov stability, we will then introduce the global behavioural constraint modeling language, average-timed $\forall$-automata and briefly introduce the rules developed for constraint satisfaction. We will conclude with an application to a mobile robotic surveillance system.

2 Modeling Stochastic Constraints within PCN

Dynamical systems have requirements imposed on their dynamics. For instance, some given restrictions can stipulate how a mobile robot should roam around, or which laws of physics the system should obey. In addition, to define a system completely, one also needs to impose constraints on the system's desired global behaviour such as: a mobile robot should not run into human beings when interacting in a human-populated environment. The set of constraints on the dynamics and the behaviour defines the dynamical system as a whole.

In general, most constraint modeling frameworks fall short of being able to model these constraints in an efficient and sound manner. Moreover, with systems exhibiting uncertainty, the notion of stochastic constraints is usually beyond the capability of existing frameworks. To bridge the gap between such systems and available methodologies, we introduce a formal framework which allows the user to model uncertain dynamics as well as behavioural constraints. 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 in many different forms: probabilistic as in a Markov Chain, or stochastic as with Brownian Motion and stochastic differential equations. The flexibility of our framework allows a designer to model a complex system while remaining under the umbrella of a single modeling language. Let us now introduce the basic syntax of PCN.

2.1 Syntax of PCN

Definition 21 (Probabilistic Constraint Nets) *A probabilistic constraint net PCN 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 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, and associated with a sort. Each probabilistic transduction is associated with a given 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.*

A location can be regarded as a wire, a channel, a variable, or a memory location. Its value changes over time, whether it be based on a certain reference time, or on external events. We say that a location l is an output location of a transduction F if and only if it is connected to an output port of the transduction; otherwise l is an input location. A probabilistic constraint net is *open* if it possesses an input location, otherwise it is *closed*. A PCN can be represented by a bipartite graph where locations are depicted by circles, transductions by boxes (doubled boxes for probabilistic transductions) and connections by arcs. Individual PCN modules are denoted by rounded boxes.

As a running example throughout this paper, we will consider a museum director who wants to use a fleet of mobile robots to survey the different exposition

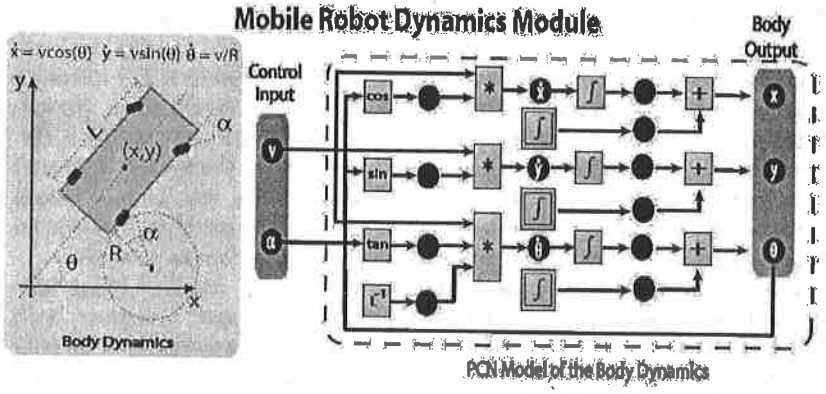


Fig. 2. Dynamics of a mobile robot as a PCN

rooms in the museum. Each robot would offer surveillance in a given subset of the rooms, ensuring that the visitors do not damage, or steal, the art work. A precise continuous model of the (uncertain) dynamics of the mobile robot can easily be modeled in PCN, as shown in Fig. 2, where the double-boxed integral transduction represents an Itô integral, $\int_{t_0}^{t_1} dW_t$, and W_t is a standard Wiener process. In this PCN model, the control inputs v and α dictate the velocity and the direction to the body of the mobile robot, whose position is defined by its (noisy) (x, y) location and its (noisy) orientation θ.

2.2 Topological Approach to Stochastic Dynamic Systems

The PCN framework is built upon the topological structure of time, domains, traces and transductions; each of which are basic elements of stochastic dynamic systems. Let us now present the elementary definitions.

Time structures One important concept in the modeling of dynamical systems is the notion of time. Dynamical systems evolve over time, some in discrete increments while others do in continuous or event-based fashion. Since we are interested in modeling and analyzing systems with discrete and continuous components, we need to formalize time using abstract structures which capture the important aspects of time: linearity, metric and measure. A time structure, in general, can be considered as a linearly ordered set with a start time point, an associated metric for quantifying the distance between any two time points and a measure for estimating the duration of time. Formally, a *time structure* is a triple $\langle \mathcal{T}, d, \mu \rangle$, 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: $\forall t_0 \leq t_1 \leq t_2$, $d(t_0, t_2) = d(t_0, t_1) + d(t_1, t_2)$,
- $\langle \mathcal{T}, d, \mu \rangle$ forms a measure space with μ as a Borel measure.

Although our definition of time structures is general, discrete or continuous time structures are most commonly used.

2.3 Domain structures

Similarly to time, we formalize domains as abstract structures. This approach allow us to uniformly define discrete as well as continuous domains. We consider two main types of domains: (1) a simple domain or, (2) a composite 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, and d_A is a metric on A. A *composite* domain is the product of a family of domains $\{\langle A_i \cup \{\perp_{A_i}\}, d_{A_i} \rangle\}_{i=1}^{n}$.

2.4 Trace and event structures

A *trace* intuitively denotes changes of values over time, characterized by the set of input and output locations. Formally, a trace is a mapping $v : \mathcal{T} \times \Omega \rightarrow A$ from time $\mathcal{T}$ to a domain A, where Ω is the event space of the system under study. Ω may be omitted when no ambiguity arises. For example if $X_t(\omega), \omega \in \Omega$ is a random variable defined on a proper probability space, $\mathcal{T} = \mathbb{R}^+$, and $A = \mathbb{R}$, $v_1 = \lambda t. \cos(X_t(\omega))$ and $v_2 = \lambda t. e^{-X_t(\omega)}$ are traces. 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. Any event trace generates a sample time structure which is semi-discrete and well-defined [1].

2.5 Deterministic and Probabilistic Transductions

Deterministic transductions are causal mapping from input traces to output traces over time: $F : \mathcal{A}_1^{\mathcal{T}_1} \rightarrow \mathcal{A}_2^{\mathcal{T}_2}$, where $\mathcal{A}^{\mathcal{T}}$ denotes a trace space. Transductions are activated either by a certain reference clock or by external events. By *causal* mapping we mean that the output at any time depends only on inputs up to that time, that is, the future cannot influence the present or the past. In order to model the uncertainty in systems, we introduce the notion of *probabilistic* transductions. A probabilistic transduction acts as a random number generator, following a given probability distribution. In practice, probabilistic transductions can be represented as discrete (e.g. Poisson, uniform) or continuous (Gaussian, exponential) probability distributions. Within PCN, we consider two distinct classes of transductions: *primitive* transductions and *event-driven* transductions. Primitive transduction are composed of two basic types of transductions: transliteration and delays.

For a time structure $\mathcal{T}$, a *transliteration* $f_{\mathcal{T}} : \mathcal{A}_1^{\mathcal{T}_1} \rightarrow \mathcal{A}_2^{\mathcal{T}_2}$ is the pointwise extension of the function $f : \mathcal{A}_1 \rightarrow \mathcal{A}_2$. Intuitively, transliterations can be seen as

a transformational process without internal state (memory). A simple example of a transliteration is a combinational circuit without delay.

A *delay* on discrete time structure is a process where the output value at time t is the input value at time $t-1$. The extensions to transport delays, i.e., delays on non-discrete time structures, is straightforward. Intuitively, delays can be seen as a unit memory. Note that although transliterations can be basic probabilistic transductions, delays are only well-defined as deterministic transductions.

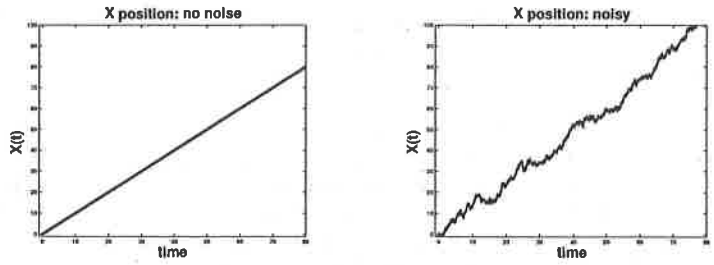
3 Semantics of PCN

So far, we have introduced the syntactical structure of probabilistic constraint nets. However, although syntax is useful in creating a model, it does not provide a meaning to this model. Therefore, we need to provide a proper semantics for our modeling framework. As mentioned earlier, transductions are mappings from input traces to output traces. By observing the relationship between each transduction and its set of input/output locations, we can see that a PCN $\mathcal{P}$ denotes a set of equations $\mathbf{o} = \mathbf{F}(\mathbf{i}, \mathbf{o})$ where each function $\mathbf{F}$ corresponds to a transduction of $\mathcal{P}$ and each variables corresponds to a location in $\mathcal{P}$. Hence, the semantics of the PCN $\mathcal{P}$ is the *solution* of the set of equations, which we denote $[\![\mathcal{P}]\!]$.

Although the PCN framework allows for the modeling of almost any type of uncertainty, in order to guarantee that we obtain well defined systems at all times we need to impose some restrictions on the choice of uncertainty models. One such restriction is the *non-Zeno* property. This condition is necessary to prevent degenerate systems which produce an infinite number of transitions within a finite interval of time. Moreover, we will assume that when modeling systems with (stochastic) differential equations, the noise term is to be modeled via a Wiener process (Brownian motion). In fact, this is not really a restriction since the Wiener process is the only stochastic process, providing an adequate model of noise, which has continuous sample paths [5].

Example 1. To illustrate the difference between a deterministic system and a perturbed system, we compare in Fig. 3, the two dynamical systems with nominal component $\dot{X}_t = v\cos(\theta)$, representing the x position of the mobile robot in the museum surveillance example. For simplicity, we assume that the control signals v (velocity) and θ (direction) are constant. Fig. 3(a) show a noise-free system. In this case, we can see that the system is moving forward linearly, with slope $v\cos(\theta)$. In Fig. 3(b) we show what happens for a system affected by a white noise. This system, although moving forward on average, moves unpredictably as its progress is retarded or advanced by the noise.

We can show, under the assumptions of non-Zenoness and Brownian motion noise, that PCN models generate equations whose solutions are stochastic process, namely Markov processes. For PCN models on discrete or event-based time structure and with finite memory, it is straightforward to show that the generated processes will either be Markov chains or Markov processes, depending on



(a) (b)

Fig. 3: (a) ODE: $\dot{X}_t = vcos(\theta)$; $X_0 = 0$
(b) SDE: $\dot{X}_t = vcos(\theta) + N_t$; $X_0 = 0$

the nature of the domains of the locations of the models. The semantics of the PCN framework is related to the concept of *labelled Markov Processes* as introduced in [6]; however, those authors adopt a higher level modelling approach and focus on the notion of bisimulation for such models.

To prove the result for continuous time structures, we need to use the following result on measurability of PCN components. Results are stated here without proof.

Theorem 1. *Let $\mathcal{T}$ a time structure and let $\mathcal{F}$ be the set of all transductions for a given PCN, then transliterations, transport delays and event-driven transductions in $\mathcal{F}$ are Borel-measurable. Furthermore, if $\mathcal{T}$ is discrete, then all transductions in $\mathcal{F}$ are Borel-measurable.*

Combining the result of Theorem 1 with the result arising from the well-known *Existence and Uniqueness Theorem* for stochastic differential equations [7, 8], we obtain the main result of this section, which states that the solution of any PCN model of stochastic differential equations is a Markov process. Furthermore, with this result, we can show that if the transductions in $\mathcal{F}$ are time-invariant, then the resulting Markov process is also a stationary process [7].

Theorem 2 (PCN Semantics as Markov Process). *Let $\mathcal{P}$ be a PCN model whose set of equations corresponds to*

$$dX_t = f(t, X_t)dt + G(t, X_t)dW_t, X_{t_0} = c, t_0 \leq t \leq T < \infty, \tag{1}$$

where W_t is an $\mathbb{R}^m$-valued Wiener process and c is a random variable independent of $W_t - W_{t_0}$ for $t \geq t_0$. If the $\mathbb{R}^d$-valued function $f(t,x)$ and the $(d \times m$ matrix$)$-valued function $G(t,x)$ are defined and measurable on $[t_0, T] \times \mathbb{R}^d$ and have the following properties:

(a) Lipschitz condition: $\exists K > 0$, a constant, such that $\forall t \in [t_0, T], x \in \mathbb{R}^d, y \in \mathbb{R}^d$, $|f(t,x) - f(t,y)| + |G(t,x) - G9t, y)| \leq K|x - y|$;
(b) Restriction on growth: $\exists C > 0$, a constant, such that $\forall t \in [t_0, T]$ and $x \in \mathbb{R}^d$ $|f(t,x)| + |G(t,x)| \leq C(1 + |x|)$,

then $[\![\mathcal{P}]\!] = X_t$ *is the unique t-continuous solution of Equation 1. Moreover,* $[\![\mathcal{P}]\!]$ *is a Markov process on* $[t_0, T]$ *whose initial probability distribution at the instant* t_0 *is the distribution of c [7, 8].*

Example 2 (Example 1 revisited). Since deterministic systems are special cases of stochastic systems, it is not surprising that the solution to the system of Fig. 3(a) is a Markov process. However, the lack of uncertainty in the dynamics yields, at each instant t, a probability distribution concentrated at value $X_0 + tvcos(\theta)$, with zero variance. On the other hand the stochastic system of Fig. 3(b) has for solution a Markov process, which is in fact a Gauss-Markov process, with a Gaussian probability distribution of the form $\mathcal{N}(X_0 + tvcos(\theta), t)$.

3.1 Stability of Stochastic Dynamical Systems

An important notion of stability for stochastic systems is *Lyapunov* stability [9]. This type of stability can be obtained when dealing with systems whose uncertainty "dies down" as the system evolves in time. Such systems are very common in real life application. For example, remember the mobile robot whose actuators are noisy. The uncertainty induced by the actuators will affect the robot's travel but this effect will gradually diminish as the robot's speed approaches 0.

Let us consider the general class of PCN inducing stochastic differential equations of the form of Equation 1, for which we assume, without loss of generality, that the unperturbed system has an equilibrium at $x = 0$. We can show that if the conditions:

1. the transductions $G \in \mathcal{F}$ affecting the noise term satisfy the condition: $\int_{t_0}^{\infty} |G(s)|^2 ds < \infty$,
2. there exist a (Lyapunov) positive-definite function $v(t, x)$, continuously differentiable with respect to t and twice continuously differentiable with respect to the components x_i of x
3. $Lv(t, x) \leq 0, t \geq t_0, 0 < |x| \leq h$, where $L = \frac{\partial}{\partial t} + \sum_{i=1}^{d} f_i(t, x) \frac{\partial}{\partial x_i} + 1/2 \sum_{i=1}^{d} \sum_{j=1}^{d} (G(t, x)G(t, x)')_{ij} \frac{\partial^2}{\partial x_i \partial x_j}$,

are satisfied, then the equilibrium of Equation 1 is stochastically stable. Stochastic Lyapunov stability of a system X_t relies on the property that the distance, measured by $v(t, X_t)$, of X_t from the equilibrium does not increase on the average. This concept along with the notion of Lyapunov functions, are central to the behavioural constraint satisfaction method that we present in the next section.

4 Behavioural Constraints

Given the semantics presented above, the online satisfaction of the constraints on the dynamics of a stochastic dynamical system ensures that its behaviour is a stochastic process that is a solution to the set of equations it generates. Such constraint satisfaction does not, however, guarantee that the behaviour of

the system will satisfy global temporal constraints. For example, consider once again the dynamics of our mobile surveillance robot. Although the robot will travel according to the laws of motion, it does not guarantee that it won't hit or hurt people during its travels. Such restrictions constitute global constraints on the behaviour of the system and cannot easily be represented within the PCN modeling language. Hence, we introduced a behavioural constraint language, *average-timed ∀-automata*, that allows us to represent global constraints on a system's behaviour [3, 4]. Together with PCN, these two languages will specify the entire set of constraints on a stochastic dynamical system.

Before formally discussing the notion of behavioural constraint satisfaction, 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 $\langle I, O\rangle$ is the tuple of input and output locations of the module. Formally, an input/output pair $\langle i, o\rangle$ is an *observable trace* of $\mathcal{P}(I, O)$ iff $\exists F \in (Td \cup Tp)$ such that $o = F(i)$. We define the *behaviour* $\mathcal{B}$ of $\mathcal{P}(I, O)$ as the set of all observable traces.

For the purpose of this paper, we will restrict ourselves to time-invariant Markovian behaviours over discrete-time structures. A generalization of this approach is available in [3]. Any discrete-time, time invariant, Markovian stochastic dynamical system corresponds to what we call a stochastic state 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 $\Theta : \mathcal{S} \to [0, 1]$ represents the distribution of the initial state of the system. For any discrete time $\mathcal{T}$, $v : \mathcal{T} \to \mathcal{S}$ is a trace of $\langle \mathcal{S}, \mathbb{P}, \Theta\rangle$ iff $\forall t > 0, \mathbb{P}(v(pre(t)), v(t)) > 0$ and $\Theta(v(0)) > 0$, where $pre(t)$ is 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 $\langle \mathcal{S}, \mathbb{P}, \Theta\rangle$.

4.1 ∀-Automata

An important method for representing behavioural constraints (or requirements specification) of systems is ∀-automata or, more precisely, the languages accepted by a ∀-automata. This class of methods is well suited to the PCN framework since we can view traces as a generalization of infinite sequences. A desired property of the system (hence its traces) can be specified by an automaton; a trace of a system satisfies the behavioural constraints iff the associated automaton accepts the trace. ∀-automata was proposed in [10, 11] as a specification language for concurrent programs and deterministic dynamical systems, respectively.

Formally, a *∀-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 $(q, q') \in Q \times Q$, we associate a state proposition

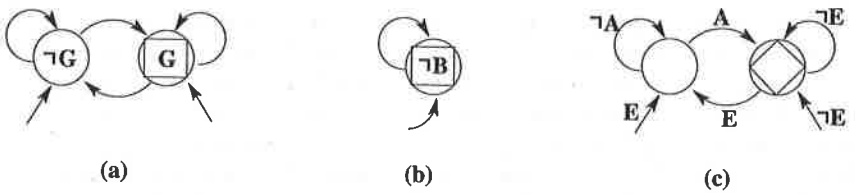


Fig. 4. ∀-Automata Constraints: (a) goal achievement (b) safety (c) bounded response

$c(q, q')$, which characterizes the *transition condition* under which the automaton may move from q to q'. R and S are generalizations 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 $v : \mathcal{T} \to A$ be a trace. A *run* of $\mathcal{A}$ over v is a mapping $r : \mathcal{T} \to Q$ such that (1) $v(\mathbf{0}) \models e(r(\mathbf{0}))$; and (2) for all $t > \mathbf{0}$, $v(t) \models c(r(pre(t)), r(t))$, where $\phi \models \psi$ denotes that ϕ satisfies ψ.

If r is a run then let $Inf(r)$ denote the set of automaton states which appears infinitely often in r. Let $\mathcal{A}$ be a ∀-automaton. A run r over $\mathcal{A}$ is defined to be *accepting* iff it satisfies at least 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 specifies that the system either always eventually returns to the set of recurrent states R or eventually remains forever within the stable set S. Formally, a ∀-automaton $\mathcal{A}$ *accepts* a trace v, written $v \models \mathcal{A}$, iff *all* possible runs of $\mathcal{A}$ over v are accepting.

Let us now consider some typical behavioural constraints for dynamical systems, where stable and recurrent states in Q are marked with $\Box$ and $\Diamond$, respectively. In Fig. 4(a) we represent a goal satisfaction constraint which accepts the traces of a system that will eventually always satisfy the goal condition G. Fig. 4(b) is a global safety constraint which states that an acceptable system should never satisfy the *bad* condition B. Fig. 4(c) is a constraint of bounded response. It states that whenever event E occurs, the response A will follow in bounded time.

4.2 Average-timed ∀-automata

Meaningful behavioural constraints on dynamical systems often include temporal components such as deadlines and real-time response. Since we are interested in solving behavioural constraints on stochastic systems, we usually cannot reason about satisfying a given time constraint perfectly. Rather, we need to reason

about satisfying time constraints on average and thus we proposed the notion of *average-timed* $\forall$*-automata* [3, 4].

Average-timed $\forall$-automata are $\forall$-automata augmented with timed automaton states and average time bounds 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 denoted by a positive real number indicating its average time bound, and $\tau : T \cup \{bad\} \rightarrow \mathbb{R}^+ \cup \{\infty\}$ is an *average* timing function.

A run r is *accepting* for $\mathcal{ATA}$ iff

1. r is accepting for $\mathcal{A}$ and
2. r satisfies the average time constraints. If $I \subseteq \mathcal{T}$ is an interval of $\mathcal{T}$ and $q^* : I \rightarrow Q$ is a segment of run r, where $q^* = r_{|I}$, let $\mu(q^*)$ denote the measure of q^*, i.e., $\mu(q^*) \doteq \mu(I) = \sum_{t \in I} \mu(t)$, since I is discrete. Let $Sg(q)$ be the set of segments of consecutive q's in r, i.e., $q^* \in Sg(q)$ implies $\forall t \in I, q^*(t) = q$. The run r satisfies the local time constraint iff $\forall q \in T, q^* \in Sg(q)$, $\mathbb{E}(\mu(q^*)) \leq \tau(q)$.

An average-timed $\forall$-automaton $\mathcal{ATA}$ *accepts* a trace v, written $v \models \mathcal{ATA}$, iff all possible runs of $\mathcal{ATA}$ over v are accepting. As an example, Fig. 4(d) depicts the real-time response constraint which states that R will be reached within 30 time units of B.

5 Model-checking Approach to Constraint-Based Behaviour Verification

The formal behaviour verification method consists of a set of model-checking rules. The rules, which were introduced in [4, 3], are a generalization of the rules for deterministic dynamical systems [11]. Here we briefly summarize the verification rules and results for the simplest possible situation: discrete time and discrete domains. A generalization of the rules for arbitrary time and domains, and the proofs of the following results are available in [3].

Our verification method has three categories of rules: Invariance rules (I), Stability (Lyapunov-based) rules (S) and Average Timeliness rules (AT). Assume $\mathcal{ATA}$ is an average-timed $\forall$-automaton $\langle \mathcal{A}, T, \tau \rangle$ which represents the behavioural constraints for a stochastic state transition system $\langle \mathcal{S}, \mathbb{P}, \Theta \rangle$. Moreover, let $\{\varphi\}\mathcal{B}\{\psi\}$ denotes the consecutive condition: $\varphi(s) \wedge (s \rightsquigarrow s') \rightarrow \psi(s')$.

(I) Invariance Rules We define the 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*: $\forall q, q' \in Q, \{\alpha_q\}\mathcal{B}\{c(q, q') \rightarrow \alpha_{q'}\}$,

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

(AT) Average-Timeliness Rules Let $\mathcal{ATA} = \langle \mathcal{A}, T, \tau \rangle$ be an average-timed $\forall$-automaton. Assume, without any loss of generality, that time is encoded in the stochastic state transition system. We now define local timing functions, associated with the local average time bounds. 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:

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

The following theorem stipulates that if we satisfy rules (I), (S) and (AT), then the behaviour verification is sound and complete.

Theorem 3 (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 local timing functions.*

The above rules, however, do not guarantee the existence of an automatic verification method. Nevertheless, for PCNs with finite domains we can fully automate the verification of an average-timed $\forall$-automata constraint on the behaviour. We have implemented an automatic verification algorithm which we apply to our robotic surveillance system in the next section. Details of the algorithm are omitted here but presented in [3].

6 Modeling and Analyzing a Robotic Surveillance System

To demonstrate the utility of our approach, we analyze a simple version of the museum surveillance task introduced earlier. For the sake of simplicity, we will, in this example, model a lone mobile robot surveying two of the museum rooms, trying to ensure that the many works of art are not being damaged or stolen by the visitors. We emphasize the behavioural constraint satisfaction problem, i.e., show that the safety of the work of art will be ensured. For a complex example of an hybrid elevator system where the modeling and the constraint satisfaction tasks are presented in detail, the reader is referred to [3].

In Fig. 5, we depict the environment in which the mobile surveillance robot (located on square 1) must operate. The environment is made up of two rooms, *Regular Exposition* and *Main Showroom*. We will assume that the limitations on

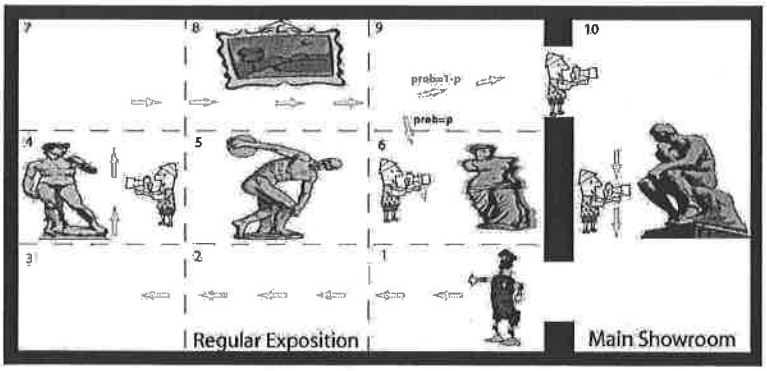


Fig. 5. Museum Environment in the Surveillance Task

the vision system of the robot are such that, the robot can only survey elements located in the same subdivision as itself. These subdivisions are numbered from 1 to 10 in Fig. 5. In addition, we will assume that the velocity of the robot is such that movement from one square to another takes constant time, $\delta = 3$ minutes.

Let us consider the scenario where the museum director is interested in ensuring that the main showroom (room 10 in Fig. 5) is surveyed regularly by the mobile robot. Specifically, the robot should come back to the main showroom every $\tau = 30$ minutes, on the average. This behavioural constraint can be represented by the average-timed $\forall$-automaton of Fig. 6(a). In addition, the director decides to adopt the strategy for the dynamics that entails the robot to move clockwise (going from square $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 7 \rightarrow 8 \rightarrow 9 \rightarrow 10 \rightarrow 1$ as shown in Fig. 5). However, the entrance to the main showroom can be obstructed (like it is the case in Fig. 5 where the entrance from square 9 to 10 is blocked by a visitor). In this case, the robot, rather than pushing the visitor aside, would simply change its trajectory and move toward square 6, then 1, and then resume a tour of the museum. We will assume that the robot can only enter the main showroom via square 9, the other access being reserved exclusively for exiting the main show room.

One non-desirable situation is the one where visitors obstruct the entrance to the main showroom permanently. In this case, the robot would never access the room, hence never offering surveillance in that room. However, this situation is quite unlikely. Rather, a more realistic scenario would be that there is some probability, say $p = 0.2$, that the entrance is blocked at any given time. More complex models for uncertainty such as a Poisson process could be used to model the arrivals of visitors at the entrance door [3]. Given the dynamics of our system and the specification of the behavioural constraints, we can associate the behaviour of our system with the stochastic transition system shown in Fig. 6(b).

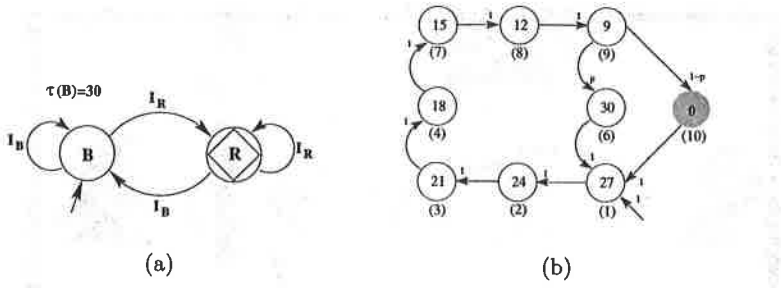


Fig. 6. Robotic Surveillance System: (a) Average-timed ∀-automata Behavioural Constraint; (b) Stochastic State Transition System of the Behaviour

The corresponding state space consists solely of the square on which the robot is located: $\mathcal{S} = \{loc\}$. The evolution kernel $\mathbb{P}$ is very simple as most transitions are deterministic, except for the one leading to the main showroom, which is uncertain due to the possible presence of visitors blocking the entry. Nevertheless, the transition probability values are depicted beside the head of the arrows in Fig. 6(b).

The verification rules presented previously require that we find invariants values, Lyapunov functions and timing functions:

I: To find the set of invariants for the average-timed ∀-automata in Fig. 4(d), let us define $I_B : loc \neq 10$ and $I_R : loc = 10$. It is easy to see that I_B and I_R are invariants for states B and R, respectively. Note that $S = \emptyset$ in our example. In Fig. 6(b), bad states B are denoted by white nodes while the only recurrent state R is denoted by a filled node.

S: Given the invariants constructed above, we define a set of Lyapunov functions ρ, whose value is the average number of transitions for reaching state R. It is easy to show that these Lyapunov functions satisfy the Stability rules.

AT: Given the invariants and the Lyapunov functions, we need to choose a set of local timing functions γ. It is easy to show that a set of functions which corresponds to the average time to reach the recurrent state R satisfies the average-timeliness rules. The values of γ are shown for each state as the number in the state nodes in Fig. 6(b).

Therefore, using the rules, we have shown that the average time for the surveillance robot to perform a round and come back to the main showroom is $30 \leq 30$ minutes, which satisfies the behavioural constraint. Note that this is not an absolute bound on the value. The completion time of an instance of the request may exceed 30 minutes. With our method, we can additionally obtain probability bounds on possible times of service. For our surveillance example we can show that the probability that the time for the mobile robot to return to main showroom be greater than 120 minutes is less than or equal to 0.075. This provides the museum director valuable additional knowledge about the safety of the surveillance system.

7 Conclusion

In this paper, we have provided the formal syntax and semantics for a useful framework, PCN, for constraint-based modeling of the dynamics and behaviours of systems exhibiting uncertainty. We show that the semantics lead to Markov processes, and in many cases to stable probabilities on the state space. The PCN framework abstracts the notions of time and domains for a general approach while allowing the user to model uncertainty of several different types. To allow the expression of constraints on both the dynamics and behaviours of systems, we provide two modeling languages: PCN and average-timed ∀-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. Average-timed ∀-automata are non-deterministic automata augmented with average time bounds. 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 a simple hybrid robotic system for surveillance.

Future directions for this work include the modification of our behavioural constraint satisfaction methodology to allow for probabilistic satisfaction, that is, satisfying the constraints not on average but with a given probability threshold. In addition, we are also investigating the use of the stochastic Lyapunov stability method for performing control synthesis for PCN models.

References

1. Zhang, Y., Mackworth, A.K.: Constraint nets: a semantic model for hybrid systems. Journal of theoretical computer science **138** (1995) 211–239
2. Mackworth, A.K., Zhang, Y.: A formal approach to agent design: An overview of constraint-based agents. Constraints **8** (2003) 229–242
3. St-Aubin, R., Mackworth, A.K.: Constraint-based approach to modeling and verification of probabilistic hybrid systems. Technical Report TR-2004-05, University of British Columbia, www.cs.ubc.ca/spider/staubin/Papers/TR-04-05.pdf (2004)
4. St-Aubin, R., Mackworth, A.K.: Probabilistic constraint nets: A constraint-based approach to hybrid dynamical systems with uncertainty. (In: Submitted to CP04)
5. Knight, F.B.: Essentials of Brownian motion and diffusion. American Mathematical Society (1981)
6. Desharnais, J., Edalat, A., Panangaden, P.: Bisimulation for labeled markov processes. Information and Computation **2** (2002) 163–193
7. Arnold, L.: Stochastic Differential Equations: theory and applications. (1974)
8. Øksendal, B.: Stochastic Differential Equations: an introduction with applications. 4th edn. Springer (1995)
9. Khas'minskiy, R.Z.: Stability of systems of differential equations in the presence of random disturbances. Nauka Press, Moscow (1969)
10. Manna, Z., Pnueli, A.: Specification and verification of concurrent programs by ∀-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. Springer-Verlag (1996)