CLP($\mathcal{BN}$): Constraint Logic Programming for Probabilistic Knowledge

Vítor Santos Costa
CRACS-INESC LA and DCC/FCUP
Portugal

CLP($\mathcal{BN}$) [3, 2] is an extension of Prolog that aims at integrating Bayesian variables within the logic programming framework. In this approach, random variables are represented as constraints on logical variables. Each constraint defines a probability distribution on the set of values that the variable may assume, or domain. Each constraint also has a name or skolen function that associates the logical variable with a random variable. As an example, consider:

```
ability(PKey,Abi) :-
   { Abi = ability(PKey) with tab([h,m,l],[0.5,0.4,0.1]) }.
```

the clause says that `Abi` can take three values, `h`, `m`, `l` with probabilities 0.5, 0.4 and 0.1. Moreover, the clause says that `Abi` is an instance of the random variable `ability(PKey)`.

CLP($\mathcal{BN}$) is designed to construct bayesian networks. We can encode that the popularity `Pop` depends on the ability `Abi` by writing:

```
popularity(PKey, Pop) :-
        ability(PKey, Abi),
        cpt(CPT)
        { Pop =  popularity(PKey) with tab([y,n],CPT,[Abi]) }.
```

Prolog execution will first call `ability/2` and obtain a constrained variable `Abi`. In the following execution steps, `Abi` becomes the *parent* for variable `Pop`, creating a small bayesian network. Notice that the value of `CPT` is encoded off-line for conciseness.

The user can introduce evidence on the values of random variables by using unification. The query

$$?-\texttt{popularity}(\texttt{X},\texttt{Pop}),\texttt{ability}(\texttt{X},\texttt{Abi}),\texttt{Pop} = \texttt{y}$$

will succeed, and the top-level will inform the user of the new distribution for `Abi`, given the evidence that `Pop` is `y`.

CLP($\mathcal{BN}$) is distributed with the YAP Prolog system, available at `http://www.dcc.fc.up.pt/~vsc/Yap`. The distribution includes several examples,

including a simulated school data-base. The system relies on the main principles of Knowledge Based Model Construction (KBMC), and it supports four difference methods: belief propagation, gibbs sampling, junction trees, and variable elimination [1].

The package includes an interface to the Inductive Logic Programming System Aleph that can be used to extend Prolog clauses into CLP($\mathcal{BN}$) clauses, simply by declaring special `random` types. Clause learning is performed by search, and distribution parameters are computed through EM.

References

[1] V. Santos Costa. On the implementation of the clp($\mathcal{BN}$) language. In M. Carro and R. Peña, editors, *Practical Aspects of Declarative Languages, 12th International Symposium, PADL 2010, Madrid, Spain, January 18-19, 2010. Proceedings*, volume 5937 of *Lecture Notes in Computer Science*, pages 234–248. Springer, 2010.

[2] V. Santos Costa, C. D. Page, and J. Cussens. *Probabilistic Inductive Logic Programming*, chapter CLP($\mathcal{BN}$): Constraint Logic Programming for Probabilistic Knowledge, pages 156–188. Springer-Verlag, 2008.

[3] V. Santos Costa, D. Page, M. Qazi, and J. Cussens. CLP($\mathcal{BN}$): Constraint Logic Programming for Probabilistic Knowledge. In *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence (UAI03)*, pages 517–524, Acapulco, Mexico, August 2003.