Relations, generalizations and the reference-class problem: A logic programming / Bayesian perspective

David Poole

Joint work with Michael Chiang.

University of British Columbia

☞

Overview

- Learning from relations: the reference class problem
- Inductive logic programming
- Probabilities and logic programs: aggregating vs quantifying
- Hierarchical priors
- Putting it all together

Relational Learning: Example

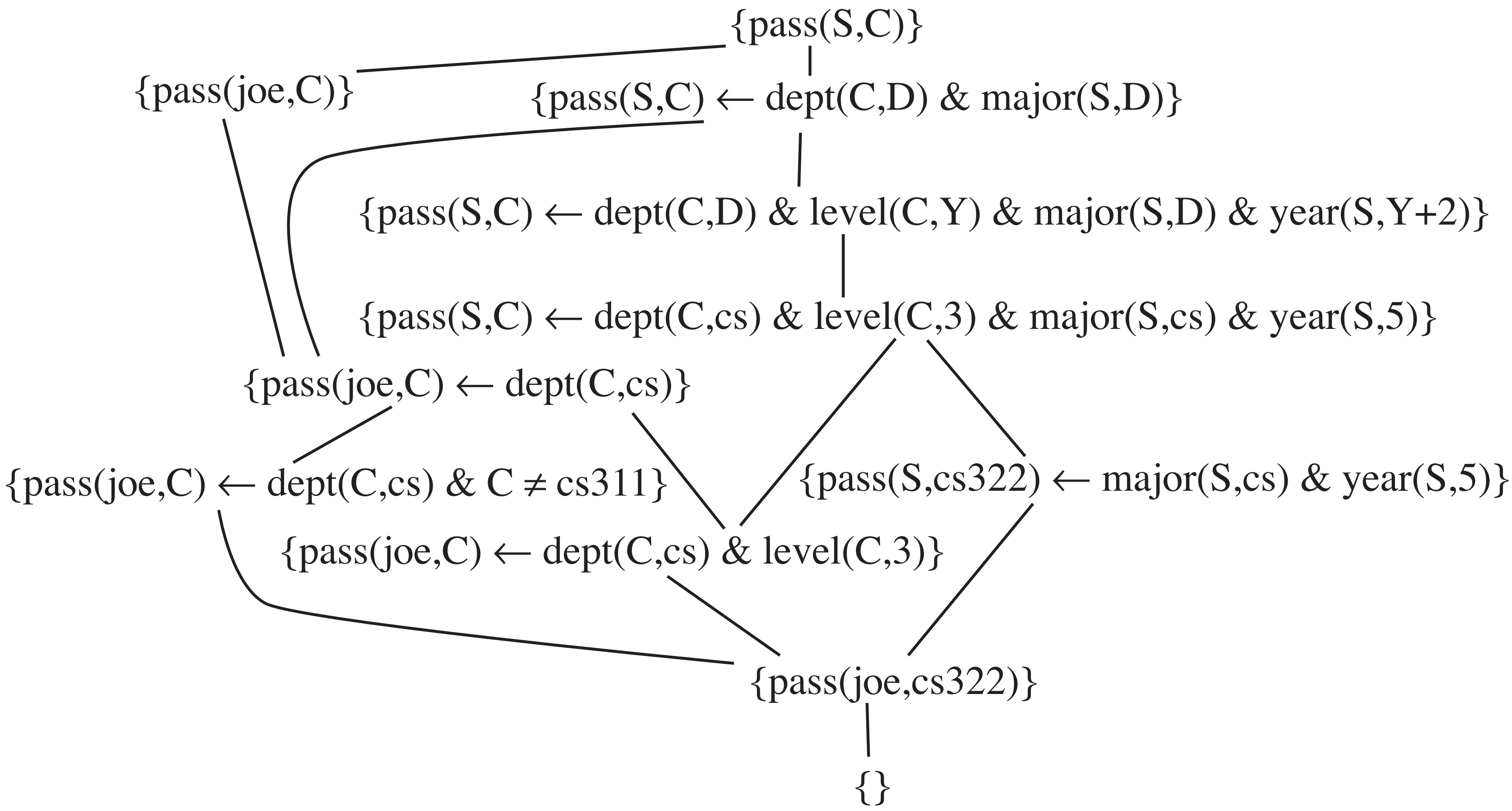
- Given a database containing the relations:
  - $grade(Student, Course, Grade)$
  - $dept(Course, Department)$
  - $level(Course, Year)$
  - $major(Student, Department)$
  - $year(Student, Year)$
  - $\ldots$
- Predict the value (distribution) of G for:

$$grade(joe, cs322, G)$$

Where do the probabilities come from?

- To get probabilities from data, you need to aggregate.
- To get distribution for $grade(joe, cs322, G)$
  - distribution of grades of Joe over all courses
  - distribution of grades for all students in CS322
  - distribution of grades for all students over all courses
- $\Longrightarrow$ reference class problem
  - as you generalize you get better statistics, but less specificity
  - conventional wisdom: choose narrowest reference class with adequate statistics

Inductive Logic Programming



Adding probabilities to logic programs

Simplest way:

- add exogenous "choices of nature" that have probabilities
- logic programs give consequences of choices
  - logic programs have standard syntax and semantics
- it suffices to have independent choices
- these can represent any belief network:
  local transformation that doesn't increase the number of parameters

Independent Choice Logic

- **C**, the choice space is a set of alternatives.
  An alternative is a set of atomic choices.
  An atomic choice is a ground atomic formula.
  An atomic choice can only appear in one alternative.

- **F**, the facts is an acyclic logic program.
  No atomic choice unifies with the head of a rule.

- P_0 a probability distribution over alternatives:

$$\forall A \in \mathbf{C} \sum_{a \in A} P_0(a) = 1.$$

Meaningless Example

$$\mathbf{C} = \{\{c_1, c_2, c_3\}, \{b_1, b_2\}\}$$

$$\begin{aligned} \mathbf{F} = \{\ & f \leftarrow c_1 \wedge b_1, & f \leftarrow c_3 \wedge b_2, \\ & d \leftarrow c_1, & d \leftarrow \overline{c}_2 \wedge b_1, \\ & e \leftarrow f, & e \leftarrow \overline{d}\} \end{aligned}$$

$P_0(c_1) = 0.5 \quad P_0(c_2) = 0.3 \quad P_0(c_3) = 0.2$

$P_0(b_1) = 0.9 \quad P_0(b_2) = 0.1$

Semantics of ICL

- A total choice is a set containing exactly one element of each alternative in **C**.
- For each total choice τ there is a possible world w_τ.
- Proposition f is true in w_τ (written $w_\tau \models f$) if f is true in the (unique) stable model of $\mathbf{F} \cup \tau$.
- The probability of a possible world w_τ is

$$\prod_{a \in \tau} P_0(a).$$

- The probability of a proposition f is the sum of the probabilities of the worlds in which f is true.

Meaningless Example: Semantics

There are 6 possible worlds:

$$
\begin{array}{lllllll}
w_1 & \models & c_1 & b_1 & f & d & e \\
w_2 & \models & c_2 & b_1 & \overline{f} & \overline{d} & e \\
w_3 & \models & c_3 & b_1 & \overline{f} & d & \overline{e} \\
w_4 & \models & c_1 & b_2 & \overline{f} & d & \overline{e} \\
w_5 & \models & c_2 & b_2 & \overline{f} & \overline{d} & e \\
w_6 & \models & c_3 & b_2 & f & \overline{d} & e
\end{array}
\qquad
\begin{array}{l}
P(w_1) = 0.45 \\
P(w_2) = 0.27 \\
P(w_3) = 0.18 \\
P(w_4) = 0.05 \\
P(w_5) = 0.03 \\
P(w_6) = 0.02
\end{array}
$$

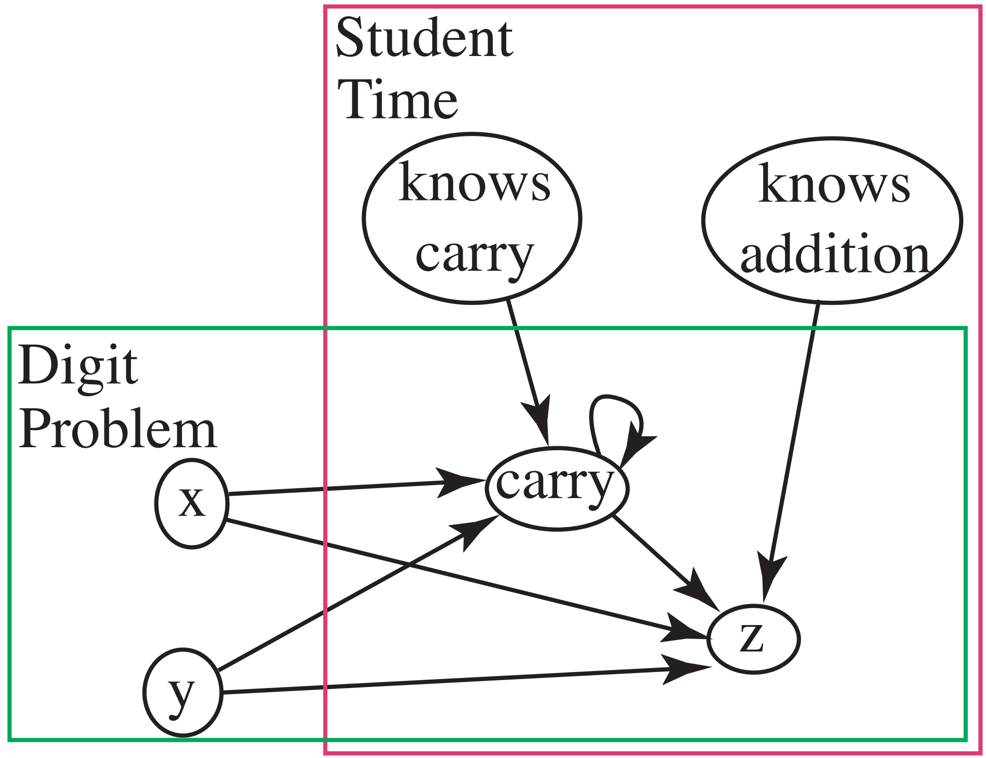
$$P(e) = 0.45 + 0.27 + 0.03 + 0.02 = 0.77$$

Logical variables ≡ plates

- In logic programming, logical variables are universally quantified
- A program means its grounding; multiple instances, one for each individual
- Buntine's plates: parametrized parts of belief networks

Example: Multi-digit addition

$$
\begin{array}{cccccc}
 & x_{j_x} & \cdots & x_2 & x_1 \\
+ & y_{j_z} & \cdots & y_2 & y_1 \\
\hline
 & z_{j_z} & \cdots & z_2 & z_1
\end{array}
$$



Rules for multi-digit addition

$$
\begin{aligned}
z(D,P,S,T) = V \leftarrow \\
&x(D,P) = Vx \wedge \\
&y(D,P) = Vy \wedge \\
&carry(D,P,S,T) = Vc \wedge \\
&knowsAddition(S,T) \wedge \\
&noMistake(D,P,S,T) \wedge \\
&V \text{ is } (Vx + Vy + Vc) \text{ div } 10.
\end{aligned}
$$

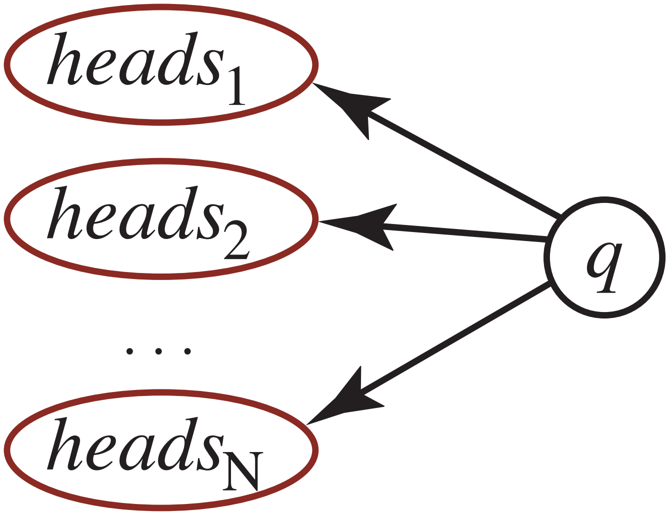
$$
\begin{aligned}
z(D,P,S,T) = V \leftarrow \\
&knowsAddition(S,T) \wedge \\
&mistake(D,P,S,T) \wedge \\
&selectDig(D,P,S,T) \\
&\quad = V.
\end{aligned}
$$

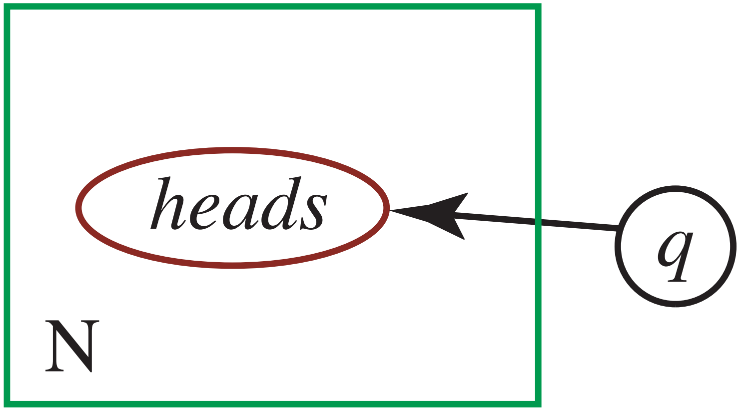
$$\forall DPST\{noMistake(D,P,S,T), mistake(D,P,S,T)\} \in \mathbf{C}$$

$$\forall DPST\{selectDig(D,P,S,T) = V \mid V \in \{0..9\}\} \in \mathbf{C}$$

Plates for Learning

Example: parameter estimation for probability of heads
(from [Buntine, JAIR, 94])





ICL Version of Parameter Learning

$$heads(C) \leftarrow$$

$$turns\_heads(C, \Theta) \wedge prob\_heads(\Theta).$$

$$tails(C) \leftarrow$$

$$turns\_tails(C, \Theta) \wedge prob\_heads(\Theta).$$

$$\forall C \forall \Theta \; \{turns\_heads(C, \Theta), turns\_tails(C, \Theta)\} \in \mathbf{C}$$

$$\{prob\_heads(\theta) : \theta \in [0, 1]\} \in \mathbf{C}$$

$$Prob(turns\_heads(C, \theta)) = \theta$$

$$Prob(turns\_tails(C, \theta)) = 1 - \theta$$

$$Prob(prob\_heads(\theta)) = 1$$ ⟵ uniform on [0, 1].

Explaining Data

If you observe:

$$heads(c_1), tails(c_2), tails(c_3), heads(c_4), heads(c_5), \ldots$$

For each $\theta \in [0, 1]$ there is an explanation:

$\{prob\_heads(\theta), turns\_heads(c_1, \theta), turns\_tails(c_2, \theta), turns\_tails(c_3, \theta), turns\_heads(c_4, \theta), turns\_heads(c_5, \theta), \ldots\}$

Aggregating versus quantifying

Consider the difference between:

- The distribution of grades for all students in all courses
- For all students, the distribution of grades in all courses
- For all courses, the distribution of grades over all students
- For all students and all courses, the distribution of grades for that student in that course

Quantifying and Aggregating in ICL

- for all students, use distribution of grades over all courses

$$\mathbf{F} = \{grade(S, C, G) \leftarrow grSt(S, G)\}$$

$$\mathbf{C} = \{\{grSt(S, G) \mid G \in [0, 100]\} \mid S \text{ is a student}\}$$

- for all courses, use distribution of grades over all students

$$\mathbf{F} = \{grade(S, C, G) \leftarrow grC(C, G)\}$$

$$\mathbf{C} = \{\{grSt(C, G) \mid G \in [0, 100]\} \mid C \text{ is a course}\}$$

Probabilistic Inductive Logic Programming

- Given a dataset, choose the best probabilistic logic program given the data... taking into account:
  - fit to the data
  - prior probability of the program

Probabilistic Inductive Logic Programming

- Given a dataset, choose the best probabilistic logic program given the data... taking into account:
  - fit to the data
  - prior probability of the program

Is there an alternative?

Probabilistic Inductive Logic Programming

- Given a dataset, choose the best probabilistic logic program given the data... taking into account:
  - fit to the data
  - prior probability of the program

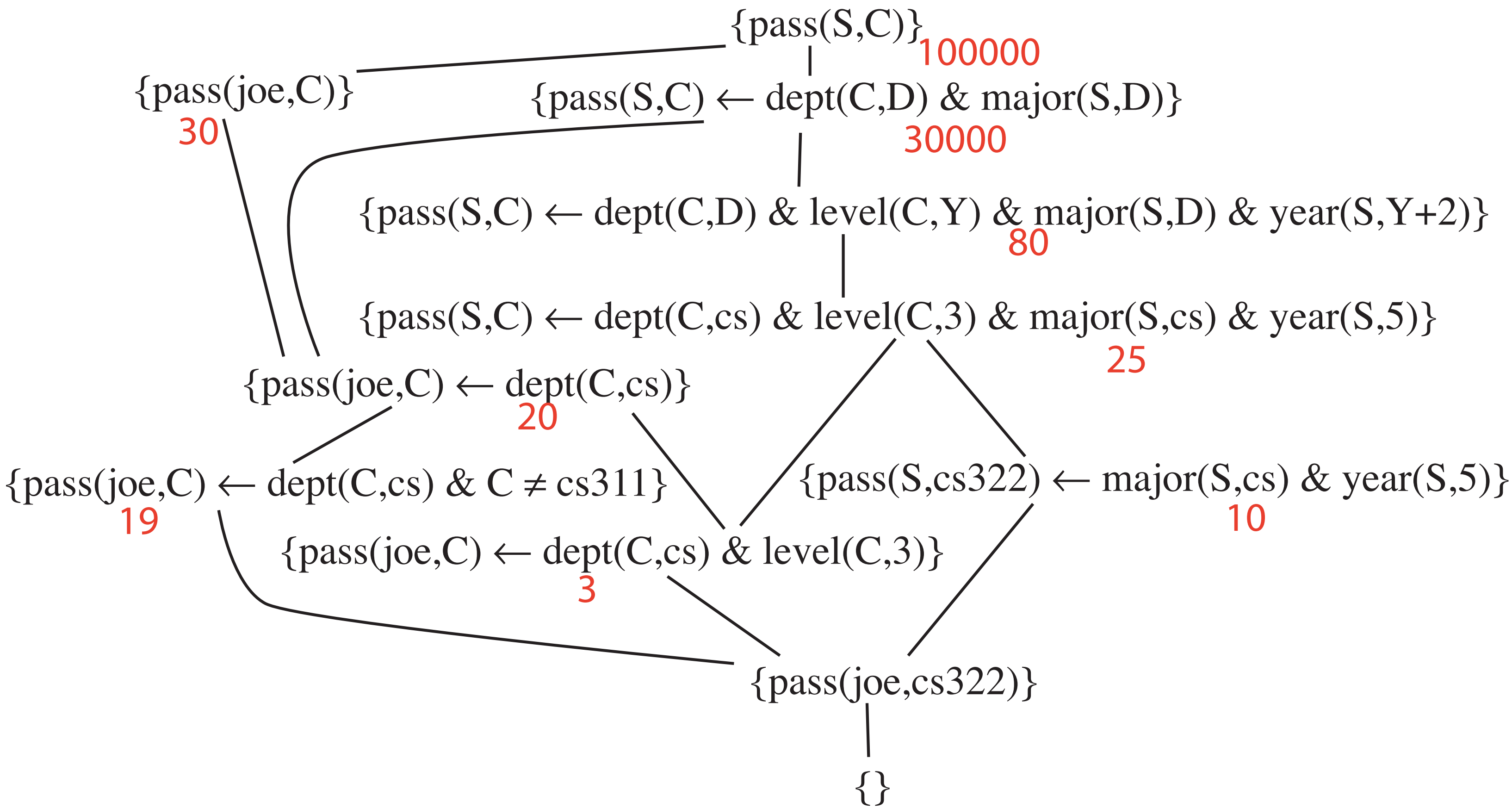
Is there an alternative?

- Bayesian: don't choose the best model, but have a probability distribution over the models
  - combine all of the models

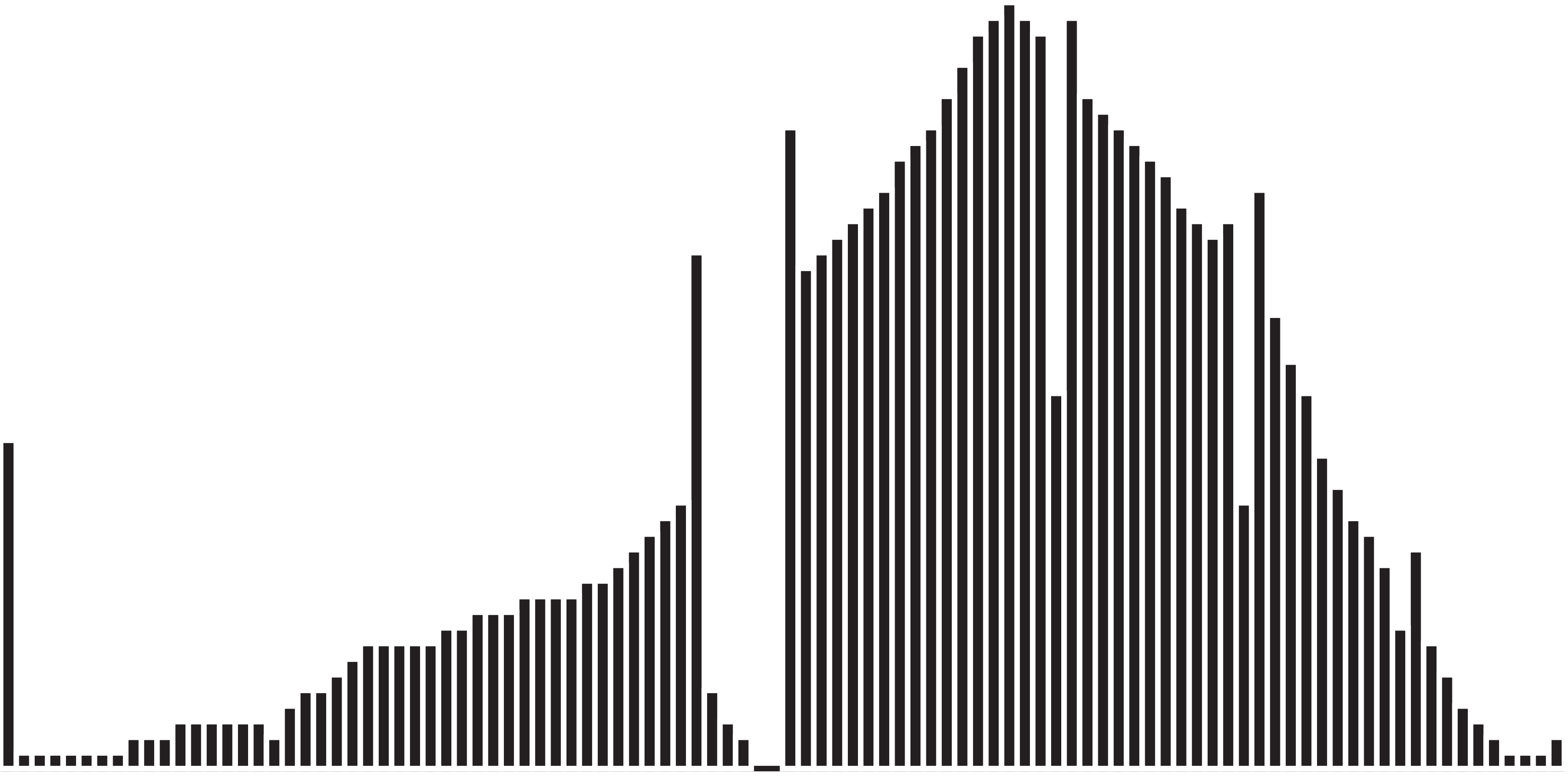
Issues with Bayesian ILP

- Need to use all of the reference classes; even the most general one!
- The lowest reference classes will all have very few observed instances.
- Need to use more general reference classes to get the prior on the more specific.
- Need a way to combine different most specific reference classes.

Specificity and Counts

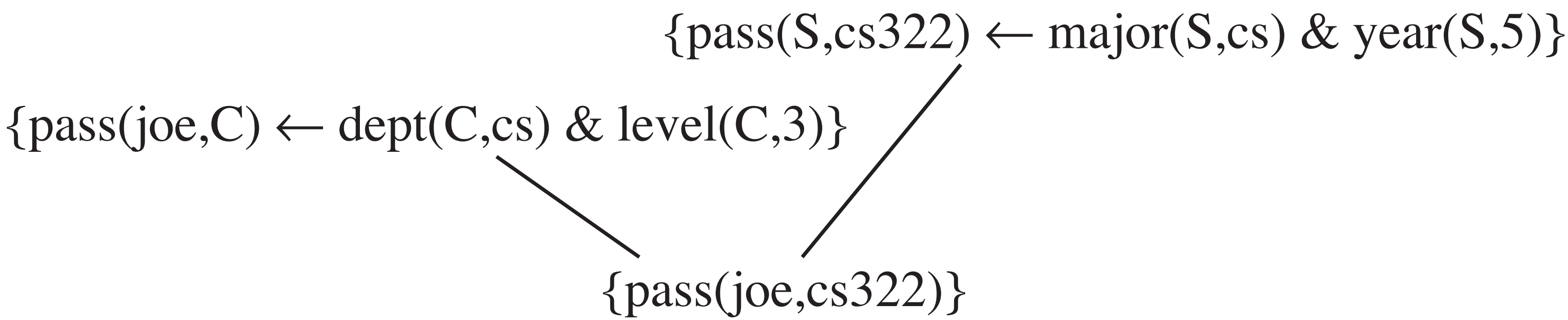


Using The Most General Reference Class



Inferring distributions from generalizations

Even if you knew the distribution of immediate generalizations, how do you infer the appropriate distribution?



Other Sorts of Rules

$$passed(S, C) \leftarrow$$
$$passed(S, C') \wedge$$
$$similarCourses(C, C').$$

$$passed(S, C) \leftarrow$$
$$passed(S', C) \wedge$$
$$similarStudents(S, S').$$

$\Longrightarrow$ Collaborative Filtering

Can we also use the same technique to learn similar grades?

Lessons from history

- In the Seventeenth century, there were accurate models predicting the motion of stars and planets using universal function approximaters (epicycles).
- Even when Newton came up with the “correct” model, it took a long time to fit the data as well.
- We need representations that can express the “correct” models, even if these may be difficult to find.

Conclusion

- Mix of logic programming + Bayesian learning seems to be most promising
- Many problems still to be solved
  - some, such as the *reference class problem*, have a long history
  - some are new
  - the combination is relatively unexplored
- You can anticipate many different solutions