Clinical Decision Support Systems, 23/24

Inês Dutra and Pedro Rodrigues
DCC-FCUP & MEDCIDS-FMUP
ines@dcc.fc.up.pt, pprodrigues@med.up.pt

March 1st, 2024

Logic Programming

- Logic Programming: **knowledge** representation (KR) + programming (program = logic + control)
- Logic Programming: usually based on a subset of first-order logic $\rightarrow$ **Horn clauses**
- Horn clauses allow implications with **at most one positive** literal in the consequent
- This subset allows for a more efficient inference procedure: **linear resolution** (by Robinson)
- From KR point-of-view: easy way of representing **relations** and **relational** data
- **Probabilistic** logic programming: adding probabilities to logic programming

Knowledge Representation in Logic Programming

- Classical and popular example of KR in logic programming (using the Prolog syntax): family tree

```
mother(beryl,carol).        mother(carol,john).
father(arthur, carol).
parent(X,Y) :- mother(X,Y); father(X,Y).
grandparent(X,Z) :- parent(X,Y), parent(Y,Z).
```

- In Prolog:

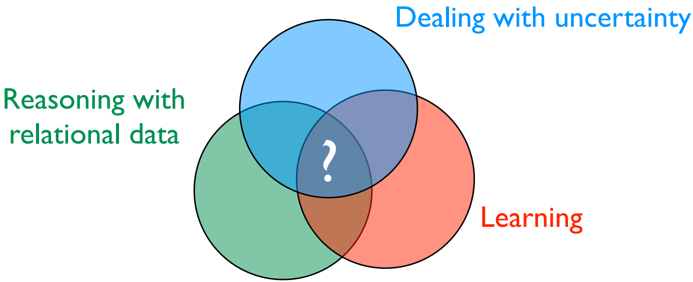
`:-`	implication ($\leftarrow$)
`,`	conjunction (and)
`;`	disjunction (or)

- first letter uppercase: logical variable
- first letter lowercase: constant (atom, predicate, literal, argument)

- Practice with the family tree
  - Go to the swish webpage
  - Click in the “Program” tab
  - copy the family tree program to the swish editor area
  - try queries according to list ex1

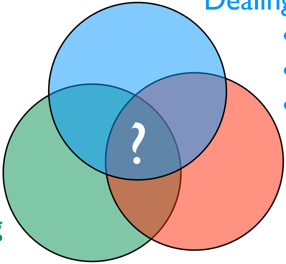
(Source: https://logic-data-science.github.io/Slides/DeRaedt.pdf - excellent presentation by de Raedt and Kimmig)

A key question in AI:



2

A key question in AI:



Reasoning with relational data

- logic
- databases
- programming
- ...

Dealing with uncertainty

- probability theory
- graphical models
- ...

Learning

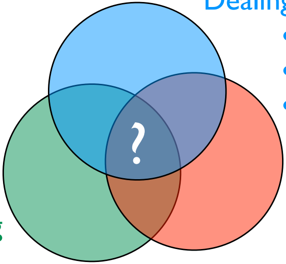
- parameters
- structure

2

A key question in AI:

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

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



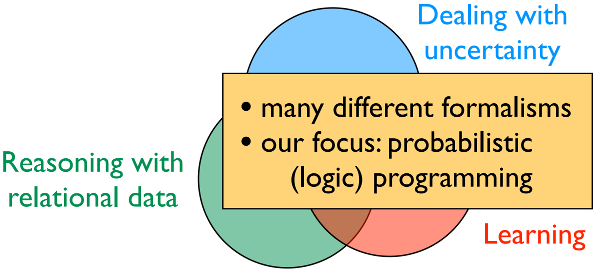
Learning
- parameters
- structure

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

2

Probabilistic Logic Programming

Common theme

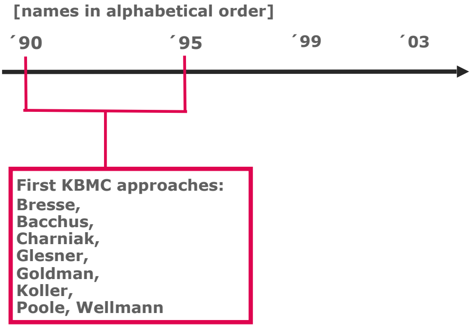


Statistical relational learning, probabilistic logic learning, probabilistic programming, ...

10

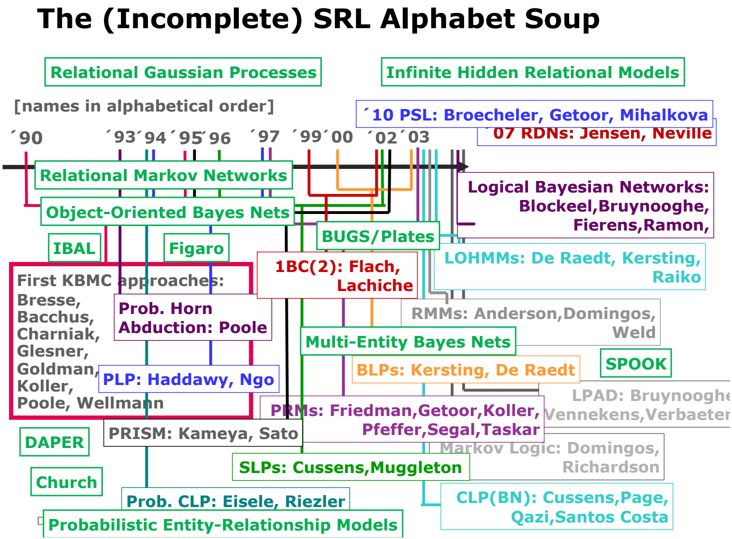
Probabilistic Logic Programming

The (Incomplete) SRL Alphabet Soup



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

Probabilistic Logic Programming



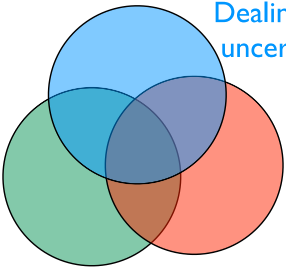
Probabilistic Logic Programs

- 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, …

ProbLog

probabilistic Prolog



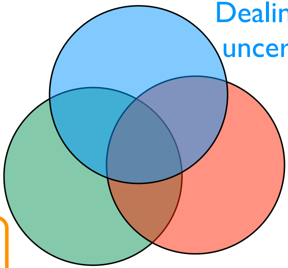

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

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

16

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

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).
```

16

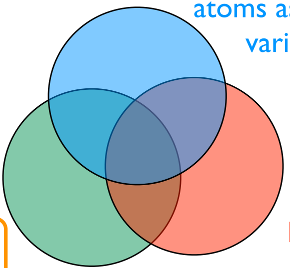
http://dtai.cs.kuleuven.be/problog/

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).
```

16

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

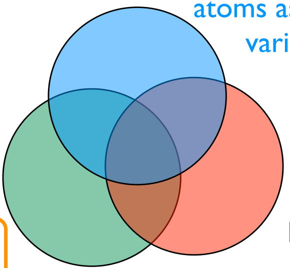
ProbLog

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).
```

16

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

Probabilistic Logic Programming

ProbLog

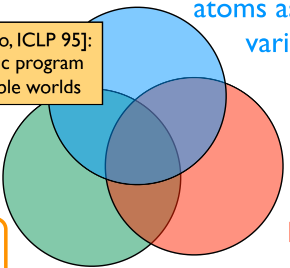
probabilistic Prolog

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

Learning

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

16

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

Probabilistic Logic Programming

ProbLog

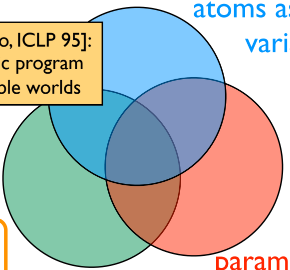
probabilistic Prolog

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).
```

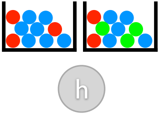
parameter learning, adapted relational learning techniques

16

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

ProbLog by example:

A bit of gambling

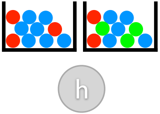


- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

17

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)

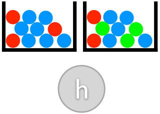
```
0.4 :: heads.
```

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

17

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)

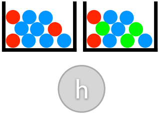
```
0.4 :: heads.

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

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

17

ProbLog by example:

A bit of gambling



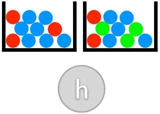
- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

```
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

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)

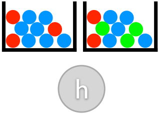
```
0.4 :: heads.

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

17

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).
```

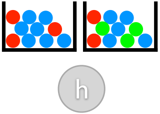
logical rule encoding background knowledge

17

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).
```

17

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).
```

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

18

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?

18

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 $\boxed{\texttt{col(2,green)}}$?

evidence

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

18

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

18

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).
```

19

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

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).
```

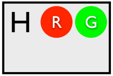
0.4×0.3

19

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).
```

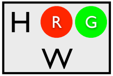
$0.4 \times 0.3 \times 0.3$



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).
```

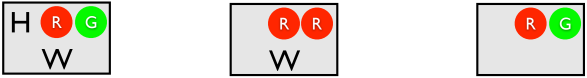
$0.4 \times 0.3 \times 0.3$



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).
```

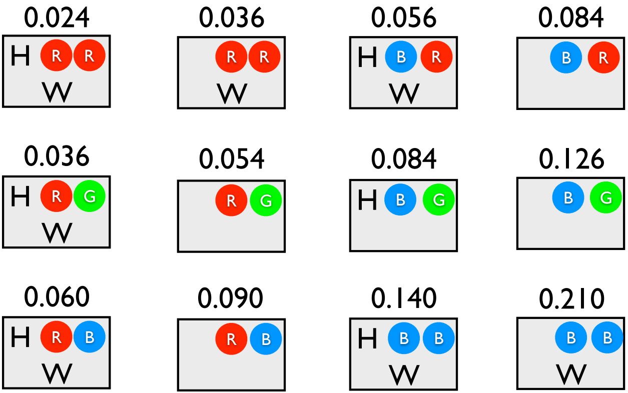
$0.4 \times 0.3 \times 0.3$ $\quad$ $(1-0.4) \times 0.3 \times 0.2$ $\quad$ $(1-0.4) \times 0.3 \times 0.3$



20

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

All Possible Worlds

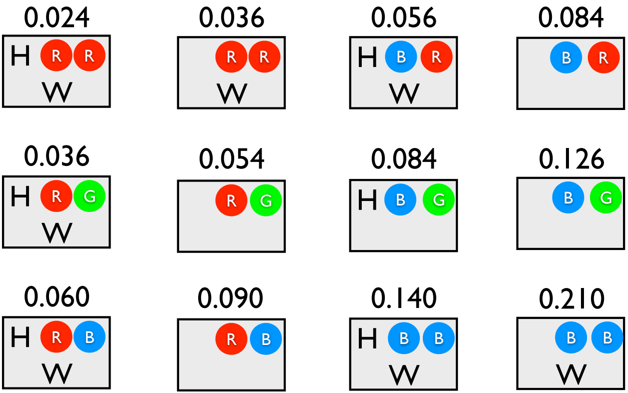


21

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

De Raedt, Kersting, Nat… …elational AI

MPE Inference

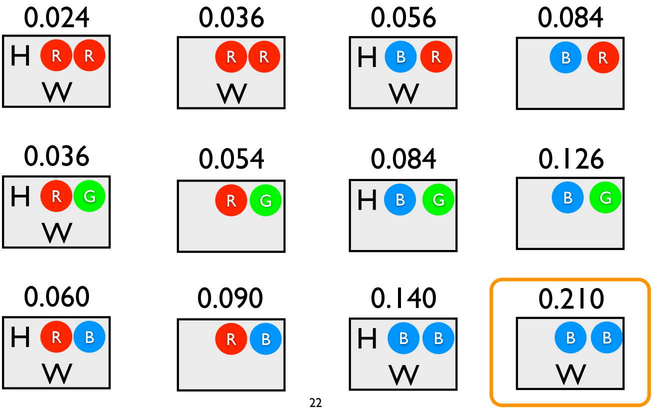


22

De Raedt, Kersting, Nat Relational AI

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

MPE Inference



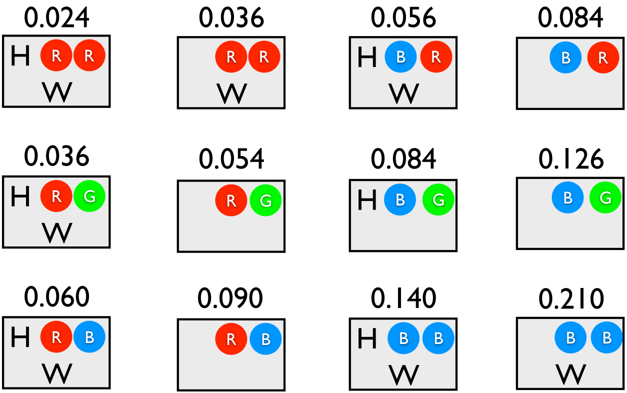
22

Probabilistic Logic Programming

De Raedt, Kersting, Natara… …ical Relational AI

$P(win) = ?$

Marginal Probability



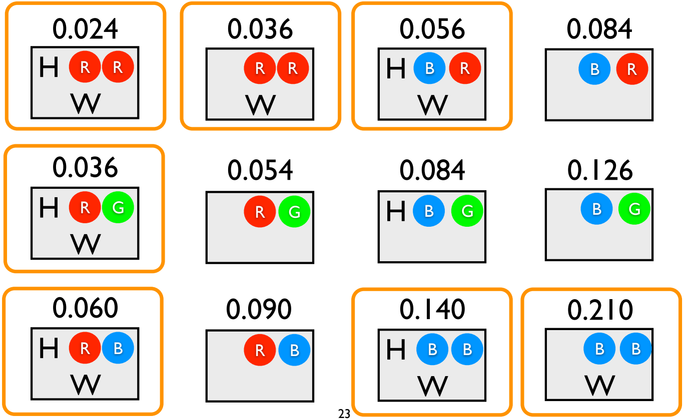
23

Probabilistic Logic Programming

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

Marginal Probability

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

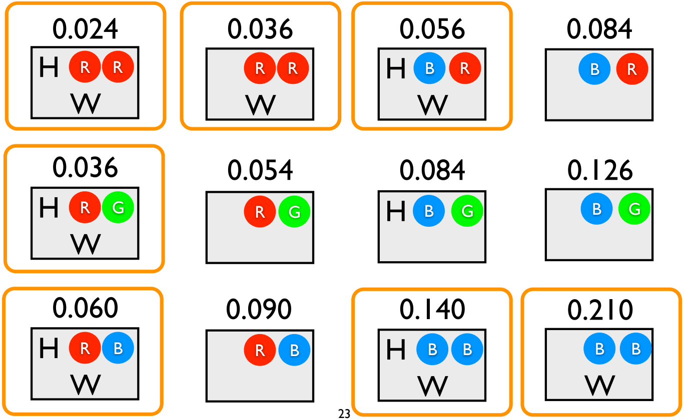


23

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

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

Marginal Probability



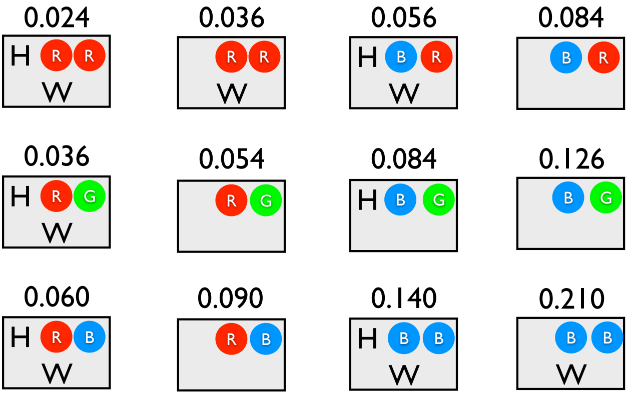
23

Probabilistic Logic Programming

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

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

Conditional Probability

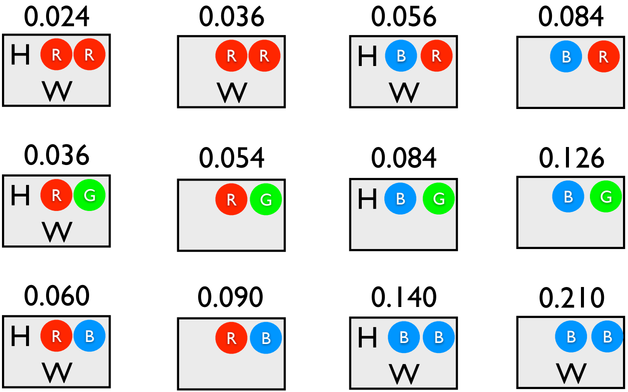


24

De Raedt, Kersting, Natara… …tical Relational AI

$$P(win|col(2,green)) = \sum / \sum$$
$$= P(win \wedge col(2,green)) / P(col(2,green))$$

Conditional Probability



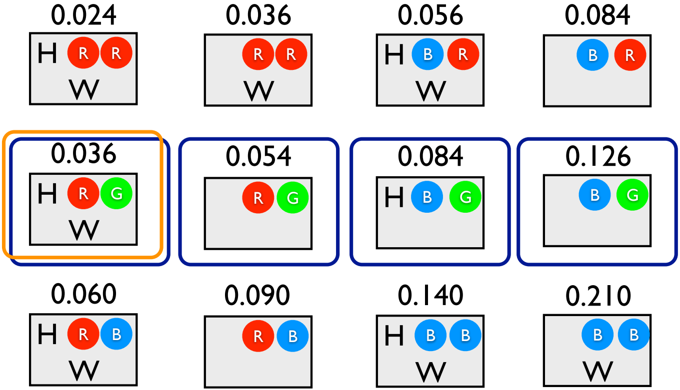
24

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

Conditional Probability

P(win|col(2,green))= ∑/∑

=P(win∧col(2,green))/P(col(2,green))



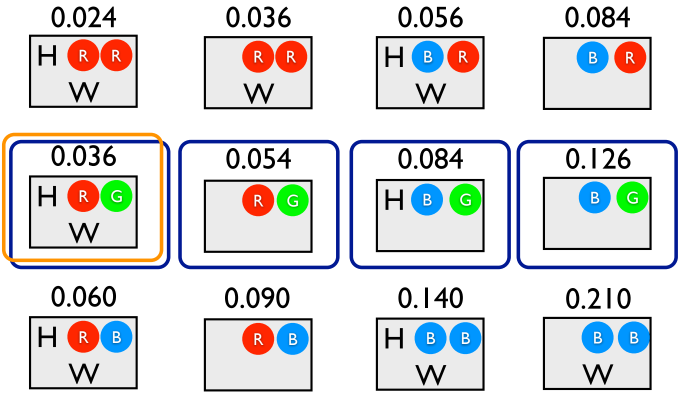
24

Probabilistic Logic Programming

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

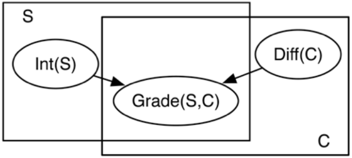
Conditional Probability

$$P(win|col(2,green)) = \Sigma / \Sigma$$
$$= 0.036/0.3 = 0.12$$



24

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:

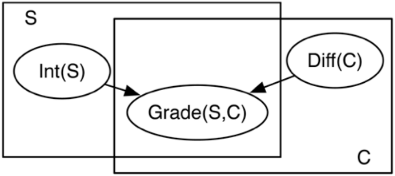
- for every student s, there is a random variable Int(s)
- for every course c, there is a random variable Di(c)
- for every s, c pair there is a random variable Grade(s,c)
- all instances share the same structure and parameters

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).
```

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).
```

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

Inference

The challenge : disjoint sum problem

```
0.4::heads(1).
0.7::heads(2).
0.5::heads(3).
win :- heads(1).
win :- heads(2), heads(3).
```

$win \leftrightarrow h(1) \vee (h(2) \wedge h(3))$

$P(win) = P(h(1) \vee (h(2) \wedge h(3))$

$=/= P(h(1)) + P(h(2) \wedge h(3))$

should be

$= P(h(1)) + P(h(2) \wedge h(3)) - P(h(1) \wedge h(2) \wedge h(3))$

35

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

Inference

Map to Weighted Model Counting Problem and Solver

```
0.4::heads(1).
0.7::heads(2).
0.5::heads(3).
win :- heads(1).
win :- heads(2), heads(3).
```

$$\text{win} \leftrightarrow h(1) \vee (h(2) \wedge h(3))$$

Ground out

+ Put formula in CNF format

+ weights

+ call WMC

$$(\neg\text{win} \vee h(1) \vee h(2)) \wedge (\neg\text{win} \vee h(1) \vee h(3)) \wedge (\text{win} \vee \neg h(1)) \wedge (\text{win} \vee \neg h(2) \vee \neg h(3))$$

$h(1) \rightarrow 0.4$ $h(2) \rightarrow 0.7$ $h(3) \rightarrow 0.5$
$\neg h(1) \rightarrow 0.6$ $\neg h(2) \rightarrow 0.3$ $\neg h(3) \rightarrow 0.5$

36

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

Weighted Model Counting

$$WMC(\phi) = \sum_{I_V \models \phi} \prod_{l \in I_V} w(l)$$

37

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

Weighted Model Counting

propositional formula in conjunctive normal form (CNF)

$$WMC(\phi) = \sum_{I_V \models \phi} \prod_{l \in I_V} w(l)$$

interpretations (truth value assignments) of propositional variables

weight of literal

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

Weighted Model Counting

propositional formula in conjunctive normal form (CNF)

given by SRL model & query

$$WMC(\phi) = \sum_{I_V \models \phi} \prod_{l \in I_V} w(l)$$

interpretations (truth value assignments) of propositional variables

possible worlds

weight of literal

for p::f,
w(f) = p
w(not f) = 1−p

37

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

Weighted Model Counting

- Simple WMC solvers based on a generalisation of DPLL algorithm for SAT (Davis Putnam Logeman Loveland algorithm)
- Current solvers often use knowledge compilation (is also state of the art for inference in graphical models) — here an OBDD, many variations s-dDNNF, SDDs, …

win $\leftrightarrow$ h(1) $\vee$ (h(2) $\wedge$ h(3))

A good source for Binary Decision Diagrams OBDD: ordered binary decision diagrams

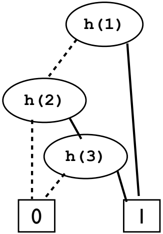
s-dDNNF: structured-deterministic Decomposable Negation Normal Form SDD: Sentential Decision Diagram

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

Weighted Model Counting

- Simple WMC solvers based on a generalisation of DPLL algorithm for SAT (Davis Putnam Logeman Loveland algorithm)
- Current solvers often use knowledge compilation (is also state of the art for inference in graphical models) — here an OBDD, many variations s-dDNNF, SDDs, ...

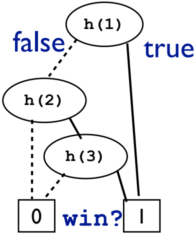
$\text{win} \leftrightarrow h(1) \lor (h(2) \land h(3))$



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

Weighted Model Counting

- Simple WMC solvers based on a generalisation of DPLL algorithm for SAT (Davis Putnam Logeman Loveland algorithm)
- Current solvers often use knowledge compilation (is also state of the art for inference in graphical models) — here an OBDD, many variations s-dDNNF, SDDs, …



$\text{win} \leftrightarrow h(1) \vee (h(2) \wedge h(3))$

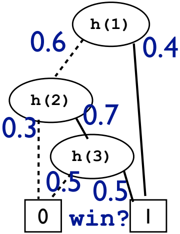
De Raedt, Kersting, Natarajan, Poole: Statistical Relational AI

Weighted Model Counting

- Simple WMC solvers based on a generalisation of DPLL algorithm for SAT (Davis Putnam Logeman Loveland algorithm)
- Current solvers often use knowledge compilation (is also state of the art for inference in graphical models) — here an OBDD, many variations s-dDNNF, SDDs, …

P(**win**) =
probability of
reaching 1-leaf

win ↔ h(1) ∨ (h(2) ∧ h(3))



More inference

- Many variations / extensions
- Approximate inference
- Lifted inference
  - infected(X) :- contact(X,Y), sick(Y).