A tutorial on logic-based approaches to SRL

James Cussens
University of York, UK

SIM 2009, Joint Conference SRL-ILP-MLG

Overview

- Some connections between logic and probability
- Briefly: first-order probabilistic models using 'parfactors' (e.g. Markov logic)
- In more detail: a generative approach for first-order probabilistic models (PRISM)
- An example of using SRL in statistical genetics

Making the connections

$$
\begin{array}{ccc}
\text{First-order logic} & \longrightarrow & \text{First-order probabilistic models} \\
\uparrow & & \uparrow \\
\text{Propositional logic} & \longrightarrow & \text{Propositional probabilistic models}
\end{array}
$$

Propositional logic

$$
\begin{tikzcd}
\text{First-order logic} \arrow[r] & \text{First-order probabilistic models} \\
\textcolor{red}{\text{Propositional logic}} \arrow[u] \arrow[r] & \text{Propositional probabilistic models} \arrow[u]
\end{tikzcd}
$$

Propositional formulae as zero-one factors

- Propositional atoms are binary (0-1) variables.
- A joint instantiation of all atoms/variables satisfying a propositional formula is a *model* of that formula.
- If A and B are the only propositions in our language then $A, \neg A \vee B$ has only one model.

```
A |                  A | B |              A | B |
- | -                - | - | -            - | - | -
0 | 0       *        0 | 0 | 1     =      0 | 0 | 0
1 | 1                0 | 1 | 1            0 | 1 | 0
                     1 | 0 | 0            1 | 0 | 0
                     1 | 1 | 1            1 | 1 | 1
```

Propositional probabilistic models

$$\begin{tikzcd}
\text{First-order logic} \arrow[r] & \text{First-order probabilistic models} \\
\text{Propositional logic} \arrow[r] \arrow[u] & \textcolor{red}{\text{Propositional probabilistic models}} \arrow[u]
\end{tikzcd}$$

Generalising propositional logic

- Allow arbitrary non-negative values in the factors.
- Allow variables to have more than 2 values.

A	
0	4
1	6

*

A	B	
0	0	5
0	1	5
1	0	0
1	1	7

=

A	B	
0	0	20
0	1	20
1	0	0
1	1	42

Dividing by a normalising constant Z defines a probability distribution over full joint instantiations (when $Z > 0$). Here $Z = 20 + 20 + 0 + 42 = 82$.

Further examples

```
A | B |              A | D |              B | C |              C | D |
- | - | ----         - | - | ----         - | - | ----         - | - | ----
0 | 0 | 0.10         0 | 0 | 0.90         0 | 0 | 0.40         0 | 0 | 0.50
0 | 1 | 0.20    *    0 | 1 | 0.20    *    0 | 1 | 0.70    *    0 | 1 | 0.20
1 | 0 | 0.30         1 | 0 | 0.70         1 | 0 | 0.30         1 | 0 | 0.40
1 | 1 | 0.20         1 | 1 | 0.10         1 | 1 | 0.10         1 | 1 | 0.10
```

```
A | B |              A | C |              B | C |
- | - | ----         - | - | ----         - | - | ----
0 | 0 | 0.46         0 | 0 | 0.90         0 | 0 | 0.40
0 | 1 | 0.92    *    0 | 1 | 0.20    *    0 | 1 | 0.70
1 | 0 | 1.39         1 | 0 | 0.70         1 | 0 | 0.30
1 | 1 | 0.92         1 | 1 | 0.10         1 | 1 | 0.10
```

Weighted clauses

$\infty : A$ and $2 : \neg A \vee B$

```
A |                 A | B |                A | B |
- | -               - | - | -              - | - | -
0 | 0     *         0 | 0 | 1      =       0 | 0 | 0
1 | 1               0 | 1 | 1              0 | 1 | 0
                    1 | 0 | exp(-2)        1 | 0 | exp(-2)
                    1 | 1 | 1              1 | 1 | 1
```

Finding the most probable instantiation (highest weighted model) is the weighted MAX-SAT problem.

Bayesian networks

$$
\begin{tikzcd}
\text{Alan} \arrow[d] & \\
\text{Bob} \arrow[d] & \text{Diane} \arrow[dl] \\
\text{Cate} &
\end{tikzcd}
$$

If we restrict so that:

1. each variable has an associated factor,
2. this factor is a probability distribution for the variable conditional on its ‘parents’, and
3. the associated directed graph is acyclic

then we have a *Bayesian network*

Inference in propositional probabilistic models

Key problems are:

- Compute the marginal distribution of one or more variables.
- Find the instantiation with the highest probability.

Variable elimination:

- Can replace two factors by their product.
- If a variable is in only one factor then can sum it out of the distribution by summing it out of this factor.

First-order logic

$$
\begin{tikzcd}
\text{First-order logic} \arrow[r] & \text{First-order probabilistic models} \\
\text{Propositional logic} \arrow[r] \arrow[u] & \text{Propositional probabilistic models} \arrow[u]
\end{tikzcd}
$$

Characteristics of first-order logic

- Propositions now assert:
  - Properties of objects
  - Relations between objects
- Objects are represented by ground terms
- Can *quantify over* objects
- Universally quantified formula $\approx$ template for all its ground instances.

$$
\begin{aligned}
\forall X : even(X) \rightarrow odd(s(X)) &\vdash even(0) \rightarrow odd(s(0)) \\
\forall X : even(X) \rightarrow odd(s(X)) &\vdash even(s(0)) \rightarrow odd(s(s(0))) \\
\forall X, Y : p(X, Y) \land p(Y, Z) \rightarrow p(X, Z) &\vdash p(a, b) \land p(b, c) \rightarrow p(a, c) \\
\forall X, Y : p(X, Y) \land p(Y, Z) \rightarrow p(X, Z) &\vdash p(a, e) \land p(e, c) \rightarrow p(a, c)
\end{aligned}
$$

Factor representation of universally quantified formulae

$$\forall X, Y : p(X) \rightarrow q(X, Y)$$

```
p(X) | q(X,Y) |
---- | ------ | -
  0  |    0   | 1
  0  |    1   | 1
  1  |    0   | 0
  1  |    1   | 1

p(a) | q(a,b) |          p(b) | q(b,c) |
---- | ------ | -        ---- | ------ | -
  0  |    0   | 1          0  |    0   | 1
  0  |    1   | 1    *     0  |    1   | 1   * ...
  1  |    0   | 0          1  |    0   | 0
  1  |    1   | 1          1  |    1   | 1
```

First-order models

- Restrict attention to *Herbrand* models.
- Each Herbrand model assigns TRUE or FALSE to each ground atomic formula (*atom* for short).
- So ground atoms act like binary variables.

$$
\begin{aligned}
M_1 &: p(a) = T, p(b) = F, q(a,b) = T, \ldots \\
M_2 &: p(a) = T, p(b) = T, q(a,b) = F, \ldots
\end{aligned}
$$

Inference in first-order logic

- Crucially, we can stay first-order when doing inference.
- Here's an example of *resolution*

$$\frac{\forall X, Y : p(X, Y) \vee q(X), \qquad \forall X, Y : \neg p(X, a) \vee r(Y, b)}{\forall X : q(X) \vee r(a, b)}$$

First-order probabilistic models (parfactors)

$$\begin{tikzcd}
\text{First-order logic} \arrow[r] & \text{First-order probabilistic models} \\
\text{Propositional logic} \arrow[u] \arrow[r] & \text{Propositional probabilistic models} \arrow[u]
\end{tikzcd}$$

Quantifying over random variables

Parfactors represent many factors with a single factor parameterised by logical variables (and constraints).

```
    X != Y
p(X) | q(X,Y) |
---- | ------ | ----
  0  |    0   | 0.10
  0  |    1   | 0.20   stands for the product of all its
  1  |    0   | 0.30   ground instances where X != Y:
  1  |    1   | 0.20

p(a) | q(a,b) |             p(b) | q(b,c) |
---- | ------ | ----        ---- | ------ | ----
  0  |    0   | 0.10          0  |    0   | 0.10
  0  |    1   | 0.20    *     0  |    1   | 0.20   * ...
  1  |    0   | 0.30          1  |    0   | 0.30
  1  |    1   | 0.20          1  |    1   | 0.20
```

What sort of probability distribution is defined?

- Each ground atom becomes a random variable.
- (If these are binary), the distribution is over Herbrand models: *possible worlds*.
- A parfactor with only finitely many instances can be replaced by these instances: *grounding out*.
- It's just a probability distribution: don't *necessarily* have to use logic to analyse/manipulate it.

Lifted inference in first-order probabilistic models

- Can we maintain the first-order representation when doing inference (marginalisation, maximisation) in FOPMs?
- Can we *implicitly* sum out $p(a), p(b), \ldots,$ the ground instances of $p(X)$, by summing out $p(X)$?

Lifted inference in first-order probabilistic models

- Can we maintain the first-order representation when doing inference (marginalisation, maximisation) in FOPMs?
- Can we *implicitly* sum out $p(a), p(b), \ldots,$ the ground instances of $p(X)$, by summing out $p(X)$?
- You're in the right place (wrong talk) to get the full story!
- Exploit repeated structure when it's available.

Markov logic parfactors

$2 : \neg A \vee B$

A	B	
0	0	1
0	1	1
1	0	exp(-2)
1	1	1

$2 : \forall X, Y : p(X, Y) \rightarrow q(X, Y)$

p(X,Y)	q(X,Y)	
0	0	1
0	1	1
1	0	exp(-2)
1	1	1

Markov logic distribution

Let $n_i(x)$ be the number of ground instances of clause F_i which hold in world x.

$$P(x) = Z_w^{-1} \exp\left(\sum_i w_i n_i(x)\right) \tag{1}$$

An MLN defines an *exponential-family* distribution where $(n_1(x), n_2(x), \ldots n_k(x))$ is the canonical statistic.

What's the data?

- The distribution is over possible worlds, so the obvious option is to assume *independent and identically distributed (iid)* instances of possible worlds.
- A single observed world—a relational database—can be enough, due to parameter sharing.
- For Markov logic networks we just need counts of true groundings.

First-order probabilistic models (generative)

$$
\begin{tikzcd}
\text{First-order logic} \arrow[r] & \text{First-order probabilistic models} \\
\text{Propositional logic} \arrow[u] \arrow[r] & \text{Propositional probabilistic models} \arrow[u]
\end{tikzcd}
$$

Dynamic probabilistic models

We already quantify over random variables (sometimes impliclitly) in dynamic probabilistic models:

- Time series: $\forall t : X_t \sim aX_{t-1} + \epsilon$
- Similarly in spatial statistics
- Stochastic grammars, including HMMs
- Dynamic Bayesian networks

A number of formalisms (e.g. ICL, SLPs, PRISM) generalise this approach.

The PRISM approach

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'. Suppose $P(X_i = y) = 0.3$.
- 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:

	1	2	3	4	5	6	...
X	y	n	y	y	y	n	...
Y	0	1	0	0	1	1	...
Z	4	3	1	5	5	2	...

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).
```

A joint instantiation determines a logical theory

	1	2	3	4	5	6	...
X	y	n	y	y	y	n	...
Y	0	1	0	0	1	1	...
Z	4	3	1	5	5	2	...

- This joint instantiation determines the following logical theory:

$$\mathrm{msw}('X', 1, y), \mathrm{msw}('X', 2, n), \mathrm{msw}('X', 3, y), \ldots$$
$$\mathrm{msw}('Y', 1, 0), \mathrm{msw}('Y', 2, 1), \mathrm{msw}('Y', 3, 0), \ldots$$
$$\mathrm{msw}('Z', 1, 4), \mathrm{msw}('Z', 2, 3), \mathrm{msw}('Z', 3, 1), \ldots$$

Using a fixed, arbitrary logical theory to extend a base distribution

$$
\begin{aligned}
F_1, R &\vdash ? \\
F_2, R &\vdash ? \\
F_3, R &\vdash ?
\end{aligned}
$$

- 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 fla be some first-order sentence. $\mathrm{Prob}(\mathrm{fla})$ is the probability of getting a base joint instantiation F such that $F, R \vdash \mathrm{fla}$.
- Use Closed World Assumption to get this to work.

Working with target predicates

$$
\begin{aligned}
F_1, R &\vdash t(a) \\
F_2, R &\vdash t(b) \\
F_3, R &\vdash t(a)
\end{aligned}
$$

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

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 $\mathrm{Prob}(t(a))$ 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

`hmm([a,b,a])`

$\Leftrightarrow$ `msw(out(s0),1,a)` $\wedge$ `msw(tr(s0),1,s0)` $\wedge$ *hmm(s0,[b,a])*

$\vee$ `msw(out(s0),1,a)` $\wedge$ `msw(tr(s0),1,s1)` $\wedge$ `hmm(s1,[b,a])`

Abduction: a HMM example

$\texttt{hmm([a,b,a])}$

$\Leftrightarrow \texttt{msw(out(s0),1,a)} \wedge \texttt{msw(tr(s0),1,s0)} \wedge \texttt{hmm(s0,[b,a])}$

$\vee\ \texttt{msw(out(s0),1,a)} \wedge \texttt{msw(tr(s0),1,s1)} \wedge$ *hmm(s1,[b,a])*

$\texttt{hmm(s0,[b,a])}$

$\Leftrightarrow \texttt{msw(out(s0),2,b)} \wedge \texttt{msw(tr(s0),2,s0)} \wedge \texttt{hmm(s0,[a])}$

$\vee\ \texttt{msw(out(s0),2,b)} \wedge \texttt{msw(tr(s0),2,s1)} \wedge \texttt{hmm(s1,[a])}$

Abduction: a HMM example

$\texttt{hmm([a,b,a])}$
$\Leftrightarrow \texttt{msw(out(s0),1,a)} \wedge \texttt{msw(tr(s0),1,s0)} \wedge \texttt{hmm(s0,[b,a])}$
$\vee\ \texttt{msw(out(s0),1,a)} \wedge \texttt{msw(tr(s0),1,s1)} \wedge \texttt{hmm(s1,[b,a])}$
$\texttt{hmm(s0,[b,a])}$
$\Leftrightarrow \texttt{msw(out(s0),2,b)} \wedge \texttt{msw(tr(s0),2,s0)} \wedge$ *hmm(s0,[a])*
$\vee\ \texttt{msw(out(s0),2,b)} \wedge \texttt{msw(tr(s0),2,s1)} \wedge \texttt{hmm(s1,[a])}$
$\texttt{hmm(s1,[b,a])}$
$\Leftrightarrow \texttt{msw(out(s1),1,b)} \wedge \texttt{msw(tr(s1),1,s0)} \wedge$ *hmm(s0,[a])*
$\vee\ \texttt{msw(out(s1),1,b)} \wedge \texttt{msw(tr(s1),1,s1)} \wedge \texttt{hmm(s1,[a])}$

Abduction: a HMM example

hmm([a,b,a])

$\Leftrightarrow$ msw(out(s0),1,a) $\wedge$ msw(tr(s0),1,s0) $\wedge$ hmm(s0,[b,a])

$\vee$ msw(out(s0),1,a) $\wedge$ msw(tr(s0),1,s1) $\wedge$ hmm(s1,[b,a])

hmm(s0,[b,a])

$\Leftrightarrow$ msw(out(s0),2,b) $\wedge$ msw(tr(s0),2,s0) $\wedge$ hmm(s0,[a])

$\vee$ msw(out(s0),2,b) $\wedge$ msw(tr(s0),2,s1) $\wedge$ *hmm(s1,[a])*

hmm(s1,[b,a])

$\Leftrightarrow$ msw(out(s1),1,b) $\wedge$ msw(tr(s1),1,s0) $\wedge$ hmm(s0,[a])

$\vee$ msw(out(s1),1,b) $\wedge$ msw(tr(s1),1,s1) $\wedge$ *hmm(s1,[a])*

hmm(s0,[a])

$\Leftrightarrow$ msw(out(s0),3,a) $\wedge$ msw(tr(s0),3,stop)

Abduction: a HMM example

$\texttt{hmm([a,b,a])}$

$\Leftrightarrow \texttt{msw(out(s0),1,a)} \wedge \texttt{msw(tr(s0),1,s0)} \wedge \texttt{hmm(s0,[b,a])}$

$\vee \texttt{msw(out(s0),1,a)} \wedge \texttt{msw(tr(s0),1,s1)} \wedge \texttt{hmm(s1,[b,a])}$

$\texttt{hmm(s0,[b,a])}$

$\Leftrightarrow \texttt{msw(out(s0),2,b)} \wedge \texttt{msw(tr(s0),2,s0)} \wedge \texttt{hmm(s0,[a])}$

$\vee \texttt{msw(out(s0),2,b)} \wedge \texttt{msw(tr(s0),2,s1)} \wedge \texttt{hmm(s1,[a])}$

$\texttt{hmm(s1,[b,a])}$

$\Leftrightarrow \texttt{msw(out(s1),1,b)} \wedge \texttt{msw(tr(s1),1,s0)} \wedge \texttt{hmm(s0,[a])}$

$\vee \texttt{msw(out(s1),1,b)} \wedge \texttt{msw(tr(s1),1,s1)} \wedge \texttt{hmm(s1,[a])}$

$\texttt{hmm(s0,[a])}$

$\Leftrightarrow \texttt{msw(out(s0),3,a)} \wedge \texttt{msw(tr(s0),3,stop)}$

$\texttt{hmm(s1,[a])}$

$\Leftrightarrow \texttt{msw(out(s1),2,a)} \wedge \texttt{msw(tr(s1),2,stop)}$

Computing probabilities by abduction

$\texttt{hmm([a,b,a])}$

$\Leftrightarrow \texttt{msw(out(s0),1,a)} \wedge \texttt{msw(tr(s0),1,s0)} \wedge \texttt{hmm(s0,[b,a])}$

$\vee\ \texttt{msw(out(s0),1,a)} \wedge \texttt{msw(tr(s0),1,s1)} \wedge \texttt{hmm(s1,[b,a])}$

$\texttt{hmm(s0,[b,a])}$

$\Leftrightarrow \texttt{msw(out(s0),2,b)} \wedge \texttt{msw(tr(s0),2,s0)} \wedge \texttt{hmm(s0,[a])}$

$\vee\ \texttt{msw(out(s0),2,b)} \wedge \texttt{msw(tr(s0),2,s1)} \wedge \texttt{hmm(s1,[a])}$

$\texttt{hmm(s1,[b,a])}$

$\Leftrightarrow \texttt{msw(out(s1),1,b)} \wedge \texttt{msw(tr(s1),1,s0)} \wedge \texttt{hmm(s0,[a])}$

$\vee\ \texttt{msw(out(s1),1,b)} \wedge \texttt{msw(tr(s1),1,s1)} \wedge \texttt{hmm(s1,[a])}$

$\texttt{hmm(s0,[a])}$

$\Leftrightarrow \texttt{msw(out(s0),3,a)} \wedge \texttt{msw(tr(s0),3,stop)}$

$\texttt{hmm(s1,[a])}$

$\Leftrightarrow \texttt{msw(out(s1),2,a)} \wedge \texttt{msw(tr(s1),2,stop)}$

Computing probabilities by abduction

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

`hmm(s0,[b,a])`
$\Leftrightarrow$ `msw(out(s0),2,b)` $\wedge$ `msw(tr(s0),2,s0)` $\wedge$ `hmm(s0,[a])`
$\vee$ `msw(out(s0),2,b)` $\wedge$ `msw(tr(s0),2,s1)` $\wedge$ `hmm(s1,[a])`
`hmm(s1,[b,a])`
$\Leftrightarrow$ `msw(out(s1),1,b)` $\wedge$ `msw(tr(s1),1,s0)` $\wedge$ `hmm(s0,[a])`
$\vee$ `msw(out(s1),1,b)` $\wedge$ `msw(tr(s1),1,s1)` $\wedge$ `hmm(s1,[a])`
`hmm(s0,[a])`
$\Leftrightarrow$ `msw(out(s0),3,a)` $\wedge$ `msw(tr(s0),3,stop)`
`hmm(s1,[a])`
$\Leftrightarrow$ `msw(out(s1),2,a)` $\wedge$ `msw(tr(s1),2,stop)`

Computing probabilities by abduction

Pr(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(hmm(s0,[b,a]))
$= \Pr(\text{msw(out(s0),2,b)}) \times \Pr(\text{msw(tr(s0),2,s0)}) \times \Pr(\text{hmm(s0,[a])})$
$+ \Pr(\text{msw(out(s0),2,b)}) \times \Pr(\text{msw(tr(s0),2,s1)}) \times \Pr(\text{hmm(s1,[a])})$
`hmm(s1,[b,a])`
$\Leftrightarrow$ `msw(out(s1),1,b)` $\wedge$ `msw(tr(s1),1,s0)` $\wedge$ `hmm(s0,[a])`
$\vee$ `msw(out(s1),1,b)` $\wedge$ `msw(tr(s1),1,s1)` $\wedge$ `hmm(s1,[a])`
`hmm(s0,[a])`
$\Leftrightarrow$ `msw(out(s0),3,a)` $\wedge$ `msw(tr(s0),3,stop)`
`hmm(s1,[a])`
$\Leftrightarrow$ `msw(out(s1),2,a)` $\wedge$ `msw(tr(s1),2,stop)`

Computing probabilities by abduction

Pr(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(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(hmm(s1,[b,a]))
$= \Pr(\text{msw(out(s1),1,b)}) \times \Pr(\text{msw(tr(s1),1,s0)}) \times \Pr(\text{hmm(s0,[a])})$
$+ \Pr(\text{msw(out(s1),1,b)}) \times \Pr(\text{msw(tr(s1),1,s1)}) \times \Pr(\text{hmm(s1,[a])})$
`hmm(s0,[a])`
$\Leftrightarrow$ `msw(out(s0),3,a)` $\wedge$ `msw(tr(s0),3,stop)`
`hmm(s1,[a])`
$\Leftrightarrow$ `msw(out(s1),2,a)` $\wedge$ `msw(tr(s1),2,stop)`

Computing probabilities by abduction

Pr(hmm([a,b,a]))
= Pr(msw(out(s0),1,a)) × Pr(msw(tr(s0),1,s0)) × Pr(hmm(s0,[b,a]))
+ Pr(msw(out(s0),1,a)) × Pr(msw(tr(s0),1,s1)) × Pr(hmm(s1,[b,a]))
Pr(hmm(s0,[b,a]))
= Pr(msw(out(s0),2,b)) × Pr(msw(tr(s0),2,s0)) × Pr(hmm(s0,[a]))
+ Pr(msw(out(s0),2,b)) × Pr(msw(tr(s0),2,s1)) × Pr(hmm(s1,[a]))
Pr(hmm(s1,[b,a]))
= Pr(msw(out(s1),1,b)) × Pr(msw(tr(s1),1,s0)) × Pr(hmm(s0,[a]))
+ Pr(msw(out(s1),1,b)) × Pr(msw(tr(s1),1,s1)) × Pr(hmm(s1,[a]))
Pr(hmm(s0,[a]))
= Pr(msw(out(s0),3,a)) × Pr(msw(tr(s0),3,stop))

```
hmm(s1,[a])
⇔ msw(out(s1),2,a) ∧ msw(tr(s1),2,stop)
```

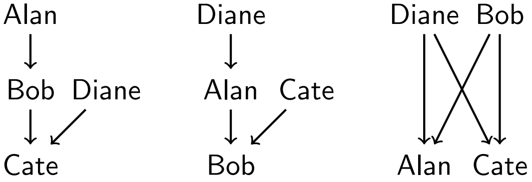
Computing probabilities by abduction

Pr(hmm([a,b,a]))
= Pr(msw(out(s0),1,a)) × Pr(msw(tr(s0),1,s0)) × Pr(hmm(s0,[b,a]))
+ Pr(msw(out(s0),1,a)) × Pr(msw(tr(s0),1,s1)) × Pr(hmm(s1,[b,a]))
Pr(hmm(s0,[b,a]))
= Pr(msw(out(s0),2,b)) × Pr(msw(tr(s0),2,s0)) × Pr(hmm(s0,[a]))
+ Pr(msw(out(s0),2,b)) × Pr(msw(tr(s0),2,s1)) × Pr(hmm(s1,[a]))
Pr(hmm(s1,[b,a]))
= Pr(msw(out(s1),1,b)) × Pr(msw(tr(s1),1,s0)) × Pr(hmm(s0,[a]))
+ Pr(msw(out(s1),1,b)) × Pr(msw(tr(s1),1,s1)) × Pr(hmm(s1,[a]))
Pr(hmm(s0,[a]))
= Pr(msw(out(s0),3,a)) × Pr(msw(tr(s0),3,stop))
Pr(hmm(s1,[a]))
= Pr(msw(out(s1),2,a)) × Pr(msw(tr(s1),2,stop))

What's the data?

- In PRISM we assume iid worlds, but we only see the instance of the target predicate true in each world
- Thus have to use EM to fit parameters

Bayesian network learning for pedigrees



- At most 2 parents per child (hurray!)
- Parameters are known, pure structure learning
- Lots of logical constraints
- Relations between objects

Some genetics

- Different variants of a gene are called *alleles*.
- You get one allele from your mother and one from your father: your *genotype*.
- Typically one observes *unordered genotypes*: don't know parental orgin of the allele.

The problem

Given

- A set G of possible pedigrees;
- a prior over pedigrees $P(g)$;
- observed marker data x_o;
- and an assumption of Mendelian segregation

Find

- $\arg\max_{g \in G} P(g|x_o)$

Defining a joint probability distribution

- Good Bayesians reduce learning to probabilistic inference.
- There will be four disjoint collections of binary variables to encode:
    1. the pedigree (g)
    2. the unobserved ordered genotypes (y)
    3. the observed and unobserved unordered genotypes ($x = (x_h, x_o)$)
    4. and the (possibly observed) auxilliary variables giving e.g. relative age information (z).
- An exponential-family distribution will be defined for $P(g, x, y, z)$ using Markov logic.

Pedigree and auxiliary variables

Pedigree variables $\text{father}(\text{bob}, \text{alice})$, $\text{mother}(\text{alice}, \text{rob})$, $\ldots$

Auxiliary variables $\text{older}(\text{bob}, \text{alice})$, $\ldots$

There are many constraints, for example:

- $\forall X, Y : \text{father}(X, Y) \rightarrow \text{older}(X, Y)$, $\forall X, Y, Z : \text{older}(X, Y) \wedge \text{older}(Y, Z) \rightarrow \text{older}(X, Z)$,...
- $\forall X, Y, Z : X \neq Y \rightarrow \neg\text{father}(X, Z) \vee \neg\text{father}(Y, Z)$, $\ldots$

Ordered genotype variables

Ordered genotype variables $\mathrm{pat}(\mathrm{bob}, \mathrm{a2}), \mathrm{mat}(\mathrm{alice}, \mathrm{a4}), \ldots$

- pat and mat are functional relations:
  $\forall X, A, B : A \neq B \rightarrow \neg\mathrm{mat}(X, A) \vee \neg\mathrm{mat}(X, B)$,
  $\forall X : \exists A : \mathrm{pat}(X, A), \ldots$
- Homozygous inheritance:
  $\forall X, Y, A : \mathrm{pat}(X, A) \wedge \mathrm{mat}(X, A) \wedge \mathrm{father}(X, Y) \rightarrow \mathrm{pat}(Y, A)$.

Unordered genotype variables

Unordered genotype variables $\mathrm{genotype}(\mathrm{bob}, \mathrm{a1}, \mathrm{a2}), \ldots$

- $\forall X, A, B : \mathrm{genotype}(X, A, B) \leftrightarrow (\mathrm{pat}(X, A) \wedge \mathrm{mat}(X, B)) \vee (\mathrm{mat}(X, A) \wedge \mathrm{pat}(X, B))$

An example possible world

As is typical only true ground atoms are listed:

```
father(m1,m2)     pa(m2,a1)
older(m1,m2)      pa(f1,a3)
mother(f1,m1)     ma(m1,a1)
older(f1,m1)      ma(m2,a2)
mother(f1,m2)     ma(f1,a1)
older(f1,m2)      genotype(m1,a1,a1)
pa(m1,a1)         genotype(m2,a1,a2)
                  genotype(f1,a1,a3)
```

Penalty for heterozygosity

Assuming Mendelian segregation

$$
\begin{aligned}
& -\log 0.5 : \forall X, Y, A, B : \\
& \quad \neg(\mathrm{father}(X, Y) \wedge \mathrm{pat}(X, A) \wedge \mathrm{mat}(X, B) \wedge A \neq B)
\end{aligned}
$$

Encoding population frequencies

$$-\log p_i : \forall Y : \exists X : \mathrm{father}(X, Y) \vee \neg\mathrm{pat}(Y, a_i)$$

$$-\log p_i : \forall Y : \exists X : \mathrm{mother}(X, Y) \vee \neg\mathrm{mat}(Y, a_i)$$

Priors on pedigrees

For example:

$$30 : \forall X, Y, Z : mother(X, Y) \wedge father(Y, Z) \rightarrow \neg mother(X, Z)$$

$$20 : \forall X, Y, Z : mother(X, Z) \wedge father(Y, Z) \rightarrow \neg related(X, Y)$$

Just a different way of writing down the prior of (Sheehan & Egeland, 2007).

Incorporating evidence

- Just add in the appropriate ground atoms, e.g.
  - $\text{genotype}(\text{bob}, \text{a1}, \text{a2})$
  - $\text{father}(\text{john}, \text{robin})$
- thus ruling out all worlds in which these are not true.
- The intelligent approach is to 'propagate' the evidence to specialise the general-purpose logical knowledge base.

An simple example

With a uniform prior on pedigrees and this (unordered) genotype data:

```
genotype(m1,a1,a2)  genotype(f1,a2,a2)
genotype(m2,a1,a2)  genotype(f2,a1,a2)
genotype(m3,a2,a2)  genotype(f3,a2,a2)
genotype(m4,a1,a2)  genotype(f4,a1,a2)
genotype(m5,a1,a1)  genotype(f5,a1,a1)
mother(f1,f3)
```

A result

Grounding out and using the exact weighted MAX-SAT solver (minimaxsat1.0, Heras *et al*) took 30 seconds to establish that this 'possible world' is the most probable:

`father(m2,m4)`	`mother(f1,m1)`	`pa(m1,a1)`	`ma(m1,a2)`
`father(m2,f1)`	`mother(f1,m3)`	`pa(m2,a1)`	`ma(m2,a2)`
`father(m2,f5)`	`mother(f1,m4)`	`pa(m3,a2)`	`ma(m3,a2)`
`father(m4,m1)`	`mother(f1,f3)`	`pa(m4,a1)`	`ma(m4,a2)`
`father(m4,m3)`	`mother(f2,m2)`	`pa(m5,a1)`	`ma(m5,a1)`
`father(m4,m5)`	`mother(f2,f1)`	`pa(f1,a2)`	`ma(f1,a2)`
`father(m4,f3)`	`mother(f2,f4)`	`pa(f2,a1)`	`ma(f2,a2)`
`father(m4,f4)`	`mother(f2,f5)`	`pa(f3,a2)`	`ma(f3,a2)`
	`mother(f5,m5)`	`pa(f4,a2)`	`ma(f4,a1)`
		`pa(f5,a1)`	`ma(f5,a1)`

Another result

Took 145s.

```
genotype(m1,a1,a2)  genotype(f1,a2,a2)
genotype(m2,a1,a2)  genotype(f2,a1,a2)
genotype(m3,a2,a2)  genotype(f3,a2,a2)
genotype(m4,a1,a2)  genotype(f4,a1,a2)
genotype(m5,a1,a1)  genotype(f5,a1,a1)
```

If a total order is added, this reduces to 0.076s.

Not there yet!

- This is a nice way of solving $\arg\max_{g,y} P(x,y|g)P(g)$, but we actually want to solve $\arg\max_g P(x|g)P(g) = \arg\max_g \sum_y P(x,y|g)P(g)$.
- Domingos's group (University of Washington) working on this right now.