A Bayesian Network Framework for Stochastic Discrete-Event Control

Gregory Provan

***Abstract*— This article focuses on the use of Bayesian networks for stochastic Discrete-Event control applications. Bayesian networks offer several advantages for such applications, including a well-developed suite of efficient inference algorithms, model generality and compactness, and ease of model construction and/or model-learning. We show how we can formalise the control-theoretic semantics of a stochastic discrete-event control representation using a Bayesian network. We prove the space-efficiency of a Bayesian network relative to a probabilistic finite state machine. We demonstrate our approach on a simple elevator system.**

I. INTRODUCTION

This article addresses stochastic discrete control applications in which noise and other forms of uncertainty are associated with the evolution of a discrete-event system. Many formalisations of such models exist in the literature, such as [4], [13].

Bayesian networks [12] provide a method for modeling a stochastic system (defined over variables $\mathcal{V}$) using a factored probability distribution $P(\mathcal{V})$, and for efficiently computing arbitrary posterior distributions over $P(\mathcal{V})$. Bayesian networks have been used for building models for higher-level decision-making (e.g., [11]), and also for a wide variety of lower-level control applications, e.g., [16]. The Bayesian network (BN) framework is attractive for a number of reasons, including its ability to model both high-level decision-making and low-level control in a semantically clear manner, its large literature on efficient inference algorithms, and the compactness of the generated models (relative to many other modeling formalisms), and its ability to model system topology and take computational advantage of such topology.[1] The BN computational framework can provide stochastic control applications with more efficient algorithms than are currently being used, similar to that shown in the use of hidden Markov models and Kalman filters for speech recognition [19], assuming that we can show equivalence of the BN and control representations.

However, the Bayesian network literature contains little work addressing the control-theoretic issues when there is a control process underlying the system dynamics. For example, standard control-theoretic issues like controllability, liveness, etc. have not been studied with systems modeled as Bayesian networks. For real-world applications in which safety, liveness, and controllability are critical, it is important to be able to formalise and verify such properties for Bayesian network control models. This article provides the framework for addressing such formal properties.

Gregory Provan is with the Computer Science Department, University College Cork, Cork, Ireland g.provan@cs.ucc.ie

[1] A finite state machine cannot model system topology.

This article describes the properties necessary for a Bayesian network, or its decision-theoretic extension, the Influence diagram (ID) [7], to correctly model a stochastic control system. This is important from two perspectives. First, Bayesian networks have already been used to model several control applications, in some cases in inconsistent fashions. Second, increasingly many applications must integrate control with higher-level decision-making (e.g., air traffic control, integrated battle planning), and it is preferable to have a single representation for such applications rather than to use an ID for the higher-level decision-making and a control model, e.g., Petri net, for the control.

We formalise a Bayesian network for a discrete-event system [14] as follows. We start with a typical BN model $\mathcal{B}$, which describes the set of probability distributions over a set $\mathcal{V}$ of variables. We describe a transformation of a Bayesian network $\mathcal{B}$ into an equivalent network $\mathcal{B}_e$ in which states and events are made explicit. We can then use our representation $\mathcal{B}_e$ to describe the control-theoretic properties, given its equivalence to standard control representations. We adopt the probabilistic finite state machine (PFSM) [15] as our standard representation, although other probabilistic representations, such as probabilistic Petri nets ([9]), could also be used.

This article makes the following contributions. First, we outline a semantically clear representation of control models that capture event and action sequencing using a Bayesian network (BN) framework, which we call a BN control model $\mathcal{B}_e$. Second, we show that a BN control model $\mathcal{B}_e$ is equivalent to a stochastic finite state machine model. Third, we describe an algorithm to map a BN model into a BN control model, thereby verifying the control properties of the BN. Fourth, we show the improvements that can be obtained by using BN control models, from the perspectives of more efficient algorithms, and of representational power and efficiency.

We organize the remainder of the document as follows. Section II briefly summarizes the Bayesian network and finite state machine approach upon which our general modeling framework is defined, and introduces an illustrative example. Section III defines the control properties necessary for a Bayesian network in terms of order relations on the BN. Section IV guides the reader through our proposed event-based Bayesian network framework, as well as an algorithm form mapping this framework into a control framework. Section V show the improvements that can be obtained by using BN control models, from several perspectives. Finally, we conclude in Section VI with discussions on the implications of our framework and possible future studies.

II. Illustrative Example

We first introduce a model of an elevator system that is used for illustrative purposes. We then introduce the notation for our Bayesian Network (BN) and Finite State Machine (FSM) frameworks, and show how we build a model for the elevator system within each framework.

A. Elevator System

Figure 1(a) depicts a simple elevator system, which transports passengers from the ground floor to the upper floor. From a physical perspective, the system consists of an elevator, a cable, two sensors, a motor, and an actuator/controller. There are three marked positions on the elevator system, x, y, and z. The sensor turns on whenever the elevator is at its corresponding position. The motor can move the elevator cable up (from x to z), down (from z to x), or not move it at all, according to the actuator/controller command it receives. The controller is programmed to move the elevator cable up until the elevator has arrived at position z (*i.e.*, the sensor-z is "on"), when it automatically returns to position x, at which time (*i.e.*, the elevator is sensed to be present at position x and the sensor-x is "on") the system resets.

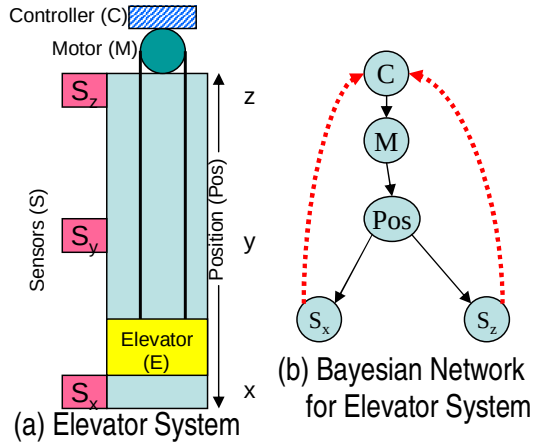

Fig. 1. (a) shows a simple elevator system. (b) shows the Bayesian Network for the elevator system, where the arcs for the physical causal links are solid, and the arcs for the control causal links are dashed.

B. Bayesian Network Model for Elevator

1) Bayesian Networks: Our underlying model representation is a directed acyclic graph (DAG) $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of vertices $\mathcal{V}$ and edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. We denote a directed edge from V_i to V_j by (V_i, V_j). The edges show dependence relations, such that the absence of an edge from V_i to V_j denotes that V_j is independent of V_i. Variable $V_i \in \mathcal{V}$ has a set of values, its domain, denoted by D_i.

Definition 1 (Bayesian Network): $\mathcal{B}$ is a tuple $(\mathcal{G}, \mathcal{P})$, where $\mathcal{G}$ is a DAG, and $\mathcal{P}$ a set of *probability distributions* constructed from vertices in $\mathcal{G}$ using the topological structure of $\mathcal{G}$. $\mathcal{P}$ satisfies $Pr\{\mathcal{V}\} = \prod_{i=1}^{n} Pr\{V_i | pa(V_i)\}$, where $pa(V_i)$ are the parents of V_i in $\mathcal{G}$.

It is well known that the BN is a superset of many stochastic models, such as a Markov model and a hidden Markov model (HMM).[2] In addition, restrictions of several BN inference algorithms are equivalent to well-known model-specific algorithms. For example, a hidden Markov model (HMM) focuses on representing the string configuration of maximal probability, which is referred to as the most probable explanation; the Viterbi algorithm [17] for HMM inference is a special case of the Max-prob inference algorithm [3] for Bayesian networks.

2) BN Elevator Model: To model this simple system directly as a Bayesian network, we first construct a graphical model as shown in Figure 1(b). This model shows a directed acyclic graph with nodes for the controller (actuator) (C), motor (M), position (Pos), and sensors (S_X, S_Z). Directed arcs show the causal influences, e.g., the motor's state (*on* or *off*) is causally influenced by the actuator's state (*on* or *off*), etc. We show feedback control using dashed arcs from each sensor to the actuator.[3] We can compose the model for the complete elevator system from models of the components of the elevator, such as sensors, motor, controller, etc.[4]

It is important to note that this model:

- Explicitly represents variables and not states. Each node in the network represent a variable, and each arc (V_1, V_2) represents a causal relationship of V_1 on V_2. States of the system (e.g., elevator at ground floor) are thus represented implicitly.
- Has no explicit representation of state transitions.
- Explicitly shows physical structural relationships. For example, it shows the physical structural relationship between motor and actuator through their causal relationship, i.e., the motor's state (on or off) is causally influenced by the actuator's state (on or off), but is not directly causally related to the sensor outputs.

C. Finite State Machine Model for Elevator

1) Finite State Machine Representation:

Definition 2 (FSM): An FSM is defined as $\mathcal{F} = (\mathcal{X}, \Sigma, \delta, x_0)$, where $\mathcal{X}$ is the state space, Σ is the set of events, δ is the partial transition function $\delta : \mathcal{X} \times \Sigma \rightarrow \mathcal{X}$ (and defines the transitions between states in $\mathcal{X}$), and x_0 is the initial state of the system. More precisely, equation $\delta(X_1, \sigma) = X_2$ means that there is a transition of event σ from state X_1 to state X_2 in $\mathcal{F}$.

A probabilistic FSM [15] augments an FSM with a set of probability distributions for all the transitions of the FSM.

Definition 3 (Probabilistic FSM): A probabilistic FSM is an FSM $\mathcal{F}_\Pi = (\mathcal{X}, \Sigma, \delta, \Pi, x_0)$, where $\Pi : \delta \rightarrow [0, 1]$ denotes the probabilities assigned to δ such that, for every state $x \in X$ and every event label $\sigma \in \Sigma$, one of the following holds: either $\delta(X, a) = \emptyset$, or $\sum_{X' \in \delta(X,a)} Pr\{X \xrightarrow{a} X'\} = 1$.

[2] [5] shows the equivalence of a BN to several stochastic models.

[3] We can remove the cycles induced by the dashed arcs using a temporal BN, as we will show in later sections.

[4] We use a composability relation $\otimes$ that ensures that we can compose large models from a library of model components; see [8] for details.

We can also define an FSM using a graphical representation, in which a node representing X_1 is connected to a node representing X_2 by a directed edge labeled with transition σ iff $\delta(X_1, \sigma) = X_2$.

2) FSM Elevator Model: For the FSM approach, a model for the full elevator system is large and complex. Rather than overwhelm the reader with the details of this full model, we present a FSM model of one of its components, a position sensor. Note that, given the component FSMs, the complete system behaviour can then be obtained through parallel synthesis of these components [1]. We now present an FSM component model (and its corresponding BN model).

D. Component-Based Elevator Models

This section compares component models for the position sensor of the elevator system. A position sensor triggers when the elevator is at a fixed location. For example, upon the arrival of the elevator at position x (*i.e.*, event ix occurs), the sensor S_x moves from initial state X_{off} to state X_{on}, indicating that the sensor turns on (*i.e.*, outputs an "on" signal). Any motor actions (*i.e.*, $m+$) will then move the elevator from x to $\bar{x}$, and bring the sensor from state X_{on} to its initial state. We also assume that the sensor can fail, moving it from either X_{on} or X_{off} to failure state X_{fail}, through transition σ_f.

The states of a sensor involve the variables:

- *Pos*, the elevator position, which we can assume for this example has values $\{x, \bar{x}\}$;
- S_x, the sensor state, which has values {*on*,*off*};
- M, the sensor mode, which has values {*OK*, *fail*}.

If we represent a sensor state by the pair $\langle Pos, S, M\rangle$, then we have $X_{off} = \langle \bar{x}$,*off*,*OK*$\rangle$, $X_{on} = \langle x$,*on*,*OK*$\rangle$, and $X_{fail} = \langle \cdot, \cdot$,*fail*$\rangle$. Figure 2(a) shows the state machine model $\mathcal{F}_{S_x}$ for sensor S_x. In our PFSM, we describe the transitions using (transition, probability) pairs. The model has (a) two state-change transitions, (ix, π_2) and (m^+, π_1); (b) two self-transitions, $(\overline{ix}, \pi_4)$ and $(\overline{m^+}, \pi_3)$; and (c) two failure transitions (σ_{f1}, π_5) and (σ_{f2}, π_6). The transition probabilities obey: $\pi_1 + \pi_3 + \pi_6 = 1$ and $\pi_2 + \pi_4 + \pi_5 = 1$.

The two self-transitions have the following semantics. $\overline{ix}$ takes X_{off} back to itself, and $\overline{m^+}$ takes X_{on} back to itself. For example, the X_{off} self-transition corresponds to the sensor's response to the elevator waiting at the ground floor for a customer to arrive, and the probability for this self-transition corresponds to the relative amount of time that the elevator is idle on the ground floor.

The BN component model for S_x is straight-forward. As shown in Figure 2(b), the position *Pos* of the object on the conveyor and the failure mode M are the only variables that affect the output of S_x (assuming normal sensor operation).[5] For this BN we need to specify three distributions: *Pr*{*Pos*}, *Pr*{*M*} and *Pr*{S_x|*Pos*,*M*}.

Although this pair of corresponding models has great intuitive appeal, it permits only trivial BN control models.

[5]We assume here that we have what we call "naive" BN models, *i.e.*, atemporal variable-based models typically defined for diagnosis and control tasks.

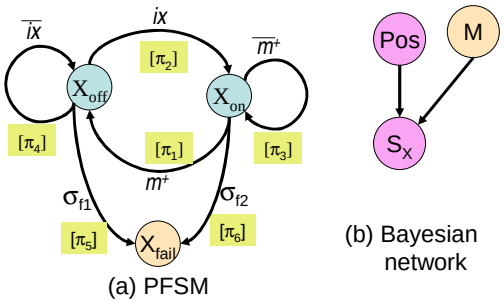

Fig. 2. (a) shows $\mathcal{F}_{S_x}$, the state machine for Sensor S_x. We show the event labels, as well as the transition probabilities as shaded $[\pi_i]$ labels. (b) shows the BN for the sensor, noting that S_x depends only on *Pos*, the position of the elevator, and M, the sensor mode variable.

Closer inspection of the BN models reveals some important restrictions, stemming from the order relations inherent in BNs. We explore these in Section III.

III. Equivalence of PFSM and Bayesian Network Representations

This section introduces the framework to prove that a temporal Bayesian network [18] can represent a PFSM. In particular, we examine the different order relations present in the BN/ID framework and in a typical discrete-event control framework. We show that a standard BN cannot represent a non-trivial control model, but that a temporal BN can capture the sequencing and timing inherent in control actions..

A. Order Relations for Decision-Making

We start out by examining the order relations for the BN/ID framework. We assume that we are addressing control, rather than control optimisation, tasks. As a consequence, we could represent the class of control problems in which we are interested as Influence Diagrams (IDs) that contain decision nodes but no value nodes. Under this interpretation, the chance nodes represent random variables, and the decision nodes represent actions (controls) available to the decision maker. We assume that there is no value function assigned to the outcomes of actions. Further, we assume that the arcs entering a chance node are quantified by conditional probabilities, and the arcs entering a decision node D indicate which quantities and previous decisions can be consulted before making decision D.

When formulating a decision (or control) problem as an influence diagram, a linear temporal ordering of the decision variables is required. This constraint ensures that the decision problem is well defined; however, it does not ensure that the control problem is well defined.

This ID-based interpretation of control is important, but does not allow us to address the rich set of control-theoretic issues defined within the control community. Further, it does not have a clear method for taking control actions on an event-based framework. For example, a factory lift-and-carry robot will close a clamp when a proximity sensor triggers, which is an event based on a part being positioned correctly.

It is precisely this event-based framework that is at the heart of many control specifications.

In the following, we extend this interpretation to allow a clear discrete-event semantics to be defined. To simplify our analysis, we adopt the approach specified in [2] for using BNs as IDs, as this does not change our analysis or ordering among nodes in the network.

B. Order Relations for Control Applications

This section examines the order relations specified for BNs and IDs (without temporal indexing of variables), and compares this with the order relations required by control applications.

One of the first key points to note is the importance of an event-based representation. Loosely speaking, an event-based representation specifies an ordered set of states with events defining state transitions. Such a system can be described using models such as the FSM model summarised in Section II-C.1. The representational power of a discrete-event control model is well-known; see, e.g., [1].

Lemma 1: For a discrete control model, an event-based control representation (FSM, Petri net) can adequately specify any control properties involving control sequencing of a discrete-event system.

We will use the PFSM representation as the gold-standard control model, and compare the ordering of a BN with that of a PFSM.

In a BN, an order relation $\preceq_B$ is imposed by the DAG $\mathcal{G}(V, E)$. For vertices $V_1,\ V_2 \in V$, we say that $V_1 \preceq_B V_2$ if V_1 is a predecessor of V_2 in $\mathcal{G}$.[6] In addition, we must have $V_1 \not\preceq_B V_1$, since this would make the DAG cyclic.

In a PFSM $\mathcal{F}$ (or other event-based control representation), we have an order relation $\preceq$ over *states*, which in turn will imply an order relation over the variables defining those states. For states $X_1,\ X_2 \in X$, we say that $X_1 \preceq X_2$ if $\exists$ a transition $(X_1, \sigma, X_2) \in \mathcal{F}$, or, loosely speaking, if X_1 is a predecessor of X_2 in $\mathcal{F}$. Since we allow self-loops and other forms of cycles in the graph depicting $\mathcal{F}$, the ordering $\preceq$ in general will violate a BN's ordering $\preceq_B$.

This analysis can be generalised into a result showing that only trivial deterministic BN control models can be generated using atemporal BNs.

Theorem 1: An atemporal, variable-based BN/ID cannot represent an event-based control model.[7]

For a BN to be able to capture the ordering $\preceq$ inherent in a PFSM $\mathcal{F}$, the variables in the BN must be composable into states that obey $\preceq$. One way to accomplish this is to define a temporal BN, in which we have temporally-indexed variables.[8] A temporal BN is defined over a set of temporal random variables. A temporal random variable is a random variable indexed by a time point t, $[V]_t$. A *time point* is one of t, i, $t+i$ or $t-i$, where i is a positive integer. Given these notions, we define a temporal BN as follows:

Definition 4 (Temporal Bayesian Network): A temporal Bayesian network $\mathcal{B}^t$ is a tuple $(\mathcal{G}, [\mathcal{P}]_t)$, where $\mathcal{G}$ is a DAG, and $[\mathcal{P}]_t$ is a set of *temporal probability distributions* constructed from vertices in $\mathcal{G}$ based on the topological structure of $\mathcal{G}$. $\mathcal{P}$ satisfies $Pr\{[\mathcal{V}]_t\} = \prod_{i=1}^{n} Pr\{[V_i]_\tau | pa([V_i]_t\}$, where $pa([V_i]_t)$ are the parents of $[V_i]_\tau$ in $\mathcal{G}$, and τ is either t or $t+1$.

In this representation, variables corresponding to temporal indices t and $t+1$ will obey the ordering $[V_i]_t \preceq_B [V_i]_{t+1}$ for all domain values of V_i. If we represent variables in different states using different temporal indices, then we obtain the desired correspondence between the PFSM and BN representations.

Lemma 2: A temporal, variable-based BN/ID can represent an discrete event-based stochastic control model.

C. Temporal Elevator Models

This section now shows what temporal models will look like for the elevator sensor.

Figure 3(a) shows the temporal BN for the sensor with faults for time slices t and $t+1$; the solid arcs cover edges within a time slice, and dashed arcs cover edges between time slices. Figure 3(b) shows the distributions for these variables. The failure-mode distribution shows that the sensor will move from being OK to failed with probability p; however, once failed, the sensor will remain failed.

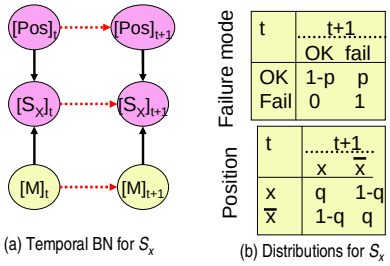

Fig. 3. (a) Temporal BN for failure scenario for Sensor S_x. (b) Distributions for failure scenario for Sensor S_x.

IV. Event-Based Bayesian Networks

This section describes our Event-Based Bayesian Network (EBN) representation.

A. Event-Based Bayesian Network Representation

Generating a PFSM from an EBN requires defining: (1) states, (2) event-based transitions, and (3) transition probabilities. We now define the notions of state, event and transition probability that will be generated from a BN representation.

Definition 5 (State): The state $\mathcal{X}$ of a system represented by a BN $\mathcal{B}$ is an assignment of values to V, i.e., $X \subseteq \times_i D_i$. Further, the state $\mathcal{X}_S$ of a sub-system represented by a BN sub-network $\mathcal{B}_S$ is an assignment of values to the nodes V_S in $\mathcal{B}_S$ such that $V_S \subseteq V$, i.e., $\mathcal{X}_S \subseteq \times_{i:V_i \in V_S} D_i$.

As in a PFSM, we denote Σ as the set of events. We define an event as follows:

[6]Note that this ordering captures the linear ordering on decision nodes required in an ID, since this linear ordering is represented by a predecessor ordering in a graph $\mathcal{G}$.

[7]We omit proofs due to space constraints.

[8]Note that we do not require every pair of variables to satisfy $\preceq$, but rather variable-sets (states) must satisfy $\preceq$. This allows a temporal BN with mixed temporal and atemporal variables.

Definition 6 (BN-Event): An event $\sigma \in \Sigma$ is a state transition. We denote a transition function over σ using $\delta : \Sigma \times \mathcal{X} \rightarrow \mathcal{X}$. Given an event (X, σ, X'), with states X, X' and corresponding variable domain-values denoted by $[V_X = D]$, $[V_{X'} = D']$, we define a BN-event as the variable-based representation of the event, *i.e.*, $([V_X = D], \sigma, [V_{X'} = D'])$.

We now define the transition probability corresponding to an arc in an EBN $\mathcal{B}_e$.

Definition 7 (Transition Probability): Given an event transition (X, σ, X'), the corresponding transition probability is given by $Pr\{X'|X\}$.

Given these definitions, we can now specify an Event-Based Bayesian Network.

Definition 8 (Event-Based Bayesian Network): An Event-Based Bayesian Network $\mathcal{B}_e = \mathcal{H}(V, E)$ is a BN model of a control system in which each node represents a state and every edge $E_i = (V_j, V_k)$ represents a transition distribution over the transition set defined in $V_j \times V_k$.

Using this notion of B_e, we will use the equivalence of B_e with an appropriate PFSM to denote its validity for control purposes. In other words, such an equivalence means that B_e will possess the same control properties than a PFSM has.

Definition 9 (Valid Event-Based BN): An Event-Based Bayesian Network $\mathcal{B}_e$ is a valid BN model of a control system if a 1:1 onto transformation exists to a PFSM for that control system.

Example: Figure 4 compares the BN, EBN and PFSM models for Sensor S_x. This figure shows that: (a) the BN captures the variables of the sensor as its nodes; (b) the Event-based BN defines the sensor states as its nodes, specifying a single causal arc from the state at t, $[X]_t$ to $[X]_{t+1}$, together with distributions for $Pr\{[X]_t\}$ and $Pr\{[X]_{t+1}|[X]_t\}$; and (c) the Probabilistic FSM models the sensor states as its nodes, specifying arcs for transitions, together with transition probabilities.

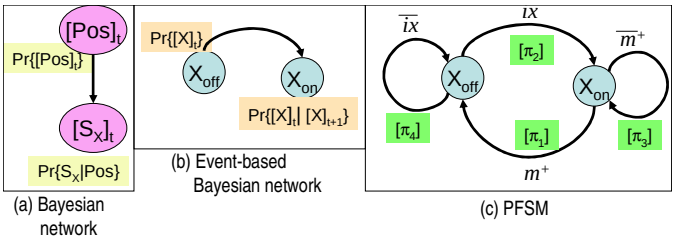

Fig. 4. Comparison of (a) BN, (b) Event-based BN and (c) Probabilistic FSM models for Sensor S_x. We explicitly show the probability distributions represented in each model by the shaded labels associated with nodes and edges.

B. Event-Based Bayesian Network Transformation

We can determine if a BN corresponds to a PFSM by transforming that BN into a PFSM. This transformation is performed in two steps, first transforming the BN into an EBN, and then transforming the EBN into a PFSM.

This section describes how we transform a BN into an EBN. We assume that we have the following representations:

- $\mathcal{B} = (\mathcal{G}(V, E), \mathcal{P})$ is a BN;
- $\mathcal{B}_e = (\mathcal{H}(V', E'), \mathcal{P}')$ is an event-based BN;
- $\mathcal{F} = (\mathcal{H}(V', E'), \Pi)$ is a PFSM.

We transform a BN $\mathcal{B}$ into an EBN by converting each family of $\mathcal{B}$ into an by EBN. For any BN family f, we:

- convert variables $V(f)$ into corresponding state $X(f)$;
- convert distributions $\mathcal{P}(f)$ into distributions for the corresponding states $X(f)$, which we call $\mathcal{P}(X)$.

To transform an EBN $\mathcal{B}_e$ into a PFSM $\mathcal{F}$ we:

- expand state $X(f)$ of $\mathcal{B}_e$ into a set of distinct (state,value) pairs, and order these based on the ordering induced by the cross-product ordering $\preceq_B$ of $V(f)$. Each (state,value) pair becomes a node of the PFSM.
- define transitions and transition probabilities for the ordered nodes of the PFSM. A transition (X_1, δ, X_2) exists if there is a non-zero probability $Pr\{X_2|X_1\}$ in $\mathcal{P}(X)$; the transition probability for δ is given by $Pr\{X_2|X_1\}$.

This series of transformations will map an arbitrary BN into a PFSM, thereby determining if the BN possesses the necessary ordering for control purposes.

V. Advantages of Bayesian Networks for Integrated Control Systems

BNs can provide improved inference and modeling for integrated control systems, as described below.

A. Efficient Algorithms

Significant research has been devoted to developing efficient BN inference algorithms, and a wide range of efficient exact and approximation algorithms have been developed [6]. For example, HMMs and Kalman filters have been outperformed by their more general temporal BN representations in applications such as speech recognition [19], due to BN optimisations and to limitations of HMMs and Kalman filters such as:

- the assumption of uni-modal (and often Gaussian) observation and state probability densities (e.g. in the Kalman filter),
- exponential growth of their parameterisation with the number of state variables (e.g. HMM), and
- static model structure (both Kalman filter and HMMs).

In general, stochastic DES applications can make direct use of these advances, e.g., by converting a PFSM into the equivalent stochastic model (such as an HMM or BN)[9], and then performing inference.

B. Modeling

A BN offers several potential modeling advantages over a PFSM, including greater generality, modeling compactness, and faster, more intuitive modeling.

Generality: A BN is a strict superset of the PFSM we have defined, and as such can model more complex systems than can a PFSM [5].

[9]The PFSM can represent the same distribution class as that modeled by an HMM, with the same time- and space-complexity [15].

Modeling Compactness: A BN can be used to model the states of a system or the variables defining the system. By adopting the latter approach, one needs a much more compact model in terms of number of nodes, if we rule out a trivial model where each variable has a single value.[10]

Lemma 3: Given a BN $\mathcal{B}$ with n variables $V_1, ..., V_n$, each of arity $k \geq 2$, the variable-based BN representation has fewer nodes than the state-based BN representation.

As an example, in the model of the elevator system, the BN model has 10 nodes, and the corresponding PFSM model has 96 nodes. This scale of difference in model size is typical for many typical models.

It is important to note the the BN implicitly represents states and state transitions, and a simple inference algorithm can generate all states and state transitions as required, as described in Section IV-B.

Intuitive Modeling: The modeling compactness of a BN means that system models are simpler to construct, store and visualise, using tools such as Hugin, Netica and Genie. Moreover, one is not restricted to building models by hand, as there is a significant literature in learning temporal BNs, e.g. [5], so that one can learn stochastic control models using a BN representation. Finally, as opposed to FSMs, BNs can directly encode system topology, and many BN inference algorithms use optimisations that take advantage of the topology of the BN. The techniques include the clique-tree algorithm [10], node-pruning [12], and cutset-conditioning [12].

VI. Concluding Remarks

This article has described a representation for stochastic discrete-event systems using a "traditional" variable-based Bayesian network representation. We now have a formalism that captures the control-theoretic properties of stochastic discrete-event systems but can also make use of the wealth of inference techniques developed in the Bayesian-network community. From the Bayesian network perspective, this represents the first specification of control-theoretic semantics where issues such as controllability, observability, etc. can be precisely defined. This representation also defines a single model that can be used for control synthesis and monitoring/diagnostics.

This work illustrates key differences between control and Bayesian network representations. First, control representations, such as PFSMs, have a graphical framework in which the nodes represent states, and arcs represent transitions between states. This representation does not explicitly represent physical relationships of the device being modeled; we have introduced the system causal graph to capture such relationships. On the other hand, Bayesian networks also have an underlying graphical framework, but in which the nodes typically represent *variables*, and arcs represent potential dependence relations among the variables. The notions of events and state transitions are *implicit*. The graphical structure can also capture physical relationships of the device being modeled. Second, an event-based control representation like PFSM can specify a sequence of state transitions without defining variables representing time; in contrast, in Bayesian networks specific ordering relations, euch as time-indexed variables, are needed for this task.

Much work remains to be completed. Additional work is needed to examine the applicability of this approach to real-world problems. These results extend the range of choices available for modeling (and monitoring/diagnosing) control systems; however, whether one implements an FSM with the extended properties outlined here, or a BN, requires additional study.

[10]The number of probabilities that must be specified in both model types is identical. In a variable-based model we specify a distribution over a variable given its parents in $\mathcal{G}$, i.e. $P(V_i|pa(V_i))$; in a state-based model we specify an equivalent set of probabilities in terms of state-transitions, i.e., $P(X_i|X_j)$, for appropriate X_j.

References

[1] C. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Kluwer, 1999.

[2] G.F. Cooper. A method for using belief networks as influence diagrams. In *Proceedings of the Workshop on Uncertainty in Artificial Intelligence*, pages 55–63, 1988.

[3] A. P. Dawid. Applications of a general propagation algorithm for probabilistic expert systems. *Statistics and Computing*, 2:25–36, 1992.

[4] E. Doberkat. *Stochastic Automata: Stability, Nondeterminism and Prediction*. Number 113. Springer Verlag, Lecture Notes in Computer Science, 1981.

[5] Zoubin Ghahramani. Learning dynamic Bayesian networks. *Lecture Notes in Computer Science*, 1387:168–197, 1998.

[6] H. Guo and W. Hsu. A survey on algorithms for real-time bayesian network inference. In *Joint AAAI-02/KDD-02/UAI-02 workshop on Real-Time Decision Support and Diagnosis Systems*, Edmonton, Alberta, Canada, 2002.

[7] R.A. Howard and J.E. Matheson. Influence diagrams. In R. Howard and J. Matheson, editors, *The Principles and Applications of Decision Analysis*, pages 720–762. Strategic Decisions Group, CA, 1981.

[8] J. Keppens and Q. Shen. On compositional modelling. *Knowledge Engineering Review*, 16(2):157–200, 2001.

[9] Pieter Kritzinger and F. Bause. *Introduction to Stochastic Petri Net Theory*. Advanced Studies in Computer Science, Vieweg Verlag, 1995.

[10] S. Lauritzen and D. Spiegelhalter. Local Computations with Probabilities on Graphical Systems and their Application to Expert Systems. *J. of the Royal Stat. Soc*, 50(2):157–224, 1988.

[11] A. Terry Morris and Peter A. Beling. Space Shuttle RTOS Bayesian Network. In *20th Digital Avionics Systems Conference*, 2001.

[12] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

[13] M. O. Rabin. Probabilistic Automata. *Information and Control*, 6:230–245, 1963.

[14] Peter Ramadge and W. Murray Wonham. The control of discrete-event systems. *Proceedings of the IEEE*, 77(1), 1989.

[15] E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta, and R. C. Carrasco. Probabilistic finite-state machines – Part I. *IEEE Trans. on Pattern analysis and Machine Intelligence*, 27(7):1013–1025, 2005.

[16] K. Virtanen, T. Raivio, and R.P. Hmlinen. Modeling pilot's sequential maneuvering decisions by a multistage influence diagram. *Journal of Guidance, Control, and Dynamics*, 27(4):665–677, 2004.

[17] A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13:260–269, 1967.

[18] Joel D. Young and Jr. Santos, Eugene. Introduction to temporal bayesian networks. In *Midwest Conference on Artificial Intelligence and Cognitive Science*, 1996.

[19] G. Zweig and S.J. Russel. Speech recognition with dynamic bayesian networks. In *American Association for Artificial Intelligence*, 1998.