Efficient Probabilistic Inference for Dynamic Relational Models

Jonas Vlasselaer and **Wannes Meert** and **Guy Van den Broeck** and **Luc De Raedt**
Department of Computer Science, KU Leuven, Belgium
firstname.lastname@cs.kuleuven.be

Abstract

Over the last couple of years, the interest in combining probability and logic has grown strongly. This led to the development of different software packages like PRISM, ProbLog and Alchemy, which offer a variety of exact and approximate algorithms to perform inference and learning. What is lacking, however, are algorithms to perform efficient inference in relational temporal models by systematically exploiting temporal and local structure. Since many real-world problems require temporal models, we argue that more research is necessary to use this structure to obtain more efficient inference and learning. While existing relational representations of dynamic domains focus rather on approximate inference techniques we propose an exact algorithm.

Motivation

A dynamic relational model allows to compactly model a stochastic process by means of a relational or logic representation. Although temporal domains can be modeled in many statistical relational languages, only some representations have been proposed that treat time as a first-class citizen, e.g. CPT-L (Thon, Niels, and De Raedt 2011) and Relational Dynamic Bayesian Networks (RDBNs) (Manfredotti 2009). Hence, only some languages offer specific techniques for inference and learning in temporal models. Existing general-purpose implementations typically scale exponentially with the number of time steps.

There are two distinct properties of dynamic relational representations that can be exploited for efficient inference. First, the logical structure that is abundant in relational models leads to local structure in the probabilistic dependencies and determinism in the distribution. It is well-known that exploiting such properties leads to exponential speed gains (Darwiche 2009). Second, these models contain repetitive structures, both over time, and within one time slice.

To the best of our knowledge, no existing work exploits both of the aforementioned properties for exact inference. Most research on inference for (R)DBNs is centered around approximate techniques like (relational) particle filters and sampling. In this work, however, we focus on exact inference and propose a knowledge compilation approach to maximally exploit all the available structure: we compile a circuit once, and reuse it multiple times in the course of answering a single query.

Example Below we show a temporal model which handles the task of finding failures in a digital circuit (see Figure 1). The model is expressed in ProbLog[1], but any probabilistic logic language can be used. The goal is to compute the health state of a digital component at time T, i.e the probability the component is faulty, given a set of electrical inputs and outputs up until time τ. As such, the query is $P(\textbf{healthy}(G,T) \mid \textbf{input}(W,0{:}\tau), \textbf{output}(W,0{:}\tau))$.

```
wire(1). wire(2). wire(3). wire(4).
in(1). in(2). out(4).
gate(a,not,[1],3). gate(b,and,[3,2],4).

0.990::healthy(G,0) :- gate(G,_,_,_).
0.990::healthy(G,T) :- T>0,   healthy(G,T-1).
0.001::healthy(G,T) :- T>0, \+healthy(G,T-1).

0.5::high(W,T) :- in(W).
0.5::high(W,T) :- gate(G,_,_,W), \+healthy(G,T).
high(W,T) :- gate(G,not,[I],W), healthy(G,T), \+high(I,T).
high(W,T) :- gate(G,and,[I,J],W), healthy(G,T),
             high(I,T), high(J,T).

input(W,T) :- in(W), high(W,T).
output(W,T) :- out(W), high(W,T).
```

Inference in (R)DBNs

None of the deterministic facts in our logic program depend on *time* as the structure of a digital circuit typically does not change. Hence, the model defined above can be seen as a Relational Dynamic Bayesian Network (RDBN) which is fully defined by a pair of networks $(M_0, M_{t\to t+1})$ with M_0 a Relational Bayesian Network (RBN) that defines the prior state distribution, and $M_{t\to t+1}$ a two-time-slice RDBN that defines the transition model. In the transition model, every predicate at time t has a set of *parent predicates* at time $t-1$ or t. As the structure does not change over time, grounding out the relational DBN results in a propositional DBN which is completely defined by a pair of static BNs $(B_0, B_{t\to t+1})$.

One way to do inference in DBNs is to unroll the network and apply any algorithm for Bayesian Networks. This approach, however, is not guaranteed to scale linear with the

[1] http://dtai.cs.kuleuven.be/problog

number of time steps. To obtain an algorithm which does scale linear in time, one has to define a set of variables in the transition model which d-separate the past from the future. Typically, this set is called the *interface* and contains all variables at time t which have an outgoing arc to $t + 1$.

Exploiting Recursive Structure

The recursive structure in (R)DBNs, given by the transition model, in combination with the definition of the interface allows to perform inference by recursively computing the joint probability distribution of the interface, often referred to as the *forward* and *backward message* (Murphy 2002). We propose to use knowledge compilation (Darwiche 2009) to first compile the structure into an Arithmetic Circuit (AC) and, next, evaluate the obtained circuit multiple times to compute the messages. As the transition model remains static over time, it suffices to compile it only once. As such, this one time compilation cost is amortized over all evaluation steps necessary to compute the messages. One evaluation can be done in time polynomial in the size of the AC.

The recursive structure which we need to compile contains all nodes from the second time slice in the transition model together with all their parents in the first time slice. As this does not take into account the dependencies between the variables of the incoming interface, i.e. the variables from the first time-slice, we introduce a second compilation step. This step is necessary and conditions the obtained AC on all combinations of truth values of the incoming interface variables. Once this is done, the circuit is reused $N \cdot M$ times to recursively compute the messages, where $N = 2^I$ (with I the number of variables in the interface) and M is the number of time steps.

Exploiting Local Structure

Our proposed approach is closely related to the interface algorithm (Murphy 2002) where one constructs a junction tree for the recursive structure. Murphy's approach, however, only exploits conditional independences while our compilation approach also exploits local structure. Compiling a network results in an AC which is not necessary exponential in the treewidth. This allows inference in cases where this is impossible with other algorithms which do not exploit this structure. As relational models typically introduce determinism and equal parameters, our approach should outperform the classic implementation of the interface algorithm.

Proof of Concept

As a proof of concept we generated random digital circuits of varying size where we also modeled digital gates with more than two input wires. To vary both the complexity (size) of the digital circuit and the number of variables in the interface, we allowed multiple digital gates to have the same health state. Gates share a health variable, for example, when they share a power line.

We implemented our approach in ProbLog and did some preliminary experiments to compare it with the DBN algorithms implemented in the Bayesian Network Toolbox (bnt)[2]. The task was to compute the probability of the health predicates at $T = 10$, given evidence on the input and output of the circuit up until $\tau = 9$. The experiments were conducted on a computer with a working space of 4GB RAM.

We can drawn two conclusions from the results shown in figure 1. Firstly, the compilation approach is somewhat faster compared to bnt. Secondly, and more importantly, is that our approach allows inference in RDBNs for digital circuits that are almost three times larger compared to bnt as the latter already runs out of memory for models with more than about 30 digital gates. Also important to notice is that, although the total inference time takes approximately 120 seconds for the biggest circuit, the computation of the forward message (the recursive step) only takes approximately 0.5 seconds. As such, most of the time is spent in the (off-line) compilation step which only needs to be conducted once.

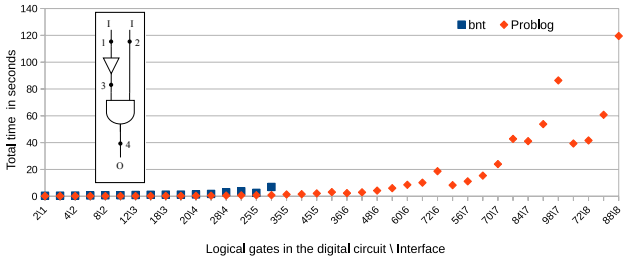

Figure 1: The digital circuit used as running example and the experimental results. The x axis shows the number of gates in the digital circuit and the number of ground atoms h in the interface at each time t. The y axis shows the total inference time in seconds (compilation + evaluation).

Conclusion and Challenges

We presented an algorithm for exact inference in relational DBNs, based on knowledge compilation, to exploit as well the recursive as the local structure in the model. The preliminary results show the promise of our technique for relational DBNs with closed domains. Two remaining challenges are (i) relational DBNs with open domains, i.e. where the number of ground atoms might change over time, and (ii) applying the insights from exploiting local structure to approximate inference techniques.

References

Darwiche, A. 2009. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press.

Manfredotti, C. 2009. Modeling and inference with relational dynamic Bayesian networks. In *Advances in artificial intelligence*. Springer. 287–290.

Murphy, K. 2002. *Dynamic Bayesian Networks: Representation, Inference and Learning*. Ph.D. Dissertation, UC Berkeley.

Thon, I.; Niels, L.; and De Raedt, L. 2011. Stochastic relational processes: Efficient inference and applications. *Machine Learning* 82(2):239–272.

[2] https://github.com/bayesnet/bnt