Probabilistic Modeling of Failure Dependencies
Using Markov Logic Networks

Shalini Ghosh∗ Wilfried Steiner† Grit Denker∗ Patrick Lincoln∗
∗Computer Science Laboratory, SRI International, Menlo Park, CA 94025.
†TTTech Computertechnik AG, Chip IP Design, A-1040 Vienna, Austria.

Abstract—We present a methodology for the probabilistic modeling of failure dependencies in large, complex systems using Markov Logic Networks (MLNs), a state-of-the-art probabilistic relational modeling technique in machine learning. We illustrate this modeling methodology on example system architectures, and show how the the Probabilistic Consistency Engine (PCE) tool can create and analyze failure-dependency models. We compare MLN-based analysis with analytical symbolic analysis to validate our approach. The latter method yields bounds on the expected system behaviors for different component-failure probabilities, but it requires closed-form representations and is therefore often an impractical approach for complex system analysis. The MLN-based method facilitates techniques of early design analysis (e.g., probabilistic sensitivity analysis). We analyze two examples — a portion of the Time-Triggered Ethernet (TTEthernet) communication platform used in space, and an architecture based on Honeywell’s Cabin Air Compressor (CAC) — that highlight the value of the MLN-based approach for analyzing failure dependencies in complex cyber-physical systems.

I. INTRODUCTION

Modern engineering methods enable constructing reliable, complex systems consisting of tens to thousands of self-contained components, each of which is possibly composed of many sub-components. In some cases the components of these complex systems can each exhibit remarkably low individual component-failure rates of $10^{-6}$ per hour, and the design goal of the overall system-failure rate is $10^{-9}$ per hour or lower. We seek an approach to analyze interdependencies between component failures as they propagate through a system, potentially leading to overall system failure. We consider probabilistic sensitivity analysis techniques for assessing the relationship between component and system failures, which scale reasonably well with system complexity and enable designers to experiment with system parameters early in the design process.

Classical approaches to testing such complex systems for extreme reliability rates can be prohibitively expensive. Many past approaches to reliability analysis make simplifying assumptions (e.g., all component fault arrival rates are independent and identically distributed (i.i.d.)). However, in many real-world situations, a fault in one component may influence the arrival of faults in another component. For example, if one of several compressors fails in an airplane, the load on the other compressors generally increases, leading to an increased probability that the remaining compressors may fail. Fault tree analysis captures the dependencies in failure propagation, but does not consider probabilistic failure dependencies. Even when probabilistic models of failures are considered (e.g., using stochastic Petri nets), the models are fully grounded versions of the system being analyzed, which can limit the scale of systems that can be considered.

We propose modeling and analyzing failure dependencies in complex systems by using Markov Logic Networks. A Markov Logic Network (MLN) [1] is a statistical first-order relational model that combines the powerful formalism of first-order logic with probabilities, using weighted rules. The probabilistic relational framework enables the modeling of stochastic failure interactions, while the first-order logic framework enables using a compact representation for capturing the repetitive structure in failure interactions between the different components of the modeled system.

In this paper, we use an MLN-based modeling and inference tool called the Probabilistic Consistency Engine (PCE) [2], [3] to model failure propagation in different fault-tolerant architectures. PCE offers the advantage of a symbolic first-order specification language, enabling high-level abstract representation of parametric system architectures and properties, even of complex systems with many components. PCE also enables efficient statistical inference in large data spaces, based on sampling techniques. The combination of these properties makes PCE well suited to reasoning about failure dependencies, even in complex, highly reliable systems. MLN-based tools like PCE enable a kind of putative probabilistic property testing, effectively implementing spot checking of critical properties — this can be done in concrete specific instances as well as at an abstract level, where a single symbolic computation can represent millions of possible concrete states. The logical basis of PCE and similar tools effectively enables compositional reasoning at multiple levels of abstraction. The core contribution of this paper is a new scalable approach for analyzing failure dependencies in a system at different levels of abstraction, especially for complex cyber-physical systems.

To our knowledge, a first-order probabilistic relational modeling paradigm like MLN has not been used before in fault analysis for complex systems. In this paper, we develop a novel fault modeling technique using MLNs, and demonstrate its benefits using an extension to TTEthernet [4] and an architecture based on Honeywell’s Cabin Air Compressor as case studies. We validate our methodology by comparing the
inference results obtained from the fault models in PCE\(^1\) to analytical results that we get from applying classical probability theory. For simple architectures we can build the complete analytical model for system-failure probability, because the interactions are relatively easy to model. However, in general, it could be difficult for a system engineer to write the comprehensive set of analytical equations for a system, but much easier to model the interactions in complex systems through probabilistic (weighted) rules in compact first-order logic — this is the main advantage of using PCE for failure dependency modeling.

II. BACKGROUND

A. Markov Logic Networks

A Markov Logic Network (MLN) is a probabilistic relational model where all random variables are Boolean and all functions are Boolean formulas in first-order logic. The formulas in the MLN have weights that are associated with their probabilities — given a knowledge base (conjunction of first-order formulas), the weights on the formulas are used to compute the associated probability of the model in the rule knowledge base.

MLNs enable model representation in terms of weighted first-order formulas, which is much more compact than representing a model in terms of propositional formulas, as is done in Markov networks or Bayesian networks [1]. One can compute the marginal probability of a given formula \(F\) in an MLN as the probability aggregate over the models where \(F\) evaluates to true. In typical use, an MLN is used to infer the marginal probability distributions for the output random variables and formulas based on the distribution for the input random variables, over the space of models defined by the formulas and their corresponding probabilities. The MCSAT inference algorithm [5] for MLNs computes these marginal probabilities by efficiently averaging the probabilities over a sequence of models.

B. Priors and Marginals

In probability theory, the conditional and marginal probabilities of two random variables \(A\) and \(B\) are related as:

\[
P(B) = \sum_a P(B|A = a).P(A = a), \quad \text{where} \quad P(A = a) \quad \text{is the prior probability of} \quad A \quad \text{taking the value} \quad a, \quad P(B|A = a) \quad \text{is the conditional probability of} \quad B \quad \text{given} \quad A = a, \quad \text{and} \quad P(B) \quad \text{is the marginal probability of} \quad B.
\]

In probabilistic models like MLNs, one has a joint probability distribution over random variables \(X = \{X_1, \ldots, X_n\}\), from which one would like to compute marginal probabilities \(P(X_i = x_i)\). This is usually done by computing the conditional probabilities of \(P(X_i|X - X_i)\), typically using Markov Chain Monte Carlo (MCMC) sampling. For computing the marginal probability estimates in MLNs using sampling-based inference techniques, one typically sets prior probabilities based on domain knowledge.

\(^1\)A sample model, explained later in the paper, is available at: http://www.srlsi.com/users/shalini/prdc2013/network-3-hl-seq.mln

C. Probabilistic Consistency Engine (PCE)

SRI’s PCE [2], [3] is a tool that does efficient inference with probabilistic first-order rules, using the framework of MLNs. PCE uses multi-sorted first-order logic for describing a network of formulas and weights to build probabilistic relational models. PCE provides a language for representing MLNs and implements an optimized version of the MCSAT algorithm of Poon and Domingos [5] for probabilistic inference, to compute marginal probabilities of formulas.

MCSAT is a Markov Chain Monte Carlo (MCMC) sampling algorithm, which uses a combination of simulated annealing, SampleSAT and WalkSAT algorithms to estimate marginal probabilities. The WalkSAT algorithm is used to build the initial model. The SampleSAT and simulated annealing algorithms are used in each subsequent iteration – SampleSAT constructs a random model of the chosen subset of clauses, and interleaves simulated annealing steps to converge on an assignment that satisfies all the chosen clauses. Finally, MCSAT inference averages the probabilities over the sequence of random models generate by sampling, to compute the marginal probability.

Various probabilistic inference problems can be represented as MLNs in the form of facts and weighted or unweighted rules. The weight of a model is given by the weight of the formulas that hold in the model, and the probability of a formula is the normalized sum of the weights of the models in which the formula holds. PCE outputs the marginal probabilities for the atomic formulas and any query formulas. PCE is currently used in different application areas, e.g., machine reading (automatic extraction of entities and relations from natural language text) [2], and health informatics [3].

III. METHODOLOGY OF FAILURE DEPENDENCY MODELING

Our methodology suggests two levels of abstraction for system design. In the first stage, called “sensitivity analysis” (I), the system designer studies design tradeoffs between different architectures that represent specific design choices, using our mechanism of probabilistic sensitivity analysis that uses PCE for failure-dependency modeling. After the system designer chooses a particular design specification or a set of candidate solutions at this level of analysis, she performs “fine-grained analysis” (II) of the chosen architecture, by using approaches like fault-tree analysis.

Methodology: Below is a summary of our proposed methodology of failure-dependency modeling:

1.1. Given a complex system architecture and requirements specification, the system and requirements are modeled in a MLN tool such as PCE, where interactions between failures in different system components are specified compactly through weighted first-order logic equations.

1.2. The PCE models for failure dependencies are specified at multiple levels of modeling abstraction (e.g., the designer models component-level failures and more detailed sub-component-level failures).

1.3. While specifying the PCE models, the designer tests the models during design iteration by spot checking — she can
inject faults into the PCE model and get estimates of failure probabilities for the intermediate model, which corresponds to a simplified version of the whole system. Thus, the designer can incrementally check that the model is behaving as expected. The injected component failures may represent component-failure data from the application domain.

I.4. The designer performs sensitivity analysis to test alternative architectures and to obtain reasonably accurate estimates of the system-failure probabilities. Sensitivity to environmental and system conditions such as high loads and component failure probabilities are plotted, enabling a designer to look for sweet spots in design trade offs.

II. Once the designer settles on a specific system design, she performs fine-grained analysis of the chosen system by doing one or more of the following:

(a) Fine-grained simulation or fault-tree analysis of the chosen system design to get more precise failure estimates,
(b) Exhaustive symbolic analysis, if feasible, or
(c) Formal verification of the necessary system properties of the chosen architecture, by using a mechanical proof assistant.

Through our analysis and experiments in this paper, we demonstrate that on some standard architectures, for which it is possible to analytically enumerate failure dependencies, the corresponding PCE models give reasonably accurate estimates of system failure probabilities compared to the analytical result. The differences with the corresponding analytical approaches are within stochastic variation due to sampling. For large complex networks, as typically found in aerospace, vehicles and other real-world applications, such analytical derivation can become infeasible for the system designer. In such cases, having this PCE-based simulation approach could prove to be an invaluable design-aid tool, especially during the initial design stages.

Often, precise estimates are not critical at a particular stage of the design flow, because the reasonably accurate estimates provided by PCE are good enough for making high-level design choices at that stage. This is the case for the failure probability of Best-Effort traffic in TTEthernet networks (described in detail in Section V). In such cases, the designer can directly use the PCE probability estimates from phase I for making system design choices, without going into the fine-grained analysis of phase II.

In the subsequent sections, we show how a system designer can use the probabilistic first-order relational PCE models to efficiently capture interactions between different component failures in a system. In the literature, various architectures have been proposed for increasing system dependability. These include replication of components and self-checking pairs to increase integrity, system or path redundancy to increase availability, or load balancing by distributing traffic or tasks among similar system components.

In this paper, we illustrate our probabilistic failure propagation methodology with the help of two architectures. We demonstrate the generality and efficacy of our approach by first doing detailed analysis of the parallel-3-hl fault-tolerant architecture, followed by a concrete example from the area of fault-tolerant networking, TTEthernet. We also show the effectiveness of our methodology in another domain, the Cabin Air Compressor (CAC) from Honeywell. However, our methodology is not restricted to these — it is generally applicable to other dependability strategies commonly used in safety-critical applications.

IV. Example Fault-tolerant Architecture

In the parallel-3-hl architecture, three components are connected in parallel in a system, and the system is operational if any one of the components is functioning. Failure of one or more components puts a high load on the remaining components, increasing their probability of failure. In this section, we discuss phase I of our methodology (i.e., sensitivity analysis) for this architecture. Because this architecture is reasonably simple, we first show how we can derive the corresponding analytical equation in closed-form.

A. Symbolic Analysis for parallel-3-hl

If we make the simplifying assumption that the failure in one component is independent of the failure of the other components, then the probability of system failure will be simply: \( ps = pc^3 \) (based on the notation in Table I).

**TABLE I**

<table>
<thead>
<tr>
<th>Notation for Symbolic Analysis of System Failure Probability.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( ps )</td>
</tr>
<tr>
<td>( pc )</td>
</tr>
<tr>
<td>( ph_1 )</td>
</tr>
<tr>
<td>( ph_2 )</td>
</tr>
</tbody>
</table>

However, in real-world systems, as described in Section I, the failure of one component potentially affects other components, e.g., by putting a high load on the other components and raising their probability of failure. This is captured in the parallel-3-hl architecture. For failure analysis in this architecture, we add a high-load interaction term to the system-failure probability equation:

\[
ps = pc^3 + \binom{3}{2} pc^2 (1 - pc) ph_2. \tag{1}
\]

This version of high-load interaction considers that failures due to high load only happen when two components fail and put high load on the third component, i.e., it considers \( ph_1 \) (see Table I) to be zero.

Figure 1 shows the relevant rules of the MLN that corresponds to the simplified model in Equation 1. In this simplified model, we consider fully grounded propositional rules, to illustrate the equivalence to the symbolic equation — we later discuss the power of first-order logic rules, going beyond the propositional rules. In Figure 1, failComponent, failSystem and failHighLoadTwo are Boolean predicates, which respectively indicate whether a component has failed, the system has failed, or whether a component has failed due to high load caused by two other components failing. Given a system \( s \) having three components \( c_1, c_2, \) and \( c_3 \), rule R1 enumerates the different ways in which the system could fail — either due to all three components failing, or due to two...
# Rules encoding system failure.
R1. add (failComponent(c1) and failComponent(c2) and failComponent(c3))
    or (failComponent(c2) and failComponent(c3) and failHighloadTwo(c1))
    or (failComponent(c1) and failComponent(c2) and failHighloadTwo(c3))
    or (failComponent(c1) and failComponent(c3) and failHighloadTwo(c2))
iff failSystem(s);

# Rules encoding component failure under high load.
R2. add failComponent(c2) and failComponent(c3) iff failHighloadTwo(c1);
R3. add failComponent(c1) and failComponent(c3) iff failHighloadTwo(c2);
R4. add failComponent(c1) and failComponent(c2) iff failHighloadTwo(c3);

Fig. 1. Relevant rules from the simplified parallel-3-hl MLN.

of the components failing and forcing the third component to fail due to high load. Rules R2, R3, and R4 enumerate the different ways in which a component can fail due to high load, when two other components have failed. We then run statistical inference on this PCE model to get the failSystem probability estimate.

B. Full Analysis for parallel-3-hl
To further capture the failure interactions between components, we refine Equation 1 such that high load due to one component failing is also modeled, i.e., $ph_1$ is non-zero. This gives us Equation 2:

$$ps = pc^3 + \left(\frac{3}{2}\right) pc^2 (1 - pc) ph_2 + \frac{3}{1} pc (1 - pc)^2 ph_1^2$$

$$+ \frac{3}{1} pc (1 - pc)^2 \left(\frac{2}{1}\right) ph_1 ph_2.$$  \hspace{1cm} (2)

This equation, which captures our full formulation of high load interactions in the parallel-3-hl architecture, has four terms:

a. The first term is the probability of all three components failing due to reasons other than high load;

b. The second term is the probability of two out of three components (hence the $\frac{3}{2}$ term) failing due to non-high load reasons, putting high load on the third component, causing it to fail;

c. The third term is the probability of one out of three components failing due to non-high load, putting high load on the remaining two components, which as a result both fail together;

d. The fourth term is the probability of one out of three components failing due to non-high load, putting high load on one of the two remaining components (hence the $\frac{3}{1}$ term) that fails. These two failures now put high load on the remaining component, which fails as a result. Note that the fourth term is the sequential equivalent of the third term, i.e., in this case the second and third components fail in sequence due to failure of the first one, rather than failing together.

For the full formulation of highload interactions, specifying the rules in PCE using grounded rules in propositional logic can lead to a large model, especially for complex networks like those considered in Section V. PCE offers the advantage of expressing rules in first-order logic, which would give a more compact representation of the fully-grounded model shown in Figure 1, with equivalent system failure and high load interaction logic.

The relevant rules of the PCE model for parallel-3-hl in first-order logic are shown in Figure 3.

The first four rules R1-R4 encode the four terms in Equation 2 for estimating system failure probability. R1 indicates that if a system has three distinct components (variables
# Rules encoding system failure.

R1. add \([c, d, e, s]\) \((c \neq d)\) and \((d \neq e)\) and \((c \neq e)\) and \(\text{failComponent}(c)\) and \(\neg\text{failComponent}(d)\) and \(\text{failSystem}(s)\);  
R2. add \([c, d, e, s]\) \((c \neq d)\) and \((d \neq e)\) and \((c \neq e)\) and \(\text{failComponent}(c)\) and \(\text{failComponent}(d)\) and \(\text{failSystem}(s)\);  
R3. add \([c, d, e, s]\) \((c \neq d)\) and \((d \neq e)\) and \((c \neq e)\) and \(\text{failComponent}(c)\) and \(\neg\text{failComponent}(d)\) and \(\text{failHighloadTwo}(e)\) => \(\text{failSystem}(s)\);  
R4. add \([c, d, e, s]\) \((c \neq d)\) and \((d \neq e)\) and \((c \neq e)\) and \(\text{failComponent}(c)\) and \(\neg\text{failComponent}(d)\) and \(\text{failHighloadOne}(d)\) and \(\text{failHighloadTwo}(e)\) => \(\text{failSystem}(s)\) 0.1;  
R5. add \([c, d, e, s]\) \((c \neq d)\) and \((d \neq e)\) and \((c \neq e)\) and \(\text{failComponent}(c)\) and \(\text{failComponent}(d)\) and \(\text{failSystem}(s)\);  
R6. add \([c, d, e, s]\) \((c \neq d)\) and \((d \neq e)\) and \((c \neq e)\) and \(\text{failComponent}(c)\) and \(\text{failHighloadOne}(d)\) and \(\text{failHighloadTwo}(e)\) => \(\text{failSystem}(s)\) 0.1;  
R7. add \([c, d, e, s]\) \((c \neq d)\) and \((d \neq e)\) and \((c \neq e)\) and \(\text{failComponent}(c)\) and \(\neg\text{failComponent}(d)\) and \(\text{failHighloadOne}(d)\) and \(\text{failHighloadTwo}(e)\) => \(\text{failSystem}(s)\);  

Fig. 3. Relevant rules from the parallel-3-hl MLN.

c, d and e) and if all three of them fail due to non-high load (predicate \(\text{failComponent}\)), then the system also fails (predicate \(\text{failSystem}\)). Similarly, R4 indicates that if one of the three components fails due to non-high load, then it puts high load on another component (predicate \(\text{failHighloadOne}\)), and finally the two components failing puts load on the third component (\(\text{failHighloadTwo}\)), causing it to fail. Expressing the rules in first-order logic allows the system designer to express the system failure interactions quite compactly.

Another useful property of MLNs is that a first-order logic rule in the model can have weights, corresponds to the probability of the rule being true. The rules R1, R2 and R4 are deterministic since the weights on the rules are effectively infinite when not explicitly specified, implying that if the implicants of these rules are true then the system fails with very high probability (close to 1.0). However, the weight on rule R3 is 0.1, indicating that if the implicant of R3 is true then the system fails with lower probability, close to \(1/(1+e^{-0.1}) = 0.5\). R3 is given a lower weight in the model since the probability of both the remaining components failing simultaneously due to high load, if one component fails, is not very high.

The weights in the PCE model are selected to be approximately the logodds ratio of the probability values given by the domain expert, in the absence of actual data on which the weights can be trained. This assumption is motivated from the theory of Markov Logic Networks, which says that for one formula in the knowledge base, the weight \(w\) and the probability \(p\) are related by the equation: \(w = \log(p/(1-p))\).

Let us now consider the rules in the model that encode the failure dependency due to high load. R5 encodes the system behavior that if two components in the system fail, then it puts high load on the third component with high probability. R6 indicates that if one component fails due to non-high load and the second component fails due to high load caused by the first component failing, then the third component fails due to high load caused by the first two components failing with some probability. Similarly, R7 indicates that if one of the components fails due to non-high load, then it can cause any one of the remaining two components to fail due to high load with some probability. Note that in rules R6 and R7 the system has a lower probability (around 0.5) of failing given the implicants. The weights on R3, R6 and R7 are selected such that the probabilities of the rules are close to 0.5 — this is a value we picked for illustrating our approach. In practice, the system designer can choose the weights based on domain knowledge or available data in the domain.

Let us compare how the sensitivity plots of PCE compare to those derived from the analytical equations. Figure 5 shows the system failure probability \(ps\) plotted against the non-high load component failure probability \(pc\), based on the analytical formulation in Equation 2. In this plot, we consider the probabilities of failure due to high load to be: \(ph_2 = 0.8\) and \(ph_1 = 0.08\). The figure also plots the system failure probability of 1-component, a simple architecture that just has one component in the system, so that the system failure probability \(ps\) is effectively the same as the non-high load component failure probability \(pc\). As we see in the figure, for low values of non-high load component failure probability, parallel-3-hl is a better configuration than 1-component, since for the same component failure probability \(pc\) the system failure probability \(ps\) is lower.

There is however a cross-over around \(pc = 0.6\), beyond which the parallel-3-hl configuration does worse than 1-component. The main reason for this is that at this high value of non-high load component failure probability, the cascade of failures caused with high probability due to the high load-driven failure dependency in the parallel-3-hl configuration increases the system failure probability to the extent that it is no longer useful to have the redundancy provided by the three components connected in this configuration.\(^2\) This crossover happens at a relatively large value of component failure probability \(pc\). Real systems operate at much lower probabilities of component failures – in those ranges, parallel-3-hl is a configuration with better failure tolerance. Note that in this case we are ignoring the fact that 1-component will have to operate at high load all the time to provide the same quality of service, and hence the failure probability \(pc\) for 1-component will be actually higher in practice. So, in real systems the cross-over can

\(^2\)We selected the high value of \(ph_2 = 0.8\) to be able to observe this cross-over behavior.
Fig. 4. Plot of analytical estimates (left) and PCE estimates (right) of system failure probability for parallel-3-hl for $p_{h2} = 0.8$ and $p_{h1} = 0.08$.

(a) Analytical estimates

(b) PCE estimates

Fig. 5. Plot of analytical estimates of system failure probability for parallel-3-hl, for $p_{h2} = 0.8$ and $p_{h1} = 0.08$.

The PCE model encoding the parallel-3-hl configuration is simulated with different probabilities of component failures, which gives us a plot similar to the analytical plot in Figure 5. Figure 6 shows the system failure probability values obtained from the PCE simulation for different values of the non-high load component failure probabilities. The PCE simulation results in Figure 6 have similar characteristics to the analytical results in Figure 5, showing that the MLN is a reasonably accurate model of the system failure probability in the parallel-3-hl configuration. The PCE simulation of the system failure probability gives reasonably accurate estimates of the actual values from the analytical model, within stochastic variation due to sampling.

V. RESULTS

In this section, we discuss results of our experiments on real-life domains.

Fig. 6. Plot of PCE estimates of system failure probability for parallel-3-hl, for $p_{h2} = 0.8$ and $p_{h1} = 0.08$.

Fig. 7. Frame transmission in Simple TTEthernet Cluster.
A. Case Study on TTEthernet

TTEthernet is a communication infrastructure for safety-critical and mixed-criticality applications. It supports the replication of communication channels and simultaneous transmission of critical traffic over all redundant communication channels. A simple TTEthernet network connecting a sender to a receiver via three redundant channels is depicted in Figure 7. Non-critical traffic, called the Best-Effort (BE) traffic, is statically assigned to only one of the redundant channels. Hence, in the event of a failure of a channel, the BE traffic assigned to this channel is lost.

To improve the probability of successful transmissions of BE frames we want to investigate a simple TTEthernet middleware extension: in case of a failure of a channel, the BE traffic of this channel will be re-routed to the other remaining channel(s). However, the re-routing of the BE traffic may cause buffer overflows in the switches of the non-faulty communication channels.

We can cast the TTEthernet middleware extension problem in terms of the high load model described in Section III, using the following mapping:

A. We consider the parallel-3-hl configuration for this case, which means that if one channel fails, then that puts high load on the two other channels (corresponding to \( ph_1 \)); when a second channel fails, it puts even higher load for failure on the third channel (corresponding to \( ph_2 \)).

B. We have the following three failure modes in TTEthernet:
1) Frame loss due to component failure
2) Frame loss due to bit error in transmission
3) Frame loss due to buffer overflow
3a) Frame loss due to buffer overflow without rerouting
3b) Frame loss due to buffer overflow with rerouting

In our model, \( ps \) is the probability that a frame is lost, while \( pc \) is a function of 1) and 2). 3a) should not occur if the buffers are sufficiently large, which is reasonable in closed systems like avionics; \( ph_1 \) and \( ph_2 \) capture the probability of 3b), i.e., a frame can get lost due to a buffer overflow in any one (\( ph_1 \)) or any two (\( ph_2 \)) switches on the path between the frame sender and the receiver, due to frames being rerouted from a failed path.

For simple TTEthernet networks, we are able to analytically calculate the system failure probabilities and to compare with the PCE estimates – analogous to the parallel-3-hl configuration in the previous section. However, to demonstrate the scalability of the PCE approach, we analyze a TTEthernet network consisting of sixteen end systems and three channels (Figure 8). Again, the channels are used to transmit redundant copies of the Ethernet frames – concurrently for critical frames and on-demand sequentially for BE traffic. Each channel is composed of three switches, which are connected through backbone links. We assume that the end systems implement a TTEthernet middleware, which reliably and consistently detects the failures of switches. We also assume that a single switch failure is sufficient for the complete channel it belongs to being classified as being faulty.

For this network, we used our PCE-based sensitivity analysis tool to model the failure probability for frame transmission between any two end systems as a function of the probability of buffer overflow due to high load. Figure 10(b) is the sensitivity plot of system failure probability for low values of probability of buffer overflow due to high load, with switch failure probability being \( 10^{-6} \). The y-axis is in log scale to demonstrate the low values, e.g., \( 10^{-9} \) reached by the system failure probability. Doing the analytical calculation is infeasible in this case. Figure 9 is the same plot for the full range of \( ph \), with the y-axis in the probability scale. Using this plot, the system designer can estimate the buffer overflow probability for the system failure probability she is willing to tolerate in the design. Modeling and analysis of the probability that best-effort delivery fails to deliver a packet can help a designer choose system parameters (e.g., buffer sizes, link bandwidths) based on models of expected traffic.

Detailed analysis of concrete examples based on measured network traffic in actual systems is left for future work.

---

3The PCE model for the large TTEthernet network is available at: http://www.csl.sri.com/users/shalini/prdc2013/network-3-hl-seq.mln

---

Fig. 8. Configuration of Large TTEthernet Cluster.

For this network, \( pc = 10^{-6} \), assuming \( ph_1 = ph_2/2 \).

B. PCE Approximation of Analytical Results

In a probabilistic inference tool like PCE, the marginal probability estimate of a predicate depends on its prior probability,
as discussed in Section II. Given a predicate \( R \) (e.g., failSystem) in the model, the prior probability of \( R \) can encode the system designer’s domain knowledge, or can be derived from typical parameters that can be found in the literature. In the PCE model for TTEthernet middleware extension, the systems designer would usually have domain knowledge regarding the relationship between the component prior failure probability due to non-high load interaction and the system prior failure probability, e.g., if components fail with probability in the order of \( 10^{-6} \), the system failure probability would be of the order of \( 10^{-9} \).

In the PCE model for the TTEthernet middleware extension, we chose pessimistic values for prior probabilities (e.g., priors on \( ph_2 \) and \( ph_3 \) are 1.0), which would give us conservative estimates of system failure probability. This represents the worst case when the system designer does not specify priors based on domain knowledge — we assumed this to illustrate how PCE behaves in the worst case. For simple TTEthernet networks, we deduced analytic models — as expected, the PCE models give more conservative estimates of the system failure probabilities than the analytical models in the sensitivity plots, since the analytical models in the plots are not simulating the worst case behavior. Even in these worst-case plots, for the region of primary interest for low values of \( ph_2 \), the PCE plots track the analytical plots reasonably closely (Figure 10(a)).

Note: When the system designer has better estimates of the prior probability values for the TTEthernet middleware extension, incorporating those actual prior values into the PCE models should make the sensitivity plots generated by PCE track the analytical plots even more closely.

VI. ALTERNATE DOMAIN

The proposed failure dependency modeling methodology has been applied to other domains and architectures as well. In collaboration with Honeywell International, Inc., Aerospace, we performed analysis on some examples from the area of Aircraft Environmental Control Systems (ECS). Using our methodology, we modeled a part of the ECS architecture, the dual Cabin Air Compressor (CAC), through inner-loop controls and supervisory modes. The dual CAC subsystem is used to deliver fresh air to the cabin/cockpit and provide cabin pressurization. The subsystems outside the dual CAC condition the air before sending it to the cabin/cockpit, and cool the aircraft avionics and ECS motor controllers using a special coolant.

Under normal operation, both compressors should work to provide the same airflow rate, though not necessarily at the same speed. The flow control loops of the system controller regulate the airflow rate to a predefined flow set point. The supervisory control of the system controller determines the flow set point depending on whether the dual CAC subsystem exhibits a failure in one of the compressors. In the case of a single CAC failure, the flow demand for the second CAC is increased and the system is run in a degraded mode to provide the minimum air supply.

We utilized failure modes and our probabilistic verification methodology to evaluate constraints on the dual CAC design. Component interactions are captured through higher-level operational modes and scenarios. Using various potential CAC architectures, we performed trade-off analyses to calculate component-level probabilities for which the required system-level probabilities hold. The analysis results provided useful input regarding component requirements, which can guide the design choices in early design stages. A natural generalization of the dual CAC design is a triple CAC, which gives more reliability for mission-critical applications. Since the CAC supplies cabin air, a conservative mode of applying the triple CAC system is in the “voting” mode — if two components fail, then the CAC system fails and a backup system takes over.

A. Analysis of voting-3-hl architecture

Motivated by this application, another architecture we consider in our analysis is voting-3-hl. Equation 3 shows the system failure probability for voting-3-hl. The second term in the equation indicates that if two components fail, then the voting configuration ensures that the system fails even if
the third component does not fail. The third term considers that one of the three components fails due to non-high load, and has the following 2 cases:

1. The remaining two components both fail due to high load caused by the first component failing, causing the system to fail with probability $ph_1^2$.
2. One of the two components fails first with probability $(1/2)ph_1$ and the system fails irrespective of whether the third component fails in sequence or not, since the architecture is voting.

$$ps = pc^3 + (3/2)pc^2(1 - pc) + (1/2)pc(1 - pc)^2$$
$$= (ph_1^2 + (1/2)ph_1). \quad (3)$$

Figures 11 and 12 show the plots of failure probabilities vs component failure probabilities for the voting-3-hl architecture from the analytical model and from PCE simulation respectively – as can be seen from the figures, PCE simulation estimates are close to the actual analytical values of system failure, within stochastic variation due to sampling, similar to what we observed for the parallel-3-hl architecture.

VII. RELATED WORK

In this section, we describe different relevant related work, highlight their main difference to our approach in this paper, and propose some follow-up future work.

A. Modeling expressivity

Fault Tree Analysis (FTA) [6] and Failure Modes and Effects Analysis (FMEA) [7] are classical approaches to failure analysis. Recent times have seen the advent of multiple advanced probabilistic analysis techniques and tools, e.g., stochastic Petri nets [8], Markov chains [9], statistical model checkers [10], probabilistic model checkers like PRISM [11], equational logic (EL) [12], OpenSESAME [13]. One of the benefits of our PCE-based approach is that it simultaneously facilitates probabilistic reasoning as well as succinct representations of complex structures and failure interactions using first-order relational models. So, our models scale well with the size of the underlying architecture, as demonstrated by the TTEthernet large network example in Section V. In the future, we would like to apply our PCE-based sensitivity analysis approach to other large complex cyber-physical systems.

B. Probabilistic Sensitivity Analysis

One of the important aspects of our methodology is probabilistic sensitivity analysis. Paulitsch et al. [14] motivate the importance of performing probabilistic sensitivity analysis for low-cost fault-tolerant system designs, to facilitate the study of trade-offs, e.g., improving fault-tolerance properties while respecting cost constraints. They present a probabilistic sensitivity analysis of the overall dependability of the Braided Ring Availability Integrity Network (BRAIN), with respect to different fault types and failure rates in a safety-relevant application, using the ASSIST/SURE/STEM tool suite for model evaluations [15]. ASSIST is a rule specification language in propositional logic for generating semi-Markov models for SURE, which model failure behavior and recovery. A recent integrated environment is Mobius, which provides a robust infrastructure for combining multiple modeling formalisms and evaluators [16]. In future work, we would like to explore integrating PCE into the Mobius environment.

C. Rare Events

Failures in reliable complex systems are typically rare events. Tools like PCE, which use sampling to drive inference, have to be run for larger number of sampling iterations as system failure events become more rare. In this paper, we have shown how PCE can be used to model system failure probabilities as low as $10^{-9}$. In the future we will work to extend the techniques to efficiently handle even rarer events using techniques similar to recent advances in statistical model checking [10], state density estimation in combinatorial
problems [17], or by symbolically categorizing the population and then sampling the categories separately [18], [19].

D. Compositionality and Abstraction

One advantage of a tool like PCE is that the models can be specified at multiple levels of abstraction, e.g., component level and system level. Since the models in PCE are modular and can be created at different levels of abstraction, it may be possible to compose our failure models with those of other tools, e.g., equational logic [12]. Other compositional approaches to fault analysis, which construct system failure models from several component failure models, include the Hierarchically Performed Hazard Origin and Propagation Studies (HiP-HOPS) [20], State/Event-Fault Trees (SEFTs) [9], Failure Propagation and Transformation Analysis (FPTA) [21].

E. Temporal Failure Modeling

There are several techniques in the literature for modeling temporal ordering of failure events, e.g., Temporal Fault Trees (TFTs) [22], Priority Dynamic Fault Trees (PDFTs) [23], and Timed failure propagation graphs (TFPGs) [24]. The TTEthernet failure model in PCE represents sequential failure propagation at a higher level of abstraction and takes into account non-iid interactions, since PCE is a relational modeling tool. In the future, we plan to investigate extending the PCE approach to support temporal analysis of realtime behavior.

VIII. CONCLUSION AND FUTURE WORK

We discussed a novel and scalable methodology for probabilistic modeling of failure dependencies using Markov Logic Networks (MLNs) for large, complex systems. Using the Probabilistic Consistency Engine (PCE) tool, we demonstrated probabilistic sensitivity analysis and other advantages of MLN modeling for failure analysis in complex cyber-physical systems. We performed abstract symbolic analysis to obtain bounds and for comparison to the PCE results. The method was illustrated on real-world architectures for avionics, space, and automotive electronics, including the Time-Triggered Ethernet (TTEthernet) communication platform and the Honeywell Cabin Air Compressor (CAC) architecture.

The benefits of the methodology discussed here include a novel ability to perform probabilistic sensitivity analysis at design time. A designer faced with the challenge of building a complex system with failure dependencies is able to quickly build scalable MLN models at a higher level of abstraction. These models can be used for probabilistic calculation of potential reliability in a principled manner, despite possibilities of cascading failures and a potential state space explosion problems. Using advances in MLN tools such as PCE, designers can explore the reliability of increasingly complex and highly reliable systems.

Apart from the future areas of work already outlined in Section VII, some other next steps for this research direction can include expanding the applicability of the described methodology. For example, we can expand the set of fault classes, perhaps into a so-called hybrid fault model [18], [19]. Further, we hope to apply the techniques to other cyber-physical systems with complex failure dependencies, such as the SPIDER architecture [25] and other commercially-relevant applications, where our methodology could also be beneficial.

Acknowledgment: We would like to thank our collaborators at Honeywell for their help with the CAC analysis. We would also like to thank Natarajan Shankar, Ashish Tiwari and Bruno Duterte, for valuable feedback regarding this work.

REFERENCES