CP-Logic Theory Inference with Contextual Variable Elimination and Comparison to BDD Based Inference Methods

Wannes Meert, Jan Struyf and Hendrik Blockeel

Dept. of Computer Science, Katholieke Universiteit Leuven, Belgium
{Wannes.Meert, Jan.Struyf, Hendrik.Blockeel}@cs.kuleuven.be

Abstract. There is a growing interest in languages that combine probabilistic models with logic to represent complex domains involving uncertainty. Causal probabilistic logic (CP-logic), which has been designed to model causal processes, is such a probabilistic logic language. This paper investigates inference algorithms for CP-logic; these are crucial for developing learning algorithms. It proposes a new CP-logic inference method based on contextual variable elimination and compares this method to variable elimination and to methods based on binary decision diagrams.

1 Introduction

In many applications, the goal is to model the probability distribution of a set of random variables that are related by a causal process, that is, the variables interact through a sequence of non-deterministic or probabilistic events. Causal probabilistic logic (CP-logic) [1] is a probabilistic logic modeling language that can model such processes. The model takes the form of a CP-logic theory (CP-theory), which is a set of events in which each event is represented as a rule of the following form:

$$(h_1 : \alpha_1) \vee \ldots \vee (h_n : \alpha_n) \leftarrow b_1, \ldots, b_m.$$

with h_i atoms and b_i literals, and α_i *causal* probabilities; $0 < \alpha_i \leq 1$, $\sum \alpha_i \leq 1$. We call the set of all $(h_i : \alpha_i)$ the *head* of the event, and the conjunction of literals b_i the *body*. If the body of a CP-event evaluates to true, then the event will happen and cause at most one of the head atoms to become true; the probability that the event causes h_i is given by α_i (if $\sum \alpha_i < 1$, it is also possible that *nothing* is caused).

Example. The CP-theory

$$\begin{aligned} &shops(john) : 0.2. && (C_1)\\ &shops(mary) : 0.9. && (C_2)\\ &(spaghetti : 0.5) \vee (steak : 0.5) \leftarrow shops(john). && (C_3)\\ &(spaghetti : 0.3) \vee (fish : 0.7) \leftarrow shops(mary). && (C_4) \end{aligned}$$

Presented at the 19th International Conference on Inductive Logic Programming (ILP'09).

models the situation that John and his partner Mary may independently decide to go out to buy food for dinner. The causal probability associated with each meal indicates the probability that the fact that John (respectively Mary) goes shopping causes that particular meal to be bought.

CP-logic is closely related to other probabilistic logics such as ProbLog, Independent Choice Logic (ICL), Programming in Statistical Modelling (PRISM), and Bayesian Logic Programs (BLPs). Meert et al. [2] compares CP-logic to these other formalisms.

Since CP-logic was introduced, several inference methods have been proposed for (a subset of) CP-logic. The efficiency of these methods is crucial for developing fast parameter and structure learning algorithms [2]. In this paper, we propose a new CP-theory inference method and present an experimental comparison with known methods.

2 Known CP-Theory Inference Methods

Variable Elimination. Meert et al [2] describe a transformation that can transform any acyclic CP-theory with a finite Herbrand universe to an equivalent Bayesian network (EBN). Based on this transformation, CP-theory inference can be performed by applying the transformation on the given theory and then running a Bayesian network (BN) inference algorithm, such as variable elimination (VE), on the resulting EBN.

ProbLog. ProbLog [3] is a probabilistic logic programming language that can serve as a target language to which other probabilistic logic modeling languages can be compiled. In particular, acyclic CP-theories without negation can be translated into ProbLog.

ProbLog's inference engine works as follows. Given a query, it first computes all proofs of the query and collects these in a DNF formula. Next, it converts this formula to a binary decision diagram (BDD). Relying on this BDD representation, ProbLog then computes the query's probability in one bottom-up pass through the BDD (using dynamic programming).

cplint. Inspired by ProbLog, Riguzzi [4] proposes cplint, which is a CP-theory inference system that makes use of BDDs in a similar way as ProbLog. There are two differences with the transformation to ProbLog. First, cplint uses a different encoding to represent which head atom is caused by a CP-event. Second, cplint supports negation. When it encounters a negative body literal $\neg a$ in a proof, it computes all proofs of a

and includes the negation of the DNF resulting from all these proofs into the original DNF (note that this process is recursive).

3 Inference with Contextual Variable Elimination

As said before, a CP-theory can be transformed to an EBN [2], which can be represented by a set of factors (Fig. 1.a). A CP-theory may contain more structural information than a BN, and this structural information has to be encoded numerically in the factors of the EBN. This can result in factors with redundant information (a factor may have many identical columns) and cause suboptimal inference. This effect can be seen in the top-left factor of Fig. 1.a, which represents that *spaghetti* is true if John or Mary buys spaghetti. This factor has many identical columns.

To address this problem, we propose to use contextual variable elimination (CVE) [5], which is an extension to VE that exploits contextual independence to speed up inference by representing the joint probability distribution as a set of *confactors* instead of factors. Confactors can be more compact than factors because they explicitly encode structural information by means of so-called contexts.

A confactor i consists of two parts, a *context* and a *table*:

$$< \underbrace{v_1 \in V_{1i} \wedge \ldots \wedge v_k \in V_{ki} \wedge \ldots \wedge v_n \in V_{ni}}_{\text{context}}, \quad \underbrace{\mathit{factor}_i(v_k, \ldots, v_m)}_{\text{table}} >$$

The context is a conjunction of set membership tests ($v_j \in V_{ji}$, $V_{ji} \subseteq \mathit{domain}(v_j)$), which indicates the condition under which the table is applicable. The table stores probabilities for given value assignments for a set of variables ($v_k \ldots v_m$). In the original CVE algorithm, the context was limited to equality tests. Our implementation also allows set membership tests, which are required to concisely represent CP-theories (e.g., to represent the inequality tests in the contexts in Fig 1.b, left).

Converting a CP-theory to a set of confactors. We explain the transformation for the example given before (see Fig. 1.b):

1. For every CP-event, create a variable (called a *choice*) whose value indicates which head atom is chosen (e.g., C_4 indicates the fourth CP-event's choice). The probability distribution for this variable is represented by multiple confactors. The context of one confactor represents the case that the body is true. The other confactors constitute

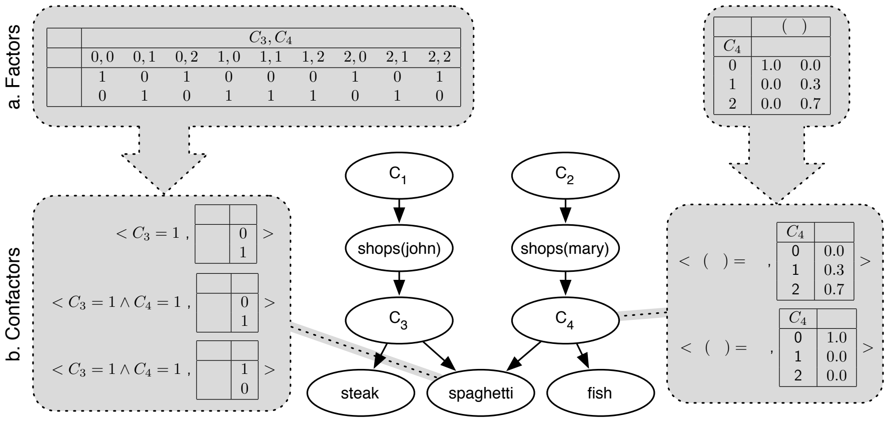

Fig. 1. Factor and confactor representation for node *spaghetti* (left) and C_4 (right)

the case that the body is false, and make the set of confactors complete and mutually exclusive.

For example, the first confactor for C_4 (Fig. 1, right) represents the case that the body $shops(mary)$ is true and the event chooses to make one of the head atoms true ($C_4 = 1$ for spaghetti, $C_4 = 2$ for fish). The other C_4 confactor corresponds to the case that $shops(mary)$ is false; in that case no head atom is caused ($C_4 = 0$).

2. For every atom in the theory, create a Boolean variable. The probability distribution of this variable is factorized in multiple confactors that together encode an OR-function (by means of the contexts). If at least one of the events where the atom is in the head has selected the atom, it becomes true; otherwise, it will be false.

The transformation above can be extended to improve inference efficiency. For example, we represent CP-events that have the same atoms in the head and mutually exclusive bodies by a single choice variable. Also, a factor is not split up if the resulting confactors are not more compact (in terms of the number of parameters) than the original factor (e.g., C_4 is not split into two confactors like in Fig. 1, but kept as a single factor). Once the set of confactors representing the CP-theory is constructed, we use the CVE algorithm [5] to perform CP-theory inference.

4 Results

We evaluate the inference methods on the task of inferring the marginal distribution of one designated variable in four CP-theories of varying

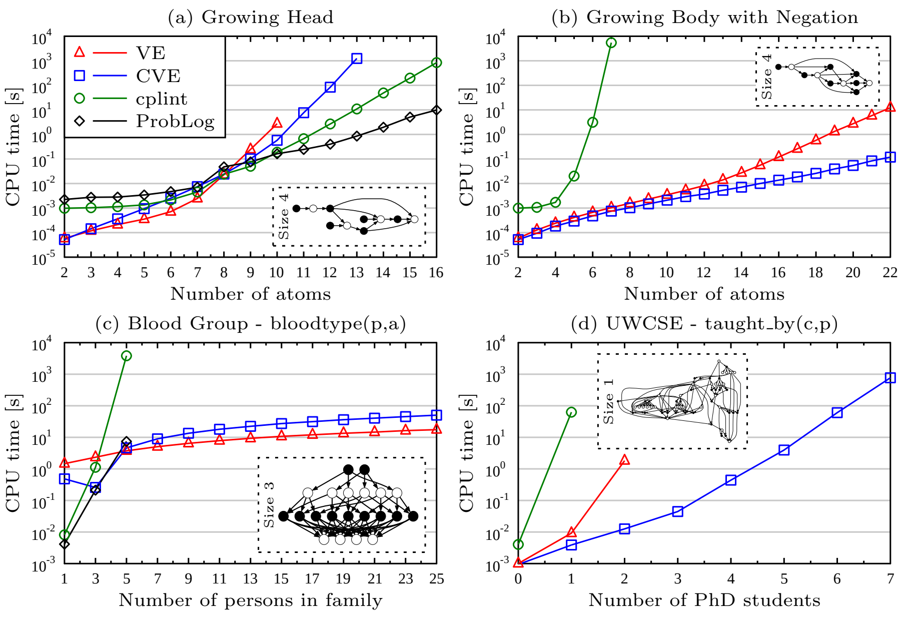

Fig. 2. Experimental results (including example EBNs for small theories).

complexity. We always select the variable with the highest inference cost and do not include any evidence (i.e., we consider the most difficult case). The theories are available at `http://www.cs.kuleuven.be/~dtai/cplve/ilp09`. Fig. 2 presents the results. Graphs (b) and (d) do not include results for ProbLog because they include negation.

For theory (a), the BDD based inference methods (cplint and ProbLog) are faster than CVE and VE for large problem instances. They are slower partly because they perform computations to calculate the probability that a variable is true, but also for the probability that it is false (separately). It is well known that for Noisy-AND, which occurs in the ProbLog program that theory (a) is transformed into, it is more efficient to only compute the probability P_T that its outcome is true and to calculate the probability that it is false as $1 - P_T$. An advantage of the BDD based methods is that they only compute the probability that an atom is true.

For theories (b)-(d), CVE and VE outperform the BDD based methods. For theory (b) and (d), this is partly due to the complexity of cplint's method for handling negation. A second reason for the lower performance of the BDD methods is the following. If the same atom is encountered multiple times in the proofs, then the DNF formula will contain an identical subexpression for each occurrence, and computing all these subexpressions will require repeatedly proving the same goal. Some of these redundant computations can be avoided by 'tabling' proofs [6, 7].

CVE outperforms VE for large problem instances on theories (a), (b), and (d). This is due to the compact representation with confactors instead of factors. VE runs out of memory after size 10 in (a) and size 2 in (d).

5 Conclusion and Future Work

We have proposed a new CP-theory inference method that transforms the given CP-theory to a set of confactors and then performs inference by running contextual variable elimination (CVE) on this representation. CVE outperforms VE with regard to both time and memory consumption for most large problem instances. Depending on the theory, CVE may also be faster than current BDD based methods.

In future work, we plan to incorporate (some of) the above inference methods into CP-theory learning algorithms. Second, we would like to investigate lifted inference for CP-theories. Known lifted-inference methods employ VE; we will try to extend this to CVE. A third item of interest is to investigate inference and learning methods for cyclic CP-theories.

Acknowledgments: Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen) to WM. Research Fund K.U.Leuven to JS. GOA/08/008 'Probabilistic Logic Learning'. The authors are grateful to A. Kimmig and D. Fierens for the fruitful discussions and to F. Riguzzi for his suggestions about cplint.

References

1. Vennekens, J., Denecker, M., Bruynooghe, M.: Representing causal information about a probabilistic process. Lecture Notes in Comp Sci **4160** (2006) 452–464
2. Meert, W., Struyf, J., Blockeel, H.: Learning ground CP-logic theories by leveraging Bayesian network learning techniques. Fundamenta Informaticae **89**(1) (2008) 131–160
3. Kimmig, A., Santos Costa, V., Rocha, R., Demoen, B., De Raedt, L.: On the efficient execution of ProbLog programs. Lecture Notes in Comp Sci **5366** (2008) 175–189
4. Riguzzi, F.: A top down interpreter for LPAD and CP-logic. Lecture Notes in Comp Sci **4733** (2007) 109–120
5. Poole, D., Zhang, N.: Exploiting contextual independence in probabilistic inference. J Artificial Intelligence Res **18** (2003) 263–313
6. Riguzzi, F.: SLGAD reolution for inference on logic programs with annotated disjunctions. J Algorithms in Logic, Informatics and Cognition (2009) (To appear).
7. Mantadelis, T., Janssens, G.: Tabling relevant parts of SLD proofs for ground goals in a probabilistic setting. In: 9th Int'l Coll. on Implementation of Constraint and Logic Programming Systems. (2009) (To appear).