GENERALIZED QUERIES AND BAYESIAN STATISTICAL MODEL CHECKING IN DYNAMIC BAYESIAN NETWORKS: APPLICATION TO PERSONALIZED MEDICINE

Christopher James Langmead
Computer Science Department, and
Lane Center for Computational Biology,
Carnegie Mellon University,
Pittsburgh, PA 15213, USA
Email: cjl@cs.cmu.edu

We introduce the concept of generalized probabilistic queries in Dynamic Bayesian Networks (DBN) — computing $P(\phi_1|\phi_2)$, where ϕ_i is a formula in temporal logic encoding an equivalence class of *trajectories* through the variables of the model. Generalized queries include as special cases traditional query types for DBNs (i.e., filtering, smoothing, prediction, and classification), but can also be used to express inference problems that are either impossible, or impractical to answer using traditional algorithms for inference in DBNs. We then discuss the relationship between answering generalized queries and the *Probabilistic Model Checking Problem* and introduce two novel algorithms for efficiently estimating $P(\phi_1|\phi_2)$ in a Bayesian fashion. Finally, we demonstrate our method by answering generalized queries that arise in the context of critical care medicine. Specifically, we show that our approach can be used to make treatment decisions for a cohort of 1,000 simulated sepsis patients, and that it outperforms Support Vector Machines, Neural Networks, and Random Forests on the same task.

Keywords: Dynamic Bayesian Networks, Model Checking, Systems Biology, Personalized Medicine, Sepsis

1. INTRODUCTION

The emerging field of *Translational Systems Biology*[18] seeks to optimize clinical practice in the context of *personalized medicine* by using the principles and methods of Systems Biology. In this paper, we consider scenarios where the dynamics of a particular disease are captured by a given *parameterized* model, $(\mathcal{M}, \pi)$. Here, $\mathcal{M}$ is a Dynamic Bayesian Network (DBN), and $\pi = P(\Theta)$ is a multivariate probability distribution over the parameters of $\mathcal{M}$. That is, π models our uncertainty with regard to the exact specification of the DBN, and thus $(\mathcal{M}, \pi)$ refers to a family of DBNs. In a clinical setting, the first goal is to obtain a *patient-specific* estimate of the posterior $\hat{\pi} = P(\Theta|O_i)$ where O_i refers to the clinical observations from patient i. The second goal is to query the resulting family of DBNs, $(\mathcal{M}, \hat{\pi})$, to make patient-specific predictions relevant to selecting an appropriate medical intervention. In this paper, we present two algorithms for answering such queries in a Bayesian fashion, given $\mathcal{M}$ and the distribution $\hat{\pi}$.

A unique feature of our algorithms is that in addition to being able to compute the probability of being in a particular state at a particular time (e.g., *"What is the probability that the patient will be in renal failure exactly 12 hours from now?"*), they can *also* be used to compute probabilities over sets of *trajectories* satisfying a *logical* constraint (e.g., *"What is the probability that the amount of tissue damage never exceeds critical value c?"*), or a logical ordering of events (e.g., *"What is the probability that the patient will enter renal failure* ***before*** *clearing the primary infection?"*). In a clinical setting, the ability to answer such questions will enable physicians to make more nuanced decisions regarding treatment strategies.

Our paper makes several contributions: *First*, we introduce the notion of a *generalized probabilistic queries* for DBNs. Generalized probabilistic queries are a strict superset of more traditional probabilistic queries over DBNs (i.e., filtering, smoothing, prediction, classification). They have the form $P(\phi_1|\phi_2)$, where ϕ_i is a formula in *temporal logic* encoding a set of trajectories. *Second*, we reduce the problem of answering generalized queries over DBNs to an instance of the *Probabilistic Model Checking Problem* which is: given a stochastic model, a formula in temporal logic, ϕ, and a probability threshold, $\rho \in [0, 1]$, decide whether the model satisfies ρ. *Third*, we introduce two novel algorithms for solving the Probabilis-

tic Model Checking Problem in a Bayesian fashion. Our algorithms differ from related work, including our own[9], in that they are fully Bayesian. *Finally*, we demonstrate our method on a model of the acute inflammatory response to sepsis[14]. We show that our approach to answering generalized queries is more accurate than Support Vector Machine, Neural Networks, and Random Forests on classification tasks relevant to making treatment decisions.

This paper is organized as follows: We present the necessary background on Dynamic Bayesian Networks and introduce the notion of generalized queries in DBNs in Sec. 2. Our algorithms are presented in Sec. 3. We demonstrate our method on a model of acute inflammation in Sec. 4. We compare and contrast our method to related work in Sec. 5. We conclude and discuss ideas for future work in Sec. 6.

2. BACKGROUND AND NOTATION

In what follows, $\mathbf{X} = \{X_1, ..., X_n\}$ and $\mathbf{Y} = \{Y_1, ..., Y_m\}$ are sets of random variables modeling the *hidden* (aka latent or unobservable) and *observable* states of a stochastic process, respectively. The vectors $\mathbf{x}_t = (x_1(t), ..., x_n(t))$ and $\mathbf{y}_t = (y_1(t), ..., y_m(t))$ denote the state of $\mathbf{X}$ and $\mathbf{Y}$ at discrete time $t \in \mathbb{Z}$, respectively. A *trajectory*, or sequence of states from time $t = \tau_1$ to $t = \tau_2$, such that $\tau_2 > \tau_1$ is denoted by $\mathbf{x}_{\tau_1:\tau_2} = (\mathbf{x}(\tau_1), ..., \mathbf{x}(\tau_2))$.

2.1. Dynamic Bayesian Networks

Dynamic Bayesian Networks (DBN) comprise a family of *probabilistic graphical models* for modeling sequential data. Familiar instances of DBNs include Hidden Markov Models and Kalman Filters, which have been used in numerous domains, including Biology. Informally, a DBN consists of factored encodings of three probability distributions: (i) a prior over $\mathbf{X}$, $P(\mathbf{X}_0)$; (ii) a state transition model, $P(\mathbf{X}_t|\mathbf{X}_{0:t-1})$; and (iii) an observation model, $P(\mathbf{Y}_t|\mathbf{X}_{0:t}, \mathbf{Y}_{0:t-1})$. The parameters needed to specify these three distributions will be denoted by the vector Θ.

The exact size and nature of Θ will vary based on conditional independence assumptions made by the model. The parameters for a Hidden Markov Model, for example, correspond to the elements of conditional probability tables encoding $P(\mathbf{X}_t|\mathbf{X}_{t-1})$ and $P(\mathbf{Y}_t|\mathbf{X}_t)$.

In many settings, a point-estimate for Θ is learned from a set of training data. The resulting DBN is then used to answer probabilistic queries. In this paper, however, we adopt a Bayesian view and instead consider the distribution over parameters, $P(\Theta)$. This distribution models our residual uncertainty in the parameters due to, for example, measurement errors, fixed samples sizes, etc. We refer to the family of DBNs implied by $P(\Theta)$ as a *Parameterized Dynamic Bayesian Network*.

Definition 2.1. Parameterized DBNs: *A parameterized DBN is a pair, $(\mathcal{M}, \pi)$, where $\mathcal{M}$ is a Dynamic Bayesian Network over state variables $\boldsymbol{X}$ and $\boldsymbol{Y}$, and $\pi = P(\Theta)$ is a probability distribution over the parameters defining $P(\boldsymbol{X}_0)$, $P(\boldsymbol{X}_t|\boldsymbol{X}_{0:t-1})$, and $P(\boldsymbol{Y}_t|\boldsymbol{X}_{0:t}, \boldsymbol{Y}_{0:t-1})$.*

2.2. Probabilistic Queries in DBNs

DBNs are traditionally used to answer probabilistic queries concerning the state of the system at a particular position in the sequence. Traditional queries fall into several categories:

Definition 2.2. Traditional DBN Queries

- *Filtering*: Computing $P(\mathbf{x}_\tau|\mathbf{y}_{0:\tau})$
- *Smoothing*: Computing $P(\mathbf{x}_{\tau-l}|\mathbf{y}_{0:\tau})$
- *Prediction*: Computing $P(\mathbf{x}_{\tau+h}|\mathbf{y}_{0:\tau})$
- *Decoding*: Computing $\arg\max_{\mathbf{x}} P(\mathbf{x}_{0:\tau}|\mathbf{y}_{0:\tau})$
- *Classification*: Computing $P(\mathbf{y}_{0:\tau})$

Here, $l > 0$ and $h > 0$.

These queries can be answered using one of many existing algorithms for probabilistic inference. The reader is directed to [12] for more information on traditional inference algorithms for DBNs.

Notice that the queries in Def. 2.2 each involve conditioning the model on a *single* trajectory (i.e., $\mathbf{y}_{0:\tau}$), and then computing either the belief state at a *single* instant in time (for filtering, smoothing, and prediction), or the probability of a *single* trajectory (for decoding and classification). While useful, there are limits to the expressive power of these queries. For example, consider a DBN where the state variables track the absence or presence of specific somatic

mutations in a tumor. One might be interested in the likelihood that some particular mutation m_1 arises before some other mutation m_2 at some point within the next t time units. Unfortunately, such questions are not easily expressed in terms of the queries in Def. 2.2. The problem is that such questions refer to the set of trajectories that satisfy a *logical* property (i.e., the set of all trajectories such that m_1 arises before m_2). This set can be exponentially large in t for finite-state models, like HMMs, or infinitely large for continuous-state models, like Kalman Filters. Thus it is not possible, in general, to compute the probability mass of the set through explicit enumeration of all trajectories.

Our first goal in this paper is to generalize the notion of a probabilistic query by considering the problem of conditioning the DBN on a set of trajectories: $\mathcal{Y} = \{\mathbf{y}^0_{0:\tau_1}, ..., \mathbf{y}^k_{0:\tau_k}\}$, and/or performing inference over a set of hidden trajectories: $\mathcal{X} = \{\mathbf{x}^1_{0:\tau_1}, ..., \mathbf{x}^{k'}_{0:\tau_{k'}}\}$. Note that we do not assume that the trajectories in $\mathcal{X}$ (resp. $\mathcal{Y}$) are of equal length.

Generalized queries give rise to two technical challenges. The first challenge is to compactly represent $\mathcal{X}$ and/or $\mathcal{Y}$. The second challenge is solving the inference problem without resorting to an explicit enumeration of all trajectories in $\mathcal{X}$ or $\mathcal{Y}$. We deal with the first challenge in the next section, and the second challenge in Section 3.

2.3. Temporal Logic

Temporal logic is a formalism for describing behaviors in finite-state systems. It has been used since 1977 to reason about the properties of concurrent programs[13]. There are a variety of temporal logics from which to chose. The interested reader is directed to [5] for more information. The probabilistic queries we wish to consider can be written as formulas in *Bounded Linear Temporal Logic* (BLTL). The syntax and semantics of the BLTL are defined below.

Recall that $\mathbf{X}$ is a set of random variables. Let $\bowtie \in \{>, \geq, <, \leq, = \}$. A Boolean predicate over $\mathbf{X}$ is a constraint of the form $X_i \bowtie c$, where $X_i \in \mathbf{X}$ and $c \in \mathbb{R}$. A BLTL formula is built on a finite set of Boolean predicates over $\mathbf{X}$ using Boolean connectives and temporal operators. The syntax of BLTL is given by the following grammar:

Definition 2.3. Syntax of BLTL

$$\begin{aligned} \textit{State formulas:} \quad & \phi \equiv \mathbf{True} \mid X_i \bowtie c \mid \neg\phi \mid \phi_1 \vee \phi_2 \mid \\ & \phi_1 \wedge \phi_2 \mid \psi \\ \textit{Path formulas:} \quad & \psi \equiv \mathbf{N}\ \phi \mid \phi_1 \mathbf{U^t} \phi_2 \end{aligned}$$

where $\vee$, $\wedge$, and $\neg$ are the usual logical operators and **True** is the Boolean value for true. The temporal operator **N** is read "next" and requires that its argument, ϕ, is true in the next time step. The temporal operator $\mathbf{U^t}$ is read "until time t" and requires that its second argument, ϕ_2, becomes true by time t and that its first argument, ϕ_1, is true until ϕ_2 becomes true.

We can derive additional logical and temporal operators from the definitions above. For example, the logical implication operator can be defined as $A \implies B = \neg A \vee B$. Derived temporal operators include:

$$\begin{aligned} \mathbf{F^t}\phi &\equiv \mathbf{True}\ \mathbf{U^t}\ \phi \\ \mathbf{G^t}\phi &\equiv \neg\mathbf{F^t}\neg\phi \\ \phi_1 \mathbf{R}^t \phi_2 &\equiv \neg(\neg\phi_1 \mathbf{U^t} \neg\phi_2) \end{aligned}$$

Here **F** is read "eventually", **G** is read "always", and **R** is read "release". The bounded eventually operator requires that its argument becomes true within t units of time. The bounded always operator requires that its argument stays true during at least t units of time. Finally, the bounded release operator requires that ϕ_2 is true until the first position where ϕ_1 is true, or until time t, if ϕ_1 does not become true within the time bound.

Let $\mathbf{Z} \subseteq \{\mathbf{X} \cup \mathbf{Y}\}$, and let $\mathbf{z}_{0:\tau}$ be a trajectory through the DBN. The fact that $\mathbf{z}_{0:\tau}$ satisfies property ϕ is denoted by $\mathbf{z}_{0:\tau} \models \phi$. The semantics of BLTL are defined recursively as follows:

Definition 2.4. Semantics of BLTL

- $\mathbf{z}_{0:\tau} \models \mathbf{True}$, always;
- $\mathbf{z}_{0:\tau} \models Z_i \bowtie c$ iff $Z_i \in \mathbf{Z}$ and $Z_i(0) \bowtie z$;
- $\mathbf{z}_{0:\tau} \models \neg\phi$ iff $\mathbf{z}_{0:\tau} \not\models \phi$.
- $\mathbf{z}_{0:\tau} \models \phi_1 \vee \phi_2$ iff $\mathbf{z}_{0:\tau} \models \phi_1$ or $\mathbf{z}_{0:\tau} \models \phi_2$;
- $\mathbf{z}_{0:\tau} \models \phi_1 \wedge \phi_2$ iff $\mathbf{z}_{0:\tau} \models \phi_1$ and $\mathbf{z}_{0:\tau} \models \phi_2$;
- $\mathbf{z}_{0:\tau} \models \mathbf{N}\phi$ iff $\mathbf{z}_1 \models \phi$.
- $\mathbf{z}_{0:t} \models \phi_1 \mathbf{U}^t \phi_2$ iff $\exists\ i \in \mathbb{N}$ such that (i) $i \leq t$, (ii) $\mathbf{x}_i \models \phi_2$, and (iii) $\forall\ 0 \leq j < i$, $\mathbf{z}_j \models \phi_1$.

Notice that a "traditional" observation sequence, $\mathbf{y}_{0:\tau}$, can be encoded as a formula in temporal logic by construction:

$$\mathbf{y}_{0:\tau} \equiv \phi ::= [\phi_0 \wedge \mathbf{N}(\phi_1 \wedge \mathbf{N}(\phi_2 \wedge ... \wedge \mathbf{N}(\phi_t)...))]$$

where ϕ_i is a Boolean predicate that is true if $\bigwedge_{j \in \mathbf{Y}} Y_j(t) = y_j(t)$. Of course, the real advantage of a formula is that it defines a set of trajectories. Returning to our earlier example, suppose X_1 and X_2 are Boolean random variables indicating the presence of mutations m_1 and m_2, respectively. That is, X_i is 1 if mutation m_i is present, and 0 otherwise. The formula $\phi = \neg X_2 U^t X_1$ refers to the set of trajectories such that mutation m_1 arises before mutation m_2 within the next t time units.

2.4. Generalized Queries in DBNs

Given the definition of BLTL, we can now formally define generalized queries for DBNs. We note that the distinction between filtering, smoothing, and prediction goes away in this context, and so we will simply refer to these tasks as generalized filtering.

Definition 2.5. Generalized DBN Queries

- *Generalized Filtering*:
  Computing $P(\phi_1|\phi_2) = P(\phi_1 \wedge \phi_2)/P(\phi_2)$
- *Generalized Decoding*:
  Computing $\arg\max_{\mathbf{x}} P(\mathbf{x}_{0:\tau}|\phi)$
- *Generalized Classification*:
  Computing $P(\phi)$

Here, ϕ, ϕ_1, and ϕ_2 are formulas in BLTL.

This paper focuses on the generalized filtering and classification problems for parameterized DBNs. The algorithms presented here can be applied as-is to traditional (i.e., non-parameterized) DBNs as well. Developing algorithms for solving the generalized decoding problem is part of ongoing research.

2.5. Probabilistic Model Checking and Generalized Queries in DBNs

In this section, we briefly summarize the relationship between the task of answering generalized queries in DBNs and the *Probabilistic Model Checking Problem*. We begin by first defining the Model Checking Problem. Let $\mathbf{M}$ be a (non-stochastic) model over a set of states, $\mathcal{S}$, let $\mathcal{S}_0 \subseteq \mathcal{S}$ be a set of initial states, and let ϕ be a formula in temporal logic. The Model Checking Problem is deciding whether $\mathbf{M}, \mathcal{S}_0 \models \phi$. That is, whether, when starting from $\mathcal{S}_0$, $\mathbf{M}$ satisfies ϕ. Historically, Model Checking algorithms were developed for formally proving properties of non-stochastic concurrent systems, such as circuit designs. The interested reader is directed to [5] for more information on traditional algorithms for, and applications of Model Checking. More recently, there has been considerable interest in the development of Model Checking algorithms for stochastic systems, such as Discrete and Continuous-Time Markov Chains. For stochastic systems, the Probabilistic Model Checking Problem (PMCP) is deciding whether the model satisfies ϕ with probability greater than or equal to some given $\rho \in [0, 1]$.

It is easy to see the relationship between solving the generalized filtering and classification problems in DBNs, and solving the PMCP. In particular, any algorithm that can be used to compute (or approximate) $P(\phi)$, can then be used to create a decision procedure to solve the PMCP by comparing $P(\phi)$ to ρ. Alternatively, any algorithm that solves the PMCP can be used to compute upper and lower bounds on $P(\phi)$ by performing a binary search over ρ. The algorithms in this paper estimate $P(\phi)$ for parameterized DBNs, and therefore can also be used to solve the PMCP for such models. To the best of our knowledge, the only previous use of graphical models to perform Probabilistic Model Checking is [22], who considered Hidden Markov Models (HMMs). In contrast, this paper considers parameterized DBNs, which are a much broader class of graphical models. Moreover, unlike existing algorithms for solving the PMCP, ours are fully Bayesian.

3. ALGORITHMS

The inputs to both of our algorithms are a parameterized DBN, as defined in Def. 2.1, and a generalized probabilistic classification query, as defined in Def. 2.5.

3.1. Sample-Based Algorithm

The simplest, and most widely applicable algorithm involves sampling trajectories from the model in an

Algorithm 3.1 Bayesian Statistical Model Checking

Require: (i) A Parameterized DBN $(\mathcal{M}, \pi)$; (ii) a Probabilistic BLTL formula, ϕ; (iii) Beta distribution shape parameters, α and β, encoding the prior density for unknown parameter ρ; and (iv) threshold T.

$n := 0$ {*number of trajectories drawn so far*}
$k := 0$ {*number of trajectories satisfying ϕ so far*}
repeat
  $\theta :=$ draw an i.i.d. sample from π. {*Sample a set of parameters for the DBN*}
  $\sigma :=$ draw an i.i.d. sample trajectory from $\mathcal{M}$ with parameters θ. {*The length of the trajectory is determined by the temporal operators in ϕ.*}
  $n := n + 1$
  if $\sigma \models \phi$ **then**
    $k := k + 1$
  end if
  $\hat{\rho} :=$ Estimate(n, k, α, β) {*compute according to Equation (1)*}
  $\hat{\nu} :=$ Variance(n, k, α, β) {*compute according to Equation (2)*}
until $\hat{\nu} < T$
return $\hat{\rho}$,$\hat{\nu}$

i.i.d. fashion. Notice that any trajectory through the model either satisfies a given BLTL formula, ϕ, or it does not. We can therefore model $P(\phi)$ using a Bernoulli distribution and use the sampled trajectories to statistically estimate the Bernoulli success parameter, ρ.

The true value of ρ can be estimated in a maximum likelihood fashion by simply keeping track of the ratio of the number of satisfying trajectories versus the total number of trajectories sampled. In contrast, the algorithms in this paper estimate ρ in a Bayesian fashion. Here, it is necessary to specify a prior distribution over ρ. The conjugate prior of a Bernoulli distribution is the Beta distribution, and so we define the prior over ρ as:

$$P(\rho) = \frac{1}{\mathfrak{B}(\alpha, \beta)} \rho^{\alpha-1}(1-\rho)^{\beta-1}$$

where $\mathfrak{B}$ is the Beta function, and α and β are the shape parameters of the Beta distribution. In the context of critical care medicine, the shape parameters can be obtained from clinical trial data. If desired, an unbiased prior can be specified by using a Beta distribution with parameters $\alpha = \beta = 1$, which is equivalent to a uniform distribution.

Algorithm 3.1 performs Bayesian Statistical Model Checking by sampling trajectories. It proceeds by iteratively sampling initial parameters from π (i.e., the prior distribution over the parameters of the DBN), instantiating a DBN with those parameters, and then generating a trajectory of appropriate length. Notice that the path formulas use either the next operator ($\mathbf{N}\phi$), or the bounded until operator, ($\phi_1 \mathbf{U}^t \phi_2$). Thus, the length of the sampled trajectory is 2 if the formula is of the form $P(\mathbf{N}\phi)$, or $t+1$ if the formula is of the form $P(\phi_1 \mathbf{U}^t \phi_2)$.

Each sampled trajectory is tested to determine whether it satisfies the formula ϕ. The number of satisfying trajectories is then used to compute the mean, $\hat{\rho}$, and the variance, $\hat{\nu}$, of the Bernoulli distribution using the following well-known formulas:

$$\hat{\rho} = \frac{k + \alpha}{\alpha + \beta + n} \tag{1}$$

$$\hat{\nu} = \frac{(\alpha + k)(n - k + \beta)}{(\alpha + n + \beta)^2(\alpha + n + \beta + 1)} \tag{2}$$

These equations compute the mean and variance of the posterior distribution over ρ, which is a Beta function with parameters $\alpha' = \alpha + k$ and $\beta' = n - k + \beta$. The algorithm terminates when $\hat{\nu}$ is less than some user-specified threshold T.

3.2. Recursive Algorithm

The second algorithm solves the same problem for any DBN that can be explicitly converted into

a Discrete-Time Markov Chain (DTMC). This includes any finite-state DBN, and assumes that the number of unique states $S = (X_1 \times X_2 \times ... \times X_n \times Y_1 \times ... \times Y_m)$ can be explicitly enumerated. Consequently, the second algorithm is less widely applicable than the first, but does have the advantage of being exact.

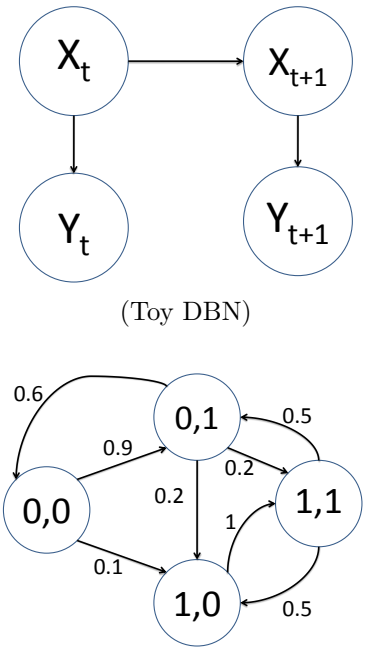

Fig. 1. (Top) A toy DBN. X and Y are binary random variables. (Bottom) The DTMC corresponding to the DBN. Nodes are labeled by the state of (x, y). Edges are labeled with transition probabilities.

Let $\mathcal{D} = (S, \mathcal{P})$ be a DTMC where $\mathcal{P} : S \times S \rightarrow [0, 1]$ is a stochastic transition matrix where element $\mathcal{P}(i, j)$ is the probability that state s_i transitions to state s_j in the next time step. For parameterized DBNs we assume that each $\mathcal{P}(i, j)$ is the *maximum a posteriori* (MAP) transition probability between states s_i and s_j. The prior distribution over S will denoted as $\mathcal{B}_0$.

Example: Figure 1-top depicts a simple DBN over two binary random variables. The DTMC corresponding to the DBN is shown in Fig. 1-bottom. Here, $S = \{(0, 0), (0, 1), (1, 0), (1, 1)\}$ and

$$\mathcal{P} = \begin{pmatrix} 0 & .9 & 0.1 & 0 \\ 0.6 & 0 & 0.2 & 0.2 \\ 0 & 0 & 0 & 1 \\ 0 & 0.5 & 0.5 & 0 \end{pmatrix}$$

Assume that $\mathcal{B}_0 = (0.1, 0.1, 0.3, 0.5)$.

Evaluating State Formulas We first consider how to evaluate state formulas (Def. 2.3). Recall that we have assumed that it is possible to explicitly enumerate each state of the system. Therefore, we can also assume that it is possible to label each state as to whether it satisfies a given atomic proposition. By 'atomic proposition', we mean any sub-formula of the form $\phi \equiv Z_i \bowtie c$. Assuming a fixed ordering of the states, we can represent the set of states satisfying an atomic proposition using a bit vector, Φ, of length $|S|$. We evaluate sub-formulas involving the logical operators $\neg, \wedge$, and $\vee$ by implementing set differences, intersections, and unions as bit-vector operations.

In order to compute the probability of satisfying a given state formula at time t, we need to have the belief state at that time. Let $\mathcal{B}_t$ be a vector encoding the belief state at time t. The probability of being in any state that satisfies a given state formula is obtained by first computing the Hadamard product of Φ and $\mathcal{B}_t$, and then summing the non-zero elements of the resulting vector.

Example: Suppose $\phi \equiv X = 1$. States s_3 and s_4 in Fig. 1 satisfy ϕ. Therefore, $\Phi = (0, 0, 1, 1)$. The prior probability of being in a state that satisfies ϕ is thus $P(\phi) = \sum_{s_i \in S} \mathcal{B}_0(s_i) \circ \Phi(s_i) = 0.8$.

Evaluating Path Formulas The probability of path formulas is also computed via matrix operations. There are two cases to consider: the next operator ($\mathbf{N}\ \phi$) and the bounded until operator ($\phi_1 \mathbf{U^t} \phi_\mathbf{2}$).

$P(\mathbf{N}\ \phi)$ is computed as follows: Let Φ be the bit vector labeling the states satisfying ϕ. We first compute the maxtrix-vector product $V = \mathcal{P} \times \Phi$. Note that the ith element of vector V is the probability that, when starting in state $s_i \in S$ at time t, the system is in some state satisfying ϕ at time $t + 1$. We obtain $P(\mathbf{N}\ \phi)$ by summing the elements of the Hadamard product of V and the belief state $\mathcal{B}_t$.

Example: Returning to our example, $V = \mathcal{P} \times$

$\Phi = (0.1, 0.4, 1, 0.5)$, and so starting at time $t = 0$ $P(\mathbf{N}\ \phi) = \sum_{s_i \in S} \mathcal{B}_0(s_i) \circ V(s_i) = 0.6$.

$P(\phi_1 \mathbf{U^t} \phi_\mathbf{2})$ is computed as follows: We first partition S into three disjoint sets S_1, S_2, and S_3. The elements of S_1 include those states satisfying ϕ_2. The elements of S_2 include those states that satisfy neither ϕ_1 or ϕ_2. The remaining states are placed into set S_3. Informally, S_1 contains the states that trivially satisfy the formula $\phi_1 \mathbf{U^t} \phi_\mathbf{2}$, S_2 contains those that trivially do not satisfy the formula, and S_3 contains those that may or may not satisfy the formula. We next define a new transition probability matrix $\mathcal{P}'$ such that:

$$\mathcal{P}'(i,j) = \begin{cases} 1 & \text{if state } s_i \in S_1 \text{ and } i == j, \\ \mathcal{P}(i,j) & \text{if state } s_i \in S_3, \\ 0 & \text{otherwise} \end{cases}$$

We use $\mathcal{P}'$ to recursively compute the vector $V_t = \mathcal{P}' \times V_{t-1}$. The base case, V_0, is simply the vector Φ_2, which is a bit vector labeling the states satisfying ϕ_2. The complete calculation requires t matrix-vector calculations. We obtain $P(\phi_1 \mathbf{U^t} \phi_\mathbf{2})$ by summing the elements of the Hadamard product of V_t and the belief state $\mathcal{B}_0$.

Example: Let $\phi_1 \equiv Y = 0$, $\phi_2 \equiv X = 1$, and $t = 2$. Therefore, $S_1 = \{s_3, s_4\}$, $S_2 = \{s_2\}$, and $S_3 = \{s_1\}$. Thus,

$$\mathcal{P}' = \begin{pmatrix} 0 & .9 & 0.1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Hence, $V_1 = \mathcal{P}' \times \Phi_2 = (0.1, 0, 1, 1)$ and $V_2 = \mathcal{P}' \times V_1 = (0.1, 0, 1, 1)$. So, starting at time $t = 0$ $P(\phi_1 \mathbf{U^2} \phi_\mathbf{2}) = \sum_{\mathbf{s_i} \in \mathbf{S}} \mathcal{B}_\mathbf{0}(\mathbf{s_i}) \circ \mathbf{V_2}(\mathbf{s_i}) = \mathbf{0.81}$.

4. APPLICATION TO A MODEL OF THE DYNAMICS OF ACUTE INFLAMMATION

We applied our method to a model of the acute inflammatory response to infection presented in [14]. The model has 4-equations and 22-parameters.

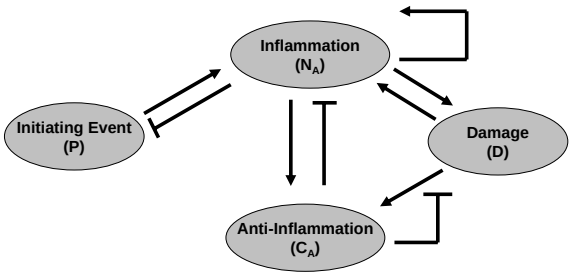

Fig. 2. Cartoon representation of the 4-equation model of the acute immune response. Arrows represent up-regulation, bars represent down-regulation. Figure is adapted from Figure 1 in [14].

The acute inflammatory response to infection has evolved to promote healing by ridding the organism of the pathogen. The actual response is a complex and carefully regulated combination of molecular and cellular cascades that exhibit both pro and anti-inflammatory behaviors. The pro-inflammatory elements are primarily responsible for eliminating the pathogen, but killing bacteria can cause collateral tissue damage. Tissue damage, in turn, triggers an escalation in the pro-inflammatory response creating a positive feedback cycle (Figure 2). The anti-inflammatory elements counteract this cycle, thereby minimizing tissue damage and promoting healing. However, in cases of extreme infection, the delicate balance between pro and anti-inflammatory elements is destroyed, resulting in a potentially lethal amount of tissue damage.

The 4-equation model is as follows:

$$\frac{dB}{dt} = k_{pg} B\Big(1 - \frac{B}{p_\infty}\Big) - \frac{k_{pm} s_m B}{\mu_m + k_{mp} B} - k_{pm} f(N_A) B,$$

$$\frac{dN_A}{dt} = \frac{s_{nr} R}{\mu_{nr} + R} - \mu_n N_A,$$

$$\frac{dD}{dt} = k_{dn} f_s(f(N_A)) - \mu_d D,$$

$$\frac{dC_A}{dt} = s_c + \frac{k_{cn} f(N_A + k_{cmd} D)}{1 + f(N_A + k_{cmd} D} - \mu_c C_A,$$

where:

$$R = f(k_{nn} N_A + k_{np} B + k_{nd} D)$$

$$f(V) = \frac{V}{(1 + (C_A/c_\infty)^2)}$$

$$f_s(V) = \frac{V^6}{x_{dn}^6 + V^6}$$

Here, k_*, μ_*, s_*, p_* are parameters, as defined in [14]. The state variables B, N_A, D, and C_A, correspond to the amounts of pathogen, pro-inflammatory mediators (e.g., activated neutrophils), tissue damage, and anti-inflammatory mediators (e.g., cortisol and interleukin-10), respectively. Figure 3 presents sample trajectories from the model. There are 3 clinically important outcomes: (i) a return to health, characterized by a drop in the pathogen load (B) to basal levels with minimal amounts of tissue damage (D), (ii) aseptic death, where the pathogen is cleared, but the tissue damage is large enough to cause death, and (iii) septic death, where the pathogen is not cleared, and the tissue damage is large enough to cause death.

Converting the ODE model into a DBN

We used the method of [17] to convert the ODE model into a DBN. That same method can be used to estimate the distribution over the parameters, $P(\Theta|O)$, given observations. The unobserved variables of the DBN correspond to the variables B, N_a, C_a, and D in the ODE model. That is, $\mathbf{X} = \{B, N_a, C_a, D\}$. The observable variables were taken to be noisy measurements of N_a and C_a, as it is reasonable to assume that these variables can be estimated from blood draws. The noisy observations were generated from the true values by adding a Gaussian noise.

Experiments

The unknown parameters of our DBN correspond to mean and the covariance of the four state variables, B, N_a, C_a, and D, and the parameter k_{pg}, which corresponds to the growth rate of the pathogen. We generated a simulated cohort of 1,000 patients by randomly generating a multivariate Gaussian probability distribution over Θ. Thus for the ith patient, $P_i(\Theta) = \mathcal{N}(\mu_i, \Sigma_i)$. These distributions define a family of DBNs, and represent the uncertainty of the state of the patient. The means and the covariances for each patient were obtained by sampling from a master distribution consistent with the study described in [14]. We note that the parameters are not mutually independent. That is, Σ_i has non-zero elements off of the diagonal.

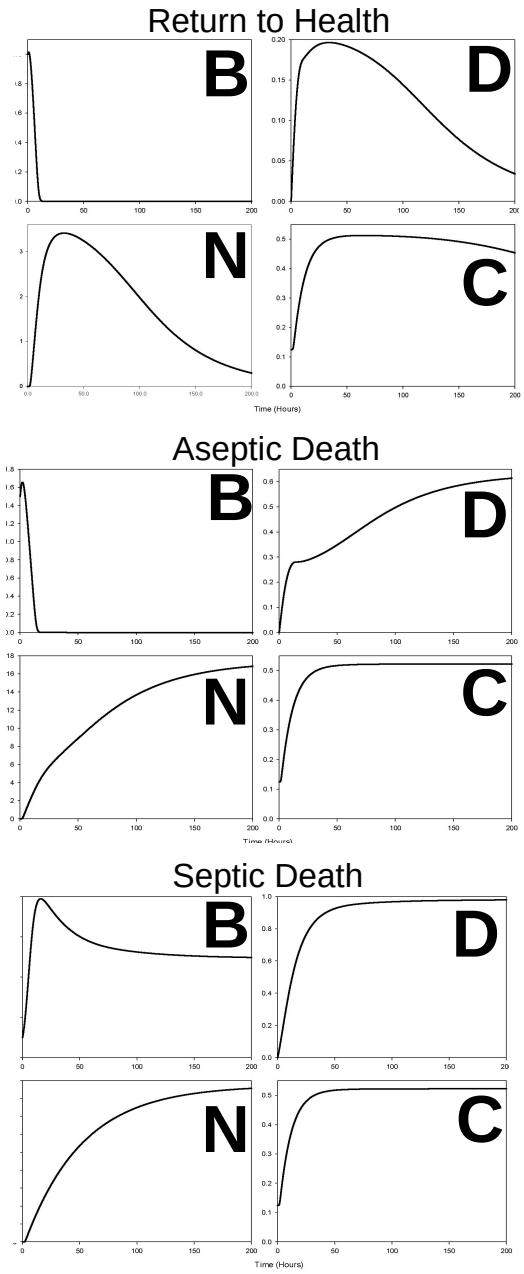

Fig. 3. Example trajectory from the 4-equation model. These three trajectories correspond to three typical outcomes (return to health, aseptic death, septic death). Time is measured in hours.

Generalized Queries

From a clinical perspective, the pathogen load and the amount of tissue damage are the primary threats to the patient. Thus, we will consider generalized queries that compute the likelihood that the patient reaches one of the three primary outcomes (i.e.,

health, aseptic death, septic death). Additionally, among the treatment options available to ICU physicians, who are the ones most likely to be treating patients with sepsis, is an anti-inflammatory therapy. This can be modeled by increasing the amount of state variable C_a. The goal of this therapy is to de-escalate the pro-inflammatory response, which induces tissue damage. However, it has been shown that such therapies can also be ineffective or worse, harmful to patients[6]. Therefore, we will also consider queries that compute the likelihood that the use of anti-inflammatory therapy will achieve the desired treatment goals.

Estimating the likelihood of the various outcomes: We define the following formulas in BLTL:

- $\phi_H \equiv \neg(D > \kappa_1)\ U^{72}\ (B < \kappa_2)$,
- $\phi_A \equiv \textbf{True}\ U^{72}\ (D > \kappa_1) \wedge (B < \kappa_2)$
- $\phi_S \equiv \textbf{True}\ U^{72}\ (D > \kappa_1) \wedge \neg(B < \kappa_2)$

ϕ_H defines the set of trajectories where the pathogen falls below a threshold κ_2 within the first 72-hours, post-admission, and where the total amount of tissue damage never exceeds threshold κ_1. In our experiments, $\kappa_1 = 10$ and $\kappa_2 = 0.5$. ϕ_A defines the set of aseptic death trajectories where the pathogen falls below a threshold κ_2 within the first 72-hours, but where the total amount of tissue damage exceeds threshold κ_1. ϕ_S defines the set of septic death trajectories where the pathogen never falls below a threshold κ_2 within the first 72-hours, and where the total amount of tissue damage exceeds threshold κ_1.

Table 1. Confusion Matrix For Outcome Classifier: Our classifier achieves a predictive accuracy of 92.8%.

		Predicted Value		
		Health	**Aseptic Death**	**Septic Death**
Actual Value	**Health**	497	37	16
	Aseptic	12	302	5
	Septic	1	1	129

Using our cohort of 1,000 simulated patients, we estimated $P(\phi_H)$, $P(\phi_A)$, and $P(\phi_S)$. Each patient was then classified in terms of likely outcome by taking the maximum of $P(\phi_H)$, $P(\phi_A)$, and $P(\phi_S)$. Our classifier achieves a predictive accuracy of 92.8% (Table 1). In contrast, the predictive accuracy of a Support Vector Machine, a Neural Network, and Random Forest Classifier on the same task achieved accuracies of 80.8%, 78.8%, and 81.1%, respectively. These classifiers were trained using 10-fold cross validation across a variety of parameter settings. The best results are reported. The input features were the same as for our method. In comparison to our approach, these other algorithms most often misclassified aseptic death as a healthy outcome.

In order to determine the sensitivity of our method to the distribution $P(\Theta)$, we increased the covariance by a factor of ten, thus increasing the uncertainty in the parameters. On this task, our classifier achieves an accuracy of 79.6% (Table 2). In contrast, the SVM, Neural Network, and Random Forest Accuracies drop to 53.8%, 55.4%, and 53.2%, respectively. With the higher uncertainty, these other algorithms often misclassified aseptic death as a healthy outcome, and visa-versa.

Table 2. Confusion Matrix For Outcome Classifier with high covariance $P(\Theta)$: Our classifier achieves a predictive accuracy of 79.6%.

		Predicted Value		
		Health	**Aseptic Death**	**Septic Death**
Actual Value	**Health**	401	91	58
	Aseptic	26	275	18
	Septic	4	7	120

Predicting the Benefit of anti-inflammatory therapy: Next, we consider some more complicated generalized queries and consider the effects of applying an anti-inflammatory therapy. We define the following formulas in BLTL:

- $\phi_{Help} \equiv (\phi_A \vee \phi_S) \wedge (treat \implies \phi_H)$,
- $\phi_{Harm} \equiv \phi_H \wedge (treat \implies (\phi_A \vee \phi_S))$
- $\phi_{Lives} \equiv \phi_H \wedge (treat \implies \phi_H)$
- $\phi_{Dies} \equiv (\phi_A \vee \phi_S) \wedge (treat \implies (\phi_A \vee \phi_S))$

Here, "treat" means we set the value of C_a to a therapeutic level. ϕ_{Help} defines the set of trajectories where the patient would have died without treatment, but will live with the therapy. ϕ_{Harm} defines the set of trajectories where the patient would have lived without treatment, but will die with the therapy. ϕ_{Lives} and ϕ_{Dies} define the set of trajectories

where the patient lives or dies, regardless of the treatment, respectively.

Using our cohort of 1,000 simulated patients, we estimated $P(\phi_{Help})$, $P(\phi_{Harm})$, $P(\phi_{Lives})$, and $P(\phi_{Dies})$. Each patient was then classified in terms of likely outcome by taking the maximum of these values. Our classifier achieves a predictive accuracy of 96.9% (Table 3). In contrast, the predictive accuracy of a Support Vector Machine, a Neural Network, and Random Forest Classifier on the same task achieved accuracies of 67.6%, 66.9%, and 64.9%, respectively. In comparison to our approach, these other algorithms most often misclassified helping the patient with lives, and none of them did well at detecting when the treatment actually harms the patient.

Table 3. Confusion Matrix For Treatment Classifier: Our classifier achieves a predictive accuracy of 96.9%.

		Predicted Value			
		Help	**Harm**	**Lives**	**Dies**
Actual Values	**Help**	272	28	0	0
	Harm	0	400	0	0
	Lives	0	0	150	0
	Dies	0	0	3	147

Once again, in order to determine the sensitivity of our method to the distribution $P(\Theta)$, we increased the covariance by a factor of ten, thus increasing the uncertainty in the parameters. On this task, our classifier achieves an accuracy of 80.6% (Table 4). In contrast, the SVM, Neural Network, and Random Forest Accuracies drop to 53.5 %, 43.0%, and 40.0%, respectively.

Table 4. Confusion Matrix For Treatment Classifier with high covariance $P(\Theta)$: Our classifier achieves a predictive accuracy of 80.6%.

		Predicted Value			
		Help	**Harm**	**Lives**	**Dies**
Actual Values	**Help**	150	150	0	0
	Harm	0	400	0	0
	Lives	0	0	150	0
	Dies	0	2	42	106

5. RELATED WORK

There are a number of algorithms for solving the Probabilistic Model Checking Problem. Methods for computing the exact value of ρ been reported (e.g., [1, 2, 3, 7, 11]) for various classes of models, although not for DBNs. These methods perform an explicit exploration of the state space of the model, which is generally exponential in the number of state variables. This suggests that exact Probabilistic Model Checking algorithms may not scale well to larger systems. Moreover, these methods are not Bayesian, and instead assume fixed values for parameters. Simulation-based methods fall into two broad categories: those that estimate the value of ρ (e.g., [8, 15, 16]), and those that treat the PMC problem as a hypothesis testing problem (e.g., [4, 19, 20, 21]). These methods also assume fixed parameters, and are thus not Bayesian.

Our method is most closely related to a method we recently introduced[9]. There are two key differences between the present method and the one in [9]. First, the method in [9] casts the Probabilistic Model Checking Problem as a compound hypothesis testing problem, while our method is based on estimation. The potential advantage of a hypothesis testing based approach is that it may require fewer samples, especially if the exact value of ρ is not as important as knowing whether it is above some threshold. For example, a treatment decision might be based on whether ρ is greater than 0.5. Deciding whether ρ is greater than (or less than) some threshold is likely to require fewer sampled trajectories than trying to estimate the true value of ρ. The second difference between the present method and [9] is that while [9] does incorporate a prior on ρ, it assumes that the model parameters are fixed and known. In that sense, it is only partially Bayesian. In the context of Systems Biology, we feel that it is more likely that there will be some uncertainty in the model parameters.

6. DISCUSSION AND CONCLUSION

We have introduced the notion of generalized probabilistic queries for Dynamic Bayesian Networks, and presented two algorithms for answering such queries. Generalized queries provide the option of reasoning about sets of trajectories through the DBN. Our algorithms can also be thought of as Bayesian Statistical Model Checking algorithms. We applied one of our algorithms on two tasks relevant to personalized medicine and found that it results in more accuracy

classification than special-purposed classifiers. We are presently evaluating the method on a wider class of DBNs and developing algorithms for performing generalized decoding.

There are a number of potential applications of our method that we have not considered here. For example, one might imagine using the method to decide when to terminate treatment by considering the likelihood that the outcome will remain the same, whether or not a given intervention is discontinued. Such information can be used to reduce health care costs or to minimize adverse side-effects. Another interesting possibility is to use the method to help select patients for enrollments in a clinical trials. The use of modeling for selecting clinical trial cohorts was first suggested by Clermont and co-workers[6]. Here, a pharmaceutical company might elect to enroll only those patients who stand a chance of benefiting from the therapy, thus increasing the odds of being able to demonstrate the efficacy of the drug to regulatory agencies, like the FDA.

Acknowledgments

This research was supported by a U.S. Department of Energy Career Award (DE-FG02-05ER25696), and a Pittsburgh Life-Sciences Greenhouse Young Pioneer Award to C.J.L.

References

1. C. Baier, E.M. Clarke, V. Hartonas-Garmhausen, M.Z. Kwiatkowska, and M. Ryan. Symbolic model checking for probabilistic processes. In *ICALP '97: Proceedings of the 24th International Colloquium on Automata, Languages and Programming*, pages 430–440, London, UK, 1997. Springer-Verlag.
2. C. Baier, B. R. Haverkort, Hermanns H., and J.P. Katoen. Model-checking algorithms for continuous-time Markov Chains. *IEEE Trans. Software Eng.*, 29(6):524–541, 2003.
3. F. Ciesinski and M. Größer. On probabilistic computation tree logic. In *Validation of Stochastic Systems*, volume 2925, pages 147–188. Springer, 2004.
4. E.M. Clarke, J. R. Faeder, C.J. Langmead, L. A. Harris, S.K. Jha, and A. Legay. Statistical model checking in biolab: Applications to the automated analysis of t-cell receptor signaling pathway. In *CMSB*, pages 231–250, 2008.
5. E.M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, Cambridge, MA, 1999.
6. G. Clermont, J. Bartels, R. Kumar, G. Constantine, Y. Vodovotz, and C. Chow. In silico design of clinical trials: A method coming of age. *Crit. Care Med.*, 32(10):2061–2070, 2004.
7. C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *Journal of the ACM*, 42(4):857–907, 1995.
8. R. Grosu and S.A. Smolka. Monte Carlo Model Checking. In *CAV*, pages 271–286, 2005.
9. S.K. Jha, E.M. Clarke, C.J. Langmead, A. Legay, A. Platzer, and P. Zuliani. A bayesian approach to model checking biological systems. *Proc. of The 7th Annual Conference on Computational Methods in Systems Biology (CMSB)*, page *in press*, 2009.
10. S.J. Julier and J.K. Uhlmann. A new extension of the kalman filter to nonlinear systems. *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls*, pages 182–193, 1997.
11. M. Z. Kwiatkowska, G. Norman, and D. Parker. PRISM 2.0: A tool for probabilistic model checking. In *QEST*, pages 322–323. IEEE, 2004.
12. K. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, University of California, Berkeley; Computer Science Division, 2002.
13. A. Pnueli. The temporal logic of programs. In *FOCS*, pages 46–57, 1977.
14. A. Reynolds, J. Rubin, G. Clermont, J. Day, Y. Vodovotz, and B. Ermentrout. A reduced mathematical model of the acute inflammatory response: I. derivation of model and analysis of anti-inflammation. *J Theor Biol*, 242(1):220–236, 2006.
15. K. Sen, M. Viswanathan, and G. Agha. Statistical model checking of black-box probabilistic systems. In *CAV*, LNCS 3114, pages 202–215. Springer, 2004.
16. K. Sen, M. Viswanathan, and G. Agha. On statistical model checking of stochastic systems. In *CAV*, LNCS 3576, pages 266–280, 2005.
17. A. Sitz, U. Schwarz, J. Kurths, and H.U. Voss. Estimation of parameters and unobserved components for nonlinear systems from noisy time series. *Phys Rev E Stat Nonlin Soft Matter Phys.*, 66(1):016210, 2002.
18. Y. Vodovotz, M. Csete, J. Bartels, S. Chang, and G. An. Translational systems biology of inflammation. *PLoS Computational Biology*, 4(4), 2008.
19. H.L.S. Younes, M.Z. Kwiatkowska, G. Norman, and D. Parker. Numerical vs. statistical probabilistic model checking. *STTT*, 8(3):216–228, 2006.
20. H.L.S. Younes and R.G. Simmons. Probabilistic verification of discrete event systems using acceptance sampling. In *CAV*, LNCS 2404, pages 223–235. Springer, 2002.
21. H.L.S. Younes and R.G. Simmons. Statistical probabilistic model checking with a focus on time-bounded properties. *Information and Computation*, 204(9):1368–1409, 2006.
22. L. Zhang, H. Hermanns, and D.N. Jansen. Logic and model checking for hidden Markov models. *Formal techniques for networked and distributed systems (FORTE)*, pages 98–112, 2005.