Extending the CLP engine for reasoning under uncertainty

Nicos Angelopoulos

nicos@cs.york.ac.uk

http://www.cs.york.ac.uk/˜nicos

Department of Computer Science

University of York

talk structure

- Logic Programming (LP)
- Uncertainty and LP
- Constraint LP
- clp(pfd(Y))
- clp(pfd(c))
- Caesar’s coding experiments

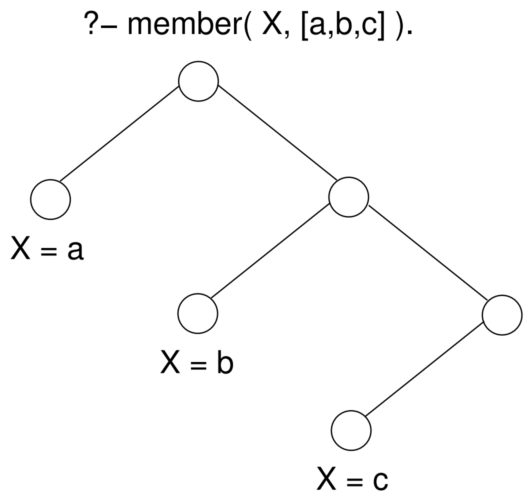
logic programming

Used in AI for crisp problem solving and for building executable models and intelligent systems.

Programs are formed from logic based rules.

```
member( H, [H|T] ).
member( El, [H|T] ) :- member( El, T ).
```

execution tree



```
member( H, [H|T] ).
member( El, [H|T] ) :- member( El, T ).
```

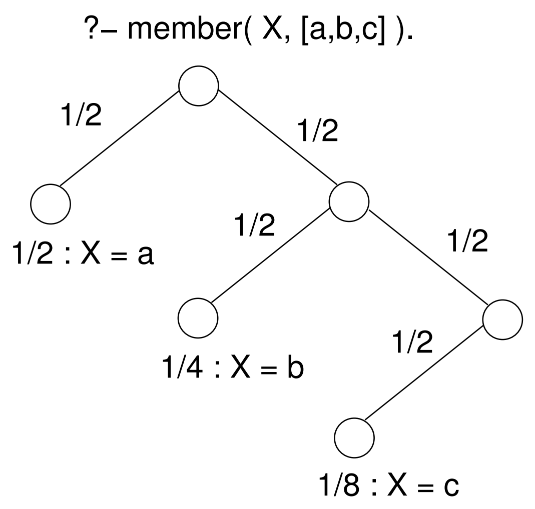
uncertainty in logic programming

Most approaches use Probability Theory but there are fundamental questions unresolved.

In general,

```
0.5 : member( H, [H|T] ).
0.5 : member( El, [H|T] ) :- member( El, T ).
```

stochastic tree



0.5 : member(H, [H|T]).

0.5 : member(El, [H|T]) :- member(El, T).

constraints in lp

Logic Programming :

- execution model is inflexible, and
- its relational nature discourages use of state information.

constraints in lp

Logic Programming :

- execution model is inflexible, and
- its relational nature discourages use of state information.

Constraints add

- specialised algorithms

constraints in lp

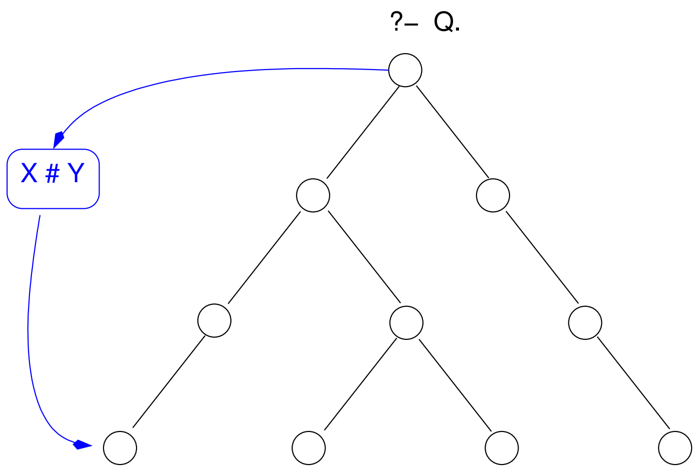
Logic Programming :

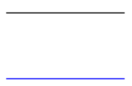
- execution model is inflexible, and
- its relational nature discourages use of state information.

Constraints add

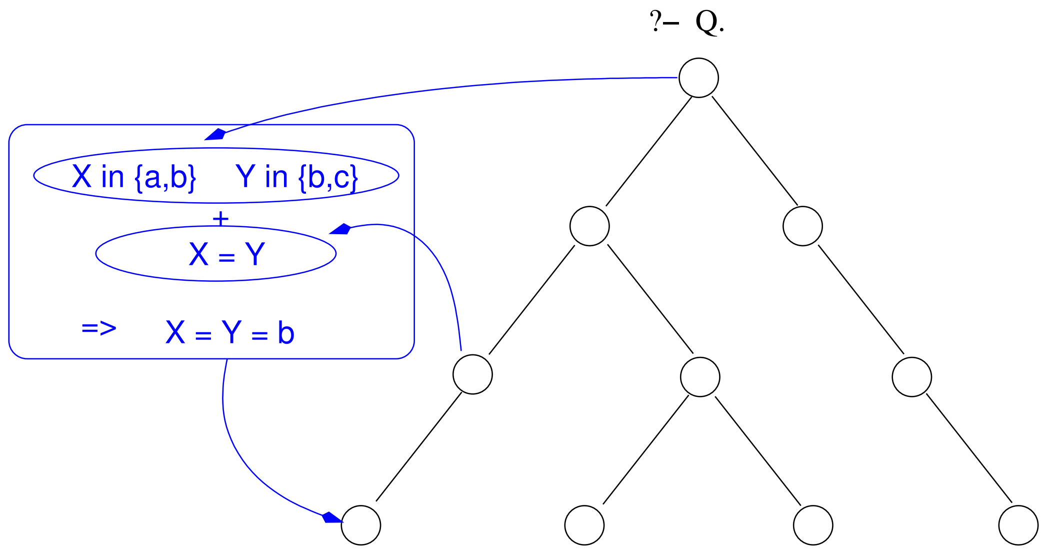
- specialised algorithms
- state information

constraint store





constraints inference



finite domain distributions

For discrete probabilistic models clp(pfd(Y)) extends the idea of finite domains to admit distributions.

from clp(fd)

$X\ in\ \{a, b\}$ (i.e. $X = a$ or $X = b$)

to clp(pfd(Y))

$$p(X = a) + p(X = b)$$

finite domain distributions

For discrete probabilistic models clp(pfd(Y)) extends the idea of finite domains to admit distributions.

from clp(fd)

$$X\ in\ \{a, b\} \qquad \text{(i.e. } X = a \quad \text{or} \quad X = b)$$

to clp(pfd(Y))

$$[\ p(X = a) + p(X = b)\ \ ] = 1$$

constraint based integration

Execution, assembles the probabilistic model in the store according to program and query.

Dedicated algorithms can be used for probabilistic inference on the model present in the store.

probability of predicates

- $pvars(E)$ - variables in predicate E,
- e - vector of finite domain elements
- $p(e_i)$ - probability of element e_i
- $\mathcal{S}$ - a constraint store.
- E/e - E with variables replaced by e.

The probability of predicate E with respect to store $\mathcal{S}$ is

$$P_{\mathcal{S}}(E) = \sum_{\substack{\forall e \\ \mathcal{S} \vdash E/e}} P_{\mathcal{S}}(e) = \sum_{\substack{\forall e \\ \mathcal{S} \vdash E/e}} \prod_i p(e_i)$$

clp(pfd(c))

Probabilistic variable definitions

$$Coin \in_{\mathbf{p}} finite\_geometric([head, tail], 2)$$

Conditional constraint

$Dependent$ [illegible] $Qualifier$

labelling

In clp(fd) systems labelling instantiates a group of variables to elements from their domains.

In clp(pfd(Y)) the probabilities can be used to guide labelling. For instance we have implemented

$$label(Vars, mua, Prbs, Total)$$

Selector mua approximates best-first algorithm which instantiates a group of variables to most likely combinations.

Caesar's coding

Each letter is encrypted to a random letter. Words drawn from a dictionary are encrypted. Programs try to decode them. We compared a clp(fd) solution to clp(pfd(c)).

clp(fd) no probabilistic information, labelling in lexicographical order.

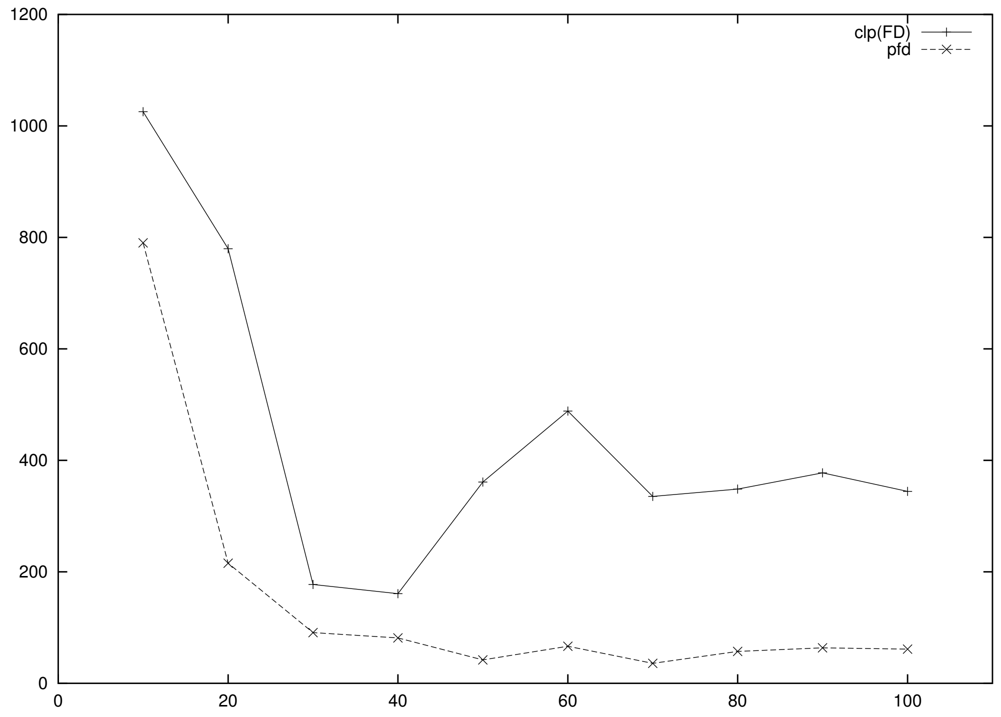
clp(pfd(c)) distributions based on frequencies, labelling using mua.

proximity functions

- X_i - variable for ith encoded letter
- D_i - dictionary letter
- freq() - frequency of letter

$$px(X_i, D_j) = \frac{1/\mid freq(X_i) - freq(D_j)\mid}{\sum_k 1/\mid freq(X_i) - freq(D_k)\mid}$$

execution times



clp(pfd(c)) and clp(fd) on SICStus 3.8.6

bottom line

Constraint LP based techniques can be used for frameworks that support probabilistic problem solving.

clp(pfd(Y)) can be used to take advantage of probabilistic information at an abstract level.

Monty Hall

Three curtains hiding a car and two goats.
Contestant chooses an initial curtain.
A close curtain opens to reveal a goat.
Contestant is asked for their final choice.

What is the best strategy ?
Stay or Switch ?

Monty Hall solution

If probability of switching is Swt,
($Swt = 0$ for strategy $Stay$ and $Swt = 1$ for $Switch$)

then probability of win is $P(\gamma) = \frac{1+Swt}{3}$.

Monty Hall in clp(cfd(c))

curtains(gamma, Swt, Prb) :-

$Gift \in_{\mathbf{p}} uniform([a, b, c])$,

$First \in_{\mathbf{p}} uniform([a, b, c])$,

$Reveal \in_{\mathbf{p}} uniform([a, b, c])$,

$Second \in_{\mathbf{p}} uniform([a, b, c])$,

$Reveal \neq Gift, Reveal \neq First, Second \neq Reveal$,

$Second \ \mathbf{I}\ddagger_{Swt} \ First$,

$Prb \ is \ \mathbf{p}(Second{=}Gift)$.

Strategy γ Query

Querying this program

```
?- curtains(gamma, 1/2, Prb)
```

$Prb = 1/2$.

Strategy γ Query

Querying this program

```
?- curtains(gamma, 1/2, Prb)
```

$Prb = 1/2$.

```
?- curtains(gamma, 1, Prb)
```

$Prb = 2/3$.

Strategy γ Query

Querying this program

```
?- curtains(gamma, 1/2, Prb)
```

$Prb = 1/2.$

```
?- curtains(gamma, 1, Prb)
```

$Prb = 2/3.$

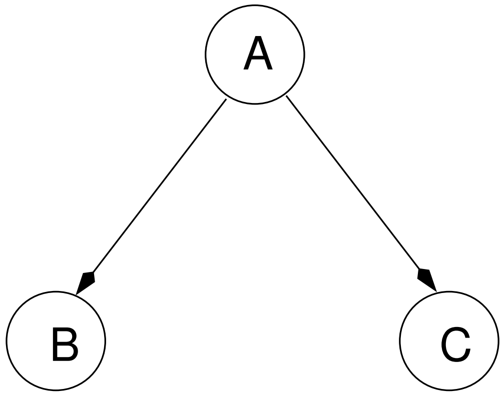
```
?- curtains(gamma, 0, Prb)
```

$Prb = 1/3.$

clp(pfd(BN))

Other discrete probabilistic inference is possible. For instance Bayesian Networks representation and inference.

example BN


$$\begin{tikzcd} & A \arrow[dl] \arrow[dr] & \\ B & & C \end{tikzcd}$$

	A = y	A = n
B = y	0.80	0.10
B = n	0.20	0.90

	A = y	A = n
C = y	0.60	0.90
C = n	0.40	0.10

clp(pfd(BN)) program

```
example_bn( A, B, C ) :-
    cpt(A,[],[y,n]),
    cpt(B,[A],[(y,y,0.8),(y,n,0.2),
               (n,y,0.1),(n,n,0.9)]),
    cpt(C,[A],[(y,y,0.6),(y,n,0.4),
               (n,y,0.9),(n,n,0.1)]).
```

program

```
example_bn( A, B, C ) :-
    cpt(A,[],[y,n]),
    cpt(B,[A],[(y,y,0.8),(y,n,0.2),
               (n,y,0.1),(n,n,0.9)]),
    cpt(C,[A],[(y,y,0.6),(y,n,0.4),
               (n,y,0.9),(n,n,0.1)]).

?- example_bn(X,Y,Z),
   evidence(X,[(y,0.8),(n,0.2)],
   Zy is p(Z = y).
```

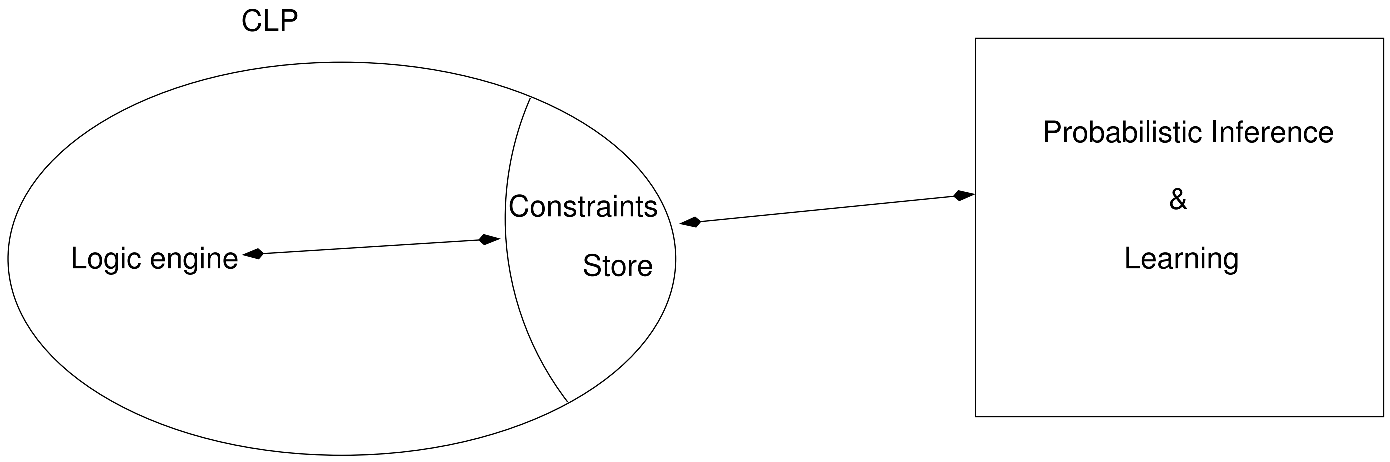
program

```
example_bn( A, B, C ) :-
        cpt(A,[],[y,n]),
        cpt(B,[A],[(y,y,0.8),(y,n,0.2),
                   (n,y,0.1),(n,n,0.9)]),
        cpt(C,[A],[(y,y,0.6),(y,n,0.4),
                   (n,y,0.9),(n,n,0.1)]).
```

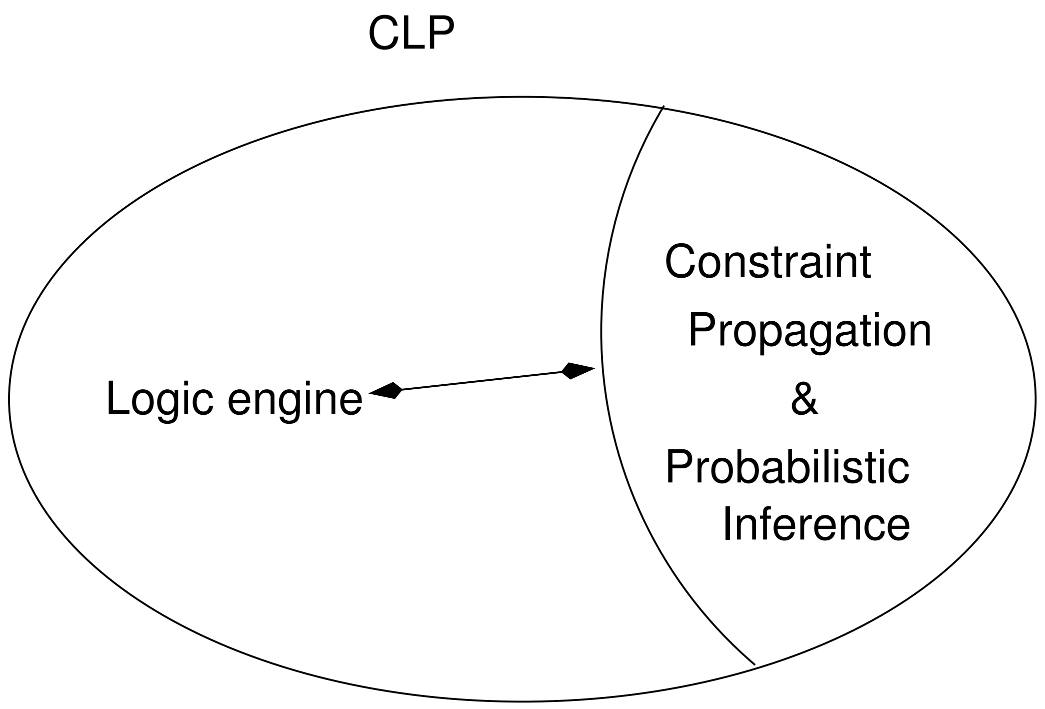
```
?- example_bn(X,Y,Z),
   evidence(X,[(y,0.8),(n,0.2)],
   Zy is p(Z = y).
```

Zy = 0.66

current inference scheme



bottom line


CLP
Logic engine
Constraint
Propagation
&
Probabilistic
Inference