Applying algebraic statistics to probabilistic-logical representations

James Cussens

Leuven, 2007-09-24
Mixing logic and probability: the PRISM approach

Statistical models

Algebraic statistics

Future work
Outline

Mixing logic and probability: the PRISM approach
   The base distribution
   The target predicate distribution
   PRISM pragmatics

Statistical models

Algebraic statistics
   Algebra of PRISM programs
   Geometry of PRISM programs

Future work
The division of labour is:

**Probability**  Very simple—families of independent and identically distributed random variables.

**Logic**  Arbitrarily complex—using a standard first-order theory.
An example ‘base’ probability distribution

- Let $X_1, X_2, X_3, \ldots$ be an infinite collection of independent and identically distributed (iid) random variables taking values ‘y’ and ‘n’.
- Similarly let $Y_1, Y_2, Y_3, \ldots$ and $Z_1, Z_2, Z_3, \ldots$ also be iid families with values in \{0, 1\} and \{0, 1, 2, \ldots 9\}, respectively.
- Here’s (the beginning) of a joint instantiation of all these variables:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X$</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>...</td>
</tr>
<tr>
<td>$Y$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>...</td>
</tr>
<tr>
<td>$Z$</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>...</td>
</tr>
</tbody>
</table>
A joint instantiation determines a logical theory

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>…</td>
</tr>
<tr>
<td>Y</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>…</td>
</tr>
<tr>
<td>Z</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>…</td>
</tr>
</tbody>
</table>

▶ This joint instantiation determines the following logical theory:

\[
\text{msw}('X', 1, y), \text{msw}('X', 2, n), \text{msw}('X', 3, y), \ldots \\
\text{msw}('Y', 1, 0), \text{msw}('Y', 2, 1), \text{msw}('Y', 3, 0), \ldots \\
\text{msw}('Z', 1, 4), \text{msw}('Z', 2, 3), \text{msw}('Z', 3, 1), \ldots 
\]
Defining a ‘base’ distribution in PRISM

Here’s the PRISM source defining the example distribution:

values('X',[y,n]).
values('Y',[0,1]).
values('Z',[0,1,2,3,4,5,6,7,8,9]).

:- set_sw('X',0.3+0.7).
:- set_sw('Y',0.4+0.6).
:- set_sw('Z',0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1).
Using a fixed, arbitrary logical theory to extend a base distribution

- Can extend this simple base distribution by considering what becomes true once a probabilistically chosen theory is added to an existing fixed logical theory $R$.

- Let $\text{fla}$ be some first-order sentence. $\text{Prob}(\text{fla})$ is the probability of getting a base joint instantiation $F$ such that $F, R \vdash \text{fla}$.
Working with target predicates

- It is convenient to specify a *target predicate* such that exactly one ground atom with this predicate symbol follows from any choice of $F$.

- This defines a distribution over the success set of $t$.

- This can be generalised to allow *at most one* target ground atom to follow ('failure' models).

\[
F_1, R \vdash t(a) \\
F_2, R \vdash t(b) \\
F_3, R \vdash t(a)
\]
Sampling from a PRISM distribution

- We don’t sample an infinite collection of ground ‘msw’ facts!
- Instead use standard backward chaining starting from a goal: `t(X)`
- Base random variables are sampled ‘on demand’

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Sampling from a PRISM distribution

- We don’t sample an infinite collection of ground ‘msw’ facts!
- Instead use standard backward chaining starting from a goal := t(X)
- Base random variables are sampled ‘on demand’

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td></td>
<td></td>
<td>y</td>
<td></td>
<td></td>
<td></td>
<td>…</td>
</tr>
<tr>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>…</td>
</tr>
<tr>
<td>Z</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>…</td>
</tr>
</tbody>
</table>
Sampling from a PRISM distribution

- We don’t sample an infinite collection of ground ‘msw’ facts!
- Instead use standard backward chaining starting from a goal: `- t(X)`
- Base random variables are sampled ‘on demand’

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>…</td>
</tr>
<tr>
<td>Y</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>…</td>
</tr>
<tr>
<td>Z</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>…</td>
</tr>
</tbody>
</table>
Sampling from a PRISM distribution

- We don’t sample an infinite collection of ground ‘msw’ facts!
- Instead use standard backward chaining starting from a goal: \( \neg t(X) \)
- Base random variables are sampled ‘on demand’

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>Y</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>Z</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>
Sampling from a PRISM distribution

- We don’t sample an infinite collection of ground ‘msw’ facts!
- Instead use standard backward chaining starting from a goal: \(- t(X)\)
- Base random variables are sampled ‘on demand’

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>…</td>
</tr>
<tr>
<td>Y</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>…</td>
</tr>
<tr>
<td>Z</td>
<td>4</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>…</td>
</tr>
</tbody>
</table>
Sampling from a PRISM distribution

- We don’t sample an infinite collection of ground ‘msw’ facts!
- Instead use standard backward chaining starting from a goal: \(- t(X)\)
- Base random variables are sampled ‘on demand’

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>y</td>
<td>n</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>Y</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>Z</td>
<td>4</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>
Sampling from a PRISM distribution

- We don’t sample an infinite collection of ground ‘msw’ facts!
- Instead use standard backward chaining starting from a goal: 
  \[- \text{t}(X)\]
- Base random variables are sampled ‘on demand’

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>y</td>
<td>n</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>Y</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>Z</td>
<td>4</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>
Sampling from a PRISM distribution

- We don’t sample an infinite collection of ground ‘msw’ facts!
- Instead use standard backward chaining starting from a goal: \(- t(X)\)
- Base random variables are sampled ‘on demand’

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>y</td>
<td>n</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z</td>
<td>4</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Sampling from a PRISM distribution

- We don’t sample an infinite collection of ground ‘msw’ facts!
- Instead use standard backward chaining starting from a goal: $\neg t(X)$
- Base random variables are sampled ‘on demand’

```
X  y  n  
Y  0  1  0  ...  
Z  4  3  1  ...  
```

- It’s a PRISM requirement that only a finite sample is needed to determine which target predicate atom is true, if any.
- PRISM system does not ‘remember’ which msw atoms turn out to be true (unlike ProbLog).
Sampling from a PRISM distribution

- We don’t sample an infinite collection of ground ‘msw’ facts!
- Instead use standard backward chaining starting from a goal: \( \text{:- } t(X) \)
- Base random variables are sampled ‘on demand’

\[
\begin{array}{cccccccc}
& 1 & 2 & 3 & 4 & 5 & 6 & \ldots \\
X & y & n & y & & & & \\
Y & 0 & 1 & 0 & & & & \\
Z & 4 & 3 & 1 & & & & \\
\end{array}
\]

- It’s a PRISM requirement that only a finite sample is needed to determine which target predicate atom is true, if any.
- PRISM system does not ‘remember’ which msw atoms turn out to be true (unlike ProbLog).
Sampling from a PRISM distribution

- We don’t sample an infinite collection of ground ‘msw’ facts!
- Instead use standard backward chaining starting from a goal: \(- t(X)\)
- Base random variables are sampled ‘on demand’

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Sampling from a PRISM distribution

- We don’t sample an infinite collection of ground ‘msw’ facts!
- Instead use standard backward chaining starting from a goal: `:- t(X)`
- Base random variables are sampled ‘on demand’

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>…</td>
</tr>
<tr>
<td>Y</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>…</td>
</tr>
<tr>
<td>Z</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>…</td>
</tr>
</tbody>
</table>

It’s a PRISM requirement that only a finite sample is needed to determine which target predicate atom is true, if any. PRISM system does not ‘remember’ which msw atoms turn out to be true (unlike ProbLog).
Sampling from a PRISM distribution

- We don’t sample an infinite collection of ground ‘msw’ facts!
- Instead use standard backward chaining starting from a goal: \(- t(X)\)
- Base random variables are sampled ‘on demand’

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>...</td>
</tr>
<tr>
<td>Y</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>Z</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>

It's a PRISM requirement that only a finite sample is needed to determine which target predicate atom is true, if any. PRISM system does not ‘remember’ which msw atoms turn out to be true (unlike ProbLog).
Sampling from a PRISM distribution

- We don’t sample an infinite collection of ground ‘msw’ facts!
- Instead use standard backward chaining starting from a goal: \( t(X) \)
- Base random variables are sampled ‘on demand’

```
<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>...</td>
</tr>
<tr>
<td>Y</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>
```
Sampling from a PRISM distribution

- We don’t sample an infinite collection of ground ‘msw’ facts!
- Instead use standard backward chaining starting from a goal :– t(X)
- Base random variables are sampled ‘on demand’

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>n</td>
</tr>
<tr>
<td>Y</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Sampling from a PRISM distribution

- We don’t sample an infinite collection of ground ‘msw’ facts!
- Instead use standard backward chaining starting from a goal: \(- t(X)\)
- Base random variables are sampled ‘on demand’

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>…</td>
</tr>
<tr>
<td>Y</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td>…</td>
</tr>
<tr>
<td>Z</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td></td>
<td></td>
<td>…</td>
</tr>
</tbody>
</table>
Sampling from a PRISM distribution

- We don’t sample an infinite collection of ground ‘msw’ facts!
- Instead use standard backward chaining starting from a goal: `:- t(X)`
- Base random variables are sampled ‘on demand’

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>…</td>
</tr>
<tr>
<td>Y</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>…</td>
</tr>
<tr>
<td>Z</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>…</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Sampling from a PRISM distribution

- We don’t sample an infinite collection of ground ‘msw’ facts!
- Instead use standard backward chaining starting from a goal: \(- t(X)\)
- Base random variables are sampled ‘on demand’

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>y</td>
<td>n</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>n</td>
<td>…</td>
</tr>
<tr>
<td>Y</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>…</td>
</tr>
<tr>
<td>Z</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td></td>
<td></td>
<td>…</td>
</tr>
</tbody>
</table>

- It’s a PRISM requirement that only a finite sample is needed to determine which target predicate atom is true, if any.
- PRISM system does not ‘remember’ which msw atoms turn out to be true (unlike ProbLog).
Computing target probabilities from a PRISM distribution

- We don’t consider all possible infinite instantiations of the base distribution!
- It’s a PRISM requirement that \( \text{Prob}(t(a)) \) for any target atom \( t(a) \) is a finite sum of finite products of base distribution probabilities.
- For a given \( t(a) \), abduction is used to find (conjunctions of) ‘msw’ facts that make \( t(a) \) true.
Abduction: a HMM example

\[ \text{hmm}([a, b, a]) \]
\[ \iff \text{msw}(\text{out}(s0), 1, a) \land \text{msw}(\text{tr}(s0), 1, s0) \land \text{hmm}(s0, [b, a]) \]
\[ \lor \text{msw}(\text{out}(s0), 1, a) \land \text{msw}(\text{tr}(s0), 1, s1) \land \text{hmm}(s1, [b, a]) \]
Abduction: a HMM example

\[
\text{hmm}([a,b,a]) \\
\Leftrightarrow \text{msw}(\text{out}(s0),1,a) \land \text{msw}(\text{tr}(s0),1,s0) \land \text{hmm}(s0,[b,a]) \\
\lor \text{msw}(\text{out}(s0),1,a) \land \text{msw}(\text{tr}(s0),1,s1) \land \text{hmm}(s1,[b,a]) \\
\text{hmm}(s0,[b,a]) \\
\Leftrightarrow \text{msw}(\text{out}(s0),2,b) \land \text{msw}(\text{tr}(s0),2,s0) \land \text{hmm}(s0,[a]) \\
\lor \text{msw}(\text{out}(s0),2,b) \land \text{msw}(\text{tr}(s0),2,s1) \land \text{hmm}(s1,[a])
\]
Abduction: a HMM example

\[ \text{hmm}([a,b,a]) \]
\[ \Leftrightarrow \text{msw}(\text{out}(s0),1,a) \land \text{msw}(\text{tr}(s0),1,s0) \land \text{hmm}(s0,[b,a]) \]
\[ \lor \text{msw}(\text{out}(s0),1,a) \land \text{msw}(\text{tr}(s0),1,s1) \land \text{hmm}(s1,[b,a]) \]

\[ \text{hmm}(s0,[b,a]) \]
\[ \Leftrightarrow \text{msw}(\text{out}(s0),2,b) \land \text{msw}(\text{tr}(s0),2,s0) \land \text{hmm}(s0,[a]) \]
\[ \lor \text{msw}(\text{out}(s0),2,b) \land \text{msw}(\text{tr}(s0),2,s1) \land \text{hmm}(s1,[a]) \]

\[ \text{hmm}(s1,[b,a]) \]
\[ \Leftrightarrow \text{msw}(\text{out}(s1),1,b) \land \text{msw}(\text{tr}(s1),1,s0) \land \text{hmm}(s0,[a]) \]
\[ \lor \text{msw}(\text{out}(s1),1,b) \land \text{msw}(\text{tr}(s1),1,s1) \land \text{hmm}(s1,[a]) \]
Abduction: a HMM example

\[
\text{hmm}([a,b,a]) \\
\Leftrightarrow \text{msw}(\text{out}(s0),1,a) \land \text{msw}(\text{tr}(s0),1,s0) \land \text{hmm}(s0,[b,a]) \\
\lor \text{msw}(\text{out}(s0),1,a) \land \text{msw}(\text{tr}(s0),1,s1) \land \text{hmm}(s1,[b,a]) \\
\text{hmm}(s0,[b,a]) \\
\Leftrightarrow \text{msw}(\text{out}(s0),2,b) \land \text{msw}(\text{tr}(s0),2,s0) \land \text{hmm}(s0,[a]) \\
\lor \text{msw}(\text{out}(s0),2,b) \land \text{msw}(\text{tr}(s0),2,s1) \land \text{hmm}(s1,[a]) \\
\text{hmm}(s1,[b,a]) \\
\Leftrightarrow \text{msw}(\text{out}(s1),1,b) \land \text{msw}(\text{tr}(s1),1,s0) \land \text{hmm}(s0,[a]) \\
\lor \text{msw}(\text{out}(s1),1,b) \land \text{msw}(\text{tr}(s1),1,s1) \land \text{hmm}(s1,[a]) \\
\text{hmm}(s0,[a]) \\
\Leftrightarrow \text{msw}(\text{out}(s0),3,a) \land \text{msw}(\text{tr}(s0),3,\text{stop})
\]
Abduction: a HMM example

\[
hmm([a,b,a]) \\
\Leftrightarrow msw(out(s0),1,a) \land msw(tr(s0),1,s0) \land hmm(s0,[b,a]) \\
\lor msw(out(s0),1,a) \land msw(tr(s0),1,s1) \land hmm(s1,[b,a]) \\
hmm(s0,[b,a]) \\
\Leftrightarrow msw(out(s0),2,b) \land msw(tr(s0),2,s0) \land hmm(s0,[a]) \\
\lor msw(out(s0),2,b) \land msw(tr(s0),2,s1) \land hmm(s1,[a]) \\
hmm(s1,[b,a]) \\
\Leftrightarrow msw(out(s1),1,b) \land msw(tr(s1),1,s0) \land hmm(s0,[a]) \\
\lor msw(out(s1),1,b) \land msw(tr(s1),1,s1) \land hmm(s1,[a]) \\
hmm(s0,[a]) \\
\Leftrightarrow msw(out(s0),3,a) \land msw(tr(s0),3,stop) \\
hmm(s1,[a]) \\
\Leftrightarrow msw(out(s1),2,a) \land msw(tr(s1),2,stop)
\]
Computing probabilities by abduction

\[ \text{hmm([a,b,a])} \]
\[ \Leftrightarrow \text{msw(out(s0),1,a)} \land \text{msw(tr(s0),1,s0)} \land \text{hmm(s0,[b,a])} \]
\[ \lor \text{msw(out(s0),1,a)} \land \text{msw(tr(s0),1,s1)} \land \text{hmm(s1,[b,a])} \]
\[ \text{hmm(s0,[b,a])} \]
\[ \Leftrightarrow \text{msw(out(s0),2,b)} \land \text{msw(tr(s0),2,s0)} \land \text{hmm(s0,[a])} \]
\[ \lor \text{msw(out(s0),2,b)} \land \text{msw(tr(s0),2,s1)} \land \text{hmm(s1,[a])} \]
\[ \text{hmm(s1,[b,a])} \]
\[ \Leftrightarrow \text{msw(out(s1),1,b)} \land \text{msw(tr(s1),1,s0)} \land \text{hmm(s0,[a])} \]
\[ \lor \text{msw(out(s1),1,b)} \land \text{msw(tr(s1),1,s1)} \land \text{hmm(s1,[a])} \]
\[ \text{hmm(s0,[a])} \]
\[ \Leftrightarrow \text{msw(out(s0),3,a)} \land \text{msw(tr(s0),3,stop)} \]
\[ \text{hmm(s1,[a])} \]
\[ \Leftrightarrow \text{msw(out(s1),2,a)} \land \text{msw(tr(s1),2,stop)} \]
Computing probabilities by abduction

\[
\Pr(hmm([a,b,a])) \\
= \Pr(msw(out(s0),1,a)) \times \Pr(msw(tr(s0),1,s0)) \times \Pr(hmm(s0,[b,a])) \\
+ \Pr(msw(out(s0),1,a)) \times \Pr(msw(tr(s0),1,s1)) \times \Pr(hmm(s1,[b,a])) \\
hmm(s0,[b,a]) \\
\Leftrightarrow msw(out(s0),2,b) \land msw(tr(s0),2,s0) \land hmm(s0,[a]) \\
\lor msw(out(s0),2,b) \land msw(tr(s0),2,s1) \land hmm(s1,[a]) \\
hmm(s1,[b,a]) \\
\Leftrightarrow msw(out(s1),1,b) \land msw(tr(s1),1,s0) \land hmm(s0,[a]) \\
\lor msw(out(s1),1,b) \land msw(tr(s1),1,s1) \land hmm(s1,[a]) \\
hmm(s0,[a]) \\
\Leftrightarrow msw(out(s0),3,a) \land msw(tr(s0),3,stop) \\
hmm(s1,[a]) \\
\Leftrightarrow msw(out(s1),2,a) \land msw(tr(s1),2,stop)
\]
Computing probabilities by abduction

\[\Pr(hmm([a, b, a]))\]
\[= \Pr(msw(out(s0), 1, a)) \times \Pr(msw(tr(s0), 1, s0)) \times \Pr(hmm(s0, [b, a]))\]
\[+ \Pr(msw(out(s0), 1, a)) \times \Pr(msw(tr(s0), 1, s1)) \times \Pr(hmm(s1, [b, a]))\]
\[= \Pr(msw(out(s0), 2, b)) \times \Pr(msw(tr(s0), 2, s0)) \times \Pr(hmm(s0, [a]))\]
\[+ \Pr(msw(out(s0), 2, b)) \times \Pr(msw(tr(s0), 2, s1)) \times \Pr(hmm(s1, [a]))\]

\[hmm(s1, [b, a]) \Leftrightarrow msw(out(s1), 1, b) \land msw(tr(s1), 1, s0) \land hmm(s0, [a])\]
\[\lor msw(out(s1), 1, b) \land msw(tr(s1), 1, s1) \land hmm(s1, [a])\]
\[hmm(s0, [a]) \leftrightarrow msw(out(s0), 3, a) \land msw(tr(s0), 3, stop)\]
\[hmm(s1, [a]) \leftrightarrow msw(out(s1), 2, a) \land msw(tr(s1), 2, stop)\]
Computing probabilities by abduction

\[
\begin{align*}
\Pr(\text{hmm}([a,b,a])) &= \Pr(\text{msw}(\text{out}(s0),1,a)) \times \Pr(\text{msw}(\text{tr}(s0),1,s0)) \times \Pr(\text{hmm}(s0,[b,a])) \\
&+ \Pr(\text{msw}(\text{out}(s0),1,a)) \times \Pr(\text{msw}(\text{tr}(s0),1,s1)) \times \Pr(\text{hmm}(s1,[b,a])) \\
\Pr(\text{hmm}(s0,[b,a])) &= \Pr(\text{msw}(\text{out}(s0),2,b)) \times \Pr(\text{msw}(\text{tr}(s0),2,s0)) \times \Pr(\text{hmm}(s0,[a])) \\
&+ \Pr(\text{msw}(\text{out}(s0),2,b)) \times \Pr(\text{msw}(\text{tr}(s0),2,s1)) \times \Pr(\text{hmm}(s1,[a])) \\
\Pr(\text{hmm}(s1,[b,a])) &= \Pr(\text{msw}(\text{out}(s1),1,b)) \times \Pr(\text{msw}(\text{tr}(s1),1,s0)) \times \Pr(\text{hmm}(s0,[a])) \\
&+ \Pr(\text{msw}(\text{out}(s1),1,b)) \times \Pr(\text{msw}(\text{tr}(s1),1,s1)) \times \Pr(\text{hmm}(s1,[a])) \\
\text{hmm}(s0,[a]) &\iff \text{msw}(\text{out}(s0),3,a) \land \text{msw}(\text{tr}(s0),3,\text{stop}) \\
\text{hmm}(s1,[a]) &\iff \text{msw}(\text{out}(s1),2,a) \land \text{msw}(\text{tr}(s1),2,\text{stop})
\end{align*}
\]
Computing probabilities by abduction

\[
\Pr(hmm([a,b,a])) \\
= \Pr(msw(out(s0),1,a)) \times \Pr(msw(tr(s0),1,s0)) \times \Pr(hmm(s0,[b,a])) \\
+ \Pr(msw(out(s0),1,a)) \times \Pr(msw(tr(s0),1,s1)) \times \Pr(hmm(s1,[b,a])) \\
\Pr(hmm(s0,[b,a])) \\
= \Pr(msw(out(s0),2,b)) \times \Pr(msw(tr(s0),2,s0)) \times \Pr(hmm(s0,[a]))) \\
+ \Pr(msw(out(s0),2,b)) \times \Pr(msw(tr(s0),2,s1)) \times \Pr(hmm(s1,[a]))) \\
\Pr(hmm(s1,[b,a])) \\
= \Pr(msw(out(s1),1,b)) \times \Pr(msw(tr(s1),1,s0)) \times \Pr(hmm(s0,[a]))) \\
+ \Pr(msw(out(s1),1,b)) \times \Pr(msw(tr(s1),1,s1)) \times \Pr(hmm(s1,[a]))) \\
\Pr(hmm(s0,[a])) \\
= \Pr(msw(out(s0),3,a)) \times \Pr(msw(tr(s0),3,stop)) \\
hmm(s1,[a]) \\
\Leftrightarrow msw(out(s1),2,a) \land msw(tr(s1),2,stop)
\]
Computing probabilities by abduction

\[ \Pr(hmm([a,b,a])) = \Pr(msw(out(s0),1,a)) \times \Pr(msw(tr(s0),1,s0)) \times \Pr(hmm(s0,[b,a])) + \Pr(msw(out(s0),1,a)) \times \Pr(msw(tr(s0),1,s1)) \times \Pr(hmm(s1,[b,a])) \]

\[ \Pr(hmm(s0,[b,a])) = \Pr(msw(out(s0),2,b)) \times \Pr(msw(tr(s0),2,s0)) \times \Pr(hmm(s0,[a])) + \Pr(msw(out(s0),2,b)) \times \Pr(msw(tr(s0),2,s1)) \times \Pr(hmm(s1,[a])) \]

\[ \Pr(hmm(s1,[b,a])) = \Pr(msw(out(s1),1,b)) \times \Pr(msw(tr(s1),1,s0)) \times \Pr(hmm(s0,[a])) + \Pr(msw(out(s1),1,b)) \times \Pr(msw(tr(s1),1,s1)) \times \Pr(hmm(s1,[a])) \]

\[ \Pr(hmm(s0,[a])) = \Pr(msw(out(s0),3,a)) \times \Pr(msw(tr(s0),3,stop)) \]

\[ \Pr(hmm(s1,[a])) = \Pr(msw(out(s1),2,a)) \times \Pr(msw(tr(s1),2,stop)) \]
Graphical and logical representations of Bayesian networks

\[
\text{target}(bn,4).
\]

\[
\text{values}(\_, [0,1]).
\]

\[
:- \text{set_sw}(\text{cpt}(a), 0.38+0.62).
\]

\[
\ldots
\]

\[
\text{bn}(A,B,C,D) :-
    \text{msw}(\text{cpt}(a), A),
    \text{msw}(\text{cpt}(b, A), B),
    \text{msw}(\text{cpt}(c, A, B), C),
    \text{msw}(\text{cpt}(d, A, C), D).
\]
Outline

Mixing logic and probability: the PRISM approach
  The base distribution
  The target predicate distribution
  PRISM pragmatics

Statistical models

  Algebraic statistics
    Algebra of PRISM programs
    Geometry of PRISM programs

Future work
A statistical model is just a set of probability distributions.

Usually a particular distribution in a model is specified by setting the model’s parameters.

Two key problems:

- **Parameter estimation**  Given a statistical model, find the parameters which best ‘fit’ some data.
- **Model equivalence**  Given two syntactically distinct representations, determine whether they, in fact, define the same model (= set of probability distributions).

Understanding model equivalence is important for structure learning (and sometimes parameter estimation).
Model equivalence between Bayesian nets

- Model equivalence for Bayesian networks is well-understood.
- These two graphs represent the same set of distributions, since they have the same undirected skeleton and same immoralities.
Model equivalence for PRISM programs

- The structure of the statistical model defined by a PRISM program is determined by the fixed logical theory $R$.
- The key task: Use the structure of $R$ to ‘get to’ the structure of its associated statistical model.
- Can often translate to a graphical model and use known results from graphical modelling . . .
- . . .but what about in general?
Outline

Mixing logic and probability: the PRISM approach
  The base distribution
  The target predicate distribution
  PRISM pragmatics

Statistical models

Algebraic statistics
  Algebra of PRISM programs
  Geometry of PRISM programs

Future work
Computing probabilities by abduction

\[
\begin{align*}
\Pr(\text{hmm(}[a,b,a])) &= \Pr(\text{msw(out(s0),1,a)}) \times \Pr(\text{msw(tr(s0),1,s0)}) \times \Pr(\text{hmm(s0,[b,a])}) \\
&+ \Pr(\text{msw(out(s0),1,a)}) \times \Pr(\text{msw(tr(s0),1,s1)}) \times \Pr(\text{hmm(s1,[b,a])}) \\
\Pr(\text{hmm(s0,[b,a])}) &= \Pr(\text{msw(out(s0),2,b)}) \times \Pr(\text{msw(tr(s0),2,s0)}) \times \Pr(\text{hmm(s0,[a])}) \\
&+ \Pr(\text{msw(out(s0),2,b)}) \times \Pr(\text{msw(tr(s0),2,s1)}) \times \Pr(\text{hmm(s1,[a])}) \\
\Pr(\text{hmm(s1,[b,a])}) &= \Pr(\text{msw(out(s1),1,b)}) \times \Pr(\text{msw(tr(s1),1,s0)}) \times \Pr(\text{hmm(s0,[a])}) \\
&+ \Pr(\text{msw(out(s1),1,b)}) \times \Pr(\text{msw(tr(s1),1,s1)}) \times \Pr(\text{hmm(s1,[a])}) \\
\Pr(\text{hmm(s0,[a])}) &= \Pr(\text{msw(out(s0),3,a)}) \times \Pr(\text{msw(tr(s0),3,stop})) \\
\Pr(\text{hmm(s1,[a])}) &= \Pr(\text{msw(out(s1),2,a)}) \times \Pr(\text{msw(tr(s1),2,stop}))
\end{align*}
\]
Abbreviating . . .

\begin{align*}
p_{0,a,b,a} & = t_{0a} \times t_{00} \times p_{0,b,a} \\
& + t_{0a} \times t_{01} \times p_{1,b,a} \\
p_{0,b,a} & = t_{0b} \times t_{00} \times p_{0,a} \\
& + t_{0b} \times t_{01} \times p_{1,a} \\
p_{1,b,a} & = t_{1b} \times t_{10} \times p_{0,a} \\
& + t_{1b} \times t_{11} \times p_{1,a} \\
p_{0,a} & = t_{0a} \times t_{0s} \\
p_{1,a} & = t_{1a} \times t_{1s}
\end{align*}
... and eliminating terms

\[ p_{0,a,b,a} = t_0 a t_00 (t_0 b t_00 (t_0 a t_0 s) + t_0 b t_01 (t_1 a t_1 s)) + t_0 a t_01 (t_1 b t_10 (t_0 a t_0 s) + t_1 b t_11 (t_1 a t_1 s)) \]
\[ = t_{00}^2 t_0 a t_0 b t_0 s + t_0 a t_00 t_0 b t_01 t_1 a t_1 s + t_{01}^2 t_01 t_1 b t_10 t_0 s + t_0 a t_01 t_1 b t_11 t_1 a t_1 s \]
... and eliminating terms

\[ p_{0,a,b,a} = t_0a t_{00} (t_{0b} t_{00} (t_0a t_{0s}) + t_{0b} t_{01} (t_1a t_{1s})) + t_0a t_{01} (t_{1b} t_{10} (t_0a t_{0s}) + t_{1b} t_{11} (t_1a t_{1s})) \]
\[ = t_{00} t_{0a} t_{0b} t_{0s} + t_{0a} t_{00} t_{0b} t_{01} t_{1a} t_{1s} + t_{0a} t_{01} t_{1b} t_{10} t_{0s} + t_{0a} t_{01} t_{1b} t_{11} t_{1a} t_{1s} \]

- Each target probability will be a polynomial function of the model parameters.
- (For failure models they are rational functions.)
General form of a PRISM distribution

- Each $f_i$ is a polynomial.
- The $p_i$ are the target probabilities; in general, there can be infinitely many of these.
- The $t_i$ are the parameters (which are ‘msw’ probabilities).
- (For failure models each RHS is divided by $Z = \sum_i \text{RHS}_i$.)

\[
\begin{align*}
    p_1 &= f_1(t_1, t_2, \ldots t_m) \\
    p_2 &= f_2(t_1, t_2, \ldots t_m) \\
    \vdots \\
    p_i &= f_i(t_1, t_2, \ldots t_m) \\
    \vdots
\end{align*}
\]
Polynomial ideals

\[ p_i = f_i(t_1, t_2, \ldots t_m) \]

\[ \Leftrightarrow \quad p_i - f_i(t_1, t_2, \ldots t_m) = 0 \]

- The set of such LHS polynomials and all polynomials that follow from them form a (polynomial) ideal \( I = \langle p_i - f_i \rangle \).
- \( f, g \in I \Rightarrow f + g \in I, \)
  \( f \in I \Rightarrow fh \in I \) where \( h \) is an arbitrary polynomial.
- Any ideal can be represented by a finite basis (Hilbert’s theorem)
Implicitisation

- Each polynomial in the ideal is a *constraint* on target probabilities and/or parameters.
- Buchberger’s algorithm allows us to *eliminate* parameters and derive polynomials only involving target probabilities.
- This process, called *implicitisation*, finds all constraints between target probabilities.
- All conditional independence relations can be found in this way.
with(PolynomialIdeals);
simple := <a0+a1-1, b0+b1-1, ab00+ab01+ab10+ab11-1, a0*b0-ab00, a0*b1-ab01, a1*b0-ab10, a1*b1-ab11>
EliminationIdeal(simple, {ab00, ab01, ab10, ab11})
<
  ab00 + ab01 + ab10 + ab11 - 1,
  2
ab00 ab10 + ab00 + ab01 ab10 + ab00 ab01 - ab00
>

Implicitisation using MAPLE
Solving the PRISM model equivalence problem

- Given two PRISM programs defining *finite* distributions over the same target predicate...
Solving the PRISM model equivalence problem

▶ Given two PRISM programs defining \emph{finite} distributions over the same target predicate . . .
▶ . . . yank out all the polynomials using abduction . . .
Solving the PRISM model equivalence problem

- Given two PRISM programs defining finite distributions over the same target predicate . . .
- . . . yank out all the polynomials using abduction . . .
- . . . do implicitisation and check to see whether each elimination ideal is contained in the other.
Solving the PRISM model equivalence problem

- Given two PRISM programs defining finite distributions over the same target predicate . . .
- . . . yank out all the polynomials using abduction . . .
- . . . do implicitisation and check to see whether each elimination ideal is contained in the other.
- There are some short cuts, but this is the basic idea.
with(PolynomialIdeals);
> bns := <pa0 + pa1 - 1, pba01 + pba11 - 1,
pdac001 + pdac101 - 1, pcb01 + pcb11 - 1,
pcb00 + pcb10 - 1, pdac010 + pdac110 - 1,
pdac011 + pdac111 - 1, pdac000 + pdac100 - 1,
pba00 + pba10 - 1>
> newcons := [pa0*pa1*pba00*pba01*
(pdac001*pdac010-pdac000*pdac011)]
> IdealMembership(newcons, bns)
    false
Geometry of probability distributions

- A finite probability distribution is just a collection of \( n \) real numbers and thus can be considered a point in \( n \)-dimensional space.

- A statistical model is a set of probability distributions and is thus a subspace of \( n \)-dimensional space.

- Since a PRISM model is defined by polynomials, the space it ‘carves out’ is a *variety*—the set of points where all polynomials in the ideal vanish.
Implicitisation is projection

```latex
with(PolynomialIdeals);

simple := <a0+a1-1, b0+b1-1, ab00+ab01+ab10+ab11-1,
            a0*b0-ab00, a0*b1-ab01, a1*b0-ab10, a1*b1-ab11>

EliminationIdeal(simple, {ab00, ab01, ab10, ab11})
< ab00 + ab01 + ab10 + ab11 - 1,
  2
ab00 ab10 + ab00 + ab01 ab10 + ab00 ab01 - ab00 >
```

- simple defines a ‘curve’ in 8-dimensional space.
- This is the projected down onto 4 dimensions.
Outline

Mixing logic and probability: the PRISM approach
  The base distribution
  The target predicate distribution
  PRISM pragmatics

Statistical models

Algebraic statistics
  Algebra of PRISM programs
  Geometry of PRISM programs

Future work
Beyond propositionalisation

- Eliminating indeterminates via Buchberger’s algorithm is incredibly slow (at least using MAPLE).
- The approach presented uses propositionalisation and thus fails miserably to exploit the logical structure of the PRISM program.
- We need a ‘first-order’ approach, presumably based on program transformation.