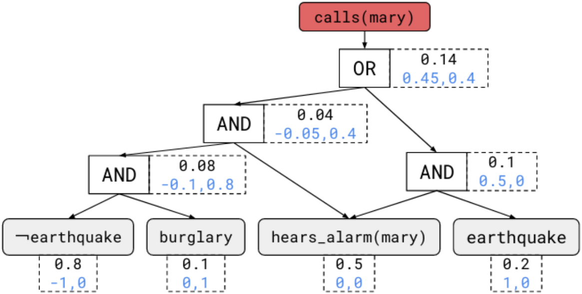
Probabilistic Logic Programming

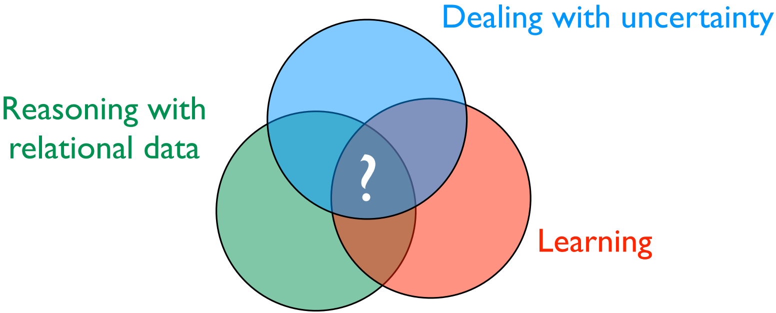


(source: https://arxiv.org/pdf/1805.10872.pdf)

Probabilistic Logic Programming

(Source for next slides: https://logic-data-science.github.io/Slides/DeRaedt.pdf - excellent presentation by de Raedt and Kimmig)

A key question in AI:

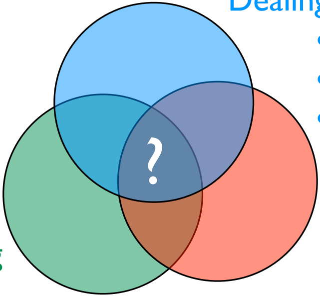


Probabilistic Logic Programming

A key question in AI:

Dealing with uncertainty
- probability theory
- graphical models
- ...

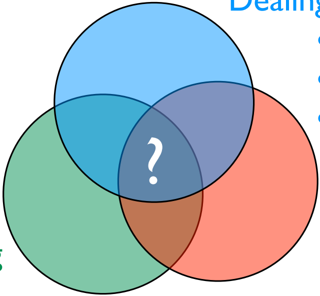
Reasoning with relational data
- logic
- databases
- programming
- ...



Learning
- parameters
- structure

Probabilistic Logic Programming

A key question in AI:



Dealing with uncertainty
- probability theory
- graphical models
- ...

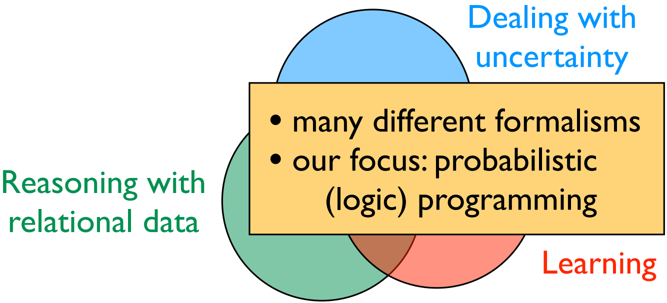
Reasoning with relational data
- logic
- databases
- programming
- ...

Learning
- parameters
- structure

Statistical relational learning, probabilistic logic learning, probabilistic programming, ...

Probabilistic Logic Programming

Common theme

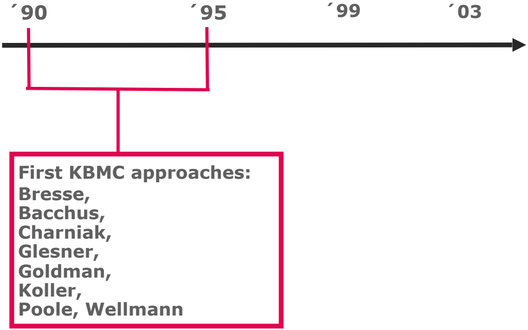


Statistical relational learning, probabilistic logic learning, probabilistic programming, ...

Probabilistic Logic Programming

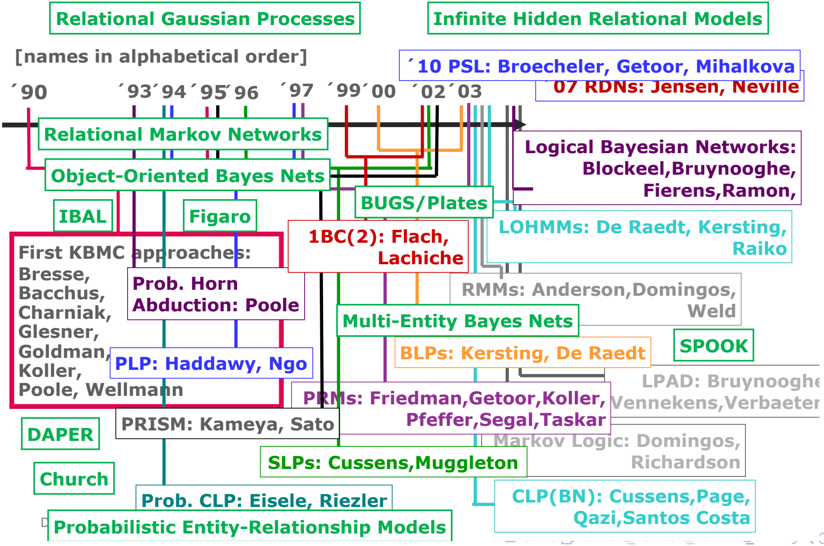
The (Incomplete) SRL Alphabet Soup

[names in alphabetical order]



Probabilistic Logic Programming

The (Incomplete) SRL Alphabet Soup



Probabilistic Logic Programming

Probabilistic Logic Programs

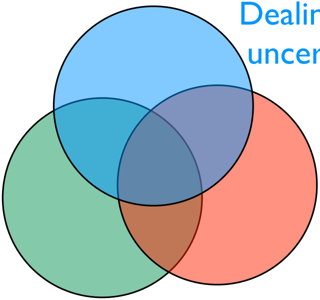
- devised by Poole and Sato in the 90s.
- built on top of the *programming language* Prolog
- upgrade *directed* graphical models
  - combines the advantages / expressive power of programming languages (Turing equivalent) and graphical models
- Generalises probabilistic databases (Suciu et al.)

- Implementations include: PRISM, ICL, ProbLog, LPADs, CP-logic, Dyna, Pita, DC, …

Probabilistic Logic Programming

ProbLog

probabilistic Prolog



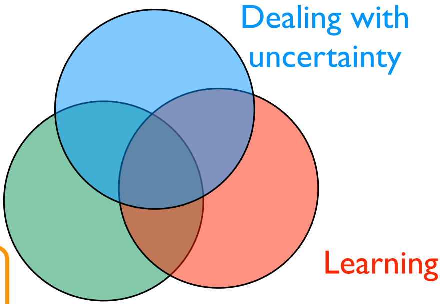

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X), smokes(Y).
```


http://dtai.cs.kuleuven.be/problog/

Probabilistic Logic Programming

ProbLog

probabilistic Prolog



Prolog / logic programming

```
stress(ann).
influences(ann,bob).
influences(bob,carl).
```

one world

```
smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X), smokes(Y).
```

http://dtai.cs.kuleuven.be/problog/

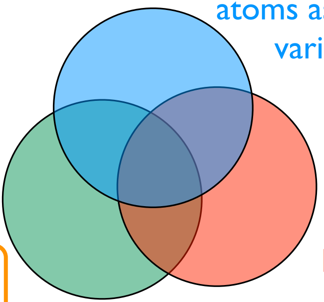
Probabilistic Logic Programming

ProbLog

probabilistic Prolog

```
0.8::stress(ann).
0.6::influences(ann,bob).
0.2::influences(bob,carl).
```

atoms as random variables



Prolog / logic programming

Learning

```
stress(ann).
influences(ann,bob).
influences(bob,carl).
```

one world

```
smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X), smokes(Y).
```

http://dtai.cs.kuleuven.be/problog/

Probabilistic Logic Programming

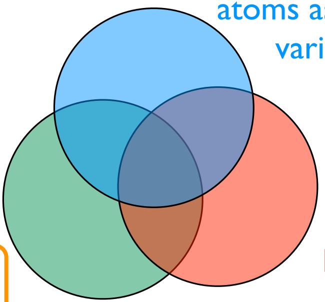
ProbLog

probabilistic Prolog

several possible worlds

```
0.8::stress(ann).
0.6::influences(ann,bob).
0.2::influences(bob,carl).
```

atoms as random variables



Prolog / logic programming

Learning

```
stress(ann).
influences(ann,bob).
influences(bob,carl).
```

one world

```
smokes(X) :- stress(X).
smokes(X) :-
    influences(Y,X), smokes(Y).
```


http://dtai.cs.kuleuven.be/problog/

Probabilistic Logic Programming

ProbLog

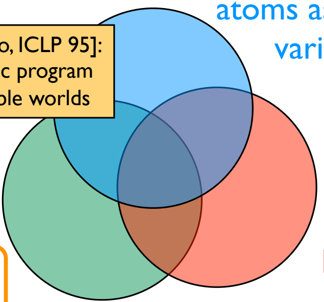
probabilistic Prolog

several possible worlds

```
0.8::stress(ann).
0.6::influences(ann,bob).
0.2::influences(bob,carl).
```

atoms as random variables

Distribution Semantics [Sato, ICLP 95]:
probabilistic choices + logic program
→ distribution over possible worlds



Prolog / logic programming

Learning

```
stress(ann).
influences(ann,bob).
influences(bob,carl).
```

one world

```
smokes(X) :- stress(X).
smokes(X) :-
    influences(Y,X), smokes(Y).
```


http://dtai.cs.kuleuven.be/problog/

Probabilistic Logic Programming

ProbLog

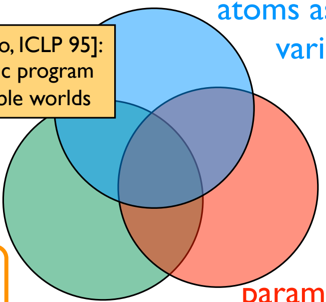
probabilistic Prolog

several possible worlds

```
0.8::stress(ann).
0.6::influences(ann,bob).
0.2::influences(bob,carl).
```

atoms as random variables

Distribution Semantics [Sato, ICLP 95]:
probabilistic choices + logic program
→ distribution over possible worlds



Prolog / logic programming

```
stress(ann).
influences(ann,bob).
influences(bob,carl).
```

one world

```
smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X), smokes(Y).
```

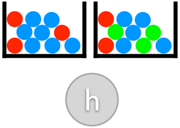
parameter learning, adapted relational learning techniques

http://dtai.cs.kuleuven.be/problog/

Probabilistic Logic Programming

ProbLog by example:

A bit of gambling

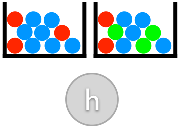


- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

Probabilistic Logic Programming

ProbLog by example:

A bit of gambling



- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

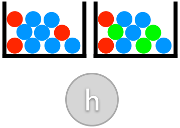
```
0.4 :: heads.
```

probabilistic fact: heads is true with probability 0.4 (and false with 0.6)

Probabilistic Logic Programming

ProbLog by example:

A bit of gambling



- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

```
0.4 :: heads.
```

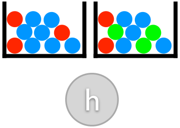
annotated disjunction: first ball is red with probability 0.3 and blue with 0.7

```
0.3 :: col(1,red); 0.7 :: col(1,blue).
```

Probabilistic Logic Programming

ProbLog by example:

A bit of gambling



- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

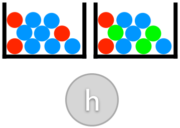
```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green);
                   0.5 :: col(2,blue).
```

annotated disjunction: second ball is red with probability 0.2, green with 0.3, and blue with 0.5

Probabilistic Logic Programming

ProbLog by example:

A bit of gambling



- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

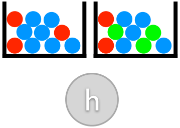
```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green);
                   0.5 :: col(2,blue).

win :- heads, col(_,red).
```

logical rule encoding background knowledge

Probabilistic Logic Programming

ProbLog by example:

A bit of gambling



- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

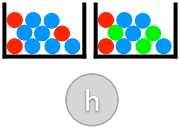
```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green);
                   0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

logical rule encoding background knowledge

Probabilistic Logic Programming

ProbLog by example:

A bit of gambling



- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

```
0.4 :: heads.                              probabilistic choices

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green);
                   0.5 :: col(2,blue).
```

```
win :- heads, col(_,red).                  consequences
win :- col(1,C), col(2,C).
```

Probabilistic Logic Programming

Questions

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

- Probability of **`win`**?
- Probability of **`win`** given **`col(2,green)`**?
- Most probable world where **`win`** is true?

Probabilistic Logic Programming

De Raedt, Kersting, Natarajan, Poole: Statistical Relational AI

Questions

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

marginal probability

- Probability of **`win`**?

query

- Probability of **`win`** given **`col(2,green)`**?
- Most probable world where **`win`** is true?

Probabilistic Logic Programming

Questions

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

marginal probability

- Probability of **`win`**?

conditional probability

- Probability of **`win`** given $\boxed{\texttt{col(2,green)}}$?

evidence

- Most probable world where **`win`** is true?

Probabilistic Logic Programming

De Raedt, Kersting, Natarajan, Poole: Statistical Relational AI

Questions

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

marginal probability

- Probability of **`win`**?

conditional probability

- Probability of **`win`** given **`col(2,green)`**?

- Most probable world where **`win`** is true?

MPE inference

Probabilistic Logic Programming

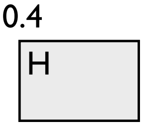
Possible Worlds

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

Probabilistic Logic Programming

Possible Worlds

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```



Probabilistic Logic Programming

Possible Worlds

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

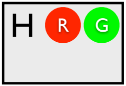
0.4×0.3

Probabilistic Logic Programming

Possible Worlds

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

$0.4 \times 0.3 \times 0.3$

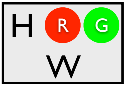


Probabilistic Logic Programming

Possible Worlds

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

$0.4 \times 0.3 \times 0.3$

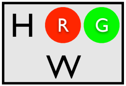


Probabilistic Logic Programming

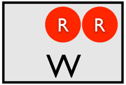
Possible Worlds

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue) <- true.
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue) <- true.

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

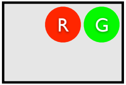
$0.4 \times 0.3 \times 0.3$



$(1-0.4) \times 0.3 \times 0.2$



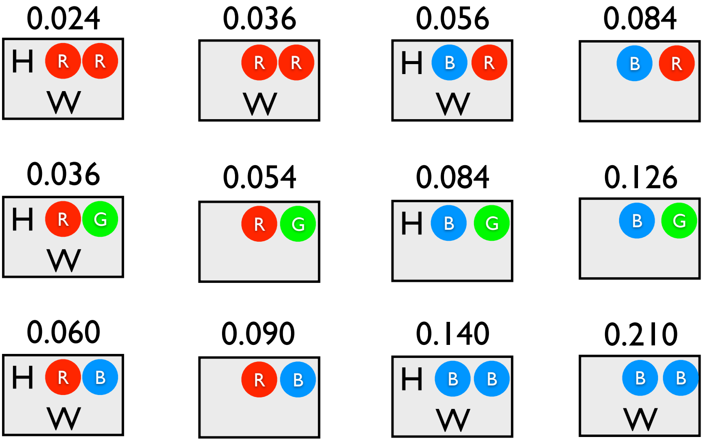
$(1-0.4) \times 0.3 \times 0.3$



Probabilistic Logic Programming

De Raedt, Kersting, Natarajan, Poole: Statistical Relational AI

All Possible Worlds

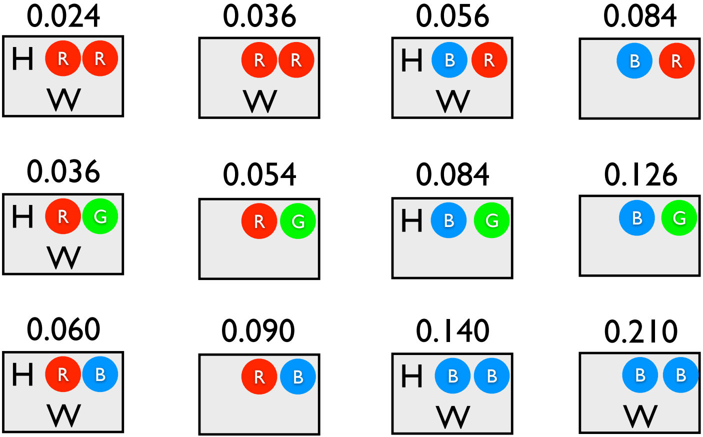


Probabilistic Logic Programming

De Raedt, Kersting, Nat elational AI

Most likely world where **`win`** is true?

MPE Inference



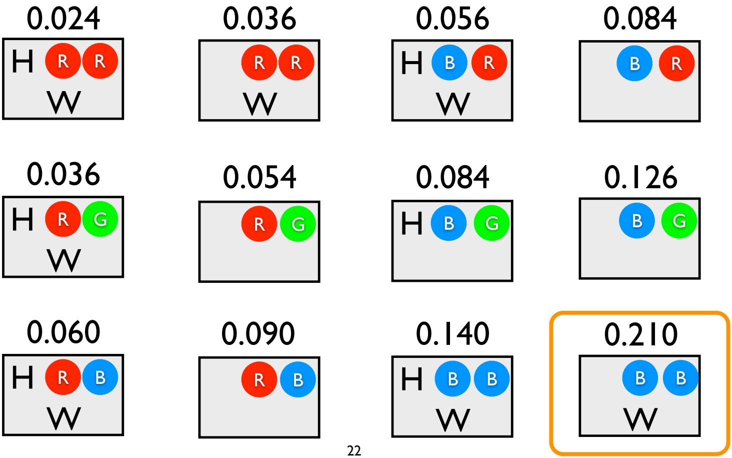
22

Probabilistic Logic Programming

De Raedt, Kersting, Nat … Relational AI

MPE Inference

Most likely world where **`win`** is true?



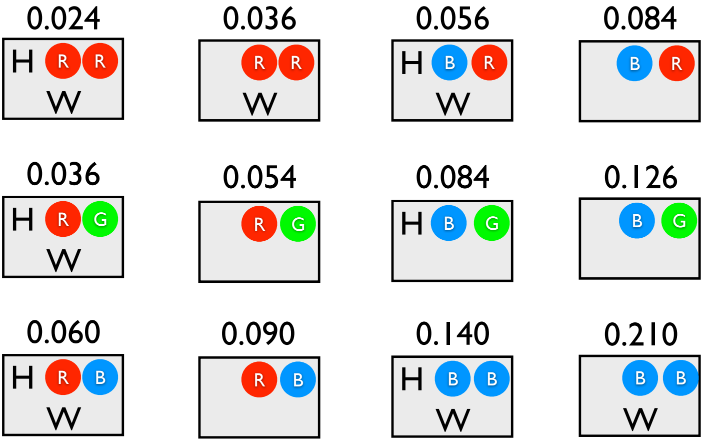
22

Probabilistic Logic Programming

De Raedt, Kersting, Natara... ...tical Relational AI

Marginal Probability

P(win)=?

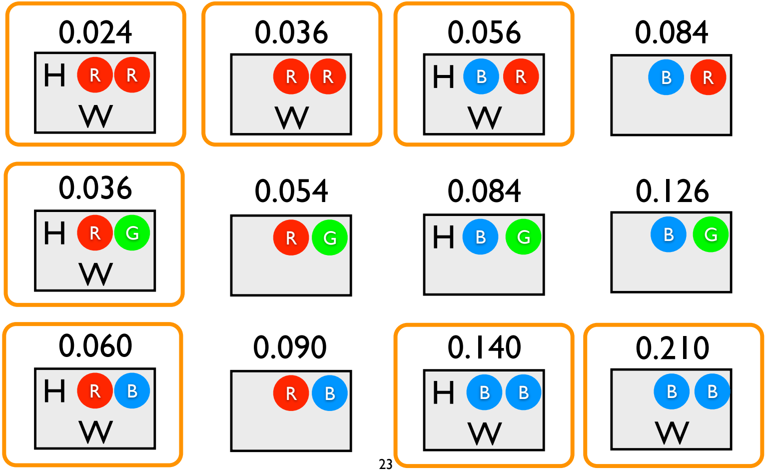


Probabilistic Logic Programming

De Raedt, Kersting, Natara… …ical Relational AI

$$P(\underline{\text{win}}) = \Sigma$$

Marginal Probability

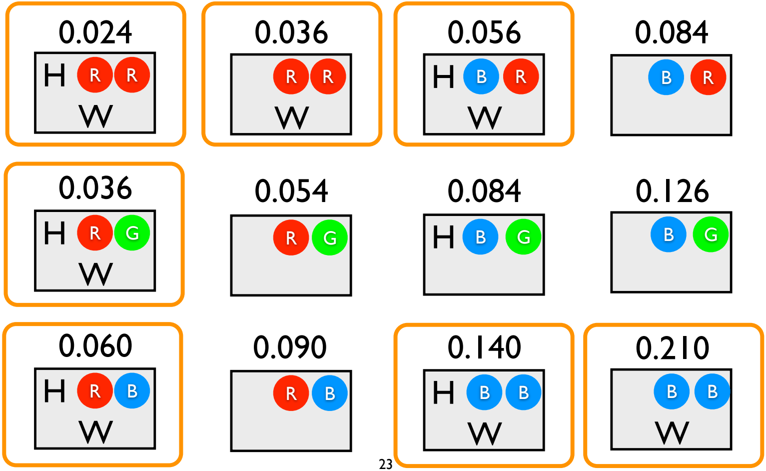


Probabilistic Logic Programming

De Raedt, Kersting, Natara... ...Relational AI

$$P(\underline{win}) = \sum = 0.562$$

Marginal Probability

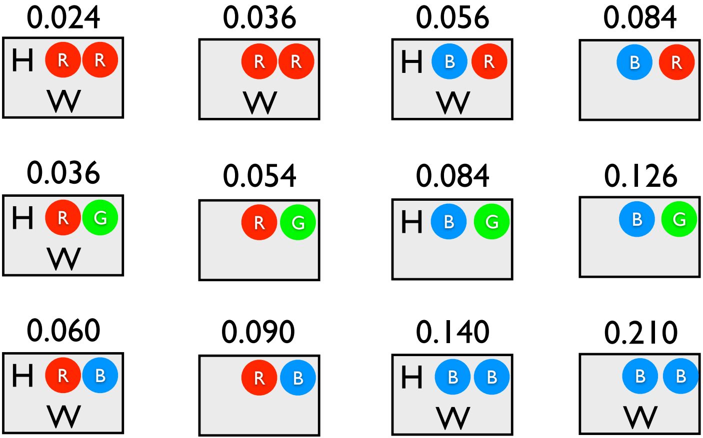


Probabilistic Logic Programming

De Raedt, Kersting, Natara ... ical Relational AI

P(win|col(2,green))= ?

Conditional Probability



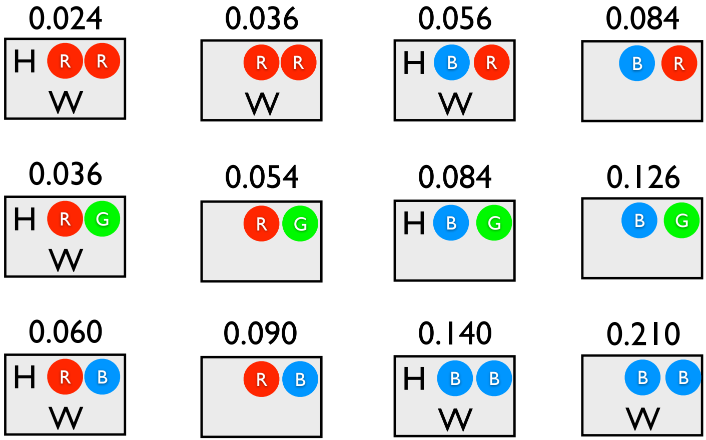
Probabilistic Logic Programming

De Raedt, Kersting, Natara… …tical Relational AI

Conditional Probability

$$P(\text{win}|\text{col}(2,\text{green})) = \Sigma / \Sigma$$

$$= P(\text{win} \wedge \text{col}(2,\text{green})) / P(\text{col}(2,\text{green}))$$



24

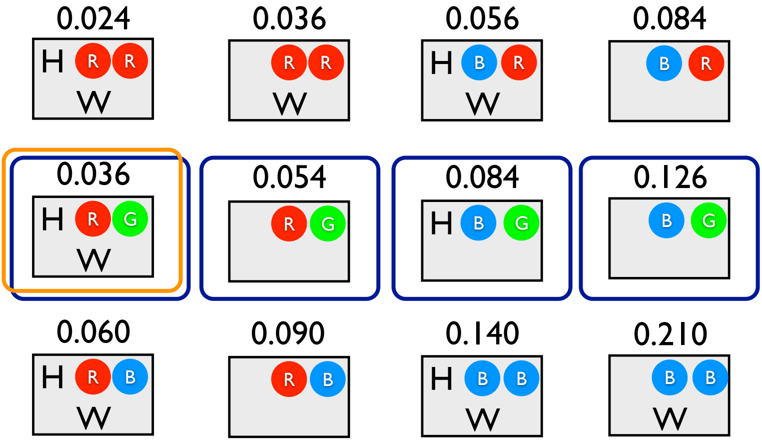
Probabilistic Logic Programming

De Raedt, Kersting, Natara ... ical Relational AI

Conditional Probability

$$P(win|\underline{col(2,green)}) = \Sigma / \Sigma$$

$$= P(\underline{win \wedge col(2,green)}) / P(\underline{col(2,green)})$$



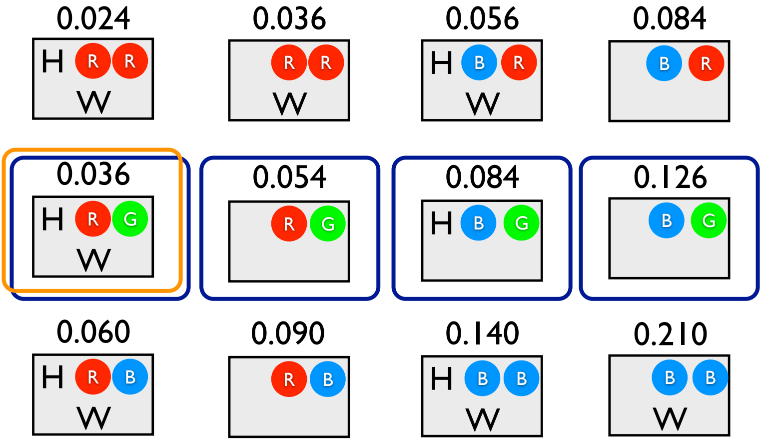
Probabilistic Logic Programming

De Raedt, Kersting, Natara ... tical Relational AI

P(win|col(2,green))= ∑/∑

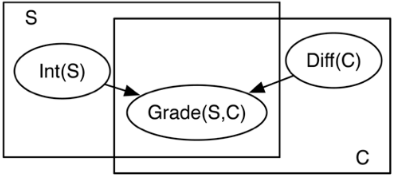
=0.036/0.3=0.12

Conditional Probability



Probabilistic Logic Programming

Flexible and Compact Relational Model for Predicting Grades



"Program" Abstraction:

- S, C logical variable representing students, courses
- the set of individuals of a type is called a population
- Int(S), Grade(S, C), D(C) are parametrized random variables

Grounding:

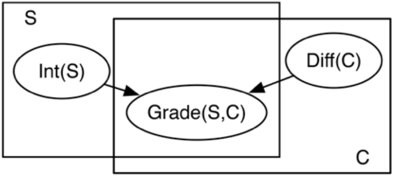
- for every student s, there is a random variable Int(s)
- for every course c, there is a random variable Di(c)
- for every s, c pair there is a random variable Grade(s,c)
- all instances share the same structure and parameters

De Raedt, Kersting, Natarajan, Poole: Statistical Relational AI

Probabilistic Logic Programming

ProbLog by example:

Grading




```
0.4 :: int(S) :- student(S).
0.5 :: diff(C):- course(C).

student(john). student(anna). student(bob).
course(ai).    course(ml).    course(cs).

gr(S,C,a) :- int(S), not diff(C).
0.3::gr(S,C,a); 0.5::gr(S,C,b);0.2::gr(S,C,c) :-
           int(S), diff(C).
0.1::gr(S,C,b); 0.2::gr(S,C,c); 0.2::gr(S,C,f) :-
           student(S), course(C),
           not int(S), not diff(C).
0.3::gr(S,C,c); 0.2::gr(S,C,f) :-
           not int(S), diff(C).
```

Probabilistic Logic Programming

ProbLog by example: Grading

```
unsatisfactory(S) :- student(S), grade(S,C,f).

excellent(S) :- student(S), not grade(S,C,G), below(G,a).
excellent(S) :- student(S), grade(S,C,a).

0.4 :: int(S) :- student(S).
0.5 :: diff(C):- course(C).

student(john). student(anna). student(bob).
course(ai).    course(ml).    course(cs).

gr(S,C,a) :- int(S), not diff(C).
0.3::gr(S,C,a); 0.5::gr(S,C,b);0.2::gr(S,C,c) :-
           int(S), diff(C).
0.1::gr(S,C,b); 0.2::gr(S,C,c); 0.2::gr(S,C,f) :-
           student(S), course(C),
           not int(S), not diff(C).
0.3::gr(S,C,c); 0.2::gr(S,C,f) :-
           not int(S), diff(C).
```