A Calculus of Stochastic Systems
for the Specification, Simulation, and Hidden State Estimation of Hybrid Stochastic/Non-stochastic Systems

Albert Benveniste[1], Bernard C. Levy[2], Eric Fabre[1], Paul Le Guernic[1]

[1] IRISA-INRIA, Campus Universitaire de Beaulieu, 35042 Rennes Cedex, France, name@irisa.fr
[2] Dept. of of Electrical and Computer Engineering, Univ. of California, Davis, CA 95616, USA, levy@ece.ucdavis.edu

Abstract. In this paper, we consider *hybrid systems* containing both stochastic and non-stochastic[3] components. To compose such systems, we introduce a general combinator which allows the specification of an arbitrary hybrid system in terms of elementary primitives of only two types. Thus, systems are obtained hierarchically, by composing subsystems, where each subsystem can be viewed as an "increment" in the decomposition of the full system. The resulting hybrid stochastic system specifications are generally not "executable", since they do not necessarily permit the incremental simulation of the system variables. Such a simulation requires compiling the dependency relations existing between the system variables. Another issue involves finding the most likely internal states of a stochastic system from a set of observations. We provide a small set of primitives for transforming hybrid systems, which allows the solution of the two problems of incremental simulation and estimation of stochastic systems within a common framework. The complete model is called CSS (*a Calculus of Stochastic Systems*), and is implemented by the SIG language, derived from the SIGNAL synchronous language. Our results are applicable to pattern recognition problems formulated in terms of Markov random fields or hidden Markov models (HMMs), and to the automatic generation of diagnostic systems for industrial plants starting from their risk analysis. A full version of this paper is available [1], omitted proofs can be found in this reference.

Keywords: stochastic systems, hybrid systems, belief functions, communicating processes, simulation, estimation.

[3] Throughout this paper, we use the word "non-stochastic" to refer to systems which have no random part. In control science or statistics, such systems would be called "deterministic" as opposed to "stochastic"; however this name would be misleading in computer science, where "deterministic" vs. "nondeterministic" has a totally different meaning. This is why we decided to use the word "non-stochastic" here.

1 Introduction and motivation

This paper proposes a general framework for the specification and use of probabilistic models in applications of large computational complexity. To serve as reference in our subsequent discussion, we now describe several real applications which either employ, or could benefit from the use of probabilistic methods.

- *Queuing networks, performance evaluation, and risk analysis* typically require a number of tools for the specification and simulation of systems, and to compute statistics of interest. The modelling and simulation tasks usually require modular models, which are often variations of stochastic Petri nets [2]. The computation of statistics relies on the underlying Markov chain associated to the Petri net specification.
- *Pattern recognition applications*, depending on whether they focus on one-dimensional signals, such as for speech recognition, or multidimensional ones, as in image analysis and understanding, frequently rely on hidden Markov models (HMM) [3] or Markov random fields [4, 5]. Both classes of models have proved quite succesful in their respective application areas. In particular, the best speech recognition systems currently available are based on HMMs. The nonintrusive appliance load monitoring problem described recently in [6] represents another interesting pattern recognition problem, where one seeks to determine which appliances switch on and off in an individual household, based on measurements of the total load power. In this context, appliances can be modeled in terms of communicating stochastic automata.
- *Model based monitoring and diagnostics procedures for complex systems* rely often on a blend of statistical approaches [7] for numerical systems, and symbolic techniques of artificial intelligence for systems of a combinatorial nature. However, somewhat surprisingly, while models play a significant role in the development of monitoring schemes, risk analysis considerations are usually not included. Risk analysis is mainly used to assess the safety margins of designs, but does not seem to enter the synthesis of on-line monitoring and diagnostics systems, even though such an inclusion would be highly beneficial.

Such applications require the following functions:

- *System specification* is a first issue for complex systems. Because most of the applications we have described, such as load appliance monitoring, or the monitoring and diagnostics of large-scale systems, involve a mixture of random and nonrandom phenomena, a *hybrid* stochastic/non-stochastic form of modelling is desirable. Several other key features that would need to be included are modularity, i.e. the ability to specify large subsystems from small interacting modules, ease of modification, and the possibility to reuse subsystems in new applications.
- The ability to *simulate* systems, as well as evaluate statistics of interest is also a necessity. Again, modularity would be desirable in this context, although it may

be less critical than for system specification. As for simulation, an important challenge is the *fast simulation of rare events* of interest, such as for fault-tolerance applications.

- Pattern recognition and diagnostics applications require the *estimation of hidden quantities of interest*, such as spoken words in speech recognition, appliance loads for the nonintrusive appliance load monitoring application, or the origin and assessment of faults in failure diagnostics. Modularity would again be welcome in this context.

There exists a vast literature on the application of statistics and probability to the modelling, estimation, identification [8], and diagnostics [7] of dynamical systems. Unfortunately, modularity issues are almost never addressed by either statisticians or control engineers, and as a consequence, probabilistic and statistical techniques are used only rarely in the analysis of large scale systems (except in the area of performance evaluation, see below).

Stochastic Petri net models [2, 9] are often used to specify stochastic systems, in applications such as queing networks with synchronization, or fault-tolerance studies. They are commonly employed to evaluate statistics of interest in performance evaluation. However, such computations rely on the underlying Markov chain of the Petri net model, so that Petri nets by themselves do not simplify the computation of statistics. In [10, 11], however, particular structures of the transition matrix associated to certain Markov chains are used to decompose the statistical analysis of the system under consideration. Clearly, many real applications have been tackled by employing approaches developed within the Petri net community, and software products are available.

In a different area, *probabilistic communicating process algebras* and related logics have been studied in theoretical computer science [12, 13]. The common approach to such studies consists in enriching with probability available models of communicating process algebras, such as CCS, CSP, etc., and related kinds of temporal logics [14, 15, 16]. Expressive power and system equivalence are analyzed, as well as the decidability of related logics. These approaches benefit from the fundamental advances achieved by this community to handle modularity, communication, and interaction between processes. However, to our knowledge, no real application has been reported based on such approaches, and no service is really provided beyond modelling.

This paper proposes a new and flexible form of calculus, called CSS, for the *specification, simulation, and hidden state estimation of hybrid stochastic/non-stochastic systems*. The model of hybrid stochastic/non-stochastic systems that we employ is introduced in Section 2.1. Hybrid stochastic/non-stochastic systems interact via a single combinator that we call the *composition* and denote by " $\mid$ ". The combination of hybrid systems with $\mid$ yields again hybrid systems, and $\mid$ is both associative and commutative. When applied to purely non-stochastic systems, the composition operator $\mid$ behaves like the conjunction of systems of relations in mathematics. The shared variables of the two systems provide the only mechanism for system interaction. On the other hand, when two purely probabilistic systems with no shared variables are combined, we obtain two statistically independent systems. Also, combining a purely stochastic system with a purely non-stochastic one, viewed as a constraint, gives the conditional distribution of the original stochastic system, given that the constraint is satisfied; this provides a very simple mechanism to specify conditional distributions. The combination of purely non-stochastic and stochastic building blocks with $|$ allows the specification of arbitrary hybrid systems. A concrete syntax based on the SIG minilanguage is provided to implement the operations of CSS. Note that we restrict our attention here to systems with only a finite number of variables. Dynamical systems, i.e., systems defined over infinite index sets, have been examined in [17], and their study raises a number of technical issues that will be tackled elsewhere. Also, throughout the paper, only *finitely or countably valued* variables are considered. Although our results hold in more general situations, such as for the case of linear Gaussian systems which is examined in detail in [18], a precise description of such cases will not be attempted here.

Section 3 examines the *simulation* of hybrid stochastic/non-stochastic systems. Simulation is operational in nature. In contrast, the system specification provided by CSS is nonoperational, since it relies on relations. We are therefore confronted with the issue of converting a system specification into a simulation. Since many of the applications we have in mind are of a real time nature, we would like to perform simulations *incrementally*, in order to ensure their efficiency. For instance, Markov chains or stochastic automata are naturally simulated by using the Kolmogorov chain rule, so that states are generated incrementally. The Bayes rule $\mathrm{p}(x, y) = \mathrm{p}(y|x)\mathrm{p}(x)$ provides a way to simulate incrementally the random variables (X, Y) with joint distribution $\mathrm{p}(x, y)$. We only need to draw X according to the distribution $\mathrm{p}(x)$ and then, for X given, draw Y based on the conditional distribution $\mathrm{p}(y|X)$. We generalize the notions of marginal and conditional distributions to hybrid systems, and use them to extend Bayes rule to these systems. The primitives implementing the marginal and conditional are introduced in SIG and are used to derive graph transformation rules which can be used to convert a compound system to an equivalent form which admits an incremental simulation.

The maximum likelihood (ML) *estimation* of hybrid stochastic/non-stochastic systems is also considered in [1], we outline this topic here. Consider a triple (X, Y, Z) of random variables, where Z is observed, and the two unknown random variables X and Y admit the conditional distribution $\mathrm{p}(x, y|z)$. The ML estimate, also known as the maximum a posteriori (MAP) estimate, of (X, Y) given Z is given by $(\widehat{x}, \widehat{y}) = \arg\max_{x,y} \mathrm{p}(x, y|z)$. To find these estimates incrementally, we can first compute the so-called "generalized likelihood" function $\mathrm{p}_{\mathcal{L}}(x|z) = \max_y \mathrm{p}(x, y|z)$. Next, compute the conditional likelihood $\mathrm{p}_{\mathcal{L}}(y|x, z) = \mathrm{p}(x, y|z)/\mathrm{p}_{\mathcal{L}}(x|z)$ of Y given X and Z, so that the following factorization holds: $\mathrm{p}(x, y) = \mathrm{p}_{\mathcal{L}}(y|x)\mathrm{p}_{\mathcal{L}}(x)$. Then, the desired ML estimates can be generated sequentially from $\widehat{x} = \arg\max_x \mathrm{p}_{\mathcal{L}}(x|z)$ and $\widehat{y} = \arg\max_y \mathrm{p}_{\mathcal{L}}(y|\widehat{x}, z)$. This incremental estimation procedure, which is called the Viterbi alborithm in the HMM literature [3, 19], just corresponds to a simple case of dynamic programming. We extend the notions of generalized likelihood and conditional likelihood, which now take the form of primitives, to hybrid systems, and show that the above dynamic programming procedure can be generalized accordingly. These primitives are implemented in SIG, and simple graph manipulations can be used to convert the given system to a form which can be incrementally estimated. In fact, the graph transformations applied for both simulation and estimation turn out to be *identical.*

Finally, Section 4 contains some conclusions and perspectives.

It was not until recently, through discussions with A. P. Dempster, that we became aware that the work reported in this paper is in fact closely related to the Dempster-Shafer theory of belief functions [20, 21, 22] and belief networks [23, 24, 25, 26] in statistics and artificial intelligence. Although our work has independent origins, several aspects are common with belief network theories. First, like the Dempster-Shafer model of belief functions, the hybrid systems we consider are not fully probabilized, and combine both random and unknown types of uncertainties. However, while the Dempster-Shafer approach relies on an axiomatic different from probability theory, we achieve comparable results by blending probabilistic methods with constraint analysis. The composition we employ for building complex systems from simpler ones takes a form analog to Dempster's "product-intersection" rule [21] for combining belief functions. Also, our incremental simulation scheme is similar in nature to the fusion/propagation mechanism of [24, 25]. However, there exists an important difference between the partly directed, partly undirected graphs that we use to compile the dependency relations existing between the variables of a compound system, and the standard viewpoint of artificial intelligence, where directed branches encode "subjective causality." Our graphs encode "objective causality" according to the terminology of [24], since they are used to "*direct and activate the data flow in the computations* ..." [24]. In addition, while artificial intelligence emphasizes Bayesian estimation, we show that similar ideas can be applied to the solution of ML estimation problems. Finally, the practical implementation of our model has the syntactic form of a data flow programming language, which differs from the network formalism of artificial intelligence.

2 CSS and the SIG mini-language

The CSS model relies on a formal definition of hybrid stochastic/non-stochastic systems, which is then used to express the composition rule $\mid$.

2.1 CSS

Model of hybrid stochastic/non-stochastic systems. The hybrid systems we consider are described by a quadruple

$$\pi = \{\mathbf{X}, \Omega, \mathbf{W}, \mathrm{p}\} \tag{2.1}$$

where

- $\mathbf{X} = \{X_1, \ldots, X_p\}$ denotes a finite set of *variables* whose values are written as $x = (x_1, \ldots, x_p)$. The variables are the observable objects of our model. The domain of each variable X_i is denoted as V_{X_i}, so that the domain of the vector $\mathbf{X}$ can be expressed as $V_{\mathbf{X}} = \prod_i V_{X_i}$.

- $\mathbf{W} = \{W_1, \ldots, W_q\}$ denotes a finite set of *random variables* or simply *randoms* for short. Values of W_j are written as w_j, and we refer to the complete set of values $w = (w_1, \ldots, w_q)$ as a *random experiment*. The domain of W_i (resp. $\mathbf{W}$) is denoted by V_{W_i} (resp. $V_{\mathbf{W}}$). Randoms are hidden, i.e., not visible from outside the system. The reason for this property will become clear below. $\mathbf{W}$ models the random part of the system, so that if $\mathbf{W}$ is empty, the system is completely non-stochastic.
- $\mathbf{p}$ constitutes an unnormalized probability distribution for $\mathbf{W}$[4]. Specifically, we only require $\mathbf{p} \geq 0$ and $0 < \sum_w \mathbf{p}(w) < \infty$.
- Ω denotes a relation on the pair $(\mathbf{X}, \mathbf{W})$. We shall sometimes write it more explicitly as $\Omega(X_1, \ldots, X_p \; ; \; W_1, \ldots, W_q)$.

A system $\pi = \{\mathbf{X}, \Omega, \mathbf{W}, \mathbf{p}\}$ is observable only through its variables. Randoms cannot be seen, but transfer their behaviour to the system variables $\mathbf{X}$ through the relation Ω. In doing so, some predicates over the variables become random, namely those which are completely expressible in terms of $\mathbf{W}$. In the purely non-stochastic case, we may consider that $V_{\mathbf{W}}$ consists of a single point w_{triv}, with $\mathbf{p}(w_{\text{triv}}) = 1$ and $\mathbf{p}(\emptyset) = 0$.

The unique system for which $\mathbf{X} = \emptyset$ is called NIL. Finally, given an arbitray system $\pi = \{\mathbf{X}, \Omega, \mathbf{W}, \mathbf{p}\}$, the system obtained by replacing its distribution $\mathbf{p}$ by a uniform one, say equal to one, is denoted by FLAT(π), so that we have FLAT$(\pi) \triangleq \{\mathbf{X}, \Omega, \mathbf{W}, 1\}$.

EXAMPLES :

1. The above hybrid systems contain as a subclass purely non-stochastic systems described by a set of relations, without any randoms. Another subclass corresponds to purely stochastic systems, for which we have $\mathbf{X} = \mathbf{W}$, and where the relation Ω is defined by $X_i = W_i$ for all i, so that all randoms are observed as variables.
2. A system $\pi = \{(X_1, X_2), W, \Omega, \mathbf{p}\}$, with $\Omega : f(X_1, X_2) = W$ for some function f, is a system with two variables. For instance, take $X_1 + X_2 = W$. In this case, each variable X_i cannot be viewed as random since its probability distribution is not defined, but the sum $X_1 + X_2$ is random. For a general function f, not all predicates on (X_1, X_2) are random, only those which involve $f(X_1, X_2)$.
3. For a system
$$\pi = \{X, (W_1, W_2), \Omega, \mathbf{p}\} \qquad \text{with} \qquad \Omega : X = f(W_1, W_2) \tag{2.2}$$
where f is a non injective function, the variable X is random. However, because f is not injective, there are "too many" randoms ; for instance, if f depends only on W_1, we can remove W_2. This operation, called "compression," is described below in Sec. 3.1.

[4] Handling unnormalized distributions may seem unusual, but has several advantages. It simplifies the definition of the composition $|$ and the specification of conditional probabilities, and significantly decreases the computational cost of incremental simulation and estimation. Furthermore for many applications where the space $V_{\mathbf{W}}$ has a very large cardinality, such as for the study of Markov random fields in statistical mechanics [27, 28, 29], the computation of the normalizing constant (the partition function) which transforms $\mathbf{p}$ into a true probability is often unnecessary.

4. The class of linear Gaussian hybrid stochastic/non-stochastic systems of the form $EY = AX + BW$ was studied in detail in [18]. For such systems, E, A, B are matrices of suitable dimensions, W is a Gaussian random vector with zero mean and unit covariance matrix, and the variables correspond to the vector pair (X, Y).

Our model of hybrid systems is closely related to the one employed by Dempster and Shafer [20, 21, 22] to formulate their theory of belief functions. Like the systems examined here, Dempster's belief functions are specified by a quadruple consisting of a probability space $(V_{\mathbf{W}}, \mathrm{p})$, which is not directly visible, and a pair $(V_{\mathbf{X}}, \Gamma)$ formed by a set of system configurations, and a mapping Γ associating to each element $w \in V_{\mathbf{W}}$ a set $\Gamma(w) \subset V_{\mathbf{X}}$. For our model, the set-valued mapping Γ is specified implicitly by the relation Ω, which associates to each random w the set

$$\Gamma(w) = \{x : \Omega(x; w)\}\,, \tag{2.3}$$

of variables x which, together with w, satisfy the relation Ω.

Let us examine the modelling implications of the hybrid system specification $\pi = \{\mathbf{X}, \Omega, \mathbf{W}, \mathbf{p}\}$. First, observe that by eliminating the randoms $\mathbf{W}$ from the relation Ω, we obtain a family of hard constraints for the variables $\{X_1, \ldots, X_p\}$. These constraints are often called "parity checks" in the failure detection literature [7]. The subset $V_{\mathbf{X}}^{\Omega}$ of $V_{\mathbf{X}}$ satisfying these constraints can be used to test the validity of the model π, by checking whether the visible variables belong to this set.

Next, we note that each set $B \subseteq V_{\mathbf{W}}$ of random experiments admits the prior probability

$$\mathbf{P}(B) = \frac{\sum_{w \in B} \mathrm{p}(w)}{\sum_{w} \mathrm{p}(w)}\,. \tag{2.4}$$

Eliminating the variables $\mathbf{X}$ from the relation Ω yields hard constraints that must be satisfied by the randoms $\{W_1, \ldots, W_q\}$. Let $V_{\mathbf{W}}^{\Omega}$ be the set of w's satisfying these constraints. The posterior probability on the randoms, given the set $V_{\mathbf{W}}^{\Omega}$ of allowable configurations, takes the form

$$\mathbf{P}^{\Omega}(B) = \frac{\sum_{w \in B \cap V_{\mathbf{W}}^{\Omega}} \mathrm{p}(w)}{\sum_{w \in V_{\mathbf{W}}^{\Omega}} \mathrm{p}(w)} = \frac{\mathbf{P}(B \cap V_{\mathbf{W}}^{\Omega})}{\mathbf{P}(V_{\mathbf{W}}^{\Omega})} = \mathbf{P}(B \mid V_{\mathbf{W}}^{\Omega})\,. \tag{2.5}$$

The posterior probability $\mathbf{P}^{\Omega}$ is the result of the interaction of the relation Ω with the prior distribution $\mathbf{p}$ in the system specification $\pi = \{\mathbf{X}, \Omega, \mathbf{W}, \mathbf{p}\}$. Unfortunately, this new probability cannot be transferred to the variables $\mathbf{X}$ because, since Ω is a relation, the sets $\Gamma(w)$ specified by (2.3) are not singletons, and may not be disjoints for different w's. This is just a manifestation of the fact that, because projection is a monotonic operator on sets, but is not additive, projecting a probability from one space onto another does not yield a probability, but a different object, called a *Choquet capacity* [30]. On $V_{\mathbf{X}}$, this capacity provides a partial probabilistic knowledge which was described by Dempster [20, 21] in terms of upper and lower probabilities for the subsets of $V_{\mathbf{X}}$. These upper and lower probabilities provide bounds describing the limits of our information concerning predicates of the $\mathbf{X}$ variables. In this paper, instead of adopting the Dempster-Shafer upper/lower probability framework, we shall remain within the realm of standard probability theory by considering exclusively probabilities over the set $V_{\mathbf{W}}$ of randoms.

The " $|$ " system combinator. The composition of hybrid systems can be performed in the same manner as the combination of belief functions described in [20]. The main aspect of the combination operation is that different systems are allowed to share common variables, which describe their interaction, but not randoms. In other words, randoms are always private, and do not play a role in the combination of systems. For two systems $\pi_i = \{\mathbf{X}_i, \Omega_i, \mathbf{W}_i, \mathrm{p}_i\}$ with $i = 1, 2$, the combinator $\pi_1 \mid \pi_2 = \{\mathbf{X}, \Omega, \mathbf{W}, \mathrm{p}\}$ is defined as

$$\mathbf{X} = \mathbf{X}_1 \cup \mathbf{X}_2 \tag{2.6a}$$

$$\mathbf{W} = \mathbf{W}_1 \times \mathbf{W}_2 \tag{2.6b}$$

$$\mathrm{p}(w) = \mathrm{p}(w_1, w_2) = \mathrm{p}_1(w_1) \times \mathrm{p}_2(w_2) \tag{2.6c}$$

$$\Omega = \Omega_1 \wedge \Omega_2\,, \tag{2.6d}$$

where $\Omega_1 \wedge \Omega_2$ denotes the conjunction of relations Ω_1 and Ω_2, which is the usual way of defining systems of equations in mathematics. The expressions (2.6b) and (2.6c) indicate respectively that variables may be shared, but not randoms.

The identities (2.6a–2.6d) show that systems interact only through their shared variables. The NIL system is a neutral element for the combinator " $|$ ".

EXAMPLES :

1. Consider two systems $\pi_i = \{\mathbf{X}_i, \Omega_i, \mathbf{W}_i, \mathrm{p}_i\}$ with $i = 1, 2$, where π_1 is purely stochastic, so that $\mathbf{X}_1$ and $\mathbf{W}_1$ have same cardinality and $\Omega_1 : X_1 = W_1, \ldots X_p = W_p$, and π_2 is purely non-stochastic, i.e., $\mathbf{W}_2 = \emptyset$, with the nontrivial relation Ω_2. Assume also that $\mathbf{X}_1 = \mathbf{X}_2$. Then, it is easy to check that $\pi_1 \mid \pi_2 = \{\mathbf{X}_1, (\Omega_1 \wedge \Omega_2), \mathbf{W}_1, \mathrm{p}_1\}$. The combined system $\pi_1 \mid \pi_2$ has still the feature that randoms are visible through the variables, since $\Omega_1 : X_1 = W_1, \ldots, X_p = W_p$. However, the variables $X_1, \ldots, X_p$ behave now according to the conditional distribution $\mathrm{p}_1^{\Omega_2}$ of p_1 based on the constraint Ω_2. Thus the composition $|$ provides a simple mechanism for specifying conditional probabilities, which will be used extensively in the SIG examples presented below.
2. Let $\pi_i = \{\mathbf{X}_i, \Omega_i, \mathbf{W}_i, \mathrm{p}_i\}$ with $i = 1, 2$ be two systems which do not interact, so that $\mathbf{X}_1 \cap \mathbf{X}_2 = \emptyset$. Then, in the combination $\pi_1 \mid \pi_2$, the randoms $\mathbf{W}_1$ and $\mathbf{W}_2$ are independent.

2.2 The SIG mini-language

We now proceed to decribe a syntax, in the form of the langage SIG, which implements both the modelling format and composition rule of CSS.

The primitives of SIG. The SIG language has the following primitives:

```
  (i) R(x1,...,xp)
 (ii) potential U(x1,...,xp)
(iii) P | Q
```

They admit the following informal interpretation.

(i) `R(x1,...,xp)` specifies a relation among the variables `x1,...,xp`. The corresponding system in the sense of (2.1) admits the `xi`'s as variables, has no randoms, and Ω is the relation R. Thus, `R(x1,...,xp)` is a purely non-stochastic system.

(ii) `potential U(x1,...,xp)`, where U is a function taking values over the line $(-\infty, +\infty]$, specifies random variables with unnormalized joint distribution

$$\exp -U(x_1, \ldots, x_p) \,. \tag{2.7}$$

The corresponding system in the sense of (2.1) has `x1,...,xp` as variables, its randoms $\mathbf{W} = (W_1, \ldots, W_p)$ have the distribution $\exp -U(w_1, \ldots, w_p)$, and Ω relates variables and randoms via the relations $\mathtt{x1} = W_1, \ldots, \mathtt{xp} = W_p$. Thus, `potential U(x1,...,xp)` is a purely stochastic system.

(iii) `P | Q` denotes the application of the " | " combinator to systems P and Q.

Specifying systems with SIG. A system in the sense of CSS and definition (2.1) can be declared as shown below, where we omit variable type declarations of the form "`integer`", etc:

```
system PI =
    { variable X,Y,Z }          % declaration of variables
    (| potential U(X,Y)         % distribution of (X,Y)
     | Z = f(X,Y)               % constraint on (X,Y,Z)
     |)
end
```

Several examples of SIG programs are now presented. To generate a hidden Markov model (HMM) of the type discussed in [3, 18], we can employ the following program:

```
system  HMM_0 = (integer N)
    { variable X[i] i=0 to N, Y[i] i=1 to N }
    (| X[0] = 0
     | loop i=1 to N
         (| potential U(X[i-1],X[i])
          | potential V(X[i-1],X[i],Y[i])
          |)
       end
     |)
end
```

The first constraint fixes the initial condition, and the `loop` statement specifies the joint density of the internal states X_i and outputs Y_i. The resulting system HMM_0 is a HMM with state X and output Y. It has 0 for initial state, and its state transitions and outputs are specified by the interactions U and V, so that

$$\mathbf{p}(x_0,\ldots,x_N\ ;\ y_1,\ldots,y_N) \propto \delta_0(x_0)\ \exp - \sum_{i=1}^{N} \left[U(x_{i-1},x_i) + V(x_{i-1},x_i,y_i)\right] \tag{2.8}$$

where $\propto$ denotes "proportional to", and $\delta_0(x) = 1$ if $x = 0$, $= 0$ otherwise. If we want to consider the same HMM *given that the final condition* `X[N] = X_MAX` *also holds,* one needs only to add the final constraint to the previous SIG program, thus yielding

```
system  HMM = (integer N)
    { variable X[i] i=0 to N, Y[i] i=1 to N }
    (| X[0] = 0
     | X[N] = X_MAX
     | loop i=1 to N
         (| potential U(X[i-1],X[i])
          | potential V(X[i-1],X[i],Y[i])
          |)
       end
     |)
end
```

To explain the interest of this simple trick, suppose X models the occupation level of a buffer, which behaves according to HMM_0. Assume X_MAX corresponds to a critical level, and we want to know the conditional distribution of the buffer evolution given that level X_MAX is reached at instant N. Then we only need to include the conditioning event `X[N] = X_MAX` as an additional constraint in our original program HMM_0 in order to obtain the desired behavior HMM. This mechanism can be employed whenever one seeks to concentrate on the set of experiments satisfying a condition of interest. Note for example that a common technique of risk analysis involves tracking cascades of events leading to a specific failure.

3 Simulation

We now turn to the simulation of hybrid systems. Simulation is operational in nature. In contrast, the system specification provided by CSS relies on relations, which are intrinsically nonoperational. This raises the issue of converting a system specification into an equivalent simulation. In this context, since we naturally wish to generate efficient simulations, we restrict our attention to *incremental simulations.* For instance, Markov chains or stochastic automata can be simulated incrementally by employing the Kolmogorov chain rule to generate the states one at a time. Such a feature is obviously mandatory for real-time applications.

Consider a pair (X, Y) of standard random variables with joint distribution $\mathbf{p}(x, y)$. These two random variables can be simulated incrementally by employing the following procedure.

1. Compute the marginal
$$\mathbf{p}(x) = \sum_{y} \mathbf{p}(x,y) \tag{3.1}$$
of $\mathbf{p}$ with respect to X.

2. Compute the conditional distribution $\mathbf{p}(y|x) = \mathbf{p}(x,y)/\mathbf{p}(x)$ of Y given X, so that we obtain the following factorization, also known as Bayes rule:

$$\mathbf{p}(x,y) = \mathbf{p}(y|x)\mathbf{p}(x) . \tag{3.2}$$

3. Draw X at random following the marginal $\mathbf{p}(x)$, and then, for a given X, draw Y at random according to the conditional distribution $\mathbf{p}(y|X)$.

We now generalize this technique to the case of hybrid systems.

3.1 Compressing the random part of a system

Since randoms are hidden, only their visible effect upon the system variables is of interest. But as we have already seen in example (2.2), the domain V_W of all randoms may include too many details. For example, consider a pair (W_1, W_2) and assume that W_1 is visible but not W_2. The corresponding CSS model has a single variable X_1 and constraint $X_1 = W_1$. Since W_2 is unneeded, it can be removed from the original system by computing the compressed distribution $\mathbf{p}_{\text{co}}(w_1) = \sum_{w_2} \mathbf{p}(w_1, w_2)$, which for this simple case reduces to the marginal distribution with respect to w_1. This is just an elementary case of the compression operation we now introduce.

To a system π, we can associate the following equivalence relation between randoms

$$w \sim_\pi w' \text{ iff } \forall x \ : \ \Omega(x; w) \Leftrightarrow \Omega(x, w') , \tag{3.3}$$

which just indicates that two randoms w and w' are equivalent if they cannot be distinguished by the variables. Accordingly, a set $B \subseteq V_W$ is visible through the variables if and only if it satisfies the property

$$\left.\begin{array}{r} w \in B \\ w' \sim_\pi w \end{array}\right\} \Rightarrow w' \in B . \tag{3.4}$$

It is natural to restrict $\mathbf{p}$ to the sets of randoms satisfying this condition. Note in this respect that the family $\mathcal{W}$ of all sets B satisfying the condition (3.4) forms a σ-algebra, since it is closed under intersection and complementation, and contains the empty set. Hence, in order to characterize the random behavior of the system π, we only need to specify the conditional probability $\mathbf{P}(.|\mathcal{W})$ of $\mathbf{P}$ given $\mathcal{W}$. This can be accomplished by constructing what we shall call the *compression* π_{co} of π. The compression is obtained from π and the equivalence relation $\sim_\pi$ in the following manner.

1. First, we compress the set V_W of random experiments by retaining only the equivalence classes of the relation $\sim_\pi$. Thus, an experiment w belongs to an equivalence class w_{co}, and the set of all equivalence classes forms the compressed domain V_{co}.
2. Compress the relation Ω accordingly, by setting

$$\Omega_{\text{co}}(x\,; w_{\text{co}}) \stackrel{\Delta}{=} \Omega(x\,; w) \quad \text{for } w \in w_{\text{co}} . \tag{3.5a}$$

3. Finally, to each equivalence class w_{co} of randoms, we assign the probability

$$\mathbf{p}_{\text{co}}(w_{\text{co}}) \stackrel{\Delta}{=} \sum_{w \in w_{\text{co}}} \mathbf{p}(w) . \tag{3.5b}$$

Two systems π and π' admitting the same compressed form are said to be equivalent, which is denoted as

$$\pi \equiv \pi' . \tag{3.6}$$

Since the procedure employed to compress a system does not affect its external behavior as seen from the variables, we have the following result.

Theorem 3.1 *If $\pi_i = \{\mathbf{X}_i, \Omega_i, \mathbf{W}_i, \mathbf{p}_i\}$ $i = 1, 2$ are equivalent in the sense of (3.6), they cannot be distinguished under simulation. In particular, they have*

1. *the same variables: $\mathbf{X}_1 = \mathbf{X}_2$;*
2. *the same parity checks: $V_{\mathbf{X}_1}^{\Omega_1} = V_{\mathbf{X}_2}^{\Omega_2}$;*
3. *the probability spaces $\{V_{\mathbf{W}_i}^{\Omega_i}, \mathcal{W}_i, \mathbf{P}_i\}$ are isomorphic, so that there exists a one-to-one map ϕ from the σ-algebra $\mathcal{W}_1$ onto $\mathcal{W}_2$ such that $\forall B_1 \in \mathcal{W}_1$, $\mathbf{P}_2(\phi(B_1)) = \mathbf{P}_1(B_1)$.*

The property

$$\pi \equiv \text{FLAT}(\pi) \mid \pi , \tag{3.7}$$

which is proved in [1], is a straightforward consequence of the notion of system equivalence. This identity generalizes to hybrid systems the idempotence of composition property $\pi \mid \pi = \pi$ of purely non-stochastic systems.

Although the factorization (3.2) cannot be extended directly to hybrid stochastic/non-stochastic systems, by employing Theorem 3.1, we develop below a general procedure for decomposing an arbitrary hybrid system π into marginal and conditional components which extends the factorization (3.2) of standard probability distributions. This decomposition will provide the key element required for incremental system simulation.

3.2 Two primitives

Consider a system $\pi = \{\mathbf{X}, \Omega, \mathbf{W}, \mathbf{p}\}$ and a subset of variables $\mathbf{X}' \subset \mathbf{X}$. The concepts of marginal and conditional distributions can be extended to hybrid systems by constructing the *marginal* and *conditional* systems

$$\overline{\mathcal{S}}_{\mathbf{X}'}(\pi) \qquad \text{and} \qquad \mathcal{S}_{\mathbf{X}'}(\pi)$$

respectively, where $\mathcal{S}$ represents here a mnemonic for $\mathcal{S}$imulation.

The marginal. It consists of eliminating from π the variables not in $\mathbf{X}'$, which gives

$$\overline{\mathcal{S}}_{\mathbf{X}'}(\pi) = \{\mathbf{X}', \Omega', \mathbf{W}, \mathbf{p}\} , \tag{3.8}$$

where Ω' denotes the relation obtained by employing the existential qualifier $\exists$ to eliminate from Ω the variables not in $\mathbf{X}'$, so that

$$\Omega'(x'; w) \triangleq \exists x" : \Omega\left((x', x"); w\right) . \tag{3.9}$$

Note that neither the set of randoms $\mathbf{W}$ nor the density $\mathbf{p}$ are changed by this construction, which involves only tracking the effect of the projection of $\mathbf{X}$ onto $\mathbf{X}'$ in the relation Ω.

The conditional. The conditional system has the structure

$$\mathcal{S}_{\mathbf{X}'}(\pi) = \{\mathbf{X}, \Omega, \mathbf{W}, \mathbf{p}''\} \ , \tag{3.10a}$$

where the distribution $\mathbf{p}''$ is selected such that the factorization

$$\pi \equiv \overline{\mathcal{S}}_{\mathbf{X}'}(\pi) \ \mid \ \mathcal{S}_{\mathbf{X}'}(\pi) \tag{3.10b}$$

holds. Note that the relation $\equiv$ indicates that both sides have the same compressed form. The decomposition (3.10b) represents the extension to hybrid systems of the factorization (3.2) of a probability distribution into marginal and conditional components. A constructive proof of existence of the conditional is given in [1].

Notation. In the following, it will be convenient to extend the definition of $\overline{\mathcal{S}}_{\mathbf{Z}}(\pi)$ and $\mathcal{S}_{\mathbf{Z}}(\pi)$ to the case where $\mathbf{Z}$ is not necessarily a subset of the variables $\mathbf{X}$ of π, by denoting

$$\overline{\mathcal{S}}_{\mathbf{Z}}(\pi) \ \stackrel{\Delta}{=} \ \overline{\mathcal{S}}_{\mathbf{Z}\cap\mathbf{X}}(\pi) \ \ , \ \ \mathcal{S}_{\mathbf{Z}}(\pi) \ \stackrel{\Delta}{=} \ \mathcal{S}_{\mathbf{Z}\cap\mathbf{X}}(\pi) \ . \tag{3.11}$$

3.3 Properties of the primitives

The operations that we have just introduced admit a number of algebraic properties which are stated in [1]. We shall only discuss here the very nature of system interaction. In what follows, $f \circ g(x)$ denotes the composition of maps $f(g(x))$.

Systems with no shared variables have no interaction, and involve independent families of randoms. Hence if π_1 and π_2 have no shared variables, in order to simulate the composition $\pi_1 \ \mid \ \pi_2$, we only need to simulate π_1 and π_2 separately. This corresponds to the easiest, but trivial, case of incremental simulation. But our discussion at the beginning of this section indicates that incremental simulation can be performed under more general circumstances. Specifically, the reason why the factorization (3.2) allows the simulation of first $\mathbf{X}$ followed by $\mathbf{Y}$ is that combining $\mathbf{p}(y|x)$ with $\mathbf{p}(x)$ does not modify the behavior of $\mathbf{X}$. In other words, $\mathbf{p}(y|x)$ represents totally new information with no bearing on $\mathbf{X}$. This feature leads us to introduce the notion of *innovation* which extends to hybrid systems the familiar concept of innovations process in filtering and detection theory.

Definition 3.1 (innovation) *Let π_i and $\mathbf{X}_i$ $i = 1, 2$ be two systems and their variables. If $\mathbf{Y}$ denotes an arbitrary set of variables, the system π_2 is said to be a $\mathbf{Y}$–innovation of π_1, which we denote as*

$$\pi_2 \perp\!\!\!\perp_{\mathbf{Y}} \pi_1 \ ,$$

if

$$\overline{\mathcal{S}}_{\mathbf{X}_1\cup\mathbf{Y}} \circ \mathcal{S}_{\mathbf{Y}}(\pi_2) \ \mid \ \mathcal{S}_{\mathbf{Y}}(\pi_1) \ \equiv \ \mathcal{S}_{\mathbf{Y}}(\pi_1) \ . \tag{3.12}$$

Thus, π_2 represents a $\mathbf{Y}$–innovation of π_1 if composing $\mathcal{S}_{\mathbf{Y}}(\pi_2)$ with $\mathcal{S}_{\mathbf{Y}}(\pi_1)$ does not modify $\mathcal{S}_{\mathbf{Y}}(\pi_1)$. More intuitively, this means that given $\mathbf{Y}$, the interaction between π_1 and π_2 is oriented from π_1 to π_2. For the special case when $\mathbf{Y}$ is empty, we just say that π_2 is an innovation *of π_1, which is written as $\pi_2 \perp\!\!\!\perp \pi_1$.*

From the above definition and comments it is clear that the relations $\perp\!\!\!\perp_{\mathbf{Y}}$ and $\perp\!\!\!\perp$ are not commutative. Also the selection of the conditioning set $\mathbf{Y}$ affects strongly whether a system constitutes an innovation of another. For example, if $\mathbf{X}_1 \cap \mathbf{X}_2 \subseteq \mathbf{Y}$, the two relations $\pi_1 \perp\!\!\!\perp_{\mathbf{Y}} \pi_2$ and $\pi_2 \perp\!\!\!\perp_{\mathbf{Y}} \pi_1$ hold trivially. The concept of innovation will form the basis for the derivation of compilation rules for decomposing a system into an *ordered* sequence of subsystems which can be simulated in accordance to this order. The compilation rules will rely on the following properties of innovations.

Lemma 3.1 *Given an arbitrary system π, and a subset $\mathbf{X}'$ of its variables, we have*

$$\mathcal{S}_{\mathbf{X}'}(\pi) \perp\!\!\!\perp \overline{\mathcal{S}}_{\mathbf{X}'}(\pi) \ , \tag{3.13}$$

i.e. the conditional innovates with respect to the marginal. Furthermore, if $\pi_2 \perp\!\!\!\perp_{\mathbf{Y}} \pi_1$, i.e. π_2 is a $\mathbf{Y}$–innovation of π_1, the following identities hold:

$$\overline{\mathcal{S}}_{\mathbf{X}_1 \cup \mathbf{Y}}(\ \mathcal{S}_{\mathbf{Y}}(\pi_2)\ \mid \mathcal{S}_{\mathbf{Y}}(\pi_1)\ ) = \mathcal{S}_{\mathbf{Y}}(\pi_1) \tag{3.14}$$

$$\mathcal{S}_{\mathbf{Y}}(\pi_1 \mid \pi_2) \equiv \mathcal{S}_{\mathbf{Y}}(\pi_1) \mid \mathcal{S}_{\mathbf{Y}}(\pi_2) \tag{3.15}$$

$$\overline{\mathcal{S}}_{\mathbf{Y}}(\pi_1 \mid \pi_2) \equiv \overline{\mathcal{S}}_{\mathbf{Y}}(\pi_1) \mid \overline{\mathcal{S}}_{\mathbf{Y}}(\pi_2) \ . \tag{3.16}$$

3.4 Incremental system simulation

Consider now a compound system of the form

$$\pi = \ \big|_{i \in I}\ \pi_i \tag{3.17}$$

where I denotes a finite index set. We seek to develop an incremental simulation procedure for such a system, so as to be able to evaluate progressively the probabilities of complex events.

Graphical representation. Let π_1 and π_2 be two systems admitting a nonempty set $\mathbf{X}$ of common variables. For these two systems, we employ the graphical notation

$$\pi_1 \ \text{———}\ \mathbf{X}\ \text{———}\ \pi_2 \quad \text{if} \quad \begin{cases} \pi_2 \text{ is not an innovation of } \pi_1, \text{ and} \\ \pi_1 \text{ is not an innovation of } \pi_2\ . \end{cases} \tag{3.18a}$$

Similarly, we write

$$\pi_1 \longrightarrow \mathbf{X} \longrightarrow \pi_2 \quad \text{if} \quad \begin{cases} \pi_2 \text{ is an innovation of } \pi_1, \text{ but} \\ \pi_1 \text{ is not an innovation of } \pi_2\ , \end{cases} \tag{3.18b}$$

and

$$\pi_1 \longleftrightarrow \mathbf{X} \longleftrightarrow \pi_2 \quad \text{if} \quad \begin{cases} \pi_2 \text{ is an innovation of } \pi_1 \text{ and} \\ \pi_1 \text{ is an innovation of } \pi_2\ . \end{cases} \tag{3.18c}$$

Obviously, it is rather uncommon that two systems should be mutual innovations, and still share common variables. However, this situation may occur in certain instances, such as when $\pi_1 = \pi_2 = \pi$ with π non-stochastic, since in this case the composition rule $\pi \mid \pi \equiv \pi$ implies π is its own innovation.

Next, consider each pair (π_i, π_j) of components of the compound system π given by (3.17). If π_i and π_j share common variables, we say they are *neighbors* and draw a branch between them. The choice of branch orientation or the lack thereof depends on which of the three cases (3.18a)–(3.18c) holds. In this manner, we generate a bipartite graph, where systems and variables alternate, which we call the *execution graph* of π, and denote by

$$\text{EXECGRAPH}(\pi)\ .$$

This graph has the effect of visualizing all the statistical dependency relations existing between the variables of subsystems π_i, $i \in I$.

It is worth noting that graphs of a similar nature have been introduced recently by a number of authors under the name of influence diagrams, or belief networks, to perform local computations on large networks of interconnected conditional probability distributions [26, 24, 25] or belief functions [23, 31]. Such networks, as well as the execution graphs described above, find their root in the standard graphical representation of Markov random fields in terms of cliques of neighbors [27]. However, while the graphs of Markov random fields are undirected, like the branches produced in (3.18a), the goal of belief networks is to perform local computations in a causal manner, which as will be shown below, requires a directed acyclic graph. At this stage, the execution graph associated to a compound system π of the form (3.17) is in general partly undirected, and partly directed. Our objective is now to develop compilation rules for transforming this graph into a directed one.

Graph compilation. The structure of the execution graph of a compound system provides all the information required to determine whether this system can be simulated incrementally, as shown by the following result.

Lemma 3.2 *Consider a partition $I = J \cup J^c$ with $J \cap J^c = \emptyset$, for which we write*

$$\pi_J = \ \Big|_{j \in J} \ \pi_j \qquad \pi_J^c = \ \Big|_{j \in J^c} \ \pi_j$$

$$\begin{aligned}
\mathbf{X}_J &= \textit{set of private variables of } \pi_J \\
\mathbf{X}_J^c &= \textit{set of private variables of } \pi_J^c \\
\partial\mathbf{X} &= \textit{set of shared variables of } \pi_J \textit{ and } \pi_J^c\ .
\end{aligned}$$

1. *We have*

$$\overline{\mathcal{S}}_{\mathbf{X}_J}(\pi) \ = \ \overline{\mathcal{S}}_{\mathbf{X}_J}\left(\ \pi_J \ \middle| \ \overline{\mathcal{S}}_{\partial\mathbf{X}}\left(\pi_J^c\right) \ \right)\ . \tag{3.19}$$

2. *Under the stronger assumption*

$$\pi_J \longrightarrow \partial\mathbf{X} \longrightarrow \pi_J^c \quad \textit{or} \quad \pi_J \longleftrightarrow \partial\mathbf{X} \longleftrightarrow \pi_J^c\ , \tag{3.20}$$

the identity (3.19) reduces to

$$\overline{\mathcal{S}}_{\mathbf{X}_J}(\pi) \ = \ \overline{\mathcal{S}}_{\mathbf{X}_J}(\pi_J)\ , \tag{3.21}$$

which indicates that to simulate the variables $\mathbf{X}_J$ of the compound system π, we only need to simulate the subsystem π_J.

Based on the above lemma, we can readily determine from the execution graph of a compound system π whether this system can be simulated incrementally.

Theorem 3.2 *A compound system π of the form (3.17) admits an incremental simulation if and only if* ExecGraph(π) *is an acyclic directed graph. This means that this graph contains no undirected branch of the form (3.18a), and no directed cycle. When determining whether the graph contains cycles, all bidirectional branches of the form (3.18c) can be used as "wild cards" whose orientation can be selected so as to break potential cycles.*

As a side remark, note that fixed-point equations of the form (3.19) can be solved iteratively by employing stochastic relaxation methods such as the Metropolis algorithm or the Gibbs sampler [4]. However, such schemes fall outside the scope of the incremental simulation procedures described here.

Next, since most compound systems of the form (3.17) usually give rise to execution graphs which contain either undirected branches or cycles, it is of interest to develop transformation/compilation rules, which when applied to a given system π, will yield a new system which can be incrementally simulated. In doing so, we restrict our attention to transformations which preserve the local connectivity of ExecGraph(π). Otherwise, we could always aggregate all the subsystems π_i and their variables into the full π system which contains only one increment, and thus admits a trivial, but uninteresting, incremental simulation. Consequently, we require for the time being that the transformations applied to π should preserve the structure of the interaction graph obtained by removing all branch orientations from ExecGraph(π), as well as the variables $\mathbf{X}_i$ of the subsystems forming its vertices.

Theorem 3.3 *Given a compound system π whose interaction graph forms a tree, we can transform π into an equivalent system π' such that* ExecGraph(π') *is a directed tree, and is thus amenable to incremental simulation.*

Proof: Since the interaction graph of π forms a tree, the index set I admits a natural distance, where for $i, j \in I$, $d(i, j) = k$ if the unique path linking π_i to π_j has k branches. Select now an arbitrary node $i_0 \in I$ as the root of the tree. A partial partial order can be defined over I by considering the distance of i to i_0. Thus we write $i \prec j$ if i is closer to i_0 than j. Consider the following rules:

Rule 1: Select $i \in I$, and let i_- be the unique neighbour of i such that $i_- \prec i$, i.e. i_- denotes the parent of i. If π_i and π_{i_-} share variables, and π_i is not already an innovation of π_{i_-}, then factor π_i as

$$\pi_i \equiv \overline{\mathcal{S}}_{\mathbf{X}_{i_-}}(\pi_i) \mid \mathcal{S}_{\mathbf{X}_{i_-}}(\pi_i)\,, \tag{3.22}$$

otherwise do nothing. Here, $\mathbf{X}_{i_-}$ denotes the set of variables of π_{i_-}.

Rule 2: If the factorization (3.22) has been performed, reorganize the compound system π by rewriting

$$\pi_{i_-} \mid \pi_i \equiv \pi'_{i_-} \mid \pi'_i \tag{3.23a}$$

with

$$\pi'_{i\_} \triangleq \pi_{i\_} \mid \overline{\mathcal{S}}_{X_{i\_}}(\pi_i) \qquad \pi'_i \triangleq \mathcal{S}_{X_{i\_}}(\pi_i)\,. \tag{3.23b}$$

This reorganization clearly preserves the structure of the interaction graph of π, as well as the variables of subsystems $\pi_{i\_}$ and π_i.

The index set I can be ordered so that successive indices i are *nonincreasing* with respect to the partial order $\prec$. By successively applying RULE 1 and RULE 2 to this sequence, we find that once the transformation (3.23a)–(3.23b) has been applied to node i, the new system π' includes the branch

$$\pi'_{i\_} \longrightarrow \mathbf{X}_{i\_,i} \longrightarrow \pi'_i \tag{3.24}$$

in its execution graph, where $\mathbf{X}_{i\_,i}$ represents the set of shared variables of $\pi'_{i\_}$ and π'_i. Then when RULE 1 and RULE 2 are subsequently applied to system $\pi'_{i\_}$, $\pi'_{i\_}$ may change, but the orientation of the branch (3.24) remains the same. Thus, to transform the given tree into a fully oriented tree, we need to apply the rules only once at each node of the tree, by moving gradually from its extremities towards its root i_0, so that the complexity of the compilation procedure is proportional to the cardinality of I. Note that in the above procedure, the choice of root i_0 is completely arbitrary. □

3.5 The SIG simulation compiler

We now implement the marginal and conditional primitives in the SIG language, and use them to incrementally simulate compound systems. Let SYSTEM denote a system and X, Y be two of its variables. The two operators

```
extract X,Y in SYSTEM
given X,Y SYSTEM
```

denote respectively the marginal $\overline{\mathcal{S}}_{X,Y}$ (SYSTEM) and conditional $\mathcal{S}_{X,Y}$ (SYSTEM).

To illustrate the application of these operators, we consider the HMM example. As a first step, examine the system

```
system  HMM_inc = (integer N)
    { variable X[i] i=0 to N, Y[i] i=1 to N }
    (| X[0] = 0
     | loop i=1 to N
        (| given X[i-1] potential U(X[i-1],X[i])
         | given X[i-1],X[i] potential V(X[i-1],X[i],Y[i])
         |)
       end
     |)
end .
```

Since only " given ... potential ... " statements are used, the SIG program HMM_inc admits the execution graph

$$\begin{tikzcd}
\pi_0 \arrow[r] & x_0, x_1 \arrow[r] \arrow[d] & \pi_1 \arrow[r] & x_1, x_2 \arrow[r] \arrow[d] & \pi_2 & \cdots\cdots & \pi_{N-1} \arrow[r] & x_{N-1}, x_N \arrow[r] \arrow[d] & \pi_N \\
& \sigma_1 & & \sigma_2 & & & & \sigma_N &
\end{tikzcd} \tag{3.25}$$

where the subsystems appearing in the graph are defined by

```
   PI[i] ::= given X[i-1] potential U(X[i-1],X[i])
SIGMA[i] ::= given X[i-1],X[i] potential V(X[i-1],X[i],Y[i])
```

Since this execution graph is an oriented tree, according to Theorem 3.2, we can simulate `HMM_inc` "on-line" for increasing values of the index `i`. On the other hand, this is not the case if we consider the original HMM program, even if it contains only `given ...` statements, because of the presence of the two-point boundary-value condition

```
(| X[0] = 0
 | X[N] = X_MAX
 |) .
```

In fact, the execution graph of HMM takes the form

$$\begin{tikzcd}
\pi_0 \arrow[r, dash] & x_0, x_1 \arrow[r, dash] \arrow[d, dash] & \pi_1 \arrow[r, dash] & x_1, x_2 \arrow[r, dash] \arrow[d, dash] & \pi_2 & \cdots\cdots & \pi_{N-1} \arrow[r, dash] & x_{N-1}, x_N \arrow[r, dash] \arrow[d, dash] & \pi_N \\
& \sigma_1 & & \sigma_2 & & & & \sigma_N &
\end{tikzcd} \tag{3.26}$$

It is a nonoriented tree, which can be transformed into a directed one by employing the two compilation rules described in the proof of Theorem 3.3. The algorithm proceeds in two phases: we first apply the rules to the vertical branches of the tree, which model the HMM observations, and then perform a right to left sweep over the horizontal branches, which model the Markov chain dynamics.

1. Applying RULE 1, the `potential V(X[i-1],X[i],Y[i])` can be decomposed as follows, where <=> means $\equiv$:

   ```
       potential V(X[i-1],X[i],Y[i])
   <=>
     (| extract X[i-1],X[i] in potential V(X[i-1],X[i],Y[i])
      | given X[i-1],X[i] potential V(X[i-1],X[i],Y[i])
      |) .
   ```

   For each index `i`, the subsystem

   ```
   SIGMA[i] ::= given X[i-1],X[i] potential V(X[i-1],X[i],Y[i])
   ```

   is an innovation with respect to all other subsystems, and is executable as soon as `X[i-1]` and `X[i]` have been simulated.
2. Applying RULE 2, define

   ```
   PI[i] ::=
   (| extract X[i-1],X[i] in potential V(X[i-1],X[i],Y[i])
    | potential U(X[i-1],X[i])
    |)
   ```

   where the boundary constraints `X[0] = 0` and `X[N] = X_MAX` need also to be included for `i = 1` and `i = N`, respectively.
3. Recursively, for `i` decreasing from `N` to `1`,
   (a) apply RULE 1 and decompose

   ```
   PI[i] <=> (| extract X[i-1] in PI[i]
              | given X[i-1] PI[i]
              |) ;
   ```

   (b) apply RULE 2 and redefine

```
PI[i]   ::= (| given X[i-1] PI[i]
             |)
PI[i-1] ::= (| extract X[i-1] in PI[i]
             | PI[i-1]
             |)
```

The resulting system is equivalent to the original one, and has the execution graph (3.25), so that it is ready for simulation.

4 Discussion and Conclusions

We have introduced the CSS model and associated SIG minilanguage for describing stochastic/non-stochastic systems. CSS is a relational model where systems are defined by relations and unnormalized probability densities. This feature has several advantages. First, it makes the definition of the composition operation " | " relatively easy. Second, it provides us with a simple mechanism for specifying the conditional behavior of a system given that certain contraints are satisfied, which has the potential to be very useful when tracking cascades of events leading to system failures.

However, the system specification provided by CSS is generally not executable, i.e., it does not readily lead to a system implementation. To convert it to a form which can be simulated, we rely on a compilation, which examines the dependency relations, both non-stochastic and statistical, existing between the system variables. This compilation employs two operations. The marginal $\overline{\mathcal{S}}(.)$ and conditional $\mathcal{S}(.)$ extend to hybrid systems the standard marginal and conditional probability distributions of fully probabilized systems. With their help, we were able to introduce the notion of innovation, whereby π' is an innovation of π if, roughly speaking, π' does not influence π in the composition $\pi \mid \pi'$, but π may influence π', so that the interaction between π and π' is *oriented,* and $\pi \mid \pi'$ is amenable to incremental simulation. In general, systems interact in a non-oriented way. When the interaction graph of a system forms a tree, we have presented rules which can be used to convert the tree into a directed one while preserving equivalence of the compound system. In combination with the results of [26] for aggregating a triangulated graph into a tree, these rules can be used to compile arbitrary interaction graphs. A SIG implementation of the compilation rules was presented. Finally, it turns out that our simulation results can be adapted, with minor modifications, to the hidden state estimation of hybrid systems. We only need to replace the $\sum$ by the max in performing random compression.

Although CSS is obviously related to the theory of belief functions and belief networks developed in [20, 21, 22, 23], it differs from it in several respects. First, as mentioned earlier, unlike the Dempster-Shafer approach which relies on upper and lower probabilities in the space V_X of visible variables, we keep track of probability distributions on the random configurations. Second, through the introduction of the concept of innovation, which does not appear in the belief networks literature, CSS provides concrete solutions to basic problems such as hybrid system simulation and estimation. To our knowledge, no other approach offers this range of facilities.

The research presented here can be extended in several directions, we discuss only two of them, see [1] for more details.

- A first issue involves the introduction of two features currently missing from CSS, namely the specification of timing information, or the absence thereof, and the ability to define hybrid systems over infinite time intervals. These two features are already present in the previously introduced SIGNalea language [17], which represents an an extension of the SIGNAL synchronous real-time language [32, 33, 34]. The SIGNalea language generalizes stochastic Büchi automata, Petri nets, and our SIG minilanguage. But the mathematical foundations of SIGNalea in [17] are somewhat shaky and estimation is not included. Thus, generalizing CSS to SIGNalea is a high priority task, particularly since SIGNalea is currently under implementation.
- Also, our results need to be tested on real applications. Two applications of SIGNalea are now under consideration. The first one involves the implementation for Electricté de France of the nonintrusive appliance load monitoring scheme proposed in [6], which presents strong similarities with speech recognition, and for which Viterbi-style estimation algorithms are expected to be successful. A second potential application in the area of power generation concerns the design of a monitoring and diagnostic system from its risk analysis description. In this context, we would like to determine whether our relational model, because of its ability to track cascades of events leading to specific failures, presents advantages for risk analysis.

References

1. A. Benveniste, B. Levy, E. Fabre, and P. L. Guernic, "A calculus of stochastic systems for the specification, simulation, and hidden state estimation of hybrid stochastic/non-stochastic systems," Tech. Rep. to appear, Institut National de Recherche en Informatique et Automatique, Rocquencourt, France.
2. N. Viswanadham and Y. Narahari, *Performance Modeling of Automated Manufacturing Systems.* Englewood Cliffs, NJ: Prentice Hall, 1992.
3. L. R. Rabiner and B. H. Juang, "An introduction to hidden Markov models," *IEEE ASSP Magazine*, vol. 3, pp. 4–16, Jan. 1986.
4. S. Geman and D. Geman, "Stochastic relaxation, Gibbs distribution, and the Bayesian restoration of images," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 6, pp. 721–741, Nov. 1984.
5. R. C. Dubes and A. K. Jain, "Random field models in image analysis," *J. Applied Stat.*, vol. 12, pp. 131–164, 1989.
6. G. W. Hart, "Nonintrusive applicance load monitoring," *Proc. IEEE*, vol. 80, pp. 1870–1891, Dec. 1992.
7. M. Basseville and I. V. Nikiforov, *Detection of Abrupt Changes : Theory and Applications.* Englewood Cliffs, NJ: Prentice Hall, 1993.
8. T. Soderstrom and P. Stoica, *System Identification.* Englewood Cliffs, NJ: Prentice Hall, 1989.
9. M. Molloy, "Performance analysis using stochastic Petri nets," *IEEE Trans. Computers*, vol. 31, pp. 913–917, Sept. 1982.
10. B. Plateau and K. Atif, "Stochastic automata network for modeling parallel systems," *IEEE Trans. on Software Engineering*, vol. 17, pp. 1093–1108, Oct. 1991.
11. B. Plateau and J.-M. Fourneau, "A methodology for solving Markov models of parallel systems," *J. Parallel and Distributed Comput.*, vol. 12, pp. 370–387, 1991.

12. H. Hansson and B. Jonsson, "A calculus for communicating systems with time and probabilities," in *Proc. of the 11th IEEE Real-Time Systems Symposium*, (Los Alamitos), pp. 278–287, Dec. 1990.
13. B. Jonsson and K. Larsen, "Specification and refinement of probabilistic processes," in *Proc. 6th IEEE Int. Symp. on Logic in Computer Science*, (Amsterdam), pp. 266–277, July 1991.
14. A. Giacalone, C. Jou, and S. Smolka, "Algebraic reasoning for probabilistic concurrent systems," in *Proc. IFIP TC2 Working Conference on Programming Concepts and Methods*, 1989.
15. S. Hart and M. Sharir, "Probabilistic propositional temporal logic," *Information and Control*, vol. 70, pp. 97–155, 1986.
16. R. Alur, C. Courcoubetis, and D. Dill, "Model checking for probabilistic real-time systems," in *Proc. 18th Int. Coll. on Automata Languages and Programming (ICALP)*, 1991.
17. A. Benveniste, "Constructive probability and the SIGN*alea* language: Building and handling random processes with programming," Tech. Rep. 1532, Institut National de Recherche en Informatique et Automatique, Rocquencourt, France, Oct. 1991.
18. B. C. Levy, A. Benveniste, and R. Nikoukhah, "High-level primitives for recursive maximum likelihood estimation," Tech. Rep. 767, IRISA, Rennes, France, Oct. 1993.
19. G. D. Forney, "The Viterbi algorithm," *Proc. IEEE*, vol. 61, pp. 268–278, Mar. 1973.
20. A. P. Dempster, "Upper and lower probabilities induced by a multivalued mapping," *Annals Math. Statistics*, vol. 38, pp. 325–339, 1967.
21. A. P. Dempster, "A generalization of Bayesian inference (with discussion)," *Royal Stat. Soc., Series B*, vol. 30, pp. 205–247, 1968.
22. G. Shafer, *A Mathematical Theory of Evidence*. Princeton, NJ: Princeton Univ. Press, 1976.
23. P. P. Shenoi and G. Shafer, "Axioms for probability and belief function propagation," in *Uncertainty in Artificial Intelligence* (R. D. Shachter, T. S. Levitt, L. N. Kanal, and J. F. Lemmer, eds.), vol. 4, pp. 169–198, Amsterdam: North-Holland, 1990.
24. J. Pearl, "Fusion, propagation, and structuring in belief networks," *Artificial Intelligence*, vol. 29, pp. 241–288, Sept. 1986.
25. M. A. Peot and R. D. Shachter, "Fusion and propagation with multiple observations in belief networks," *Artificial Intelligence*, vol. 48, pp. 299–318, 1991.
26. S. L. Lauritzen and D. J. Spiegelhalter, "Local computations with probabilities on graphical structures and their application to expert systems (with discussion)," *J. Royal Stat. Soc., Series B*, vol. 50, pp. 157–224, 1988.
27. R. Kindermann and J. L. Snell, *Markov Random Fields and their Applications*. Providence, RI: American Mathematical Society, 1980.
28. C. Robert, *Modèles Statistiques pour l'Intelligence Artificielle*. Paris: Masson, 1991.
29. B. Prum and J. Fort, *Stochastic Processes on a Lattice and Gibbs Measure*. Boston, MA: Kluwer Acad. Publ., 1991.
30. C. Dellacherie and P. Meyer, *Probabilités et Potentiels*. Paris: Hermann, 1976.
31. A. P. Dempster, "Construction and local computation aspects of network belief functions," in *Influence Diagrams, Belief Nets, and Decision analysis* (R. M. Oliver and J. Q. Smith, eds.), ch. 6, pp. 121–141, Chichester, England: J. Wiley, 1990.
32. P. Le Guernic, T. Gauthier, M. Le Borgne, and C. Le Maire, "Programming real-time applications with SIGNAL" *Proc. IEEE*, vol. 79, pp. 1321–1336, Sept. 1991.
33. A. Benveniste and P. Le Guernic, "Hybrid dynamical systems theory and the SIGNAL language," *IEEE Trans. Automat. Contr.*, vol. 35, pp. 535–546, May 1990.
34. A. Benveniste, M. Le Borgne, and P. Le Guernic, "Hybrid systems : the SIGNAL approach," in *Lecture Notes in Computer Science*, vol. 736, pp. 230–254, Berlin: Springer Verlag, 1993.