A Context-Partitioned Stochastic Modeling System with Causally Informed Context Management and Model Induction

Nikita A. Sakhanenko[1], Roshan R. Rammohan, George F. Luger, and Carl R. Stern[2]

[1] Computer Science Department,
University of New Mexico, Albuquerque NM 87131, USA,
sanik@cs.unm.edu
[2] Management Sciences, Inc.,
6022 Constitution Ave NE, Albuquerque, NM 87110, USA

Abstract. We describe a flexible multi-layer architecture for context-sensitive stochastic modeling. The architecture incorporates a high performance stochastic modeling core based on a recursive form of probabilistic logic. On top of this modeling core, causal representations and reasoning direct a long-term incremental learning process that produces a context-partitioned library of stochastic models. The failure-driven learning procedure for expanding and refining the model library employs a combination of abductive inference together with EM model induction to construct new models when current models no longer perform acceptably. The system uses a causal finite state machine representation to control on-line model switching and model adaptation along with embedded learning. Our system is designed to support operational deployment in real-time monitoring, diagnostic, prognostic, and decision support applications. In this paper we describe the basic multi-layer architecture along with new learning algorithms inspired by developmental learning theory.

1 Introduction

Structured probabilistic modeling, including probabilistic graphical models as described by Pearl [14] as well as more recent extensions into logic-based [8], frame-based [6, 12], and relational representations [2, 3], have found increasing uses across a range of applications, including systems for classification, diagnostics, prognostics, and decision support.

A central reason for the attractiveness of this approach is the availability of proven techniques for learning models from data, especially techniques for automated calibration of critical model parameters to enhance accuracy and performance. However a key limitation of these model induction techniques is that they can only be applied in domains where systems exhibit a certain second-order form of global consistency and invariance. Probabilistic modeling techniques can be used, of course, to model dynamic systems, and can be used very effectively to

induce state transition probabilities capturing system dynamics. However most often the first-order system dynamics themselves must be assumed to be stationary and invariant over the entire body of training data. When we confront a system with changing dynamics dependent on states and events of an external environment, and when it is not known a priori which aspects of the environment are altering system dynamics, then every conceivable aspect of the world must be explicitly represented in the training data and the learning algorithm in order to avoid overlooking hidden determinants and relationships. This, of course, leads to significant challenges in complexity and tractability.

The approach described here is intended to address these complexity issues by using a causal theory to focus on a smaller set of relevant environmental dependencies. This approach begins with a limited set of models and a set of long term incremental learning mechanisms. Learning is typically triggered by detection of model failure indicating the presence of a context transition event. The system then uses a causal theory and causal reasoning to focus on the subset of elements in the current environment relevant to the observed changes. Over the long term, failure detection, causal analysis, and model repair mechanisms incrementally populate a context partitioned library of models. Benefits of this approach include the ability to achieve improved learning accuracy by focusing learning episodes on homogeneous context-constrained data, the ability to incrementally learn and refine a set of increasingly effective context partitions, and the ability to learn and explain changes in system behavior in terms of environmental conditions and events.

In the next section we give necessary background information and describe the basic multi-layer architecture for context-sensitive stochastic modeling along with new incremental learning algorithms inspired by developmental learning theory. Then we provide a simple example to illustrate how context can be used in a probabilistic logic-based modeling system. In section 2 we give a formal view on context and its connection with causal models. Section 3 shows the context splitting mechanism and its relationship to developmental learning. In section 4 we describe the system for managing a context partitioned library of models. A datailed example is given in section 5. Finally, in section 6 we provide an overview of related work and conclude in section 7.

1.1 Background

The context-sensitive probabilistic modeling system proposed in this paper is based on our first-order, stochastic, and Turing complete "Loopy Logic" language [17]. Loopy Logic is a knowledge-based model construction approach whose knowledge base consists of declaritive specifications of the form:

`head | body`$_1$`, body`$_2$`, ..., body`$_n$`=[ p`$_1$`, p`$_2$`, ..., p`$_m$`]`.

The size m of the conditional probability table at the end of the sentence is equal to the arity (number of states) of the head times the product of the arities of the body terms. Suppose `X` is a predicate that returns either `red`, `green`, or `blue`

and `Y` is a typical boolean predicate. The conditional probability distribution $P(X|Y)$ is represented in Loopy Logic as follows:

```
X | Y = [[0.1,0.2,0.7],[0.3,0.3,0.4]].
```

Product distributions are used as a combining function to deal with several rules having a matching head. In order to perform inference, Loopy Logic converts the sentences from the knowledge base into a bipartite Markov random field with two types of nodes (see Fig. 1): variable nodes (round) corresponding to clauses of a sentence and cluster nodes (square) representing the conditional probability distribution of its immediate neighbors. Loopy belief propagation

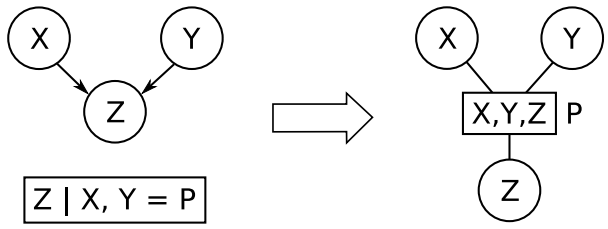

Fig. 1. A Loopy Logic sentence and a Bayesian network it defines (on the left) is converted into a bipartite Markov random field (on the right).

[15] (hence the name Loopy Logic) is used to perform inference within a Markov random field.

We choose Loopy Logic as a base for the system presented in this paper for several reasons including the ability to perform a form of EM parameter learning [1]. Another advantage is that Loopy Logic supports dynamic probabilistic models via recursive definitions.

The context-sensitive stochastic modeling system proposed in this paper has the multi-layer architecture illustrated in Fig. 2. Using a knowledge-based model

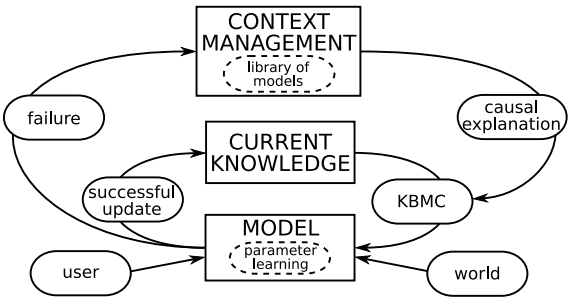

Fig. 2. A schematic illustration of the failure-driven architecture of our system.

construction (KBMC) approach, a probabilistic graphical model (MODEL) is constructed by converting Loopy Logic sentences in a knowledge base (CURRENT KNOWLEDGE) into a Markov random field. As new data arrives, we store the successful model updates in the knowledge base. Otherwise, failure forces a context switching mechanism (CONTEXT MANAGEMENT) to switch to a new model according to a likely causal explanation of the failure. Note that the incremental learning mechanisms of this architecture are inspired by developmental learning theory: successful updates to a current model (the small loop in Fig. 2) correspond to learning by assimilation whereas partitioning models by contexts and switching between the models (the large loop) corresponds to learning by accommodation [16, 4].

1.2 Example

Consider a burglar alarm guarding a residence. Suddenly the owner, at a different location, receives a message that the alarm has been triggered. We would like to compute the probability that the owner's home has been burgled. We assume that the alarm system is installed exclusively in Albuquerque, New Mexico. The local police department records the data when an alarm goes off and when the residence monitored by the alarm is actually burgled.

Using a declarative language representation [17], the following sentence is added to the knowledge base (KB):

```
Alarm(x)|Burglary(x) = [0.9,0.1,0.001,0.999].
```

This sentence shows that the event that the alarm goes off is dependent on the event that the residence is burgled, where the conditional probability distribution is discovered after inferencing over a Markov random field constructed from the sentences of the KB. The probability table indicates that the alarm goes off in 90% of the cases involving a burglary, and it does not go off 99% of the time when there is no burglary.

Note that this probabilistic information is learned only from Albuquerque data. To make this example more interesting, let us assume the alarm company starts selling its alarm system in Los Angeles, California, thus the police database from which we draw probabilistic information receives additional data tuples obtained in LA. Let us assume that there are many more false positive instances of the alarm going off in the Los Angeles area (LA is earthquake-active) causing the distribution of LA data to be very different from that of Albuquerque. As a result, our graphical model trained only on Albuquerque data does not fit LA.

Hence, the model will fail and must be reorganized in order to account for the new data that do not fit the existing structure of the model. In our example we see that better performance can be acheived by splitting the model on cases depending on location. As a result the initial model is split into two distinct contexts corresponding to Albuquerque and Los Angeles. The model for Albuquerque is identical to the original one, but the model for Los Angeles has to be learned using parameter estimation just as it learned the original model in the

beginning of our example. Consequently, the structure of the KB is changed and its distributions are updated:

```
Alarm|Burglary = [0.9,0.1,0.001,0.999] ⟝ Albuquerque
Alarm|Burglary = [0.9,0.1,0.1,0.9] ⟝ Los Angeles.
```

In our example we represent a context as a predicate that is attached to a rule using the symbol $\leftharpoondown$ similar to the notation of Haddawy [6].

When the alarm company next moves to San Francisco, we can perform a procedure similar to the one in the case of Los Angeles. However, if we have some underlying information that "ties" the Los Angeles case with that of San Francisco, then before building a new model for San Francisco we try to apply the Los Angeles model to the San Francisco data. Such underlying information can include a close distance relation or, perhaps, a relation of being in an earthquake active zone. If the Los Angeles model is successfully applied to San Francisco data, then the updated model will have a context `Los Angeles` $\wedge$ `San Francisco`.

This approach is inspired by developmental learning in humans: an individual learns a strategy that works and then uses it until it stops working, in which case the individual will have to then learn a new strategy [16]. The technique described in this example we call *context splitting*.

2 Causality and context

A context-sensitive probabilistic model-management system that is intended to operate in real time must be capable of handling sudden and drastic changes in observed data. These changes might, in turn, trigger the system to switch between several probabilistic models each one fine-tuned to suit a particular context of operation. Real-time inference for each context is possible and useful only if the probabistic models are appropriate and tractable enough to yield meaningful results given limited time and resources available for parameter adjustments and inference. Thus, if models are small enough to be fast, they are often not likely to be stable for long periods of operation in dynamic settings.

To replace a failing model with a more relevant one that suits the current context, a context switching mechanism must be capable of operating at speeds related to the dynamics of the data. Exhaustive searches to determine the correct context of operation at each instance of model failure are impractical. In fact, at the frontier of a context change we must be able to prune away many context switches and restrict our search to a select few. At this point, we could rely on referring to a causal theory, an external dictionary, if we may say so, of feasible and pertinent causal relationships that could become active at that point. Such a causal theory would invalidate certain context switches while validating others.

For example, the causal theory for a weather modeling system should not allow us to consider the context switch from *winter* to *summer* directly, but would suggest a context switch to *spring* instead. The causal theory also must suggest what variables to include in the new probabilistic model that is to be loaded or learned in the new context. So, in the context of *spring* the causal

theory would suggest that we include *humidity* and *temperature* in the model but not *snowfall*.

The next subsection describes the nature of the causal theory and how it can assist in the process of abduction and context management.

2.1 Representation

In intuitive terms, a causal theory is a representation of the understanding of the mechanisms that govern the behavior of the world. This causal theory is used to explain observations. We require such a theory to be complete, so that it describes all relevant causal relationships that are necessary to explain away observed phenomena. Ideally this should be done in the simplest way possible, in accordance with Occam's principle of parsimony. Therefore, a good causal theory is both complete and minimal. A causal theory, can be represented in several ways.

First, according to the Laplacian view [9], natural phenomena are predominantly deterministic with some stochastic effects introduced by unknown or unobserved variables and noise. Such representations can involve deterministic functional equations approximating the natural phenomena with probabilistic terms included to represent uncertainties.

Second, Bayesian causal relationships are inherently stochastc in nature and determinism is merely a convenient approximation. Such causal representations can be modeled by relational terms on random variables together with conditional probability distributions on the variables. Although the Laplacian view is more general and more intuitive to a human, in this paper we adhere to the Bayesian approach.

More specifically, in our approach we also abstract away conditional probabilities from the causal theory and allow the causal theory to represent the existence (or more importantly the non-existence) of causal relationships among variables. The details of associating these relationships with a distribution is left to the the context-management and modeling system that utilizes this theory.

The causal theory includes causal relationships relevant to the model, where specific subsets of these become active in each context. We represent each of these relationships using a *Horn clause* also referred to as a *rule*. We assume that such a theory is complete and minimal, i.e., it includes all the necessary rules but no more than is necessary. We refer to a causal theory of the system as Θ. Note that in order to perform causal reasoning, our causal theory Θ should satisfy certain conditions. However, this is not a focus of our paper, and thus we assume a domain expert provides a detailed and valid causal theory Θ.

2.2 Definitions

In this section we formally define the notion of context.

Definition 1. *Let $B = \{b_1, \ldots, b_n\}$ be a set of boolean variables for some positive integer n. Assume that a set of all possible truth assignments to B is enumerated and π_j is a truth assignment with an index j, a positive integer. A*

background assignment σ *is defined as*

$$\sigma(\{b_1, \ldots, b_n\}, j) \equiv \bigwedge_{i=1}^{n} \pi_j(b_i).$$

A background assignment essentially provides a way to construct a logical formula from a set of boolean variables. We define a background in Definition 2.

Definition 2. *A* background set $\mathbb{B}$ *is*

$$\mathbb{B} \equiv \left\{ \langle B, T, \sigma \rangle \;\middle|\; \begin{array}{l} B \text{ is a set of boolean variables,} \\ T \text{ is a set of positive integers,} \\ \sigma \text{ is a background assignment:} \\ \quad \forall y \in T \quad \sigma(B, y) \models_\Theta \top \\ \quad \forall y \notin T \quad \sigma(B, y) \models_\Theta \bot \end{array} \right\}$$

An element of $\mathbb{B}$, called a *background*, is a tuple constructed from a set of boolean variables, a subset of positive integers, and a background assignment. Intuitively, a background is a configuration of the system during the time intervals specified by a set of positive integers. We assume time is divided into disjoint intervals, thus the entire time of our system's operation can be represented using positive integers, a number for each interval. By using conditions on a background assignment we make sure that a background is "valid" during specified time intervals and only during these intervals. In other words, a formula generated by the background assignment applied on the boolean variables of the background is valid under the causal theory Θ for every time interval specified in T, and is invalid for all the other time intervals.

Definition 3. $\mathbb{C} \subset \mathbb{B}$ *is a set of* contexts *iff*

1. $\forall$ *integer* $x \geq 0 \quad \forall C_1 = \langle B_1, T_1, \sigma_1 \rangle, C_2 = \langle B_2, T_2, \sigma_2 \rangle \in \mathbb{C}$
   $C_1 \neq C_2 \Rightarrow \sigma_1(B_1, x) \wedge \sigma_2(B_2, x) \models_\Theta \bot$
2. $\forall$ *integer* $x \geq 0 \quad \exists C_1 = \langle B_1, T_1, \sigma_1 \rangle \in \mathbb{C} : \sigma_1(B_1, x) \models_\Theta \top$

An element of $\mathbb{C}$, called a *context*, is a background that satisfies two conditions. The first condition ensures that there are no two distinct contexts that are valid under the causal theory Θ during the same time interval. The second condition ensures that for any time interval there exists a context that is valid under the causal theory Θ during that interval.

Note that definitions 2 and 3 imply that for any two contexts their respective sets of time intervals, when the contexts are valid, do not overlap. Further, using this observation and the second condition of definition 3, there exists *one and only one* valid context under the causal theory Θ during each interval of time.

Note that these definitions serve to clarify our notion of context. The task of formally defining the connection of contexts with current graphical models and data is not a focus of this paper, and hence, is left implicit.

2.3 Advantages

There are several advantages of using a causally driven context management system. Pruning irrelevant information and restricting focus to a smaller, coherent, and tractable set of variables is one advantage. In the absence of causal information one often has to rely on statistical information, such as regression, to prune the model. This tends to be computationally intractable and choices of what variables to include in the model can seem very arbitrary.

Especially, when dealing with real-time systems that exhibit non-stationarity, it is very expensive to maintain, update, and carry out inference in a single model that captures all possible states of the system and includes all relationships. For example, if we are observing vibration recordings on an aircraft, we must know that the contact vibration between landing gear and the ground along with the wind turbulence are relevant quantities while the aircraft is taxiing. Once airborne and the landing gear is retracted, this becomes irrelevant and need not be considered in the model. Perhaps an additional variable, namely "g" force that was irrelevant earlier, becomes relevant now and must be included. In the absence of a pruned model all three variables will have to be captured in a single model at the expense of tractability, which often becomes crucial in fast response, real-time systems.

With the assistance of causal information however, we can use a different model in each context of the system's operation, pruning away unnecessary variables as required. We require that only one context be valid at a time; a model capturing this may be learned from data acquired when the context is active. When the observed data undergo a significant qualititative or quantitative change, such that the current model *fails*, we consider this an indication of a context transition event. If the change is not severe, a modification of the parameters of the model can account for the unexpected data, or we must archive our model as no longer relevant and assume that we are operating in a new context.

A context transition event is also indicated by changes in key variables that are external to the model, and that are relatively stationary with respect to the model. These are called *exogenous* variables. *Endogenous* variables for a particular context, however, are explicitly included in the model. When a certain context become active, *exogenous* variables take fixed values, for example $taxiing = true$. Constraints on the set of exogenous variables help determine a context and also which rules in the causal theory become active for that context. Switching between contexts may therefore be triggered by observation on these exogenous variables [15, 19].

3 A context splitting mechanism

3.1 Model failure and its connection to developmental learning

We have adapted ideas from developmental learning to motivate our current approach. Developmental studies consider different stages of human learning. When a child sees a new pattern, she tries to fit it into her existing cognitive

structures. When the pattern has never been encountered before, it may be inconsistent with the child's cognitive structures. The normal child uses her mental structures to fit the new pattern. Human learning that assumes that the pattern fits well into the existing cognitive structures, is referred to as *learning by assimilation* [16]. On the other hand, reorganization of the mental structures to account for a new pattern is called *learning by accommodation* [16]. Psychologists believe that these forms of learning happen over the life of a person [16, 4, 5].

We feel that our probabilistic inference system benefits greatly by emulating these mechanisms. The traditional outlook in Machine Learning and Artificial Intelligence has been to look at all the evidence at once and learn the best model from this evidence. At every stage of development in life, however, humans continuously reinforce their beliefs using learning through assimilation, or revise their beliefs using learning through accommodation.

In probabilistic logic systems that use this framework of learning, models tend to become more tractable. Most of the time new evidence is rapidly "assimilated" into existing small context-sensitive models. Data sets that exhibit large scale deviations from previously learned models bring the more expensive "accommodation" mechanism into play. If new data patterns are observed more frequently, the system spends a lot of cycles developing new models. However, for a mature system, one that has been exposed to a considerable amount of data ranging across a variety of situations, this would be rare.

Two related and common problems in machine learning are the problem of over-fitting and over-generalization. When single models are learned on a data set that is not diverse, models tend to become too specific and are said to over-fit and unable to generalize and account for slightly varying datasets. The converse problem of over generalization is when a very general model is learned from well distributed and possibly sparse data in the learning stage and, therefore, performs badly on all types of data in the operational stage. When used in probabilistic systems, the mechanisms of assimilation and accommodation along with the notion of context change, will help minimize these problems.

Context splitting mechanisms which are one of the main components of our system employ these two forms of learning within a failure-driven architecture. The following pseudo-code captures the idea of context splitting.

```
while( new-data := get-next-block( dataset ) ) do
   cluster := check-failure( current-model, new-data );
   if( not( cluster ) ) then
      update( current-model, new-data );
   else
      effect := get-effect( cluster, new-date );
      pair := find-existing-model( effect, new-data,
                                          current-model );
      current-model := head( pair );
      if( equal( tail( pair ), F ) ) then
         new-model := create-new-model( effect, new-data,
                                                current-model );
```

```
      current-model := new-model;
   update( current-model, new-data );
```

When new data are available, the system checks whether the current model fits the dataset well. If it does, we incorporate the data into the model by updating its probability distribution. Otherwise, if the model fails to fit the data, we save the current model and look for a new version that will account for the new data.

The pseudo-code, when the `if`-statement holds, corresponds to learning by assimilation: the model is consistent with new data and it is fine-tuned by assimilating the dataset. On the other hand, when the `else`-statement holds, learning is by accommodation: the model is inconsistent with new data, hence in order to account for the dataset we have to reorganize our model.

It can be seen from our pseudo-code that failure identified by the function `check-failure` is the core of the architecture. The notion of failure represents the situation when new data are inconsistent with the current model. One way to do this is by computing the conditional probability of the data given the current model. If this probability is below a certain threshold, then we say that the model fails. The failure threshold can be provided by a domain expert. Alternatively, it can be found by computing the conditional probability of the sampling of the previous dataset given the model.

3.2 Building failure explanations from causal knowledge

It is important to emphasize that model failure is closely related to a context change. Thus, in order to realize the particulars of the model change, the failure must be *explained*. In the next section this situation is addressed. For now we focus on the problem of building an explanation.

This task consists of two parts: searching for the failure *effect* and searching for the *cause* of the effect. Since we use probabilistic graphical models (represented by a *Markov random field* [14]), the failure test (represented by the function `check-failure` in the pseudo-code above) can be carried out in a flexible way. In the case of the failure of a specific cluster node in the failed Markov random field, this cluster can be considered as a failure effect.

In order to find a cause of a failure effect we employ a causal theory such as that discussed in section 2. The cause of a failure effect is derived using *abductive reasoning*.

```
explanation-list := [];
to-explain-queue := [effect];
while( to-explain-queue ) do
   elem := remove-from-front( to-explain-queue );
   if( observed( elem ) ) then
      if( consistent( elem, data ) ) then
         add-to-tail( elem, explanation-list );
   else
      tail-list := get-matched-rule-tails( elem, causal-theory );
      push( tail-list, to-explain-queue );
```

We start the reasoning by matching the failure effect with the head of causal rules. If the head matches, we analyze the tail of the rule. The tail is a possible explanation of the failure effect, when it is observed and consistent with the evidence. If the tail is inconsistent with the data, we discard the rule and go to the next one. If the tail is not observed, it is added to the list of elements to match against causal rules. The pair `[cause, effect]` constitutes an explanation of the failure. In the next section we discuss how such explanations can be used to efficiently manage models. Note that our algorithm is a simple form of abductive search. More sophisticated algorithms [20, 10] can be used to improve explanation construction.

4 Causally informed state transition mechanisms

Let us assume we have learned a model that fits the incoming data well. But suddenly new data are obtained and the current model fails on this dataset (note the definition of model failure presented in section 3). Recall that failure triggers context switching. The context transition is modeled by learning a new model on the new data corresponding to a new context.

Note however that the new context could have been encountered before, if, for example, the underlying physical system returned to its equilibrium after some outside shock. Instead of learning a new model in this case, we want to find a previously learned model corresponding to the context. Only after no suitable model is found a new one has to be learned. Further, in order to find the appropriate model fast, we want to use causal information along with the current data. The following algorithm is proposed to implement these ideas.

Consider a graph $G = (V, E)$ with a set of vertices V and a set of edges E. For each vertex $v \in V$ there is a model associated with it. And for any two vertices of G, $v_1, v_2 \in V$, there exists an edge $e \in E$ connecting v_1 and v_2 *iff* the model associated with v_1 has failed at some moment of time and the next model that captured the context change at that moment is associated with v_2. Each time a model fails and a new model is constructed, we want to store that in the graph G by adding an edge between the vertices corresponding to these models. This edge is then tagged by the *causal explanation* of the failure of the model. The explanation is a pair `[cause, effect]`. Recall that the effect that characterizes the model failure is determined by fitting the new data to the model. Assuming our causal theory contains all the necessary information, the cause of the failure is derived using abduction (section 3.2). The following pseudo-code represents a function for creating a new model:

```
create-new-model( effect, data, model )
   cause := abduction( effect, causal-theory, data );
   new-model := make-model( data );
   edge := link( model, new-model );
   tag( edge, [cause, effect] );
   return new-model;
```

As noted above, before learning a new model, the system searches for an existing model in the graph G. The following pseudo-code sketches the function `find-existing-model` that returns a model and a value `T`, if it is appropriate for the new data, or `F`, if it still requires repairing.

```
find-existing-model( effect, data, model )
   edges := get-edge-list( model );
   return search-edges( effect, data, model, edges );

search-edges( effect, data, model, edge-list )
   if( empty( edge-list ) ) then
      return [model, F];
   else
      edge := head( edge-list );
      if( matches( effect, data, get-edge-tag( edge ) ) ) then
         new-model := get-neighbor-model( model, edge );
         cluster := check-failure( new-model, data );
         if( cluster ) then
            new-effect := get-effect( cluster, data );
            new-edges := get-edge-list( new-model );
            new-edges := remove-elem-from-list( edge,
                                                new-edges );
            return search-edges( new-effect, data,
                                 new-model, new-edges );
         else
            return [new-model, T];
      else
         return search-edges( effect, data,
                              model, tail( edge-list ) );
```

The search proceeds recursively. It starts with the model that just failed as a current model and considers every edge connected to it. If the failure effect matches the pair `[cause, effect]` associated with the edge (when the failure effect matches the second element of the pair and the first element of the pair is supported by the data), then we progress along this edge to make the next model indicated to be our new current model. Otherwise, the next edge on the list is considered. If we progress to a new model, we must check whether or not it fails on the new data (using `check-failure`). If no failure is detected (no cluster node of the model has changed considerably), then we have found a model that represents the current context appropriately. Otherwise, we repeat this process for the new model. If none of the edges matches the failure, then a new model must be created.

Note that the mechanism described above is a *finite state automaton*. A state of this automaton is a current context represented by a current model. An edge between models corresponds to a transition in the automaton indicating a context transition event. The edge is described by a `[cause, effect]` pair that

has to be matched with current data and a current model for the transition to be enabled.

There is a difference, however, between a traditional finite state automaton and the mechanism we have presented. The set of all possible states, i.e., contexts, is not known in our case, but rather learned incrementally. Hence, when the system encounters an unseen situation, a new state and a new transition is added to the automaton. Assuming we have very detailed causal knowledge, if our system operates over the long term, the underlying finite state automaton together with the set of models representing the states will closely capture the intrinsic dependencies of the modeled world.

5 A detailed example

We now extend the burglar-alarm example. Using Loopy Logic notation, the initial model is specified by the following rules from our knowledge base:

```
Alarm|Burglary = [...]
Burglary|Location = [...]
Alarm|Rattling = [...]
Rattling|Location = [...]
Rattling|Earthquake = [...]
```

(where the probability distributions of these rules are not important).

In order to perform inference on the model, the system transforms the rules into a bipartite Markov random field (Fig. 3) consisting of two types of nodes: variable nodes (round) associated with the clauses of the rules, and cluster nodes (rectangular shape) storing conditional probability tables between variable nodes [14]. Let us assume that the model specified by the Markov random field in Fig. 3

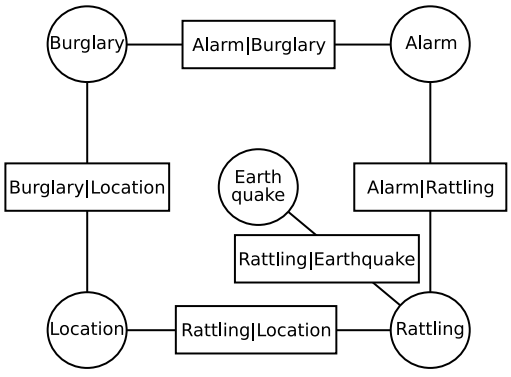

Fig. 3. A bipartite Markov random field generated from the rules of our knowledge base.

performs well on the data obtained during the summer when the temperature is regular. Let us denote this model as $\texttt{M}_{reg}$.

Let us next assume that the following Horn clauses [11] define the causal theory in this example. Note that the terms on both sides of $\Leftarrow$, such as `High-Temp`, are the descriptive predicates.

`Alarm-Off` $\Leftarrow$ `Broken-Circuit`	`Rattling` $\Leftarrow$ `Construction-Site`
`Broken-Circuit` $\Leftarrow$ `Broken-Window`	`Rattling` $\Leftarrow$ `Strong-Wind`
`Broken-Circuit` $\Leftarrow$ `Bad-Connections`	`Rattling` $\Leftarrow$ `Earthquake`
`Broken-Circuit` $\Leftarrow$ `Opened-Window`	`Rattling` $\Leftarrow$ `Thunder`
`Broken-Circuit` $\Leftarrow$ `Opened-Door`	`Rattling` $\Leftarrow$ `Noisy-Neighbor`
`Broken-Circuit` $\Leftarrow$ `Rattling`	`Bad-Connections` $\Leftarrow$ `Pest`
`Broken-Window` $\Leftarrow$ `Kids-Playing`	`Bad-Connections` $\Leftarrow$ `High-Temp`
`Broken-Window` $\Leftarrow$ `Burglary`	`Bad-Connections` $\Leftarrow$ `Corrosion`
`Opened-Window` $\Leftarrow$ `Burglary`	`High-Temp` $\Leftarrow$ `Hot-Summer`
`Opened-Door` $\Leftarrow$ `Burglary`	`Corrosion` $\Leftarrow$ `High-Age`

In the next sections we illustrate different situations in the system's operation as described by the previous pseudo-code. Two situations are considered: first the situation when a brand new context change occurs for which no suitable model exists, and second, the situation when a context change has been handled before and an appropriate model already exists in our system.

5.1 A new context change

Recall that our current model ($\texttt{M}_{reg}$) performs well on the data corresponding to regular summer temperature. We are going to model the situation when the temperature becomes irregularly high. New data (`hot-data`) that correspond to a high temperature situation do not fit the current model $\texttt{M}_{reg}$, thus $\texttt{M}_{reg}$ fails. This is captured by the function `check-failure` in the code:

`Alarm|Burglary` $\leftarrow$ `check-failure(`$\texttt{M}_{reg}$`, hot-data)`.

This function returns the cluster node `Alarm|Burglary` of the Markov random field (Fig. 3) indicating that the corresponding conditional probability distribution changed the most. Using the function `get-effect` on the cluster node `Alarm|Burglary` the system determines that multiple cases of the alarm going off without burglary is a failure effect:

`Alarm-Off` $\leftarrow$ `get-effect(Alarm|Burglary, hot-data)`.

The system then tries to find an existing model that is appropriate for the new data by matching the failure effect with the tags of the edges connected to our current model $\texttt{M}_{reg}$:

`[`$\texttt{M}_{reg}$`,F]` $\leftarrow$ `find-existing-model(Alarm-Off, hot-data,` $\texttt{M}_{reg}$`)`.

We further assume that there are no edges whose tags match the effect, hence the pair `[`$\texttt{M}_{reg}$`,F]` is returned (`F` indicates that no suitable models are found). Because no existing model was found appropriate for the new data, the system creates a new model $\texttt{M}_{hot}$ and trains it on the high temperature data:

$\texttt{M}_{hot} \leftarrow$ `create-new-model(Alarm-Off, hot-data,` $\texttt{M}_{reg}$`)`
$\texttt{M}_{hot} \leftarrow$ `update(`$\texttt{M}_{hot}$`, hot-data)`.

Consequently, the new model $\texttt{M}_{hot}$ becomes current.

Note that the function `create-new-model` involves searching for the explanation of a failure. Once the explanation is built, the new model is added to the global state transition mechanism by attaching it to the previous model with a new edge tagged with a `[cause,effect]` pair, `[High-Temp,Alarm-Off]` in this example.

The search for an explanation is done abductively. Initially, we start by adding the failure effect `Alarm-Off` to the queue `to-explain-queue` (which initially is empty) of those predicates whose explanation can possibly provide the explanation for the effect `Alarm-Off`:

`to-explain-queue` $\leftarrow$ `[Alarm-Off]`.

Then, we traverse the queue every time removing the first element. In this example the first element is the predicate `Alarm-Off` and it is not observed, i.e., there are no sensor data available to support or disprove the validity of this predicate. Consequently, the system obtaines the tails (conclusions) of those rules whose head matches the current element `Alarm-Off`:

`tail-list` $\leftarrow$ `[Broken-Circuit]`,

and adds them to the end of the queue:

`to-explain-queue` $\leftarrow$ `[Broken-Circuit]`.

This process iterates further and the predicate `Broken-Window` is produced from the queue. It is not observed, hence the system obtaines the list of the tails of the rules whose head matches the predicate `Broken-Window`:

`to-explain-queue` $\leftarrow$ `[Kids-Playing]`.

Note that we disregarded another tail, `Burglary`, because it is already represented in the Markov random field (Fig. 3) as a corresponding variable node, hence `Burglary` cannot be a cause of the failure.

After iterating through the queue further, the system encounters the predicate `Earthquake`. As compared to all the previous cases, this predicate is observed, i.e., there are sensors in the system that record data connected to the predicate. Consequently, the system checks whether this predicate is consistent with the data:

`False` $\leftarrow$ `consistent(Earthquake, hot-data)`.

In this example no earthquake happened, hence the data do not support the predicate `Earthquake`. Consequently, the system discards this predicate without matching it to the other causal rules.

Several iterations later, the system produces the predicate `High-Temp` from the queue. It is observed and consistent with the data. Hence the system records it as an explanation of the failure. The predicate `High-Temp` is then used for connecting the new model to the global structure. This predicate, together with the failure effect, defines the new context captured by the modeling system.

5.2 A context change encountered before

Let us now assume that after being hot for a while, the weather returns to its moderate condition. As in the previous case, new data (`reg-data`) obtained for this situation forces our current model $\texttt{M}_{hot}$ to fail. Recall that the model $\texttt{M}_{hot}$ was trained on the data when the temperature was high, this is why a very different dataset (`reg-data`) obtained after the temperature dropped does not fit $\texttt{M}_{hot}$.

Intuitively, what we want is to go back to the model $\texttt{M}_{reg}$. The system determines that the failure effect is the lower frequency of the false positive alarms (`Alarm-Off`) as before. Consequently, the system attempts to find an existing model that is relevant to the new data. It retrieves the list of edges connected to the model (pseudo-code in section 4). In this example only the edge tagged `[High-Temp,Alarm-Off]`, let us call it $\texttt{edge}_{hot}$, is attached to $\texttt{M}_{hot}$. However, more edges can be connected to a single model:

$\texttt{[edge}_{hot}\texttt{]} \leftarrow \texttt{get-edge-list(M}_{hot}\texttt{)}.$

The system considers the first element on the list ($\texttt{edge}_{hot}$) and attempts to match its tag `[High-Temp,Alarm-Off]` with the new data (`reg-data`) and the failure effect (`Alarm-Off`):

`True` $\leftarrow$ `matches(Alarm-Off, reg-data, [High-Temp,Alarm-Off])`.

The edge tag `[High-Temp,Alarm-Off]` indicates that the model on the other side of the edge from $\texttt{M}_{hot}$ failed because the high temperature caused false positive alarms and as a result the system constructed a new model, $\texttt{M}_{hot}$. Therefore, when the temperature decreases (the predicate `High-Temp` is no longer true) causing the amount of false positive alarms to reduce, we want to return to the model adjacent to $\texttt{M}_{hot}$ via $\texttt{edge}_{hot}$. The predicate `matches` does exactly that: it returns `True` when the current conditions, data, and a failure effect allow the transition from a current model to the adjacent model. Since, in this example, `matches` returns `True` on the $\texttt{edge}_{hot}$, the system obtains the model $\texttt{M}_{reg}$ adjacent to the current model $\texttt{M}_{hot}$ via $\texttt{edge}_{hot}$. Consequently, the system checks whether the new model $\texttt{M}_{reg}$ fails on the new data. In this example, $\texttt{M}_{reg}$ does not fail, therefore the system makes it current and updates it using the new data `reg-data`.

6 Related work

There are a number of research directions investigating the ways of combining deterministic logic and probabilistic modeling. The major goal of such a combination is to obtain both the representational power in logical worlds and the ability to cope with uncertainty in probabilistic worlds. Haddawy and Ngo [6, 12, 13] define a first-order probabilistic logic that is used to specify a class of networks in a knowledge base. They propose an approach for focussing on the relevant information of the knowledge base in order to reduce the complexity of computing with large knowledge bases.

Probabilistic relational models proposed by Friedman et al. [2] and extended by Getoor et al. [3] specify a probability model on classes of objects rather than on simple attributes. A probabilistic relational model can be used for any object from a class, since the model defines dependencies at the class level. Further, a probabilistic relational model is able to represent probabilistic dependencies between attributes of related objects due to its explicitly identified relational structure.

Kersting and DeRaedt [8] propose another framework, Bayesian logic programs, that attempts to combine Bayesian networks with Horn-clause logic. Bayesian logic programs use a knowledge-based model construction approach to generate query-specific Bayesian networks.

Richardson and Domingos propose [18] another probabilistic logic-based system, called Markov logic networks. As opposed to the previous approaches that use restricted subsets of the general logic, Markov logic networks use a general first-order logic. The logical sentences are converted into a conjunctive normal form and then mapped to a Markov network. Richardson and Domingos [18] propose a *complete* mapping from the first-order predicate calculus with function symbols to probability distributions.

Our computational context-based environment extends our earlier first-order, stochastic, and Turing complete "Loopy Logic" language [17]. The major difference between our stochastic system and the probabilistic logic-based modeling approaches described in this paper is the ability to model real-time data streams in dynamic environments that can have sudden changes. Most of the probabilistic logic-based modeling research assumes that the entire dataset is given and the modeling task is performed once. These modeling systems are not suitable for real-time tasks where new data is constantly obtained from the sensors.

Our modeling system employs a failure-driven architecture in order to capture contextual changes of the outside environment. This learning architecture is motivated by advances in research on cognitive development in psychology. Gopnik et al. [4] investigate the role of causal representation in several related types of learning mechanisms in humans. They argue that understanding causal dependencies allows humans to make wide-ranging predictions about future events and even to intervene in the world to trigger more events that are far distant from normal predictions. Gopnik et al. [4] argue that it is unlikely that people memorize large amounts of data and then apply a learning procedure to that data. Instead people use small data samples to form hypotheses. They then forget the data and revise their hypotheses as suggested by new data.

Granott et al. [5] define the notion of *bridging*, a mechanism that draws cognitive development toward more advanced and more stable levels. Bridging is accomplished by partially defined *shells* that are "scaffolds" directing the development of new knowledge by providing a viewpoint for processing new experiences. Granott et al. [5] argue that bridging is a transition mechanism that people use while learning, though it is different from hypothesizing, since bridging operates with unknown, not yet discovered objects.

Causality and the notion of context are the key concepts in our paper. The previous research involving context include the approach by Haddawy and Ngo [6, 12] to use context information to index the probabilistic information in order to reduce the size of the model. Logical expressions are used as contextual constraints to determine the applicability of a probabilistic rule from a knowledge base representing a class of Bayesian networks. This is the first research that explicitly defines and uses the notion of context in probabilistic logic-based modeling. It can be seen that the notion of causally informed context proposed in this paper is very different from the context defined by Haddawy and Ngo [6, 12]. We specify the context for a specific system and capture its evolution.

The importance of a causal model when searching for a good explanation of events under consideration is emphasized in many different papers [15, 7, 19]. Halpern and Pearl argue that the explanation must acknowledge the actual cause of events. They define the actual cause using the language of structural equations. Sakhanenko et al. [19] note that causal models can be used to specify context. They argue that depending on the underlying causal model there will be different explanations of a sequence of events. Hence, a causal structure is a natural candidate for the context specification. In his book, Pearl [15] shows several major advantages of causal models such as the modularity of causal structures, their stability to local data changes, and their ability to capture regularities of underlying data.

In this paper we employ abductive algorithms proposed by Stern and Luger [20]. In their abductive approach to diagnosis Stern and Luger use so-called *schemas* to guide the construction of explanations in domain knowledge. Our notion of context further extends the idea of schemas as a way to specify probabilistic reasoning for a special real-time task. In this paper we further defined contexts evolving over time.

7 Conclusions and future work

In this paper we formally define a notion for context that is used to improve probabilistic logic-based reasoning to be able to handle real-time diagnostic tasks. Model failure is recognized as a poor fit between a model and data used to determine a context transition event. Hence, we incrementally update the same model when a context stays the same and partition the model when the context changes. This is the basic idea behind a failure-driven architecture with incremental learning mechanisms proposed in this paper. The architecture is motivated by developmental learning theory hypothesizing that humans either assimilate new patterns into existing mental structures, or, if the pattern is inconsistent, reorganize the cognitive structures to accommodate the new pattern.

The main contribution of this paper is a mechanism for long-term learning of a context-partitioned library of models and a mechanism for switching between the models. The core of the mechanism is implemented as a finite state automaton whose states correspond to contexts represented by different models in the library and transitions are described by causal explanations of context transi-

tion events. We use abductive reasoning to generate a causal explanation from an underlying causal theory. A transition between models is done by matching a corresponding explanation with current data and model failure. If none of the transitions apply to a current situation, a new model is built and added to the automaton. Over the long term execution of the system, the library of models is populated such that it represents the evolution of context transitions and the conditions under which a transition event occurs. The corresponding incremental learning mechanisms are shown by pseudo-code. To clarify how these mechanisms operate we presented a detailed example.

There are a number of further research directions. One direction is to expand the formalism presented in section 2 to define the relationships between contexts and models. Moreover, a precise definition of a causal theory has to be given. We assume that causal dependencies determine current contexts. Investigating how context may influence causal structures is another research direction. Currently, our system uses no information about a model other than a failure event to construct the failure explanation. Using model information to investigate possible causal dependencies in the model to improve the search for an appropriate context while traversing a structure of the automaton is another direction of future research. Note that we have a deterministic notion of context, hence our system that partitions models depending on a context transition event captures only the structure of a context change. A promissing future direction is to expand the notion of context to accept stochastic context. This will potentially capture a long-term evolution of the underlying contextual structure.

8 Acknowledgements

We thank the National Science foundation (115-9800929, INT-9900485) and the US Navy (SBIR N00T001, STTR N0421-03-C-0041) for their earlier support in the design of our Turing-complete and first-order stochastic logic language, Loopy Logic. The research presented in this paper is funded under an Air Force Research Laboratory SBIR contract (FA8750-06-C0016). We would like to acknowledge Martin Walter at AFRL Rome for his supportive management and direction. We also thank Daniel Pless and Chayan Chakrabarti for many thought-provoking discussions.

References

1. Dempster, A., Laird, N., Rubin, D.: Maximum likelihood from incomplete data via the EM algorithm. J. Royal Statistical Society, Series B (Methodological) **39** (1977) 1–38
2. Friedman, N., Getoor, L., Koller, D., Pfeffer, A.: Learning Probabilistic Relational Models. Proc. International Joint Conference on Artificial Intelligence (1999) 1300–1307
3. Getoor, L., Friedman, N., Koller, D., Pfeffer, A.: Learning Probabilistic Relational Models. J. Relational Data Mining (2001) 307–335

4. Gopnik, A., Glymour, C., Sobel, D. M., Schulz, L. E., Kushnir, T., Danks, D.: A theory of causal learning in children: Causal maps and Bayes nets. J. Psychological Review **111** (2004) 3–32
5. Granott, N., Fischer, K. W., Parziale, J.: Bridging to the unknown: a transition mechanism in learning and development. Microdevelopment: transition processes in development and learning (2002)
6. Haddawy, P.: Generating Bayesian Networks from Probability Logic Knowledge Bases. Proc. Uncertainty in Artificial Intelligence (1994) 262–269
7. Halpern, J., Pearl, J.: Causes and Explanations: A Structural-Model Approach — Part 1: Causes. Proc. Uncertainty in Artificial Intelligence (2001) 194–202
8. Kersting, K., DeRaedt, L.: Bayesian Logic Programs. Tech. Report 00151, Albert-Ludwigs University at Freiburg (2000)
9. Laplace, P. S.: Essai philosophique sur les probabilites. Paris: Courcier (1814) reprinted Wiley (1912)
10. Leake, D. B.: Abduction, Experience, and Goals: a Model of Everyday Abductive Explanation. J. Experimental and Theoretical AI **7** (1995) 407–428
11. Luger, G. F.: Artificial intelligence: Structures and Strategies for Complex Problem Solving. Addison-Wesley (2004)
12. Ngo, L., Haddawy, P.: Answering queries from context-sensitive probabilistic knowledge bases. J. Theoretical Computer Science **171** (1997) 147–177
13. Ngo, L., Haddawy, P., Krieger, R. A., Helwig, J.: Efficient Temporal Probabilistic Reasoning via Context-Sensitive Model Construction. J. Computers in Biology and Medicine **27** (1997) 453–476
14. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, San Francisco, CA (1988)
15. Pearl, J.: Causality: Models, Reasoning, and Inference. Cambridge University Press (2000)
16. Piaget, J.: Piaget's theory. J. Handbook of Child Psychology **1** (1983)
17. Pless, D. J.: First-Order Stochastic Modeling. PhD Thesis, University of New Mexico (2003)
18. Richardson, M., Domingos, P.: Markov Logic Networks. J. Machine Learning **62** (2006) 107–136
19. Sakhanenko, N. A., Luger, G. F., Stern, C. R.: Managing dynamic contexts using failure-driven stochastic models. Proc. of 20th Conf. of FLAIRS (2007) to appear
20. Stern, C. R., Luger, G. F.: Abduction and abstraction in diagnosis: a schema-based account. J. Expertise in context (1997) 363–381