Probabilistic Programming Languages

Dagstuhl, 27.04.2015
Angelika Kimmig
angelika.kimmig@cs.kuleuven.be
probabilistic programming language
  =
  programming language
  +
  probabilistic primitives
  +
  inference methods
probabilistic programming language

= programming language + probabilistic primitives + inference methods

user-defined probabilistic models
general purpose reasoning tools
## Information Extraction

### Recently-Learned Facts

<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_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/)
Information Extraction

Recently-Learned Facts

<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
Information Extraction

Recently-Learned Facts

<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</td>
<td>827</td>
<td>02-apr-2014</td>
<td>93.8</td>
</tr>
<tr>
<td>rutgers_university</td>
<td></td>
<td></td>
<td></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>

instances for many different relations

degree of certainty

NELL: http://rtw.ml.cmu.edu/rtw/
Biomine network
Biomine network
Biomine Network
Notch receptor processing
BiologicalProcess
GO:GO:0007220

desensitization of receptor
BiologicalProcess
GO:GO:0007220

presenilin 2
Gene
EntrezGene:81751
Biomine Network

- participates_in 0.220

Gene

Biomolecular Process

- participates_in 0.220

Notch receptor processing BiologicalProcess GO:GO:0007220

Integral to nuclear inner CellularComponent GO:GO:0005639
Biomine Network

- 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, ...

Gene

0.220

BiologicalProcess

participates_in

participates_in

is_homologous_to

0.512

0.190

0.197

0.265

presenilin 2
Gene
EntrezGene:81751

Notch receptor processing
BiologicalProcess
GO:GO:0007220

integral to nuclear inner CellularComponent

0.186
**Word-based information representation**

[Cohen’s WHIRL]

<table>
<thead>
<tr>
<th>Cinema</th>
<th>Movie</th>
<th>Show Times</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roberts Theater Chatham</td>
<td>Brassed Off</td>
<td>7:15 - 9:10</td>
</tr>
<tr>
<td>Berkeley Cinema</td>
<td>Hercules</td>
<td>2:00 - 4:15 - 7:30</td>
</tr>
<tr>
<td>Sony Mountainside Theater</td>
<td>Men in Black</td>
<td>7:40 - 8:40 - 9:30 - 10:10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Movie</th>
<th>Review</th>
</tr>
</thead>
<tbody>
<tr>
<td>Men in Black, 1997</td>
<td>(***) One of the summer’s biggest hits, this ...</td>
</tr>
<tr>
<td>Face/Off, 1997</td>
<td>(**1/2) After a somewhat slow start, Cage and Travolta ...</td>
</tr>
<tr>
<td>Space Balls, 1987</td>
<td>(1/2) While not one of Mel Brooks’ better efforts, this Star Wars spoof ...</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

?- movieListing(Cinema, Movie1, Times)  
∧ review(Movie2, Review)  
∧ Movie1 ~ Movie2  
∧ Review ~ “comedy with space aliens”.

ranked retrieval +  
soft joins based on  
similarity
Molecular interaction networks

Can we find the mechanism connecting causes to effects?

[De Maeyer et al, Molecular Biosystems 13]
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]
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?
Travian: A massively multiplayer real-time strategy game

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

Dealing with uncertainty

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

Learning
Common theme

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

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

Learning
Common theme

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

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

Learning
• parameters
• structure
Common theme

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

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

Learning
- parameters
- structure

Statistical relational learning, probabilistic logic learning, probabilistic programming, probabilistic databases,...
Dealing with uncertainty
- probability theory
- graphical models

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

Learning
- parameters
- structure

Many different formalisms
- our focus: probabilistic logic programming

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

Dealing with uncertainty
Reasoning with relational data
Learning

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

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

---

**Prolog / logic programming**

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

**Dealing with uncertainty**

**Learning**

---

one world

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

---

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

ProbLog
probabilistic Prolog

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).

atoms as random variables

one world

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

Prolog / logic programming

several possible worlds

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

Ps/Macdonald stress

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

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

atoms as random variables

Learning

one world

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

one world

Learning

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).

stresses(ann).
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).

one world

Parameter learning, adapted relational learning techniques

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

random graph
random walk
**Random graph:**
which edges exist?

```
0.1  0.3
  0.5
0.7  0.4
```

**Random walk:**
which edge to take from a given node?

```
0.1  0.2
0.7
```
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

- continuous RVs
- decisions
- constraints
- time & dynamics
- semiring labels
- programming constructs
- ...
Roadmap

- Modeling (with detours to related work)
- Reasoning (and a bit of learning)
- Language extensions
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)
ProbLog by example:

A bit of gambling

• toss (biased) coin & draw ball from each urn

• win if (heads and a red ball) or (two balls of same color)

**probabilistic fact:** heads is true with probability 0.4 (and false with 0.6)
ProbLog by example:

A bit of gambling

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

0.4 :: heads. **annotated disjunction**: first ball is red with probability 0.3 and blue with 0.7

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

\[
\begin{align*}
0.4 & \:: \text{ heads.} \\
0.3 & \:: \text{ col}(1,\text{red}); \quad 0.7 & \:: \text{ col}(1,\text{blue}) . \\
0.2 & \:: \text{ col}(2,\text{red}); \quad 0.3 & \:: \text{ col}(2,\text{green}); \quad 0.5 & \:: \text{ col}(2,\text{blue}) .
\end{align*}
\]

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

A bit of gambling

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

0.4 :: heads.

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

win :- heads, col(_,red).
ProbLog by example:

A bit of gambling

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

0.4 :: heads.

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

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

logical rule encoding
background knowledge
ProbLog by example:

A bit of gambling

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

0.4 :: heads.

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

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

0.4 :: heads.

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

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

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

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

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

marginal probability

• Probability of \textit{win} query

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

• Most probable world where win is true?
Questions

0.4 :: heads.

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

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

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

• Probability of win?

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

• Most probable world where win is true?

0.4 :: heads.

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

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

marginal probability

conditional probability

MPE inference
Possible Worlds

0.4 :: heads.

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

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

0.4 :: heads.

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

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

0.4 :: heads.

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

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

0.4 :: heads.

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

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

0.4 \times 0.3
Possible Worlds

0.4 :: heads.

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

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

0.4 × 0.3 × 0.3
Possible Worlds

0.4 :: heads.

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

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

0.4 \times 0.3 \times 0.3
Possible Worlds

0.4 :: heads.

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

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

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

0.4 :: heads.

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

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

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

\[
0.4 \times 0.3 \times 0.3 = (1-0.4)\times0.3
\]
Possible Worlds

0.4 :: heads.

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

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

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

\[
\begin{align*}
0.4 \times 0.3 \times 0.3 & \quad (1-0.4) \times 0.3 \times 0.2 \\
\text{H} & \quad \text{R} & \quad \text{G} \\
\text{W} & \quad \text{R} & \quad \text{R}
\end{align*}
\]
Possible Worlds

0.4 :: heads.

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

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

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

0.4 :: heads.

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

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

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

\[
\begin{array}{c}
\text{H} \\
\text{R} \\
\text{G} \\
\text{W}
\end{array} \quad \begin{array}{c}
\text{R} \\
\text{R} \\
\text{W}
\end{array} \quad \text{blank}
\]
Possible Worlds

0.4 :: heads.

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

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

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

0.4 :: heads.

0.3 :: \text{col}(1,\text{red}); 0.7 :: \text{col}(1,\text{blue})

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

\text{win} :- \text{heads}, \text{col}(_,\text{red}).
\text{win} :- \text{col}(1,C), \text{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 \times 0.3 & \\
\end{align*}
Possible Worlds

0.4 :: heads.

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

\text{win} :- \text{heads}, \text{col}(_,\text{red}).
\text{win} :- \text{col}(1,C), \text{col}(2,C).
Possible Worlds

0.4 :: heads.

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

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

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

0.024
H R R
W

0.036
R R
W

0.056
H B R
W

0.084
B R

0.036
H R G
W

0.054
R G

0.084
H B G

0.126
B G

0.060
H R B
W

0.090
R B

0.140
H B B

0.210
B B
W
Most likely world where `win` is true?

<table>
<thead>
<tr>
<th>Probability</th>
<th>World 1</th>
<th>World 2</th>
<th>World 3</th>
<th>World 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.024</td>
<td>HRR</td>
<td>RBR</td>
<td>HB</td>
<td>BRB</td>
</tr>
<tr>
<td>0.036</td>
<td>HRG</td>
<td>RGR</td>
<td>HBG</td>
<td>BGB</td>
</tr>
<tr>
<td>0.056</td>
<td>RR</td>
<td>RB</td>
<td>BB</td>
<td>BB</td>
</tr>
<tr>
<td>0.084</td>
<td>B</td>
<td>R</td>
<td>G</td>
<td>G</td>
</tr>
</tbody>
</table>
Most likely world where \textit{win} is true?

MPE Inference
Most likely world where \texttt{col(2, blue)} is false?

- $H R R W$: 0.024
- $H R R W$: 0.036
- $H B R W$: 0.056
- $H B R W$: 0.084
- $H R G W$: 0.036
- $R G W$: 0.054
- $H B G W$: 0.084
- $B G W$: 0.126
- $H R B W$: 0.060
- $R B W$: 0.090
- $H B B W$: 0.140
- $B B W$: 0.210

\textit{MPE Inference}
Most likely world where \texttt{col(2,blue)} is false?

\begin{tabular}{cccc}
0.024 & 0.036 & 0.056 & 0.084 \\
\begin{tabular}{c}
H \quad R \quad R \\
W 
\end{tabular} & \begin{tabular}{c}
R \quad R \\
W 
\end{tabular} & \begin{tabular}{c}
H \quad B \quad R \\
W 
\end{tabular} & \begin{tabular}{c}
B \quad R \\
\end{tabular} \\
0.036 & 0.054 & 0.084 & 0.126 \\
\begin{tabular}{c}
H \quad R \quad G \\
W 
\end{tabular} & \begin{tabular}{c}
R \quad G \\
\end{tabular} & \begin{tabular}{c}
H \quad B \quad G \\
\end{tabular} & \begin{tabular}{c}
B \quad G \\
\end{tabular} \\
0.060 & 0.090 & 0.140 & 0.210 \\
\begin{tabular}{c}
H \quad R \quad B \\
W 
\end{tabular} & \begin{tabular}{c}
R \quad B \\
\end{tabular} & \begin{tabular}{c}
H \quad B \quad B \\
W 
\end{tabular} & \begin{tabular}{c}
B \quad B \\
W 
\end{tabular} \\
\end{tabular}
P(win) = ?

Marginal Probability
\[ P(\text{win}) = \sum \]

1. \(0.024\)  
   \[
   \begin{array}{c}
   H \quad R \quad R \\
   W
   \end{array}
   \]

2. \(0.036\)  
   \[
   \begin{array}{c}
   H \quad R \quad R \\
   W
   \end{array}
   \]

3. \(0.056\)  
   \[
   \begin{array}{c}
   H \quad B \quad R \\
   W
   \end{array}
   \]

4. \(0.084\)  
   \[
   \begin{array}{c}
   B \quad R
   \end{array}
   \]

5. \(0.036\)  
   \[
   \begin{array}{c}
   H \quad R \quad G \\
   W
   \end{array}
   \]

6. \(0.054\)  
   \[
   \begin{array}{c}
   R \quad G
   \end{array}
   \]

7. \(0.084\)  
   \[
   \begin{array}{c}
   H \quad B \quad G \\
   \end{array}
   \]

8. \(0.126\)  
   \[
   \begin{array}{c}
   B \quad G
   \end{array}
   \]

9. \(0.060\)  
   \[
   \begin{array}{c}
   H \quad R \quad B \\
   W
   \end{array}
   \]

10. \(0.090\)  
    \[
    \begin{array}{c}
    R \quad B
    \end{array}
    \]

11. \(0.140\)  
    \[
    \begin{array}{c}
    H \quad B \quad B \\
    \end{array}
    \]

12. \(0.210\)  
    \[
    \begin{array}{c}
    B \quad B
    \end{array}
    \]
\[ P(\text{win}) = \sum = 0.562 \]
P(win|col(2, green)) = ?

Conditional Probability

\[
\begin{align*}
0.024 & \quad H & R & R & \quad W \\
0.036 & \quad R & R & & \quad W \\
0.056 & \quad H & B & R & \quad W \\
0.084 & \quad B & R & & \\
0.036 & \quad H & R & G & \quad W \\
0.054 & \quad R & G & & \\
0.084 & \quad H & B & G & \\
0.126 & \quad B & G & \\
0.060 & \quad H & R & B & \quad W \\
0.090 & \quad R & B & & \\
0.140 & \quad H & B & B & \quad W \\
0.210 & \quad B & B & \\
\end{align*}
\]
\[ P(\text{win}|\text{col}(2,\text{green})) = \frac{\sum}{\sum} = P(\text{win} \land \text{col}(2,\text{green}))/P(\text{col}(2,\text{green})) \]

<table>
<thead>
<tr>
<th>Event</th>
<th>Probability</th>
<th>outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>HHH</td>
<td>0.024</td>
<td>W</td>
</tr>
<tr>
<td>HHR</td>
<td>0.036</td>
<td>W</td>
</tr>
<tr>
<td>HHR</td>
<td>0.056</td>
<td>W</td>
</tr>
<tr>
<td>HBB</td>
<td>0.084</td>
<td>W</td>
</tr>
<tr>
<td>HHR</td>
<td>0.036</td>
<td>W</td>
</tr>
<tr>
<td>HRR</td>
<td>0.054</td>
<td>W</td>
</tr>
<tr>
<td>HRR</td>
<td>0.084</td>
<td>W</td>
</tr>
<tr>
<td>HBB</td>
<td>0.126</td>
<td>W</td>
</tr>
<tr>
<td>HRR</td>
<td>0.060</td>
<td>W</td>
</tr>
<tr>
<td>HRR</td>
<td>0.090</td>
<td>W</td>
</tr>
<tr>
<td>HBB</td>
<td>0.140</td>
<td>W</td>
</tr>
<tr>
<td>HRR</td>
<td>0.210</td>
<td>W</td>
</tr>
</tbody>
</table>

Conditional Probability
\[ P(\text{win}|\text{col}(2,\text{green})) = \frac{\sum}{\sum} \]

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

Conditional Probability

![Conditional Probability Diagram](image-url)
P(win|col(2, green)) = \frac{\sum}{\sum} = \frac{0.036}{0.3} = 0.12
Distribution Semantics
(with probabilistic facts)

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

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

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

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

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

query

subset of probabilistic facts

Prolog rules

[Sato, ICLP 95]
**Distribution Semantics**

(with probabilistic facts)

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

*query*

*subset of probabilistic facts*

*sum over possible worlds where Q is true*

*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

sum over possible worlds where Q is true

subset of probabilistic facts

Prolog rules

probability of possible world

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

\[ \text{throws(john).} \]
\[ 0.5 :: \text{throws(mary)}. \]
\[ 0.8 :: \text{break} \leftarrow \text{throws(mary)}. \]
\[ 0.6 :: \text{break} \leftarrow \text{throws(john)}. \]

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

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

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

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

\[
\text{length(Obj) } \sim \text{ gaussian}(6.0, 0.45) :- \text{ type(Obj, glass)}. \\
\text{stackable(OBot, OTop) :-}
\begin{align*}
\approx \text{length(OBot)} & \geq \approx \text{length(OTop)}, \\
\approx \text{width(OBot)} & \geq \approx \text{width(OTop)}. 
\end{align*}
\]

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

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

\[
\text{stackable}(\text{OBot}, \text{OTop}) :\!
\begin{align*}
\approx & \text{length}(\text{OBot}) \geq \approx \text{length}(\text{OTop}), \\
\approx & \text{width}(\text{OBot}) \geq \approx \text{width}(\text{OTop}).
\end{align*}
\]

\[
\text{ontype}(\text{Obj}, \text{plate}) \sim \text{finite}([0 : \text{glass}, 0.0024 : \text{cup}, \\
0 : \text{pitcher}, 0.8676 : \text{plate}, \\
0.0284 : \text{bowl}, 0 : \text{serving}, \\
0.1016 : \text{none}])
\begin{align*}
:\!-\! \text{obj}(\text{Obj}), \text{on}(\text{Obj}, \text{O2}), \text{type}(\text{O2}, \text{plate}).
\end{align*}
\]

random variable with discrete distribution
Distributional Clauses (DC)

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

length(Obj) \sim \text{gaussian}(6.0, 0.45) :- \text{type}(Obj, \text{glass}).
stackable(\text{OBot}, \text{OTop}) :-
\begin{align*}
\approx & \text{length}(\text{OBot}) \geq \approx \text{length}(\text{OTop}), \\
\approx & \text{width}(\text{OBot}) \geq \approx \text{width}(\text{OTop}).
\end{align*}
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}])
\begin{align*}
:- & \text{obj}(\text{Obj}), \text{on}(\text{Obj}, \text{O2}), \text{type}(\text{O2}, \text{plate}).
\end{align*}
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:

- [H R R W] with probability 0.024
- [H B B W] with probability 0.140
- [H R B W] with probability 0.036
- [H B B W] with probability 0.126
- [H B B W] with probability 0.210

Infeasible state
Answering Questions

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

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

![Diagram](image)

Logical reasoning

Data structure

Probabilistic inference
Answering Questions

Given:
- program
- queries
- evidence

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

more on this later
Probabilistic Databases

Dealing with uncertainty

Reasoning with relational data

Learning

[Suciu et al 2011]
Probabilistic Databases

Dealing with uncertainty

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

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

Dealing with uncertainty

relational database

Learning
Probabilistic Databases

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>

Dealing with uncertainty

Learning

relational database

Suciu et al 2011
Probabilistic Databases

<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>

\[
\text{select x.person, y.country from bornIn x, cityIn y where x.city=y.city}
\]

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

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

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>

tuples as random variables

Learning

relational database

[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>

probabilistic tables + database queries → distribution over possible worlds

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

one world

ewise

<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>

bornIn
cityIn

<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>

tuples as random variables

relational database

Learning [Suciu et al 2011]
Example: Information Extraction

Recently-Learned Facts

<table>
<thead>
<tr>
<th>instance</th>
<th>iteration</th>
<th>date learned</th>
<th>confidence</th>
<th>ticks</th>
</tr>
</thead>
<tbody>
<tr>
<td>kelly_andrews is a female</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>investment_next_year is an economic sector</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>shibenik is a geopolitical entity that is an organization</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>quality_web_design_work is a character trait</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mercedes_benz_cls_by_carlsson is an automobile manufacturer</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>social_work is an academic program at the university rutgers_university</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dante wrote the book the_divine_comedy</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>willie_aames was born in the city los_angeles</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>kitt_peak is a mountain in the state or province arizona</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>greenwich is a park in the city london</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

instances for many different relations

degree of certainty

NELL: http://rtw.ml.cmu.edu/rtw/
### Querying: relational database

<table>
<thead>
<tr>
<th>Company</th>
<th>Product</th>
<th>Company</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>sony</td>
<td>walkman</td>
<td>microsoft</td>
<td>redmond</td>
</tr>
<tr>
<td>microsoft</td>
<td>mac_os_x</td>
<td>ibm</td>
<td>san_jose</td>
</tr>
<tr>
<td>ibm</td>
<td>personal_computer</td>
<td>emirates_airlines</td>
<td>dubai</td>
</tr>
<tr>
<td>microsoft</td>
<td>mac_os</td>
<td>honda</td>
<td>torrance</td>
</tr>
<tr>
<td>adobe</td>
<td>adobe_indesign</td>
<td>egyptair</td>
<td>seattle</td>
</tr>
<tr>
<td>adobe</td>
<td>adobe_dreamweaver</td>
<td>adobe</td>
<td>cairo</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
### 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_indeisgn</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'
```
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: 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>

```
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: probabilistic db

**Example from Suciu et al 2011**

<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'

**same query** - probabilities handled implicitly
Querying: probabilistic db

select x.Product, x.Company
from ProducesProduct x, HeadquarteredIn y
where x.Company=y.Company and
y.City=‘san_jose’

0.96x0.99=0.95

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

### ProducesProduct

<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>

### HeadquarteredIn

<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'
```

0.9x0.93=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>
</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>
</tbody>
</table>

**Example from Suciu et al 2011**

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

```
<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>
```

0.87x0.93=0.80
Querying: probabilistic db

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

[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>

<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>

```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]
PDB with tuple-level uncertainty in ProbLog?

\[\begin{align*}
0.96 &:: \text{producesProduct}(\text{sony,walkman}). \\
0.96 &:: \text{producesProduct}(\text{microsoft,mac\_os\_x}). \\
0.96 &:: \text{producesProduct}(\text{ibm,personal\_computer}). \\
0.9 &:: \text{producesProduct}(\text{microsoft,mac\_os}). \\
0.9 &:: \text{producesProduct}(\text{adobe,adobe\_indesign}). \\
0.87 &:: \text{producesProduct}(\text{adobe,adobe\_dreamweaver}). \\
&\ldots \\
1.00 &:: \text{headquarteredIn}(\text{microsoft,redmond}). \\
0.99 &:: \text{headquarteredIn}(\text{ibm,san\_jose}). \\
0.93 &:: \text{headquarteredIn}(\text{emirates\_airlines,dubai}). \\
0.93 &:: \text{headquarteredIn}(\text{honda,torrance}). \\
0.93 &:: \text{headquarteredIn}(\text{horizon,seattle}). \\
0.93 &:: \text{headquarteredIn}(\text{egyptair,cairo}). \\
0.93 &:: \text{headquarteredIn}(\text{adobe,san\_jose}). \\
&\ldots \\
\text{result}(\text{Product},\text{Company}) &:: \text{producesProduct}(\text{Company,Product}), \\
&\text{headquarteredIn}(\text{Company,san\_jose}). \\
\text{query}(&\text{result}(\_\_))\).
ProbLog by example:

Rain or sun?
ProbLog by example: 

Rain or sun?

\[ 0.5 :: \text{weather(sun,0)} ; 0.5 :: \text{weather(rain,0)}. \]
ProbLog by example:

Rain or sun?

0.5::weather(sun,0) ; 0.5::weather(rain,0).

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).

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

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

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).
friend(2,1).
friend(2,4).
friend(3,4).
friend(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).
    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).

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**

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

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

**flexible probability:**
computed from the weight of the item

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

list of all 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([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).

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

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).

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).

do **not** pack first item,
continue with rest of 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).

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
  \[\text{person(bob)}.\]

• probabilistic facts
  \[\text{0.5::stress(bob)}.\]

• annotated disjunctions

  \[\text{0.5::stress(X) :- person(X)}.\]

  \[\text{0.4::a(X); 0.3::b(X); 0.2::c(X); 0.1::d(X) :- q(X)}.\]

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

• flexible probabilities

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

• Prolog clauses

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

  \[\text{excess([I|R],Limit) :- \text{+pack(I)}, excess(R,Limit)}.\]
let’s query those...

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

**Friends & smokers**

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

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

example possible world

\[\text{friend}(1,2).\]
\[\text{friend}(3,4).\]
\[\text{friend}(4,2).\]
\[\text{stress}(1).\]
\[\text{stress}(2).\]
\[\text{stress}(3).\]
\[\text{smokes}(1).\]
\[\text{smokes}(2).\]
\[\text{smokes}(3).\]
\[\text{smokes}(4).\]

- several instances of \text{smokes}(X) in same world
- \text{smokes}(2): multiple derivations in same world
- distribution over worlds not (always) a distribution over computations / answers
ProbLog by example:

Rain or sun?

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

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

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

\[ P_1 = 0.12 \]

\[ P_2 = 0.16 \]

\[ P_3 = 0.04 \]

\[ P_4 = 0.32 \]
Mutually Exclusive Rules:
no two rules apply simultaneously

0.5::weather(sun,0) ; 0.5::weather(rain,0).

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

0.5::weather(sun,0) ; 0.5::weather(rain,0).

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).

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).

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 without condition
• switch gives fresh result on each call
• Prolog rules
• limited support for negation (compiling away)
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).


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).
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).
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).
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).

random variables and their values
probability distributions

set \textbf{W0} to random value of \textbf{init}

set \textbf{WNext} to random
value of \textbf{tr}(\textbf{WNow}), using
\textbf{fresh} value on every call
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]).

0.5::init(sun); 0.5::init(rain) <- true.
0.6::tr(T,sun,sun); 0.4::tr(T,sun,rain) <- true.
0.2::tr(T,rain,sun); 0.8::tr(T,rain,rain) <- true.

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).

wind現在関数を説明するproblog

ProbLog needs to explicitly use different facts at each call
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]

Dealing with uncertainty

functional programming

Learning

(define plus5 (lambda (x) (+ x 5)))

(map plus5 '(1 2 3))
Church
probabilistic functional programming

[Goodman et al, UAI 08]

Dealing with uncertainty

Functional programming

Learning

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]

(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))

one execution

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

(random primitives)
(Learning)

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

(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))
Church
probabilistic functional programming
[Goodman et al, UAI 08]

(probablistic primitives + functional program) → distribution over possible executions

several possible executions

(one execution)

(random primitives)

Learning

(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))

http://probmods.org
Church vs ProbLog

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

(map randplus5 '(1 2))
Church vs ProbLog

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

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

(map randplus5 '(1 2))

Church 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

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],_)).
Church vs ProbLog

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

(map randplus5 '(1 2))

Church result: (1 2) with 0.4\times0.4
(1 7) with 0.4\times0.6
(6 2) with 0.6\times0.4
(6 7) with 0.6\times0.6

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).

ProbLog result: (1 2) with 0.4\times0.4
(1 7) with 0.4\times0.6
(6 2) with 0.6\times0.4
(6 7) with 0.6\times0.6

query(lp5([1,2],_)).
results for \([1,1]\)?

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

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

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

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

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

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

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

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

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

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

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

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([1,1],_)).
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\times0.4

(1 6) with 0.4\times0.6

(6 1) with 0.6\times0.4

(6 6) with 0.6\times0.6

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).

ProbLog result: (1 1) with 0.4

(1 6) with 0.0

(6 1) with 0.0

(6 6) with 0.6

stochastic memoization
Solution

(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],_)).
Solution

(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],_)).
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
<table>
<thead>
<tr>
<th>ProbLog</th>
<th>PRISM</th>
<th>Church</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 (with detours to related work)
- Reasoning (and a bit of learning)
- Language extensions
Reasoning

• Exact inference with knowledge compilation
  • using proofs
  • using models
  • under additional assumptions
• 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 worlds

infeasible
Answering Questions

Given:
- program
- queries
- evidence

Find:
- marginal probabilities
- conditional probabilities
- MPE state

logical reasoning
data structure
probabilistic inference
Answering Questions

Given:
- program
- queries
- evidence

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

Given:
- program
- queries
- evidence

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

1. using proofs
2. using models

knowledge compilation
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).
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).
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(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).

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

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

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

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

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

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

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

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

?- smokes(bob).
?- 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

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

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

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

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).
?- influences(Y2,ann),smokes(Y2).

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

\begin{align*}
0.8 &:: \text{stress(ann)}.
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{influences(bob,carl)} &\land \text{influences(ann,bob)} &\land \text{stress(ann)}
\
\text{probability of proof} &= 0.2 \times 0.6 \times 0.8 = 0.096
\end{align*}

\begin{align*}
\text{influences(Y,carl)} &\land \text{smokes(Y)}.
\text{Y=bob}
\
\text{influences(Y1,bob)} &\land \text{smokes(Y1)}.
\text{Y1=ann}
\
\text{influences(Y2,ann)} &\land \text{smokes(Y2)}.
\end{align*}
Proofs in ProbLog

0.8::stress(ann).
0.4::stress(bob).
0.6::influences(ann,bob).
0.2::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).
?- influences(Y2,ann),smokes(Y2).

\[
0.2 \times 0.6 \times 0.8 = 0.096
\]
Proofs in ProbLog

?- smokes(carl).

?- stress(carl).

Y=bob

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

?- smokes(bob).

Y1=ann

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

?- smokes(ann).

?- stress(ann).

?- influences(Y2,ann),smokes(Y2).

\[
0.2 \times 0.6 \times 0.8 = 0.096
\]

0.8::stress(ann).

0.4::stress(bob).

0.6::influences(ann,bob).

0.2::influences(bob,carl).

smokes(X) :- stress(X).

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

influences(bob,carl) & influences(ann,bob) & stress(ann)
Proofs in ProbLog

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

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

\[
\begin{align*}
\text{smokes}(\text{carl}). \\
\text{smokes}(\text{bob}). \\
\text{smokes}(\text{ann}). \\
\end{align*}
\]

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

Diagram:

- \text{influences(bob,carl)}
- \text{stress(bob)} 0.2 \times 0.4 = 0.08
- \text{influences(ann,bob)} 0.2 \times 0.6 \times 0.8 = 0.096
Proofs in ProbLog

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

\[
\begin{align*}
\text{smokes(X)} & :- \text{stress(X).} \\
\text{smokes(X)} & :- \\
& \quad \text{influences(Y,X), smokes(Y).} \\
\end{align*}
\]

- \( Y = \text{bob} \)

- \( Y_1 = \text{ann} \)

\[
\begin{align*}
0.2 \times 0.4 & = 0.08 \\
0.2 \times 0.6 \times 0.8 & = 0.096
\end{align*}
\]

Proofs overlap! cannot sum probabilities (disjoint-sum-problem)
Disjoint-Sum-Problem

possible worlds

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

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

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

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

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

...
Disjoint-Sum-Problem

possible worlds

\[
\begin{align*}
\text{infl(bob, carl)} & \land \text{infl(ann, bob)} \land \text{st(ann)} \land \neg \text{st(bob)} \\
\text{infl(bob, carl)} & \land \text{infl(ann, bob)} \land \text{st(ann)} \land \text{st(bob)} \\
\text{infl(bob, carl)} & \land \neg \text{infl(ann, bob)} \land \text{st(ann)} \land \text{st(bob)} \\
\text{infl(bob, carl)} & \land \text{infl(ann, bob)} \land \neg \text{st(ann)} \land \text{st(bob)} \\
\text{infl(bob, carl)} & \land \neg \text{infl(ann, bob)} \land \neg \text{st(ann)} \land \text{st(bob)} \\
\text{infl(bob, carl)} & \land \neg \text{infl(ann, bob)} \land \neg \text{st(ann)} \land \neg \text{st(bob)}
\end{align*}
\]

...
Disjoint-Sum-Problem

possible worlds

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

\[
\begin{align*}
\text{infl}(\text{bob}, \text{carl}) & \land \text{infl}(\text{ann}, \text{bob}) & \land \text{st}(\text{ann}) & \land \neg \text{st}(\text{bob}) \\
\text{infl}(\text{bob}, \text{carl}) & \land \text{infl}(\text{ann}, \text{bob}) & \land \text{st}(\text{ann}) & \land \text{st}(\text{bob}) \\
\text{infl}(\text{bob}, \text{carl}) & \land \neg \text{infl}(\text{ann}, \text{bob}) & \land \text{st}(\text{ann}) & \land \text{st}(\text{bob}) \\
\text{infl}(\text{bob}, \text{carl}) & \land \text{infl}(\text{ann}, \text{bob}) & \land \neg \text{st}(\text{ann}) & \land \text{st}(\text{bob}) \\
\text{infl}(\text{bob}, \text{carl}) & \land \neg \text{infl}(\text{ann}, \text{bob}) & \land \neg \text{st}(\text{ann}) & \land \text{st}(\text{bob}) \\
\text{infl}(\text{bob}, \text{carl}) & \land \neg \text{infl}(\text{ann}, \text{bob}) & \land \text{st}(\text{ann}) & \land \text{st}(\text{bob}) \\
\end{align*}
\]

... \[ \text{influences}(\text{bob}, \text{carl}) \land \text{stress}(\text{bob}) \]
Disjoint-Sum-Problem

possible worlds

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

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

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

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

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

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

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

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

possible worlds

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

\[
\begin{align*}
infl(bob, carl) \& infl(ann, bob) \& st(ann) \& \text{\textbackslash st(bob)} & \quad 0.0576 \\
infl(bob, carl) \& infl(ann, bob) \& st(ann) \& st(bob) & \quad 0.0384 \\
infl(bob, carl) \& \text{\textbackslash infl(ann, bob)} \& st(ann) \& st(bob) & \quad 0.0256 \\
infl(bob, carl) \& infl(ann, bob) \& \text{\textbackslash st(ann)} \& st(bob) & \quad 0.0096 \\
infl(bob, carl) \& \text{\textbackslash infl(ann, bob)} \& \text{\textbackslash st(ann)} \& st(bob) & \quad 0.0064 \\
\end{align*}
\]

\[ \sum = 0.1376 \]

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

possible worlds

solution: knowledge compilation

\[
\begin{align*}
\text{infl}(\text{bob, carl}) & \land \text{infl}(\text{ann, bob}) & \land \text{st}(\text{ann}) & \land \lnot\text{st}(\text{bob}) \\
\text{infl}(\text{bob, carl}) & \land \text{infl}(\text{ann, bob}) & \land \text{st}(\text{ann}) & \land \text{st}(\text{bob}) \\
\text{infl}(\text{bob, carl}) & \land \lnot\text{infl}(\text{ann, bob}) & \land \text{st}(\text{ann}) & \land \text{st}(\text{bob}) \\
\text{infl}(\text{bob, carl}) & \land \text{infl}(\text{ann, bob}) & \land \lnot\text{st}(\text{ann}) & \land \text{st}(\text{bob}) \\
\text{infl}(\text{bob, carl}) & \land \lnot\text{infl}(\text{ann, bob}) & \land \lnot\text{st}(\text{ann}) & \land \text{st}(\text{bob}) \\
\ldots & \text{influences}(\text{bob, carl}) & \land \text{stress}(\text{bob}) \\
\end{align*}
\]

\[\sum = 0.1376\]

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
  $\rightarrow$ efficient probability computation
Binary Decision Diagrams [Bryant 86]

\[ X \lor Y \lor Z \]
**Binary Decision Diagrams**  
[Bryant 86]  

$$X \lor Y \lor Z$$

<table>
<thead>
<tr>
<th>X</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>
</tr>
</thead>
<tbody>
<tr>
<td>Y</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>
</tr>
<tr>
<td>Z</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>
Binary Decision Diagrams

\[ X \lor Y \lor Z \]

\begin{align*}
X & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
Y & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
Z & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1
\end{align*}
Binary Decision Diagrams [Bryant 86]

\[ X \lor Y \lor Z \]

\[
\begin{array}{cccccccc}
0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]
**Binary Decision Diagrams**

[Bryant 86]

\[ X \lor Y \lor Z \]

\[
\begin{array}{cccccccccccc}
X & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
Y & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
Z & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\
\end{array}
\]
Binary Decision Diagrams

$X \lor Y \lor Z$

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

[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>
</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>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>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

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

64
Binary Decision Diagrams

$X \lor (\neg X \land Y) \lor (\neg X \land \neg Y \land Z)$

$X \lor Y \lor Z$

<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>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>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></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></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[242x166]X   0   0   0   0   1   1   1   1
[246x117]Y   0   0   1   1   0   0   1   1
[244x68]Z   0   1   0   1   0   1   0   1

[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>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>
</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>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>1</td>
<td></td>
</tr>
</tbody>
</table>
Binary Decision Diagrams [Bryant 86]
Binary Decision Diagrams

[Bryant 86]

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

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

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

[Bryant 86]
Binary Decision Diagrams

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

influences(bob, carl)?
Binary Decision Diagrams

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

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

Diagram:

- Node i(b,c) with yes and no branches
- Node s(b) with yes and no branches
- Node s(a) with yes and no branches

Root nodes 0 and 1
Binary Decision Diagrams

- \( i(b,c) \)
  - yes: \( i(a,b) \)
  - no: \( s(b) \)
    - no: \( s(a) \)
    - yes: \( \text{influences(bob, carl) ?} \)
  - no: \( \text{stress(bob) ?} \)
    - yes: \( \text{influences(ann, bob) ?} \)
    - no: \( \text{stress(ann) ?} \)

- 0
- 1
Binary Decision Diagrams

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

\[
\begin{align*}
\text{influences}(\text{bob}, \text{carl})? & \quad \text{stress}(\text{bob})? \\
\text{influences}(\text{ann}, \text{bob})? & \quad \text{stress}(\text{ann})?
\end{align*}
\]
Binary Decision Diagrams

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

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

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

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

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

\[
\begin{array}{c}
\text{influences}(\text{bob}, \text{carl})? \\
\text{stress}(\text{bob})? \\
\text{influences}(\text{ann}, \text{bob})? \\
\text{stress}(\text{ann})?
\end{array}
\]
Binary Decision Diagrams

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

Influences (bob, carl)?

Influences (ann, bob)?

Stress (bob)?

Stress (ann)?

Probability of smokes (c)?

0.2 \times 0.0 + 0.8 \times 1.0 = 0.8
**Binary Decision Diagrams**

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

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

\[
\begin{align*}
\text{probability of } \text{smokes}(c) & = 0.0 \times 0.0 + 0.8 \times 1.0 = 0.8 \\
0.6 \times 0.48 + 0.4 \times 1.0 & = 0.688 \\
0.4 \times 0.0 + 0.6 \times 0.8 & = 0.48 \\
0.2 \times 0.0 + 0.8 \times 1.0 & = 0.8
\end{align*}
\]
Binary Decision Diagrams

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

\[
\begin{align*}
\text{smokes}(c) &= i(b,c) \& s(b) \lor i(b,c) \& i(a,b) \& s(a) \\
0.8 \times 0.0 + 0.2 \times 0.688 &= 0.1376 \\
0.6 \times 0.48 + 0.4 \times 1.0 &= 0.688 \\
0.4 \times 0.0 + 0.6 \times 0.8 &= 0.48 \\
0.2 \times 0.0 + 0.8 \times 1.0 &= 0.8 \\
\end{align*}
\]
Initial Approach
(ProbLog1 & others)

Find all proofs of query

Binary Decision Diagram (BDD)

calculate marginal by
dynamic programming

\[ P(\text{win}) = \text{probability of reaching 1-leaf} \]

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

[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
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).
```

- Forward reasoning to construct unique model:
Logical Reasoning: Models in Prolog

- Forward reasoning to construct unique model:
  - Start with database 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).

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).
```
Logical Reasoning: Models in Prolog

• Forward reasoning to construct unique model:

• Start with database facts

• Use rules to add more facts

?– smokes(carl).

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

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

models in 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**
  - **ProbLog:** each possible world is a model, probability of query is sum over models where query is true
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

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

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

```prolog
?- smokes(carl).
```

→ weighted model counting
Weighted Model Counting

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

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) $\phi$ can be counted using the weighted model counting formula:

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

This counts the interpretations (truth value assignments) of the propositional variables.
Weighted Model Counting

A propositional formula in conjunctive normal form (CNF) $\phi$ is evaluated under interpretations (truth value assignments) of its propositional variables.

The weighted model count $WMC(\phi)$ is given by:

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

- $I_V$ is the set of propositional variables.
- $w(l)$ is the weight of literal $l$.

This formula counts the number of interpretations $I_V$ that satisfy $\phi$ and multiplies the weights of the literals under these interpretations.
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.
Weighted Model Counting

propositional formula in conjunctive normal form (CNF) given by 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
Weighted Model Counting

A propositional formula in conjunctive normal form (CNF) given by 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.
  - For \( p :: f \), \( w(f) = p \)
  - \( w(\text{not } f) = 1 - p \)

Possible worlds
Weighted Model Counting

**Formulas:**

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

**Definition:**

Weighted Model Counting (WMC) of formula \( \phi \)

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

**Interpretations:**

Truth value assignments of propositional variables

**Possible worlds**

**Weights:**

- For \( p :: f \), \( w(f) = p \)
- For \( \neg f \), \( w(\neg f) = 1 - p \)
?- smokes(carl).

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

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

?- smokes(carl).

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

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

• Find relevant ground rules by backward reasoning
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).
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
  
  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

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

    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) ↔ (i(b,c) ∧ sm(b))
    ∧  sm(b) ↔ (st(b) ∨ (i(a,b) ∧ sm(a)))
    ∧  sm(a) ↔ st(a)
ProbLog → CNF

?- smokes(carl).

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

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

• 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

  sm(c) ↔ (i(b,c) ∧ sm(b))
  ∧ sm(b) ↔ (st(b) ∨ (i(a,b) ∧ sm(a)))
  ∧ sm(a) ↔ st(a)

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

Find relevant ground program for queries & evidence

Weighted CNF

use weighted model counting / satisfiability

Win

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

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

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

\begin{align*}
h(1) &\rightarrow 0.4 & h(2) &\rightarrow 0.7 & h(3) &\rightarrow 0.5 \\
\neg h(1) &\rightarrow 0.6 & \neg h(2) &\rightarrow 0.3 & \neg h(3) &\rightarrow 0.5
\end{align*}

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

\[ \text{alarm} \iff \text{burglary} \vee \text{earthquake} \]
\[ \text{calls(john)} \iff \text{alarm}, \text{hears_alarm(john)} \]
\[ \text{calls(john)} \]

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

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
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,
    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,
  msw(init,W0),
  w(0,Time,W0,W).

w(T,T,W,W).

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

weather(s,2) <=> msw(init,s) & w(0,2,s,s) 
  v msw(init,r) & w(0,2,r,s)
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(s,2) <=> msw(init,s) & w(0,2,s,s)
    v msw(init,r) & w(0,2,r,s)

w(0,2,s,s) <=> msw(tr(s),s) & w(1,2,s,s)
    v msw(tr(s),r) & w(1,2,r,s)
\[
\text{weather}(W, \text{Time}) \leftarrow \\
\text{Time} \geq 0, \\
\text{msw}(\text{init}, W_0), \\
\text{w}(0, \text{Time}, W_0, W).
\]

\[
\text{w}(T, T, W, W).
\]

\[
\text{w}(\text{Now}, T, W_{\text{Now}}, W_T) \leftarrow \\
\text{Now} < T, \\
\text{msw}(\text{tr}(W_{\text{Now}}), W_{\text{Next}}), \\
\text{Next} \text{ is} \text{ Now} + 1, \\
\text{w}(\text{Next}, T, W_{\text{Next}}, W_T).
\]

\[
\text{weather}(s, 2) \iff \text{msw}(\text{init}, s) \& w(0, 2, s, s) \\
\lor \text{msw}(\text{init}, r) \& w(0, 2, r, s)
\]

\[
w(0, 2, s, s) \iff \text{msw}(\text{tr}(s), s) \& w(1, 2, s, s) \\
\lor \text{msw}(\text{tr}(s), r) \& w(1, 2, r, s)
\]

\[
w(0, 2, r, s) \iff \text{msw}(\text{tr}(r), s) \& w(1, 2, r, s) \\
\lor \text{msw}(\text{tr}(r), r) \& w(1, 2, r, s)
\]
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(s,2) <=> msw(init,s) & w(0,2,s,s)
    v msw(init,r) & w(0,2,r,s)

w(0,2,s,s) <=> msw(tr(s),s) & w(1,2,s,s)
    v msw(tr(s),r) & w(1,2,r,s)

w(0,2,r,s) <=> msw(tr(r),s) & w(1,2,r,s)
    v msw(tr(r),r) & w(1,2,r,s)

w(1,2,s,s) <=> msw(tr(s),s)
weather(W,Time) :-
    Time >= 0,
    msw(init,W0),
    w(0,Time,W0,W).

w(T,T,W,W).

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

weather(s,2) <-> msw(init,s) & w(0,2,s,s)
    v msw(init,r) & w(0,2,r,s)

w(0,2,s,s) <-> msw(tr(s),s) & w(1,2,s,s)
    v msw(tr(s),r) & w(1,2,r,s)

w(0,2,r,s) <-> msw(tr(r),s) & w(1,2,r,s)
    v msw(tr(r),r) & w(1,2,r,s)

w(1,2,s,s) <-> msw(tr(s),s)

w(1,2,r,s) <-> msw(tr(r),s)
PRISM: compute probability by dynamic programming

\[
\text{weather}(s,2) \iff \text{msw}(\text{init},s) \land w(0,2,s,s) \lor \text{msw}(\text{init},r) \land w(0,2,r,s)
\]

\[
w(0,2,s,s) \iff \text{msw}(\text{tr}(s),s) \land w(1,2,s,s) \lor \text{msw}(\text{tr}(s),r) \land w(1,2,r,s)
\]

\[
w(0,2,r,s) \iff \text{msw}(\text{tr}(r),s) \land w(1,2,r,s) \lor \text{msw}(\text{tr}(r),r) \land w(1,2,r,s)
\]

\[
w(1,2,s,s) \iff \text{msw}(\text{tr}(s),s)
\]

\[
w(1,2,r,s) \iff \text{msw}(\text{tr}(r),s)
\]
PRISM: compute probability by dynamic programming

\[
\text{weather}(s,2) = \text{msw}(\text{init},s) \times \text{w}(0,2,s,s) + \text{msw}(\text{init},r) \times \text{w}(0,2,r,s)
\]

\[
\text{w}(0,2,s,s) = \text{msw}(\text{tr}(s),s) \times \text{w}(1,2,s,s) + \text{msw}(\text{tr}(s),r) \times \text{w}(1,2,r,s)
\]

\[
\text{w}(0,2,r,s) = \text{msw}(\text{tr}(r),s) \times \text{w}(1,2,r,s) + \text{msw}(\text{tr}(r),r) \times \text{w}(1,2,r,s)
\]

\[
\text{w}(1,2,s,s) = \text{msw}(\text{tr}(s),s)
\]

\[
\text{w}(1,2,r,s) = \text{msw}(\text{tr}(r),s)
\]

- replace
  - conjunction by multiplication
  - disjunction by summation
PRISM: compute probability by dynamic programming

\[
\text{weather}(s,2) = 0.5 \times w(0,2,s,s) + 0.5 \times w(0,2,r,s)
\]
\[
w(0,2,s,s) = 0.6 \times w(1,2,s,s) + 0.4 \times w(1,2,r,s)
\]
\[
w(0,2,r,s) = 0.2 \times w(1,2,r,s) + 0.8 \times w(1,2,r,s)
\]
\[
w(1,2,s,s) = 0.6
\]
\[
w(1,2,r,s) = 0.2
\]

- replace
  - conjunction by multiplication
  - disjunction by summation
  - msw-atoms by their probability
PRISM: compute probability by dynamic programming

\[
\text{weather}(s, 2) = 0.5 \times 0.44 + 0.5 \times 0.28 = 0.36
\]

\[
w(0, 2, s, s) = 0.6 \times 0.6 + 0.4 \times 0.2 = 0.44
\]

\[
w(0, 2, r, s) = 0.2 \times 0.6 + 0.8 \times 0.2 = 0.28
\]

\[
w(1, 2, s, s) = 0.6
\]

\[
w(1, 2, r, s) = 0.2
\]

- replace
- conjunction by multiplication
- disjunction by summation
- msw-atoms by their probability
- propagate values
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
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)
- no tied parameters
How hard is query evaluation?

q1 :- stress(X), influences(X,Y).

0.7::stress(1).
0.4::stress(2).
0.9::stress(3).

0.83::influences(1,2).
0.41::influences(1,3).
0.17::influences(3,2).
How hard is query evaluation?

q1 :- stress(X), influences(X,Y).
    0.83::influences(1,2).
    0.7::stress(1).
    0.41::influences(1,3).
    0.4::stress(2).
    ... 
    0.9::stress(3).
    0.17::influences(3,2).

polynomial in database size / number of persons
How hard is query evaluation?

\[ q_1 : - \text{stress}(X), \text{influences}(X,Y). \]
\[
0.83::\text{influences}(1,2).
0.7::\text{stress}(1).
0.41::\text{influences}(1,3).
0.4::\text{stress}(2).
0.9::\text{stress}(3).
0.17::\text{influences}(3,2).
\]

polynomial in database size / number of persons

\[ q_2 : - \text{stress}(X), \text{influences}(X,Y), \text{male}(Y). \]

0.3::\text{male}(1).
0.7::\text{stress}(1).
0.8::\text{male}(2).
0.4::\text{stress}(2).
0.9::\text{male}(3).
0.9::\text{stress}(3).
0.83::\text{influences}(1,2).
0.41::\text{influences}(1,3).
...
How hard is query evaluation?

\[ q_1 \leftarrow \text{stress}(X), \text{influences}(X,Y). \]
  \[
  0.83::\text{influences}(1,2).
  0.7::\text{stress}(1).
  0.41::\text{influences}(1,3).
  0.4::\text{stress}(2).
  \ldots
  0.9::\text{stress}(3).
  0.17::\text{influences}(3,2).
\]

polynomial in database size / number of persons

\[ q_2 \leftarrow \text{stress}(X), \text{influences}(X,Y), \text{male}(Y). \]

\[
0.3::\text{male}(1).
0.7::\text{stress}(1).
0.8::\text{male}(2).
0.4::\text{stress}(2).
0.9::\text{male}(3).
0.9::\text{stress}(3).
0.83::\text{influences}(1,2).
0.41::\text{influences}(1,3).
\ldots
0.17::\text{influences}(3,2).
\]

#P-hard
### Dichotomy of UCQ Evaluation

**Figure 5.4:** The query compilation hierarchy for Unions of Conjunctive Queries (UCQ).

- **#P-hard**
- **Polynomial**

*Fig. from [Suciu et al 2011]*
Dichotomy of UCQ Evaluation

Figure 5.4: The query compilation hierarchy for Unions of Conjunctive Queries (UCQ).

Fig. from [Suciu et al 2011]
Dichotomy of UCQ Evaluation

\[ H_0 = s(X), i(X,Y), m(Y) \]

#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]
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
Domain-lifted Inference

[Van den Broeck, NIPS 11]

• a formal definition of lifted inference: probabilistic inference in time polynomial in the number of objects in the domain

• completeness result: probabilistic models with up to two logical variables per clause are domain-liftable
Example: Weighted First-Order Model Counting (WFOMC)

First-Order d-DNNF Circuit

Weighted First-Order Model Count is 1479.85

Domain

Alice
Bob
Charlie

Smokes → 1
¬Smokes → 1
Friends → 1
¬Friends → 1
F → exp(3.14)
¬F → 1

Weight Function
Lifted Inference for PLP

- LP^2: translate ProbLog program to Prolog Factor Language & run lifted variable elimination (GC-FOVE with new operator) [Bellodi et al, ICLP 14]

- new skolemization procedure that is sound for WFOMC [Van den Broeck et al, KR 14]

- thus: ProbLog programs with up to two logical variables per clause (and tied parameters) are domain-liftable
Approximate Inference

• Lower and upper bounds

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

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

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

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

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

<table>
<thead>
<tr>
<th>Evidence holds</th>
<th>Evidence does not hold</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Diagram" /></td>
<td><img src="image2" alt="Diagram" /></td>
</tr>
</tbody>
</table>
Rejection Sampling

\[ P(\text{query} \mid \text{evidence}) \approx \frac{\# \text{ query & evidence holds}}{\# \text{ evidence holds}} \]

Diagram:
- Query holds
- Query does not hold
- Evidence holds
- Evidence does not hold
Likelihood Weighting

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

e.g., webpage classification model

for each \textit{CLASS1, CLASS2} and each \textit{WORD}

\begin{verbatim}
?? :: link_class(Source, Target, CLASS1, CLASS2).
?? :: word_class(WORD, CLASS).
\end{verbatim}

class(Page, C) :- has_word(Page, W), word_class(W, C).

class(Page, C) :- links_to(OtherPage, Page),
    class(OtherPage, OtherClass),
    link_class(OtherPage, Page, OtherClass, C).

Sampling Interpretations
Sampling
Interpretations
Parameter Estimation
Parameter Estimation

\[ p(\text{fact}) = \frac{\text{count(fact is true)}}{\text{Number of interpretations}} \]
Learning from partial interpretations

• Not all facts observed
• Soft-EM
• use expected count instead of count
• \( P(Q \mid E) \) -- conditional queries!

[Gutmann et al, ECML 11; Fierens et al, TPLP 14]
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

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).

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).

prior

ask for posterior

data
ProbLog Example

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).

Prior distribution:

<table>
<thead>
<tr>
<th>Probability</th>
<th>0.1</th>
<th>0.3</th>
<th>0.5</th>
<th>0.7</th>
<th>0.9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.05</td>
<td>0.2</td>
<td>0.3</td>
<td>0.5</td>
<td>0.7</td>
</tr>
<tr>
<td>0.2</td>
<td>0.9</td>
<td>0.675</td>
<td>0.45</td>
<td>0.225</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Posterior distribution for c1:

<table>
<thead>
<tr>
<th>Probability</th>
<th>0.1</th>
<th>0.3</th>
<th>0.5</th>
<th>0.7</th>
<th>0.9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.1</td>
<td>0.3</td>
<td>0.5</td>
<td>0.7</td>
<td>0.9</td>
</tr>
<tr>
<td>0.2</td>
<td>0.9</td>
<td>0.675</td>
<td>0.45</td>
<td>0.225</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Posterior distribution for c2:

<table>
<thead>
<tr>
<th>Probability</th>
<th>0.1</th>
<th>0.3</th>
<th>0.5</th>
<th>0.7</th>
<th>0.9</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.1</td>
<td>0.3</td>
<td>0.5</td>
<td>0.7</td>
<td>0.9</td>
</tr>
<tr>
<td>0.2</td>
<td>0.9</td>
<td>0.675</td>
<td>0.45</td>
<td>0.225</td>
<td>0.2</td>
</tr>
</tbody>
</table>
Roadmap

• Modeling (with detours to related work)
• Reasoning (and a bit of learning)
• Language extensions
Dynamics
Causal Probabilistic Time-Logic (CPT-L)

how does the world change over time?
Causal Probabilistic Time-Logic (CPT-L)

how does the world change over time?

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

\textbf{city}(C,Owner), \textbf{city}(C2,Attacker), \textbf{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::conquest(Attacker,C); 0.6::nil :-

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

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 \textbf{effects} holds at time $T+1$

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

city(C,Owner),city(C2,Attacker),close(C,C2).

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

[Thon et al, MLJ 11]
Relational State Estimation over Time

Magnetism scenario

• object tracking
• category estimation from interactions

Box scenario

• object tracking even when invisible
• estimate spatial relations

[Nitti et al, IROS 13]
IROS 13

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):
[]

Particles

Box ID=4  Cube ID=3
ProbLog for activity recognition from video

- Separation between low-level events (LLE) and high-level events (HLE)
  - LLE: walking, running, active, inactive, abrupt
  - HLE: meeting, moving, fighting, leaving_object
- Probabilistic Logic approach: Event Calculus in ProbLog (Prob-EC) to infer the high-level events from an **algebra** of low-level events.
- Example:

  \[
  \text{initiatedAt(fighting}(P_1, P_2) = \text{true}, T) \leftarrow \\
  \text{happensAt(abrupt}(P_1), T), \\
  \text{holdsAt(close}(P_1, P_2, 44) = \text{true}, T), \\
  \text{not happensAt(inactive}(P_2), T).
  \]
Decisions
Viral Marketing

Which advertising strategy maximizes expected profit?

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?

$+$5

$-$3

Van den Broeck et al., AAAI 10
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).

decision 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

? :: 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 facts: cost/reward if true

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).

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(\(P\)).

\(\text{buys}(X) :: \text{friend}(X, Y), \text{buys}(Y), \text{buy_trust}(X, Y).\)
\(\text{buys}(X) :: \text{marketed}(X), \text{buy_marketing}(X).\)

\(\text{buys}(P) \Rightarrow 5 :: \text{person}(P).\)
\(\text{marketed}(P) \Rightarrow -3 :: \text{person}(P).\)
? :: 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)
bt(2,1) bt(2,4) bm(1)
buys(1) buys(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).

utility = \(-3 + -3 + 5 + 5 = 4\)

probability = 0.0032

marketed(1) marketed(3) 
bt(2,1)  bt(2,4)  bm(1) 
buys(1)  buys(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).

utility = -3 + -3 + 5 + 5 = 4
probability = 0.0032

world contributes 0.0032\times4 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]
Roadmap

- Modeling (with detours to related work)
- Reasoning (and a bit of learning)
- Language extensions
Dealing with uncertainty

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

Learning
- parameters
- structure

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

Common theme
Dealing with uncertainty

Reasoning with relational data

Learning
- Statistical relational learning
- Probabilistic logic learning
- Probabilistic programming

- Logic
- Databases
- Programming
- ...

- Probability theory
- Graphical models
- ...

- Parameters
- Structure
- ...

Our answer: probabilistic programming =

programming language
+ probabilistic primitives
+ inference methods

- Many languages, systems, applications, ...
- ... and much more to do!

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

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

Introduction.
Probabilistic logic programs are logic programs in which some of the facts are annotated with probabilities.

ProbLog is a tool that allows you to intuitively build programs that do not only encode complex interactions between a large sets of heterogeneous components but uncertainties that are present in real-life situations.

The engine tackles several tasks such as computing the marginals given evidence and learning from (partial) interpretations. ProbLog is a suite of efficient algorithms for these tasks. It is based on a conversion of the program and the queries and evidence to a weighted Boolean formula. This allows us to reduce the inference tasks to well-understood and well-studied algorithms for weighted model counting, which can be solved using state-of-the-art methods known from the graphical model and knowledge compilation literature.

The Language. Probabilistic Logic Programming.
ProbLog makes it easy to express complex, probabilistic models.

0.3 :: stress(X) :- person(X).
• 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


Bellodi E, Riguzzi F (2014b) Structure learning of probabilistic logic programs by searching the clause space. Theory and Practice of Logic Programming (TPLP) FirstView


2014