Probabilistic Logic Programming

ALP Summer School on Computational Logic 2014

Angelika Kimmig
angelika.kimmig@cs.kuleuven.be
Probabilistic Logic Programming

Thanks to Luc De Raedt & many others working on PLP and especially ProbLog!
A key question in AI:

Dealing with uncertainty

Reasoning with relational data

Learning
A key question in AI:

Dealing with uncertainty

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

Learning
A key question in AI:

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

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

Learning
A key question in AI:

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

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

- Learning
  - parameters
  - structure
A key question in AI:

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

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

Learning
- parameters
- structure

Statistical relational learning, probabilistic logic learning, probabilistic programming, ...
### Example:
Information Extraction

<table>
<thead>
<tr>
<th>instance</th>
<th>iteration</th>
<th>date learned</th>
<th>confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>kelly_andrews is a female</td>
<td>826</td>
<td>29-mar-2014</td>
<td>98.7</td>
</tr>
<tr>
<td>investment_next_year is an economic sector</td>
<td>829</td>
<td>10-apr-2014</td>
<td>95.3</td>
</tr>
<tr>
<td>shibenik is a geopolitical entity that is an organization</td>
<td>829</td>
<td>10-apr-2014</td>
<td>97.2</td>
</tr>
<tr>
<td>quality_web_design_work is a character trait</td>
<td>826</td>
<td>29-mar-2014</td>
<td>91.0</td>
</tr>
<tr>
<td>mercedes_benz_cl_5_by_carlsson is an automobile manufacturer</td>
<td>829</td>
<td>10-apr-2014</td>
<td>95.2</td>
</tr>
<tr>
<td>social_work is an academic program at the university rutgers_university</td>
<td>827</td>
<td>02-apr-2014</td>
<td>93.8</td>
</tr>
<tr>
<td>dante wrote the book the_divine_comedy</td>
<td>826</td>
<td>29-mar-2014</td>
<td>93.8</td>
</tr>
<tr>
<td>willie_aames was born in the city los_angeles</td>
<td>831</td>
<td>16-apr-2014</td>
<td>100.0</td>
</tr>
<tr>
<td>kitt_peak is a mountain in the state or province arizona</td>
<td>831</td>
<td>16-apr-2014</td>
<td>96.9</td>
</tr>
<tr>
<td>greenwich is a park in the city london</td>
<td>831</td>
<td>16-apr-2014</td>
<td>100.0</td>
</tr>
</tbody>
</table>

NELL: [http://rtw.ml.cmu.edu/rtw/](http://rtw.ml.cmu.edu/rtw/)
Example: Information Extraction

NELL: http://rtw.ml.cmu.edu/rtw/

instances for many different relations
Example: Information Extraction

instances for many different relations

degree of certainty

NELL: http://rtw.ml.cmu.edu/rtw/
Networks of Uncertain Information

- pathway
  - participates in
  - participates in
  - is homologous to
- gene
  - belongs to
  - is homologous to
  - is found in
- locus
  - is located in
- phenotype
  - is related to
  - refers to
- homologgroup
  - codes for
- biological process
  - participates in
  - participates in
  - is found in
- cellular component
  - is found in
  - participates in
  - has
  - subsumes, interacts with
- molecular function
  - participates in
  - participates in
- protein
  - refers to

Biomine database @ Helsinki
http://biomine.cs.helsinki.fi/
Biomine Network
Notch receptor processing
Biological Process
GO:GO:0007220

presenilin 2
Gene
EntrezGene:81751
Biomine Network

- `participates_in` 0.220

**Gene**

**BiologicalProcess**

**presenilin 2**
Gene EntrezGene:81751
Biomine Network

- **Gene**
  - participates_in
  - BiologicalProcess
  - participates_in
  - is_homologous_to
  - participates_in

- **Notch receptor processing**
  - BiologicalProcess
  - GO:GO:0007220

- **presenilin 2**
  - Gene
  - EntrezGene:81751

- **different types of nodes & links**
- **automatically extracted from text, databases, ...**
- **probabilities quantifying source reliability, extractor confidence, ...**
- **similar in other contexts, e.g., linked open data, knowledge graphs, ...**
What is the most relevant subnetwork (with at most k edges) connecting these?
What is the most relevant subnetwork (with at most k edges) connecting these?
Should there be a link? If so, of what type? Why?
Node Classification

Can we predict the type of a node given information on its neighbors?

e.g., the type of a webpage given its links and the words on the page?
Entity Resolution

Automatically extracted co-author network: which nodes refer to the same person?
Viral Marketing

Which advertising strategy maximizes expected profit?

Van den Broeck et al., AAAI 10
Voter Opinion Modeling

Can we predict preferences?

[Bach et al, NIPS 12]
Can we predict strength of ties?

[Huang et al, SBP 13]
Dynamic networks

*Travian*: A massively multiplayer real-time strategy game

Can we build a model of this world?
Can we use it for playing better?
Dynamic networks

Travian: A massively multiplayer real-time strategy game

Can we build a model of this world?
Can we use it for playing better?
Molecular interaction networks

Can we find the mechanism connecting causes to effects?
Diagnosing machine failures

Can we build a model of the robot’s working and use it to find causes of failures?

[Schramm, Meert and Driessens]
Robotics

- How to achieve a specific configuration of objects on the shelf?
- Where’s the orange mug?
- Where’s something to serve soup in?

[Moldovan et al, ICRA 12, 14]
• Track people or objects over time? Even if temporarily hidden?
• Recognize activities?
• Infer object properties?

[Skarlatidis et al, TPLP 14; Nitti et al, IROS 13, ICRA 14]
Common theme

Dealing with uncertainty

Reasoning with relational data

Learning

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

Dealing with uncertainty

Reasoning with relational data

Learning

• many different formalisms
• our focus: probabilistic logic programming

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

Dealing with uncertainty
Reasoning with relational data
Learning

http://dtai.cs.kuleuven.be/problog/
Prolog / logic programming

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

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

Dealing with uncertainty

Learning

Problog
probabilistic Prolog

http://dtai.cs.kuleuven.be/problog/
ProbLog
probabilistic Prolog

Dealing with uncertainty

Prolog / logic programming

stresses(ann).
influences(ann,bob).
influences(bob,carl).

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

one world

Learning

http://dtai.cs.kuleuven.be/problog/
Prolog / logic programming

atoms as random variables

Prolog / logic programming

one world

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

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

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

ProbLog probabilistic Prolog

Learning

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

probabilistic Prolog

**Prolog / logic programming**

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

**several possible worlds**

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

**atoms as random variables**

**Learning**

**one world**

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

http://dtai.cs.kuleuven.be/problog/
ProbLog
probabilistic Prolog

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

Prolog / logic programming

several possible worlds

atoms as random variables

Learning

one world

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

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

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

smokes(X) :- stress(X).
smokes(X) :-
  influences(Y,X), smokes(Y).
ProbLog
probabilistic Prolog

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

Prolog / logic programming

several possible worlds

atoms as random variables

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

Parameter learning, adapted relational learning techniques

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

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

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

one world
Probabilistic Prologs: Two Views

• Distribution semantics:
  • probability distribution over interpretations
  • degree of belief

• Stochastic Logic Programs (SLPs):
  • probability distribution over query answers
  • like in probabilistic grammars
Probabilistic Logic Programming

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

e.g., PRISM, ICL, ProbLog, LPADs, CP-logic, ...

- multi-valued switches
- probabilistic alternatives
- probabilistic facts
- annotated disjunctions
- causal-probabilistic laws
Extensions of basic PLP

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

• Modeling
• Reasoning
• Language extensions
• Advanced topics

... with some detours on the way
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)
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)

\[
0.4 :: \text{heads}.
\]

**probabilistic fact:** heads is true with probability 0.4 (and false with 0.6)
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) <- true.
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) <- true.
0.2 :: col(2,red); 0.3 :: col(2,green);
    0.5 :: col(2,blue) <- true.

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

win :- heads, col(_,red).  **logical rule** encoding background knowledge
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) <- true.
0.2 :: col(2,red); 0.3 :: col(2,green);
0.5 :: col(2,blue) <- true.

win :- heads, col(_,red).
**logical rule** encoding

win :- col(1,C), col(2,C).
**background knowledge**
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) <- 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).
```

consequences
Questions

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).

• Probability of win?

• Probability of win given col(2,green)?

• Most probable world where win is true?
Questions

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.

\[\text{win} \leftarrow \text{heads, col(\_, red)}\].
\[\text{win} \leftarrow \text{col(1,}\ C\text{), col(2,}\ C\text{)}.\]

marginal probability

- Probability of \textbf{win} query
- Probability of \textbf{win} given \texttt{col(2,green)}?
- Most probable world where \textbf{win} is true?
Questions

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).

marginal probability

• Probability of win?

conditional probability

• Probability of win given \texttt{col(2,green)}?

evidence

• Most probable world where \texttt{win} is true?
Questions

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).

marginal probability

• Probability of win?

conditional probability

• Probability of win given col(2,green)?

• Most probable world where win is true?

MPE inference
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).
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 :: 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).
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

\begin{array}{|c|c|}
\hline
H & R \\
\hline
\end{array}
Possible Worlds

0.4 :: heads.

0.3 :: \texttt{col(1,red)}; 0.7 :: \texttt{col(1,blue)} \leftarrow \text{true}.

0.2 :: \texttt{col(2,red)}; 0.3 :: \texttt{col(2,green)}; 0.5 :: \texttt{col(2,blue)} \leftarrow \text{true}.

\texttt{win} :: \texttt{heads, col(_,red)}.
\texttt{win} :: \texttt{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.

\[
\begin{align*}
\text{win} & : - \text{heads}, \text{col}(_,\text{red}). \\
\text{win} & : - \text{col}(1,C), \text{col}(2,C).
\end{align*}
\]

\[0.4 \times 0.3 \times 0.3\]
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).

\[0.4 \times 0.3 \times 0.3 \quad (1-0.4)\]
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).

\[
\begin{align*}
0.4 \times 0.3 \times 0.3 & \quad (1-0.4) \times 0.3 \\
\begin{array}{c} H \ \ R \ \ G \\
W \end{array} & \begin{array}{c} R \end{array}
\end{align*}
\]
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).

\[
0.4 \times 0.3 \times 0.3 \quad (1-0.4) \times 0.3 \times 0.2
\]
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.

\[
\begin{align*}
0.4 \times 0.3 \times 0.3 & \quad (1-0.4)\times0.3 \times0.2
\end{align*}
\]

\[
\begin{array}{c}
H \quad R \quad G \\
W \\
\end{array}
\quad \quad \quad
\begin{array}{c}
R \quad R \\
W \\
\end{array}
\]
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).

\[
0.4 \times 0.3 \times 0.3 \\
H \ R \ G \\
W
\]

\[
(l-0.4) \times 0.3 \times 0.2 \\
R \ R \\
W
\]

\[
(l-0.4)
\]
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).

\[
\begin{align*}
0.4 \times 0.3 \times 0.3 \\
(1-0.4) \times 0.3 \times 0.2 \\
(1-0.4) \times 0.3
\end{align*}
\]
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).

\[
\begin{align*}
0.4 \times 0.3 \times 0.3 & \quad (1-0.4) \times 0.3 \times 0.2 & \quad (1-0.4) \times 0.3 \times 0.3
\end{align*}
\]

\[
\begin{array}{c}
H \quad R \quad G \\
W
\end{array} 
\begin{array}{c}
R \quad R \\
W
\end{array} 
\begin{array}{c}
R \quad G \\
W
\end{array}
\]
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).

\[
0.4 \times 0.3 \times 0.3 \quad (1-0.4) \times 0.3 \times 0.2 \quad (1-0.4) \times 0.3 \times 0.3
\]
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).
All Possible Worlds

0.024

0.036

0.056

0.084

0.036

0.054

0.084

0.126

0.060

0.090

0.140

0.210
Most likely world where `win` is true?

<table>
<thead>
<tr>
<th>0.024</th>
<th>0.036</th>
<th>0.056</th>
<th>0.084</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="HHHRRR" /></td>
<td><img src="image" alt="RRR" /></td>
<td><img src="image" alt="HBRR" /></td>
<td><img src="image" alt="BBR" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>0.036</th>
<th>0.054</th>
<th>0.084</th>
<th>0.126</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="HHHRGG" /></td>
<td><img src="image" alt="RRG" /></td>
<td><img src="image" alt="HBGG" /></td>
<td><img src="image" alt="BBG" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>0.060</th>
<th>0.090</th>
<th>0.140</th>
<th>0.210</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="HHHRBB" /></td>
<td><img src="image" alt="RRB" /></td>
<td><img src="image" alt="HBBB" /></td>
<td><img src="image" alt="BBB" /></td>
</tr>
</tbody>
</table>
Most likely world where win is true?

<table>
<thead>
<tr>
<th></th>
<th>0.024</th>
<th>0.036</th>
<th>0.056</th>
<th>0.084</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>R</td>
<td>R</td>
<td>H</td>
<td>B</td>
</tr>
<tr>
<td>W</td>
<td>R</td>
<td>R</td>
<td>W</td>
<td>R</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0.036</th>
<th>0.054</th>
<th>0.084</th>
<th>0.126</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>R</td>
<td>G</td>
<td>H</td>
<td>B</td>
</tr>
<tr>
<td>W</td>
<td>R</td>
<td>G</td>
<td>W</td>
<td>G</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0.060</th>
<th>0.090</th>
<th>0.140</th>
<th>0.210</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>R</td>
<td>B</td>
<td>H</td>
<td>B</td>
</tr>
<tr>
<td>W</td>
<td>R</td>
<td>B</td>
<td>W</td>
<td>B</td>
</tr>
</tbody>
</table>

MPE Inference

28
Most likely world where $\text{col}(2, \text{blue})$ is false?
Most likely world where $\text{col}(2, \text{blue})$ is false?

- 0.024
  - $H \text{ } R \text{ } R \text{ } W$

- 0.036
  - $R \text{ } R \text{ } W$
  - $H \text{ } B \text{ } R \text{ } W$

- 0.036
  - $H \text{ } R \text{ } G \text{ } W$
  - $R \text{ } G \text{ } W$
  - $H \text{ } B \text{ } G \text{ } W$

- 0.054
  - $H \text{ } B \text{ } G \text{ } W$

- 0.056
  - $H \text{ } B \text{ } R \text{ } W$

- 0.084
  - $H \text{ } B \text{ } B \text{ } W$

- 0.084
  - $B \text{ } R \text{ } W$

- 0.090
  - $R \text{ } B \text{ } W$

- 0.126
  - $B \text{ } G \text{ } W$

- 0.140
  - $B \text{ } B \text{ } W$

- 0.210
  - $B \text{ } B \text{ } W$
\[ P(\text{win}) = ? \]

- \[ H \quad R \quad R \quad W \quad 0.024 \]
- \[ H \quad R \quad G \quad W \quad 0.036 \]
- \[ H \quad B \quad R \quad W \quad 0.056 \]
- \[ H \quad B \quad G \quad W \quad 0.084 \]
- \[ H \quad R \quad B \quad W \quad 0.060 \]
- \[ H \quad R \quad B \quad W \quad 0.090 \]
- \[ H \quad B \quad B \quad W \quad 0.140 \]
- \[ H \quad B \quad B \quad W \quad 0.210 \]

Marginal Probability
$P(\text{win}) = \sum$
\[ P(\text{win}) = \sum = 0.562 \]
$P(\text{win}|\text{col}(2,\text{green})) = \ ?$

<table>
<thead>
<tr>
<th>Probability</th>
<th>Colors</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.024</td>
<td>HRR</td>
</tr>
<tr>
<td>0.036</td>
<td>RRR</td>
</tr>
<tr>
<td>0.056</td>
<td>HRB</td>
</tr>
<tr>
<td>0.084</td>
<td>BR</td>
</tr>
<tr>
<td>0.036</td>
<td>HRG</td>
</tr>
<tr>
<td>0.054</td>
<td>RGR</td>
</tr>
<tr>
<td>0.084</td>
<td>HBG</td>
</tr>
<tr>
<td>0.126</td>
<td>BGR</td>
</tr>
<tr>
<td>0.060</td>
<td>HRB</td>
</tr>
<tr>
<td>0.090</td>
<td>RRB</td>
</tr>
<tr>
<td>0.140</td>
<td>HBB</td>
</tr>
<tr>
<td>0.210</td>
<td>BBB</td>
</tr>
</tbody>
</table>
$$P(\text{win}|\text{col}(2,\text{green})) = \Sigma / \Sigma$$

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

Conditional Probability

<table>
<thead>
<tr>
<th>0.024</th>
<th>0.036</th>
<th>0.056</th>
<th>0.084</th>
</tr>
</thead>
<tbody>
<tr>
<td>H R R</td>
<td>R R W</td>
<td>H B R</td>
<td>B R</td>
</tr>
<tr>
<td>W</td>
<td>W</td>
<td>W</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>0.036</th>
<th>0.054</th>
<th>0.084</th>
<th>0.126</th>
</tr>
</thead>
<tbody>
<tr>
<td>H R G</td>
<td>R G W</td>
<td>H B G</td>
<td>B G</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>0.060</th>
<th>0.090</th>
<th>0.140</th>
<th>0.210</th>
</tr>
</thead>
<tbody>
<tr>
<td>H R B</td>
<td>R B W</td>
<td>H B B</td>
<td>B B</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
\[ P(\text{win}|\text{col}(2,\text{green})) = \frac{\Sigma}{\Sigma} \]

\[ = P(\text{win} \land \text{col}(2,\text{green}))/P(\text{col}(2,\text{green})) \]

Conditional Probability
\[ P(\text{win}|\text{col}(2,\text{green})) = \frac{\sum}{\sum} = 0.036/0.3 = 0.12 \]
Distribution Semantics
(with probabilistic facts)

\[ P(Q) = \sum_{F \cup R \models Q} \prod_{f \in F} p(f) \prod_{f \notin F} 1 - p(f) \]

[Sato, ICLP 95]
Distribution Semantics
(with probabilistic facts)

\[ P(Q) = \sum_{F \cup R \models Q} \prod_{f \in F} p(f) \prod_{f \notin F} (1 - p(f)) \]

[Sato, ICLP 95]
Distribution Semantics
(with probabilistic facts)

[Sato, ICLP 95]

\[ P(Q) = \sum_{F \cup R = Q} \prod_{f \in F} p(f) \prod_{f \not\in F} (1 - p(f)) \]
Distribution Semantics
(with probabilistic facts)

\[
P(Q) = \sum_{F \cup R \vdash Q} \prod_{f \in F} p(f) \prod_{f \notin F} (1 - p(f))
\]

[Sato, ICLP 95]
Distribution Semantics
(with probabilistic facts)

\[ P(Q) = \sum_{F \cup R \models Q} \prod_{f \in F} p(f) \prod_{f \not\in F} (1 - p(f)) \]

query

sum over possible worlds where \( Q \) is true

subset of probabilistic facts

Prolog rules
Distribution Semantics
(with probabilistic facts)

$$P(Q) = \sum_{F \cup R \models Q} \prod_{f \in F} p(f) \prod_{f \notin F} (1 - p(f))$$

query

subset of probabilistic facts

sum over possible worlds where Q is true

Prolog rules

probability of possible world

[Sato, ICLP 95]
 Alternative view: CP-Logic

\[
\begin{align*}
\text{throws}(\text{john}). & \quad 0.5 :: \text{throws}(\text{mary}). \\
0.8 :: \text{break} & \leftarrow \text{throws}(\text{mary}). \\
0.6 :: \text{break} & \leftarrow \text{throws}(\text{john}). 
\end{align*}
\]

\[P(\text{break}) = 0.6 \times 0.5 \times 0.8 + 0.6 \times 0.5 \times 0.2 + 0.6 \times 0.5 + 0.4 \times 0.5 \times 0.8\]
Answering Questions

Given:
- program
- queries
- evidence

Find:
- marginal probabilities
- conditional probabilities
- MPE state
Answering Questions

Given:
- program
- queries
- evidence

Find:
- marginal probabilities
- conditional probabilities
- MPE state

possible worlds
infeasible
Answering Questions

**Given:**
- program
- queries
- evidence

**Find:**
- logical reasoning
- data structure
- probabilistic inference
- marginal probabilities
- conditional probabilities
- MPE state
Answering Questions

**Given:**
- program
- queries
- evidence

**Find:**
- marginal probabilities
- conditional probabilities
- MPE state

more on this later
Probabilistic Databases

Dealing with uncertainty

Reasoning with relational data

Learning

[Suciu et al 2011]
Dealing with uncertainty

Probabilistic Databases

select x.person, y.country
from bornIn x, cityIn y
where x.city=y.city

bornIn
<table>
<thead>
<tr>
<th>person</th>
<th>city</th>
</tr>
</thead>
<tbody>
<tr>
<td>ann</td>
<td>london</td>
</tr>
<tr>
<td>bob</td>
<td>york</td>
</tr>
<tr>
<td>eve</td>
<td>new york</td>
</tr>
<tr>
<td>tom</td>
<td>paris</td>
</tr>
</tbody>
</table>

cityIn
<table>
<thead>
<tr>
<th>city</th>
<th>country</th>
</tr>
</thead>
<tbody>
<tr>
<td>london</td>
<td>uk</td>
</tr>
<tr>
<td>york</td>
<td>uk</td>
</tr>
<tr>
<td>paris</td>
<td>usa</td>
</tr>
</tbody>
</table>

Dealing with uncertainty

Relational database

Learning

[Suciu et al 2011]
Dealing with uncertainty

Probabilistic Databases

select x.person, y.country
from bornIn x, cityIn y
where x.city=y.city

one world

BornIn

<table>
<thead>
<tr>
<th>person</th>
<th>city</th>
</tr>
</thead>
<tbody>
<tr>
<td>ann</td>
<td>london</td>
</tr>
<tr>
<td>bob</td>
<td>york</td>
</tr>
<tr>
<td>eve</td>
<td>new york</td>
</tr>
<tr>
<td>tom</td>
<td>paris</td>
</tr>
</tbody>
</table>

cityIn

<table>
<thead>
<tr>
<th>city</th>
<th>country</th>
</tr>
</thead>
<tbody>
<tr>
<td>london</td>
<td>uk</td>
</tr>
<tr>
<td>york</td>
<td>uk</td>
</tr>
<tr>
<td>paris</td>
<td>usa</td>
</tr>
</tbody>
</table>

Learning
Probabilistic Databases

select x.person, y.country
from bornIn x, cityIn y
where x.city=y.city

one world

bornIn
city | person |
--- | --- |
london | ann |
york | bob |
new york | eve |
paris | tom |

cityIn
city | country |
--- | --- |
london | uk |
york | uk |
paris | usa |

[Suciu et al 2011]
Probabilistic Databases

**several possible worlds**

<table>
<thead>
<tr>
<th>person</th>
<th>city</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>ann</td>
<td>london</td>
<td>0.87</td>
</tr>
<tr>
<td>bob</td>
<td>york</td>
<td>0.95</td>
</tr>
<tr>
<td>eve</td>
<td>new york</td>
<td>0.90</td>
</tr>
<tr>
<td>tom</td>
<td>paris</td>
<td>0.56</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>city</th>
<th>country</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>london</td>
<td>uk</td>
<td>0.99</td>
</tr>
<tr>
<td>york</td>
<td>uk</td>
<td>0.75</td>
</tr>
<tr>
<td>paris</td>
<td>usa</td>
<td>0.40</td>
</tr>
</tbody>
</table>

select x.person, y.country from bornIn x, cityIn y where x.city=y.city

**one world**

<table>
<thead>
<tr>
<th>person</th>
<th>city</th>
</tr>
</thead>
<tbody>
<tr>
<td>ann</td>
<td>london</td>
</tr>
<tr>
<td>bob</td>
<td>york</td>
</tr>
<tr>
<td>eve</td>
<td>new york</td>
</tr>
<tr>
<td>tom</td>
<td>paris</td>
</tr>
</tbody>
</table>

**cityIn**

<table>
<thead>
<tr>
<th>city</th>
<th>country</th>
</tr>
</thead>
<tbody>
<tr>
<td>london</td>
<td>uk</td>
</tr>
<tr>
<td>york</td>
<td>uk</td>
</tr>
<tr>
<td>paris</td>
<td>usa</td>
</tr>
</tbody>
</table>

**tuples as random variables**

Learning
# Probabilistic Databases

Probabilistic databases represent data as probability distributions over possible worlds. This allows for the handling of uncertainty in the data, capturing the uncertainty in the relationships and attributes of entities.

## Relational Database

<table>
<thead>
<tr>
<th>person</th>
<th>city</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>ann</td>
<td>london</td>
<td>0.87</td>
</tr>
<tr>
<td>bob</td>
<td>york</td>
<td>0.95</td>
</tr>
<tr>
<td>eve</td>
<td>new york</td>
<td>0.90</td>
</tr>
<tr>
<td>tom</td>
<td>paris</td>
<td>0.56</td>
</tr>
</tbody>
</table>

## City Database

<table>
<thead>
<tr>
<th>city</th>
<th>country</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>london</td>
<td>uk</td>
<td>0.99</td>
</tr>
<tr>
<td>york</td>
<td>uk</td>
<td>0.75</td>
</tr>
<tr>
<td>paris</td>
<td>usa</td>
<td>0.40</td>
</tr>
</tbody>
</table>

### Probabilistic Tables + Database Queries

```sql
SELECT person, P(city) FROM bornIn x, cityIn y
WHERE x.city = y.city
```

This query selects persons and their probabilities of being in certain cities, given the probability distributions from the tables.

### Several Possible Worlds

- One world: With the probabilistic tables and database queries, we can compute the distribution over possible worlds.

- Tuples as random variables:

  - A probabilistic database is a database with a probability distribution on the possible worlds.
  - Each world represents a possible state of the database.

- Learning:

  - Probabilistic databases can be learned from data, allowing for the inference of uncertainty in the data.

[Suciu et al. 2011]
Example: Information Extraction

NELL: http://rtw.ml.cmu.edu/rtw/

<table>
<thead>
<tr>
<th>instance</th>
<th>iteration</th>
<th>date learned</th>
<th>confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>kelly_andrews is a female</td>
<td>826</td>
<td>29-mar-2014</td>
<td>98.7</td>
</tr>
<tr>
<td>investment_next_year is an economic sector</td>
<td>829</td>
<td>10-apr-2014</td>
<td>95.3</td>
</tr>
<tr>
<td>shibenik is a geopolitical entity that is an organization</td>
<td>829</td>
<td>10-apr-2014</td>
<td>97.2</td>
</tr>
<tr>
<td>quality_web_design_work is a character trait</td>
<td>826</td>
<td>29-mar-2014</td>
<td>91.0</td>
</tr>
<tr>
<td>mercedes_benz.cls_by_carlsson is an automobile manufacturer</td>
<td>829</td>
<td>10-apr-2014</td>
<td>95.2</td>
</tr>
<tr>
<td>social_work is an academic program at the university rutgers_university</td>
<td>827</td>
<td>02-apr-2014</td>
<td>93.8</td>
</tr>
<tr>
<td>dante wrote the book the divine comedy</td>
<td>826</td>
<td>29-mar-2014</td>
<td>93.8</td>
</tr>
<tr>
<td>willie_ames was born in the city los_angeles</td>
<td>831</td>
<td>16-apr-2014</td>
<td>100.0</td>
</tr>
<tr>
<td>kitt_peak is a mountain in the state or province arizona</td>
<td>831</td>
<td>16-apr-2014</td>
<td>96.9</td>
</tr>
<tr>
<td>greenwich is a park in the city london</td>
<td>831</td>
<td>16-apr-2014</td>
<td>100.0</td>
</tr>
</tbody>
</table>

instances for many different relations
degree of certainty
Querying: relational database

<table>
<thead>
<tr>
<th>Company</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>sony</td>
<td>walkman</td>
</tr>
<tr>
<td>microsoft</td>
<td>mac_os_x</td>
</tr>
<tr>
<td>ibm</td>
<td>personal_computer</td>
</tr>
<tr>
<td>microsoft</td>
<td>mac_os</td>
</tr>
<tr>
<td>adobe</td>
<td>adobe_indesign</td>
</tr>
<tr>
<td>adobe</td>
<td>adobe_dreamweaver</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Company</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>microsoft</td>
<td>redmond</td>
</tr>
<tr>
<td>ibm</td>
<td>san_jose</td>
</tr>
<tr>
<td>emirates_airlines</td>
<td>dubai</td>
</tr>
<tr>
<td>honda</td>
<td>torrance</td>
</tr>
<tr>
<td>horizon</td>
<td>seattle</td>
</tr>
<tr>
<td>egyptair</td>
<td>cairo</td>
</tr>
<tr>
<td>adobe</td>
<td>san_jose</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

[Example from Suciu et al 2011]
**Querying: relational database**

<table>
<thead>
<tr>
<th>ProducesProduct</th>
<th>HeadquarteredIn</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Company</strong></td>
<td><strong>Company</strong></td>
</tr>
<tr>
<td>sony</td>
<td>microsoft</td>
</tr>
<tr>
<td>microsoft</td>
<td>ibm</td>
</tr>
<tr>
<td>ibm</td>
<td>emirates_airlines</td>
</tr>
<tr>
<td>microsoft</td>
<td>honda</td>
</tr>
<tr>
<td>adobe</td>
<td>egyptair</td>
</tr>
<tr>
<td>adobe</td>
<td>adobe</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Product</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>walkman</td>
<td>redmond</td>
</tr>
<tr>
<td>mac_os_x</td>
<td>san_jose</td>
</tr>
<tr>
<td>personal_computer</td>
<td>dubai</td>
</tr>
<tr>
<td>mac_os</td>
<td>torrance</td>
</tr>
<tr>
<td>adobe_indesign</td>
<td>seattle</td>
</tr>
<tr>
<td>adobe_dreamweaver</td>
<td>cairo</td>
</tr>
<tr>
<td>...</td>
<td>san_jose</td>
</tr>
</tbody>
</table>

```sql
select x.Product, x.Company
from ProducesProduct x, HeadquarteredIn y
where x.Company=y.Company and y.City='san_jose'
```
select x.Product, x.Company  
from ProducesProduct x, HeadquarteredIn y  
where x.Company=y.Company and  
y.City='san_jose'
select x.Product, x.Company
from ProducesProduct x, HeadquarteredIn y
where x.Company=y.Company and
y.City='san_jose'
**Querying: relational database**

**ProducesProduct**

<table>
<thead>
<tr>
<th>Company</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>sony</td>
<td>walkman</td>
</tr>
<tr>
<td>microsoft</td>
<td>mac_os_x</td>
</tr>
<tr>
<td>ibm</td>
<td>personal_computer</td>
</tr>
<tr>
<td>microsoft</td>
<td>mac_os</td>
</tr>
<tr>
<td>adobe</td>
<td>adobe_indesign</td>
</tr>
<tr>
<td>adobe</td>
<td>adobe_dreamweaver</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**HeadquarteredIn**

<table>
<thead>
<tr>
<th>Company</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>microsoft</td>
<td>redmond</td>
</tr>
<tr>
<td>ibm</td>
<td>san_jose</td>
</tr>
<tr>
<td>emirates_airlines</td>
<td>dubai</td>
</tr>
<tr>
<td>honda</td>
<td>torrance</td>
</tr>
<tr>
<td>horizon</td>
<td>seattle</td>
</tr>
<tr>
<td>egyptair</td>
<td>cairo</td>
</tr>
<tr>
<td>adobe</td>
<td>san_jose</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

```sql
SELECT x.Product, x.Company
FROM ProducesProduct x, HeadquarteredIn y
WHERE x.Company = y.Company AND y.City = 'san_jose';
```

**Example from Suciu et al 2011**
select x.Product, x.Company
from ProducesProduct x, HeadquarteredIn y
where x.Company=y.Company and
  y.City='san_jose'

<table>
<thead>
<tr>
<th>Company</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>sony</td>
<td>walkman</td>
</tr>
<tr>
<td>microsoft</td>
<td>mac_os_x</td>
</tr>
<tr>
<td>ibm</td>
<td>personal_computer</td>
</tr>
<tr>
<td>microsoft</td>
<td>mac_os</td>
</tr>
<tr>
<td>adobe</td>
<td>adobe_indesign</td>
</tr>
<tr>
<td>adobe</td>
<td>adobe_dreamweaver</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Company</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>microsoft</td>
<td>redmond</td>
</tr>
<tr>
<td>ibm</td>
<td>san_jose</td>
</tr>
<tr>
<td>emirates_airlines</td>
<td>dubai</td>
</tr>
<tr>
<td>honda</td>
<td>torrance</td>
</tr>
<tr>
<td>horizon</td>
<td>seattle</td>
</tr>
<tr>
<td>egyptair</td>
<td>cairo</td>
</tr>
<tr>
<td>adobe</td>
<td>san_jose</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

[Example from Suciu et al 2011]

<table>
<thead>
<tr>
<th>Product</th>
<th>Company</th>
</tr>
</thead>
<tbody>
<tr>
<td>personal_computer</td>
<td>ibm</td>
</tr>
<tr>
<td>adobe_indesign</td>
<td>adobe</td>
</tr>
<tr>
<td>adobe_dreamweaver</td>
<td>adobe</td>
</tr>
</tbody>
</table>

[Example from Suciu et al 2011]
Querying: probabilistic db

<table>
<thead>
<tr>
<th>Company</th>
<th>Product</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>sony</td>
<td>walkman</td>
<td>0.96</td>
</tr>
<tr>
<td>microsoft</td>
<td>mac_os_x</td>
<td>0.96</td>
</tr>
<tr>
<td>ibm</td>
<td>personal_computer</td>
<td>0.96</td>
</tr>
<tr>
<td>microsoft</td>
<td>mac_os</td>
<td>0.9</td>
</tr>
<tr>
<td>adobe</td>
<td>adobe_indeisgn</td>
<td>0.9</td>
</tr>
<tr>
<td>adobe</td>
<td>adobe_dreamweaver</td>
<td>0.87</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Company</th>
<th>City</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>microsoft</td>
<td>redmond</td>
<td>1.0</td>
</tr>
<tr>
<td>ibm</td>
<td>san_jose</td>
<td>0.99</td>
</tr>
<tr>
<td>emirates_airlines</td>
<td>dubai</td>
<td>0.93</td>
</tr>
<tr>
<td>honda</td>
<td>torrance</td>
<td>0.93</td>
</tr>
<tr>
<td>horizon</td>
<td>seattle</td>
<td>0.93</td>
</tr>
<tr>
<td>egyptair</td>
<td>cairo</td>
<td>0.93</td>
</tr>
<tr>
<td>adobe</td>
<td>san_jose</td>
<td>0.93</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

[Example from Suciu et al 2011]
select x.Product, x.Company
from ProducesProduct x, HeadquarteredIn y
where x.Company=y.Company and
y.City='san_jose'

same query -
probabilities handled implicitly

[Example from Suciu et al 2011]
**Querying: probabilistic db**

```
select x.Product, x.Company
from ProducesProduct x, HeadquarteredIn y
where x.Company=y.Company and
y.City='san_jose'
```

\[
0.96 \times 0.99 = 0.95
\]

[Example from Suciu et al 2011]
**Querying: probabilistic db**

<table>
<thead>
<tr>
<th>Company</th>
<th>Product</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>sony</td>
<td>walkman</td>
<td>0.96</td>
</tr>
<tr>
<td>microsoft</td>
<td>mac_os_x</td>
<td>0.96</td>
</tr>
<tr>
<td>ibm</td>
<td>personal_computer</td>
<td>0.96</td>
</tr>
<tr>
<td>microsoft</td>
<td>mac_os</td>
<td>0.9</td>
</tr>
<tr>
<td><strong>adobe</strong></td>
<td>adobe_indesign</td>
<td>0.9</td>
</tr>
<tr>
<td>adobe</td>
<td>adobe_dreamweaver</td>
<td>0.87</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Company</th>
<th>City</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>microsoft</td>
<td>redmond</td>
<td>1.00</td>
</tr>
<tr>
<td>ibm</td>
<td>san_jose</td>
<td>0.99</td>
</tr>
<tr>
<td>emirates_airlines</td>
<td>dubai</td>
<td>0.93</td>
</tr>
<tr>
<td>honda</td>
<td>torrance</td>
<td>0.93</td>
</tr>
<tr>
<td>horizon</td>
<td>seattle</td>
<td>0.93</td>
</tr>
<tr>
<td>egyptair</td>
<td>cairo</td>
<td>0.93</td>
</tr>
<tr>
<td><strong>adobe</strong></td>
<td>san_jose</td>
<td>0.93</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

The query is:

```sql
select x.Product, x.Company
from ProducesProduct x, HeadquarteredIn y
where x.Company=y.Company and
y.City='san_jose'
```

The probability is:

```
0.9 * 0.93 = 0.83
```

[Example from Suciu et al 2011]
**Querying: probabilistic db**

<table>
<thead>
<tr>
<th>Company</th>
<th>Product</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>sony</td>
<td>walkman</td>
<td>0.96</td>
</tr>
<tr>
<td>microsoft</td>
<td>mac_os_x</td>
<td>0.96</td>
</tr>
<tr>
<td>ibm</td>
<td>personal_computer</td>
<td>0.96</td>
</tr>
<tr>
<td>microsoft</td>
<td>mac_os</td>
<td>0.9</td>
</tr>
<tr>
<td>adobe</td>
<td>adobe_indesign</td>
<td>0.9</td>
</tr>
<tr>
<td>adobe</td>
<td>adobe_dreamweaver</td>
<td>0.87</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Company</th>
<th>City</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>microsoft</td>
<td>redmond</td>
<td>1.00</td>
</tr>
<tr>
<td>ibm</td>
<td>san_jose</td>
<td>0.99</td>
</tr>
<tr>
<td>emirates_airlines</td>
<td>dubai</td>
<td>0.93</td>
</tr>
<tr>
<td>honda</td>
<td>torrance</td>
<td>0.93</td>
</tr>
<tr>
<td>horizon</td>
<td>seattle</td>
<td>0.93</td>
</tr>
<tr>
<td>egyptair</td>
<td>cairo</td>
<td>0.93</td>
</tr>
<tr>
<td>adobe</td>
<td>san_jose</td>
<td>0.93</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

```
select x.Product, x.Company
from ProducesProduct x, HeadquarteredIn y
where x.Company=y.Company and
y.City='san_jose'
```

![Example from Suciu et al 2011]
Querying: probabilistic db

<table>
<thead>
<tr>
<th>Company</th>
<th>Product</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>sony</td>
<td>walkman</td>
<td>0.96</td>
</tr>
<tr>
<td>microsoft</td>
<td>mac_os_x</td>
<td>0.96</td>
</tr>
<tr>
<td>ibm</td>
<td>personal_computer</td>
<td>0.96</td>
</tr>
<tr>
<td>microsoft</td>
<td>mac_os</td>
<td>0.9</td>
</tr>
<tr>
<td>adobe</td>
<td>adobe_indesign</td>
<td>0.9</td>
</tr>
<tr>
<td>adobe</td>
<td>adobe_dreamweaver</td>
<td>0.87</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Company</th>
<th>City</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>microsoft</td>
<td>redmond</td>
<td>1.00</td>
</tr>
<tr>
<td>ibm</td>
<td>san_jose</td>
<td>0.99</td>
</tr>
<tr>
<td>emirates_airlines</td>
<td>dubai</td>
<td>0.93</td>
</tr>
<tr>
<td>honda</td>
<td>torrance</td>
<td>0.93</td>
</tr>
<tr>
<td>horizon</td>
<td>seattle</td>
<td>0.93</td>
</tr>
<tr>
<td>egyptair</td>
<td>cairo</td>
<td>0.93</td>
</tr>
<tr>
<td>adobe</td>
<td>san_jose</td>
<td>0.93</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

```
select x.Product, x.Company
from ProducesProduct x, HeadquarteredIn y
where x.Company=y.Company and y.City='san_jose'
```

answer tuples ranked by probability

<table>
<thead>
<tr>
<th>Product</th>
<th>Company</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>personal_computer</td>
<td>ibm</td>
<td>0.95</td>
</tr>
<tr>
<td>adobe_indesign</td>
<td>adobe</td>
<td>0.83</td>
</tr>
<tr>
<td>adobe_dreamweaver</td>
<td>adobe</td>
<td>0.80</td>
</tr>
</tbody>
</table>

[Example from Suciu et al 2011]
PDB with tuple-level uncertainty in ProbLog?

```
select x.Product, x.Company
from ProducesProduct x, HeadquarteredIn y
where x.Company=y.Company and
y.City='san_jose'
```

[Example from Suciu et al 2011]
PDB with tuple-level uncertainty in ProbLog?

<table>
<thead>
<tr>
<th>Company</th>
<th>Product</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>sony</td>
<td>walkman</td>
<td>0.96</td>
</tr>
<tr>
<td>microsoft</td>
<td>mac_os_x</td>
<td>0.96</td>
</tr>
<tr>
<td>ibm</td>
<td>personal_computer</td>
<td>0.96</td>
</tr>
<tr>
<td>microsoft</td>
<td>mac_os</td>
<td>0.9</td>
</tr>
<tr>
<td>adobe</td>
<td>adobe_indesign</td>
<td>0.9</td>
</tr>
<tr>
<td>adobe</td>
<td>adobe_dreamweaver</td>
<td>0.87</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

[Example from Suciu et al 2011]
PDB with tuple-level uncertainty in ProbLog?

<table>
<thead>
<tr>
<th>Company</th>
<th>Product</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>sony</td>
<td>walkman</td>
<td>0.96</td>
</tr>
<tr>
<td>microsoft</td>
<td>mac_os_x</td>
<td>0.96</td>
</tr>
<tr>
<td>ibm</td>
<td>personal_computer</td>
<td>0.96</td>
</tr>
<tr>
<td>microsoft</td>
<td>mac_os</td>
<td>0.9</td>
</tr>
<tr>
<td>adobe</td>
<td>adobe_indesign</td>
<td>0.9</td>
</tr>
<tr>
<td>adobe</td>
<td>adobe_dreamweaver</td>
<td>0.87</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

0.96::producesProduct(sony,walkman).
0.96::producesProduct(microsoft,mac_os_x).
0.96::producesProduct(ibm,personal_computer).
0.9::producesProduct(microsoft,mac_os).
0.9::producesProduct(adobe,adobe_indesign).
0.87::producesProduct(adobe,adobe_dreamweaver).
...

[Example from Suciu et al 2011]
PDB with tuple-level uncertainty in ProbLog?

```
select x.Product, x.Company
from ProducesProduct x, HeadquarteredIn y
where x.Company=y.Company and y.City='san_jose'
```
PDB with tuple-level uncertainty in ProbLog?

```
select x.Product, x.Company
from ProducesProduct x, HeadquarteredIn y
where x.Company=y.Company and y.City='san_jose'
```

```
result(Product,Company) :-
  producesProduct(Company,Product),
  headquarteredIn(Company,san_jose).
query(result(_,_,_)).
```
PDB with tuple-level uncertainty in ProbLog?

```
select x.Product, x.Company
from ProducesProduct x, HeadquarteredIn y
where x.Company=y.Company and y.City='san_jose'

result(Product,Company) :-
    producesProduct(Company,Product),
    headquarteredIn(Company,san_jose).
query(result(_,_)).
```
PDB with tuple-level uncertainty in ProbLog?

0.96::producesProduct(sony,walkman).
0.96::producesProduct(microsoft,mac_os_x).
0.96::producesProduct(ibm,personal_computer).
0.9::producesProduct(microsoft,mac_os).
0.9::producesProduct(adobe,adobe_indesign).
0.87::producesProduct(adobe,adobe_dreamweaver).

...  
1.00::headquarteredIn(microsoft,redmond).
0.99::headquarteredIn(ibm,san_jose).
0.93::headquarteredIn(emirates_airlines,dubai).
0.93::headquarteredIn(honda,torrance).
0.93::headquarteredIn(horizon,seattle).
0.93::headquarteredIn(egyptair,cairo).
0.93::headquarteredIn(adobe,san_jose).

...  
result(Product,Company) :- producesProduct(Company,Product),
                          headquarteredIn(Company,san_jose).
query(result(_,_)).
PDB with attribute-level uncertainty in ProbLog?

<table>
<thead>
<tr>
<th>item</th>
<th>color</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>mug</td>
<td>green</td>
<td>0.65</td>
</tr>
<tr>
<td></td>
<td>blue</td>
<td>0.35</td>
</tr>
<tr>
<td>plate</td>
<td>pink</td>
<td>0.23</td>
</tr>
<tr>
<td></td>
<td>red</td>
<td>0.14</td>
</tr>
<tr>
<td></td>
<td>purple</td>
<td>0.63</td>
</tr>
</tbody>
</table>
PDB with attribute-level uncertainty in ProbLog?

<table>
<thead>
<tr>
<th>item</th>
<th>color</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>mug</td>
<td>green</td>
<td>0.65</td>
</tr>
<tr>
<td></td>
<td>blue</td>
<td>0.35</td>
</tr>
<tr>
<td>plate</td>
<td>pink</td>
<td>0.23</td>
</tr>
<tr>
<td></td>
<td>red</td>
<td>0.14</td>
</tr>
<tr>
<td></td>
<td>purple</td>
<td>0.63</td>
</tr>
</tbody>
</table>

\[0.65::\text{color}(	ext{mug},\text{green}); \ 0.35::\text{color}(	ext{mug},\text{blue}) \implies \text{true.}\]
\[0.23::\text{color}(	ext{plate},\text{pink}); \ 0.14::\text{color}(	ext{plate},\text{red}); \]
\[0.63::\text{color}(	ext{plate},\text{purple}) \implies \text{true.}\]
ProbLog by example:

Rain or sun?
ProbLog by example:

Rain or sun?
ProbLog by example:

Rain or sun?

0.5::weather(sun,0) ; 0.5::weather(rain,0) <- true.

day 0
ProbLog by example:

**Rain or sun?**

0.5::weather(sun, 0) ; 0.5::weather(rain, 0) <- true.
ProbLog by example:

Rain or sun?

\[0.5::\text{weather(sun,0)} ; 0.5::\text{weather(rain,0)} \leftarrow \text{true.}\]

\[0.6::\text{weather(sun,T)} ; 0.4::\text{weather(rain,T)}
\leftarrow T>0, \text{Tprev is T-1, weather(sun,Tprev)}.\]
ProbLog by example:

Rain or sun?

0.5::weather(sun,0) ; 0.5::weather(rain,0) <- true.

0.6::weather(sun,T) ; 0.4::weather(rain,T)
<- T>0, Tprev is T-1, weather(sun,Tprev).
ProbLog by example:

Rain or sun?

0.5::weather(sun,0) ; 0.5::weather(rain,0) <- true.

0.6::weather(sun,T) ; 0.4::weather(rain,T)
<- T>0, Tprev is T-1, weather(sun,Tprev).

0.2::weather(sun,T) ; 0.8::weather(rain,T)
<- T>0, Tprev is T-1, weather(rain,Tprev).
ProbLog by example:

Rain or sun?

infinite possible worlds! BUT: finitely many partial worlds suffice to answer any given ground query
ProbLog by example:

Friends & smokers

person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
ProbLog by example:

**Friends & smokers**

**typed probabilistic facts**
= a probabilistic fact for each grounding

0.3::stress(X):- person(X).
0.2::influences(X,Y):-
    person(X), person(Y).

```
person(1).
person(2).
person(3).
person(4).
friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```
ProbLog by example:

Friends & smokers

typed probabilistic facts
= a probabilistic fact for each grounding

0.3::stress(X):- person(X).
0.2::influences(X,Y):-
    person(X), person(Y).

0.3::stress(1).
0.3::stress(2).
0.3::stress(3).
0.3::stress(4).

0.2::influences(1,1).
0.2::influences(1,2).
0.2::influences(1,3).
0.2::influences(1,4).

0.2::influences(2,1).
0.2::influences(2,2).
0.2::influences(2,3).
0.2::influences(2,4).

...
ProbLog by example:

**Friends & smokers**

0.3::stress(X):- person(X).
0.2::influences(X,Y):-
  person(X), person(Y).

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

person(1).
person(2).
person(3).
person(4).
friend(1,2).
frend(2,1).
frend(2,4).
frend(3,4).
frend(4,2).
ProbLog by example:

**Friends & smokers**

0.3::stress(X):- person(X).
0.2::influences(X,Y):-
    person(X), person(Y).

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

0.4::asthma(X) <- smokes(X).

annotated disjunction with implicit head atom:
with probability 0.6, nothing happens
ProbLog by example:

Friends & smokers

0.3::stress(X):- person(X).
0.2::influences(X,Y):-
    person(X), person(Y).

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

0.4::asthma(X) <- smokes(X).
ProbLog by example:

**Limited Luggage**

weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).
ProbLog by example:

**Limited Luggage**

weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).

\[
\text{P::pack(Item)} :\text{:- weight(Item,Weight), P is } 1.0/\text{Weight}.\]

**flexible probability:**
computed from the weight of the item
ProbLog by example:

**Limited Luggage**

weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).

1/6::pack(skis).
1/4::pack(boots).
1/3::pack(helmet).
1/2::pack(gloves).

P::pack(Item) :- weight(Item,Weight), P is 1.0/Weight.

**flexible probability:**
computed from the weight of the item
ProbLog by example:

Limited Luggage

\[
\begin{align*}
\text{weight}(\text{skis}, 6). \\
\text{weight}(\text{boots}, 4). \\
\text{weight}(\text{helmet}, 3). \\
\text{weight}(\text{gloves}, 2). \\
\text{P::pack(Item)} & : - \text{weight(Item,Weight)}, \quad \text{P is } 1.0/\text{Weight}. \\
\text{excess(Limit)} & : - \text{excess([skis,boots,helmet,gloves],Limit)}. \\
\end{align*}
\]

list of all items
weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).

P::pack(Item) :- weight(Item,Weight),  P is 1.0/Weight.

excess(Limit) :- excess([skis,boots,helmet,gloves],Limit).
excess([]),Limit) :- Limit<0.

excess([I|R],Limit) :-
    pack(I), weight(I,W), L is Limit-W, excess(R,L).
excess([I|R],Limit) :-
    \+pack(I), excess(R,Limit).
ProbLog by example:

Limited Luggage

weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).

P::pack(Item) :- weight(Item,Weight), P is 1.0/Weight.

excess(Limit) :- excess([skis,boots,helmet,gloves],Limit).

excess([],Limit) :- Limit<0.
excess([I|R],Limit) :-
    pack(I), weight(I,W), L is Limit-W, excess(R,L).
excess([I|R],Limit) :-
    \+pack(I), excess(R,Limit).

pack first item, decrease limit by its weight, and continue with rest of items
ProbLog by example:

Limited Luggage

weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).

\[ \text{P::pack(Item)} \leftarrow \text{weight(Item,Weight)}, \ P \text{ is } 1.0/\text{Weight}. \]

\[ \text{excess(Limit)} \leftarrow \text{excess([skis,boots,helmet,gloves],Limit)}. \]

\[ \text{excess([],Limit)} \leftarrow \text{Limit}<0. \]
\[ \text{excess([I|R],Limit)} \leftarrow \]
\[ \text{pack(I)}, \text{weight(I.W)}, \ L \text{ is Limit-W, excess(R,L)}. \]
\[ \text{excess([I|R],Limit)} \leftarrow \]
\[ \text{\ \ +pack(I)}, \text{excess(R,Limit)}. \]

do not pack first item,
continue with rest of items
ProbLog by example:

Limited Luggage

weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).

P::pack(Item) :- weight(Item,Weight), P is 1.0/Weight.

excess(Limit) :- excess([],Limit).

excess([I|R],Limit) :- pack(I), weight(I,W), L is Limit-W, excess(R,L).

no items left: did we add too much?
ProbLog by example:

Limited Luggage

weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).

P::pack(Item) :- weight(Item,Weight),  P is 1.0/Weight.

excess(Limit) :- excess([skis,boots,helmet,gloves],Limit).

excess([],Limit) :- Limit<0.
excess([I|R],Limit) :-
    pack(I), weight(I,W), L is Limit-W, excess(R,L).
excess([I|R],Limit) :-
    \+pack(I), excess(R,Limit).
Summary: ProbLog Syntax

• input database: ground facts
  
  person(bob).

• probabilistic facts
  
  0.5::stress(bob).

• typed probabilistic facts
  (body deterministic)

  0.5::stress(X) :- person(X).

• flexible probabilities
  
  P::pack(Item) :- weight(Item,W),
  P is 1.0/W.

• annotated disjunctions
  
  0.4::asthma(X) <- smokes(X).

  0.5::weather(sun,0) ; 0.5::weather(rain,0) <- true.

• Prolog clauses
  
  smokes(X) :- influences(Y,X), smokes(Y).

  excess([I|R],Limit) :- \+pack(I), excess(R,Limit).
ProbLog example:

Friends & smokers

0.5::stress(1).
0.1::stress(2).
0.8::stress(3).
0.3::stress(4).

0.9::friend(1,2).
0.8::friend(2,1).
0.3::friend(2,4).
0.7::friend(3,4).
0.1::friend(4,2).

smokes(X) :- stress(X).
smokes(X) :-
    friend(Y,X), smokes(Y).
ProbLog example:

Friends & smokers

0.5::stress(1).
0.1::stress(2).
0.8::stress(3).
0.3::stress(4).

smokes(X) :- stress(X).
smokes(X) :-
friend(Y,X), smokes(Y).

example possible world

friend(1,2).
frend(3,4).
friend(4,2).
stress(1).
stress(2).
stress(3).
smokes(1).
smokes(2).
smokes(3).
smokes(4).
ProbLog example:

**Friends & smokers**

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

Example possible world

```
friend(1,2).
friend(3,4).
friend(4,2).
stress(1).
stress(2).
stress(3).
smokes(1).
smokes(2).
smokes(3).
smokes(4).
```

- Several instances of `smokes(X)` in same world
- `smokes(2)`: multiple derivations in same world
- Distribution over worlds not (always) a distribution over computations / answers
ProbLog by example:

Rain or sun?

\[\begin{align*}
0.5::\text{weather}(\text{sun},0) & ; 0.5::\text{weather}(\text{rain},0) \leftarrow \text{true}. \\
0.6::\text{weather}(\text{sun},T) & ; 0.4::\text{weather}(\text{rain},T) \\
& \leftarrow T>0, T_{\text{prev}} \text{ is } T-1, \text{weather}(\text{sun},T_{\text{prev}}). \\
0.2::\text{weather}(\text{sun},T) & ; 0.8::\text{weather}(\text{rain},T) \\
& \leftarrow T>0, T_{\text{prev}} \text{ is } T-1, \text{weather}(\text{rain},T_{\text{prev}}).
\end{align*}\]

\[\leq 1\text{ proof for a ground query per possible world} \rightarrow \text{distribution over worlds is distribution over derivations!}\]
Possible worlds

?- weather(rain,2).

\[ P = P_1 + P_2 + P_3 + P_4 \]

\[ P_1 = 0.12 \]

\[ P_2 = 0.16 \]

\[ P_3 = 0.04 \]

\[ P_4 = 0.32 \]
Mutually Exclusive Rules:

no two rules apply simultaneously

\[
\begin{align*}
0.5::\text{weather}(\text{sun}, 0) & \; ; \; 0.5::\text{weather}(\text{rain}, 0) \leftarrow \text{true}. \\
0.6::\text{weather}(\text{sun}, T) & \; ; \; 0.4::\text{weather}(\text{rain}, T) \leftarrow T > 0, \ T_{\text{prev}} \text{ is } T-1, \ \text{weather}(\text{sun}, T_{\text{prev}}). \\
0.2::\text{weather}(\text{sun}, T) & \; ; \; 0.8::\text{weather}(\text{rain}, T) \leftarrow T > 0, \ T_{\text{prev}} \text{ is } T-1, \ \text{weather}(\text{rain}, T_{\text{prev}}). 
\end{align*}
\]
Mutually Exclusive Rules:
no two rules apply simultaneously

first rule for day 0, others for later days

\[
\begin{align*}
0.5::\text{weather}(\text{sun},0) & \; ; \; 0.5::\text{weather}(\text{rain},0) \quad \text{<- true.} \\
0.6::\text{weather}(\text{sun},T) & \; ; \; 0.4::\text{weather}(\text{rain},T) \\
& \quad \text{<- $T>0$, $T_{prev}$ is $T-1$, weather(sun,$T_{prev}$).} \\
0.2::\text{weather}(\text{sun},T) & \; ; \; 0.8::\text{weather}(\text{rain},T) \\
& \quad \text{<- $T>0$, $T_{prev}$ is $T-1$, weather(rain,$T_{prev}$).}
\end{align*}
\]
Mutually Exclusive Rules:
no two rules apply simultaneously

first rule for day 0, others for later days

day 0: either sun or rain

0.5::weather(sun,0) ; 0.5::weather(rain,0) <- true.

0.6::weather(sun,T) ; 0.4::weather(rain,T)
  <- T>0, Tprev is T-1, weather(sun,Tprev).

0.2::weather(sun,T) ; 0.8::weather(rain,T)
  <- T>0, Tprev is T-1, weather(rain,Tprev).
Mutually Exclusive Rules:
no two rules apply simultaneously

first rule for day 0, others for later days

day 0: either sun or rain

0.5::weather(sun,0) ; 0.5::weather(rain,0) <- true.

0.6::weather(sun,T) ; 0.4::weather(rain,T)
<- T>0, Tprev is T-1, weather(sun,Tprev).

0.2::weather(sun,T) ; 0.8::weather(rain,T)
<- T>0, Tprev is T-1, weather(rain,Tprev).

rules for T>0 cover mutually exclusive cases on previous day
PRISM

- Another probabilistic Prolog based on the distribution semantics
- Mutual exclusiveness assumption
  - allows for efficient inference by dynamic programming, cf. probabilistic grammars
  - but excludes certain models, e.g., smokers
PRISM

- “multi-valued random switches” = annotated disjunctions with body $true$
- switch gives fresh result on each call
- Prolog rules
- limited support for negation (compiling away)
values(init,[sun,rain]).
values(tr(_),[sun,rain]).

:- set_sw(init,[0.5,0.5]).
:- set_sw(tr(sun),[0.6,0.4]).
:- set_sw(tr(rain),[0.2,0.8]).

weather(W,Time) :-
    Time >= 0,
    msw(init,W0),
    w(0,Time,W0,W).

w(T,T,W,W).

w(Now,T,WNOW,WT) :-
    Now < T,
    msw(tr(WNOW),WNext),
    Next is Now+1,
    w(Next,T,WNext,WT).
values(init,[sun,rain]).
values(tr(_),[sun,rain]).
:- set_sw(init,[0.5,0.5]).
:- set_sw(tr(sun),[0.6,0.4]).
:- set_sw(tr(rain),[0.2,0.8]).

weather(W,Time) :-
    Time >= 0,
    msw(init,W0),
    w(0,Time,W0,W).

w(T,T,W,W).
w(Now,T,WN,WT) :-
    Now < T,
    msw(tr(WN),WNext),
    Next is Now+1,
    w(Next,T,WNext,WT).
Weather in PRISM

values(init,[sun,rain]).
values(tr(_),[sun,rain]).

:- set_sw(init,[0.5,0.5]).
:- set_sw(tr(sun),[0.6,0.4]).
:- set_sw(tr(rain),[0.2,0.8]).

weather(W,Time) :-
    Time >= 0,
    msw(init,W0),
    w(0,Time,W0,W).

w(T,T,W,W).

w(Now,T,WNow,WT) :-
    Now < T,
    msw(tr(WNow),WNext),
    Next is Now+1,
    w(Next,T,WNext,WT).
values(init,[sun,rain]).
values(tr(_),[sun,rain]).
:- set_sw(init,[0.5,0.5]).
:- set_sw(tr(sun),[0.6,0.4]).
:- set_sw(tr(rain),[0.2,0.8]).

weather(W,Time) :-
    Time >= 0,
    set_W0(W0),
    w(0,Time,W0,W).

w(T,T,W,W).

w(Now,T,WNow,WT) :-
    Now < T,
    msw(tr(WNow),WNext),
    Next is Now+1,
    w(Next,T,WNext,WT).
Weather in PRISM

random variables and their values
probability distributions

set \( W_0 \) to random value of \( \text{init} \)
set \( W_{\text{Next}} \) to random value of \( \text{tr}(W_{\text{Now}}) \), using fresh value on every call
Weather in PRISM / ProbLog

values(init, [sun, rain]).
values(tr(_), [sun, rain]).
:- set_sw(init, [0.5, 0.5]).
:- set_sw(tr(sun), [0.6, 0.4]).
:- set_sw(tr(rain), [0.2, 0.8]).

weather(W, Time) :-
    Time >= 0,
    msw(init, W0),
    w(0, Time, W0, W).

w(T, T, W, W).

w(Now, T, WNow, WT) :-
    Now < T,
    msw(tr(WNow), WNext),
    Next is Now+1,
    w(Next, T, WNext, WT).

weather(W, Time) :-
    Time >= 0,
    init(W0),
    w(0, Time, W0, W).

w(T, T, W, W).

w(Now, T, WNow, WT) :-
    Now < T,
    tr(Now, WNow, WNext),
    Next is Now+1,
    w(Next, T, WNext, WT).

ProbLog needs to explicitly use different facts at each call.
Probabilistic Prologs: Two Views

• Distribution semantics:
  • probability distribution over interpretations
  • degree of belief

• Stochastic Logic Programs (SLPs):
  • probability distribution over query answers
  • like in probabilistic grammars
Probabilistic Prologs: Two Views

- Distribution semantics:
  - probability distribution over interpretations
  - degree of belief

- Stochastic Logic Programs (SLPs):
  - probability distribution over query answers
  - like in probabilistic grammars
Probabilistic Context Free Grammars

1.0 : S -> NP, VP
1.0 : NP -> Art, Noun
0.6 : Art -> a
0.4 : Art -> the
0.1 : Noun -> turtle
0.1 : Noun -> turtles
...
0.5 : VP -> Verb
0.5 : VP -> Verb, NP
0.05 : Verb -> sleep
0.05 : Verb -> sleeps
....

P(parse tree) = 1x1x.5x.1x.4x.05
PCFGs

\[ P(\text{parse tree}) = \prod_i p_i^{c_i} \]
where \( p_i \) is the probability of rule \( i \)
and \( c_i \) the number of times
it is used in the parse tree

\[ P(\text{sentence}) = \sum_{p:\text{parsetree}} P(p) \]

Observe that derivations always succeed, that is
\( S \rightarrow T, Q \) and \( T \rightarrow R, U \)
always yields
\( S \rightarrow R, U, Q \)
Probabilistic Definite Clause Grammar

1.0 $S \rightarrow NP(\text{Num}), VP(\text{Num})$
1.0 $NP(\text{Num}) \rightarrow \text{Art}(\text{Num}), \text{Noun}(\text{Num})$
0.6 $\text{Art}(\text{sing}) \rightarrow \text{a}$
0.2 $\text{Art}(\text{sing}) \rightarrow \text{the}$
0.2 $\text{Art}(\text{plur}) \rightarrow \text{the}$
0.1 $\text{Noun}(\text{sing}) \rightarrow \text{turtle}$
0.1 $\text{Noun}(\text{plur}) \rightarrow \text{turtles}$

$\ldots$
0.5 $VP(\text{Num}) \rightarrow \text{Verb}(\text{Num})$
0.5 $VP(\text{Num}) \rightarrow \text{Verb}(\text{Num}), NP(\text{Num})$
0.05 $\text{Verb}(\text{sing}) \rightarrow \text{sleeps}$
0.05 $\text{Verb}(\text{plur}) \rightarrow \text{sleep}$

$\ldots$

\[
P(\text{derivation tree}) = 1 \times 1 \times 0.5 \times 1 \times 0.2 \times 0.05
\]

Stochastic Logic Programs

[Muggleton, Cussens]
In SLP notation

sentence(A, B) :- noun_phrase(C, A, D), verb_phrase(C, D, B).
noun_phrase(A, B, C) :- article(A, B, D), noun(A, D, C).
verb_phrase(A, B, C) :- intransitive_verb(A, B, C).
article(singular, A, B) :- terminal(A, a, B).
article(singular, A, B) :- terminal(A, the, B).
article(plural, A, B) :- terminal(A, the, B).
noun(singular, A, B) :- terminal(A, turtle, B).
noun(plural, A, B) :- terminal(A, turtles, B).
intransitive_verb(singular, A, B) :- terminal(A, sleeps, B).
intransitive_verb(plural, A, B) :- terminal(A, sleep, B).
terminal([A|B], A, B).

P(s([the,turtles,sleep],[[]]=1/6
Probabilistic DCG

1.0  S -> NP(Num), VP(Num)
1.0 NP(Num) -> Art(Num), Noun(Num)
0.6 Art(sing) -> a
0.2 Art(sing) -> the
0.2 Art(plur) -> the
0.1 Noun(sing) -> turtle
0.1 Noun(plur) -> turtles
...
0.5 VP(Num) -> Verb(Num)
0.5 VP(Num) -> Verb(Num), NP(Num)
0.05 Verb(sing) -> sleeps
0.05 Verb(plur) -> sleep
...

P(derivation tree) = 1x1x.5x.1x .2 x.05
Probabilistic DCG

1.0  S -> NP(Num), VP(Num)
1.0 NP(Num) -> Art(Num), Noun(Num)
0.6 Art(sing) -> a
0.2 Art(sing) -> the
0.2 Art(plur) -> the
0.1 Noun(sing) -> turtle
0.1 Noun(plur) -> turtles
...
0.5 VP(Num) -> Verb(Num)
0.5 VP(Num) -> Verb(Num), NP(Num)
0.05 Verb(sing) -> sleeps
0.05 Verb(plur) -> sleep
...

What about “A turtles sleeps”? 

P(derivation tree) = 1x1x.5x.1x .2 x.05
SLPs

\[ P_d(\text{derivation}) = \prod_i p_i^{c_i} \]
where \( p_i \) is the probability of rule \( i \) and \( c_i \) the number of times it is used in the parse tree.

Observe that some derivations now fail due to unification, 
\( np(\text{Num}) \rightarrow \text{art}(\text{Num}), \text{noun}(\text{Num}) \) and \( \text{art}(\text{sing}) \rightarrow a \text{noun}(\text{plural}) \rightarrow \text{turtles} \)

Normalization necessary
\[ P_s(\text{proof}) = \frac{P_d(\text{proof})}{\sum_i P_d(\text{proof}_i)} \]
ProPPR

- A variation on SLPs
- Integrating concepts from Personalized Page Rank
- Fast inference and rule learning abilities
- Used by CMU group for NELL (Never Ending Learning)
- See [Wang et al., CIKM 13, arXiv:1404.3301]
Sample ProPPR program….

about(X,Z) :- handLabeled(X,Z)  # base.
about(X,Z) :- sim(X,Y),about(Y,Z)  # prop.
sim(X,Y) :- links(X,Y)  # sim,link.
sim(X,Y) :-
    hasWord(X,W),hasWord(Y,W),
    linkedBy(X,Y,W)  # sim,word.
linkedBy(X,Y,W) :- true  # by(W).

Horn rules  features of rules (vars from head ok)

[Slide by William Cohen]
.. and search space...

[Slide by William Cohen]
D’oh! This is a graph!

.. and search space...

[Slide by William Cohen]
Score for a query soln (e.g., “Z=sport” for “about(a,Z)”) depends on probability of reaching a node

- learn transition probabilities based on features of the rules
- implicit “reset” transitions with (p ≥ α) back to query node
- Looking for answers supported by many short proofs
• Score for a query soln (e.g., “Z=sport” for “about(a,Z)”) depends on probability of reaching a ☐ node
  • learn transition probabilities based on features of the rules
  • implicit “reset” transitions with \((p \geq \alpha)\) back to query node
  • Looking for answers supported by many short proofs

*Exactly as in Stochastic Logic Programs [Cussens, 2001]
• Score for a query soln (e.g., “Z=sport” for “about(a,Z)”) depends on probability of reaching a □ node*
  • learn transition probabilities based on features of the rules
  • implicit “reset” transitions with (p≥α) back to query node
  • Looking for answers supported by many short proofs

“Grounding” size is O(1/αε) ... ie independent of DB size ➔ fast approx incremental inference (Reid,Lang,Chung, 08)

*Exactly as in Stochastic Logic Programs [Cussens, 2001]

[Slide by William Cohen]
• Score for a query soln (e.g., “Z=sport” for “about(a,Z)”) depends on probability of reaching a □ node*
  • learn transition probabilities based on features of the rules
  • implicit “reset” transitions with (p≥α) back to query node
  • Looking for answers supported by many short proofs

“Grounding” size is O(1/αε) ... ie independent of DB size ➔ fast approx incremental inference (Reid, Lang, Chung, 08)

Learning: supervised variant of personalized PageRank (Backstrom & Leskovic, 2011)

*Exactly as in Stochastic Logic Programs [Cussens, 2001]
Probabilistic Programming
Languages outside LP

- IBAL [Pfeffer 01]
- Figaro [Pfeffer 09]
- Church [Goodman et al 08]
- BLOG [Milch et al 05]
- and many more appearing recently
Church
probabilistic functional programming
[Goodman et al, UAI 08]

Dealing with uncertainty
Reasoning with relational data
Learning

http://probmods.org
Church
probabilistic functional programming
[Goodman et al, UAI 08]

(\texttt{define plus5 (lambda (x) (+ x 5)))}

(\texttt{map plus5 ' (1 2 3)})

Dealing with uncertainty

functional programming

Learning

http://probmods.org
Church
probabilistic functional programming
[Goodman et al, UAI 08]

Dealing with uncertainty

Learning

functional programming

one execution

(define plus5 (lambda (x) (+ x 5)))
(map plus5 '(1 2 3))

http://probmods.org
Church
probabilistic functional programming
[Goodman et al, UAI 08]

(random primitives)

One execution

(define plus5 (lambda (x) (+ x 5)))
(map plus5 '(1 2 3))

(define randplus5 (lambda (x) (if (flip 0.6) (+ x 5) x)))
(map randplus5 '(1 2 3))

[Goodman et al, UAI 08]
http://probmods.org
Church
probabilistic functional programming
[Goodman et al, UAI 08]

one execution
(define plus5 (lambda (x) (+ x 5)))
(map plus5 '(1 2 3))

several possible executions
(define randplus5
(lambda (x) (if (flip 0.6)
(+ x 5)
  x))
(map randplus5 '(1 2 3))

random primitives

functional programming

Learning

http://probmods.org
Church
probabilistic functional programming
[Goodman et al, UAI 08]

probabilistic primitives + functional program → distribution over possible executions

one execution

(define plus5 (lambda (x) (+ x 5)))
(map plus5 '(1 2 3))

several possible executions

(define randplus5 (lambda (x) (if (flip 0.6) (+ x 5) x)))
(map randplus5 '(1 2 3))

random primitives

Learning

Goodman et al, UAI 08
http://probmods.org
Church Example

\[(\text{define randplus5} \ (\lambda (x) \ (\text{if} \ (\text{flip} \ 0.6) \ (+ x \ 5) \ x)))\]

\[(\text{map randplus5} \ '(1 \ 2))\]

- Random primitive
- \(P(\text{true}) = 0.6\)
- Function
- Arguments
- Definition
- Apply function to each element
- List
Church Example

\[(\text{define randplus5} \ (\lambda (x) (\text{if} \ (\text{flip} \ 0.6) \ (+ \ x \ 5) \ x)))\]

\[(\text{map randplus5} \ '(1 \ 2))\]

- \text{function}: \(\text{randplus5}\)
- \text{arguments}: \((x)\)
- \text{definition}: \(\text{if} \ (\text{flip} \ 0.6) \ (+ \ x \ 5) \ x\)
- \text{random primitive}: \(P(\text{true}) = 0.6\)

Result:
- \((1 \ 2)\) with \(0.4 \times 0.4\)
- \((1 \ 7)\) with \(0.4 \times 0.6\)
- \((6 \ 2)\) with \(0.6 \times 0.4\)
- \((6 \ 7)\) with \(0.6 \times 0.6\)
in ProbLog?

(define randplus5 (lambda (x) (if (flip 0.6) (+ x 5) x)))

(map randplus5 '(1 2))
in ProbLog?

(define randplus5 (lambda (x) (if (flip 0.6) (+ x 5) x)))

(map randplus5 '(1 2))

0.4::p5(N,N);0.6::p5(N,M) <- M is N+5.
lp5([],[]).
lp5([N|L],[M|K]) :-
    p5(N,M),
    lp5(L,K).

query(lp5([1,2],_)).
in ProbLog?

(define randplus5 (lambda (x) (if (flip 0.6) (+ x 5) x)))

(map randplus5 '(1 2))

0.4::p5(N,N);0.6::p5(N,M) <- M is N+5.
lp5([],[]).
lp5([N|L],[M|K]) :-
    p5(N,M),
    lp5(L,K).

query(lp5([1,2],_)).

result: (1 2) with 0.4×0.4
        (1 7) with 0.4×0.6
        (6 2) with 0.6×0.4
        (6 7) with 0.6×0.6
results for \([1,1]\) ?

\[
\text{(define randplus5 (lambda (x) (if (flip 0.6) (+ x 5) x)))}
\]

\[
\text{(map randplus5 '(1 1))}
\]

\[
0.4::\text{p5}(N,N);0.6::\text{p5}(N,M) \leftarrow M \text{ is } N+5.
\]

\[
\text{lp5}([],[]).
\]

\[
\text{lp5}([N|L],[M|K]) \leftarrow
\]

\[
\quad \text{p5}(N,M),
\]

\[
\quad \text{lp5}(L,K).
\]

\[
\text{query(lp5([1,1],_)).}
\]
results for \([1,1]?)

\[
\begin{align*}
(\text{define randplus5} & \ (\lambda (x) \ (\text{if} \ (\text{flip} \ 0.6) \ (+ \ x \ 5) \ x))) \\
(\text{map randplus5} & \ '((1 \ 1))
\end{align*}
\]

Church result:

\[
\begin{align*}
(1 \ 1) \ &\text{with} \ 0.4 \times 0.4 \\
(1 \ 6) \ &\text{with} \ 0.4 \times 0.6 \\
(6 \ 1) \ &\text{with} \ 0.6 \times 0.4 \\
(6 \ 6) \ &\text{with} \ 0.6 \times 0.6
\end{align*}
\]

\[
\begin{align*}
0.4 : : p5(N,N); 0.6 : : p5(N,M) \ &\text{<- M is N+5.} \\
lp5([],[]). \\
lp5([N|L],[M|K]) \ &\text{:-} \\
& \quad \ p5(N,M), \quad \ lp5(L,K).
\end{align*}
\]

query(lp5([1,1],_)).
results for \([1, 1]\)?

(define randplus5 (lambda (x) (if (flip 0.6) (+ x 5) x)))

(map randplus5 '(1 1))

Church result: (1 1) with 0.4 × 0.4
(1 6) with 0.4 × 0.6
(6 1) with 0.6 × 0.4
(6 6) with 0.6 × 0.6

ProbLog result: (1 1) with 0.4
(1 6) with 0.0
(6 1) with 0.0
(6 6) with 0.6
results for \([1,1]\)?

\[(\text{define randplus5 } (\lambda(x) (\text{if (flip 0.6) (+ x 5) x)) ))\]

\[(\text{map randplus5 } '(1 1))\]

Church result: (1 1) with 0.4×0.4
(1 6) with 0.4×0.6
(6 1) with 0.6×0.4
(6 6) with 0.6×0.6

ProbLog result: (1 1) with 0.4
(1 6) with 0.0
(6 1) with 0.0
(6 6) with 0.6

\[0.4::p5(N,N); 0.6::p5(N,M) \leftarrow M \text{ is } N+5.\]

\[\text{lp5([],[]).} \]
\[\text{lp5([N|L],[M|K]) :-} \]
\[\quad p5(N,M),\]
\[\quad \text{lp5(L,K).} \]

\[\text{query(lp5([1,1],_)).} \]

stochastic memoization
in ProbLog?

(define randplus5 (lambda (x) (if (flip 0.6) (+ x 5) x)))

(map randplus5 '(1 1))
in ProbLog?

(define randplus5 (lambda (x) (if (flip 0.6) (+ x 5) x)))

(map randplus5 '(1 1))

0.4::p5(N,N,ID);0.6::p5(N,M,ID) <- M is N+5.
lp5([],[]).
lp5([[N|L],[M|K]]) :-
  p5(N,M,L),
  lp5(L,K).

query(lp5([1,1],_)).
(define randplus5 (lambda (x) (if (flip 0.6) (+ x 5) x)))

(map randplus5 '(1 1))
Stochastic Memoization

(define randplus5 (mem (lambda (x) (if (flip 0.6) (+ x 5) x))))
(map randplus5 '(1 1))

remember first value & reuse for all later calls
Stochastic Memoization

(define randplus5 (mem (lambda (x) (if (flip 0.6) (+ x 5) x))))
(map randplus5 '(1 1))

remember first value & reuse for all later calls

ProbLog always memoizes
PRISM never memoizes
Church allows fine-grained choice
Church 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)
Church 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)
Church 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)

```
(define heads (mem (lambda () (flip 0.4)))))
```
Church 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)

(define heads (mem (lambda () (flip 0.4)))))
Church 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)

```
(define heads (mem (lambda () (flip 0.4))))
(define color1 (mem (lambda () (if (flip 0.3) 'red 'blue))))
```
Church 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)

\[
\text{(define heads (mem (lambda () (flip 0.4))))}\\
\text{(define color1 (mem (lambda () (if (flip 0.3) 'red 'blue))))}\\
\text{(define color2 (mem (lambda () (multinomial '(red green blue) '(0.2 0.3 0.5)))))}
\]
Church 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)

```
(define heads (mem (lambda () (flip 0.4))))
(define color1 (mem (lambda () (if (flip 0.3) 'red 'blue))))
(define color2 (mem (lambda ()
    (multinomial '(red green blue) '(0.2 0.3 0.5))))))
```
Church 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)

```scheme
(define heads (mem (lambda () (flip 0.4)))))
(define color1 (mem (lambda () (if (flip 0.3) 'red 'blue)))))
(define color2 (mem (lambda ()
    (multinomial '(red green blue) '(0.2 0.3 0.5))))))
(define redball (or (equal? (color1) 'red) (equal? (color2) 'red)))
```
Church 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)

```
(define heads (mem (lambda () (flip 0.4)))))
(define color1 (mem (lambda () (if (flip 0.3) 'red 'blue)))))
(define color2 (mem (lambda ()
    (multinomial '(red green blue) '(0.2 0.3 0.5))))))
(define redball (or (equal? (color1) 'red) (equal? (color2) 'red)))
(define win1 (and (heads) redball))
```
Church 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)**

```scheme
(define heads (mem (lambda () (flip 0.4))))
(define color1 (mem (lambda () (if (flip 0.3) 'red 'blue))))
(define color2 (mem (lambda ()
        (multinomial '(red green blue) '(0.2 0.3 0.5))))))
(define redball (or (equal? (color1) 'red) (equal? (color2) 'red)))
(define win1 (and (heads) redball))
```
Church 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)**

```scheme
(define heads (mem (lambda () (flip 0.4))))
(define color1 (mem (lambda () (if (flip 0.3) 'red 'blue))))
(define color2 (mem (lambda ()
  (multinomial '(red green blue) '(0.2 0.3 0.5)))))
(define redball (or (equal? (color1) 'red) (equal? (color2) 'red)))
(define win1 (and (heads) redball))
(define win2 (equal? (color1) (color2)))
```
Church 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)

```scheme
(define heads (mem (lambda () (flip 0.4))))
(define color1 (mem (lambda () (if (flip 0.3) 'red 'blue))))
(define color2 (mem (lambda ()
    (multinomial '(red green blue) '(0.2 0.3 0.5))))))
(define redball (or (equal? (color1) 'red) (equal? (color2) 'red)))
(define win1 (and (heads) redball))
(define win2 (equal? (color1) (color2)))
```
Church 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)

```
(define heads (mem (lambda () (flip 0.4))))
(define color1 (mem (lambda () (if (flip 0.3) 'red 'blue))))
(define color2 (mem (lambda ()
    (multinomial '(red green blue) '(0.2 0.3 0.5))))))
(define redball (or (equal? (color1) 'red) (equal? (color2) 'red)))
(define win1 (and (heads) redball))
(define win2 (equal? (color1) (color2)))
(define win (or win1 win2))
```
Church 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)

```
(define heads (mem (lambda () (flip 0.4)))))
(define color1 (mem (lambda () (if (flip 0.3) 'red 'blue)))))
(define color2 (mem (lambda ()
    (multinomial '(red green blue) '(0.2 0.3 0.5))))))
(define redball (or (equal? (color1) 'red) (equal? (color2) 'red)))
(define win1 (and (heads) redball))
(define win2 (equal? (color1) (color2)))
(define win (or win1 win2))
```
Sampling execution

```
(define heads (mem (lambda () (flip 0.4))))
(define color1 (mem (lambda () (if (flip 0.3) 'red 'blue))))
(define color2 (mem (lambda ()
    (multinomial '(red green blue) '(0.2 0.3 0.5)))))
(define redball (or (equal? (color1) 'red) (equal? (color2) 'red)))
(define win1 (and (heads) redball))
(define win2 (equal? (color1) (color2)))
(define win (or win1 win2))
```

```
win query
```
Marginals via enumeration

(enumeration-query
  (define heads (mem (lambda () (flip 0.4)))))

(define color1 (mem (lambda () (if (flip 0.3) 'red 'blue))))

(define color2 (mem (lambda ()
  (multinomial '(red green blue) '(0.2 0.3 0.5)))))

(define redball (or (equal? (color1) 'red) (equal? (color2) 'red)))

(define win1 (and (heads) redball))

(define win2 (equal? (color1) (color2)))

(define win (or win1 win2))

win ← query

true )

evidence
Histogram via sampling

(repeat 1000 (lambda ()
    (rejection-query
        (define heads (mem (lambda () (flip 0.4))))
        (define color1 (mem (lambda () (if (flip 0.3) 'red 'blue))))
        (define color2 (mem (lambda ()
                                (multinomial '(red green blue) '(0.2 0.3 0.5))))))
        (define redball (or (equal? (color1) 'red) (equal? (color2) 'red)))
        (define win1 (and (heads) redball))
        (define win2 (equal? (color1) (color2)))
        (define win (or win1 win2))
    (define true)))

win query evidence
Church by example:

**Rain or sun?**

```
(define weather (mem (lambda (day) (if (equal? day 0)
  weather0)
  (weatherN day (- day 1)))))

(define weather0 (lambda () (if (flip 0.5) 'sun 'rain)))
(define weatherN (lambda (today yesterday)
  (if (equal? (weather yesterday) 'rain)
    (if (flip 0.2) 'sun 'rain)
    (if (flip 0.6) 'sun 'rain)))))
```

(define weather (mem (lambda (day) (if (equal? day 0) weather0 (weatherN day (- day 1)))))

(define weather0 (lambda () (if (flip 0.5) 'sun 'rain)))
(define weatherN (lambda (today yesterday) (if (equal? (weather yesterday) 'rain) (if (flip 0.2) 'sun 'rain) (if (flip 0.6) 'sun 'rain)))))

(list (weather 0) (weather 1) (weather 2))
Probabilistic Programming Summary

• Church: functional programming + random primitives
• probabilistic generative model
• stochastic memoization
• sampling
• increasing number of probabilistic programming languages using various underlying paradigms
<table>
<thead>
<tr>
<th><strong>ProbLog</strong></th>
<th><strong>PRISM</strong></th>
<th><strong>Church</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>probabilistic facts &amp; choices</td>
<td>probabilistic choices</td>
<td>random primitives</td>
</tr>
<tr>
<td>all RVs memoized</td>
<td>no RVs memoized</td>
<td>user-defined per RV</td>
</tr>
<tr>
<td>Prolog</td>
<td>Prolog with mutually exclusive derivations</td>
<td>$\lambda$-calculus functions</td>
</tr>
<tr>
<td>distribution over worlds</td>
<td>distribution over derivations / answers</td>
<td>distribution over computations / answers</td>
</tr>
</tbody>
</table>
Roadmap

• Modeling
• Reasoning
• Language extensions
• Advanced topics

... with some detours on the way
Reasoning

• Exact inference with knowledge compilation
  • using proofs
  • using models
  • in PRISM

• Approximate inference
Answering Questions

Given:
- program
- queries
- evidence

Find:
- marginal probabilities
- conditional probabilities
- MPE state
Answering Questions

Given:
- program
- queries
- evidence

Find:
- marginal probabilities
- conditional probabilities
- MPE state

possible world

infeasible
Answering Questions

Given:
- program
- queries
- evidence

Find:
- marginal probabilities
- conditional probabilities
- MPE state
Answering Questions

**Given:**
- program
- queries
- evidence

**Find:**
- logical reasoning
- data structure
- probabilistic inference
- marginal probabilities
- conditional probabilities
- MPE state
- knowledge compilation
Answering Questions

1. using proofs
2. using models

Given:
- program
- queries
- evidence

Find:
- marginal probabilities
- conditional probabilities
- MPE state

logical reasoning

data structure

probabilistic inference

knowledge compilation
Logical Reasoning: Proofs in Prolog

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

smokes(X) :- stress(X).
smokes(X) :-
  influences(Y,X),
  smokes(Y).
stress(ann).
influences(ann,bob).
influences(bob,carl).

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

?- smokes(carl).
Logical Reasoning: Proofs in Prolog

?- stress(carl).

?- stresses(carl).

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

smokes(X) :- stress(X).
smokes(X) :-
influences(Y,X),
smokes(Y).
Logical Reasoning: Proofs in Prolog

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

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

?- stress(carl).
?- influences(Y,carl),smokes(Y).
?- smokes(carl).
Logical Reasoning: Proofs in Prolog

?- stress(carl).

?- stress(carl).

?- influences(Y,carl),smokes(Y).

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

smokes(X) :- stress(X).
smokes(X) :-
    influences(Y,X),
    smokes(Y).
Logical Reasoning: Proofs in Prolog

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

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

?- smokes(carl).
?- stress(carl).
?- influences(Y,carl),smokes(Y).
?- smokes(bob).
Logical Reasoning: Proofs in Prolog

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

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

?- stress(carl).
?- influences(Y,carl),smokes(Y).

?- smokes(bob).

?- stress(bob).

?- smokes(carl).
Logical Reasoning: Proofs in Prolog

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

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

? - smokes(carl).

? - stress(carl).

?- influences(Y,carl),smokes(Y).

?- smokes(bob).

?- stress(bob).

?- influences(Y1,bob),smokes(Y1).
Logical Reasoning:
Proofs in Prolog

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

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

?- smokes(carl).
?- stress(carl).
?- influences(Y,carl),smokes(Y).

?- smokes(bob).
?- stress(bob).
?- influences(Y1,bob),smokes(Y1).
Logical Reasoning: Proofs in Prolog

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

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

?- stress(carl).
?- influences(Y,carl),smokes(Y).
?- stress(bob).
?- influences(Y1,bob),smokes(Y1).
?- smokes(ann).
```
Logical Reasoning: Proofs in Prolog

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

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

?- smokes(carl).
?- stress(carl).
?- influences(Y,carl),smokes(Y).

?- smokes(bob).
?- stress(bob).
?- influences(Y1,bob),smokes(Y1).

?- smokes(ann).
?- stress(ann).
Logical Reasoning: Proofs in Prolog

\[ \text{stress(ann).} \]
\[ \text{influences(ann, bob).} \]
\[ \text{influences(bob, carl).} \]
\[ \text{smokes(X) :- stress(X).} \]
\[ \text{smokes(X) :- influences(Y, X), smokes(Y).} \]

?- stress(carl).
?- stresses(carl).
?- influences(Y, carl), smokes(Y).

?- smokes(bob).
?- stress(bob).
?- influences(Y1, bob), smokes(Y1).

?- smokes(ann).
?- stress(ann).
?- influences(Y2, ann), smokes(Y2).
Logical Reasoning: Proofs in Prolog

\[\text{stress}(\text{ann}).\]
\[\text{influences}(\text{ann}, \text{bob}).\]
\[\text{influences}(\text{bob}, \text{carl}).\]

\[\text{smokes}(X) :\text{- stress}(X).\]
\[\text{smokes}(X) :\text{- influences}(Y,X),\text{smokes}(Y).\]

?- \text{smokes}(\text{carl}).

?- \text{stress}(\text{carl}).

?- \text{influences}(Y,\text{carl}),\text{smokes}(Y).

?- \text{smokes}(\text{bob}).

?- \text{stress}(\text{bob}).

?- \text{influences}(Y1,\text{bob}),\text{smokes}(Y1).

?- \text{smokes}(\text{ann}).

?- \text{stress}(\text{ann}).

?- \text{influences}(Y2,\text{ann}),\text{smokes}(Y2).
Logical Reasoning: Proofs in Prolog

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

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

?- smokes(carl).
?- stress(carl).
?- influences(Y,carl),smokes(Y).

?- smokes(bob).
?- stress(bob).
?- influences(Y1,bob),smokes(Y1).

?- smokes(ann).
?- stress(ann).
?- influences(Y2,ann),smokes(Y2).
Logical Reasoning: Proofs in Prolog

proof = facts used in successful derivation:
influences(bob, carl) & influences(ann, bob) & stress(ann)
Proofs in ProbLog

\[
\text{smokes}(X) \leftarrow \text{stress}(X).
\]

\[
\text{smokes}(X) \leftarrow \text{influences}(Y,X), \text{smokes}(Y).
\]

\[
\begin{align*}
\text{probabilities:} & \\
0.8 & : \text{stress}(\text{ann}) \\
0.6 & : \text{influences}(\text{ann}, \text{bob}) \\
0.2 & : \text{influences}(\text{bob}, \text{carl})
\end{align*}
\]

\[
\text{influences}(\text{bob}, \text{carl}) \land \text{influences}(\text{ann}, \text{bob}) \land \text{stress}(\text{ann})
\]

\[
\text{probability of proof} = 0.2 \times 0.6 \times 0.8 = 0.096
\]
Proofs in ProbLog

\[
\begin{align*}
&0.8 :: \text{stress(ann)}. \\
&0.4 :: \text{stress(bob)}. \\
&0.6 :: \text{influences(ann,bob)}. \\
&0.2 :: \text{influences(bob,carl)}. \\

\text{smokes(X)} :&= \text{stress(X)}. \\
\text{smokes(X)} :&= \text{influences(Y,X)}, \text{smokes(Y)}. \\
\end{align*}
\]

\[\begin{align*}
\text{?- } \text{smokes(carl)}. \\
\text{?- } \text{stress(carl)}. \\
\text{?- } \text{influences(Y,carl)}, \text{smokes(Y)}. \\
\text{?- } \text{smokes(bob)}. \\
\text{?- } \text{stress(bob)}. \\
\text{?- } \text{influences(Y1,bob)}, \text{smokes(Y1)}. \\
\text{?- } \text{smokes(ann)}. \\
\text{?- } \text{stress(ann)}. \\
\text{?- } \text{influences(Y2,ann)}, \text{smokes(Y2)}. \\
\text{influences(bob,carl)} &\land \text{influences(ann,bob)} \\
&\land \text{stress(ann)} \\
\end{align*}\]

\[
\begin{align*}
0.2 \times 0.6 \times 0.8 &= 0.096
\end{align*}\]
Proofs in ProbLog

\[
0.8 :: \text{stress(ann)}.
\]
\[
0.4 :: \text{stress(bob)}.
\]
\[
0.6 :: \text{influences(ann, bob)}.
\]
\[
0.2 :: \text{influences(bob, carl)}.
\]

\[
\text{smokes(X)} :- \text{stress(X)}.
\]
\[
\text{smokes(X)} :-
\]
\[
\text{influences(Y, X)}, \text{smokes(Y)}.
\]

?- \text{smokes(carl)}.

?- \text{stress(carl)}.

?- \text{influences(Y, carl)}, \text{smokes(Y)}.

\[
\text{Y = bob}
\]

?- \text{smokes(bob)}.

?- \text{stress(bob)}.

?- \text{influences(Y1, bob)}, \text{smokes(Y1)}.

\[
\text{Y1 = ann}
\]

?- \text{smokes(ann)}.

?- \text{stress(ann)}.

?- \text{influences(Y2, ann)}, \text{smokes(Y2)}.

\[
\text{influences(bob, carl)}
\]
\[
& \text{influences(ann, bob)}
\]
\[
& \text{stress(ann)}
\]
\[
0.2 \times 0.6 \times 0.8
\]
\[
= 0.096
\]
Proofs in ProbLog

\[
\begin{align*}
\text{influences}(\text{bob}, \text{carl}) & \quad & \text{stress}(\text{bob}) \\
0.2 \times 0.4 & = 0.08 \\
\text{influences}(\text{ann}, \text{bob}) & \quad \text{stress}(\text{ann}) \\
0.2 \times 0.6 \times 0.8 & = 0.096
\end{align*}
\]
Proofs in ProbLog

\begin{align*}
0.8 &:: \text{stress(ann)}.
0.4 &:: \text{stress(bob)}.
0.6 &:: \text{influences(ann,bob)}.
0.2 &:: \text{influences(bob,carl)}.

\text{smokes(X)} &::= \text{stress(X)}.
\text{smokes(X)} &::= \text{influences(Y,X)}, \text{smokes(Y)}.

? - \text{smokes(carl)}.
? - \text{stress(carl)}.
? - \text{influences(Y,carl)}, \text{smokes(Y)}.

\text{Y=bob}

\text{influences(bob,carl)} \quad & \& \text{stress(bob)}

? - \text{stress(bob)}.
? - \text{influences(Y,bob)}, \text{smokes(Y)}.

\text{Y1=ann}

\text{influences(bob,carl)} \quad & \& \text{influences(ann,bob)} \quad & \& \text{stress(ann)}

0.2 \times 0.4 = 0.08

\text{proofs overlap!}
\text{cannot sum probabilities}
\text{(disjoint-sum-problem)}

0.2 \times 0.6 \times 0.8 = 0.096
Disjoint-Sum-Problem

possible worlds

\text{infl(bob, carl) & infl(ann, bob) & st(ann) & \texttt{\+st(bob)}}
\text{infl(bob, carl) & infl(ann, bob) & st(ann) & st(bob)}
\text{infl(bob, carl) & \texttt{\+infl(ann, bob)} & st(ann) & st(bob)}
\text{infl(bob, carl) & infl(ann, bob) & \texttt{\+st(ann)} & st(bob)}
\text{infl(bob, carl) & \texttt{\+infl(ann, bob)} & \texttt{\+st(ann)} & st(bob)}

\ldots
Disjoint-Sum-Problem

possible worlds

\[ \text{influences}(\text{bob},\text{carl}) \land \text{influences}(\text{ann},\text{bob}) \land \text{stress}(\text{ann}) \]

\[ \text{infl}(\text{bob},\text{carl}) \land \text{infl}(\text{ann},\text{bob}) \land \text{st}(\text{ann}) \land \lnot\text{st}(\text{bob}) \]

\[ \text{infl}(\text{bob},\text{carl}) \land \text{infl}(\text{ann},\text{bob}) \land \text{st}(\text{ann}) \land \text{st}(\text{bob}) \]

\[ \text{infl}(\text{bob},\text{carl}) \land \lnot\text{infl}(\text{ann},\text{bob}) \land \text{st}(\text{ann}) \land \text{st}(\text{bob}) \]

\[ \text{infl}(\text{bob},\text{carl}) \land \lnot\text{infl}(\text{ann},\text{bob}) \land \lnot\text{st}(\text{ann}) \land \text{st}(\text{bob}) \]

\[ \text{infl}(\text{bob},\text{carl}) \land \lnot\text{infl}(\text{ann},\text{bob}) \land \lnot\text{st}(\text{ann}) \land \text{st}(\text{bob}) \]

\[ \text{infl}(\text{bob},\text{carl}) \land \lnot\text{infl}(\text{ann},\text{bob}) \land \lnot\text{st}(\text{ann}) \land \text{st}(\text{bob}) \]

...
Disjoint-Sum-Problem

possible worlds

\[ \text{influences}(bob, carl) \land \text{influences}(ann, bob) \land \text{stress}(ann) \]

\[ \text{infl}(bob, carl) \land \text{infl}(ann, bob) \land \text{st}(ann) \land \neg \text{st}(bob) \]

\[ \text{infl}(bob, carl) \land \neg \text{infl}(ann, bob) \land \text{st}(ann) \land \text{st}(bob) \]

\[ \text{infl}(bob, carl) \land \text{infl}(ann, bob) \land \neg \text{st}(ann) \land \text{st}(bob) \]

\[ \text{infl}(bob, carl) \land \neg \text{infl}(ann, bob) \land \neg \text{st}(ann) \land \text{st}(bob) \]

...\[ \text{infl}(bob, carl) \land \neg \text{infl}(ann, bob) \land \neg \text{st}(ann) \land \text{st}(bob) \]

...\[ \text{infl}(bob, carl) \land \neg \text{infl}(ann, bob) \land \neg \text{st}(ann) \land \text{st}(bob) \]

...\[ \text{infl}(bob, carl) \land \neg \text{infl}(ann, bob) \land \neg \text{st}(ann) \land \text{st}(bob) \]
Disjoint-Sum-Problem

possible worlds

\[
influences(bob, carl) \land
influences(ann, bob) \land stress(ann)
\]

\[
infl(bob, carl) \land infl(ann, bob) \land st(ann) \land \neg st(bob)
\]

\[
infl(bob, carl) \land infl(ann, bob) \land st(ann) \land st(bob)
\]

\[
infl(bob, carl) \land \neg infl(ann, bob) \land st(ann) \land st(bob)
\]

\[
infl(bob, carl) \land \neg infl(ann, bob) \land \neg st(ann) \land st(bob)
\]

\[
infl(bob, carl) \land \neg infl(ann, bob) \land \neg st(ann) \land \neg st(bob)
\]

\[
\ldots infl(bob, carl) \land \neg infl(ann, bob) \land \neg st(ann) \land \neg st(bob)
\]

\[
\ldots influences(bob, carl) \land stress(bob)
\]

sum of proof probabilities: 0.096 + 0.08 = 0.1760
Disjoint-Sum-Problem

possible worlds

\begin{align*}
\text{influences(bob, carl)} & \land \text{influences(ann, bob)} & \land \text{stress(ann)} \\
\text{influences(bob, carl)} & \land \text{stress(bob)}
\end{align*}

\begin{align*}
\text{infl(bob, carl)} & \land \text{infl(ann, bob)} & \land \text{st(ann)} & \land \lnot \text{st(bob)} & 0.0576 \\
\text{infl(bob, carl)} & \land \text{infl(ann, bob)} & \land \text{st(ann)} & \land \text{st(bob)} & 0.0384 \\
\text{infl(bob, carl)} & \land \lnot \text{infl(ann, bob)} & \land \text{st(ann)} & \land \text{st(bob)} & 0.0256 \\
\text{infl(bob, carl)} & \land \text{infl(ann, bob)} & \land \lnot \text{st(ann)} & \land \text{st(bob)} & 0.0096 \\
\text{infl(bob, carl)} & \land \lnot \text{infl(ann, bob)} & \land \lnot \text{st(ann)} & \land \text{st(bob)} & 0.0064
\end{align*}

\[ \Sigma = 0.1376 \]

\text{sum of proof probabilities: } 0.096 + 0.08 = 0.1760
Disjoint-Sum-Problem

possible worlds

solution: knowledge compilation

\[
\begin{align*}
\text{infl}(\text{bob}, \text{carl}) & \& \text{infl}(\text{ann}, \text{bob}) & \& \text{st}(\text{ann}) & \& \text{\textbackslash}+\text{st}(\text{bob}) & 0.0576 \\
\text{infl}(\text{bob}, \text{carl}) & \& \text{infl}(\text{ann}, \text{bob}) & \& \text{st}(\text{ann}) & \& \text{st}(\text{bob}) & 0.0384 \\
\text{infl}(\text{bob}, \text{carl}) & \& \text{\textbackslash}+\text{infl}(\text{ann}, \text{bob}) & \& \text{st}(\text{ann}) & \& \text{st}(\text{bob}) & 0.0256 \\
\text{infl}(\text{bob}, \text{carl}) & \& \text{infl}(\text{ann}, \text{bob}) & \& \text{\textbackslash}+\text{st}(\text{ann}) & \& \text{st}(\text{bob}) & 0.0096 \\
\text{infl}(\text{bob}, \text{carl}) & \& \text{\textbackslash}+\text{infl}(\text{ann}, \text{bob}) & \& \text{\textbackslash}+\text{st}(\text{ann}) & \& \text{st}(\text{bob}) & 0.0064 \\
\ldots & \ & \text{influences}(\text{bob}, \text{carl}) & \& \text{stress}(\text{bob}) & \\
\text{\textbackslash} = 0.1376
\end{align*}
\]

sum of proof probabilities: 0.096 + 0.08 = 0.1760
Binary Decision Diagrams

[Bryant 86]

- compact graphical representation of Boolean formula
- popular in many branches of CS
- automatically disjoins proofs
  → efficient probability computation
Binary Decision Diagrams

$X \lor Y \lor Z$

[Bryant 86]
Binary Decision Diagrams [Bryant 86]

\[ X \lor Y \lor Z \]

<table>
<thead>
<tr>
<th></th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Y</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Z</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Binary Decision Diagrams [Bryant 86]
Binary Decision Diagrams

\[ X \lor Y \lor Z \]

X 0 0 0 0 0 1 1 1 1 1
Y 0 0 1 1 0 0 0 1 1
Z 0 1 0 1 0 1 0 1 0 1

[Bryant 86]
Binary Decision Diagrams

$X \lor Y \lor Z$

<table>
<thead>
<tr>
<th>X</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Z</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

[Bryant 86]
Binary Decision Diagrams

[Bryant 86]
Binary Decision Diagrams [Bryant 86]

$X \lor Y \lor Z$

<table>
<thead>
<tr>
<th></th>
<th>X</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Binary Decision Diagrams [Bryant 86]
**Binary Decision Diagrams** [Bryant 86]

\[ X \lor (\neg X \land Y) \lor (\neg X \land \neg Y \land Z) \]

\[ X \lor Y \lor Z \]

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
**Binary Decision Diagrams** [Bryant 86]

\[ X \lor (\neg X \land Y) \lor \neg X \land \neg Y \land Z \]

\[ X \lor Y \lor Z \]

<table>
<thead>
<tr>
<th>X</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Z</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Binary Decision Diagrams [Bryant 86]
Binary Decision Diagrams

influences(bob, carl) & stress(bob)
Binary Decision Diagrams

\[ \text{influences}(\text{bob}, \text{carl}) \land \text{influences}(\text{ann}, \text{bob}) \land \text{stress}(\text{ann}) \]

\[ \text{influences}(\text{bob}, \text{carl}) \land \text{stress}(\text{bob}) \]
Binary Decision Diagrams [Bryant 86]

\[
\text{influences}(\text{bob, carl}) \&
\text{influences}(\text{ann, bob}) \&
\text{stress}(\text{ann})
\& \text{not stress}(\text{bob})
\]

\[
\text{influences}(\text{bob, carl}) \&
\text{stress}(\text{bob})
\]
Binary Decision Diagrams

i(b, c)

s(b)

i(a, b)

s(a)

0

1
Binary Decision Diagrams

influences (bob, carl)?

no

i(b,c)

yes

s(b)

i(a,b)

s(a)

0

1
Binary Decision Diagrams

i(b,c) yes

i(a,b) no

s(b) no

s(a) yes

influences(bob, carl)?

stress(bob)?
Binary Decision Diagrams

- influences(bob, carl)?
  - yes
  - no

- stress(bob)?
  - yes

- influences(ann, bob)?
  - yes

Diagram:

- i(b, c)
  - yes
  - i(a, b)
    - no
    - s(b)
      - no
      - s(a)
      - yes
  - no

- s(a)
  - yes
Binary Decision Diagrams

- $i(b,c)$
- $i(a,b)$
- $s(a)$
- $s(b)$

Questions:
- $influences(bob, carl)$?
- $stress(bob)$?
- $influences(ann, bob)$?
- $stress(ann)$?
Binary Decision Diagrams

\[
\text{smokes}(c) = i(b, c) \& s(b) \lor i(b, c) \& i(a, b) \& s(a)
\]
Binary Decision Diagrams

\[
\text{smokes}(c) = \text{i}(b,c) \land \text{s}(b) \lor \text{i}(b,c) \land \text{i}(a,b) \land \text{s}(a)
\]

probability of \(\text{smokes}(c)\)?

influences(bob, carl)?

stress(bob)?

influences(ann, bob)?

stress(ann)?

yes

no

no

yes

no

yes

probability of \(\text{smokes}(c)\)?

\[
\text{smokes}(c) = \text{i}(b,c) \land \text{s}(b) \lor \text{i}(b,c) \land \text{i}(a,b) \land \text{s}(a)
\]
**Binary Decision Diagrams**

- influences(bob, carl)?
- stress(bob)?
- influences(ann, bob)?
- stress(ann)?

**Probability of**

- \( \text{smokes}(c) = i(b,c) \& s(b) \lor i(b,c) \& i(a,b) \& s(a) \)**
Binary Decision Diagrams

\[
\text{smokes(c)} = \text{i(b,c) \& s(b)} \lor \text{i(b,c) \& i(a,b) \& s(a)}
\]

probability of \( \text{smokes(c)} \)?

\[
0.2 \times 0.0 + 0.8 \times 1.0 = 0.8
\]

influences(ann,bob)?

\text{stress(ann)}?

\text{stress(bob)}?

\text{influences(bob, carl)}?
Binary Decision Diagrams

\[ \text{smokes}(c) = \text{i}(b,c) \& \text{s}(b) \lor \text{i}(b,c) \& \text{i}(a,b) \& \text{s}(a) \]

\[ \text{probability of } \text{smokes}(c) = 0.8 \]

\[ \text{influences}(\text{bob}, \text{carl})? \]
\[ \text{stress}(\text{bob})? \]
\[ \text{influences}(\text{ann}, \text{bob})? \]
\[ \text{stress}(\text{ann})? \]

\[ 0.4 \times 0.0 + 0.6 \times 0.8 = 0.48 \]
\[ 0.2 \times 0.0 + 0.8 \times 1.0 = 0.8 \]
Binary Decision Diagrams

\[ \text{smokes}(c) = \text{i}(b,c) \& \text{s}(b) \lor \text{i}(b,c) \& \text{i}(a,b) \& \text{s}(a) \]

0.2 \times 0.0 + 0.8 \times 1.0 = 0.8

0.4 \times 0.0 + 0.6 \times 0.8 = 0.48

0.6 \times 0.48 + 0.4 \times 1.0 = 0.688

influences(bob, carl)?

stress(bob)?

influences(ann, bob)?

stress(ann)?

probability of \text{smokes}(c)?
Binary Decision Diagrams

\[ \text{probability of } \text{smokes}(c) = 0.0 \]

\[ \text{smokes}(c) = i(b,c) \land s(b) \lor i(b,c) \land i(a,b) \land s(a) \]

- \[ i(b,c) \]
  - 0.8 \times 0.0 + 0.2 \times 0.688 = 0.1376
  - 0.2

- \[ i(a,b) \]
  - 0.4 \times 0.48 + 0.6 \times 1.0 = 0.688
  - 0.6

- \[ s(a) \]
  - 0.4 \times 0.0 + 0.6 \times 0.8 = 0.48
  - 0.2

- \[ s(b) \]
  - 0.8 \times 0.48 + 0.4 \times 1.0 = 0.8
  - 0.6

- \[ \text{influences}(bob, carl) ? \]
  - 0.8

- \[ \text{stress}(bob) ? \]
  - 0.6

- \[ \text{influences}(ann, bob) ? \]
  - 0.4

- \[ \text{stress}(ann) ? \]
  - 0.4

- \[ \text{smokes}(c) = i(b,c) \land s(b) \lor i(b,c) \land i(a,b) \land s(a) \]
  - 0.8

- \[ \text{influences}(bob, carl) ? \]
  - 0.2

- \[ \text{stress}(bob) ? \]
  - 0.0

- \[ \text{influences}(ann, bob) ? \]
  - 0.6

- \[ \text{stress}(ann) ? \]
  - 0.8

- \[ \text{smokes}(c) = i(b,c) \land s(b) \lor i(b,c) \land i(a,b) \land s(a) \]
  - 1.0
Initial Approach

(ProbLog1 & others)

Find all proofs of query

Binary Decision Diagram (BDD)

calculate marginal by dynamic programming

[De Raedt et al, IJCAI 07; Kimmig et al, TPLP 11]
Initial Approach

(ProbLog1 & others)

Find all proofs of query

Binary Decision Diagram (BDD)

calculate marginal by dynamic programming

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

[De Raedt et al, IJCAI 07; Kimmig et al, TPLP 11]
Initial Approach
(ProbLog1 & others)

Find all proofs of query

Binary Decision Diagram (BDD)

calculate marginal by dynamic programming

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

heads(1)
heads(2) & heads(3)

[De Raedt et al, IJCAI 07; Kimmig et al, TPLP 11]
Initial Approach

(ProbLog1 & others)

Find all proofs of query

Binary Decision Diagram (BDD)

calculate marginal by dynamic programming

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

heads(1)
heads(2) & heads(3)

[De Raedt et al, IJCAI 07; Kimmig et al, TPLP 11]
Initial Approach
(ProbLog1 & others)

Find all proofs of query

Binary Decision Diagram (BDD)

calculate marginal by dynamic programming

\[
\begin{align*}
0.4 &:: \text{heads(1).} \\
0.7 &:: \text{heads(2).} \\
0.5 &:: \text{heads(3).} \\
\text{win} &:: \text{heads(1).} \\
\text{win} &:: \text{heads(2),heads(3).}
\end{align*}
\]

heads(1)
heads(2) & heads(3)

[De Raedt et al, IJCAI 07; Kimmig et al, TPLP 11]
Initial Approach
(ProbLog1 & others)

Find all proofs of query

Binary Decision Diagram (BDD)

calculate marginal by dynamic programming

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

heads(1)
heads(2) & heads(3)

P(win) = probability of reaching 1-leaf

[De Raedt et al, IJCAI 07; Kimmig et al, TPLP 11]
Answering Questions

**Given:**
- program
- queries
- evidence

**Find:**
- marginal probabilities
- conditional probabilities
- MPE state

1. using proofs
2. using models

logical reasoning

data structure

probabilistic inference

**Diagram:**
- Flowchart showing the process of answering questions with given program, queries, and evidence, leading to marginal, conditional, and MPE state probabilities through logical reasoning and data structure in probabilistic inference.
Logical Reasoning: Models in Prolog

- Forward reasoning to construct unique model:

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

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

?- smokes(carl).
```
Logical Reasoning: Models in Prolog

• Forward reasoning to construct unique model:

?– smokes(carl).

• Start with database facts

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

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

stress(ann).
influences(ann,bob).
influences(bob,carl).
Logical Reasoning: Models in Prolog

• Forward reasoning to construct unique model:
  • Start with database facts
  • Use rules to add more facts

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

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

?- smokes(carl).

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

smokes(ann).
Logical Reasoning: Models in Prolog

- Forward reasoning to construct unique model:
  - Start with database facts
  - Use rules to add more facts

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

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

?- smokes(carl).
```
Logical Reasoning: Models in Prolog

- Forward reasoning to construct unique model:
  - Start with database facts
  - Use rules to add more facts

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

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

?- smokes(carl).
```

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

smokes(ann).
smokes(bob).
smokes(carl).
Logical Reasoning: Models in Prolog

- Forward reasoning to construct unique model:
  - Start with database facts
  - Use rules to add more facts
  - Query true iff in model

\[
\text{stress}(\text{ann}).
\text{influences}(\text{ann},\text{bob}).
\text{influences}(\text{bob},\text{carl}).
\]

\[
\text{smokes}(X) :- \text{stress}(X).
\text{smokes}(X) :-
\text{influences}(Y,X),
\text{smokes}(Y).
\]

?- \text{smokes}(\text{carl}).
Logical Reasoning: Models in Prolog

- Forward reasoning to construct unique model:
  - Start with database facts
  - Use rules to add more facts
  - Query true iff in model
  
- **ProbLog**: each possible world is a model, probability of query is sum over models where query is true

---

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

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

?- smokes(carl).
```
Logical Reasoning: Models in Prolog

- Forward reasoning to construct unique model:
  - Start with database facts
  - Use rules to add more facts
  - Query true iff in model
- **ProbLog**: each possible world is a model, probability of query is sum over models where query is true

\[
\text{stress}(\text{ann}). \\
\text{influences}(\text{ann}, \text{bob}). \\
\text{influences}(\text{bob}, \text{carl}). \\
\text{smokes}(X) :\neg \text{stress}(X). \\
\text{smokes}(X) :\neg \text{influences}(Y, X), \\
\text{smokes}(Y).
\]

\[-\text{smokes} (\text{carl}).\]
Weighted Model Counting

\[ WMC(\phi) = \sum_{I_{V} \models \phi} \prod_{l \in I_{V}} w(l) \]
Weighted Model Counting

A propositional formula in conjunctive normal form (CNF)

\[ WMC(\phi) = \sum_{I_V \models \phi} \prod_{l \in I_V} w(l) \]
Weighted Model Counting

A 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
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
Weighted Model Counting

A propositional formula in conjunctive normal form (CNF) given by a ProbLog program & query:

\[ WMC(\phi) = \sum_{I_V|\models \phi} \prod_{l \in I_V} w(l) \]

- Interpretations (truth value assignments) of propositional variables
- Weight of literal
Weighted Model Counting

A propositional formula in conjunctive normal form (CNF) given by a ProbLog program & query has the following expression:

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

- Weighted Model Counting \((WMC)\)
- \(\phi\): propositional formula
- \(w(l)\): weight of literal
- \(I_V\): interpretations (truth value assignments) of propositional variables
- \(I_V \models \phi\): possible worlds

Given a ProbLog program and a query, the weighted model counting counts the number of possible worlds that satisfy the formula, with each literal having a weight associated with it.
Weighted Model Counting

A propositional formula in conjunctive normal form (CNF) given by a ProbLog program & 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 $p::f$, $w(f) = p$ and $w(\neg f) = 1 - p$
Weighted Model Counting

Propositional formula in conjunctive normal form (CNF) given by ProbLog program & query

\[ P(Q) = \sum_{F \cup R \models Q} \prod_{f \in F} p(f) \prod_{f \notin F} 1 - p(f) \]

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

Interpretations (truth value assignments) of propositional variables possible worlds

Weight of literal for \( p::f \),
\( w(f) = p \)
\( w(\text{not } f) = 1 - p \)
ProbLog → CNF

?- smokes(carl).

smoothes(X) :- stress(X).
smokes(X) :-
    influences(Y,X),
    smokes(Y).
ProbLog $\rightarrow$ CNF

?- smokes(carl).

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

Find relevant ground rules by backward reasoning
Problog $\rightarrow$ CNF

?- smokes(carl).

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

• Find relevant ground rules by backward reasoning

  smokes(carl) :- influences(bob,carl),smokes(bob).
  smokes(bob) :- stress(bob).
  smokes(bob) :- influences(ann,bob),smokes(ann).
  smokes(ann) :- stress(ann).
ProbLog $\rightarrow$ CNF

?- smokes(carl).

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

- Find relevant ground rules by backward reasoning

smokes(carl) :- influences(bob,carl),smokes(bob).
smokes(bob) :- stress(bob).
smokes(bob) :- influences(ann,bob),smokes(ann).
smokes(ann) :- stress(ann).

- Convert to propositional logic formula
ProbLog → CNF

?- smokes(carl).

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

- Find relevant ground rules by backward reasoning
  - smokes(carl) :- influences(bob,carl),smokes(bob).
  - smokes(bob) :- stress(bob).
  - smokes(bob) :- influences(ann,bob),smokes(ann).
  - smokes(ann) :- stress(ann).

- Convert to propositional logic formula

  may require loop-breaking

  \( sm(c) \leftrightarrow (i(b,c) \land sm(b)) \)
  \( \land sm(b) \leftrightarrow (st(b) \lor (i(a,b) \land sm(a))) \)
  \( \land sm(a) \leftrightarrow st(a) \)
ProbLog $\rightarrow$ CNF

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

?- smokes(carl).

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

• Find relevant ground rules by backward reasoning
  
  \[
  \begin{align*}
  \text{smokes(carl)} & :\text{- influences(bob,carl),smokes(bob)}. \\
  \text{smokes(bob)} & :\text{- stress(bob)}. \\
  \text{smokes(bob)} & :\text{- influences(ann,bob),smokes(ann)}. \\
  \text{smokes(ann)} & :\text{- stress(ann)}. 
  \end{align*}
  \]

• Convert to propositional logic formula

\[
\begin{align*}
\text{sm}(c) & \iff (i(b,c) \land \text{sm}(b)) \\
\land \text{sm}(b) & \iff (\text{st}(b) \lor (i(a,b) \land \text{sm}(a))) \\
\land \text{sm}(a) & \iff \text{st}(a)
\end{align*}
\]

• Rewrite in CNF (as usual)
Current Approach
(ProbLog2)

Find relevant ground program for queries & evidence

Weighted CNF

use weighted model counting / satisfiability

[Fierens et al, TPLP 14]
Current Approach
(ProbLog2)

- Find relevant ground program for queries & evidence
- Weighted CNF
- Use weighted model counting / satisfiability

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

[Fierens et al, TPLP 14]
Current Approach  
(ProbLog2)

Find relevant ground program for queries & evidence

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

use weighted model counting / satisfiability

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

[Fierens et al, TPLP 14]
Current Approach
(ProbLog2)

Find relevant ground program for queries & evidence

Weighted CNF

use weighted model counting / satisfiability

\[
\begin{align*}
0.4 &:: heads(1). \\
0.7 &:: heads(2). \\
0.5 &:: heads(3). \\
\text{win} &:: heads(1) . \\
\text{win} &:: heads(2), heads(3). \\
\text{win} &:: heads(1) . \\
\text{win} &:: heads(2), heads(3). \\
\hline
\text{win} &:: h(1) \lor (h(2) \land h(3))
\end{align*}
\]
Current Approach
(ProbLog2)

Find relevant ground program for queries & evidence

use weighted model counting / satisfiability

Weighted CNF

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

win :- heads(1).
win :- heads(2), heads(3).

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

(\neg \text{win} \lor \text{h(1)} \lor \text{h(2)})
\land (\neg \text{win} \lor \text{h(1)} \lor \text{h(3)})
\land (\text{win} \lor \neg \text{h(1)})
\land (\text{win} \lor \neg \text{h(2)} \lor \neg \text{h(3)})
Current Approach
(ProbLog2)

Find relevant ground program for queries & evidence

Weighted CNF

use weighted model counting / satisfiability

0.4::heads(1).
0.7::heads(2).
0.5::heads(3).

win :- heads(1).
win :- heads(2), heads(3).

\[ \text{win} \leftarrow \text{h(1)} \lor (\text{h(2)} \land \text{h(3)}) \]

\[ \lor (\neg \text{win} \lor \text{h(1)} \lor \text{h(2)}) \]

\[ \land (\neg \text{win} \lor \text{h(1)} \lor \text{h(3)}) \]

\[ \land (\text{win} \lor \neg \text{h(1)}) \]

\[ \land (\text{win} \lor \neg \text{h(2)} \lor \neg \text{h(3)}) \]

h(1) \rightarrow 0.4 \quad h(2) \rightarrow 0.7 \quad h(3) \rightarrow 0.5
\[ \neg h(1) \rightarrow 0.6 \quad \neg h(2) \rightarrow 0.3 \quad \neg h(3) \rightarrow 0.5 \]

[Fierens et al, TPLP 14]
Current Approach

(ProbLog2)

Find relevant ground program for queries & evidence

Weighted CNF

use weighted model counting / satisfiability

0.4::heads(1).
0.7::heads(2).
0.5::heads(3).

win :- heads(1).
win :- heads(2), heads(3).

win :- heads(1).
win :- heads(2), heads(3).

win ← h(1) ∨ (h(2) ∧ h(3))

(¬win ∨ h(1) ∨ h(2))
∧ (¬win ∨ h(1) ∨ h(3))
∧ (win ∨ ¬h(1))
∧ (win ∨ ¬h(2) ∨ ¬h(3))

h(1) → 0.4  h(2) → 0.7  h(3) → 0.5
¬h(1) → 0.6  ¬h(2) → 0.3  ¬h(3) → 0.5

use standard tool

[Fierens et al, TPLP 14]
WMC using d-DNNFs

1. represent formula as d-DNNF
2. transform into arithmetic circuit
3. evaluate bottom-up

\[
\text{alarm} \leftrightarrow \text{burglary} \lor \text{earthquake} \\
\text{calls}(\text{john}) \leftrightarrow \text{alarm}, \text{hears\_alarm}(\text{john}) \\
\text{calls}(\text{john})
\]
WMC using d-DNNFs

1. represent formula as d-DNNF
2. transform into arithmetic circuit
3. evaluate bottom-up

3. evaluate bottom-up

[Figure: Fierens et al, TPLP 14]
ProbLog Inference

• reduction to propositional formula

• addresses disjoint-sum-problem

• **but**: not all probabilistic logic programs face this problem! e.g., weather

• more generally: mutually exclusive proofs as assumed in PRISM
**Fig. 1.** Example of a left-to-right HMM with four states

```
target(hmm/1).
values(tr(s0),[s0,s1]).
values(tr(s1),[s1,s2]).
values(tr(s2),[s2,s3]).
values(out(_),[a,b]).

hmm(Cs):- hmm(0,s0,Cs).

hmm(T,s3,[C]):- msw(out(s3), C).  % If at the final state:
                          % output a symbol and then terminate.

hmm(T,S,[C|Cs]):- S\==s3,     % If not at the final state:
                    msw(out(S), C),       % choose a symbol to be output,
                    msw(tr(S), Next),    % choose the next state,
                    T1 is T+1,           % Put the clock ahead,
                    hmm(T1,Next,Cs).    % and enter the next loop.
```

**Fig. 2.** PRISM program for the left-to-right HMM

[Figures: Sato and Kameya 08]
Fig. 1. Example of a left-to-right HMM with four states

```
target(hmm/1).
values(tr(s0), [s0,s1]).
values(tr(s1), [s1,s2]).
values(tr(s2), [s2,s3]).
values(out(_), [a,b]).
```

```
hmm(Cs):- hmm(0,s0,Cs).

hmm(T,s3,[C]):- msw(out(s3), C). % If at the final state:
              % output a symbol and then terminate.

hmm(T,S,[C|Cs]):- S\=<s3,
              % If not at the final state:
              % choose a symbol to be output,
              % choose the next state,
              % Put the clock ahead,
              % and enter the next loop.
    msw(out(S), C),
    msw(tr(S), Next),
    T1 is T+1,
    hmm(T1,Next,Cs).
```

Fig. 2. PRISM program for the left-to-right HMM

[Figures: Sato and Kameya 08]
**Fig. 1.** Example of a left-to-right HMM with four states

target(hmm/1).
values(tr(s0),[s0,s1]).
values(tr(s1),[s1,s2]).
values(tr(s2),[s2,s3]).
values(out(_),[a,b]).

\[ \text{hmm(Cs):= hmm(0,s0,Cs).} \]

\[ \text{hmm(T,s3,[C]):= msw(out(s3), C).} \quad \text{\% If at the final state:} \]
\[ \quad \text{\% output a symbol and then terminate.} \]
\[ \text{hmm(T,S,[C|Cs]):= S\=\=s3,} \]
\[ \quad \text{msw(out(S), C),} \]
\[ \quad \text{msw(tr(S), Next),} \]
\[ \quad \text{\% If not at the final state:} \]
\[ \quad \text{\% choose a symbol to be output,} \]
\[ \quad \text{\% choose the next state,} \]
\[ \quad \text{\% Put the clock ahead,} \]
\[ \quad \text{\% and enter the next loop.} \]

**Fig. 2.** PRISM program for the left-to-right HMM

[Figures: Sato and Kameya 08]
no memoization: two different RVS

\[
E_1 = m(out(s0), a) \land m(tr(s0), s0) \land m(out(s0), b) \land m(tr(s0), s0) \land m(out(s0), b) \\
\land m(tr(s0), s1) \land m(out(s1), b) \land m(tr(s1), s2) \land m(out(s2), b) \land m(tr(s2), s3) \\
\land m(out(s3), a)
\]

\[
E_2 = m(out(s0), a) \land m(tr(s0), s0) \land m(out(s0), b) \land m(tr(s0), s1) \land m(out(s1), b) \\
\land m(tr(s1), s1) \land m(out(s1), b) \land m(tr(s1), s2) \land m(out(s2), b) \land m(tr(s2), s3) \\
\land m(out(s3), a)
\]

\[
E_6 = m(out(s0), a) \land m(tr(s0), s1) \land m(out(s1), b) \land m(tr(s1), s2) \land m(out(s2), b) \\
\land m(tr(s2), s2) \land m(out(s2), b) \land m(tr(s2), s2) \land m(out(s2), b) \land m(tr(s2), s3) \\
\land m(out(s3), a)
\]

Fig. 3. Six explanations for $\text{hmm}([a, b, b, b, b, a])$. Due to the space limit, the predicate name msw is abbreviated to m.
PRISM inference

\[
\begin{align*}
\text{hmm}([a, b, b, b, b, a]) & \iff \text{hmm}(0, s0, [a, b, b, b, b, a]) \\
\text{hmm}(0, s0, [a, b, b, b, b, a]) & \iff m(\text{out}(s0), a) \land m(\text{tr}(s0), s0) \land \text{hmm}(1, s0, [b, b, b, b, a]) \\
& \lor m(\text{out}(s0), a) \land m(\text{tr}(s0), s1) \land \text{hmm}(1, s1, [b, b, b, b, a]) \\
\text{hmm}(1, s0, [b, b, b, b, a]) & \iff m(\text{out}(s0), b) \land m(\text{tr}(s0), s0) \land \text{hmm}(2, s0, [b, b, b, a]) \\
& \lor m(\text{out}(s0), b) \land m(\text{tr}(s0), s1) \land \text{hmm}(2, s1, [b, b, b, a]) \\
\text{hmm}(2, s0, [b, b, b, a]) & \iff m(\text{out}(s0), b) \land m(\text{tr}(s0), s1) \land \text{hmm}(3, s1, [b, b, a]) \\
\text{hmm}(1, s1, [b, b, b, a]) & \iff m(\text{out}(s1), b) \land m(\text{tr}(s1), s1) \land \text{hmm}(2, s1, [b, b, b, a]) \\
& \lor m(\text{out}(s1), b) \land m(\text{tr}(s1), s2) \land \text{hmm}(2, s2, [b, b, b, a]) \\
\text{hmm}(2, s1, [b, b, b, a]) & \iff m(\text{out}(s1), b) \land m(\text{tr}(s1), s1) \land \text{hmm}(3, s1, [b, b, a]) \\
& \lor m(\text{out}(s1), b) \land m(\text{tr}(s1), s2) \land \text{hmm}(3, s2, [b, b, a]) \\
\text{hmm}(3, s1, [b, b, a]) & \iff m(\text{out}(s1), b) \land m(\text{tr}(s1), s2) \land \text{hmm}(4, s2, [b, a]) \\
\text{hmm}(2, s2, [b, b, b, a]) & \iff m(\text{out}(s2), b) \land m(\text{tr}(s2), s2) \land \text{hmm}(3, s2, [b, b, a]) \\
\text{hmm}(3, s2, [b, b, a]) & \iff m(\text{out}(s2), b) \land m(\text{tr}(s2), s2) \land \text{hmm}(4, s2, [b, a]) \\
\text{hmm}(4, s2, [b, a]) & \iff m(\text{out}(s2), b) \land m(\text{tr}(s2), s3) \land \text{hmm}(5, s3, [a]) \\
\text{hmm}(5, s3, [a]) & \iff m(\text{out}(s3), a)
\end{align*}
\]

Fig. 4. Factorized explanations for hmm([a, b, b, b, b, a])
Fig. 4. Factorized explanations for $\text{hmm}([a, b, b, b, a])$
PRISM: compute probability by dynamic programming

[Figures: Sato and Kameya 08]
PRISM: compute probability by dynamic programming

Fig. 5. Explanation graph

[Figures: Sato and Kameya 08]
Query Evaluation in PDB
Query Evaluation in PDB

• **Extensional** evaluation
  • guided by query expression only
  • exploit DB technology
  • for queries known to have polytime evaluation
Query Evaluation in PDB

• **Extensional** evaluation
  • guided by query expression only
  • exploit DB technology
  • for queries known to have polytime evaluation

• **Intensional** evaluation
  • construct **lineage** (= propositional formula)
  • compute probability of lineage
  • all queries
Approximate Inference

• Lower and upper bounds

\[ \phi_L \models \phi \models \phi_U \]

\[ P(\phi_L) \leq P(\phi) \leq P(\phi_U) \]

• Sampling
Sampling

- $P(\text{query}) ?$

$$P(\text{query}) \approx \frac{\# \text{ query holds}}{\# \text{ worlds sampled}}$$
Rejection Sampling

• $P(\text{query} | \text{evidence})$?
Rejection Sampling

- \( P(\text{query} \mid \text{evidence}) \) ?
Rejection Sampling

- \( P(\text{query} \mid \text{evidence}) \) ?

  evidence holds
  
  evidence does not hold
Rejection Sampling

\[ P(\text{query} \mid \text{evidence}) \approx \frac{\# \text{ query \& evidence holds}}{\# \text{ evidence holds}} \]
Markov Chain Monte Carlo (MCMC)

- Generate next sample by modifying current one
- Most common inference approach for PP languages such as Church, BLOG, ...
- Also considered for PRISM and ProbLog

**Key challenges:**
- how to propose next sample
- how to handle evidence
Roadmap

• Modeling
• Reasoning
• Language extensions
• Advanced topics

... with some detours on the way
Extensions of basic PLP

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

decisions

continuous RVs

programming constructs

constraints

time & dynamics

semiring labels

...
Dynamics: Evolving Networks

- *Travian*: A massively multiplayer real-time strategy game
  - Commercial game run by TravianGames GmbH
  - ~3,000,000 players spread over different “worlds”
  - ~25,000 players in one world

[Thon et al. ECML 08]
World Dynamics

Fragment of world with

~10 alliances
~200 players
~600 cities

alliances color-coded

Can we build a model of this world?
Can we use it for playing better?

[Thon, Landwehr, De Raedt, ECML08]
World Dynamics

Fragment of world with

~10 alliances
~200 players
~600 cities

alliances color-coded

Can we build a model of this world?
Can we use it for playing better?

[Thon, Landwehr, De Raedt, ECML08]
World Dynamics

Fragment of world with

~10 alliances
~200 players
~600 cities

alliances color-coded

Can we build a model of this world?
Can we use it for playing better?

[Thon, Landwehr, De Raedt, ECML08]
World Dynamics

Fragment of world with

~10 alliances
~200 players
~600 cities

alliances color-coded

Can we build a model
of this world?
Can we use it for playing
better?

[Thon, Landwehr, De Raedt, ECML08]
World Dynamics

Fragment of world with

~10 alliances
~200 players
~600 cities

alliances color-coded

Can we build a model of this world?
Can we use it for playing better?

[Thon, Landwehr, De Raedt, ECML08]
World Dynamics

Fragment of world with

- ~10 alliances
- ~200 players
- ~600 cities

alliances color-coded

Can we build a model of this world?
Can we use it for playing better?

[Thon, Landwehr, De Raedt, ECML08]
Causal Probabilistic Time-Logic (CPT-L)

how does the world change over time?

[Thon et al, MLJ 11]
Causal Probabilistic Time-Logic (CPT-L)

how does the world change over time?

0.4::conquest(Attacker,C); 0.6::nil <-

\[
\text{city}(C,\text{Owner}), \text{city}(C2,\text{Attacker}), \text{close}(C,C2).
\]

if \textbf{cause} holds at time $T$

[Thon et al, MLJ 11]
Causal Probabilistic Time-Logic (CPT-L)

how does the world change over time?

one of the effects holds at time $T+1$

\[
0.4::\text{conquest(Attacker,C)}; 0.6::\text{nil} \leftarrow \\
\text{city(C,Owner)}, \text{city(C2,Attacker)}, \text{close(C,C2)}.
\]

if cause holds at time $T$

[Thon et al, MLJ 11]
Causal Probabilistic Time-Logic (CPT-L)

how does the world change over time?

one of the effects holds at time $T+1$

$$0.4::\text{conquest}(\text{Attacker},C); \ 0.6::\text{nil} \leftarrow$$

$$\text{city}(C,\text{Owner}), \text{city}(C2,\text{Attacker}), \text{close}(C,C2).$$

if cause holds at time $T$

[Thon et al, MLJ 11]
continuous RVs
Distributional Clauses (DC)

- Discrete- and continuous-valued random variables
- Inference: particle filter

[Gutmann et al, TPLP 11; Nitti et al, IROS 13]
Distributional Clauses (DC)

- Discrete- and continuous-valued random variables
- Inference: particle filter

**random variable** with Gaussian distribution

\[
\text{length}(\text{Obj}) \sim \text{gaussian}(6.0, 0.45) \quad \text{:- type(Obj, glass)}.
\]
Distributional Clauses (DC)

- Discrete- and continuous-valued random variables
- Inference: particle filter

\[
\text{length}(\text{Obj}) \sim \text{gaussian}(6.0, 0.45) :- \text{type}(\text{Obj}, \text{glass}).
\]

\[
\text{stackable}(\text{OBot}, \text{OTop}) :-
\]

\[
\approx \text{length}(\text{OBot}) \geq \approx \text{length}(\text{OTop}),
\]

\[
\approx \text{width}(\text{OBot}) \geq \approx \text{width}(\text{OTop}).
\]

[Comparing values of random variables]

[Gutmann et al, TPLP 11; Nitti et al, IROS 13]
Distributional Clauses (DC)

- Discrete- and continuous-valued random variables
- Inference: particle filter

\[
\text{length} \text{(Obj)} \sim \text{gaussian}(6.0, 0.45) \quad \text{:-} \quad \text{type} \text{(Obj,glass)}.
\]

\[
\text{stackable} \text{(OBot,OTop)} \quad \text{:-}
\]

\[
\approx \text{length} \text{(OBot)} \geq \approx \text{length} \text{(OTop)},
\]

\[
\approx \text{width} \text{(OBot)} \geq \approx \text{width} \text{(OTop)}.
\]

\[
\text{ontype} \text{(Obj,plate)} \sim \text{finite}([0 : \text{glass}, 0.0024 : \text{cup},
0 : \text{pitcher}, 0.8676 : \text{plate},
0.0284 : \text{bowl}, 0 : \text{-serving},
0.1016 : \text{none}])
\]

\[
\text{:-} \quad \text{obj} \text{(Obj)}, \text{on} \text{(Obj,02)}, \text{type} \text{(02,plate)}.
\]

random variable with discrete distribution
Distributional Clauses (DC)

- Discrete- and continuous-valued random variables
- Inference: particle filter

\[
\begin{align*}
\text{length}(\text{Obj}) & \sim \text{gaussian}(6.0, 0.45) \quad :\quad \text{type}(\text{Obj}, \text{glass}). \\
\text{stackable}(\text{OBot}, \text{OTop}) & \quad :\quad \\
& \approx \text{length}(\text{OBot}) \geq \approx \text{length}(\text{OTop}), \\
& \approx \text{width}(\text{OBot}) \geq \approx \text{width}(\text{OTop}). \\
\text{ontype}(\text{Obj}, \text{plate}) & \sim \text{finite}([0 : \text{glass}, 0.0024 : \text{cup}, \\
& 0 : \text{pitcher}, 0.8676 : \text{plate}, \\
& 0.0284 : \text{bowl}, 0 : \text{serving}, \\
& 0.1016 : \text{none}]) \\
& \quad :\quad \text{obj}(\text{Obj}), \ \text{on}(\text{Obj}, \text{O}2), \ \text{type}(\text{O}2, \text{plate}).
\end{align*}
\]
Occluded Object Search

- DC model of objects and their spatial arrangement
- different types of objects suitable for different tasks
- shelves with objects of different shape and size
- given a task, find an object to perform that task

[Moldovan et al, ICRA 12, 14]
Relational State Estimation over Time

Magnetism scenario

- object tracking
- category estimation from interactions

Box scenario

- object tracking even when invisible
- estimate spatial relations

[Ref: Nitti et al, IROS 13]
Dynamic Distributional Clauses

3 object types: magnetic, ferromagnetic, nonmagnetic

\[
\text{type}(X) \sim \text{finite}([1/3:\text{magnet, 1/3:ferromagnetic, 1/3:nonmagnetic}]) \leftarrow \text{object}(X).
\]

2 magnets attract or repulse

\[
\text{interaction}(A,B) \sim \text{finite}([0.5:attraction,0.5:repulsion])
\]

\[\leftarrow A \neq B, \text{type}(A) = \text{magnet}, \text{type}(B) = \text{magnet}.\]

Next position after attraction

\[
\text{pos}(A)_{t+1} \sim \text{gaussian}(\text{middlepoint}(A,B)_t, \text{Cov}) \leftarrow \\
\text{near}(A,B)_t, \text{not(held(A))}, \text{not(held(B))}, \text{interaction}(A,B)_t = \text{attr}, c/d\text{dist}(A,B)^2 > \text{friction}(A).
\]

\[
\text{pos}(A)_{t+1} \sim \text{gaussian}(\text{pos}(A)_t, \text{Cov}) \leftarrow \text{not( attraction}(A,B) ).
\]

\[Nitti\ et\ al,\ IROS\ 13,\ ICRA\ 14\]
Magnet scenario

- 3 object types: magnetic, ferromagnetic, nonmagnetic
  - nonmagnetic objects do not interact
  - a magnet and a ferromagnetic object attract each other with a force that depends on the distance
  - if an object is held magnetic force is compensated.

Two complications: 1) observations uncertain
  2) no discrete states
Inference and Learning

<table>
<thead>
<tr>
<th>Pos(1)=(0, 3)</th>
<th>Pos(1)=(0, 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pos(2)=(0, 1)</td>
<td>Pos(2)=(0, 1)</td>
</tr>
<tr>
<td>right(X,Y)</td>
<td>near(1,2)=true</td>
</tr>
<tr>
<td>near(X,Y)</td>
<td>type(1)=nonmagnetic</td>
</tr>
<tr>
<td>interaction(X,Y)</td>
<td></td>
</tr>
<tr>
<td>type(X) ~ [1/3:magnet,...]</td>
<td></td>
</tr>
<tr>
<td>[...]</td>
<td>[...]</td>
</tr>
</tbody>
</table>

Classical particle filter

Distributional Clauses Particle Filter (DCPF)

127
Speed 0x

Queries
(updated every 5 steps)

on(X,Y):
[1.0:(3,(table)),1.0:(4,(table))]
inside(X,Y):
[]
tr_inside(X,Y):
[]

Box ID=4
Cube ID=3
cProbLog: constraints on possible worlds

weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).

P::pack(Item) :-
    weight(Item,Weight),
    P is 1.0/Weight.

excess(Limit) :- ...

not excess(10).
pack(helmet) v pack(boots).
weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).

P::pack(Item) :-
    weight(Item,Weight),
    P is 1.0/Weight.

excess(Limit) :- ...

not excess(10).
pack(helmet) v pack(boots).
cProbLog: constraints on possible worlds

weight(skis, 6).
weight(boots, 4).
weight(helmet, 3).
weight(gloves, 2).

P::pack(Item) :-
    weight(Item, Weight),
    P is 1.0/Weight.

excess(Limit) :- ...

not excess(10).
pack(helmet) v pack(boots).

**constraints** as first-order logic formulas

[Fierens et al, PP 12; Shterionov et al]
cProbLog: constraints on possible worlds

weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).

P::pack(Item) :-
    weight(Item,Weight),
    P is 1.0/Weight.

excess(Limit) :- ...

not excess(10).
pack(helmet) ∨ pack(boots).

constraints as first-order logic formulas

[Fierens et al, PP 12; Shterionov et al]
weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).

P::pack(Item) :-
    weight(Item,Weight),
    P is 1.0/Weight.

not excess(10).
pack(helmet) v pack(boots).

cProbLog: constraints
on possible worlds

constraints as first-order logic formulas
weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).

P::pack(Item) :-
  weight(Item,Weight),
  P is 1.0/Weight.

excess(Limit) :- ...

not excess(10).
pack(helmet) \lor pack(boots).

**constraints as first-order logic formulas**

[Fierens et al, PP 12; Shterionov et al]
cProbLog: constraints on possible worlds

weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).

P::pack(Item) :-
    weight(Item,Weight),
    P is 1.0/Weight.

excess(Limit) :- ...

not excess(10).
pack(helmet) v pack(boots).

constraints as first-order logic formulas

[Fierens et al, PP 12; Shterionov et al]
cProbLog: constraints on possible worlds

weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).

P::pack(Item) :-
    weight(Item,Weight),
    P is 1.0/Weight.

excess(Limit) :- ...

not excess(10).
pack(helmet) v pack(boots).

constraints as first-order logic formulas
cProbLog: constraints on possible worlds

weight(skis, 6).
weight(boots, 4).
weight(helmet, 3).
weight(gloves, 2).

P::pack(Item) :-
    weight(Item, Weight),
    P is 1.0/Weight.

excess(Limit) :- ...

not excess(10).
pack(helmet) v pack(boots).

constraints as first-order logic formulas

[Fierens et al, PP 12; Shterionov et al]
cProbLog: constraints on possible worlds

weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).

P::pack(Item) :-
    weight(Item,Weight),
    P is 1.0/Weight.

excess(Limit) :- ...

not excess(10).
pack(helmet) v pack(boots).

constraints as first-order logic formulas
weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).

P::pack(Item) :-
    weight(Item,Weight),
    P is 1.0/Weight.

excess(Limit) :- ...

not excess(10).

pack(helmet) v pack(boots).

constraints as first-order logic formulas
cProbLog: constraints on possible worlds

weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).

P::pack(Item) :-
    weight(Item,Weight),
    P is 1.0/Weight.

excess(Limit) :- ...

not excess(10).
pack(helmet) v pack(boots).

constraints as first-order logic formulas

normalized distribution over restricted set of possible worlds

[Fierens et al, PP 12; Shterionov et al]

[130]
Which advertising strategy maximizes expected profit?

[Van den Broeck et al, AAAI 10]
Viral Marketing

decide truth values of some atoms

Which strategy gives the maximum expected utility?
person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
DTProbLog

markdown(P) :- person(P).

definition fact: true or false?

person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
DTProbLog

? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).

probabilistic facts + logical rules

person(1).
person(2).
person(3).
person(4).
friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
DTProbLog

\[ \text{marketed}(P) :\neg \text{person}(P). \]
\[ 0.3 : \text{buy\_trust}(X,Y) :\neg \text{friend}(X,Y). \]
\[ 0.2 : \text{buy\_marketing}(P) :\neg \text{person}(P). \]

\[ \text{buys}(X) :\neg \text{friend}(X,Y), \text{buys}(Y), \text{buy\_trust}(X,Y). \]
\[ \text{buys}(X) :\neg \text{marketed}(X), \text{buy\_marketing}(X). \]

\[ \text{buys}(P) \Rightarrow 5 :\neg \text{person}(P). \]
\[ \text{marketed}(P) \Rightarrow -3 :\neg \text{person}(P). \]

**utility facts:** cost/reward if true

Utility facts:

- person(1).
- person(2).
- person(3).
- person(4).
- friend(1,2).
- friend(2,1).
- friend(2,4).
- friend(3,4).
- friend(4,2).
DTProbLog

? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).

buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).

person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
DTProbLog

? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).

buys(X) :- marketed(X), buy_marketing(X).

buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).
DTProbLog

? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).

buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).

marketed(1) marketed(3)
DTProbLog

? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).

buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).

marketed(1)       marketed(3)
bt(2,1)   bt(2,4)        bm(1)

person(1).
person(2).
person(3).
person(4).
friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
DTProbLog

? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).

buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).
DTProbLog

? :: marketed(P) :- person(P).
0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).

buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).

utility = −3 + −3 + 5 + 5 = 4
probability = 0.0032
DTProbLog

? :: marketed(P) :- person(P).
0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).

buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).

utility = -3 + -3 + 5 + 5 = 4
probability = 0.0032

world contributes 0.0032×4 to expected utility of strategy
DTProbLog

? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).

buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).

**task:** find strategy that maximizes expected utility

**solution:** using ProbLog technology
Phenetic

- **Causes:** Mutations
  - All related to similar phenotype
- **Effects:** Differentially expressed genes
  - 27,000 cause effect pairs
- **Interaction network:**
  - 3,063 nodes
  - Genes
  - Proteins
  - 16,794 edges
  - Molecular interactions
  - Uncertain
- **Goal:** connect causes to effects through common subnetwork
  - = Find mechanism
- **Techniques:**
  - DTProbLog
  - Approximate inference

[De Maeyer et al., Molecular Biosystems 13]
Dyna

[ Eisner et al 05 ]

• Weighted Logic Programs (but not Prolog)
• Inspired by NLP
• Arbitrary semiring weights
• Forward reasoning
CYK parsing in Dyna

input sentence

probabilistic grammar rules

rules $X \rightarrow W$ with $W$ spanning $I, J$

constit($X, I, J$) += rewrite($X, W$) * word($W, I, J$).
goal += constit($s, 0, N$) * end($N$).

sum over all parses

rules $X \rightarrow YZ$, split points $J$
reachable(Q) :- initial(Q).
reachable(Q) :- reachable(P), edge(P, Q).

Fig. 1. A simple bottom-up logic program for graph reachability

initial(a) = T  edge(c, d) = T
edge(a, c) = T  edge(d, b) = T
edge(a, d) = T  edge(d, c) = T
edge(b, b) = T  edge(d, d) = T
edge(c, a) = T

Fig. 2. A directed graph and the corresponding initial database
reachable(Q) ⊕= initial(Q).
reachable(Q) ⊕= reachable(P) × edge(P, Q).

![Diagram](image)

- initial(a) = 0, edge(c, d) = 15
- edge(a, c) = 4, edge(d, b) = 6
- edge(a, d) = 20, edge(d, c) = 16
- edge(b, b) = 8, edge(d, d) = 2
- edge(c, a) = 9

**Fig. 3.** A cost graph and the corresponding initial database

[Figures: Cohen et al, ICLP 08]
reachable(Q) ⊕= initial(Q).

reachable(Q) ⊕= reachable(P) ⊗ edge(P, Q).

Fig. 3. A cost graph and the corresponding initial database

Figures: Cohen et al, ICLP 08
reachable(Q) ⊕= initial(Q).
reachable(Q) ⊕= reachable(P) ⊗ edge(P, Q).

⊕ → min
⊗ → +

shortest path

Fig. 3. A cost graph and the corresponding initial database

Fig. 4. A probabilistic graph and the corresponding initial database. With stopping probabilities made explicit, this would encode a Markov model.

[Figures: Cohen et al, ICLP 08]
reachable(Q) ⊕= initial(Q).

reachable(Q) ⊕= reachable(P) ⊗ edge(P, Q).

\( \oplus \rightarrow \text{min} \)

\( \otimes \rightarrow + \)

shortest path

\( \oplus \rightarrow \text{max} \)

\( \otimes \rightarrow \cdot \)

most likely path

Fig. 3. A cost graph and the corresponding initial database

Fig. 4. A probabilistic graph and the corresponding initial database. With stopping probabilities made explicit, this would encode a Markov model.

[Figures: Cohen et al, ICLP 08]
reachable(Q) ⊕= initial(Q).

reachable(Q) ⊕= reachable(P) ⊗ edge(P, Q).

⊕ → min

⊗ → +

shortest path

Fig. 3. A cost graph and the corresponding initial database

⊕ → max

⊗ → ·

most likely path

Fig. 4. A probabilistic graph and the corresponding initial database. With stopping probabilities made explicit, this would encode a Markov model.

PRISM

[Figures: Cohen et al, ICLP 08]
Semantics

• Semiring *

• valuation function maps provable ground terms to values

• extended to expressions, e.g. \([x * y] \overset{\text{def}}{=} [x] * [y]\)

• weighted rules \(r \oplus_r = E\) constrain evaluation

\[
[r] = [E_1] \oplus_r [E_2] \oplus_r \ldots
\]

* An algebraic semiring consists of five elements \(\langle K, \oplus, \otimes, 0, 1 \rangle\), where \(K\) is a domain closed under \(\oplus\) and \(\otimes\), \(\oplus\) is a binary, associative, commutative operator, \(\otimes\) is a binary, associative operator that distributes over \(\oplus\), \(0 \in K\) is the \(\oplus\)-identity, and \(1 \in K\) is the \(\otimes\)-identity.
Semantics

• Semiring *

• valuation function maps provable ground terms to values

• extended to expressions, e.g. \([x \ast y] \overset{\text{def}}{=} [x] \ast [y]\)

• weighted rules \(r \oplus_r = E\) constrain evaluation

\(\begin{bmatrix} r \end{bmatrix} = \begin{bmatrix} E_1 \end{bmatrix} \oplus_r \begin{bmatrix} E_2 \end{bmatrix} \oplus_r \ldots\)

grounding of a rule body

* An algebraic semiring consists of five elements \(\langle K, \oplus, \otimes, 0, 1 \rangle\), where \(K\) is a domain closed under \(\oplus\) and \(\otimes\), \(\oplus\) is a binary, associative, commutative operator, \(\otimes\) is a binary, associative operator that distributes over \(\oplus\), \(0 \in K\) is the \(\oplus\)-identity, and \(1 \in K\) is the \(\otimes\)-identity.
Roadmap

- Modeling
- Reasoning
- Language extensions
- Advanced topics

... with some detours on the way
Advanced Topics

• parameter estimation
• complexity of querying
• lifted graphical models and KBMC
Parameter Learning
Parameter Learning

e.g., webpage classification model

for each \texttt{CLASS1, CLASS2} and each \texttt{WORD}

\begin{itemize}
  \item \texttt{link\_class(Source, Target, CLASS1, CLASS2)}.
  \item \texttt{word\_class(WORD, CLASS)}.
\end{itemize}

\begin{itemize}
  \item \texttt{class(Page, C) :- has\_word(Page, W), word\_class(W, C)}.
  \item \texttt{class(Page, C) :- links\_to(OtherPage, Page), class(OtherPage, OtherClass), link\_class(OtherPage, Page, OtherClass, C)}.
\end{itemize}
Sampling Interpretations
Sampling Interpretations
Parameter Estimation
Parameter Estimation

\[ p(\text{fact}) = \frac{\text{count(\text{fact is true})}}{\text{Number of interpretations}} \]
Learning from partial interpretations

- Not all facts observed
- Soft-EM
- Use **expected count** instead of **count**
- $P(Q |E)$ -- conditional queries!
Bayesian Parameter Learning

- Learning as inference (e.g., Church)
- Prior distributions for parameters
- Given data, find most likely parameter values
Example

- Flipping a coin with unknown weight
- Prior: uniform distribution on [0,1]
- Observation: 5x heads in a row
- Sampling from Church model:
ProbLog Example

Prior

0.05::weight(C,0.1); 0.2::weight(C,0.3); 0.5::weight(C,0.5);
  0.2::weight(C,0.7); 0.05::weight(C,0.9) <- coin(C).

Param::toss(_,Param,__).
heads(C,R) :- weight(C,Param),toss(C,Param,R).
tails(C,R) :- weight(C,Param),\+toss(C,Param,R).
data(C,[]).
data(C,[h|R]) :- heads(C,R), data(C,R).
data(C,[t|R]) :- tails(C,R), data(C,R).

Ask for posterior

query(weight(C,X)) :- coin(C),param(X).

evidence(data(c1,[h,h,h,h,h,h,h,h,h,h,h,h,h]),true).
evidence(data(c2,[h,t,h,h,h,h,t,t,h,t,t,h]),true).

data
query(weight(C,X)) :- coin(C),param(X).

evidence(data(c1,[h,h,h,h,h,h,h,h,h,h,h,h,h]),true).
evidence(data(c2,[h,t,h,h,h,h,t,t,h,t,t,h]),true).

0.05::weight(C,0.1); 0.2::weight(C,0.3); 0.5::weight(C,0.5);
0.2::weight(C,0.7); 0.05::weight(C,0.9) <- coin(C).
Complexity of Querying
Complexity of querying

select x.Product, x.Company
from ProducesProduct x, HeadquarteredIn y
where x.Company=y.Company and
y.City='san_jose'

\[
\begin{align*}
0.96 \times 0.99 &= 0.95 \\
0.9 \times 0.93 &= 0.83 \\
0.87 \times 0.93 &= 0.80
\end{align*}
\]
select distinct x.Product, x.Company, 
    x.P * y.P as P 
from ProducesProduct x, HeadquarteredIn y 
where x.Company = y.Company 
    and y.City = 'san_jose'
order by P desc

0.96x0.99=0.95
0.9x0.93=0.83
0.87x0.93=0.80
Complexity of querying

result(Product, Company) :-
    producesProduct(Company, Product),
    headquarteredIn(Company, san_jose).
query(result(_, _)).
Complexity of querying

result(Product, Company) :-
  producesProduct(Company, Product),
  headquarteredIn(Company, san_jose).
query(result(_, _)).

each ground query has a single proof
→ no disjoint-sum-problem,
   
   easy evaluation
Complexity of querying in probabilistic databases

- queries have fixed size (no recursion)
- size of query $\ll$ size of database
- complexity of evaluating given query measured in size of database (data complexity)
- previous example: polynomial (as all standard relational database queries)
How hard is evaluating q?

0.3::stress(X):- person(X).
0.2::influences(X,Y):-
    person(X), person(Y), X \= Y.

q :- stress(X), influences(X,Y).

person(1).
person(2).
person(3).
How hard is evaluating q?

0.3::stress(X):- person(X).
0.2::influences(X,Y):-
    person(X), person(Y), X \= Y.

q :- stress(X), influences(X,Y).

proofs

s(1),i(1,2)
s(1),i(1,3)
s(2),i(2,1)
s(2),i(2,3)
s(3),i(3,1)
s(3),i(3,2)

person(1).
person(2).
person(3).
How hard is evaluating \( q \)?

\[
\begin{align*}
0.3 & \cdot \text{stress}(X) : - \text{person}(X). \\
0.2 & \cdot \text{influences}(X,Y) : - \\
& \quad \text{person}(X), \text{person}(Y), X \neq Y. \\
q & : - \text{stress}(X), \text{influences}(X,Y).
\end{align*}
\]

proofs

\[
\begin{align*}
\text{s}(1), \text{i}(1,2) \\
\text{s}(1), \text{i}(1,3) \\
\text{s}(2), \text{i}(2,1) \\
\text{s}(2), \text{i}(2,3) \\
\text{s}(3), \text{i}(3,1) \\
\text{s}(3), \text{i}(3,2)
\end{align*}
\]

tree structure, all leaves different
How hard is evaluating q?

0.3::stress(X):- person(X).
0.2::influences(X,Y):-
  person(X), person(Y), X \= Y.
q :- stress(X), influences(X,Y).

proofs
\begin{align*}
\text{s(1), } &\text{i(1,2)} \\
\text{s(1), } &\text{i(1,3)} \\
\text{s(2), } &\text{i(2,1)} \\
\text{s(2), } &\text{i(2,3)} \\
\text{s(3), } &\text{i(3,1)} \\
\text{s(3), } &\text{i(3,2)}
\end{align*}

\[
P(\varphi_1 \land \varphi_2) = P(\varphi_1) \cdot P(\varphi_2)
\]
\[
P(\varphi_1 \lor \varphi_2) = 1 - (1 - P(\varphi_1)) \cdot (1 - P(\varphi_2))
\]

\text{tree structure, all leaves different}
\[ P(q) = 1 - \prod_{j=1..n} \left( 1 - P(s(x_j)) \left( 1 - \prod_{k=1..n, k \neq j} \left( 1 - P(i(x_j, y_k)) \right) \right) \right) \]

polynomial in database size / number of persons \( n \)

proofs
- \( s(1), i(1,2) \)
- \( s(1), i(1,3) \)
- \( s(2), i(2,1) \)
- \( s(2), i(2,3) \)
- \( s(3), i(3,1) \)
- \( s(3), i(3,2) \)

Tree structure, all leaves different

\[ P(\varphi_1 \wedge \varphi_2) = P(\varphi_1) \cdot P(\varphi_2) \]
\[ P(\varphi_1 \vee \varphi_2) = 1 - (1 - P(\varphi_1)) \cdot (1 - P(\varphi_2)) \]
How hard is evaluating q?

0.3::stress(X) :- person(X).
0.5::male(X) :- person(X).
0.2::influences(X,Y):-
    person(X), person(Y), X \= Y.

q :- stress(X), influences(X,Y), male(Y).

person(1).
person(2).
person(3).
How hard is evaluating q?

0.3::stress(X):- person(X).
0.5::male(X):- person(X).
0.2::influences(X,Y):-
    person(X), person(Y), X \= Y.

q :- stress(X), influences(X,Y), male(Y).

person(1).
person(2).
person(3).
How hard is evaluating q?

0.3::stress(X):- person(X).
0.5::male(X) :- person(X).
0.2::influences(X,Y):-
  person(X), person(Y), X \neq Y.

q :- stress(X), influences(X,Y), male(Y).

proofs

s(1),i(1,2),m(2)
s(1),i(1,3),m(3)
s(2),i(2,1),m(1)
s(2),i(2,3),m(3)
s(3),i(3,1),m(1)
s(3),i(3,2),m(2)

person(1).
person(2).
person(3).
How hard is evaluating $q$?

0.3::stress(X):- person(X).
0.5::male(X) :- person(X).
0.2::influences(X,Y):-
    person(X), person(Y), X ≠ Y.

$q$ :- stress(X), influences(X,Y), male(Y).

proofs

- $s(1),i(1,2),m(2)$
- $s(1),i(1,3),m(3)$
- $s(2),i(2,1),m(1)$
- $s(2),i(2,3),m(3)$
- $s(3),i(3,1),m(1)$
- $s(3),i(3,2),m(2)$

person(1).
person(2).
person(3).
How hard is evaluating q?

0.3::stress(X):- person(X).
0.5::male(X) :- person(X).
0.2::influences(X,Y):-
    person(X), person(Y), X \= Y.

q :- stress(X), influences(X,Y), male(Y).

proofs

cannot build tree structure with all leaves different
Read-Once Formulas

• Propositional formulas that can be rewritten such that each variable occurs at most once (= as tree with all leaves different)
Read-Once Formulas

• Propositional formulas that can be rewritten such that each variable occurs at most once (= as tree with all leaves different)

• Can be evaluated in polynomial time
Read-Once Formulas

• Propositional formulas that can be rewritten such that each variable occurs at most once (= as tree with all leaves different)

• Can be evaluated in polynomial time

• Unate formula: read once $\leftrightarrow$ P4-free & normal
Read-Once Formulas

• Propositional formulas that can be rewritten such that each variable occurs at most once (= as tree with all leaves different)

• Can be evaluated in polynomial time

• Unate formula: read once $\leftrightarrow$ P4-free & normal

  no variable appears both pos & neg
Read-Once Formulas

- Propositional formulas that can be rewritten such that each variable occurs at most once (= as tree with all leaves different)
- Can be evaluated in polynomial time
- Unate formula: read once $\leftrightarrow$ P4-free & normal

$X \vee Y$ is unate

$(X \vee Y) \land (\neg X \vee Z)$ not
P4-free and normal

• represent DNF formula as graph:
  • a node for each variable
  • edge \((X,Y) \iff X,Y\) in same conjunct
P4-free and normal

• represent DNF formula as graph:
  • a node for each variable
  • edge (X,Y) \iff X,Y in same conjunct

• **normal**: each clique is part of a conjunct
P4-free and normal

- represent DNF formula as graph:
  - a node for each variable
  - edge \((X,Y)\) \iff \(X,Y\) in same conjunct
- normal: each clique is part of a conjunct
- P4-free: no induced 4 node subgraph is a path
Our first example

\[s(1), i(1,2)\]
\[s(1), i(1,3)\]
\[s(2), i(2,1)\]
\[s(2), i(2,3)\]
\[s(3), i(3,1)\]
\[s(3), i(3,2)\]

- **normal**: each clique is part of a conjunct
- **P4-free**: no induced 4 node subgraph is a path
Our first example

- **normal**: each clique is part of a conjunct
- **P4-free**: no induced 4 node subgraph is a path
Our first example

- **normal**: each clique is part of a conjunct
- **P4-free**: no induced 4 node subgraph is a path
Our first example

- **normal**: each clique is part of a conjunct
- **P4-free**: no induced 4 node subgraph is a path
  
  read-once
Our second example

\[
\begin{align*}
  &s(1),i(1,2),m(2) \\
  &s(1),i(1,3),m(3) \\
  &s(2),i(2,1),m(1) \\
  &s(2),i(2,3),m(3) \\
  &s(3),i(3,1),m(1) \\
  &s(3),i(3,2),m(2)
\end{align*}
\]

- **normal**: each clique is part of a conjunct
- **P4-free**: no induced 4 node subgraph is a path
Our second example

\[
\begin{align*}
&\text{s(1), i(1,2), m(2)} \\
&\text{s(1), i(1,3), m(3)} \\
&\text{s(2), i(2,1), m(1)} \\
&\text{s(2), i(2,3), m(3)} \\
&\text{s(3), i(3,1), m(1)} \\
&\text{s(3), i(3,2), m(2)}
\end{align*}
\]

- **normal**: each clique is part of a conjunct
- **P4-free**: no induced 4 node subgraph is a path
Our second example

- **normal**: each clique is part of a conjunct
- **P4-free**: no induced 4 node subgraph is a path
Our second example

\[ s(1), i(1, 2), m(2) \]
\[ s(1), i(1, 3), m(3) \]
\[ s(2), i(2, 1), m(1) \]
\[ s(2), i(2, 3), m(3) \]
\[ s(3), i(3, 1), m(1) \]
\[ s(3), i(3, 2), m(2) \]

- **normal**: each clique is part of a conjunct
  ✔

- **P4-free**: no induced 4 node subgraph is a path
  ✗

not read-once
(and in fact known to be \#P-hard, cf. PDB-book)
Dichotomy of UCQ Evaluation

- **Union of Conjunctive Queries**
  \( \approx \) Datalog without recursion and negation

- Theorem: UCQ evaluation is either polynomial in database size or \#P-hard
Dichotomy of UCQ Evaluation

- Union of Conjunctive Queries ≈ Datalog without recursion and negation
- Theorem: UCQ evaluation is either polynomial in database size or \#P-hard

counting version of NP decision problems, e.g., model counting
#P-hard

polynomial

**Figure 5.4:** The query compilation hierarchy for Unions of Conjunctive Queries (UCQ).

Fig. from [Suciu et al 2011]
#P-hard

polynomial

\( s(X), i(X,Y) \)

**Figure 5.4:** The query compilation hierarchy for Unions of Conjunctive Queries (UCQ).

Fig. from [Suciu et al 2011]
Figure 5.4: The query compilation hierarchy for Unions of Conjunctive Queries (UCQ).

\[ H_0 = s(X), i(X,Y), m(Y) \]

#P-hard

polynomial

Fig. from [Suciu et al 2011]
A key question in AI:

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

Learning
- parameters
- structure

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

Statistical relational learning, probabilistic logic learning, probabilistic 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, ...
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, ...
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, ...
(lifted) graphical models
Lifted graphical models

Reasoning with relational data

Dealing with uncertainty

Learning
Lifted graphical models

Reasoning with relational data

graphical model

Learning
Lifted graphical models

Reasoning with relational data

graphical model
Lifted graphical models

fixed set of random variables

difficultCourse  smartStudent

success

graphical model

Learning

relational definition of graphical model

diff(C)  smart(S)

takes(S,C)

succ(S,C)
Lifted graphical models

fixed set of random variables

- difficultCourse
- smartStudent
- success

graphical model

data-dependent set of random variables

relational definition of graphical model

- diff(C)
- smart(S)
- takes(S,C)
- succ(S,C)

Learning
Lifted graphical models

relational definition of graphical model

relational language defines graphical model

data-dependent set of random variables

fixed set of random variables

difficultCourse

smartStudent

success

Learning

diff(C)

smart(S)

takes(S,C)

succ(S,C)
Lifted graphical models

fixed set of random variables
- difficultCourse
- smartStudent
- success

graphical model

relational language defines graphical model

data-dependent set of random variables
- diff(C)
- smart(S)
- takes(S,C)
- succ(S,C)

Learning parameters & structure

relational definition of graphical model

parameters & structure
Bayesian Network

difficult

smart

success
Bayesian Network

\[ P(\text{difficult=T}) = 0.6 \]

\[
\begin{array}{c}
\text{difficult} \\
\downarrow \\
\text{success} \\
\uparrow \\
\text{smart}
\end{array}
\]
Bayesian Network

\[ P(\text{difficult}=T) = 0.6 \]
\[ P(\text{smart}=T) = 0.7 \]
Bayesian Network

P(difficult=T) = 0.6
P(smart=T) = 0.7
P(success=T|d=T,sm=T) = 0.85
P(success=T|d=T,sm=F) = 0.10
P(success=T|d=F,sm=T) = 0.98
P(success=T|d=F,sm=F) = 0.45
Bayesian Network

\[ P(\text{difficult}=T) = 0.6 \]
\[ P(\text{smart}=T) = 0.7 \]
\[ P(\text{success}=T|d=T, sm=T) = 0.85 \]
\[ P(\text{success}=T|d=T, sm=F) = 0.10 \]
\[ P(\text{success}=T|d=F, sm=T) = 0.98 \]
\[ P(\text{success}=T|d=F, sm=F) = 0.45 \]

Joint distribution:
\[ P(\text{difficult}) \times P(\text{smart}) \times P(\text{success}|\text{difficult, smart}) \]
Bayesian Network

\[ P(\text{difficult}=T) = 0.6 \]

\[ P(\text{smart}=T) = 0.7 \]

\[ P(\text{success}=T|d=T,sm=T) = 0.85 \]
\[ P(\text{success}=T|d=T,sm=F) = 0.10 \]
\[ P(\text{success}=T|d=F,sm=T) = 0.98 \]
\[ P(\text{success}=T|d=F,sm=F) = 0.45 \]

Joint distribution:
\[ P(\text{difficult}) \times P(\text{smart}) \times P(\text{success}|\text{difficult,smart}) \]

directed
acyclic graph

\[ P(X_1, \ldots, X_n) = \prod_{i=1,\ldots,n} P(X_i|\text{par}(X_i)) \]
Bayesian Network in ProbLog?

P(difficult=T) = 0.6
P(smart=T) = 0.7
P(success=T|d=T,sm=T) = 0.85
P(success=T|d=T,sm=F) = 0.10
P(success=T|d=F,sm=T) = 0.98
P(success=T|d=F,sm=F) = 0.45
Bayesian Network in ProbLog?

$P(\text{difficult}=T) = 0.6$

$P(\text{smart}=T) = 0.7$

$P(\text{success}=T|d=T, sm=T) = 0.85$

$P(\text{success}=T|d=T, sm=F) = 0.10$

$P(\text{success}=T|d=F, sm=T) = 0.98$

$P(\text{success}=T|d=F, sm=F) = 0.45$

0.6::difficult.
0.7::smart.
0.85::success <- difficult, smart.
0.10::success <- difficult, \+smart.
0.98::success <- \+difficult, smart.
0.45::success <- \+difficult, \+smart.
Markov Network
Markov Network

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>D</th>
<th>( \phi_1(A, B, D) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>5</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>5</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>5</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>5</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>1</td>
</tr>
</tbody>
</table>
Markov Network

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>D</th>
<th>$\phi_1(A, B, D)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>5</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>5</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>5</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>5</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B</th>
<th>C</th>
<th>D</th>
<th>$\phi_2(B, C, D)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>4</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>2</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>3</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>2</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>4</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>3</td>
</tr>
</tbody>
</table>
Markov Network

\[
P(A = T, B = F, C = T, D = F) = \frac{1}{Z} \phi_1(A = T, B = F, D = F) \phi_2(B = F, C = T, D = F)
= \frac{1}{Z} \cdot 5 \cdot 4
\]
Markov Network

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>D</th>
<th>$\phi_1(A, B, D)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>5</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>5</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>5</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>5</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ P(A = T, B = F, C = T, D = F) = \frac{1}{Z} \phi_1(A = T, B = F, D = F) \phi_2(B = F, C = T, D = F) \]
\[ = \frac{1}{Z} \cdot 5 \cdot 4 \]

undirected graph

\[
P(X = x) = \frac{1}{Z} \prod_{\phi_i \in F} \phi_i(X_{\phi_i} = x_{\phi_i})
\]

\[
Z = \sum_{x'} \prod_{\phi_i \in F} \phi_i(x_{\phi_i} = x'_{\phi_i})
\]
Markov Network in ProbLog?

\[ \phi_1(A, B, D) \]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>D</th>
<th>\phi_1(A, B, D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>5</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>1</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>5</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>5</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>5</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ \phi_2(B, C, D) \]

<table>
<thead>
<tr>
<th>B</th>
<th>C</th>
<th>D</th>
<th>\phi_2(B, C, D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>4</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>2</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>3</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>2</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>4</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>3</td>
</tr>
</tbody>
</table>
Markov Network in ProbLog?

no direct mapping due to general potential functions and normalization
Factor Graph

- bi-partite undirected graph
- variables $X_1, \ldots, X_m$
- factors $\phi_1, \ldots, \phi_n$ map interpretations of subsets of variables to non-negative reals
- joint distribution

$$P(X = x) = \frac{1}{Z} \prod_{i=1\ldots n} \phi_i(X_{\phi_i} = x_{\phi_i})$$

$$Z = \sum_{x'} \prod_{i=1\ldots n} \phi_i(X_{\phi_i} = x'_{\phi_i})$$
MN as Factor Graph?
MN as Factor Graph?

\[ \phi_1(A, B, D) \]
\[ \phi_2(B, C, D) \]
BN as Factor Graph?

difficult

smart

success
BN as Factor Graph?

\[ \phi_1(d) = P(d) \]
\[ \phi_2(su, d, sm) = P(su|d, sm) \]
\[ \phi_3(sm) = P(sm) \]

\[ Z=1 \]
More students, more courses...
More students, more courses...

gets cumbersome quickly.... relational language to write template?

s(P)  su(P,C)  d(C)
Lots of proposals in the literature, e.g.

- relational Markov networks (RMNs) [Taskar et al 2002]
- Markov logic networks (MLNs) [Richardson & Domingos 2006]
- probabilistic soft logic (PSL) [Broecheler et al 2010]
- FACTORIE [McCallum et al 2009]
- Bayesian logic programs (BLPs) [Kersting & De Raedt 2001]
- relational Bayesian networks (RBNs) [Jaeger 2002]
- logical Bayesian networks (LBNs) [Fierens et al 2005]
- probabilistic relational models (PRMs) [Koller & Pfeffer 1998]
- Bayesian logic (BLOG) [Milch et al 2005]
- CLP(BN) [Santos Costa et al 2008]
- probabilistic programming languages such as ProbLog, PRISM, Church, ...
- and many more ...
Lots of proposals in the literature, e.g.

- relational Markov networks (RMNs) [Taskar et al 2002]
- Markov logic networks (MLNs) [Richardson & Domingos 2006]
- probabilistic soft logic (PSL) [Broecheler et al 2010]
- FACTORIE [McCallum et al 2009]
- Bayesian logic programs (BLPs) [Kersting & De Raedt 2001]
- relational Bayesian networks (RBNs) [Jaeger 2002]
- logical Bayesian networks (LBNs) [Fierens et al 2005]
- probabilistic relational models (PRMs) [Koller & Pfeffer 1998]
- Bayesian logic (BLOG) [Milch et al 2005]
- CLP(BN) [Santos Costa et al 2008]
- probabilistic programming languages such as ProbLog, PRISM, Church, ...
- and many more ...

common principle:
parameterized factor graph
(par-factor graph)
Par-Factor Graph

[Poole 03]
The Par-Factor Graph is parameterized factor (par-factor), which can be defined as:

\[ \phi_1(\text{grade}(S,C), \text{difficult}(C)) \]
Par-Factor Graph

[Poole 03]

\[ \phi_1(\text{grade}(S,C), \text{difficult}(C)) \]

parameterized factor (par-factor)
Par-Factor Graph

\[ \phi_1(\text{grade}(S,C), \text{difficult}(C)) \]

grounding constraint

parameterized factor (par-factor)
Par-Factor Graph

grade(S,C)

takes(S,C)

$\phi_1(\text{grade}(S,C), \text{difficult}(C))$

difficult(C)

parameterized factor (par-factor)

potential function
Par-Factor Graph

\( \text{grade}(S, C) \)

\( \text{takes}(S, C) \)
\[ \phi_1(\text{grade}(S, C), \text{difficult}(C)) \]

\( \text{difficult}(C) \)

\( \text{teaches}(P, C) \)
\[ \phi_2(\text{demanding}(P), \text{difficult}(C)) \]

\( \text{demanding}(P) \)
\[
P(.) = \frac{1}{Z} \cdot \prod_{\text{takes}(s,c)} \phi_1(\text{grade}(s,c), \text{difficult}(c)) \cdot \prod_{\text{teaches}(p,c)} \phi_2(\text{demanding}(p), \text{difficult}(c))
\]
Par-Factor Graph

$P(.) = \frac{1}{Z} \cdot \prod_{\text{takes}(s,c)} \phi_1(\text{grade}(s,c), \text{difficult}(c)) \\
\phantom{.} \cdot \prod_{\text{teaches}(p,c)} \phi_2(\text{demanding}(p), \text{difficult}(c))$
What if multiple professors teach the same course?

difficult(C)

demanding(P)

teaches(P,C)

\( \phi_2(\cdot, \cdot) \)

teaches(eve,db)
teaches(dan,db).
teaches(tom,ml).
teaches(bob,ai).
teaches(ann,ai).
teaches(ed,ai).
What if multiple professors teach the same course?

difficult(C) teaches(P,C) teaches(P,C) = \phi_2(\cdot,\cdot) 

demanding(P) 

teaches(eve,db) teaches(dan,db).
teaches(tom,ml). 
teaches(bob,ai). 
teaches(ann,ai). 
teaches(ed,ai). 

Option 1: aggregate values of RVs, then compute potential

\phi_2(\text{demanding}(P), \text{difficult}(C)) = P(\text{difficult}(C)|\text{mean}\{\text{demanding}(P)\})
What if multiple professors teach the same course?

difficult(C)

demanding(P)

teaches(P,C)

\[ \phi_2(\cdot, \cdot) \]

teaches(eve,db)
teaches(dan,db).
teaches(tom,ml).
teaches(bob,ai).
teaches(ann,ai).
teaches(ed,ai).

Option 1: aggregate values of RVs, then compute potential

\[ \phi_2(\text{demanding}(P), \text{difficult}(C)) = P(\text{difficult}(C) | \text{mean}\{\text{demanding}(P)\}) \]
What if multiple professors teach the same course?

difficult(C) 

\text{teaches}(P,C) \quad \phi_2(\cdot,\cdot) 

demanding(P) 

\text{teaches}(eve,db)
\text{teaches}(dan,db).
\text{teaches}(tom,ml).
\text{teaches}(bob,ai).
\text{teaches}(ann,ai).
\text{teaches}(ed,ai).

Option 2: compute potential for each RV, then combine

\phi_2(\text{demanding}(P),\text{difficult}(C)) = 1 - \prod_p (1 - P(\text{difficult}(C) | \text{demanding}(P)))
What if multiple professors teach the same course?

Option 2: compute potential for each RV, then combine

\[ \phi_2(demanding(P), difficult(C)) = 1 - \prod_P (1 - P(difficult(C)|demanding(P))) \]
Inference by Grounding

demanding(tom)?
Inference by Grounding

```
takes(ann, ml).
takes(bob, ml).
takes(ann, db).
teaches(dan, db).
teaches(tom, ml).
```

demanding(tom)?

1. construct factor graph
2. run any propositional inference technique
Inference by Grounding

demanding(tom) ?

1. construct factor graph
2. run any propositional inference technique

\[
\begin{align*}
\text{grade}(S,C) & \quad \phi_1(\text{grade}(S,C), \text{difficult}(C)) \\
\text{difficult}(C) & \quad \phi_2(\text{demanding}(P), \text{difficult}(C)) \\
\text{teaches}(P,C) & \quad \phi_2(\text{demanding}(P), \text{difficult}(C))
\end{align*}
\]
Inference by Grounding

demanding(tom)?

takes(S,C).
difficult(C).
demanding(P).

teaches(P,C).

Problem: often huge factor graphs with many repetitions or symmetries (even if restricted to relevant parts)

1. construct factor graph
2. run any propositional inference technique

takes(ann,ml).
takes(bob,ml).
takes(ann,db).
teaches(dan,db).
teaches(tom,ml).

demanding(tom)?
compute probability distribution over grades for every student-course pair
compute probability distribution over grades for every student-course pair same computation for each pair!
what is the probability that some professor is demanding?
what is the probability that some professor is demanding?
same computation for each professor!
what is the probability that some professor is demanding?

\[
1 - \prod_{P} (1 - P(\text{de}(P))) = 1 - (1 - P(\text{de(someP)}))^\#P
\]

\[= 1 - (1 - P(\text{de(r)}))^5\]
what is the probability that some professor is demanding?

\[ 1 - \prod_p (1 - P(\text{de}(\text{P}))) = 1 - (1 - P(\text{de}(\text{someP})))^\#P \]

\[ = 1 - (1 - P(\text{de}(\text{r})))^5 \]
Another example

\[ P(\text{de (t) = true})? \]

\[ \text{de(t)} \rightarrow \text{di(ml)} \rightarrow g(1,ml) \rightarrow \ldots \rightarrow g(n,ml) \]
Another example

\[ \sum \phi_2(\text{true}, d) \prod_{i=1}^{n} \phi_1(g_i, d) \]

for all combinations of difficulty \(d\) and students’ grades \((g_1, ..., g_n)\)
Another example

\[ \sum \phi_2(\text{true}, d) \prod_{i=1}^{n} \phi_1(g_i, d) \]
for all combinations of difficulty \( d \) and students’ grades \((g_1, \ldots, g_n)\)

\( O(\#\text{grades} \times \#\text{students}) \) assignments to sum!
Another example

\[ P(\text{de}(t) = \text{true})? \]

\[ \sum \phi_2(\text{true}, d) \prod_{i=1\ldots n} \phi_1(g_i, d) \]

for all combinations of difficulty \( d \) and students’ grades \( g_1, \ldots, g_n \)

\[ O(\#\text{grades}^{\#\text{students}}) \] assignments to sum!

but: identity of students doesn’t matter!
sufficient to count how often each grade \( m_1, \ldots, m_k \) appears
Another example

\[
\text{for all combinations of difficulty } d \text{ and students' grades } (g_1, \ldots, g_n)
\]

\[
\phi_2(\text{true}, d) \prod_{i=1..n} \phi_1(g_i, d)
\]

\[
\Theta(\#\text{grades} \cdot \#\text{students}) \text{ assignments to sum!}
\]

but: identity of students doesn’t matter!
sufficient to count how often each grade \( m_1, \ldots, m_k \) appears

\[
\text{sum } \phi_2(\text{true}, d) \prod_{i=1..k} \phi_1(m_i, d)^{\#m_i} \text{ instead}
\]
Another example

\[ \sum \phi_2(\text{true}, d) \prod_{i=1..n} \phi_1(g_i, d) \]

for all combinations of difficulty \( d \) and students’ grades \( (g_1, ..., g_n) \)

\( O(\#\text{grades}^{\#\text{students}}) \) assignments to sum!

but: identity of students doesn’t matter!
sufficient to count how often each grade \( m_1, ..., m_k \) appears

\[ \sum \phi_2(\text{true}, d) \prod_{i=1..k} \phi_1(m_i, d)^{\#m_i} \text{ instead} \]

e.g., \( k=3, n=15 \): >14M grade vectors vs 136 count vectors
Lifted Inference

• exploiting symmetries & repeated structure
• reasoning on first order level as much as possible
• aiming at independence from number of objects
• approximation: grouping similar computations
• very active research area
• one example: weighted first order model counting
Parameter Learning

- **data** = interpretations on ground level

- **estimate** parameters on first-order level taking all instances of same par-RV together
Parameter Learning

- **data** = interpretations on ground level
- **estimate** parameters on first-order level taking all instances of same par-RV together

```
\begin{align*}
\text{diff}(C) &= t \\
\text{diff}(db) &= f \\
\text{grade}(a, ml) &= h \\
\text{grade}(b, ml) &= l \\
\text{grade}(a, db) &= l \\
\text{grade}(c, db) &= h \\
\text{diff}(se) &= f \\
\text{diff}(ds) &= t \\
\text{grade}(b, se) &= l \\
\text{grade}(c, se) &= l \\
\text{grade}(d, ds) &= h \\
\text{grade}(g, ds) &= l \\
\text{diff}(ai) &= t \\
\text{grade}(f, ai) &= l \\
\text{grade}(e, ai) &= h \\
\text{grade}(g, ai) &= h \\
\text{grade}(d, ai) &= h \\
\text{grade}(a, ai) &= l
\end{align*}
```
Parameter Learning

• **data** = interpretations on ground level

• **estimate** parameters on first-order level taking all instances of same par-RV together

\[
P(\text{grade}=h|\text{diff}=t) = \]

\[
\begin{align*}
\text{diff}(C) & = t \\
\text{diff}(db) & = f \\
\text{grade}(a, ml) & = h \\
\text{grade}(b, ml) & = l \\
\text{grade}(a, db) & = l \\
\text{grade}(c, db) & = h \\
\text{grade}(S,C) \quad \text{diff}(ml) & = t \\
\text{diff}(db) & = f \\
\text{grade}(a, ml) & = h \\
\text{grade}(b, ml) & = l \\
\text{grade}(a, db) & = l \\
\text{grade}(c, db) & = h \\
\text{diff}(se) & = f \\
\text{diff}(ds) & = t \\
\text{grade}(b, se) & = l \\
\text{grade}(c, se) & = l \\
\text{grade}(d, ds) & = h \\
\text{grade}(g, ds) & = l \\
\text{diff}(ai) & = t \\
\text{grade}(f, ai) & = l \\
\text{grade}(e, ai) & = h \\
\text{grade}(g, ai) & = h \\
\text{grade}(d, ai) & = h \\
\text{grade}(a, ai) & = l
\end{align*}
\]
Parameter Learning

- **data** = interpretations on ground level
- **estimate** parameters on first-order level taking all instances of same par-RV together

\[
P(\text{grade}=h|\text{diff}=t) = \frac{5}{9}\]
Parameter Learning

• **data** = interpretations on ground level

• **estimate** parameters on first-order level taking all instances of same par-RV together

\[
P(\text{grade} = h | \text{diff} = t) = \frac{5}{9} \\
P(\text{grade} = h | \text{diff} = f) = \
\]
Parameter Learning

- **data** = interpretations on ground level
- **estimate** parameters on first-order level taking all instances of same par-RV together

\[
P(\text{grade}=h|\text{diff}=t) = \frac{5}{9} \quad P(\text{grade}=h|\text{diff}=f) = \frac{1}{4}
\]
Parameter Learning

• **data** = interpretations on ground level

• **estimate** parameters on first-order level taking all instances of same par-RV together

\[
\begin{align*}
\text{diff}(C) \\
\text{grade}(S,C) \\
\text{diff}(ml) &= t \\
\text{diff}(db) &= f \\
\text{grade}(a,ml) &= h \\
\text{grade}(b,ml) &= l \\
\text{grade}(a,db) &= l \\
\text{grade}(c,db) &= h \\
\text{diff}(se) &= f \\
\text{diff}(ds) &= t \\
\text{grade}(b,se) &= l \\
\text{grade}(c,se) &= l \\
\text{grade}(d,ds) &= h \\
\text{grade}(g,ds) &= l \\
\text{diff}(ai) &= t \\
\text{grade}(f,ai) &= l \\
\text{grade}(e,ai) &= h \\
\text{grade}(g,ai) &= h \\
\text{grade}(d,ai) &= h \\
\text{grade}(a,ai) &= l
\end{align*}
\]

\[
\begin{align*}
P(\text{grade}=h|\text{diff}=t) &= \frac{5}{9} \\
P(\text{grade}=h|\text{diff}=f) &= \frac{1}{4} \\
P(\text{diff}=t) &= \frac{3}{5}
\end{align*}
\]
Structure Learning

- often based on search over candidate structures of specific form

- close connection to both inductive logic programming and structure learning in graphical models
Languages to define par-factor graphs

- relational Markov networks (RMNs) [Taskar et al 2002]
- Markov logic networks (MLNs) [Richardson & Domingos 2006]
- probabilistic soft logic (PSL) [Broecheler et al 2010]
- FACTORIE [McCallum et al 2009]
- Bayesian logic programs (BLPs) [Kersting & De Raedt 2001]
- relational Bayesian networks (RBNs) [Jaeger 2002]
- logical Bayesian networks (LBNs) [Fierens et al 2005]
- probabilistic relational models (PRMs) [Koller & Pfeffer 1998]
- Bayesian logic (BLOG) [Milch et al 2005]
- CLP(BN) [Santos Costa et al 2008]
- probabilistic programming languages such as ProbLog, PRISM, Church, ...
- and many more ...
Languages to define par-factor graphs

- relational Markov networks (RMNs) [Taskar et al 2002]
- Markov logic networks (MLNs) [Richardson & Domingos 2006]
- probabilistic soft logic (PSL) [Broecheler et al 2010]
- FACTORIE [McCallum et al 2009]
- Bayesian logic programs (BLPs) [Kersting & De Raedt 2001]
- relational Bayesian networks (RBNs) [Jaeger 2002]
- logical Bayesian networks (LBNs) [Fierens et al 2005]
- probabilistic relational models (PRMs) [Koller & Pfeffer 1998]
- Bayesian logic (BLOG) [Milch et al 2005]
- CLP(BN) [Santos Costa et al 2008]
- probabilistic programming languages such as ProbLog, PRISM, Church, ...
- and many more ...
CLP(BN) [Santos Costa et al, 2008]
constraint logic programming
for Bayesian networks
CLP(BN) [Santos Costa et al, 2008]  
constraint logic programming for Bayesian networks

\[ \phi_1(d) = P(d) \]
\[ \phi_2(su, d, sm) = P(su|d, sm) \]
\[ \phi_3(sm) = P(sm) \]
CLP(BN) [Santos Costa et al, 2008]

constraint logic programming for Bayesian networks

difficult(Course, Diff) :-
   { Diff = d(Course)
     with p([t,f],[0.6,0.4],[[]])
   }
CLP(BN) [Santos Costa et al, 2008]

constraint logic programming for Bayesian networks

\[
\phi_1(d) = P(d)
\]

\[
\phi_2(su, d, sm) = P(su|d, sm)
\]

\[
\phi_3(sm) = P(sm)
\]

par-RV: unique skolem term
difficult(Course, Diff) :-
\{ Diff = \text{d}(\text{Course})
\text{with } p([t, f], [0.6, 0.4], []) \}
**CLP(BN)** [Santos Costa et al, 2008]

constraint logic programming for Bayesian networks

**par-RV**: unique skolem term

difficult(Course, Diff) :-
{ Diff = d(Course) 
  with p([t, f], [0.6, 0.4], []) }

possible values

\[
\phi_1(d) = P(d) \\
\phi_2(su, d, sm) = P(su|d, sm) \\
\phi_3(sm) = P(sm)
\]
**CLP(BN)** [Santos Costa et al, 2008]

**Constraint Logic Programming for Bayesian Networks**

**par-RV:** unique skolem term

difficult(Course, Diff) :-
{ Diff = d(Course)  
  with p([t,f],[0.6,0.4],[]) }  

possible values
their probabilities

\[ \phi_1(d) = P(d) \]

\[ \phi_2(su, d, sm) = P(su|d, sm) \]

\[ \phi_3(sm) = P(sm) \]
CLP(BN) [Santos Costa et al, 2008]
constraint logic programming
for Bayesian networks

\[ \phi_1(d) = P(d) \]
\[ \phi_2(su, d, sm) = P(su|d, sm) \]
\[ \phi_3(sm) = P(sm) \]

par-RV: unique skolem term

difficult(Course,Diff) :-
{ Diff = d(Course)
  with p([t,f],[0.6,0.4]) }
constraint logic programming for Bayesian networks

\[
\phi_1(d) = P(d) \quad \phi_2(su, d, sm) = P(su|d, sm) \quad \phi_3(sm) = P(sm)
\]

difficult(Course,\text{Diff}) :-
   \{ \text{Diff} = d(Course)
      with p([t,f],[0.6,0.4],[[]]) \}

smart(Student,\text{Smart}) :-
   \{ \text{Smart} = sm(Student)
      with p([t,f],[0.7,0.3],[[]]) \}
constraint logic programming for Bayesian networks

\[
\begin{align*}
\phi_1(d) &= P(d) \\
\phi_2(su, d, sm) &= P(su|d, sm) \\
\phi_3(sm) &= P(sm)
\end{align*}
\]

\texttt{difficult(Course, Diff) :-}
\{ Diff = d(Course)
  with p([t,f],[0.6,0.4],[[]]) \}

\texttt{smart(Student, Smart) :-}
\{ Smart = sm(Student)
  with p([t,f],[0.7,0.3],[[]]) \}

\texttt{success(Student, Course, Success) :-}
  takes(Student, Course),
  difficult(Course, D),
  smart(Student, Sm),
  \{ Success = su(Student, Course)
    with p([t,f],[0.85,0.1,0.98,0.45
    0.15,0.9,0.02,0.55],[D,Sm]) \}
constraint logic programming for Bayesian networks

difficult(Course,Diff) :-
{ Diff = d(Course)
  with p([t,f],[0.6,0.4],[[]]) }

smart(Student,Smart) :-
{ Smart = sm(Student)
  with p([t,f],[0.7,0.3],[[]]) }

success(Student,Course,Success) :-
takes(Student,Course),
difficult(Course,D),
smart(Student,Sm),
{ Success = su(Student,Course)
  with p([t,f],[0.85,0.1,0.98,0.45
           0.15,0.9,0.02,0.55],[D,Sm])]}

Prolog call constrains grounding
CLP(BN) [Santos Costa et al, 2008]

constraint logic programming for Bayesian networks

difficult(Course,Diff) :-
   { Diff = d(Course)
      with p([t,f],[0.6,0.4],[]) }  

smart(Student,Smart) :-
   { Smart = sm(Student)
      with p([t,f],[0.7,0.3],[])} 

success(Student,Course,Success) :-
   takes(Student,Course),
   difficult(Course,D),
   smart(Student,Sm),
   { Success = su(Student,Course)
     with p([t,f],[0.85,0.1,0.98,0.45
     0.15,0.9,0.02,0.55],[D,Sm])}
**CLP(BN)**

[Santos Costa et al, 2008]

**constraint logic programming for Bayesian networks**

\[ \phi_1(d) = P(d) \]

\[ \phi_2(su, d, sm) = P(su|d, sm) \]

\[ \phi_3(sm) = P(sm) \]

\[
\begin{align*}
difficult(Course, Diff) &:- \{ 
    Diff = d(Course) 
    \text{ with } p([t,f],[0.6,0.4],[[]]) 
\}

smart(Student, Smart) &:- \{ 
    Smart = sm(Student) 
    \text{ with } p([t,f],[0.7,0.3],[[]]) 
\}

success(Student, Course, Success) &:- 
    takes(Student, Course), 
    difficult(Course, D), 
    smart(Student, Sm), 
    \{ 
    Success = su(Student, Course) 
    \text{ with } p([t,f],[0.85,0.1,0.98,0.45,0.15,0.9,0.02,0.55],[D,Sm]) 
    \}
\]

\[ P(su|d=t,sm=t) \]
CLP(BN) \textsuperscript{[Santos Costa et al, 2008]}

constraint logic programming
for Bayesian networks

\texttt{difficult(Course,Diff) :-}
\begin{verbatim}
{ Diff = d(Course)
  with p([t,f],[0.6,0.4],[[]] ) }
\end{verbatim}

\texttt{smart(Student,Smart) :-}
\begin{verbatim}
{ Smart = sm(Student)
  with p([t,f],[0.7,0.3],[[]] ) }
\end{verbatim}

\texttt{success(Student,Course,Success) :-}
\begin{verbatim}
takes(Student,Course),
difficult(Course,D),
smart(Student,Sm),
{ Success = su(Student,Course)
  with p([t,f],[0.85,0.1,0.98,0.45
  0.15,0.9,0.02,0.55],[D,Sm])}
\end{verbatim}
CLP(BN) [Santos Costa et al, 2008] constraint logic programming for Bayesian networks

\[
\phi_1(d) = P(d)
\]
\[
\phi_2(su, d, sm) = P(su|d, sm)
\]
\[
\phi_3(sm) = P(sm)
\]

\text{difficult}(\text{Course}, \text{Diff}) :-
\{ \text{Diff} = d(\text{Course})
\text{ with } p([t,f],[0.6,0.4],[[]]) \}

\text{smart}(\text{Student}, \text{Smart}) :-
\{ \text{Smart} = sm(\text{Student})
\text{ with } p([t,f],[0.7,0.3],[[]]) \}

\text{success}(\text{Student}, \text{Course}, \text{Success}) :-
\text{takes}(\text{Student}, \text{Course}),
\text{difficult}(\text{Course}, D),
\text{smart}(\text{Student}, Sm),
\{ \text{Success} = su(\text{Student}, \text{Course})
\text{ with } p([t,f],[0.85,0.1,0.98,0.45, 0.15,0.9,0.02,0.55],[D,Sm]) \}

\text{takes}(s1,c1).
\text{takes}(s1,c2).
\text{takes}(s1,c3).
\text{takes}(s1,c4).
\text{takes}(s2,c1).
\text{takes}(s2,c2).
\text{takes}(s3,c2).
\text{takes}(s3,c4).
takes(s1,c1).  takes(s2,c1).
takes(s1,c2).  takes(s2,c2).
takes(s1,c3).  takes(s3,c2).
takes(s1,c4).  takes(s3,c4).
difficult(Course,Diff) :-
    { Diff = d(Course)
      with p([t,f],[0.6,0.4],[]) }

smart(Student,Smart) :-
    { Smart = sm(Student)
      with p([t,f],[0.7,0.3],[]) }

success(Student,Course,Success) :-
    takes(Student,Course),
    difficult(Course,D),
    smart(Student,Sm),
    { Success = su(Student,Course)
      with p([t,f],[0.85,0.1,0.98,0.45
                0.15,0.9,0.02,0.55],[D,Sm])}
difficult(Course, Diff) :-
    { Diff = d(Course)
      with p([t,f],[0.6,0.4],[]) } 
smart(Student, Smart) :-
    { Smart = sm(Student)
      with p([t,f],[0.7,0.3],[]) } 
success(Student, Course, Success) :-
    takes(Student, Course),
    difficult(Course, D),
    smart(Student, Sm),
    { Success = su(Student, Course)
      with p([t,f],[0.85,0.1,0.98,0.45
        0.15,0.9,0.02,0.55], [D,Sm])} 

0.6::difficult(C) :- takes(_,C).
0.7::smart(S) :- takes(S,_).
0.85::success(S,C) <- takes(S,C), difficult(C), smart(S).
0.10::success(S,C) <- takes(S,C), difficult(C), \+smart(S).
0.98::success(S,C) <- takes(S,C), \+difficult(C), smart(S).
0.45::success(S,C) <- takes(S,C), \+difficult(C), \+smart(S).
difficult(Course,Diff) :-
    { Diff = d(Course)
      with p([t,f],[0.6,0.4],[]) } 

smart(Student,Smart) :-
    { Smart = sm(Student)
      with p([t,f],[0.7,0.3],[]) } 

success(Student,Course,Success) :-
    takes(Student,Course),
    difficult(Course,D),
    smart(Student,Sm),
    { Success = su(Student,Course)
      with p([t,f],[0.85,0.1,0.98,0.45
      0.15,0.9,0.02,0.55],[D,Sm])} 

takes(s1,c1).
takes(s1,c2).
takes(s2,c1).
takes(s2,c2).
takes(s3,c2).
takes(s3,c4).
difficult(Course,Diff) :-
    { Diff = d(Course)
      with p([t,f],[0.6,0.4],[[]]) }
smart(Student,Smart) :-
    { Smart = sm(Student)
      with p([t,f],[0.7,0.3],[[]]) }
success(Student,Course,Success) :-
    takes(Student,Course),
    difficult(Course,D),
    smart(Student,Sm),
    { Success = su(Student,Course)
      with p([t,f],[0.85,0.1,0.98,0.45
                  0.15,0.9,0.02,0.55],[D,Sm]) }

?-success(s2,c1,S).
difficult(Course,Diff) :-
    { Diff = d(Course)
      with p([t,f],[0.6,0.4],[[]]) } 

smart(Student,Smart) :-
    { Smart = sm(Student)
      with p([t,f],[0.7,0.3],[[]]) } 

success(Student,Course,Success) :-
    takes(Student,Course),
    difficult(Course,D),
    smart(Student,Sm),
    { Success = su(Student,Course)
      with p([t,f],[0.85,0.1,0.98,0.45
                 0.15,0.9,0.02,0.55],[D,Sm])} 

?-success(s2,c1,S).
difficult(Course,Diff) :-
    { Diff = d(Course)
      with p([t,f],[0.6,0.4],[[]]) }

smart(Student,Smart) :-
    { Smart = sm(Student)
      with p([t,f],[0.7,0.3],[[]]) }

success(Student,Course,Success) :-
    takes(Student,Course),
    difficult(Course,D),
    smart(Student,Sm),
    { Success = su(Student,Course)
      with p([t,f],[0.85,0.1,0.98,0.45
                  0.15,0.9,0.02,0.55],[D,Sm]) }

?-success(s2,c1,S).

takes(s1,c1).
takes(s1,c2).
takes(s2,c1).
takes(s2,c2).
takes(s3,c2).
takes(s3,c4).
difficult(Course, Diff) :-
    { Diff = d(Course)
    with p([t,f],[0.6,0.4],[[]]) }

smart(Student, Smart) :-
    { Smart = sm(Student)
    with p([t,f],[0.7,0.3],[[]]) }

success(Student, Course, Success) :-
    takes(Student, Course),
    difficult(Course, D),
    smart(Student, Sm),
    { Success = su(Student, Course)
    with p([t,f],[0.85,0.1,0.98,0.45
    0.15,0.9,0.02,0.55],[D,Sm]) }

?-success(s2,c1,S).
difficult(Course, Diff) :-
   { Diff = d(Course)
     with p([t,f],[0.6,0.4],[[]] ) }

smart(Student, Smart) :-
   { Smart = sm(Student)
     with p([t,f],[0.7,0.3],[[]] ) }

success(Student, Course, Success) :-
   takes(Student, Course),
   difficult(Course, D),
   smart(Student, Sm),
   { Success = su(Student, Course)
     with p([t,f],[0.85,0.1,0.98,0.45
                  0.15,0.9,0.02,0.55],[D,Sm]) }

?-success(s2, c1, S).
difficult(Course,Diff) :-
    { Diff = d(Course)
        with p([t,f],[0.6,0.4],[[]]) };

smart(Student,Smart) :-
    { Smart = sm(Student)
        with p([t,f],[0.7,0.3],[[]]) };

success(Student,Course,Success) :-
    takes(Student,Course),
    difficult(Course,D),
    smart(Student,Sm),
    { Success = su(Student,Course)
        with p([t,f],[0.85,0.1,0.98,0.45
                    0.15,0.9,0.02,0.55],[D,Sm]) }

?-success(s2,c1,S).

\[\text{Sm}=\text{sm}(s2) \text{ with } p(...,[\[])\]
\[\text{D}=\text{d}(c1) \text{ with } p(...,[\[])\]
difficult(Course, Diff) :-
    { Diff = d(Course)
      with p([t,f],[0.6,0.4],[[]]) }  

smart(Student, Smart) :-
    { Smart = sm(Student)
      with p([t,f],[0.7,0.3],[[]]) }  

success(Student, Course, Success) :-
    takes(Student, Course),
    difficult(Course, D),
    smart(Student, Sm),
    { Success = su(Student, Course)
      with p([t,f],[0.85,0.1,0.98,0.45
        0.15,0.9,0.02,0.55],[D,Sm])}
difficult(Course, Diff) :-
    { Diff = d(Course)
      with p([t,f],[0.6,0.4],[[]]) }

smart(Student, Smart) :-
    { Smart = sm(Student)
      with p([t,f],[0.7,0.3],[[]]) }

success(Student, Course, Success) :-
    takes(Student, Course),
    difficult(Course, D),
    smart(Student, Sm),
    { Success = su(Student, Course)
      with p([t,f],[0.85,0.1,0.98,0.45
               0.15,0.9,0.02,0.55],[D,Sm]) }

?-success(s2, c1, S).

Success= su(s2, c1) with p(...,[D,Sm])
Sm= sm(s2) with p(...,[])
D= d(c1) with p(...,[[]])
difficult(Course,Diff) :-
    { Diff = d(Course)
      with p([t,f],[0.6,0.4],[]) }

smart(Student,Smart) :-
    { Smart = sm(Student)
      with p([t,f],[0.7,0.3],[]) }

success(Student,Course,Success) :-
    takes(Student,Course),
    difficult(Course,D),
    smart(Student,Sm),
    { Success = su(Student,Course)
      with p([t,f],[0.85,0.1,0.98,0.45
                 0.15,0.9,0.02,0.55],[D,Sm]) }

?-success(s2,c1,S).

Success= su(s2,c1) with p(...,[D,Sm])
    Sm= sm(s2) with p(...,[])
    D= d(c1) with p(...,[])

takes(s1,c1).
takes(s1,c2).
takes(s2,c1).
takes(s2,c2).
takes(s3,c2).
takes(s3,c4).
difficult(Course, Diff) :-
  { Diff = d(Course)
    with p([t,f],[0.6,0.4],[])}
smart(Student, Smart) :-
  { Smart = sm(Student)
    with p([t,f],[0.7,0.3],[])}
success(Student, Course, Success) :-
  takes(Student, Course),
  difficult(Course, D),
  smart(Student, Sm),
  { Success = su(Student, Course)
    with p([t,f],[0.85,0.1,0.98,0.45, 0.15,0.9,0.02,0.55],[D,Sm])}

?-success(s2,c1,S).

Success=su(s2,c1) with p(...,[D,Sm])
Sm=sm(s2) with p(...,[[]])
D=d(c1) with p(...,[[]])
CLP(BN) Summary

• Templating Bayesian networks via constraint logic programming

• Knowledge-based model construction (KBMC):
  • construct relevant ground BN by backward reasoning, adding constraints to constraint store
  • run any propositional inference technique
Lifted Graphical Models

Summary

- graphical model + relational templating language
- par-factor graphs as unifying framework
- aggregation / combining rules
- grounding vs lifted inference
- many different languages
  - CLP(BN): directed models via constraint LP
Advanced Topics

• parameter estimation
• complexity of querying
• lifted graphical models and KBMC
Roadmap

• Modeling
• Reasoning
• Language extensions
• Advanced topics

... with some detours on the way
A key question in AI:

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

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

Learning
- parameters
- structure

Statistical relational learning, probabilistic logic learning, probabilistic programming, ...
A key question in AI:

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

Reasoning with relational data
- logic
- databases
- programming
- parameters
- structure
- ... and much more to do!

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

Our answer: probabilistic logic programming
= probabilistic choices + logic program
- Many languages, systems, applications, ...
- ... and much more to do!
• PRISM http://sato-www.cs.titech.ac.jp/prism/
• ProbLog2 http://dtai.cs.kuleuven.be/problog/
• Yap Prolog http://www.dcc.fc.up.pt/~vsc/Yap/ includes
  • ProbLog1
  • cplint https://sites.google.com/a/unife.it/ml/cplint
  • CLP(BN)
  • LP2
• PITA in XSB Prolog http://xsb.sourceforge.net/
• AILog2 http://artint.info/code/ailog/ailog2.html
• SLPs http://stoics.org.uk/~nicos/sware/pepl
• contdist http://www.cs.sunysb.edu/~cram/contdist/
• DC https://code.google.com/p/distributional-clauses
• WFOMC http://dtai.cs.kuleuven.be/ml/systems/wfomc
References


