# ‘Proof-theoretical’ PLL models

## Strategy

- "Make everything as simple as possible, but not simpler." (Einstein)

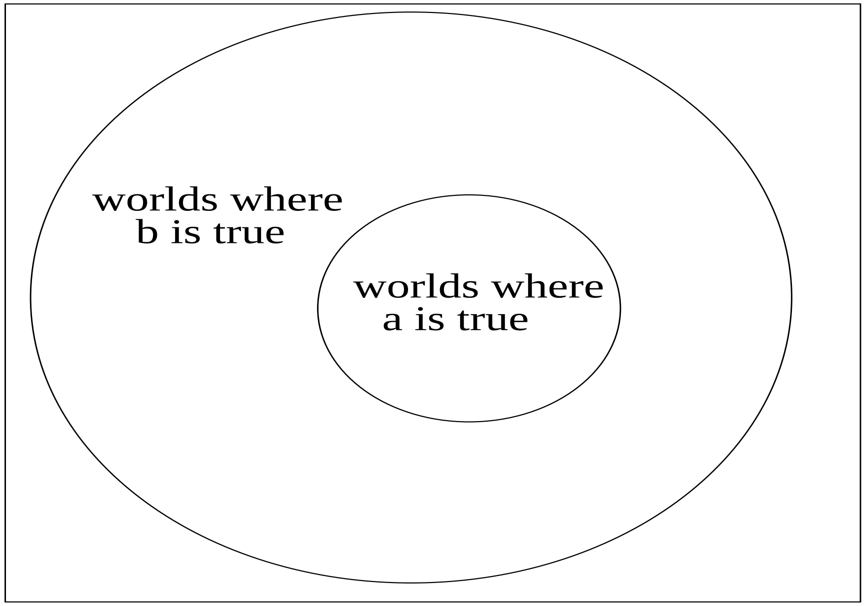
- Once you understand the models, parameter estimation (at least) is fairly easy to understand.

- Some demos

## Propositional logic and sets

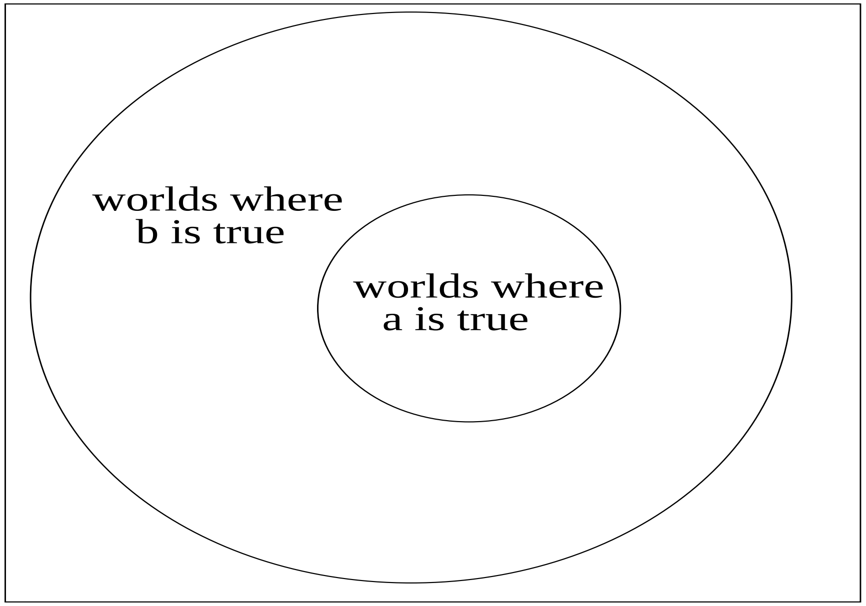
- Propositional logic is basically (naïve) set theory

- (Informally), $a \longrightarrow b$ asserts a subset relation



## Probability and sets

- Probability measures the size of a set (Kolmogorov's axioms)
- So (informally) $a \to b$ means that $P(a) \leq P(b)$



## The trouble with probability

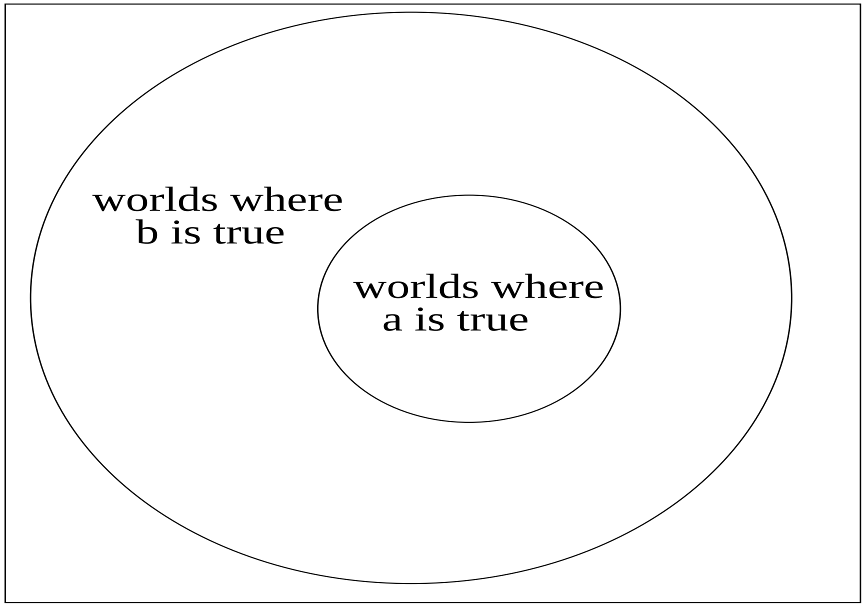
- If $a \rightarrow b$, $P(b)$ might be more than $P(a)$, since $b$ might 'happen for other reasons'.

- Special case of Boole's result of 1854.

- So far, we have been following Boole's approach: map logical and probabilistic reasoning to set theory.

## Using Halpern's logic

- Frequently, $a \rightarrow b$ really means $w(a \rightarrow b) = 1$
- We have $w(a \rightarrow b) = 1 \models w(a) \leq w(b)$



## Logic programming semantics

- Cue high-powered demo!

- First-order semantics don't give you the answers you want, so move the goalposts.

- If you can't prove it to be true, it ain't: 'minimal' model semantics

- We'll use this trick to compute exact probabilities (rather than mere bounds).

## Shapiro's "Logic programs with uncertainties"

- Shapiro worked in the more general setting of 'certainties' and 'certainty functions' (see MYCIN, PROSPECTOR),

- But it can be specialised to probabilities, with an independence assumption.

- BASIC IDEA: If we can prove, say, that $P(b) \geq p$, but no more, then infer that $P(b) = p$.

- "Our goal is to define the analog of the minimal model of standard logic program" - Shapiro (IJCAI '83)

## Digression: Before probability ruled with (AI) world

"Since the semantics defined for logic programs with uncertainties proves to be insensitive to the particular certainty functions chosen, we tend to believe that no theoretical argument will be able to decide between the reasonable alternatives. Sociologically speaking, this conjecture has proved itself so far. Each school of expert systems is using its own particular way of computing certainties, with no noticeable difference in the validity of their results.

So we pick our pick. ..." - Shapiro

## The trouble with conjunction

- Recall that $a \to b \Rightarrow P(a) \leq P(b)$
- So $a \wedge c \to b \Rightarrow P(a \wedge c) \leq P(b)$
- But $P(a \wedge c)$ not determined by $P(a)$ and $P(c)$ ...
- ... since we need to know $P(a \vee c)$ too

## Assuming independence

- The usual trick: when things get complicated, assume independence: $P(a \wedge c) = P(a)P(c)$

- "The events whose probabilities are given are to be regarded as independent of any connexion but such as is either expressed, or necessarily implied, in the data" - Boole

- If we cannot establish dependence, assume independence

- Another kind of 'minimal' model.

## A simple probabilistic logic system

- Cue demo of a version of Shapiro's interpreter.
- Since lower bounds (not actual probabilities) are computed, this isn't inconsistent:

```
| ?- solve(b,0.1,X).
X = 0.15 ? ;
X = 0.4 ? ;
```

## Poole's Probabilistic Horn Abduction (later Independent Choice Logic) (1991-)

Basically . . .

- Define a possible world by making a series of independent choices of what's true.
- These choices have probabilities, so we get a distribution over possible worlds.
- Also have formulae which are true in all possible worlds; use these to get more probabilities.

## An example from David Poole

Let $\mathcal{C} = \{\{a_1, a_2, a_3\}, \{b_1, b_2\}\}$ and $\mathcal{F} = \{c \leftarrow a_1 \wedge b_1, c \leftarrow a_3 \wedge b_2, d \leftarrow a_1, d \leftarrow \neg a_2 \wedge b_1, e \leftarrow c, e \leftarrow \neg d\}$
There are 6 possible worlds with the following truth assignments:

$$
\begin{array}{llrrrrrrrr}
w_{a_1,b_1} & \models & a_1 & \neg a_2 & \neg a_3 & b_1 & \neg b_2 & c & d & e \\
w_{a_2,b_1} & \models & \neg a_1 & a_2 & \neg a_3 & b_1 & \neg b_2 & \neg c & \neg d & e \\
w_{a_3,b_1} & \models & \neg a_1 & \neg a_2 & a_3 & b_1 & \neg b_2 & \neg c & d & \neg e \\
w_{a_1,b_2} & \models & a_1 & \neg a_2 & \neg a_3 & \neg b_1 & b_2 & \neg c & d & \neg e \\
w_{a_2,b_2} & \models & \neg a_1 & a_2 & \neg a_3 & \neg b_1 & b_2 & \neg c & \neg d & e \\
w_{a_3,b_2} & \models & \neg a_1 & \neg a_2 & a_3 & \neg b_1 & b_2 & c & \neg d & e
\end{array}
$$

Why do you think $w_{a_2,b_1} \models \neg c$ ?

## Independent Choice Logic

An *independent choice logic theory* is a pair $(\mathcal{C}, \mathcal{F})$, where

- $\mathcal{C}$, the *choice space*, is a set of non-empty sets of ground atomic formulae, such that if $\chi_1 \in \mathcal{C}, \chi_2 \in \mathcal{C}$, and $\chi_1 \neq \chi_2$ then $\chi_1 \cap \chi_2 = \emptyset$. An element of $\mathcal{C}$ is called an *alternative*. An element of an alternative is called an atomic choice.

- $\mathcal{F}$, the *facts*, is an acyclic logic program such that no atomic choice unifies with the head of any rule.

## ICL semantics

- Each total choice $R(\tau)$ specifies a possible world.

- For any atom $a$ the "base logic" should be such that either $F, R(\tau) \vdash a$ or $F, R(\tau) \vdash \neg a$.

- This is what allows us to compute exact probabilities.

- Without this there might be worlds where we don't know whether $a$ holds or not.

- One appropriate choice of base logic is: logic programs with unique stable models.

## Independence in Probabilistic Horn abduction

- "Our final assumption is to assume that logical dependencies impose the only statistical dependencies on the hypotheses" (Poole)

- "The events whose probabilities are given are to be regarded as independent of any connexion but such as is either expressed, or necessarily implied, in the data" (Boole)

- "Should the data, beside informing us of the probabilities of events, further assign among them any explicit connexion, such connexion must be logically expressed" (Boole)

## Probabilities of atoms

- An *explanation* of a closed formula $g$ is a set of hypotheses which, together with $F$ (the facts) entail $g$.

- The probability of an explanation is just the product of the probabilities of the hypotheses in it.

- The probability of any $g$ is just the sum of the probabilities of its explanations

- We'll see a connection between explanations and proofs later.

## ICL in a nutshell

- Each alternative is a discrete random variable.
- Each total choice (joint instantiation) determines a 'world'.
- The probability of a world is just the product of the probabilities of the choices which constitute it.
- For any ground formula, to get its probability just tot up the probabilities of all the worlds in which it is true. ("marginalise away the world")

## Using Halpern's syntax

$\mathcal{C} = \{\{a_1, a_2, a_3\}, \{b_1, b_2\}\}$
$\mathcal{F} = \{c \leftarrow a_1 \wedge b_1, c \leftarrow a_3 \wedge b_2, d \leftarrow a_1, d \leftarrow \neg a_2 \wedge b_1, e \leftarrow c, e \leftarrow \neg d\}$

Here's a partial formalisation:

$w(a_1) = 0.2, w(a_2) = 0.5, w(a_3) = 0.3$
$w(a_1 \wedge a_3) = 0$
$w(a_1 \wedge b_1) = w(a_1)w(b_1)$
$w(c \leftarrow a_1 \wedge b_1) = 1$

## From worlds to proofs

- Instead of a set of independent choices determining a world ...

- ... we can have these choices determine a proof.

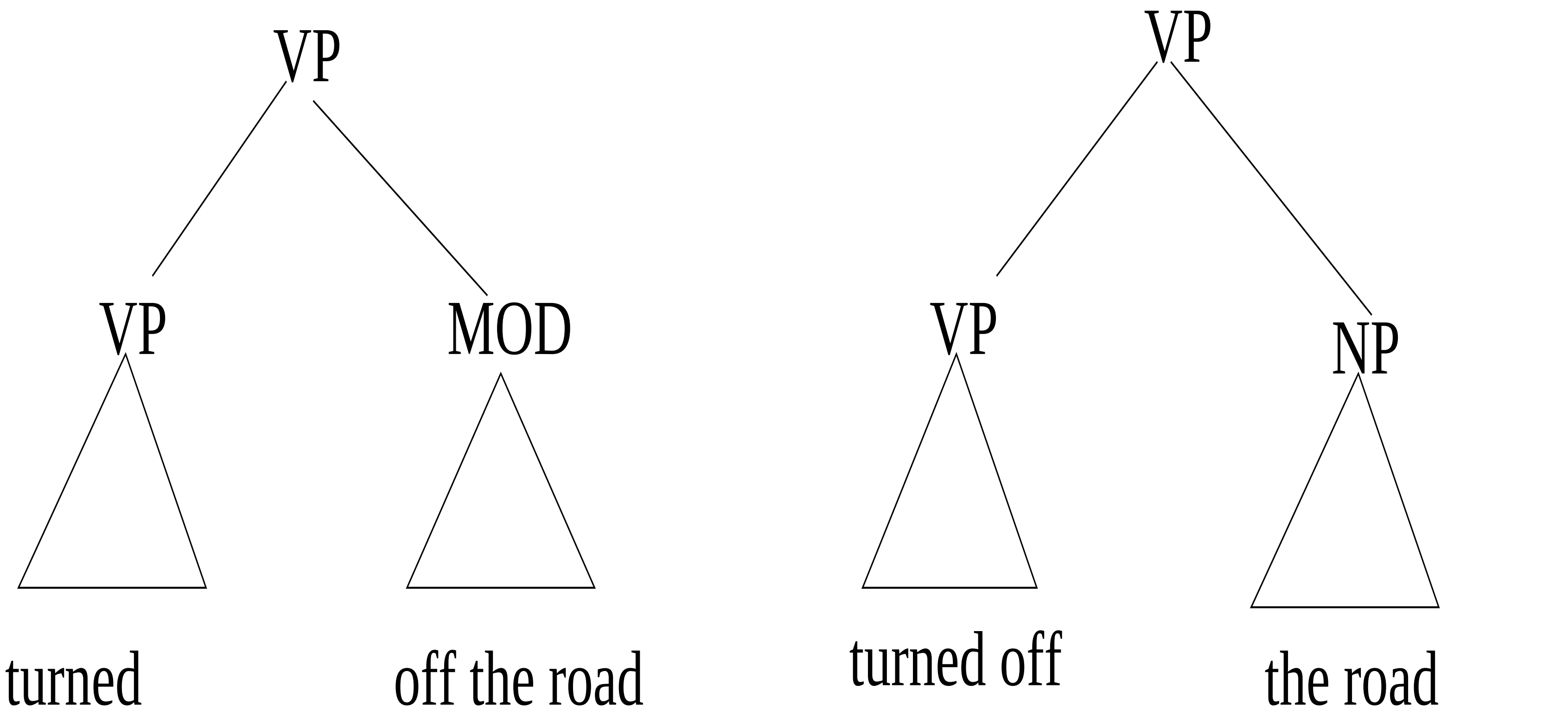
## A simple context-free grammar

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| S | $\rightarrow$ | NP VP | MOD | $\rightarrow$ | PRP NP | VP | $\rightarrow$ | V |
| NP | $\rightarrow$ | N | VP | $\rightarrow$ | V PRT | VP | $\rightarrow$ | VP NP |
| NP | $\rightarrow$ | DET N | VP | $\rightarrow$ | VP MOD | | | |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| N | $\rightarrow$ | i | N | $\rightarrow$ | road | N | $\rightarrow$ | radio |
| V | $\rightarrow$ | turned | PRP | $\rightarrow$ | off | PRT | $\rightarrow$ | off |
| DET | $\rightarrow$ | the | | | | | | |

This grammar permits *ambiguity*.

## The case of the disappearing road



## Some parses in detail

For the record . . .

S $\rightarrow$ NP VP $\rightarrow$ N VP $\rightarrow$ i VP $\rightarrow$ i VP MOD $\rightarrow$ i V MOD $\rightarrow$ i turned MOD $\rightarrow$ i turned PRP NP $\rightarrow$ i turned off NP $\rightarrow$ i turned off DET N $\rightarrow$ i turned off the N $\rightarrow$ i turned off the road

S $\rightarrow$ NP VP $\rightarrow$ N VP $\rightarrow$ i VP $\rightarrow$ i VP NP $\rightarrow$ i VP PRT NP $\rightarrow$ i turned PRT NP $\rightarrow$ i turned off NP $\rightarrow$ i turned off DET N $\rightarrow$ i turned off the N $\rightarrow$ i turned off the road

## A stochastic context-free grammar (SCFG)

| | | |
|---|---|---|
| 1.0 : S $\rightarrow$ NP VP | 1.0 : MOD $\rightarrow$ PRP NP | 0.2 : VP $\rightarrow$ V |
| 0.2 : NP $\rightarrow$ N | 0.2 : VP $\rightarrow$ V PRT | 0.2 : VP $\rightarrow$ VP NP |
| 0.8 : NP $\rightarrow$ DET N | 0.4 : VP $\rightarrow$ VP MOD | |

| | | |
|---|---|---|
| 0.5 : N $\rightarrow$ i | 0.3 : N $\rightarrow$ road | 0.2 : N $\rightarrow$ radio |
| 1.0 : V $\rightarrow$ turned | 1.0 : PRP $\rightarrow$ off | 1.0 : PRT $\rightarrow$ off |
| 1.0 : DET $\rightarrow$ the | | |

- Each derivation now gets a probability: the product of its productions.

- The probability of each sentence is just the sum of the derivation that generate it.

## Using a SCFG to choose the more likely parse

S $\stackrel{1.0}{\rightarrow}$ NP VP $\stackrel{0.2}{\rightarrow}$ N VP $\stackrel{0.5}{\rightarrow}$ i VP $\stackrel{0.4}{\rightarrow}$ i VP MOD $\stackrel{0.2}{\rightarrow}$ i V MOD $\stackrel{1.0}{\rightarrow}$ i turned MOD $\stackrel{1.0}{\rightarrow}$ i turned PRP NP $\stackrel{1.0}{\rightarrow}$ i turned off NP $\stackrel{0.8}{\rightarrow}$ i turned off DET N $\stackrel{1.0}{\rightarrow}$ i turned off the N $\stackrel{0.3}{\rightarrow}$ i turned off the road

$P = 1.0 \times 0.2 \times 0.5 \times 0.4 \times 0.2 \times 1.0 \times 1.0 \times 1.0 \times 0.8 \times 1.0 \times 0.3 = 0.001920$

S $\stackrel{1.0}{\rightarrow}$ NP VP $\stackrel{0.2}{\rightarrow}$ N VP $\stackrel{0.5}{\rightarrow}$ i VP $\stackrel{0.2}{\rightarrow}$ i VP NP $\stackrel{0.2}{\rightarrow}$ i VP PRT NP $\stackrel{1.0}{\rightarrow}$ i turned PRT NP $\stackrel{1.0}{\rightarrow}$ i turned off NP $\stackrel{0.8}{\rightarrow}$ i turned off DET N $\stackrel{1.0}{\rightarrow}$ i turned off the N $\stackrel{0.3}{\rightarrow}$ i turned off the road

$P = 1.0 \times 0.2 \times 0.5 \times 0.2 \times 0.2 \times 1.0 \times 1.0 \times 0.8 \times 1.0 \times 0.3 = 0.000960$

## Are rules the best features?

- Why should the rules used in a parse (analysis) determine its probability?

- A more general approach is to have the 'features' of a syntactic analysis.

- And so to Riezler's Probabilistic Constraint Logic Programming and MAXENT methods.

- Also Dehaspe's MACCENT: clauses are binary features

## Parameter estimation for SCFGs

Given grammar rules and sentences, estimate parameters

**Annotated data** If sentences annotated with the correct parses then it's just counting

**Unannotated data** Otherwise use EM (Inside-Outside) algorithm where the 'true' parse is the missing data.

## SCFGs and ICL

First choose a logical encoding for the parses of our grammar. Here's a fragment:

$$
\begin{array}{l}
\forall x, y : s(s'(x, y)) \leftarrow np(x) \wedge vp(y) \\
\forall x : vp(x) \leftarrow v(x) \\
\forall x, y : vp(vp'(x, y)) \leftarrow v(x) \wedge prt(y) \\
\forall x, y : vp(vp'(x, y)) \leftarrow vp(x) \wedge np(y) \\
\forall x, y : vp(vp'(x, y)) \leftarrow vp(x) \wedge mod(y) \\
n(n'(i)) \qquad n(n'(road)) \qquad n(n'(radio))
\end{array}
$$

In Prolog . . .

## SCFGs and ICL - Adding choices

$$\mathcal{C} = \{\{vp_1, vp_2, vp_3, vp_4\}, \{n_1, n_2, n_3\}\}$$

$$\mathcal{F} = \begin{array}{l} \forall x, y : s(s'(x,y)) \leftarrow np(x) \wedge vp(y) \\ \forall x : vp(x) \leftarrow vp_1 \wedge v(x) \\ \forall x, y : vp(vp'(x,y)) \leftarrow vp_2 \wedge v(x) \wedge prt(y) \\ \forall x, y : vp(vp'(x,y)) \leftarrow vp_3 \wedge vp(x) \wedge np(y) \\ \forall x, y : vp(vp'(x,y)) \leftarrow vp_4 \wedge vp(x) \wedge mod(y) \\ n(n'(i)) \leftarrow n_1 \qquad n(n'(road)) \leftarrow n_2 \qquad n(n'(radio)) \leftarrow n_3 \end{array}$$

Can map between choosing a proof and choosing a possible world. But we're not quite there yet . . .

## Extending the choice space

- But there is no limit to how often we will have to choose between the different VP expansions.

- We need an infinite supply of iid random variables to make these probabilistic choices.

- Cue PRISM

## PRISM and 'Distribution Semantics'

"When a joint distribution $P_F$ is given to a set $F$ of facts in a logic program $DB = F \cup R$ where $R$ is a set of rules, we can further extend it to a joint distribution $P_{DB}$ over the set of least models of $DB$." (Sato - ICLP '95)

- Sato does the necessary work for this to make sense when $F$ is infinite.

## PRISM at work

- Let's jump straight to the PRISM program for this SCFG

- This (and other examples) come with the PRISM distribution, available at `http://sato-www.cs.titech.ac.jp/prism/`

$$
\begin{array}{lcl}
0.4 : S & \rightarrow & SS \\
0.5 : S & \rightarrow & a \\
0.1 : S & \rightarrow & b
\end{array}
$$

## PRISM

- Random variables defined by `msw/2` are sampled on demand.
- For this to work we need an unlimited supply of random variables.
- Choosing a rule is like finding which alternate is true.
- Sampling a possible world is like choosing a proof

## Observables and abducibles

```
bloodtype(a) :- (genotype(a,a) ; genotype(a,o) ; genotype(o,a)).
bloodtype(b) :- (genotype(b,b) ; genotype(b,o) ; genotype(o,b)).
bloodtype(o) :- genotype(o,o).
bloodtype(ab) :- (genotype(a,b) ; genotype(b,a)).

genotype(X,Y) :- msw(gene,X),msw(gene,Y).
```

- If we observe genotypes, then maximum likelihood estimation is just counting.

- If we observe bloodtypes, then the genotypes are ‘hidden’ data—so use EM.

## The graphical EM algorithm

- In each iteration we compute the *expected* number of each genotype, given the observed bloodtypes, and current parameters.

- The mapping: *current parameters* $\rightarrow$ *expected numbers* is the same for each iteration.

- The gEM algorithm effectively *constructs* and *stores* this mapping once, rather than on each iteration.

## gEM and chart parsing

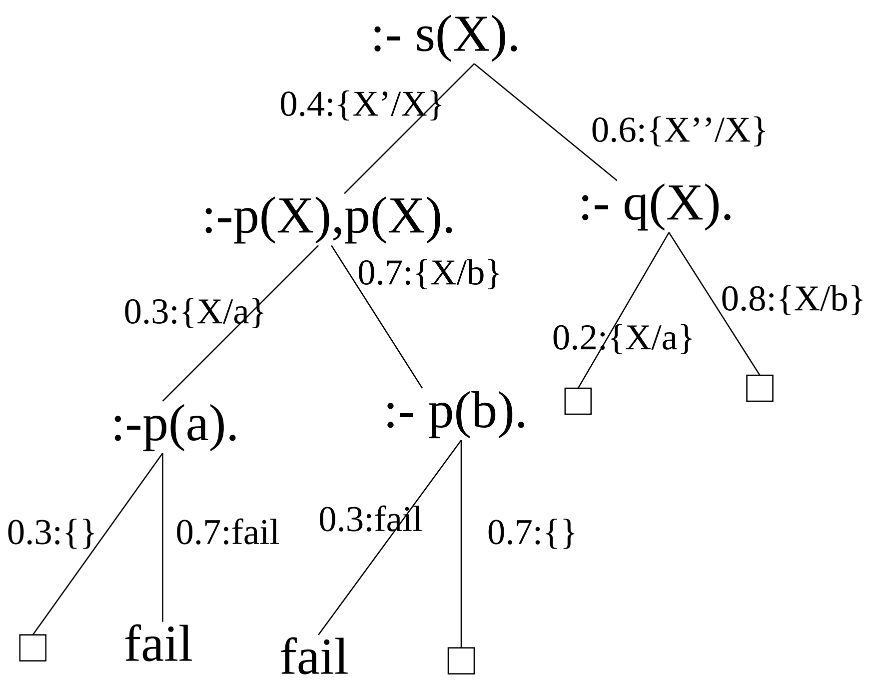
- gEM uses tabulation to store proofs (aka explanations)
- This is a generalisation of the idea of chart parsing.
- To learn the parameters of a SCFG, PRISM builds charts for input sentences *once*.
- Charts are then used to compute the EM updates.
- This can use up a lot of memory, of course.

## Stochastic Logic Programs

- Explicitly introduced by Muggleton in ’96 as a generalisation of SCFGs
- Basically Prolog with a probabilistic clause selector.
- Clearly very similar to PRISM programs.

# SLPs in a nutshell

```
0.4:s(X) :- p(X), p(X).        0.3:p(a).     0.2:q(a).
0.6:s(X) :- q(X).              0.7:p(b).     0.8:q(b).
```




$p_\lambda(s(a)) = (0.4 \times 0.3 \times 0.3 + 0.6 \times 0.2)/0.832 = 0.1875$

$p_\lambda(s(b)) = (0.4 \times 0.7 \times 0.7 + 0.6 \times 0.8)/0.832 = 0.8125$

## Generating and failing

- The 0.832 on the previous slide is the dreaded $Z$ normalising constant found in exponential-family models ($\sim$ log-linear, MAXENT models).

- In SCFGs $Z = 1$ always.

- If $Z$ not guaranteed to be 1, have to do some extra work to use EM.

## Parameter estimation for SLPs

- Keeping with our running SLP example (also found in Cussens, MLJ '01),

- suppose we have data of 7 $s(a)$ atoms and 5 $s(b)$ atoms.

- Consider how these might have been generated from our SLP (using the unknown 'true' parameters) ...

## Failure-adjusted maximisation

- For a given observed atom, say $s(a)$, there is the hidden data consisting of the rules that were used to produce it . . .

- . . . and also the hidden data consisting of an unknown number of failures which occurred before $s(a)$ got generated

- By including both sorts of hidden data, we can used EM to do maximum likelihood estimation for SLPs

## FAM implementation

- Cue FAM demo,

- FAM will be implemented in PRISM 1.8

- To do the computations associated with the failures, PRISM 1.8 produces an auxiliary program which succeeds when the original one fails.

- Nice application of logic programming technique to solve a statistical problem.

## Structure learning

- Inductive Logic Programming (ILP) is the obvious option.
- Dehaspe's MACCENT (ILP'97) applies the general MAX-ENT model learning approach of Della Pietra et al ('97).
- Dehaspe's features are clauses
- We will see a lot more about structure learning when we consider MCMC approaches