Using Continuous-Time Bayesian Networks for Standards-Based Diagnostics and Prognostics

Logan Perreault, John Sheppard, Houston King, and Liessman Sturlaugson
Department of Computer Science
Montana State University
357 EPS Building, PO Box 173880
Bozeman, Montana 59717
logan.perreault@msu.montana.edu, john.sheppard@cs.montana.edu, houston.king@msu.montana.edu, listurlaugson@gmail.com

***Abstract*—In this paper we present a proposal for a new prognostic model to be included in a future revision of the IEEE Std 1232-2010 Standard for Artificial Intelligence Exchange and Service Tie to All Test Environments (AI-ESTATE). Specifically, we introduce the continuous time Bayesian network (CTBN) as an alternative to the previously proposed dynamic Bayesian network to provide an additional model for prognostic reasoning. We specify a semantic model capable of representing a CTBN within the standard and discuss the advantages of using such a model for prognosis. As with previous work, we demonstrate the feasibility and necessity of incorporating prognostic capabilities into the standard.**

I. Introduction

Recently, the US Department of Defense has been funding research into how to advance and deploy standards-based products supporting equipment diagnostics as well as prognostics and health management (PHM). As part of the DoD Automatic Test System (ATS) Framework Working Group, several IEEE standards have been developed and are being maintained in support of this initiative. One of the standards—IEEE Std 1232-2010 Standard for Artificial Intelligence Exchange and Service Tie to All Test Environments (AI-ESTATE)—focuses specifically on defining data exchange formats for diagnostic models as well as a service interface for working with diagnostic reasoners.

Previously, we presented work describing the results of developing an AI-ESTATE-conformant reasoner implemented in Java that includes extensions for prognostics [1]. In that work, we described a new tool, the Standards Based Analysis Platform for Predictive Health and Integrated Reasoning Environment (SAPPHIRE$^{\text{TM}}$), that uses an extension of the AI-ESTATE Bayesian network model that incorporates temporal components for predicting future failures. The resulting dynamic Bayesian network (DBN) description included recommendations for how to enhance AI-ESTATE to incorporate these temporal components and associated services. In this paper, we introduce another model for addressing prognostic requirements that has the advantage of relaxing the discrete time assumption of a DBN. This relaxation has the benefit of enabling more realistic temporal predictions as well as providing a cleaner map to reliability-based models that are well recognized in the weapon system support community.

Specifically, in this paper we describe how a continuous-time Bayesian network (CTBN) can be used for fault prognosis. In short, a CTBN is a probabilistic graphical model corresponding to a multivariate Markov process where conditional independence properties are exploited to "factor" the Markov process into manageable chunks. We also describe how the CTBN can be processed through the AI-ESTATE services (including the new services proposed in [1]). We then provide an EXPRESS model of the CTBN and illustrate how it can be defined as a straightforward extension to the DBN model introduced previously.

II. Background

A. AI-ESTATE

AI-ESTATE was created in 1990 as a way to apply artificial intelligence methods to testing and diagnosis tasks. This work was started under Project Authorization Request (PAR) 1232, which granted IEEE Standards Coordinating Committee 20 (SCC20) permission to begin development. Since that time, work has been done to introduce and improve methods for exchanging diagnostic reasoner models and a standardized application programming interface (API) to communicate with diagnostic reasoners. IEEE Std 1232-2010 [2], the most recent version of the standard, defines fault trees, D-matrices, logic models, and Bayesian networks as possible diagnostic reasoner models. The semantics of these four models are described using the EXPRESS information modeling language [3], and have been used to derive eXtensible Markup Language (XML) schemata based on the International Organization for Standardization's (ISO's) Standards for the Exchange of Product model data (STEP) [4].

Previous work from 2008 evaluated the semantic interoperability of AI-ESTATE-conformant diagnostic models [5]. Results showed that the standard allows for effective semantic modeling in information exchange while maintaining a reasonable engineering burden. In 2009, another study was done on reasoner interoperability based on the AI-ESTATE service specification and likewise showed the standard to be effective [6]. These works were later extended in 2013 by adding prognostic extensions to the AI-ESTATE implementation of SAPPHIRE [1]. In particular, a dynamic Bayesian network (DBN) model was created to support inference over a time series. This work further demonstrated the diagnostic capabilities of AI-ESTATE and supported the notion of using this standard for prognostic tasks as well. At the time we published

the SAPPHIRE paper [1], the tool had yet to implement all of the services defined by the AI-ESTATE standard. Since that time, however, the remaining services have been added to make SAPPHIRE fully AI-ESTATE conformant.

B. Bayesian Networks

Both DBNs and CTBNs depend upon Bayesian networks in some way. Naïvely, a full joint probability distribution over a set of random variables exhibits a state space whose size is exponential in the number of random variables. Bayesian networks provide a means to control this complexity through the exploitation of conditional independence between variables. In probability, a random variable X_1 is said to be conditionally independent of another variable X_2 given Y if $P(X_1|Y, X_2) = P(X_1|Y)$. The presence of the variable Y reduces the total complexity of the distribution relative to X_1 and X_2.

Using these independencies, we formalize a Bayesian network as follows: a *Bayesian network*, $\mathcal{B}$, over a set of random variables, $\mathbf{X}$, is a compact and factored representation of a joint probability distribution. The network is given as a directed acyclic graph, $\mathcal{G}$, over nodes $X_i \in \mathbf{X}$. In this representation, each X_i node represents a random variable and directed edges represent dependencies between variables. Associated with each node is a conditional probability distribution, $P(X_i|\mathbf{Pa}(X_i))$, where $\mathbf{Pa}(X_i)$ represents the parents of node X_i. Under this definition, it is immediate that the factored representation can be made explicit as

$$P(\mathbf{X}) = \prod_{X_i \in \mathbf{X}} (X_i|\mathbf{Pa}(X_i)).$$

C. Dynamic Bayesian Networks (DBNs)

A dynamic Bayesian network (DBN) is an extension to Bayesian networks that explicitly models probability distributions over sequential data [7]. A *dynamic Bayesian network* is a tuple $(\mathcal{B}_0, \mathcal{B}_{2T})$: $\mathcal{B}_0$ is a Bayesian network over an initial distribution, $\mathbf{X}_0$, and $\mathcal{B}_{2T}$ is a Bayesian network that provides a conditional transition model from $\mathbf{X}_i$ to $\mathbf{X}_{i+1}$. In a DBN, $\mathcal{B}_0$ defines the probability distribution over the initial time step or 'slice' of the sequence being modeled, while $\mathcal{B}_{2T}$ defines the distribution between consecutive slices of the sequence. Under this representation, intra-slice arcs connect variables within a slice, and inter-slice arcs connect variables in one slice to the next.

D. CTBNs

A continuous-time Bayesian network is a factored representation of continuous-time Markov processes [8]. A *continuous time Markov process*, X with state space $\{x_1, x_2, ..., x_N\}$ is a random variable, which is defined as an initial probability distribution over these states, P_0^X, and a Markovian transition model (usually represented as an intensity matrix, $\mathbf{Q}_X$) of the form:

$$\mathbf{Q}_X = \begin{bmatrix} -q_1 & q_{12} & \cdots & q_{1N} \\ q_{21} & -q_2 & \cdots & q_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ q_{N1} & q_{N2} & \cdots & -q_N \end{bmatrix},$$

with $q_i = \sum_{i \neq j} q_{ij}$ and $q_i, q_{ij} \geq 0$. Each q_{ij} is a parameter that represents the instantaneous rate at which X may transition from state x_i to x_j. Each q_i represents the instantaneous rate at which X may leave x_i for any other state. From this, the probability density function and cumulative distribution for $X(t)$ staying equal to x_i are $f(t) = q_i \exp(-q_i t)$ and $F(t) = 1 - \exp(-q_i t)$ respectively with t being the amount of time spent in state x_i, making the probability of remaining in a state decrease exponentially with respect to time.

The expected sojourn time for state x_i is $1/q_i$. Each row is constrained to sum to zero, meaning that the transition probabilities from state x_i can be calculated as $\theta_{i,j} = q_{ij}/q_i$, $\forall j,\ i \neq j$. Because the sojourn time uses the exponential distribution, which is "memory-less," the Markov process model exhibits the Markov property, namely, that all future states of the process are independent of all past states of the process given its present state. In other words,

$$P(X(t + \Delta t)|X(t), X(s)) = P(X(t + \Delta t)|X(t))$$

for $0 < s < t < \infty$. Note that the model does not specify a particular time unit for the sojourn times. The modeler is responsible for choosing an appropriate time-scale (minutes, hours, days, etc.) for the specific application. The parameters are then set and interpreted accordingly.

With this, a continuous-time Bayesian network can be defined. Let $\mathbf{X}$ be a set of continuous-time Markov processes, $X_1, \ldots, X_n$, each of which can take on a discrete set of values. A *continuous time Bayesian network*, $\mathcal{N}$, over $\mathbf{X}$ consists of two components. First, an initial distribution $P_{\mathbf{X}}^0$, which can be specified as a Bayesian network $\mathcal{B}$ over $\mathbf{X}$. Second, a continuous transition model, specified as a directed graph $\mathcal{G}$ whose nodes are $X_i \in \mathbf{X}$ and a set of conditional intensity matrices, $\mathbf{Q}_{X_i|\mathbf{Pa}(X_i)}$. For each X_i, there is a corresponding conditional intensity matrix for every combination of states that $\mathbf{Pa}(X_i)$ can take on. Each of these matrices are continuous time Markov processes representative of the behavior of X_i when $\mathbf{Pa}(X_i)$ are in a particular state combination.

CTBNs have found use in wide variety of temporal applications, including the following:

- Robot monitoring [9],
- Modeling server farm failures [10],
- Modeling social network dynamics [11],
- Modeling sensor networks [12],
- Building intrusion detection systems [13], [14],
- Predicting the trajectory of moving objects [15],
- Diagnosing cardiogenic heart failure and anticipating its likely evolution [16].

One of the key concepts behind the CTBN is the use of conditional intensity matrices. That is, the dynamics of a child node can change, depending on the states of the parent nodes. Figure 1 shows an example two-node CTBN. The XY node is a traditional Markov process, while the ABC node is a conditional Markov process in which the conditionally intensity matrices are different given the different states of XY.

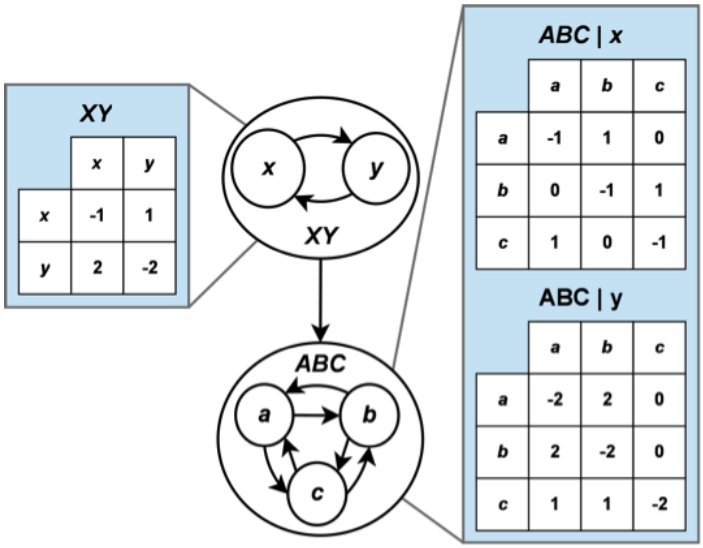

Fig. 1: Example Two-Node CTBN

E. CTBNs versus DBNs

As previously described, DBNs directly model time-sliced domains; however, these models have several potential failings. Consider a system where the health state of the components of the system change at differing rates (which is highly likely). For example, a particular capacitor in a circuit may fail ten times more often than any other component. To model such a system for diagnosis and prognosis effectively with a DBN, the estimates of the states of the system must be propagated at the finest rate of evolution present in the system, increasing the computational cost of reasoning over such a system. Furthermore, consider a system where observations are made at irregularly spaced intervals, such as occurs with unscheduled maintenance and testing or onboard monitoring at different sample rates. To apply such evidence to a DBN, the model may need to be sliced either incredibly finely, or a discretization scheme may need to be applied, resulting in a of loss of information from the collected evidence and observations.

Continuous-time Bayesian networks, as their name suggests, bypass the discretely-sliced structure inherent in DBNs, while maintaining an analogue of the factored representation present in Bayesian Networks. These networks inherently model the temporal dynamics of a system allowing for inference at arbitrary points in the future and allowing evidence to be propagated at arbitrary points through time. This enhanced temporal capability motivates the use of CTBNs for diagnostics and prognostics.

III. AI-ESTATE CTBN Model

Similar to [1], in this paper we propose an extension to the current AI-ESTATE standard to use another reasoner model. Specifically, we introduce the CTBN model that may be more suited for particular prognostics tasks than a DBN. We present a new EXPRESS model for modeling CTBNs and describe how the Reasoner Manipulation Services can interface with such a model for the purposes of diagnostics and prognostics.

A. EXPRESS Model

Similar to our description above, recent applications of artificial intelligence to fault prognosis have involved using Bayesian inference with temporal models to drive the prognosis. The AI_ESTATE_CTBN captures information necessary for creating prognostic continuous time Bayesian networks. Assumptions made with this model include that Markov processes corresponding to tests can only depend on diagnosis variables. In addition, the intensity matrices are to be fully explicated, assuming an exponential distribution, and array position in the matrix rows corresponds to array position in the dependence array. Note that the exponential distribution assumption can be relaxed using phase type distributions [17].

A variety of new entities and model constraints were necessary to specify the CTBN model, although only a subset of the more important components is discussed here. The complete EXPRESS-G model for a CTBN is shown in Figure 2.

Two major components required for CTBNs to perform diagnostic or prognostic tasks are the CTBNDiagnosis and CTBNTest entities. The CTBNDiagnosis entity corresponds to an individual diagnosis within the CTBN model. Since CTBNDiagnosis is a subtype of Diagnosis in the original AI-ESTATE Common Element Model, it inherits all of the characteristics of the Diagnosis entity. Similar to the AI-ESTATE Bayesian network model, a CTBNDiagnosis can be subtyped as a CTBNFault or a CTBNFailure. The CTBNTest entity corresponds to an individual test within the CTBN model. Here again, CTBNTest is a subtype of the AI-ESTATE Test and therefore inherits all the characteristics of a diagnostic test.

Within the original AI-ESTATE model, tests and diagnoses indicate the corresponding outcomes are optional; however, they are only optional if they do not occur at the leaves of the model. In a CTBN, this does not make sense since intensity matrices will be defined to indicate how states evolve in time. The states correspond to these outcomes; therefore, for both

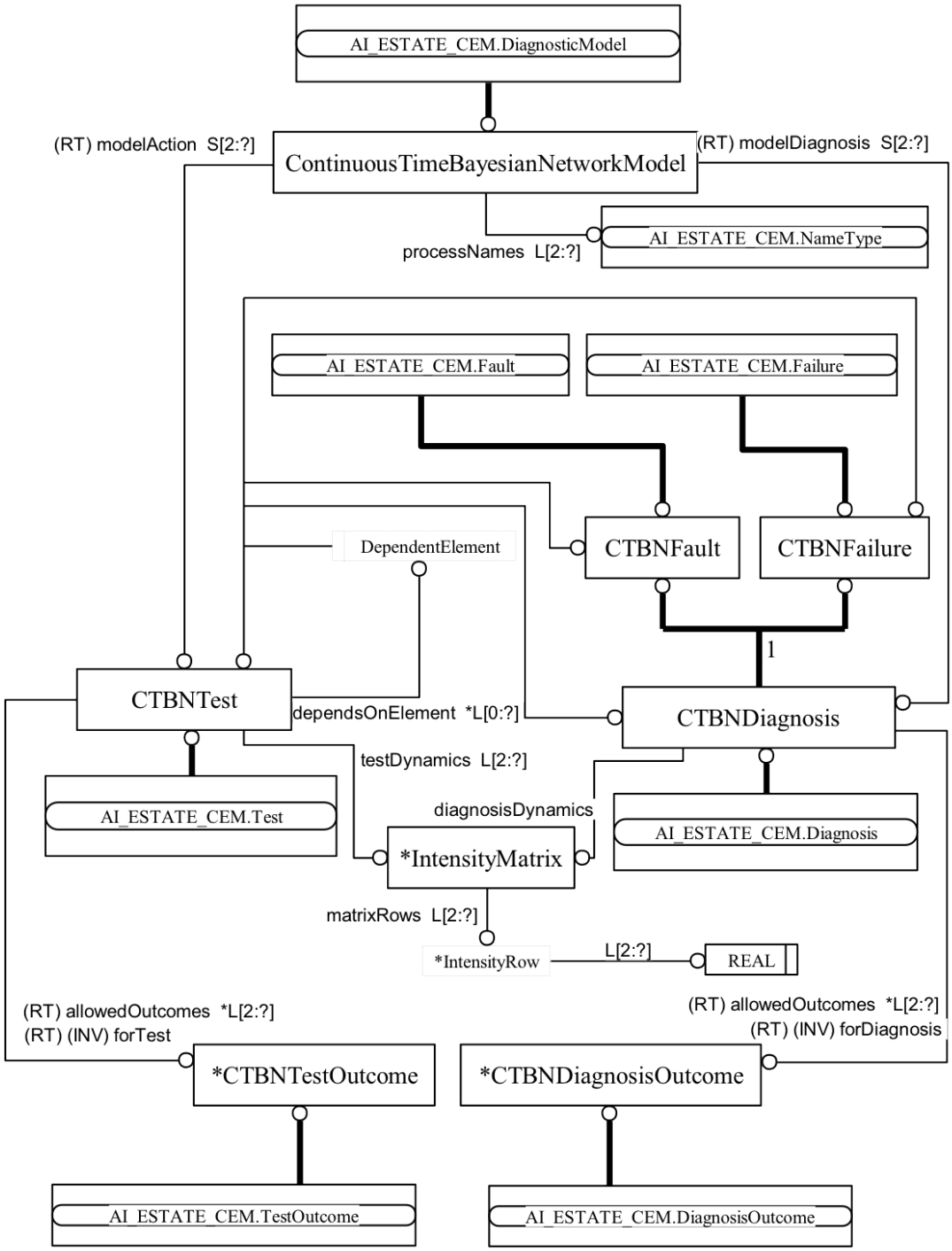

Fig. 2: AI_ESTATE_CTBN EXPRESS-G diagram

CTBNTest and CTBNDiagnosis (and its subtypes), outcomes are required at all levels.

Another crucial component for the CTBN is the IntensityMatrix entity. Figure 3 shows the EXPRESS code for an intensity matrix as encapsulated in the CTBN model. The IntensityMatrix entity directly represents any $\mathbf{Q}_x$ within a CTBN. This entity specifies the set of rows in the intensity matrix of the Markov process, thereby defining the transition probabilities between states. It is defined as a list of lists to impose order on the entries in the matrix. This list of lists has the ability to represent an unbounded size ordered array. To ensure that the matrix is square, a model constraint is imposed

```
ENTITY IntensityMatrix;
  matrixRows :
    LIST [2:?] OF IntensityRow;
WHERE
  isSquare          :
     squareCheck(SELF.matrixRows);
  properDiagonal    :
    nonPosDiag(SELF.matrixRows);
  properOffDiagonal :
    nonNegOffDiag(SELF.matrixRows);
END_ENTITY;
```

Fig. 3: IntensityMatrix Entity EXPRESS Specification

that verifies that each sublist is the same length as the outer list. Additional model constraints are also applied that ensure the values contained within the matrix form a valid representation of a CTBN intensity matrix. In particular, rules are in place to ensure that diagonal entries are negative, off-diagonal entries are positive, and rows sum to zero. As shown in Figure 4. the nonNegOffDiag constraint iterates over all non-diagonal cells, raising a flag if a negative cell is encountered. The non-positive diagonal constraint (not shown) is implemented similarly. Figure 5 shows that the row sum constraint simply sums the given row and raises a flag if the result is not zero.

A notable addition to the CTBN EXPRESS model is the inclusion of the processNames attribute. The first benefit of such an attribute is obvious: there is a central location that is used to name all Markov processes (tests and diagnoses) within the model. The second more subtle benefit comes from processNames being a list, which imposes an ordering upon the names.

Within a CTBNTest there is one conditional Intensity-Matrix per combination of values that the parent CTBN-Diagnoses can assume. From this, there must be a way to identify the matrix of interest based upon the current state of the parents. By including the processNames list, a standardized ordering is imposed upon the testDynamics of all CTBNTests: the matrices should be stored by iterating through the states of the parents, with the parents aligned according to the processNames list. For example, consider a CTBN with processNames = [A,B,C,D]. Let A-C be diagnoses and let D be a test which depends only on A and C. If A has 2 possible states and C has 3 possible states then the IntensityMatrices in testDynamics must be ordered as: $[Q_{A_0B_0}, Q_{A_1B_1}, Q_{A_0B_2}, Q_{A_1B_0}, Q_{A_0B_1}, Q_{A_1B_2},]$. While this constraint is not included explicitly within the EXPRESS model, it will be programmatically enforced.

It should be noted that the ordering of conditional probability tables in the Bayesian Network Model currently has no standardized way to describe and enforce an ordering on the structure of the tables, leading to potential ambiguity when exchanging data models using the standard. The addition of a similar 'variableNames' attribute will be suggested for inclusion in a future version of the standard to address this shortcoming.

```
FUNCTION nonNegOffDiag
  (matrix:LIST OF IntensityRow):BOOLEAN;
  LOCAL
    i,j : INTEGER;
    flag: BOOLEAN := TRUE;
    cell : REAL;
    row : IntensityRow;
  END_LOCAL;

  REPEAT i := LOINDEX(matrix)
              TO HIINDEX(matrix);
    REPEAT j := LOINDEX(matrix)
                TO HIINDEX(matrix);
      IF (i = j) THEN
        row := matrix[i];
        cell := row[j];
        IF (cell < 0) THEN
          flag := FALSE;
        END_IF;
      END_IF;
    END_REPEAT;
  END_REPEAT;
  RETURN(flag);
END_FUNCTION;
```

Fig. 4: EXPRESS Non-Negative Off-Diagonal Constraint Function

```
FUNCTION rowSum
  (row:LIST of REAL):REAL;
  LOCAL
    sum : REAL := 0.0;
    i : INTEGER;
  END_LOCAL;

  REPEAT i := LOINDEX(row) TO HIINDEX(row);
    sum := sum + row[i];
  END_REPEAT;
  RETURN(sum);
END_FUNCTION;
```

Fig. 5: EXPRESS Row Sum Constraint Function

B. AI-ESTATE Service Extensions

The AI-ESTATE extensions added for DBN-based prognostic reasoning in our previous work are nearly identical to the extensions required for CTBN-based reasoning [1]. The most notable of these are the changes to the Reasoner Manipulation Services. These services will be incorporated into a future CTBN-based AI-ESTATE-conformant tool with the expectation that they will eventually be adopted into the standard. For prognostic use with a CTBN, the *applyActions* service must be modified to accept time series based data, where test results may only be valid for particular intervals. To support the inference process, the *getDiagnosticResults* service must be modified to include an additional parameter, specifying the future timestamp at which to reason about the diagnostic state of the system. Furthermore, several CTBN

inference algorithms support queries regarding the estimated time to state transitions, which could inform the introduction of a new service, *getTimetoFailure*. This function could either be passed a particular CTBNFailure to calculate, or if omitted, the service could return the first failure to occur.

C. CTBN Classification and Inference

One of the reasoning tasks that must be accomplished with an AI-ESTATE conformant tool is the computation of diagnoses and, with our extensions from [1], prognoses. With the structure imposed by the EXPRESS model, this computation becomes strongly analogous to the traditional idea of classification from machine learning, where CTBNDiagnoses represent a class variable to be inferred and where CTBNTests represent feature variables. For our purposes, we must classify the diagnosis variables in a model, that is, determine the most likely diagnosis (or future prognosis) given a variety of test information.

Classification in CTBNs has been subject to past research. In 2012, two network structures where introduced: the continuous-time naive Bayes classifier and the continuous-time tree augmented naive Bayes classifier. [18]. This work was extended in 2014, with the introduction of the open CTBNCToolkit [19]. This open toolkit implements structure and parameter learning for CTBN classifiers, as well as a specialized inference algorithm which performs the task of classification. Unfortunately, this state-of-the-art classifier carries a strong requirement: complete knowledge of all features (CTBNTests) over the entire history of the object of interest. A centralized and standardized repository of testing information is often, at best, a lofty goal for many real-world diagnostic and prognostic systems. Whether or not a more efficient algorithm can be developed for the task of classification in the absence of complete feature information is an open research question.

In light of this, traditional inference algorithms will be our preferred method for performing classification. Exact inference in CTBNs has been shown to be NP-Hard, even when the initial distribution is known. [20] A wide variety of approximate inference algorithms for CTBNs have been introduced, including the following:

- Importance sampling [20],
- Rejection sampling [21],
- Gibbs sampling [22],
- Metropolis-Hastings [23],
- Expectation propagation [24],
- Mean-field variational approximation [25],
- Belief propagation [26],
- Particle filtering [27].

The specific algorithm to be used as part of this work is, as of yet, undecided. However, it should be noted that the AI-ESTATE standard permits the specific choice of an engine to be distinct from the CTBN model and interface, allowing the selection of an algorithm to be made independent of the representation of the CTBN.

IV. Software Implementation

As part of a Navy-funded SBIR, an existing implementation of a CTBN has been converted from C# to Java. This was done to bring the code in-line with the SAPPHIRE project discussed in previous work [1]. In addition to the CTBN code, work has been done to integrate data conforming to other standards into the project, including IEEE Std 1636.1-2013 Software Interface for Maintenance Information Collection and Analysis (SIMICA): Exchanging Test Results and Session Information via the Extensible Markup Language (XML) (Test Results) [28] and IEEE Std 1636.2-2010 Software Interface for Maintenance Information Collection and Analysis (SIMICA): Maintenance Action Information (MAI) [29]. We have implemented code to parse and perform schema validation for Test Results and MAI data. Continued efforts are being made to use information from these files to set evidence and perform model learning for the CTBN via the AI-ESTATE services.

V. Future Work

We plan to continue work on the standards-based CTBN implementation. Although functional, the CTBN code is not yet standards-conformant (even assuming all of our proposals would be accepted). We plan to make the necessary modifications so that interactions with the model will be done via the AI-ESTATE services. We also plan to store representations of CTBN models in the form of the AI_ESTATE_CTBN model proposed in this paper. The format would be based on an XML schema derived from the EXPRESS model, similar to the XML schemata currently being used by AI-ESTATE. This would provide a working demonstration of an implemented AI_ESTATE_CTBN that may support the decision to adopt it.

Additionally, the CTBN model itself will eventually need to be imported from and exported to a standards-conformant XML file. In the case of importing, the XML file will need to be validated against the schema, imported, and finally the model will need to be programmatically validated against the semantics of the defined EXPRESS model. In the case of exporting, the reverse process occurs; first the model is validated, then exported, and finally the resulting XML file is validated against the schema.

Often, CTBN applications described in the literature use the notion of a trajectory. A trajectory is a mapping of state transitions over time. For example, a component in a diagnostic system could transition from an initial state of GOOD to DEGRADED at some time t, and then at a later time, s transition from DEGRADED to BAD. These state changes could be encapsulated in a trajectory such as $[(0, \mathrm{GOOD}), (t, \mathrm{DEGRADED}), (s, \mathrm{BAD})]$. It would be possible to encapsulate such structures as additional EXPRESS / XML objects within the AI-ESTATE standard. Once encapsulated in this way, services could be designed specifically to handle trajectories as a special kind of evidence specific to prognostic reasoning, where a series of state transitions over time can inform the reasoning process. Given constraints on the times at which transitions could occur, trajectories could serve the same function within the context of DBNs as well as CTBNs.

Finally, upon the completion of the CTBN project, we hope to compare its performance to that of the DBN. This

would provide benchmarks for a variety of prognosis tasks for two independent, standards-conformant reasoner models. This information may elucidate which tasks are biased toward CTBNs and those that work better with DBNs. Since both models will be accessible via AI-ESTATE, client software will easily be able to change the reasoner model to better suit the problem. Furthermore, this provides added options and capabilities for satisfying the prognostic data (PROD) and prognostic service (PROS) elements of the DoD ATS Framework.

VI. Conclusion

This work primarily focuses on the introduction of a new CTBN reasoner model as a possible extension to IEEE Std. 1232-2010 (AI-ESTATE). Like previous work involving DBNs, the addition of a CTBN model provides AI-ESTATE with the ability to perform prognosis tasks in addition to diagnosis. The primary contribution that CTBNs offer is the ability to perform inference over a continuous time interval, as opposed to using discrete time slices. This is especially useful when evidence is provided at inconsistent time intervals and avoids having to unroll a network using an unnecessarily small granularity of time.

Work relating to the CTBN model further supports the notion that AI-ESTATE can and should be used for prognosis. Models such as the CTBN presented in this paper and DBN presented in [1] are currently being used successfully for prognostic tasks. Standardization efforts should strive to meet the requirements for these applications. This paper shows the feasibility of creating EXPRESS-based models for prognostic reasoners, further demonstrating the benefits of incorporating prognostic capabilities into the standard.

Acknowledgments

This project was supported as an SBIR under US Navy contract N132-091-0713. We thank Kihoon Choi and Deepak Haste of Qualtech Systems Inc. for their collaboration with this SBIR.

References

[1] L. Sturlaugson, N. Fortier, P. Donnelly, and J. W. Sheppard, "Implementing AI-ESTATE with prognostic extensions in Java," in *AUTOTESTCON, 2013 IEEE*. IEEE, 2013, pp. 1–8.

[2] IEEE Std. 1232-2010, *IEEE Standard for Artificial Intelligence Exchange and Service Tie to All Test Environments (AI-ESTATE)*, Piscataway, NJ: IEEE Standards Association Press, 2010.

[3] ISO 10303-11:1994, *Industrial Automation Systems-Product Data Representation and Exchange-Part 11: The EXPRESS Language Reference Manual.*, Geneva, Switzerland: The International Organization for Standardization, 1994.

[4] ISO 10303-28:2007, *Industrial Automation Systems-Product Data Representation and Exchange-Part 28: XML Representation of EXPRESS Schemas and Data Using XML Schemas*, Geneva, Switzerland: The International Organization for Standardization, 1994.

[5] J. W. Sheppard, S. G. Butcher, P. J. Donnelly, and B. R. Mitchell, "Demonstrating semantic interoperability of diagnostic models via AI-ESTATE," in *Aerospace Conference, 2009 IEEE*. IEEE, 2009, pp. 1–13.

[6] J. W. Sheppard, S. G. Butcher, and P. J. Donnelly, "Demonstrating semantic interoperability of diagnostic reasoners via AI-ESTATE," in *Aerospace Conference, 2010 IEEE*. IEEE, 2010, pp. 1–10.

[7] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. MIT press, 2009.

[8] U. Nodelman, C. R. Shelton, and D. Koller, "Continuous time Bayesian networks," in *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc., 2002, pp. 378–387.

[9] B. Ng, A. Pfeffer, and R. Dearden, "Continuous time particle filtering," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2005, pp. 1360–1365.

[10] R. Herbrich, T. Graepel, and B. Murphy, "Structure from failure," in *Proceedings of the Second USENIX Workshop on Tackling Computer Systems Problems with Machine Learning Techniques*. USENIX Association, 2007, pp. 1–6.

[11] Y. Fan and C. Shelton, "Learning continuous-time social network dynamics," in *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI)*. AUAI Press, 2009, pp. 161–168.

[12] D. Shi, X. Tang, and J. You, "An intelligent system based on adaptive CTBN for uncertainty," *Intelligent Automation & Soft Computing*, vol. 16, pp. 337–351, 2010.

[13] J. Xu and C. R. Shelton, "Continuous time Bayesian networks for host level network intrusion detection," in *Machine Learning and Knowledge Discovery in Databases*. Springer, 2008, pp. 613–627.

[14] J. Xu and C. Shelton, "Intrustion detection using continuous time Bayesian network," *Journal of Artificial Intelligence Research (JAIR)*, vol. 39, pp. 745–774, 2010.

[15] S. Qiao, C. Tang, H. Jin, T. Long, S. Dai, and Y. Ku, "PutMode: prediction of uncertain trajectories in moving objects databases," *Applied Intelligence*, vol. 33, pp. 370–386, 2010.

[16] E. Gatti, D. Luciani, and F. Stella, "A continuous time Bayesian network model for cargiogenic heart failure," *Flexible Services and Manufacturing Journal*, pp. 1–20, 2011.

[17] U. Nodelman, C. R. Shelton, and D. Koller, "Expectation maximization and complex duration distributions for continuous time Bayesian networks," in *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, 2005.

[18] F. Stella and Y. Amer, "Continuous time Bayesian network classifiers," *Journal of Biomedical Informatics*, vol. 45, no. 6, pp. 1108–1119, 2012.

[19] D. Codecasa and F. Stella, "Ctbnctoolkit: Continuous time bayesian network classifier toolkit," *arXiv preprint arXiv:1404.4893*, 2014.

[20] Y. Fan, J. Xu, and C. R. Shelton, "Importance sampling for continuous time bayesian networks," *The Journal of Machine Learning Research*, vol. 11, pp. 2115–2140, 2010.

[21] J. C. Weiss, S. Natarajan, and C. D. Page Jr, "Learning when to reject an importance sample." in *AAAI (Late-Breaking Developments)*, 2013.

[22] T. El-Hay, N. Friedman, and R. Kupferman, "Gibbs sampling in factorized continuous-time markov processes," *arXiv preprint arXiv:1206.3251*, 2012.

[23] B. Miasojedow, W. Niemiro, J. Noble, and K. Opalski, "Metropolis-type algorithms for continuous time bayesian networks," *arXiv preprint arXiv:1403.4035*, 2014.

[24] U. Nodelman, C. R. Shelton, and D. Koller, "Expectation maximization and complex duration distributions for continuous time Bayesian networks," *arXiv preprint arXiv:1207.1402*, 2012.

[25] I. Cohn, T. El-Hay, N. Friedman, and R. Kupferman, "Mean field variational approximation for continuous-time bayesian networks," *The Journal of Machine Learning Research*, vol. 11, pp. 2745–2783, 2010.

[26] T. El-Hay, I. Cohn, N. Friedman, and R. Kupferman, "Continuous-time belief propagation," 2010.

[27] B. Ng, A. Pfeffer, and R. Dearden, "Continuous time particle filtering," in *International Joint Conference on Artificial Intelligence*, vol. 19. LAWRENCE ERLBAUM ASSOCIATES LTD, 2005, p. 1360.

[28] IEEE Std. 1636.1-2013, *Software Interface for Maintenance Information Collection and Analysis (SIMICA): Exchanging Test Results and Session Information via the eXtensible Markup Language (XML)*, Piscataway, NJ: IEEE Standards Association Press, 2013.

[29] IEEE Std. 1636.2-2010, *Software Interface for Maintenance Information Collection and Analysis (SIMICA): Exchanging Maintenance Action Information via the Extensible Markup Language (XML)*, Piscataway, NJ: IEEE Standards Association Press, 2010.