ELSEVIER Theoretical Computer Science 152 (1995) 171-217

Theoretical Computer Science

Fundamental Study

A calculus of stochastic systems for the specification, simulation, and hidden state estimation of mixed stochastic/nonstochastic systems

Albert Benveniste[a,*], Bernard C. Levy[b], Eric Fabre[a], Paul Le Guernic[a]

[a] *IRISA-INRIA, Campus Universitaire de Beaulieu, 35042 Rennes Cedex, France*
[b] *Department of Electrical and Computer Engineering, University of California, Davis, CA 95616, USA*

Received November 1993; revised October 1994
Communicated by J. Sifakis

Abstract

In this paper, we consider *mixed systems* containing both stochastic and nonstochastic[1] components. To compose such systems, we introduce a general combinator which allows the specification of an arbitrary mixed system in terms of elementary components 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 mixed 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 mixed 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.

* Corresponding author. E-mail addresses: {benveniste,fabre,leguernic}@irisa.fr, levy@ece.ucdavis.edu.
[1] Throughout this paper, we use the word "nonstochastic" 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 "nonstochastic" here.

Contents

1. Introduction and motivation 172
2. CSS and the SIG mini-language 176
   2.1. Issues in mixed systems composition. 176
   2.2. CSS 177
   2.3. The SIG mini-language 178
3. Simulation. 184
   3.1. Compressing the random part of a system 184
   3.2. Two primitives. 186
   3.3. Properties of the primitives 187
   3.4. Incremental system simulation 189
   3.5. The SIG simulation compiler 194
4. Estimation. 196
   4.1. Maximum-likelihood compression of randoms. 197
   4.2. A framework for estimation, and two primitives. 198
   4.3. Properties of the primitives 199
   4.4. Incremental estimation. 201
   4.5. The SIG estimation compiler. 203
5. Discussion and conclusions. 205
Appendix A. Proof of Theorem 3.2 under Assumption A.1. 207
Appendix B. Proof of formula (3.7) and Lemma 3.1 under Assumption A.1 209
Appendix C. Proof of Lemma 3.2 under Assumption A.1. 210
Appendix D. Proof of Theorem 3.2, formula (3.7), and Lemmas 3.1 and 3.2 in the general case . . . 211
Appendix E. Examples of compound system compilation. 213
Acknowledgements. 216
References. 216

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 modeling and simulation tasks usually require modular models, which are often variations of stochastic Petri nets [36]. 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 (HMMs) [30] or Markov random fields [10, 12]. Both classes of models have proved quite successful 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 [16] 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 [2] 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 *mixed* stochastic/nonstochastic form of modeling is desirable. Several other key features that would need not 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 modeling, estimation, identification [34], and diagnostics [2] 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 [24, 36] are often used to specify stochastic systems, in applications such as queuing 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 [27, 28], 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 [14, 19, 35]. 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 [1, 13, 17]. 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 modeling.

This paper proposes a new and flexible form of calculus, called CSS, for the *specification, simulation, and hidden state estimation of mixed stochastic/nonstochastic systems.* The model of mixed stochastic/nonstochastic systems that we employ is introduced in Section 2.2. Mixed stochastic/nonstochastic systems interact via a single combinator that we call the *composition* and denote by "$|$". The combination of mixed systems with $|$ yields again mixed systems, and $|$ is both associative and commutative. When applied to purely nonstochastic systems, the composition operator $|$ 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 nonstochastic 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 nonstochastic and stochastic building blocks with $|$ allows the specification of arbitrary mixed 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 [3], and their study raises a number of technical issues that will be tackled elsewhere. Also, throughout the paper, only *finitely 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 [23], a precise description of such cases will not be attempted here.

Section 3 examines the *simulation* of mixed stochastic/nonstochastic 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 $\mathbf{p}(x,y) = \mathbf{p}(y|x)\mathbf{p}(x)$ provides a way to simulate incrementally the random variables (X, Y) with joint distribution $\mathbf{p}(x, y)$. We only need to draw X according to the distribution $\mathbf{p}(x)$ and then, for X given, draw Y based on the conditional distribution $\mathbf{p}(y|X)$. We generalize the notions of marginal and conditional distributions to mixed 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 mixed stochastic/nonstochastic systems is considered in Section 4. 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 $\mathbf{p}(x, y|z)$. The ML estimate, also known as the maximum a posteriori (MAP) estimate, of (X, Y) given Z is given by $(\hat{x}, \hat{y}) = \arg\max_{x,y} \mathbf{p}(x, y|z)$. To find these estimates incrementally, we can first compute the so-called "generalized likelihood" function $\mathbf{p}_{\mathscr{L}}(x|z) = \max_y \mathbf{p}(x, y|z)$. Next, compute the conditional likelihood $\mathbf{p}_{\mathscr{L}}(y|x,z) = \mathbf{p}(x, y|z)/\mathbf{p}_{\mathscr{L}}(x|z)$ of Y given X and Z, so that the following factorization holds: $\mathbf{p}(x, y) = \mathbf{p}_{\mathscr{L}}(y|x)\mathbf{p}_{\mathscr{L}}(x)$. Then, the desired ML estimates can be generated sequentially from $\hat{x} = \arg\max_x \mathbf{p}_{\mathscr{L}}(x|z)$ and $\hat{y} = \arg\max_y \mathbf{p}_{\mathscr{L}}(y|\hat{x}, z)$. This incremental estimation procedure, which is called the Viterbi algorithm in the HMM literature [11,30], 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 mixed systems, and show that the above dynamic programming procedure can be generalized accordingly. These primitives are implemented in SIG, and we demonstrate how 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 5 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 [7, 8, 32] and belief networks [21,25,26, 33] 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 mixed 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 [8] for combining belief functions. Also, our incremental simulation scheme is similar in nature to the fusion/propagation mechanism of [25,26]. 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 [25], since they are used to "*direct and activate the data flow in the computations* ..." [25]. 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 mixed stochastic/nonstochastic systems, which is used to express the composition rule $|$. To motivate our choice of combinator $|$, we first discuss several problems arising in the composition of mixed stochastic/nonstochastic models.

2.1. Issues in mixed systems composition

As noted in the introduction, the first requirement for the combinator $|$ is that, when applied to purely nonstochastic systems, it must behave as the conjunction of constraints (or, equivalently, as the intersection of legal behaviors). Let us examine this requirement in the context of automata. Fig. 1 illustrates the problem one faces when attempting to extend the usual product of automata to the class of stochastic

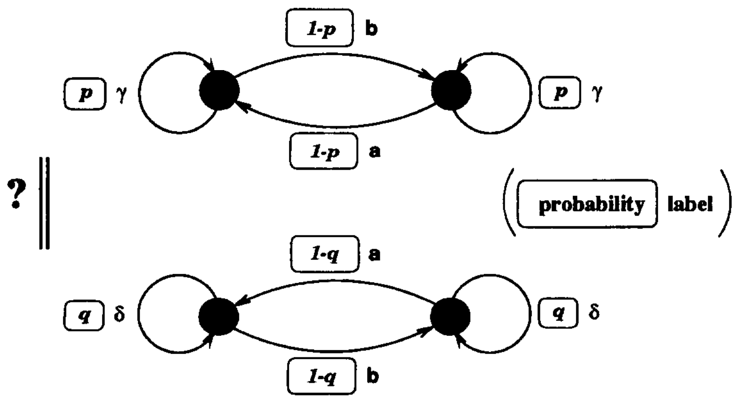

Fig. 1. Combining stochastic automata. Ignore temporarily the symbols within curved rectangles. Then the product of two automata is just the shuffle product of their associated infinite strings of symbols. This clearly satisfies our requirement concerning the intersection of behaviors. Next, take into account the transition probabilities appearing within the curved rectangles, so that we are now considering two interacting stochastic automata. Question 1: how would you define the composition of these two automata?

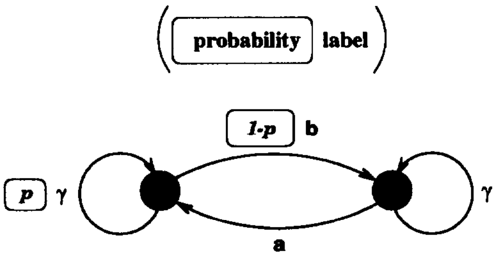

Fig. 2. Mixed stochastic/nonstochastic systems. The figure shows a system, where transitions from the state on the left are governed by probabilities, but transitions from the state on the right are just nondeterministic, i.e., unspecified. Question 2: in the language $(a\gamma^*b\gamma^*)^*$, which events would you view as random, or as nondeterministic (unspecified)? Next, let us make the labels a and b more concrete by assuming they represent the actions of generating two random variables X and Y, respectively, with values in a finite alphabet. Then, set $Z = f(X, Y)$ for some function f with values in a finite alphabet, and assume we hide a, b, X, Y, while keeping only γ, Z visible. Question 3: in the language $(Z\gamma^*Z\gamma^*)^*$; what events would you view as random, or as nondeterministic (unspecified)?

automata. Next, consider Fig. 2. It illustrates the difficulties arising in the description of mixed stochastic/nonstochastic systems. Suppose that instead of considering elementary events, such as state transitions, we seek to characterize more complex ones, such as complete behaviors. It becomes difficult to understand the status of such events. Should they be viewed as random, or just as nondeterministic? Also what are the consequences of the fact that certain variables are hidden? This is a natural consequence of modularity, since variables describing the inner interaction of subsystems are not visible when considering a large system.

Thus there is no doubt that a naive extension of nonstochastic composition will not work properly. Indeed, although the CCS composition rule has been extended to a probabilistic context by Jonsson et al. [18], it can only model specific systems but not general ones. Also, CCS composition does not perform the intersection of behaviors, and raises difficult questions of bisimulation.

On the other hand, the difficulties we have highlighted, concerning the nature of complex events or behaviors in mixed stochastic/nonstochastic system, together with the presence of hidden variables, were the motivation, in Dempster–Shafer and related theories, for the development of new axioms to define and manipulate events in large complex systems. Having demonstrated the need for a careful look at system composition, we now proceed with the introduction of the CSS model.

2.2. CSS

Model of mixed stochastic/nonstochastic systems

The mixed systems we consider are described by a quadruple

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

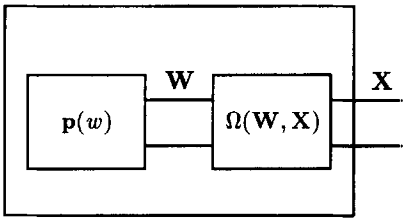

Fig. 3. A system in CSS. The outer box is the external boundary of the system, thus only X is visible from outside for further interaction.

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. $V_{\mathbf{W}}$); in this article we assume $\mathbf{W}$ is finite. 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 nonstochastic.
- $\mathbf{p}$ constitutes an unnormalized probability distribution[2] for $\mathbf{W}$. Specifically, we only require $\mathbf{p} \geqslant 0$ 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 behavior 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 nonstochastic case, we may consider that $V_{\mathbf{W}}$ consists of a single point w_{triv}, with $\mathbf{p}(w_{\text{triv}}) = 1$ and $\mathbf{p}(\emptyset) = 0$. Our notion of system is depicted in Fig. 3.

The unique system for which $\mathbf{X} = \emptyset$ is called NIL. Finally, given an arbitrary 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 $\text{FLAT}(\pi) \triangleq \{\mathbf{X}, \Omega, \mathbf{W}, 1\}$.

[2] 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 [20, 29, 31], the computation of the normalizing constant (the partition function) which transforms $\mathbf{p}$ into a true probability is often unnecessary.

Examples. (1) The above mixed systems contain as a subclass purely nonstochastic 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, Y_2) are random, only those which involve $f(X_1, X_2)$.

(3) For a system

$$\pi = \{X, (W_1, W_2), \Omega, \mathbf{p}\} \quad \text{with } \Omega : X = f(W_1, W_2). \tag{2.2}$$

where f is a noninjective 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 in Sections 3.1 and 4.1.

(4) The class of linear Gaussian mixed stochastic/nonstochastic systems of the form $EY = AX + BW$ was studied in detail in [23]. 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 mixed systems is closely related to the one employed by Dempster and Shafer [7, 8, 32] 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 $(V_\mathbf{W}, \mathbf{p})$, 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

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

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

Let us examine the modeling implications of the mixed 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 [2]. The subset $V_\mathbf{X}^\Omega$ of $V_\mathbf{X}$ satisfying the 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

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

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

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

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 not additive, projecting a probability from one space to another does not yield a probability, but a different object, called a *Choquet capacity* [6]. On $V_{\mathbf{X}}$, this capacity provides a partial probabilistic knowledge which was described by Dempster [7, 8] 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 mixed systems can be performed in the same manner as the combination of belief functions described in [7]. 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 $\pi_i=\{\mathbf{X}_i,\Omega_i,\mathbf{W}_i,\mathbf{p}_i\}$ with $i=1,2$, the combinator $\pi_1\mid\pi_2=\{\mathbf{X},\Omega,\mathbf{W},\mathbf{p}\}$ is defined as

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

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

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

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

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. Expressions (2.6a) and (2.6b) indicate, respectively, that variables may be shared, but not randoms.

The identities (2.6a)–(2.6d) show that the systems interact only through their shared variables. The NIL system is a neutral element for the combinator " | ". Our notion of composition is illustrated in Fig. 4.

Examples. (1) Consider two systems $\pi_i=\{\mathbf{X}_i,\Omega_i,\mathbf{W}_i,\mathbf{p}_i\}$ with $i=1,2$, where π_1 is purely stochastic, so that $\mathbf{X}_1$ and $\mathbf{W}_1$ have same cardinality and $\Omega_1: X_1=W_1,\ldots,X_p=$

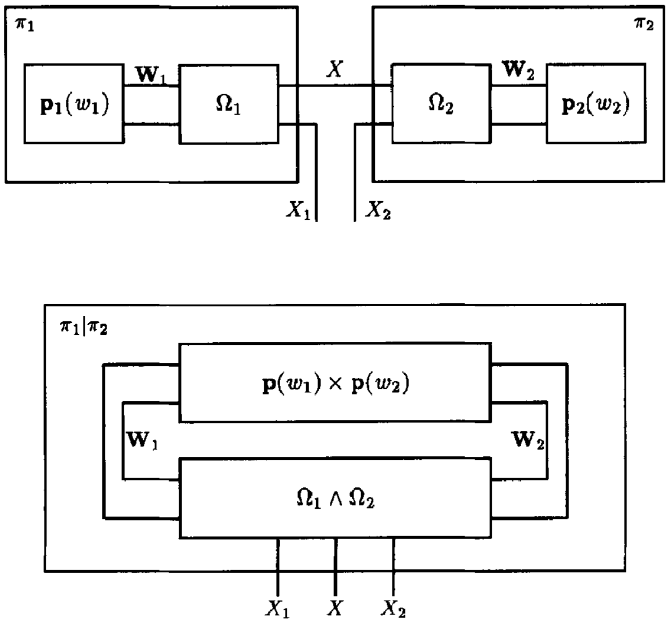

Fig. 4. The | composition. We consider two systems π_1, π_2 sharing the visible variable $\mathbf{X}$, but not $\mathbf{X}_1, \mathbf{X}_2$. The first picture illustrates how interaction occurs, and the second one shows its result.

W_p, and π_2 is purely nonstochastic, 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 $\pi_1 | \pi_2 = \{\mathbf{X}_1, (\Omega_1 \wedge \Omega_2), \mathbf{W}_1, \mathbf{p}_1\}$. The combined system $\pi_1 | \pi_2$ 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 $\mathbf{p}_1^{\Omega_2}$ of $\mathbf{p}_1$ 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 $\pi_i = \{\mathbf{X}_i, \Omega_i, \mathbf{W}_i, \mathbf{p}_i\}$, 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 $\pi_1 | \pi_2$, the randoms $\mathbf{W}_1$ and $\mathbf{W}_2$ are independent.[3]

2.3. *The* Sig *mini-language*

We now proceed to describe a syntax, in the form of the langage Sig, which implements both the modeling format and composition rule of CSS.

[3] According to the "maximum entropy principle" [31], among all joint distributions $\mathbf{p}(w_1, w_2)$ with prescribed marginals $\mathbf{p}_1(w_1)$ and $\mathbf{p}_2(w_2)$, the one which maximizes the entropy is given by $\mathbf{p}_1(w_1) \times \mathbf{p}_2(w_2)$, in which case the two components are independent.

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 nonstochastic 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

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

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

```
system LINEAR=(real, E, A, B)  % declaration of parameters
    { variable X, Y, U }
    ( | potential (U**2)/2      % gaussian noise
     | E*Y=A*X+B*U              % constraint on (X, Y, U)
     | )
end
```

specifies a "linear observation" $EY = AX + BU$ of the form introduced and studied in detail in [23]. To generate a HMM of the type discussed in [23, 30], we cam 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 distribution 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

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

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 mixed 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 $\mathbf{p}(y \mid x) = \mathbf{p}(x, y)/\mathbf{p}(x)$ of Y given X, so that we obtain the following factorization, also known as Bayes rule:

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

(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 $\mathbf{p}(y \mid X)$.

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

3.1. Compressing the random part of a system

Since the 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 $\mathbf{p}_{\mathrm{co}}(w_1) = \sum_{w_2} \mathbf{p}(w_1, w_2)$, 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 $B \subseteq V_\mathrm{W}$ 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 **p** to the sets of randoms satisfying this condition. Note in this respect that the family $\mathscr{W}$ of all sets B satisfying 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 $\mathbf{P}(\cdot\,|\,\mathscr{W})$ of $\mathbf{P}$ given $\mathscr{W}$. 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 $V_{\mathbf{W}}$ 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

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

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

$$p_{\text{co}}(w_{\text{co}}) \triangleq \sum_{w \in w_{\text{co}}} \mathbf{p}(w). \tag{3.5b}$$

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* $\{V_{\mathbf{W}_i}^{\Omega_i}, \mathscr{W}_i, \mathbf{p}_i\}$ *are isomorphic, so that there exists a one-to-one map ϕ from the σ-algebra $\mathscr{W}_1$ onto $\mathscr{W}_2$ such that* $\forall B_1 \in \mathscr{W}_1$, $\mathbf{P}_2(\phi(B_1)) = \mathbf{P}_1(B_1)$.
In addition, equivalence $\equiv$ is a congruence, which means that, if $\pi_i \equiv \pi_i'$ for $i = 1, 2$, then $(\pi_1\,|\,\pi_2) \equiv (\pi_1'\,|\,\pi_2')$.

The second statement is a direct consequence of the fact that, to get the compressed form of $(\pi_1\,|\,\pi_2)$, one can also (1) compress each π_i separately, (2) take the composition of the resulting compressed forms, and finally (3) re-compress the result. The property

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

which is proved in Appendices B and D, is a straightforward consequence of the notion of system equivalence. This identity generalizes to mixed systems the idempotence of composition property $\pi \mid \pi = \pi$ of purely nonstochastic systems.

Although the factorization (3.2) cannot be extended directly to mixed stochastic/nonstochastic systems, by employing Theorem 3.1, we develop below a general procedure for decomposing an arbitrary mixed 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 mixed systems by constructing the *marginal* and *conditional* systems

$$\text{MARGIN}_{\mathbf{X}'}(\pi) \quad \text{and} \quad \text{GIVEN}_{\mathbf{X}'}(\pi),$$

which will be denoted more compactly as

$$\bar{\mathscr{S}}_{\mathbf{X}'}(\pi) \quad \text{and} \quad \mathscr{S}_{\mathbf{X}'}(\pi),$$

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

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

$$\text{MARGIN}_{\mathbf{X}'}(\pi) = \bar{\mathscr{S}}_{\mathbf{X}'}(\pi) = \{\mathbf{X}', \Omega', \mathbf{W}, \mathbf{p}\}, \tag{3.8}$$

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

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

Note that neither the set of randoms $\mathbf{W}$ nor the distribution $\mathbf{p}$ are changed by this construction, which involves only tracking the effect of the projection of $\mathbf{X}$ onto $\mathbf{X}'$ in the relation Ω. An interesting use of the marginal is depicted in Fig. 5.

The conditional. The conditional system has the structure

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

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

$$\pi \equiv \text{MARGIN}_{\mathbf{X}'}(\pi) \mid \text{GIVEN}_{\mathbf{X}'}(\pi) = \bar{\mathscr{S}}_{\mathbf{X}'}(\pi) \mid \mathscr{S}_{\mathbf{X}'}(\pi) \tag{3.10b}$$

holds. Note that the relation $\equiv$ indicates that both sides have the same compressed form. The decomposition (3.10b) represents the extension to mixed systems of the

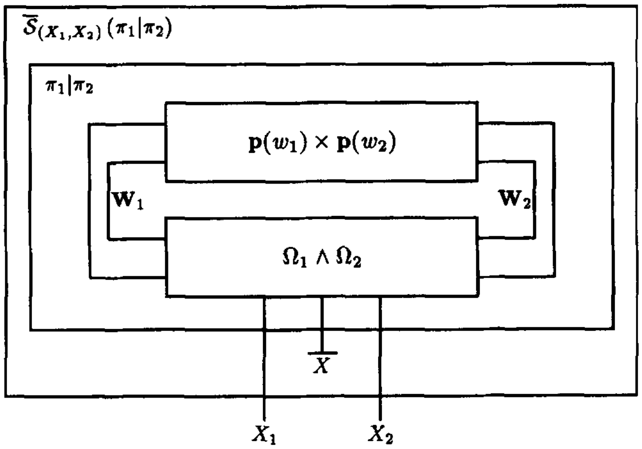
$\bar{S}_{(X_1,X_2)}(\pi_1|\pi_2)$
$\pi_1|\pi_2$
$p(w_1) \times p(w_2)$
W_1
W_2
$\Omega_1 \wedge \Omega_2$
X
X_1
X_2

Fig. 5. Hiding via the marginal. This is a continuation of Fig. 4. We consider again the system $\pi_1 | \pi_2$ and hide X in its interior. In Fig. 4, the interaction between π_1 and π_2 was carried by the shared variable **X** only. Since **X** is now hidden, we have an example of modeling systems interaction through a hidden regression variable.

factorization (3.2) of a probability distribution into marginal and conditional components. A procedure for constructing $\mathscr{S}_{\mathbf{X}'}(\pi)$ is presented in Appendices A and D, where we prove the following result.

Theorem 3.2. *There exists a system* $\bar{\mathscr{S}}_{\mathbf{X}'}(\pi)$ *with the structure* (3.10a), *and such that* (3.10b) *is satisfied.*

Although the particular construction of $\mathscr{S}_{\mathbf{X}'}(\pi)$ we provide in Appendices A and D guarantees uniqueness up to equivalence, we do not know whether the solution of factorization equation (3.10b) for a system of the form (3.10a) is unique up to equivalence.

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

$$\bar{\mathscr{S}}_{\mathbf{Z}}(\pi) \triangleq \bar{\mathscr{S}}_{\mathbf{Z} \cap \mathbf{X}}(\pi), \qquad \mathscr{S}_{\mathbf{Z}}(\pi) \triangleq \mathscr{S}_{\mathbf{Z} \cap \mathbf{X}}(\pi). \tag{3.11}$$

3.3. Properties of the primitives

The operations that we have just introduced admit a number of properties which will be employed extensively in the sequel. They are collected in the following lemma, which is proved in Appendices B and D.

Lemma 3.1. *The marginal and conditional primitives satisfy the following properties:*[4]

$$\pi \equiv \bar{\mathscr{S}}_{\mathbf{X}'}(\pi) \,|\, \mathscr{S}_{\mathbf{X}'}(\pi), \tag{3.12}$$

$$\mathscr{S}_{\mathbf{X}'} \circ \bar{\mathscr{S}}_{\mathbf{X}'}(\pi) = \bar{\mathscr{S}}_{\mathbf{X}'} \circ \mathscr{S}_{\mathbf{X}'}(\pi) = \text{FLAT} \circ \bar{\mathscr{S}}_{\mathbf{X}'}(\pi), \tag{3.13}$$

$$\bar{\mathscr{S}}_{\mathbf{Z}} \circ \bar{\mathscr{S}}_{\mathbf{Y} \cup \mathbf{Z}}(\pi) = \bar{\mathscr{S}}_{\mathbf{Z}}(\pi), \tag{3.14}$$

$$\mathscr{S}_{\mathbf{Y} \cup \mathbf{Z}}(\pi) \circ \mathscr{S}_{\mathbf{Z}}(\pi) = \mathscr{S}_{\mathbf{Y} \cup \mathbf{Z}}(\pi), \tag{3.15}$$

$$\bar{\mathscr{S}}_{\mathbf{X}_1}(\pi_1 \,|\, \pi_2) = \pi_1 \,|\, \bar{\mathscr{S}}_{\mathbf{X}_1}(\pi_2), \tag{3.16}$$

where in (3.16) $\mathbf{X}_1$ *denotes the set of variables of* π_1.

Note that (3.12) corresponds to the requirement (3.10b) for the conditional; also, $\equiv$ and $=$ symbols have been carefully used in this lemma. Identity (3.13) indicates that be successively applying the marginal and conditional primitives to a system, we obtain a flat distribution. Expression (3.14) shows that the marginal $\bar{\mathscr{S}}(\cdot)$ behaves like a projection, and (3.15) represents the dual property satisfied by the conditional.

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 $\pi_1 \,|\, \pi_2$, 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 **X** followed by **Y** is that combining $\mathbf{p}(y\,|\,x)$ with $\mathbf{p}(x)$ does not modify the behavior of **X**. In other words, $\mathbf{p}(y\,|\,x)$ represents totally new information with no bearing on **X**. This feature leads us to introduce the notion of *innovation* which extends to mixed 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 **Y** denotes an arbitrary set of variables, the system π_2 is said to be a **Y**-innovation of π_1, which we denote as

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

if

$$\bar{\mathscr{S}}_{\mathbf{X}_1 \cup \mathbf{Y}} \circ \mathscr{S}_{\mathbf{Y}}(\pi_2) \,|\, \mathscr{S}_{\mathbf{Y}}(\pi_1) \equiv \mathscr{S}_{\mathbf{Y}}(\pi_1). \tag{3.17}$$

For the special case when **Y** is empty, we just say that π_2 is an *innovation* of π_1, which is written as $\pi_2 \perp\!\!\!\perp \pi_1$.

[4] $f \circ g(x)$ denotes the composition of maps $f(g(x))$.

Thus, π_2 represents a **Y**-innovation of π_1 if composing $\mathscr{S}_{\mathbf{Y}}(\pi_2)$ with $\mathscr{S}_{\mathbf{Y}}(\pi_1)$ does not modify $\mathscr{S}_{\mathbf{Y}}(\pi_1)$. In particular, when **Y** is empty, this means that, to simulate $\pi_1 \,|\, \pi_2$, we can equivalently first simulate π_1, and then, having the outcome of this first partial simulation, subsequently simulate π_2. Thus a pictorial view of property $\pi_2 \perp\!\!\!\perp_{\mathbf{Y}} \pi_1$ could be that, given **Y**, the interaction between π_1 and π_2 is oriented from π_1 to π_2. This graphical view of innovations will be extensively used in the sequel, in particular for the examples of Section 3.5.

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 **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 of this order. The compilation rules will rely on the following properties of innovations, which are proved in Appendices C and D.

Lemma 3.2. *Given an arbitrary system π, and a subset $\mathbf{X}'$ of its variables, we have*

$$\mathscr{S}_{\mathbf{X}'}(\pi) \perp\!\!\!\perp \bar{\mathscr{S}}_{\mathbf{X}'}(\pi), \tag{3.18}$$

*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 **Y**-innovation of π_1, the following identities hold:*

$$\bar{\mathscr{S}}_{\mathbf{X}_1 \cup \mathbf{Y}}(\mathscr{S}_{\mathbf{Y}}(\pi_2) \,|\, \mathscr{S}_{\mathbf{Y}}(\pi_1)) = \mathscr{S}_{\mathbf{Y}}(\pi_1), \tag{3.19}$$

$$\mathscr{S}_{\mathbf{Y}}(\pi_1 \,|\, \pi_2) \equiv \mathscr{S}_{\mathbf{Y}}(\pi_1) \,|\, \mathscr{S}_{\mathbf{Y}}(\pi_2), \tag{3.20}$$

$$\bar{\mathscr{S}}_{\mathbf{Y}}(\pi_1 \,|\, \pi_2) \equiv \bar{\mathscr{S}}_{\mathbf{Y}}(\pi_1) \,|\, \bar{\mathscr{S}}_{\mathbf{Y}}(\pi_2), \tag{3.21}$$

3.4. Incremental system simulation

Consider now a compound system of the form

$$\pi = \,|_{i \in I}\, \pi_i, \tag{3.22}$$

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 porbabilities of complex events.

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

$$\pi_1 - \mathbf{X} - \pi_2 \quad \text{if} \begin{cases} \pi_2 \text{ is not an innovation of } \pi_1, \text{ and} \\ \pi_1 \text{ is not an innovation of } \pi_2. \end{cases} \tag{3.23a}$$

Similarly, we write

$$\pi_1 \rightarrow \mathbf{X} \rightarrow \pi_2 \quad \text{if} \begin{cases} \pi_2 \text{ is an innovation of } \pi_1, \text{ but} \\ \pi_1 \text{ is not an innovation of } \pi_2. \end{cases} \tag{3.23b}$$

and

$$\pi_1 \leftrightarrow \mathbf{X} \leftrightarrow \pi_2 \quad \text{if} \begin{cases} \pi_2 \text{ is an innovation of } \pi_1 \text{ and} \\ \pi_1 \text{ is an innovation of } \pi_2. \end{cases} \tag{3.23c}$$

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 nonstochastic, 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 (3.22). 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 (3.23a)–(3.23c) 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 [21, 25, 26] or belief functions [9, 33]. Such networks, as well as the execution of graphs described above, find their root in the standard graphical representation of Markov random fields in terms of cliques of neighbors [20]. However, while the graphs of Markov random fields are undirected, like the branches produced in (3.23a), 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 (3.22) 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 3.3. *Consider a partition $I = J \cup J^c$ with $J \cap J^c = \emptyset$, for which we write*

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

$$\mathbf{X}_J = \textit{set of private variables of } \pi_J,$$

$$\mathbf{X}_J^c = \textit{set of private variables of } \pi_J^c,$$

$$\partial \mathbf{X} = \textit{set of shared variables of } \pi_J \textit{ and } \pi_J^c.$$

(1) *We have*

$$\bar{\mathscr{S}}_{\mathbf{X}_J}(\pi) = \bar{\mathscr{S}}_{\mathbf{X}_J}(\pi_J \mid \bar{\mathscr{S}}_{\partial\mathbf{X}}(\pi_J^c)). \tag{3.24}$$

(2) *Under the stronger assumption*

$$\pi_J \to \partial\mathbf{X} \to \pi_J^c \text{ or } \pi_J \leftrightarrow \partial\mathbf{X} \leftrightarrow \pi_J^c, \tag{3.25}$$

identity (3.24) *reduces to*

$$\bar{\mathscr{S}}_{\mathbf{X}_J}(\pi) = \bar{\mathscr{S}}_{\mathbf{X}_J}(\pi_J), \tag{3.26}$$

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

Proof. We first prove (1). By using the decomposition (3.12) for π_J^c with $\mathbf{X}' = \partial\mathbf{X}$, and observing that $\mathscr{S}_{\partial\mathbf{X}}(\pi_J^c)$ is an $\mathbf{X}_J$-innovation of $\pi_J \mid \bar{\mathscr{S}}_{\partial\mathbf{X}}(\pi_J^c)$, we get

$$\begin{aligned}\bar{\mathscr{S}}_{\mathbf{X}_J}(\pi) &= \bar{\mathscr{S}}_{\mathbf{X}_J}(\pi_J \mid \pi_J^c)\\ &= \bar{\mathscr{S}}_{\mathbf{X}_J}(\pi_J \mid \bar{\mathscr{S}}_{\partial\mathbf{X}}(\pi_J^c) \mid \mathscr{S}_{\partial\mathbf{X}}(\pi_J^c))\\ &= \bar{\mathscr{S}}_{\mathbf{X}_J}(\pi_J \mid \bar{\mathscr{S}}_{\partial\mathbf{X}}(\pi_J^c)) \mid \bar{\mathscr{S}}_{\mathbf{X}_J} \circ \mathscr{S}_{\partial\mathbf{X}}(\pi_J^c),\end{aligned} \tag{3.27}$$

where the last equality is due to property (3.21) of innovations. Next, using successively properties (3.14) and (3.13) of primitives, and noting the system π_J^c has no variables in $\mathbf{X}_J$, we obtain

$$\begin{aligned}\bar{\mathscr{S}}_{\mathbf{X}_J} \circ \mathscr{S}_{\partial\mathbf{X}}(\pi_J^c) &= \bar{\mathscr{S}}_{\mathbf{X}_J} \circ \bar{\mathscr{S}}_{\mathbf{X}_J \cup \partial\mathbf{X}} \circ \mathscr{S}_{\mathbf{X}_J \cup \partial\mathbf{X}}(\pi_J^c)\\ &= \bar{\mathscr{S}}_{\mathbf{X}_J} \circ \text{FLAT} \circ \bar{\mathscr{S}}_{\partial\mathbf{X}}(\pi_J^c) = \text{NIL},\end{aligned} \tag{3.28}$$

which together with (3.27), proves (3.24).

We now derive part (2) of the lemma. Under assumption (3.25), property (3.19) of innovations implies

$$\bar{\mathscr{S}}_{\mathbf{X}_J}(\pi_J \mid \bar{\mathscr{S}}_{\partial\mathbf{X}}(\pi_J^c)) = \bar{\mathscr{S}}_{\mathbf{X}_J}(\pi_J), \tag{3.29}$$

which proves (2). □

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, i.e., if simulating sequentially the successive (ordered) components yields a simulation of the compound system. This notion is now formalized (we use Theorem 3.1 in formulating this definition).

Definition 3.2 (*Incremental simulation*). A compound system $\pi = \|_{i\in I}\, \pi_i$ is said to admit an incremental simulation if there exists a total order $\prec$ on I such that the simulation of π can be performed as follows. Denote by $\mathbf{X}_i$ the set of visible variables of component π_i. Then for all $i_0 \in I$, once actual values of the variables in $\bigcup_{i \prec i_0} \mathbf{X}_i$ are given, the variables of $\mathbf{X}_{i_0}$ can be drawn using π_{i_0} only.

Theorem 3.3. *A compound system π of the form* (3.22) *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 (3.23a), and no directed cycle. When determining whether the graph contains cycles, all bidirectional branches of the form (3.23c) can be used as "wild cards" whose orientation can be selected so as to break potential cycles.

Proof of Theorem 3.3. An important property of a directed graph is that it is acyclic if and only if we can define a total order $\prec$ on its vertices which is compatible with the orientation of its branches (see [15, p. 200]). Consequently, if ExecGraph(π) is a directed acyclic graph, we can reorder its components π_i in such a way that for all pairs (π_i, π_j) which are connected by a branch of the form (3.23b), we have $i \prec j$. Then, according to part (2) of Lemma 3.3, the system π can be simulated incrementally by generating its components π_i sequentially, for increasing values of i.

Conversely, suppose ExecGraph(π) contains either an undirected branch of the form (3.23a) or a cycle. Then we can find a partition of π such that π_J and π_J^c are linked by an undirected branch. When ExecGraph(π) contains a cycle, such a partition is generated by letting the cycle straddle π_J and its complement. Then, ExecGraph(π) contains both branches going from π_J to π_J^c, and viceversa, so that when all the subsystems of π_J and its complement are aggregated, the branches going in opposite directions between π_J and π_J^c collapse into an undirected branch of the form (3.23a). Thus, according to part (1) of Lemma 3.3, in order simulate the variables X_J of π, we must solve a fixed point equation of the form (3.24), which prohibits incremental simulation. $\square$

As a side remark, note that fixed-point equations of the form (3.24) can be solved iteratively by employing stochastic relaxation methods such as the Metropolis algorithm or the Gibbs sampler [12]. However, such schemes fall outside the scope of the incremental simulation procedures described here.

Next, since most compound systems of the form (3.22) 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 3.4. *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 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 neighbor 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

$$\pi_i \equiv \bar{\mathscr{S}}_{\mathbf{X}_{i_-}}(\pi_i) \,|\, \mathscr{S}_{\mathbf{X}_{i_-}}(\pi_i), \tag{3.30}$$

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

Rule 2: If the factorization (3.30) has been performed, reorganize the compound system π by rewriting

$$\pi_{i_-} \,|\, \pi_i \;\equiv\; \pi'_{i_-} \,|\, \pi'_i \tag{3.31a}$$

with

$$\pi'_{i_-} \triangleq \pi_{i_-} \,|\, \bar{\mathscr{S}}_{\mathbf{X}_{i_-}}(\pi_i), \qquad \pi'_i \triangleq \mathscr{S}_{\mathbf{X}_{i_-}}(\pi_i). \tag{3.31b}$$

This reorganization clearly preserves the structure of the interaction graph of π, as well as the variables of subsystems π_{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 Rules 1 and 2 to this sequence, we find that once the transformation (3.31a) and (3.31b) has been applied to node i, the new system π' includes the branch

$$\pi'_{i_-} \to \mathbf{X}_{i_-,i} \to \pi'_i \tag{3.32}$$

in its execution graph, where $\mathbf{X}_{i_-,i}$ represents the set of shared variables of π'_{i_-} and π'_i. Then when Rules 1 and 2 are subsequently applied to system π'_{i_-j}, π'_{i_-} may change, but the orientation of the branch (3.32) 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. □

Unfortunately, the above result does not tell us what to do for systems whose interaction graph is not a tree. For such systems, we can always transform the interaction graph of π into a tree by aggregating some of its vertices. Such an aggregation has the effect of regrouping the subsystems π_i of π into larger clumps. In doing so, it is desirable to keep the number of aggregations to a minimum, as well as to ensure that the structure of the resulting tree does not depend on the order in which aggregations are performed. A simple solution to this problem was presented in [21] (see also [37, Ch. 12]) for the case of triangulated graphs. Recall that a graph is triangulated if it does not include chordless cycles. This solution relies on the fact that if a graph is triangulated, the hypergraph formed by its maximum cliques is acyclic. A hypergraph differs from a graph by the fact that its "edges" are actually subsets of the vertex set. Also, the cliques of a graph are sets of mutual neighbors, and a clique is said to be maximal if it is not contained in another clique. Consequently, given a compound system π whose interaction graph is triangulated, to aggregate it into a tree, we need only to find its maximum cliques, and aggregate together the corresponding subsystems. Note that a given subsystem π_i may belong to several maximum cliques, and thus will be aggregated into several clumps. The resulting aggregated graph forms a tree, to which we can then apply the compilation procedure of Theorem 3.4. When the interaction graph of π is not triangulated, we can always triangulate it by adding branches. However, the new branches must be selected judiciously, since different branch fill-in strategies may lead to triangulated graphs with different numbers of cliques, and cliques of different sizes.

The application of the compilation rules and aggregation scheme we have just described are illustrated in Appendix E by considering two examples, one for which the interaction graph forms a tree, and one where it contains a cycle.

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 $\bar{\mathscr{S}}_{X,Y}$(SYSTEM) and conditional $\mathscr{S}_{X,Y}$(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

$$\begin{tikzcd}
\pi_0 \arrow[r] & x_0, x_1 \arrow[r] \arrow[d] & \pi_1 \arrow[r] & x_1, x_2 \arrow[r] \arrow[d] & \pi_2 & \cdots & \pi_{N-1} \arrow[r] & x_{N-1}, x_N \arrow[r] \arrow[d] & \pi_N \\
 & \sigma_1 & & \sigma_2 & & & & \sigma_N &
\end{tikzcd} \tag{3.33}$$

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 3.3, 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

$$\begin{tikzcd}
\pi_0 \arrow[r, dash] & x_0, x_1 \arrow[r, dash] \arrow[d, dash] & \pi_1 \arrow[r, dash] & x_1, x_2 \arrow[r, dash] \arrow[d, dash] & \pi_2 & \cdots & \pi_{N-1} \arrow[r, dash] & x_{N-1}, x_N \arrow[r, dash] \arrow[d, dash] & \pi_N \\
 & \sigma_1 & & \sigma_2 & & & & \sigma_N &
\end{tikzcd} \tag{3.34}$$

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 3.4. 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 $\Longleftrightarrow$ 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 (3.33), so that it is ready for simulation.

4. Estimation

Consider a pair (X, Y) of random variables with joint distribution $\mathbf{p}(x, y)$. The maximum likelihood estimate of (X, Y) is given by

$$(\hat{x}, \hat{y}) \triangleq \arg\max_{x,y} \mathbf{p}(x, y). \tag{4.1}$$

When the pair (X, Y) is unknown, but we observe a third variable Z, the ML estimate of (X, Y) given Z, which is sometimes called the maximum a posteriori (MAP) estimate, is obtained by replacing $\mathbf{p}(x, y)$ by $\mathbf{p}(x, y \mid z)$ in the above expression. This estimate can be generated incrementally by employing the following procedure:

(1) Compute the generalized likelihood

$$\mathbf{p}_{\mathscr{L}}(x) \triangleq \max_{y} \mathbf{p}(x, y) \tag{4.2}$$

of X based on $\mathbf{p}$.

(2) Compute the conditional likelihood $\mathbf{p}_{\mathscr{S}}(y \mid x) \triangleq \mathbf{p}(x, y)/\mathbf{p}_{\mathscr{S}}(x)$ of Y given X, so that the following factorization holds:

$$\mathbf{p}(x, y) = \mathbf{p}_{\mathscr{S}}(y \mid x)\mathbf{p}_{\mathscr{S}}(x). \tag{4.3}$$

Note that this factorization differs from the Bayes rule appearing in (3.2).

(3) Find $\hat{x} = \arg\max_x \mathbf{p}_{\mathscr{S}}(x)$, and then select $\hat{y} = \arg\max_y \mathbf{p}_{\mathscr{S}}(y \mid \hat{x})$. The estimated pair $(\hat{x}, \hat{y})$ coincides with (4.1).

The above incremental estimation procedure is just a simple form of dynamic programming, which is also called the Viterbi algorithm in the HMM literature [11, 30]. We now generalize this technique to mixed systems. The approach we follow parallels the one employed to extend Bayes rule to mixed systems for the simulation case.

4.1. Maximum-likelihood compression of randoms

We modify the notion of compression introduced in Section 3.1 to account for the fact that while the marginal probability (3.1) was obtained from the distribution $\mathbf{p}(x, y)$ by performing a *summation* over y, the generalized likelihood (4.2) requires a *maximization* over y. To track the effect of this change, we examine again the simple example consisting of a pair (W_1, W_2) of randoms, where W_1 is completely visible through a variable $X_1 = W_1$, but W_2 has no effect on the variables. To eliminate W_2, we can replace it by its ML estimate $\hat{W}_2$, thus yielding the new distribution $\mathbf{p}_{\mathrm{ML-co}}(w_1) = \max_{w_2} \mathbf{p}(w_1, w_2) = \mathbf{p}(w_1, \hat{w}_2)$ for the remaining random W_1. The procedure employed to reduce the original system to the quadruple $(X_1, X_1 = W_1, \mathbf{p}_{\mathrm{ML-co}}(w_1), W_1)$ represents an elementary case of the maximum likelihood compression procedure described below.

Given a system π and the equivalence relation $\sim_\pi$ introduced in (4.4), we obtain the *ML-compression* of a system π as follows:

(1) Compress the set $V_{\mathbf{W}}$ of random experiments by retaining only the equivalence classes w_{co} for the relation $\sim_\pi$.

(2) Compress the relation Ω accordingly, by setting

$$\Omega_{\mathrm{co}}(x; w_{\mathrm{co}}) \triangleq \Omega(x; w) \quad \text{for } w \in w_{\mathrm{co}}. \tag{4.4}$$

(3) Assign to each equivalence class w_{co} the generalized likelihood

$$\mathbf{p}_{\mathrm{ML-co}}(w_{\mathrm{co}}) \triangleq \max_{w \in w_{\mathrm{co}}} \mathbf{p}(w). \tag{4.5}$$

Two systems π and π' having the same ML-compressed form are said to be *ML-equivalent*, which is written as

$$\pi \equiv_{\mathscr{S}} \pi'. \tag{4.6}$$

Note that, while compression and ML-compression are *different operations*, any system which is compressed is also ML-compressed, and vice versa, since compression is a feature of the relation Ω, not of $\mathbf{p}$. The notion of ML-equivalence motivates the following result.

Theorem 4.1. *If* $\pi_i = \{\mathbf{X}_i, \Omega_i, \mathbf{W}_i, \mathbf{p}_i\}$, $i = 1, 2$ *are ML-equivalent in the sense of* (4.6), *they cannot be distinguished under estimation, so that 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 same generalized likelihoods* $\mathbf{p}_{\mathrm{ML-co}}^{i}$, $i = 1, 2$.

Remark. Since most of the properties of the marginal and conditional primitives derived in the previous section are of an algebraic nature, they remain valid if we replace the "$\sum$" operation by a "max", i.e., as we perform ML-compressions instead of compressions. Consequently, our earlier incremental simulation results can be adapted with little effort to the incremental estimation case.

4.2. A framework for estimation, and two primitives

Consider a system $\pi = \{\mathbf{X}, \Omega, \mathbf{W}, \mathbf{p}\}$, and a partition

$$\mathbf{X} = \mathbf{Y} \cup \mathbf{Z}$$

of the system variables $\mathbf{X}$ into *observations* $\mathbf{Y}$ and *unknowns* $\mathbf{Z}$. We seek to

(1) estimate the unknowns from the observations, and
(2) determine how likely the observations are by replacing the unknowns by their estimated values.

In the following, when considering a system π, we shall specify how its variables are partitioned into observations and unknowns by denoting

$$\pi = \{(\mathbf{Y}, \mathbf{Z}), \Omega, \mathbf{W}, \mathbf{p}\}.$$

For such a system, if $\mathbf{Z}' \subset \mathbf{Z}$ denotes a subset of its unknowns, we now construct the *generalized likelihood* and *conditional likelihood* systems

$$\text{ESTIMATE}_{\mathbf{Z}'}(\pi) \quad \text{and} \quad \text{KNOWING}_{\mathbf{Z}'}(\pi),$$

which will also be denoted more compactly as

$$\bar{\mathscr{E}}_{\mathbf{Z}'}(\pi) \quad \text{and} \quad \mathscr{E}_{\mathbf{Z}'}(\pi),$$

where the symbol $\mathscr{E}$ is employed here as a mnemonic for $\mathscr{E}$stimation.

The generalized likelihood. It consists of eliminating from π the unknowns not in $\mathbf{Z}'$, which yields

$$\text{ESTIMATE}_{\mathbf{Z}'}(\pi) = \bar{\mathscr{E}}_{\mathbf{Z}'}(\pi) = \{(\mathbf{Y}, \mathbf{Z}'), \Omega', \mathbf{W}, \mathbf{p}\}, \tag{4.7}$$

where Ω' denotes the relation obtained by using the existential quantifier $\exists$ to eliminate from Ω the unknowns not in $\mathbf{Z}'$, so that

$$\Omega'(y, z'; w) \triangleq \exists z'' : \Omega((y, z', z''); w). \tag{4.8}$$

Note that neither the randoms $\mathbf{W}$ nor the distribution $\mathbf{p}$ are changed by this construction, which involves only tracking the effect of the projection of $\mathbf{Z}$ onto $\mathbf{Z}'$ in the relation Ω. Note also that, at this point, there is no difference between $\bar{\mathscr{E}}(\cdot)$ and $\bar{\mathscr{S}}(\cdot)$, since we have $\bar{\mathscr{E}}_{\mathbf{Z}'}(\pi) = \bar{\mathscr{S}}_{\mathbf{Y} \cup \mathbf{Z}'}(\pi)$.

The conditional likelihood. It is uniquely specified by requiring it should have the structure

$$\text{KNOWING}_{\mathbf{Z}'}(\pi) = \mathscr{E}_{\mathbf{Z}'}(\pi) = \{(\mathbf{Y}, \mathbf{Z}), \Omega, \mathbf{W}, \mathbf{p}''\}, \tag{4.9a}$$

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

$$\pi \equiv_{\mathscr{S}} \bar{\mathscr{E}}_{\mathbf{Z}'}(\pi) \,|\, \mathscr{E}_{\mathbf{Z}'}(\pi) \tag{4.9b}$$

holds. Recall that $\equiv_{\mathscr{S}}$ indicates that both sides have the same ML-compressed form. Note that $\mathscr{E}_{\mathbf{Z}'}(\pi)$ modifies only the distribution $\mathbf{p}$. The factorization (4.9b) extends to mixed systems, the factorization (4.3) of a probability distribution into generalized and conditional likelihoods. Finally, observe that since $\equiv$ and $\equiv_{\mathscr{S}}$ are different equivalence relations, the primitives $\mathscr{S}(\cdot)$ and $\mathscr{E}(\cdot)$ are different.

Notation. In the following, it will be convenient to extend the definition of $\bar{\mathscr{E}}_{\mathbf{U}}(\pi)$ and $\mathscr{E}_{\mathbf{U}}(\pi)$ to the case where the set $\mathbf{U}$ is not included in $\mathbf{Z}$, by denoting

$$\bar{\mathscr{E}}_{\mathbf{U}}(\pi) \triangleq \bar{\mathscr{E}}_{\mathbf{U} \cap \mathbf{Z}}(\pi), \qquad \mathscr{E}_{\mathbf{U}}(\pi) \triangleq \mathscr{E}_{\mathbf{U} \cap \mathbf{Z}}(\pi). \tag{4.10}$$

4.3. Properties of the primitives

The generalized likelihood and conditional likelihood primitives admit a number of properties which are collected in the following lemma. Its proof, as well as that of almost all the results of this section, is omitted, since as observed at the end of Section 4.1, it involves only replacing summations by maximizations in the results of the previous section.

Lemma 4.1. *The generalized likelihood and conditional likelihood primitives satisfy the identities*:

$$\pi \equiv_{\mathscr{S}} \bar{\mathscr{E}}_{\mathbf{Z}'}(\pi) \,|\, \mathscr{E}_{\mathbf{Z}'}(\pi), \tag{4.11}$$

$$\mathscr{E}_{\mathbf{Z}'} \circ \bar{\mathscr{E}}_{\mathbf{Z}'}(\pi) = \bar{\mathscr{E}}_{\mathbf{Z}'} \circ \mathscr{E}_{\mathbf{Z}'}(\pi) = \text{FLAT} \circ \bar{\mathscr{E}}_{\mathbf{Z}'}(\pi), \tag{4.12}$$

$$\bar{\mathscr{E}}_{\mathbf{Z}} \circ \bar{\mathscr{E}}_{\mathbf{U} \cup \mathbf{Z}}(\pi) = \bar{\mathscr{E}}_{\mathbf{Z}}(\pi), \tag{4.13}$$

$$\mathscr{E}_{\mathbf{U} \cup \mathbf{Z}} \circ \mathscr{E}_{\mathbf{Z}}(\pi) = \mathscr{E}_{\mathbf{U} \cup \mathbf{Z}}(\pi), \tag{4.14}$$

$$\bar{\mathscr{E}}_{\mathbf{Z}_1}(\pi_1 \,|\, \pi_2) = \pi_1 \,|\, \bar{\mathscr{E}}_{\mathbf{Z}_1}(\pi_2), \tag{4.15}$$

where in (4.15) $\mathbf{Z}_1$ *denotes the set of unknowns of* π_1.

Note that (4.11) is just a restatement of the factorization requirement (4.9b) for the conditional likelihood. Identity (4.12) indicates that by successively applying the generalized likelihood and conditional likelihood primitives, we obtain a flat distribution. Expression (4.13) shows that the generalized likelihood $\bar{\mathscr{E}}(\cdot)$ behaves like a projection, and (4.14) represents the dual property satisfied by the conditional likelihood.

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 estimate the composition $\pi_1 \mid \pi_2$, we only need to estimate 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 shows that incremental estimation can be performed under weaker assumptions. The key feature of the factorization (4.3) which makes it possible to estimate X first, and they Y, is that combining $\mathbf{p}_{\mathscr{L}}(y \mid x)$ with $\mathbf{p}_{\mathscr{L}}(x)$ does not modify the estimate of X. This remark naturally leads to the notion of ML-innovation we introduce now.

Definition 4.1 (*ML-innovation*). Let π_i and $\mathbf{Z}_i$, $i = 1, 2$, denote two systems and their unknowns. If $\mathbf{U}$ denotes an arbitrary set of variables, π_2 is said to be a $\mathbf{U}$-*ML*-innovation of π_1, which we denote as

$$\pi_2 \perp\!\!\!\perp_{\mathbf{U}}^{\mathscr{L}} \pi_1$$

if

$$\bar{\mathscr{E}}_{\mathbf{Z}_1 \cup \mathbf{U}} \circ \mathscr{E}_{\mathbf{U}}(\pi_2) \mid \mathscr{E}_{\mathbf{U}}(\pi_1) \equiv_{\mathscr{L}} \mathscr{E}_{\mathbf{U}}(\pi_1). \tag{4.16}$$

If $\mathbf{U}$ is empty, we just say that π_2 is a ML-innovation of π_1, which we write $\pi_2 \perp\!\!\!\perp^{\mathscr{L}} \pi_1$.

Thus, π_2 represents a U-ML-innovation of π_1 if composing $\mathscr{E}_{\mathbf{U}}(\pi_2)$ with $\mathscr{E}_{\mathbf{U}}(\pi_1)$ does not modify $\mathscr{E}_{\mathbf{U}}(\pi_1)$

Note again that the relations $\perp\!\!\!\perp_{\mathbf{U}}^{\mathscr{L}}$ and $\perp\!\!\!\perp^{\mathscr{L}}$ are not commutative, and are not identical to the innovations $\perp\!\!\!\perp_{\mathbf{U}}$ and $\perp\!\!\!\perp$, respectively. Also, if we select the conditioning set $\mathbf{U}$ such that $\mathbf{Z}_1 \cap \mathbf{Z}_2 \subseteq \mathbf{U}$, then both $\pi_1 \perp\!\!\!\perp_{\mathbf{U}}^{\mathscr{L}} \pi_2$ and $\pi_2 \perp\!\!\!\perp_{\mathbf{U}}^{\mathscr{L}} \pi_2 \perp\!\!\!\perp_{\mathbf{U}}^{\mathscr{L}} \pi_1$ hold. In other words, as soon as all the unknowns coupling two systems are specified, these systems become innovations for each other. We now state without proof several of the properties of ML-innovations.

Lemma 4.2. *Given a system π and a subset $\mathbf{Z}'$ of its unknowns, we have*

$$\mathscr{E}_{\mathbf{Z}'}(\pi) \perp\!\!\!\perp^{\mathscr{L}} \bar{\mathscr{E}}_{\mathbf{Z}'}(\pi). \tag{4.17}$$

Also, if $\pi_2 \perp\!\!\!\perp_{\mathbf{U}}^{\mathscr{L}} \pi_1$, the following identities are satisfied:

$$\bar{\mathscr{E}}_{\mathbf{Z}_1 \cup \mathbf{U}}(\mathscr{E}_{\mathbf{U}}(\pi_2) \mid \mathscr{E}_{\mathbf{U}}(\pi_1)) = \mathscr{E}_{\mathbf{U}}(\pi_1), \tag{4.18}$$

$$\mathscr{E}_{\mathbf{U}}(\pi_1 \mid \pi_2) \equiv_{\mathscr{L}} \mathscr{E}_{\mathbf{U}}(\pi_1) \mid \mathscr{E}_{\mathbf{U}}(\pi_2), \tag{4.19}$$

$$\bar{\mathscr{E}}_{\mathbf{U}}(\pi_1 \mid \pi_2) \equiv_{\mathscr{L}} \bar{\mathscr{E}}_{\mathbf{U}}(\pi_1) \mid \bar{\mathscr{E}}_{\mathbf{U}}(\pi_2). \tag{4.20}$$

4.4. Incremental estimation

We consider a compound system of the form (3.22) and wish to estimate its variables incrementally.

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

$$\pi_1 \overset{\mathscr{L}}{—} \mathbf{Z} \overset{\mathscr{L}}{—} \pi_2 \quad \text{if} \begin{cases} \pi_2 \text{ is not an ML-innovation of } \pi_1, \text{ and} \\ \pi_1 \text{ is not an ML-innovation of } \pi_2. \end{cases} \tag{4.21a}$$

Similarly

$$\pi_1 \overset{\mathscr{L}}{\rightarrow} \mathbf{Z} \overset{\mathscr{L}}{\rightarrow} \pi_2 \quad \text{if} \begin{cases} \pi_2 \text{ is an ML-innovation of } \pi_1, \text{ but} \\ \pi_1 \text{ is not an ML-innovation of } \pi_2, \end{cases} \tag{4.21b}$$

and

$$\pi_1 \overset{\mathscr{L}}{\leftrightarrow} \mathbf{Z} \overset{\mathscr{L}}{\leftrightarrow} \pi_2 \quad \text{if} \begin{cases} \pi_2 \text{ is an ML-innovation of } \pi_1, \text{ and} \\ \pi_1 \text{ is an ML-innovation of } \pi_2. \end{cases} \tag{4.21c}$$

Now, consider each pair (π_i, π_j) of subsystems of the given compound system $\pi = \|_{i\in I}\,\pi_i$. When the two subsystems π_i and π_j share variables we say they are *neighbors* and draw a branch between them, which is oriented according to which of the three cases (4.21a)–(4.21c) applies. Collecting all branches generated in this manner yields a bipartite graph where unknowns and systems alternate, which we call the *estimation graph* of π, and denote by

$\mathrm{E{\scriptsize STIM}G{\scriptsize RAPH}}(\pi)$.

Graph compilation. The estimation graph plays a key role in the incremental estimation of compound systems, as indicated by the following results.

Lemma 4.3. *Consider a partition $I = J \cup J^c$ with $J \cap J^c = \emptyset$, and denote*

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

$\mathbf{Z}_J =$ *set of private unknowns of* π_J,

$\mathbf{Z}_J^c =$ *set of private unknowns of* π_J^c,

$\partial\mathbf{Z} =$ *set of shared unknowns of* π_J *and* π_J^c.

(1) *We have*

$$\bar{\mathscr{E}}_{\mathbf{Z}_J}(\pi) = \bar{\mathscr{E}}_{\mathbf{Z}_J}(\pi_J \,|\, \bar{\mathscr{E}}_{\partial\mathbf{Z}}(\pi_J^c)) \,|\, \mathrm{FLAT} \circ \bar{\mathscr{E}}(\pi_J^c). \tag{4.22}$$

(2) *Under the stronger assumption*

$$\pi_J \overset{\mathscr{L}}{\rightarrow} \partial\mathbf{Z} \overset{\mathscr{L}}{\rightarrow} \pi_J^c \quad \text{or} \quad \pi_J \overset{\mathscr{L}}{\leftrightarrow} \partial\mathbf{Z} \overset{\mathscr{L}}{\leftrightarrow} \pi_J^c, \tag{4.23}$$

identity (4.22) *reduces to*

$$\bar{\mathscr{E}}_{\mathbf{Z}_J}(\pi) = \bar{\mathscr{E}}_{\mathbf{Z}_J}(\pi_J) \,|\, \mathrm{FLAT} \circ \bar{\mathscr{E}}(\pi_J^c), \tag{4.24}$$

which indicates that, to estimate the variables $\mathbf{Z}_J$ of the compound system π, only the subsystem π_J is needed.[5]

Proof. We only derive part (1), since it requires introducing a small modification in the proof of Lemma 3.3, where the simulation form of the above result is presented. Employing the decomposition (4.11) of π_J^c with $\mathbf{Z}' = \partial\mathbf{Z}$, and using the fact that $\mathscr{E}_{\partial\mathbf{Z}}(\pi_J^c)$ is a $\mathbf{Z}_J$-ML-innovation for $\pi_J \,|\, \bar{\mathscr{E}}_{\partial\mathbf{Z}}(\pi_J^c)$, we get

$$\begin{aligned}\bar{\mathscr{E}}_{\mathbf{Z}_J}(\pi) &= \bar{\mathscr{E}}_{\mathbf{Z}_J}(\pi_J \,|\, \pi_J^c)\\ &= \bar{\mathscr{E}}_{\mathbf{Z}_J}(\pi_J \,|\, \bar{\mathscr{E}}_{\partial\mathbf{Z}}(\pi_J^c) \,|\, \mathscr{E}_{\partial\mathbf{Z}}(\pi_J^c))\\ &= \bar{\mathscr{E}}_{\mathbf{Z}_J}(\pi_J \,|\, \bar{\mathscr{E}}_{\partial\mathbf{Z}}(\pi_J^c)) \,|\, \bar{\mathscr{E}}_{\mathbf{Z}_J} \circ \mathscr{E}_{\partial\mathbf{Z}}(\pi_J^c),\end{aligned} \tag{4.25}$$

where the last equality is due to property (4.20) of innovations. Next, using successively properties (4.13) and (4.12) of primitives, and noting π_J^c has no unknowns in $\mathbf{Z}_J$, we find

$$\begin{aligned}\bar{\mathscr{E}}_{\mathbf{Z}_J} \circ \mathscr{E}_{\partial\mathbf{Z}}(\pi_J^c) &= \mathscr{E}_{\mathbf{Z}_J} \circ \bar{\mathscr{E}}_{\mathbf{Z}_J \cup \partial\mathbf{Z}} \circ \mathscr{E}_{\mathbf{Z}_J \cup \partial\mathbf{Z}}(\pi_J^c)\\ &= \bar{\mathscr{E}}_{\mathbf{Z}_J} \circ \text{FLAT} \circ \bar{\mathscr{E}}_{\partial\mathbf{Z}}(\pi_J^c) = \text{FLAT} \circ \bar{\mathscr{E}}(\pi_J^c),\end{aligned} \tag{4.26}$$

which is where a difference appears with respect to Lemma 3.3. Together with (4.25), this identity proves (4.22). The proof of part (2) is the same as for the simulation case. □

Theorem 4.2. *Consider a compound system π of the form* (3.22).

(1) *π admits an incremental estimation if and only if* EstimGraph(π) *is an acyclic directed graph.*

(2) *If the interaction graph obtained by removing all branch orientations from* EstimGraph(π) *forms a tree, π can be transformed into an equivalent system π' such that* ExecGraph(π) *is a directed tree, which can therefore be incrementally estimated.*

When determining whether the graph contains cycles, all bidirectional branches of the form (4.21c) can be used as "wild cards" whose orientation is selected so as to break potential cycles.

Proof of Theorem 4.2. Parts (1) and (2) are proved in the same manner as Theorems 3.3 and 3.4, respectively. We include a description of the procedure employed to convert a system represented by a tree into one which is represented by a directed tree, since this scheme will be implemented in Sig. Again, since the interaction graph of π is a tree, we select an arbitrary root i_0 on this tree, and use the distance between nodes

[5] The second factor $\text{FLAT} \circ \bar{\mathscr{E}}(\pi_J^c)$ is irrelevant for the estimation problem, since it only contains the parity checks of the observations.

and the root to specify a partial order on the index set I, where $i \prec j$ if i is closer to i_0 than j. The transformation relies again on the following rules:

Rule 1: Select $i \in I$, and let i_- be the unique neighbor 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

$$\pi_i \equiv \bar{\mathscr{E}}_{\mathbf{Z}_{i_-}}(\pi_i) \,|\, \mathscr{E}_{\mathbf{Z}_{i_-}}(\pi_i), \tag{4.27}$$

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

Rule 2: If the factorization (4.27) has been performed, reorganize the compound system π by rewriting

$$\pi_{i_-} \,|\, \pi_i \equiv \pi'_{i_-} \,|\, \pi'_i \tag{4.28a}$$

with

$$\pi'_{i_-} \triangleq \pi_{i_-} \,|\, \bar{\mathscr{E}}_{\mathbf{Z}_{i_-}}(\pi_i), \qquad \pi'_i \triangleq \mathscr{E}_{\mathbf{Z}_{i_-}}(\pi_i). \tag{4.28b}$$

The index set I can be ordered so that successive indices i are *nonincreasing* with respect to the partial order $\prec$. By successively applying Rules 1 and 2 to this sequence, we find that once the transformation (4.28a) and (4.28b) has been applied to node i, the new system π' includes the branch

$$\pi'_{i_-} \xrightarrow{\mathscr{L}} \mathbf{Z}_{i_-,i} \xrightarrow{\mathscr{L}} \pi'_i \tag{4.29}$$

in its execution graph, where $\mathbf{Z}_{i_-,i}$ represents the set of shared unknowns of π'_{i_-} and π'_i. Then when Rules 1 and 2 are subsequently applied to system π'_{i_-}, π'_{i_-} may change, but the orientation of the branch (4.29) 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 from its extremities towards its root i_0, so that the complexity of the compilation procedure is proportional to the cardinality of I. □

4.5. *The* SIG *estimation compiler*

We now implement in SIG the generalized and conditional likelihood primitives and use them to perform incremental simulation. If SYSTEM denotes a system and X, Y are two of its variables, the two operators

```
estimate X, Y in SYSTEM
knowing X, Y estimate SYSTEM
```

represent, respectively, the {X, Y}-generalized likelihood $\bar{\mathscr{E}}_{\mathrm{X,Y}}$(SYSTEM) and {X, Y}-conditional likelihood $\mathscr{E}_{\mathrm{X,Y}}$(SYSTEM). If X, Y are the only two unknowns of interest in the system, only the operator estimate X, Y in SYSTEM needs to be employed. On the other hand, if we seek to estimate other variables given X, Y and the observed data, we need also to make use of knowing X, Y estimate SYSTEM.

The application of these two operators can be illustrated by studying the HMM example. As a first step, consider the modified program

```
system HMM=(integer N)
    { variable X[i] i=0 to N, Y[i] i=1 to N
     observed Y[i] i=1 to N }          % declaration of observations
    (| 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 ,
```

where we have introduced the new information that the Y's *are observed, but not the other variables*. When estimation problems are considered, this information needs to be supplied as part of a system specification. The estimation graph of HMM is given by

$$\begin{tikzcd}
\pi_0 \arrow[r, dash, "\mathscr{L}"] & x_0, x_1 \arrow[r, dash, "\mathscr{L}"] \arrow[d, dash, "\mathscr{L}"] & \pi_1 \arrow[r, dash, "\mathscr{L}"] & x_1, x_2 \arrow[r, dash, "\mathscr{L}"] \arrow[d, dash, "\mathscr{L}"] & \pi_2 & & \pi_{N-1} \arrow[r, dash, "\mathscr{L}"] & x_{N-1}, x_N \arrow[r, dash, "\mathscr{L}"] \arrow[d, dash, "\mathscr{L}"] & \pi_N \\
& \sigma_1 & & \sigma_2 & & \cdots & & \sigma_N &
\end{tikzcd} \tag{4.30}$$

It is a nonoriented tree, which can be converted into a directed one by applying the two compilation rules introduced in the proof of Theorem 4.2. The algorithm proceeds in two stages: we first apply the rules to the vertical branches of the tree, which model the HMM observations, and then process from right to left the horizontal branches, which model the Markov chain transitions.

(1) Applying Rule 1, the potential V(X[i−1], X[i], Y[i]) can be decomposed as follows, where <=>L means $\equiv_{\mathscr{L}}$,

```
    potential V(X[i-1], X[i], Y[i])
<=>L
   (| estimate X[i-1], X[i] in potential V(X[i-1], X[i], Y[i])
    | knowing X[i-1], X[i] estimate potential V(X[i-1], X[i], Y[i])
    |)
```

For each index i, the subsystem

```
knowing X[i-1], X[i] estimate potential V(X[i-1], X[i], Y[i])
```

is an ML-innovation with respect to all other subsystems.

(2) Applying Rule 2, define

```
PI[i] ::= (| estimate 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]<=>L (| estimate X[i-1] in PI[i]
          | knowing X[i-1] estimate PI[i]
          |)
```

(b) Apply Rule 2 and redefine

```
PI[i] ::= (| knowing X[i-1] estimate PI[i]
          |)

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

The resulting system is equivalent to the original one, and its estimation graph

$$\begin{tikzcd}
\pi_0 \arrow[r, "\mathscr{L}"] & x_0, x_1 \arrow[r, "\mathscr{L}"] \arrow[d, "\mathscr{L}"] & \pi_1 \arrow[r, "\mathscr{L}"] & x_1, x_2 \arrow[r, "\mathscr{L}"] \arrow[d, "\mathscr{L}"] & \pi_2 & \cdots & \pi_{N-1} \arrow[r, "\mathscr{L}"] & x_{N-1}, x_N \arrow[r, "\mathscr{L}"] \arrow[d, "\mathscr{L}"] & \pi_N \\
 & \sigma_1 & & \sigma_2 & & & & \sigma_N &
\end{tikzcd} \tag{4.31}$$

is an oriented tree, so that it is amenable to recursive estimation.

The above ML estimation procedure for hidden Markov models corresponds in fact to a reverse form of the standard *Viterbi algorithm* [30], which in addition to the right to left compilation sweep performed above includes a backtracking phase, where the most likely hidden state trajectory is generated recursively, starting from the extimate of `X[0]`, and using the estimate of `X[i-1]` and the system `Pi[i]` to generate the estimate of `X[i]`. This algorithm is discussed in further detail in [23], where it is shown to admit the same high-level program as the Rauch–Tung–Striebel double-sweep smoother of linear Gaussian state-space models. Obviously, since the HMM model is fully reversible, the right to left compilation procedure employed here could be replaced by a left to right sweep, which would yield the standard version of the Viterbi algorithm. The advantage of our choice of compilation direction is that it highlights the close analogy existing between simulation and estimation.

5. Discussion and conclusions

We have introduced the CSS model and associated SIG minilanguage for describing stochastic/nonstochastic 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 constraints 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 nonstochastic and statistical, existing between the system variables. This compilation employs two operations. The marginal $\bar{\mathscr{S}}(\cdot)$ and conditional $\mathscr{S}(\cdot)$ extend to mixed 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 nonoriented 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 [21] 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 mixed systems. We only need to replace the $\bar{\mathscr{S}}(\cdot)$ and $\mathscr{S}(\cdot)$ operations by $\bar{\mathscr{E}}(\cdot)$ and $\mathscr{E}(\cdot)$, which simply amounts to replacing summations by maximizations while performing randoms compressions. Since the $\sum$ and max operations have similar properties (they are both commutative and associative), this makes it easy to convert simulation results to estimation, and vice versa.

Although CSS is obviously related to the theory of belief functions and belief networks developed in [7, 8, 32, 33], 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 mixed 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.

- It would be of interest to extend our results to the case of mixed system whose variables and/or randoms take values in continuous domains. The main difficulty in attempting such a generalization is that all operations we perform must be implementable in a finite number of steps. This means that an algebraic mechanism must be available for eliminating variables within relations, and all probability densities, including those generated by randoms compression, must be finitely parametrized. These two conditions are generally not satisfied, but they do hold for the case of linear relations and Gaussian distributions which is studied in detail in [23]. In fact, it turns out that for the linear-Gaussian case, the two primitives $\bar{\mathscr{S}}(\cdot)$

and $\bar{\mathscr{E}}(\cdot)$ are identical, as well as the two primitives $\mathscr{S}(\cdot)$ and $\mathscr{E}(\cdot)$. Furthermore, the primitives can be implemented efficiently with standard matrix analysis methods.

- A second issue concerns the simulation or estimation of compound systems whose interaction graph is not a tree. The incremental simulation and estimation techniques developed here do not apply to such graphs. Although we can always resort to aggregation to convert a graph into a tree, this may not be the best way to proceed, since aggregation can produce very coarse aggregates. An alternative approach would consist of formulating simulation and estimation in terms of fixed point equations, for which we could then use iterative stochastic relaxation methods of the type employed in statistical mechanics.
- A third issue involves the introduction of two features currently missing from CSS, namely the specification of timing information and the ability to define mixed systems over infinite time intervals. These two features are already present in the previously introduced SIGN*alea* language [3], which represents an extension of the SIGNAL synchronous real-time language [4, 5, 22]. The SIGN*alea* language generalizes stochastic Büchi automata, Petri nets, and our SIG minilanguage. But the mathematical foundations of SIGN*alea* in [3] are somewhat shaky and estimation is not included. Thus, generalizing CSS to SIGN*alea* is a high priority task, particularly since SIGN*alea* is currently under implementation.
- Finally, our results need to be tested on real applications. Two applications of SIGN*alea* are now under consideration. The first one involves the implementation for Electricté de France of the nonintrusive appliance load monitoring scheme proposed in [16], 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.

Appendix A. Proof of Theorem 3.2 under Assumption A.1

Several results of our paper are derived in two stages, first by proving them under the assumption shown below, which is later removed in Appendix D to obtain a general derivation.

Assumption A.1. The system π is such that, for each x, there is at most one w satisfying relation $\Omega(x; w)$.

Recall that two systems are equivalent if they have the same compressed form. Thus, we can assume without loss of generality that π is compressed, so that the factorization (3.10b) requires that π should be identical to the system obtained by compressing $\bar{\mathscr{S}}_{\mathbf{X}'}(\pi)$, combining it with $\mathscr{S}_{\mathbf{X}'}(\pi)$, and then compressing the combination.

(1) We first compute the compressed form $\pi' = \{\mathbf{X}', \Omega', \mathbf{W}', \mathbf{p}'\}$ of $\bar{\mathscr{S}}_{\mathbf{X}'}(\pi)$. We have

$$\Omega'(x'; w) = \exists x'' : \Omega((x', x''); w), \tag{A.1}$$

so that $w_1 \sim_{\pi'} w_2$ iff

$$\forall x', \quad \exists x_1'' : \Omega((x', x_1''); w_1) \Leftrightarrow \exists x_2'' : \Omega((x', x_2''); w_2). \tag{A.2}$$

Then, if w' denotes an equivalence class of the relation $\sim_{\pi'}$, the compressed form of $\mathbf{p}$ assigns to w' the probability

$$\mathbf{p}'(w') = \sum_{v \in w'} \mathbf{p}(v), \tag{A.3}$$

and the compressed relation Ω' is given by

$$\Omega(x', w') = \Omega'(x', w) \quad \text{for } w \in w'. \tag{A.4}$$

The summation appearing in (A.3) illustrates why the system $\bar{\mathscr{S}}_{\mathbf{X}'}(\pi)$ can be viewed as a "marginal" of π with respect to the variables $\mathbf{X}'$.

(2) Assume now that $\mathscr{S}_{\mathbf{X}'}(\pi)$ has the form (3.10a), where the distribution $\mathbf{p}''$ still needs to be selected. Let us compute $\tilde{\pi} \triangleq \bar{\mathscr{S}}_{\mathbf{X}'}(\pi) \,|\, \mathscr{S}_{\mathbf{X}'}(\pi)$. The resulting system $\tilde{\pi} = \{\mathbf{X}, \tilde{\Omega}, \tilde{\mathbf{W}}, \tilde{\mathbf{p}}\}$ satisfies

$$\tilde{\mathbf{W}} = \mathbf{W}' \times \mathbf{W}, \tag{A.5a}$$

$$\tilde{\mathbf{p}}(w', w) = \mathbf{p}'(w') \times \mathbf{p}''(w) \tag{A.5b}$$

and

$$\tilde{\Omega}((x', x''); (w', w)) = \Omega'(x'; w'(w)) \wedge \Omega((x', x''); w), \tag{A.5c}$$

where $w'(w)$ is the equivalence class containing w. This is where we have used the assumption that for each x, there exists at most one w satisfying relation $\Omega(x, w)$; this implies that w' must be the equivalence class containing w. Compressing (A.5c) yields again Ω: the random (w, w') with $w \in w'$ is redundant and can be compressed as w alone.

(3) To prove that the requirement (3.10b) uniquely specifies $\mathbf{p}''$, note that by equating $p(w)$ to expression (A.5b) for $\tilde{\mathbf{p}}(w, w'(w))$, we obtain

$$\mathbf{p}''(w) = \frac{\mathbf{p}(w)}{\sum_{v \in w'(w)} \mathbf{p}(v)}, \tag{A.6}$$

which uniquely defines $\mathbf{p}''$. This expression generalizes to mixed systems the factorization (3.2) of standard probability distributions.

This proves our claim under Assumption A.1.

Appendix B. Proof of formula (3.7) and Lemma 3.1 under Assumption A.1

Identity (3.7) is in fact trivially satisfied under Assumption A.1. To derive (3.13), observe from (A.6) that if the equivalence classes w' are singletons, then $\forall w, \mathbf{p}''(w) = 1$. Given a system $\pi = \{\mathbf{X}, \Omega, \mathbf{W}, \mathbf{p}\}$, this implies that the conditional with respect to all its variables satisfies $\mathscr{S}_{\mathbf{X}}(\pi) = \{\mathbf{X}, \Omega, \mathbf{W}, 1\} = \text{FLAT}(\pi)$. Thus $\mathscr{S}_{\mathbf{X}'} \circ \bar{\mathscr{S}}_{\mathbf{X}'}(\pi) = \text{FLAT} \circ \bar{\mathscr{S}}_{\mathbf{X}'}(\pi)$. Applying now the primitives in reverse order, consider first $\mathscr{S}_{\mathbf{X}'}(\pi) = \{\mathbf{X}, \Omega, \mathbf{W}, \mathbf{p}''\}$ where $\mathbf{p}''$ is given by (A.6). Taking the marginal of this system with respect to $\mathbf{X}'$ and compressing the result yields the system $\{\mathbf{X}', \Omega', \mathbf{W}, \mathbf{p}'\}$, where Ω' is obtained by eliminating from Ω the variables not in $\mathbf{X}'$, and $\mathbf{p}'$ is obtained by compressing $\mathbf{p}''$ accordingly. Using expression (A.6) for $\mathbf{p}''$, this gives

$$\mathbf{p}'(w') = \sum_{w \in w'} \mathbf{p}''(w) = 1, \tag{B.1}$$

which proves the second part of (3.13).

Property (3.14) is just a consequence of the fact that taking the marginal of a system with respect to a set of variables requires only to project Ω on the desired variables and the system randoms. To prove (3.15), we assume that π is compressed and $\mathbf{Y} \cup \mathbf{Z} \subseteq \mathbf{X}$. We only need to show that the probability distributions satisfy the chain rule (3.15). In doing so, we denote by $w'(w, \mathbf{Z})$ the equivalence class containing the random w when the equivalence relation (A.2) is specified with respect to the set $\mathbf{Z}$, i.e., we compute the marginal of π with respect to $\mathbf{Z}$. According to (A.6), the distribution of $\mathscr{S}_{\mathbf{Z}}(\pi)$ is given by

$$\mathbf{p}''_{\mathbf{Z}}(w) = \frac{\mathbf{p}(w)}{\sum_{v \in w'(w, \mathbf{Z})} \mathbf{p}(v)}, \tag{B.2}$$

But the equivalence class $w'(w, \mathbf{Z})$ is the union of equivalence classes $v'(v, \mathbf{Y} \cup \mathbf{Z})$ generated by applying (A.2) to the larger set of variables $\mathbf{Y} \cup \mathbf{Z}$. Hence, the distribution $\mathbf{p}_{\mathbf{Y} \cup \mathbf{Z} \circ \mathbf{Z}}(w)$ of $\mathscr{S}_{\mathbf{Y} \cup \mathbf{Z}} \circ \mathscr{S}_{\mathbf{Z}}(\pi)$ can be expressed as

$$\begin{aligned}
\mathbf{p}''_{\mathbf{Y} \cup \mathbf{Z} \circ \mathbf{Z}}(w) &= \frac{\mathbf{p}''_{\mathbf{Z}}(w)}{\sum_{v \in w'(w, \mathbf{Y} \cup \mathbf{Z})} \mathbf{p}''_{\mathbf{Z}}(v)} \\
&= \frac{\mathbf{p}(w)}{\sum_{v \in w'(w, \mathbf{Z})} \mathbf{p}(v)} \times \frac{1}{\sum_{v \in w'(w, \mathbf{Y} \cup \mathbf{Z})} \frac{\mathbf{p}(v)}{\sum_{u \in v'(v, \mathbf{Z})} \mathbf{p}(u)}} \\
&= \frac{\mathbf{p}(w)}{\sum_{v \in w'(w, \mathbf{Y} \cup \mathbf{Z})} \mathbf{p}(v)} = \mathbf{p}''_{\mathbf{Y} \cup \mathbf{Z}}(w),
\end{aligned} \tag{B.3}$$

which proves (3.15).

To derive (3.16), it is convenient *not* to compress systems when computing marginals or composing systems. The two systems appearing on both sides of (3.16) can be denoted, respectively, as

$$\bar{\mathscr{S}}_{\mathbf{X}_1}(\pi_1 \mid \pi_2) = \{\mathbf{X}', \Omega', \mathbf{W}', \mathbf{p}'\},$$

$$\pi_1 \mid \bar{\mathscr{S}}_{\mathbf{X}_1}(\pi_2) = \{\mathbf{X}'', \Omega'', \mathbf{W}'', \mathbf{p}''\}.$$

Clearly, we have $\mathbf{X}' = \mathbf{X}'' = \mathbf{X}_1$ and $\mathbf{W}' = \mathbf{W}'' = \mathbf{W}_1 \times \mathbf{W}_2$. Next, applying the composition rule (2.6d) and the definition (3.9) of a marginal and decomposing $x_1 = (\tilde{x}_1, x_c)$ and $x_2 = (x_c, \tilde{x}_2)$ into their shared component x_c and private ones, we get

$$\begin{aligned}\Omega'((\tilde{x}_1, x_c); (w_1, w_2)) &= \exists \tilde{x}_2 : \Omega_1((\tilde{x}_1, x_c); w_1) \wedge \Omega_2((x_c, \tilde{x}_2); w_2)\\ &= \Omega_1(x_1; w_1) \wedge \exists \tilde{x}_2 : \Omega_2((x_c, \tilde{x}_2); w_2)\\ &= \Omega''(x_1; (w_1, w_2)).\end{aligned} \tag{B.4}$$

Finally, the equality of the distributions $\mathbf{p}'$ and $\mathbf{p}''$ follows from the fact that they are both obtained by summing $\mathbf{p}_2$ over the private randoms of π_2, where the summation is performed either after or before composition with π_1, which does not change the resulting distribution.

Appendix C. Proof of Lemma 3.2 under Assumption A.1

To prove (3.18) we find, by combining property (3.13) of the primitives with the composition rule (3.7) for a system and its FLAT version, that

$$\bar{\mathscr{S}}_{\mathbf{X}'} \circ \mathscr{S}_{\mathbf{X}'}(\pi) \mid \bar{\mathscr{S}}_{\mathbf{X}'}(\pi) \equiv \bar{\mathscr{S}}_{\mathbf{X}'}(\pi), \tag{C.1}$$

which according to (3.17) shows that the conditional $\mathscr{S}_{\mathbf{X}'}(\pi)$ forms an innovation for the marginal $\bar{\mathscr{S}}_{\mathbf{X}'}(\pi)$.

Let now π_2 be a $\mathbf{Y}$-innovation of π_1, so that (3.17) holds. Substituting this expression inside property (3.16) of primitives yields

$$\begin{aligned}\bar{\mathscr{S}}_{\mathbf{X}_1 \cup \mathbf{Y}}(\mathscr{S}_{\mathbf{Y}}(\pi_2) \mid \bar{\mathscr{S}}_{\mathbf{Y}}(\pi_1)) &= \bar{\mathscr{S}}_{\mathbf{X}_1 \cup \mathbf{Y}} \circ \bar{\mathscr{S}}_{\mathbf{X}_1 \cup \mathbf{Y}} \circ \mathscr{S}_{\mathbf{Y}}(\pi_2) \mid \bar{\mathscr{S}}_{\mathbf{Y}}(\pi_1),\\ &= \bar{\mathscr{S}}_{\mathbf{Y}}(\pi_1),\end{aligned} \tag{C.2}$$

which proves (3.19).

To derive (3.20), we use the factorization (3.12) to write

$$\pi_1 \mid \pi_2 = \pi' \mid \mathscr{S}_{\mathbf{Y}}(\pi_2) \tag{C.3a}$$

with

$$\pi' \triangleq \bar{\mathscr{S}}_{\mathbf{Y}}(\pi_1) \mid \bar{\mathscr{S}}_{\mathbf{Y}}(\pi_2) \mid \mathscr{S}_{\mathbf{Y}}(\pi_1). \tag{C.3b}$$

Then, the fact that π_2 is a $\mathbf{Y}$-innovation of π_1 implies $\mathscr{S}_{\mathbf{Y}}(\pi_2)$ is an innovation of π'. To see this, note that

$$\begin{aligned}\pi' \,|\, \bar{\mathscr{S}}_{\mathbf{X}_1 \cup \mathbf{Y}} \circ \mathscr{S}_{\mathbf{Y}}(\pi_2) &= \bar{\mathscr{S}}_{\mathbf{Y}}(\pi_1) \,|\, \bar{\mathscr{S}}_{\mathbf{Y}}(\pi_2) \,|\, (\mathscr{S}_{\mathbf{Y}}(\pi_1) \,|\, \bar{\mathscr{S}}_{\mathbf{X}_1 \cup \mathbf{Y}} \circ \mathscr{S}_{\mathbf{Y}}(\pi_2)) \\ &= \bar{\mathscr{S}}_{\mathbf{Y}}(\pi_1) \,|\, \bar{\mathscr{S}}_{\mathbf{Y}}(\pi_2) \,|\, \mathscr{S}_{\mathbf{Y}}(\pi_1) \quad \text{(by (3.17))} \\ &= \pi'. \end{aligned} \tag{C.4}$$

Consequently, by applying (3.19), we find

$$\bar{\mathscr{S}}_{\mathbf{Y} \cup \mathbf{X}_1}((\bar{\mathscr{S}}_{\mathbf{Y}}(\pi_1) \,|\, \bar{\mathscr{S}}_{\mathbf{Y}}(\pi_2) \,|\, \mathscr{S}_{\mathbf{Y}}(\pi_1)) \,|\, \mathscr{S}_{\mathbf{Y}}(\pi_2)) = \bar{\mathscr{S}}_{\mathbf{Y}}(\pi_1) \,|\, \bar{\mathscr{S}}_{\mathbf{Y}}(\pi_2) \,|\, \mathscr{S}_{\mathbf{Y}}(\pi_1). \tag{C.5}$$

But $\mathscr{S}_{\mathbf{Y}}(\pi_1)$ is an innovation of $\bar{\mathscr{S}}_{\mathbf{Y}}(\pi_1) \,|\, \bar{\mathscr{S}}_{\mathbf{Y}}(\pi_2)$, so that applying again (3.19) gives

$$\bar{\mathscr{S}}_{\mathbf{Y}}((\bar{\mathscr{S}}_{\mathbf{Y}}(\pi_1) \,|\, \bar{\mathscr{S}}_{\mathbf{Y}}(\pi_2) \,|\, \mathscr{S}_{\mathbf{Y}}(\pi_1)) = \bar{\mathscr{S}}_{\mathbf{Y}}(\pi_1) \,|\, \bar{\mathscr{S}}_{\mathbf{Y}}(\pi_2), \tag{C.6}$$

which, together with (C.5), proves (3.21). Identity (3.20) for the conditional follows from the one we have just derived for the marginal, since the constraints are not affected by the conditional, and the normalization factors of the densities are obtained directly from those of the marginals. This proves the result.

Appendix D. Proof of Theorem 3.2, formula (3.7), and Lemmas 3.1 and 3.2 in the general case

Consider now an arbitrary system $\pi = \{\mathbf{X}, \Omega, \mathbf{W}, \mathbf{p}\}$. We can associate to π an augmented system

$$\bar{\pi} = \{\mathbf{X} \cup \bar{\mathbf{W}}, \Omega \wedge \tilde{\Omega}, \mathbf{W}, \mathbf{p}\}, \tag{D.1a}$$

with

$$\bar{\mathbf{W}} = \{\bar{W}_1, \ldots, \bar{W}_q\}, \tag{D.1b}$$

$$\tilde{\Omega} : \bar{W}_j = W_j \quad \text{for } 1 \leqslant j \leqslant q, \tag{D.1c}$$

which satisfies Assumption A.1. The system $\bar{\pi}$ is obtained from π just by making the randoms W_j directly visible from the outside through the introduction of the additional variables $\bar{W}_j$. Note that we have

$$\pi = \bar{\mathscr{S}}_{\mathbf{X}}(\bar{\pi}). \tag{D.2}$$

Proof of Theorem 3.2. We can apply Theorem 3.2 to $\bar{\pi}$. This yields

$$\bar{\pi} \equiv \bar{\mathscr{S}}_{\mathbf{X}'}(\bar{\pi}) \,|\, \mathscr{S}_{\mathbf{X}'}(\bar{\pi}) = \bar{\mathscr{S}}_{\mathbf{X}'}(\pi) \,|\, \mathscr{S}_{\mathbf{X}'}(\bar{\pi}), \tag{D.3}$$

since $\mathbf{X}' \subseteq \mathbf{X}$ and the formulas involving the marginal $\bar{\mathscr{S}}(\cdot)$ alone are trivially true without Assumption A.1. Next, since $\bar{\pi}$ satisfies Assumption A.1, we can apply (3.18) and (3.21). Together with (D.2), this gives

$$\pi \equiv \bar{\mathscr{S}}_{\mathbf{X}'}(\pi) \,|\, \bar{\mathscr{S}}_{\mathbf{X}} \circ \mathscr{S}_{\mathbf{X}'}(\bar{\pi}), \tag{D.4}$$

which has the form (3.10b) provided we identify

$$\mathscr{S}_{\mathbf{X}'}(\pi) \triangleq \bar{\mathscr{S}}_{\mathbf{X}} \circ \mathscr{S}_{\mathbf{X}'}(\bar{\pi}). \tag{D.5}$$

According to the construction of Appendix A, the system $\mathscr{S}_{\mathbf{X}'}(\bar{\pi})$ has the structure

$$\mathscr{S}_{\mathbf{X}'}(\bar{\pi}) = \{\mathbf{X} \cup \bar{W}, \Omega \wedge \tilde{\Omega}, \mathbf{W}, \mathbf{p}''\}, \tag{D.6}$$

with

$$\mathbf{p}''(w) = \frac{\mathbf{p}(w)}{\sum_{v \in \bar{w}(w)} \mathbf{p}(v)}, \tag{D.7}$$

where $\bar{w}(w)$ denotes the equivalence class containing w for the equivalence relation associated to the projection of $\Omega \wedge \tilde{\Omega}$ onto $\mathbf{X}'$. However, because this projection eliminates entirely the variables $\bar{\mathbf{W}}$, it coincides with the projection of Ω on $\mathbf{X}'$, so that $\bar{w}(w) = w'(w)$, where $w'(w)$ is the equivalence class containing w for the equivalence relation (A.1). Then, when we perform the marginal of $\mathscr{S}_{\mathbf{X}'}(\bar{\pi})$ with respect to $\mathbf{X}$, the variable set $\mathbf{X} \cup \bar{W}$ is projected onto $\mathbf{X}$, and the relation $\Omega \wedge \tilde{\Omega}$ reduces to Ω alone, so that the resulting system $\mathscr{S}_{\mathbf{X}'}(\pi)$ has the structure (3.10a), where $\mathbf{p}''(w)$ obeys (A.6). Thus, the procedure employed to generate the conditional $\mathscr{S}_{\mathbf{X}'}(\pi)$ does not depend on whether π satisfies Assumption A.1 or not.

Proof of formula (3.7). We start from the identity

$$\bar{\pi} \equiv \bar{\pi} \,|\, \text{FLAT}(\bar{\pi}) \tag{D.8}$$

for the system $\bar{\pi}$. Since this formula implies that $\text{FLAT}(\bar{\pi})$ is an innovation of $\bar{\pi}$, we can apply (3.21), which yields

$$\pi \equiv \pi \,|\, \bar{\mathscr{S}}_{\mathbf{X}} \circ \text{FLAT}(\bar{\pi}) = \pi \,|\, \text{FLAT}(\pi), \tag{D.9}$$

where the equality $\bar{\mathscr{S}}_{\mathbf{X}} \circ \text{FLAT}(\bar{\pi}) = \text{FLAT}(\pi)$ is due to the fact that the projection of $\Omega \wedge \tilde{\Omega}$ onto $\mathbf{X}$ coincides with that of Ω alone.

Proof of Lemma 3.1. The proof of formulas (3.14) and (3.16) given in Appendix B does not use Assumption A.1. Thus we only have to extend (3.13) and (3.15). First, we apply (3.13) to $\bar{\pi}$, which gives

$$\mathscr{S}_{\mathbf{X}'} \circ \bar{\mathscr{S}}_{\mathbf{X}'}(\bar{\pi}) = \bar{\mathscr{S}}_{\mathbf{X}'} \circ \mathscr{S}_{\mathbf{X}'}(\bar{\pi}) = \text{FLAT} \circ \bar{\mathscr{S}}_{\mathbf{X}'}(\bar{\pi}). \tag{D.10}$$

But, $\bar{\mathscr{S}}_{\mathbf{X}'}(\bar{\pi}) = \bar{\mathscr{S}}_{\mathbf{X}'}(\pi)$. Then, using (D.5), we have

$$\bar{\mathscr{S}}_{\mathbf{X}'} \circ \mathscr{S}_{\mathbf{X}'}(\bar{\pi}) = \bar{\mathscr{S}}_{\mathbf{X}'} \circ \bar{\mathscr{S}}_{\mathbf{X}} \circ \mathscr{S}_{\mathbf{X}'}(\bar{\pi}) = \bar{\mathscr{S}}_{\mathbf{X}'} \circ \mathscr{S}_{\mathbf{X}'}(\pi), \tag{D.11}$$

which, together with (D.10), proves (3.13). To prove (3.15), we can assume without loss of generality that $\bar{\mathbf{W}} \cap (\mathbf{Y} \cup \mathbf{Z}) = \emptyset$. Then, as before, we note that (3.15) holds for $\bar{\pi}$, so that

$$\mathscr{S}_{\mathbf{Y} \cup \mathbf{Z}} \circ \mathscr{S}_{\mathbf{Z}}(\bar{\pi}) = \mathscr{S}_{\mathbf{Y} \cup \mathbf{Z}}(\bar{\pi}). \tag{D.12}$$

Since $\bar{\mathbf{W}} \cap (\mathbf{Y} \cup \mathbf{Z}) = \emptyset$, we have $\overline{\mathscr{S}_{\mathbf{Z}}(\pi)} = \mathscr{S}_{\mathbf{Z}}(\bar{\pi})$. Using this remark, applying $\mathscr{S}_{\mathbf{X}}(\,)$ to both sides of (D.12), and using (D.5), gives (3.15) for an arbitrary π.

Proof of Lemma 3.2. The proof given in Appendix C does not use Assumption A.1.

Appendix E. Examples of compound system compilation

To illustrate the rules developed in Theorem 3.4 for transforming a compound system into an executable one, we consider two examples. The first example is depicted in Fig. 6. This figure should be read like ordinary text, from left to right, and from top to bottom. Each subfigure will be designated by its row and column indices, so that (3, 2) refers to the subfigure appearing in the third row and second column. The subfigure (1, 1) represents the execution graph of a compound system $\pi = |_{i \in I}\, \pi_i$. Each vertex of the graph corresponds to a subsystem π_i. Since the graph is a tree, the compilation procedure of Theorem 3.4 is applicable. The first step consists in selecting a root node, which is shown in subfigure (1, 2), where the precedence relationship existing between parent nodes and their children is used to orient the tree. The root node is the one from which all arrows originate. All extremities of the tree are represented by black patches in subfigure (2, 1). The remaining subfigures illustrate the application of Rules 1 and 2 of Theorem 3.4 to bring π to executable form. As we go from subfigure (2, 1) to (2, 2), the following changes occur:

- Three patches move from grey to black. Let π_j be the subsystem corresponding to such a patch, and let π_{i_1} and π_{i_2} be the subsystems represented by the two patches adjacent to j which switch from back to white.
- When a patch $i = i_1, i_2$ goes from black to white, the following operations are performed.
  (1) We factor $\pi_i = \bar{\mathscr{S}}_{\mathbf{X}_{i,j}}(\pi_i) \,|\, \mathscr{S}_{\mathbf{X}_{i,j}}(\pi_i)$, where π_j is the unique neighbor of π_i on the tree, and $\mathbf{X}_{i,j}$ denotes the shared variables of π_j and π_i.
  (2) The local subsystem at i is replaced by $\mathscr{S}_{\mathbf{X}_{i,j}}(\pi_i)$.
- As the patch j goes from grey to black, its local subsystem is replaced by

$$\bar{\mathscr{S}}_{\mathbf{X}_{i_1,j}}(\pi_{i_1}) \,|\, \bar{\mathscr{S}}_{\mathbf{X}_{i_2,j}}(\pi_{i_2}) \,|\, \pi_j.$$

The subfigures (2, 1) to (4, 2) describe the successive application of Rules 1, 2 to the tree. The final subfigure (4, 2) represents a compound system $\pi' = |_{i \in I}\, \pi'_i$ which is equivalent to the original system, but such that ExecGraph(π') is a directed tree, and thus executable.

The above compilation procedure can be interpreted as follows. Suppose each patch represents a person collecting information in a hierarchical organization. After gathering the desired information, each individual writes a detailed report for personal use, and forwards a synthetic memo to his/her superior. The superior merges his/her own information with the synthetic memos of his/her subordinates, but in

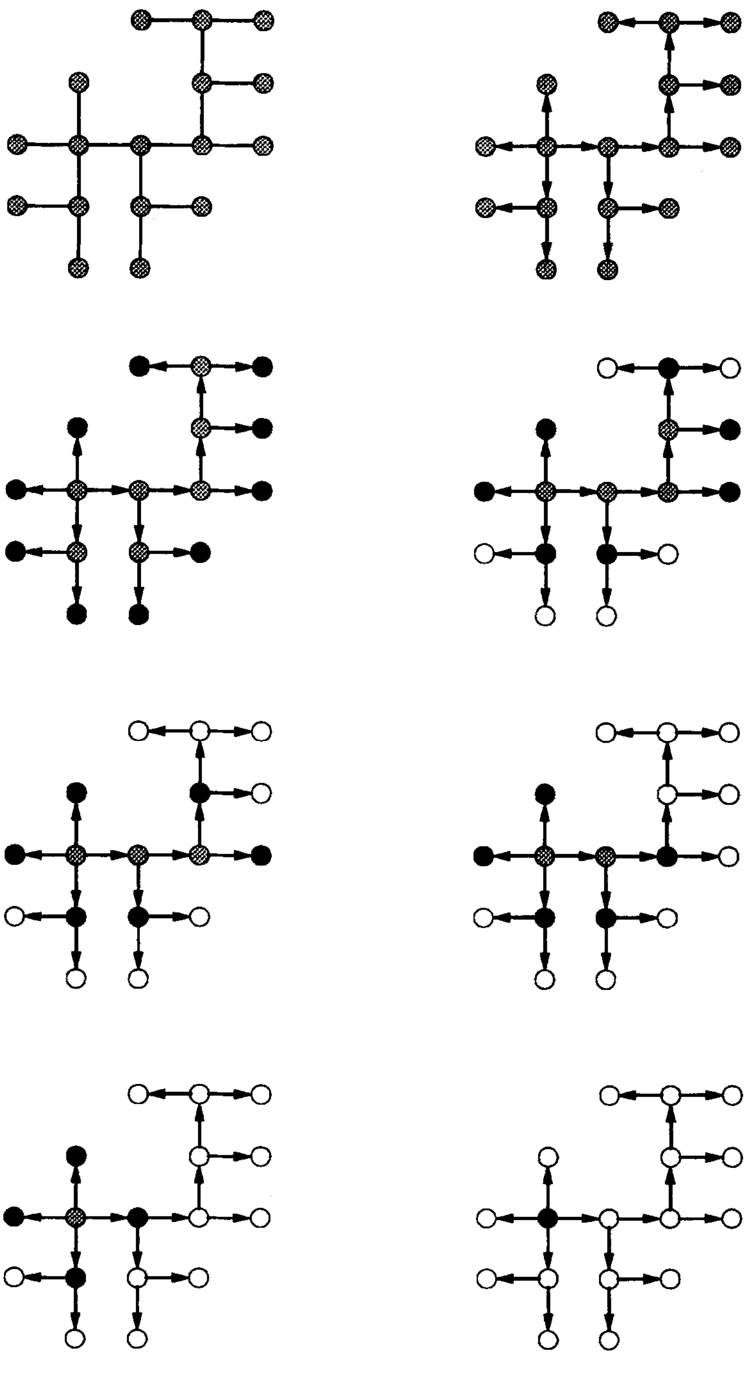

Fig. 6. Application of the compilation rules of Theorem 3.4 to a tree.

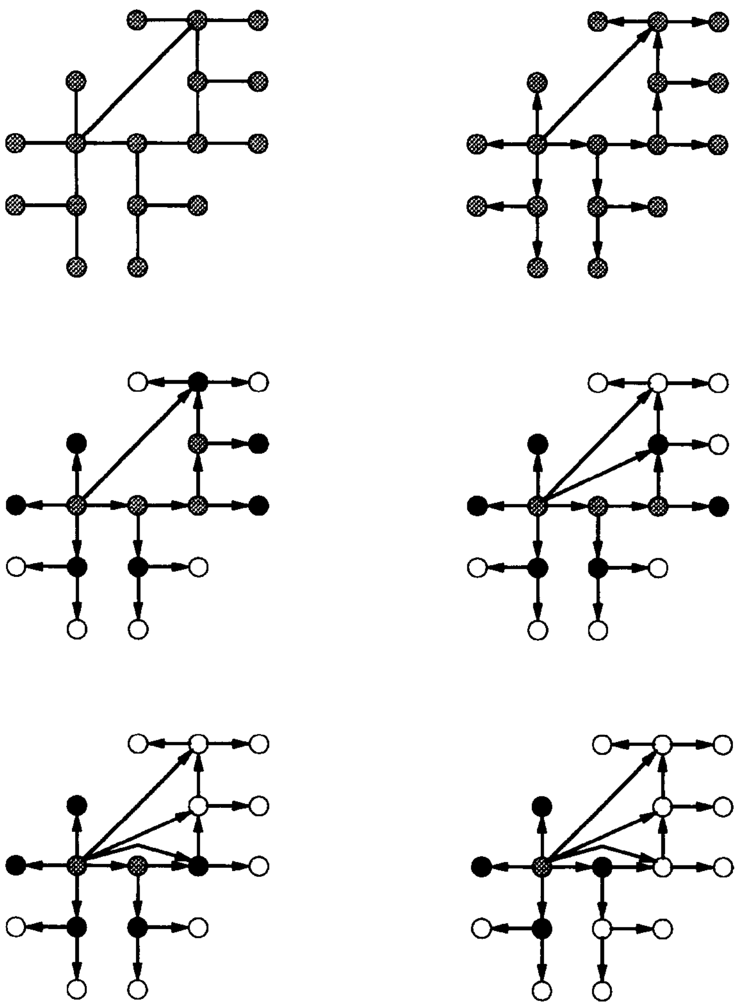
Fig. 7. Compilation procedure for a graph containing a cycle.

order to do so, must wait until all subordinates have turned in their report. Note that this information gathering scheme relies strongly on the fact that each person has a single superior.

Figure 7 shows the result of applying the same procedure to a nonhierarchial organization: the outcome is that hierarchy is re-created! The execution graph shown in subfigure (1, 1) does not form a tree since it contains a cycle. The successive subfigures illustrate what happens when Rules 1, 2 are nevertheless applied to this case. The partial order that we choose for the graph nodes is depicted in subfigure (1, 2). Consider now the transformation occurring as we go from subfigure (2, 1) to (2, 2). A single patch with index j switches from grey to black, and two neighboring patches switch from black to white. Among these two patches, we focus our attention on the one which is not a minimal vertex of the directed graph, whose index is i. This vertex is connected to two lower vertices in the sense of the partial order on the graph, namely j and the root node 0. As i switches from black to white, the following operations are performed.

(1) We factor $\pi_i = \bar{\mathscr{S}}_{\mathbf{X}_{i,(j,0)}}(\pi_i) \mid \mathscr{S}_{\mathbf{X}_{i,(j,0)}}(\pi_i)$, where $\mathbf{X}_{i,(j,0)}$ represents the union of variables shared by π_j and π_i on the one hand, and π_0 and π_i on the other hand.

(2) The local subsystem at i is replaced by $\mathscr{S}_{\mathbf{X}_{i,(j,0)}}(\pi_i)$.

(3) The new system $\pi'_j = \bar{\mathscr{S}}_{\mathbf{X}_{i,(j,0)}}(\pi_i) \mid \mathscr{S}_{\mathbf{X}_{i,(j,0)}}(\pi_i) \mid \pi_j$ is assigned to the patch j, where $\mathscr{S}_{\mathbf{X}_{i,(j,0)}}(\pi_i)$ represents the contribution of the other patch switching from black to white, so that a new branch linking 0 and j needs to be added to the execution graph.

Pursuing the compilation procedure yields the graphs of subfigures (3, 1) and (3, 2), where the procedure terminates. Note that in subfigure (3, 2), the cycle of original execution graph has been triangulated through the addition of new branches. Thus, our compilation procedure automatically triangulates the underlying execution graph and implicitly replaces it by the tree formed by its maximal cliques, as recommended in [21].

In our interpretation, when a subordinate has several direct superiors, they must all agree on the subordinate's report as they compile their own information. The outcome of the compilation procedure is therefore that hierarchy is recreated through the aggregation of all direct superiors into a single virtual one!

Acknowledgements

The authors are indebted to anonymous reviewers for their helpful comments and suggestions on an earlier version of this paper.

References

[1] R. Alur, C. Courcoubetis and D. Dill, Model checking for probabilistic real-time systems, in: *Proc. 18th Internat. Coll. on Automata Languages and Programming* (ICALP) (1991).

[2] M. Basseville and I.V. Nikiforov, *Detection of Abrupt Changes: Theory and Applications* (Prentice-Hall, Englewood Cliffs, NJ, 1993).

[3] A. Benveniste, Constructive probability and the SIGNalea language: Building and handling random processes with programming, Tech. Report 1532, Institut National de Recherche en Informatique et Automatique, Rocquencourt, France, 1991.

[4] A. Benveniste, M. Le Borgne and P. Le Guernic, Hybrid systems the SIGNAL approach, Lecture Notes in Computer Science, Vol. 736 (Springer, Berlin, 1993) 230–254.

[5] A. Benveniste and P. Le Guernic, Hybrid dynamical systems theory and the SIGNAL language, *IEEE Trans. Automat. Control* **35** (1990) 535–546.

[6] C. Dellacherie and P. Meyer, *Probabilités et Potentiels* (Hermann, Paris, 1976).

[7] A.P. Dempster, Upper and lower probabilities induced by a multivalued mapping, *Ann. Math. Statist.* **38** (1967) 325–339.

[8] A.P. Dempster, A generalization of Bayesian inference (with discussion), *J. Royal Statist. Soc. Ser. B* **30** (1968) 205–247.

[9] A.P. Dempster, Construction and local computation aspects of network belief functions, in: R.M. Oliver and J.Q. Smith, eds., *Influence Diagrams, Belief Nets, and Decision Analysis* (Wiley, Chichester, 1990) Ch. 6, 121–141.

[10] R.C. Dubes and A.K. Jain, Random field models in image analysis, *J. Appl. Statist.* **12** (1989) 131–164.

[11] G.D. Forney, The Viterbi algorithm, *Proc. IEEE* **61** (1973) 268–278.

[12] S. Geman and D. Geman, Stochastic relaxation, Gibbs distribution, and the Bayesian restoration of images, *IEEE Trans. Pattern Anal. Machine Intell.* **6** (1984) 721–741.

[13] A. Giacalone, C. Jou and S. Smolka, Algebraic reasoning for probabilistic concurrent systems, in: *Proc. IFIP TC2 Working Conf. on Programming Concepts and Methods* (1989).

[14] H. Hansson and B. Jonsson, A calculus for communicating systems with time and probabilities, in: *Proc. 11th IEEE Real-Time Systems Symp.*, Los Alamitos (1990) 278–287.

[15] F. Harary, Graph Theory (Addison-Wesley, Reading, MA, 1969).

[16] G.W. Hart, Nonintrusive appliance and load monitoring, *Proc. IEEE* **80** (1992) 1870–1891.

[17] S. Hart and M. Sharir, Probabilistic propositional temporal logic, *Inform. and Control* **70** (1986) 97–155.

[18] B. Jonsson, C. Ho-Stuart and Y. Wang, Testing and refinement for nondeterministic and probabilistic processes, Lecture Notes in Computer Science, Vol. 863 (Springer Verlag, Berlin, 1994, pp. 418–430.

[19] B. Jonsson and K. Larsen, Specification and refinement of probabilistic processes, in: *Proc. 6th IEEE Internat. Symp. on Logic in Computer Science*, Amsterdam (1991) 266–277.

[20] R. Kindermann and J.L. Snell, *Markov Random Fields and their Applications* (American Mathematical Society, Providence, RI, 1980).

[21] S.L. Lauritzen and D.J. Spiegelhalter, Local computations with probabilities on graphical structures and their application to expert systems (with discussion), *J. Royal Statist. Soc. Ser. B* **50** (1988) 157–224.

[22] P. Le Guernic, T. Gauthier, M. Le Borgne and C. Le Maire, Programming real-time applications with SIGNAL, *Proc. IEEE* **79** (1991) 1321–1336.

[23] B.C. Levy, A. Benveniste and R. Nikoukhah, High-level primitives for recursive maximum likelihood estimation, Tech. Report 767, IRISA, Rennes, France, 1993.

[24] M. Molloy, Performance analysis using stochastic Petri nets, *IEEE Trans. Comput.* **31** (1982) 913–917.

[25] J. Pearl, Fusion, propagation, and structuring in belief networks, *Artificial Intelligence* **29** (1986) 241–288.

[26] M.A. Peot and R.D. Shachter, Fusion and propagation with multiple observations in belief networks, *Artificial Intelligence* **48** (1991) 299–318.

[27] B. Plateau and K. Atif, Stochastic automata network for modeling parallel systems, *IEEE Trans. Software Eng.* **17** (1991) 1093–1108.

[28] B. Plateau and J.-M. Fourneau, A methodology for solving Markov models of parallel systems, *J. Parallel Distrib. Comput.* **12** (1991) 370–387.

[29] B. Prum and J. Fort, *Stochastic Processes on a Lattice and Gibbs Measure* (Kluwer Academic, Boston, MA, 1991).

[30] L.R. Rabiner and B.H. Juang, An introduction to hidden Markov models, *IEEE ASSP Magazine* **3** (1986) 4–16.

[31] C. Robert, *Modèles Statistiques pour l'Intelligence Artificielle* (Masson, Paris, 1991).

[32] G. Shafer, *A Mathematical Theory of Evidence* (Princeton Univ. Press, Princeton, NJ, 1976).

[33] P.P. Shenoi and G. Shafer, Axioms for probability and belief function propagation, in: R.D. Shachter, T.S. Levitt, L.N. Kanal and J.F. Lemmer, eds., *Uncertainty in Artificial Intelligence*, Vol. 4 (North-Holland, Amsterdam, 1990) 169–198.

[34] T. Soderstrom and P. Stoica, *System Identification* (Prentice-Hall, Englewood Cliffs, NJ, 1989).

[35] R. van Glabbeek, S.A. Smolka, B. Steffen and C. Toffs, Reactive, generative, and stratified models of probabilistic processes, in: *Proc. 5th IEEE Internat. Symp. on Logic in Computer Science*, Philadelphia, PA (1990) 130–141.

[36] N. Viswanadham and Y. Narahari, *Performance Modeling of Automated Manufacturing Systems* (Prentice-Hall, Englewood Cliffs, NJ, 1992).

[37] J. Whittaker, *Graphical Models in Applied Multivariate Statistics* (Wiley, New York, 1990).