Graphical models in Constraint Logic Programming

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
- Uncertainty
- Constraints
- Graphical Models integration
- Example
- Interfacing

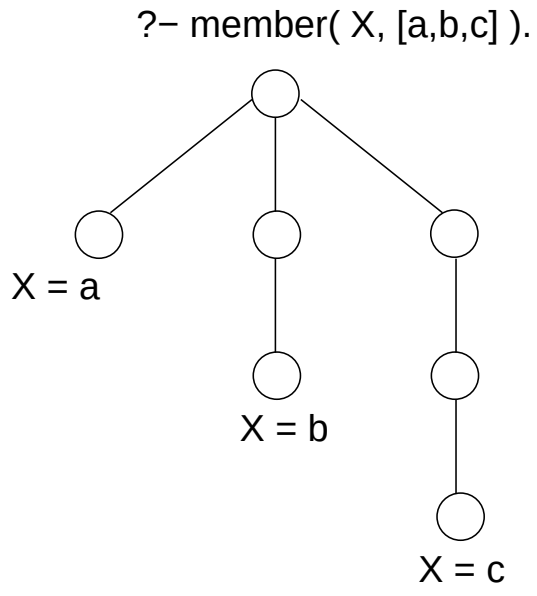
logic programming

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

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).

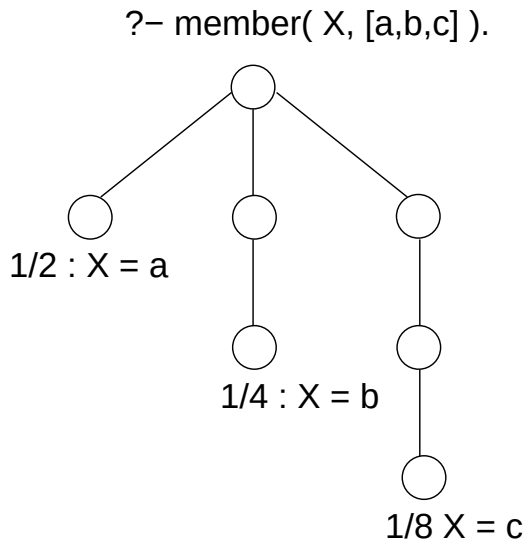
uncertainty in logic programming

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

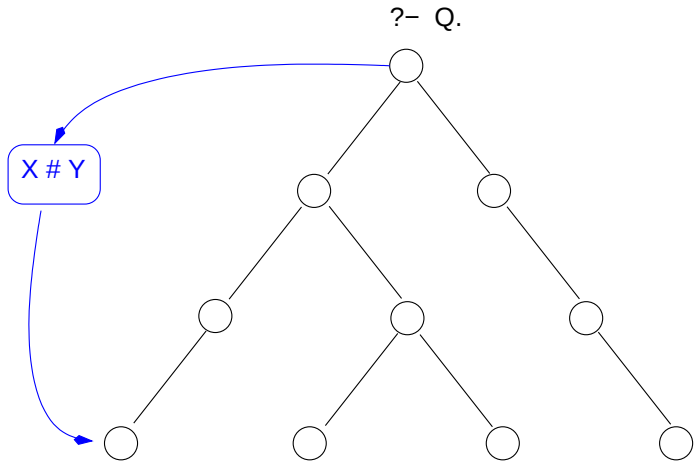
Logic Programming :

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

Constraints add

- specialised algorithms
- state information

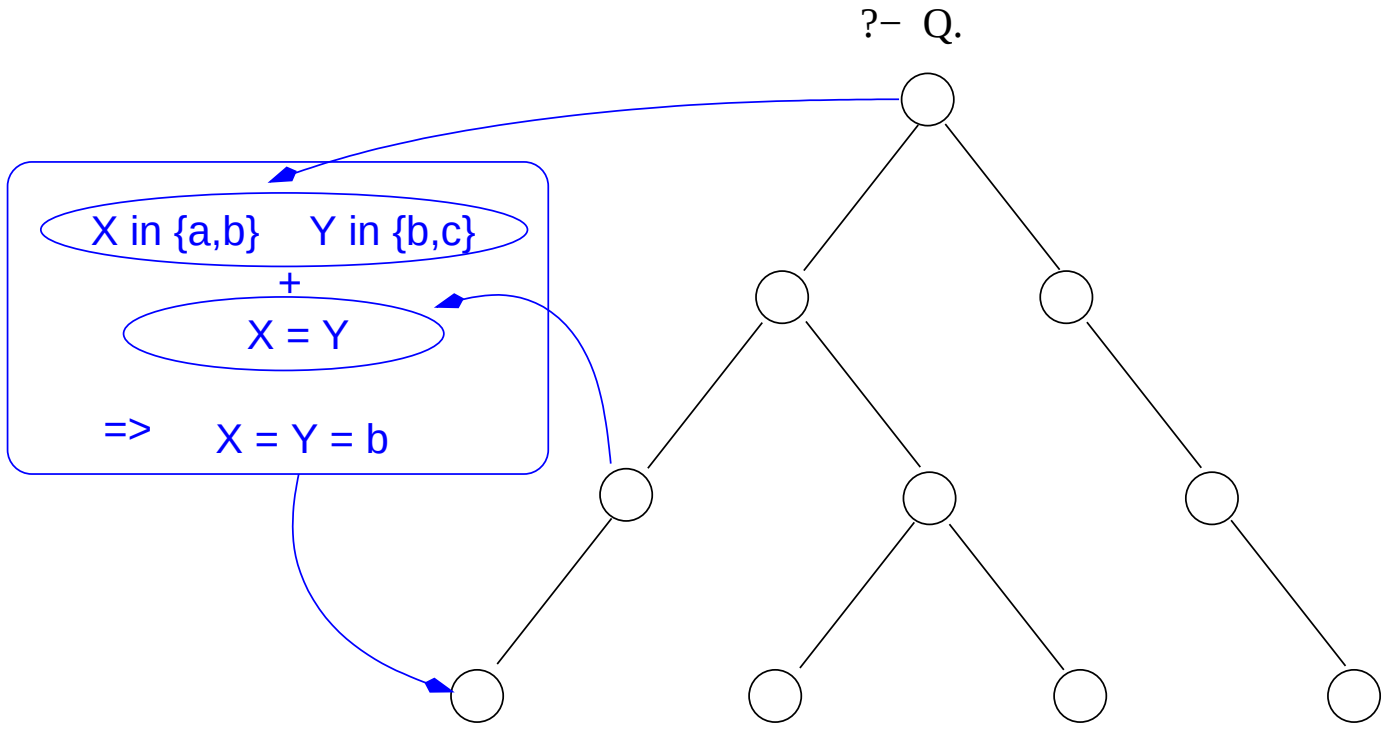
constraint store



Logic Programming engine

Constraint store interaction

constraints inference



finite domain distributions

For discrete graphical models: extend the idea of finite domains to admit distributions.

from

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

to

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

finite domain distributions

For discrete graphical models: extend the idea of finite domains to admit distributions.

from

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

to

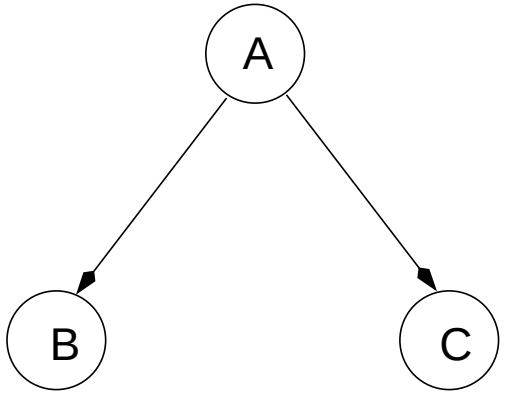
$$[\; p(X = a) + p(X = b) \;] = 1$$

GMs integration

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

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

example



	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

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)],
   Za is p(Z = a).
```

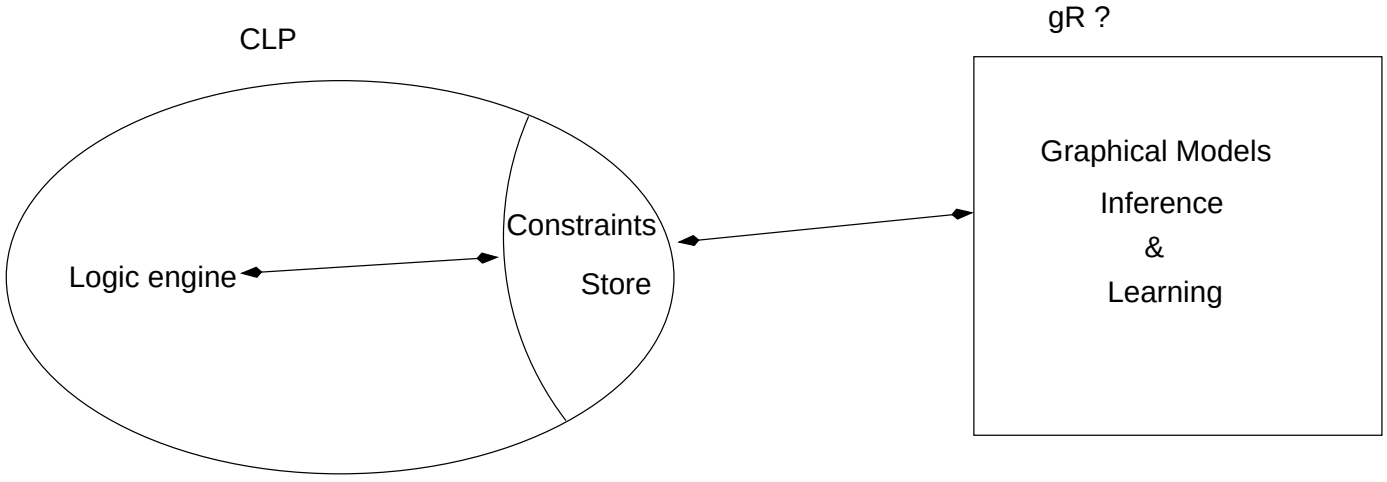
program

```
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)],
   Za is p(Z = y).
```

Za = 0.66

interactions



bottom line

Software properties:

- implementation of algorithms in literature
- open-source
- standarised data structures
- API
- session management and communication
- well characterised computational behaviour
- . . . documentation