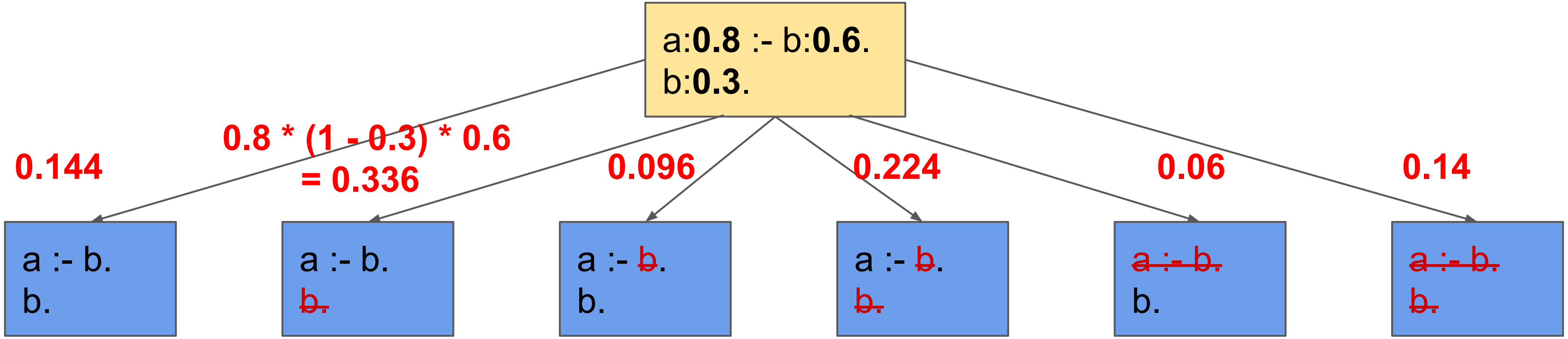
An Abductive-Inductive Algorithm for Probabilistic Inductive Logic Programming

Stanislav Dragiev, Alessandra Russo, Krysia Broda, Mark Law, Călin-Rareş Turliuc
Department of Computing, Imperial College London

Annotated Literal Programs

```
a:0.8 :- b:0.6.
b:0.3.
```

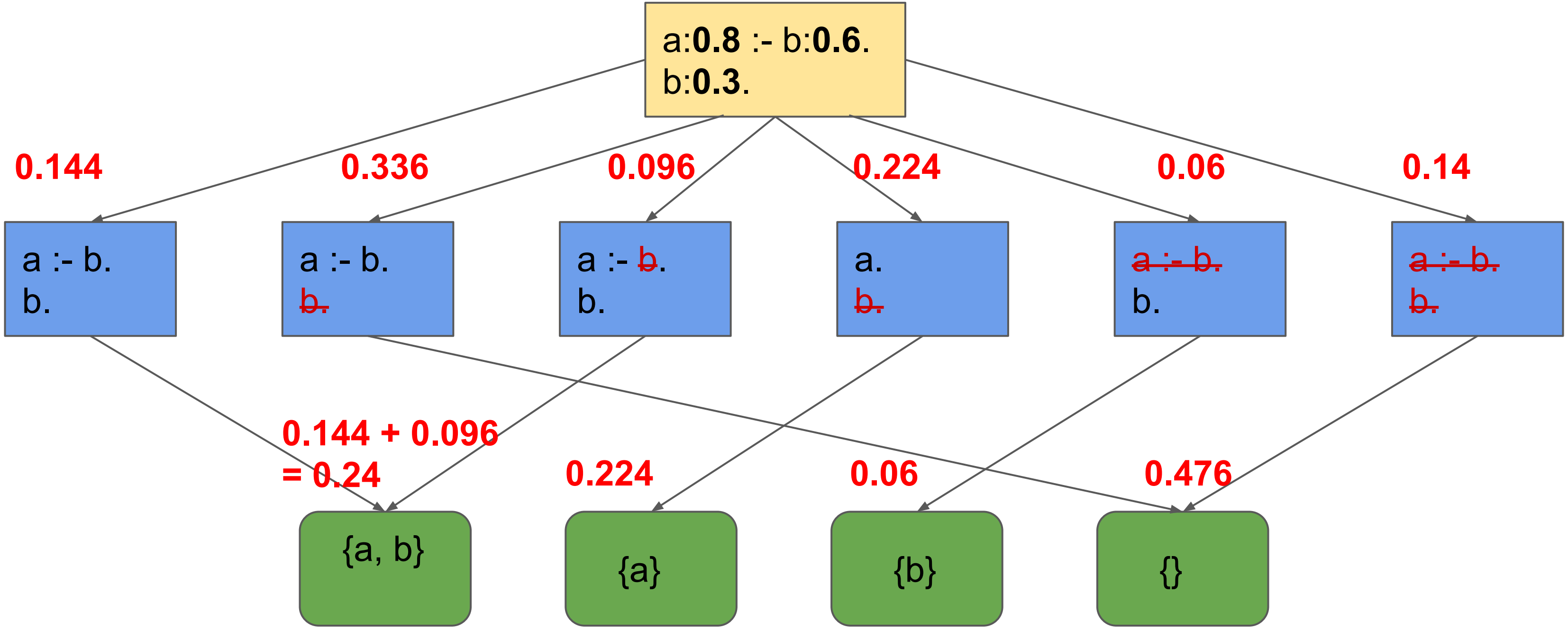
Annotated Literal Programs



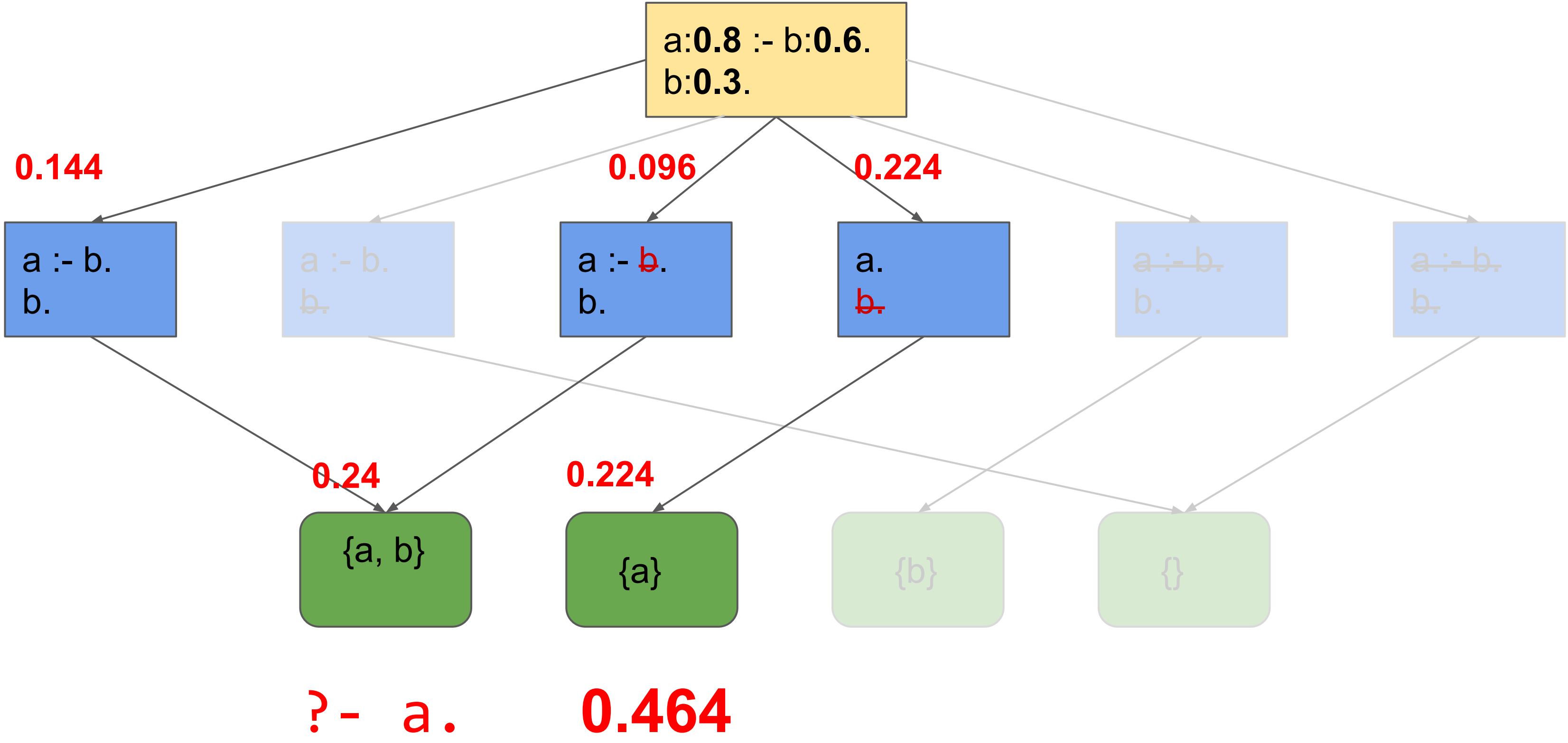
Generative story:

1. Include clause if Bernoulli trial with head parameter is successful
2. For each included clause, sample each body literal if Bernoulli trial with body parameter is successful

Annotated Literal Programs



Annotated Literal Programs



Algorithm introduction

INPUT:

partial interpretations | background knowldge | structural and probabilistic bias

Structural Learning
Loosely based on XHAIL abductive and deductive procedure

Most Specific Hypothesis

Any interpretation is entailed by at least one theory that subsumes this hypothesis.

Parameter Learning
Uses a Peircebayes statistical abduction program that is based on XHAIL's inductive step.

OUTPUT:

Annotated Literal Program

Background - XHAIL Kernel Set Learning + Induction

Kernel Set Construction

Input: A set of examples E (positive/negative ground atoms), background knowledge B, mode bias M
Output: A logic program P such that $B \cup P \Vdash E$

2 Steps:

- Abduce head of clauses Δ such that: $B \cup \Delta \Vdash E$
- Deduce body of clauses Λ such that $B \cup \Delta \Vdash \Lambda$

Induction

Expressed as an abductive task:

Background

```
a :- b, c.
b.
```

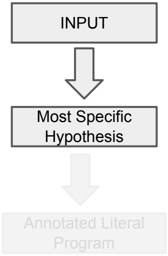
```
a :- use(0,0), try(0,1), try(0,2).
try(0,1) :- b, use(0,1).
try(0,1) :- not use(0,1).
try(0,2) :- c, use(0,1).
try(0,2) :- not use(0,2).

b :- use(1,0).
```

Abducibles: `use(#ClauseIndex, #LitIndex).`
Goal: Examples

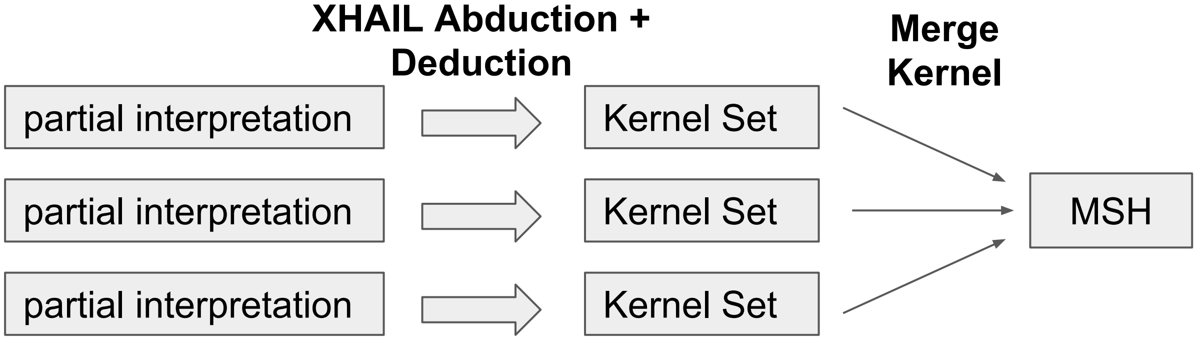
Oliver Ray. Nonmonotonic abductive inductive learning. *Journal of Applied Logic, 7(3):329 – 340, 2009.*

Structure Learning of PROBXHAIL



Two Approaches:

- **Bias-Driven Structure Learning** - Trivial, high capacity
- **Data-Driven Structure Learning**
  - ✓ Lower capacity, lower dimensionality
  - ✓ Still can be used to generate an ALP such that any partial interpretation has non-zero probability.
  - ✓ If examples are similar, result of merge is slightly larger than individual Kernel Sets due to overlap.



Examples:

Input:

Bias: { modeh(a), modeb(b), modeh(b) }

Partial Interpretations (with counts):
{a, not b} x 2
{b, a} x 6
{b} x 27

Merge Kernel:

a :- b. b.	**+**	a :- c. c.

a :- b, c.
b.
c.

Peircebayes - Statistical Abduction

Each abducible has a categorical variable as a term.
Goal: Compute the posterior of the parameters of the categorical distributions.

Background:
```
observe(heads, B) :- coin1(heads), coin2(B).
observe(tails, B) :- coin1(tails), coin3(B).

% Associate each coin toss with abducible.
```

Observations:
observe(heads, tails). x 3151
observe(tails, tails). x 310
observe(heads, heads). x 28

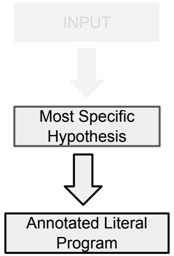
Abducibles:
abd1 - 2 categories, prior (80,20)
abd2 - 2 categories, prior (500,500)
abd3 - 2 categories, prior (100, 100)

```
coin1(heads) :- abd1(0).    coin2(heads) :- abd2(0).
coin1(tails) :- abd1(1).    coin2(tails) :- abd2(1).

coin3(heads) :- abd3(0).
coin3(tails) :- abd3(1).
```

Output:
abd1 -> [0.75, 0.25];
abd2 -> [0.5, 0.5]
abd3 -> [0.5, 0.5]

Calin Rares Turliuc, Luke Dickens, Alessandra Russo, and Krysia Broda. Probabilistic abductive logic programming using dirichlet priors. *International Journal of Approximate Reasoning, 2016.*



Parameter Learning of PROBXHAIL

Example:

a :- b, c.
b.

XHAIL Inductive task

Background:
a :- use(0,0), try(0,1), try(0,2).
try(0,1) :- b, use(0,1).
try(0,1) :- not use(0,1).
try(0,2) :- c, use(0,1).
try(0,2) :- not use(0,2).

b :- use(1,0).

Peircebayes Program

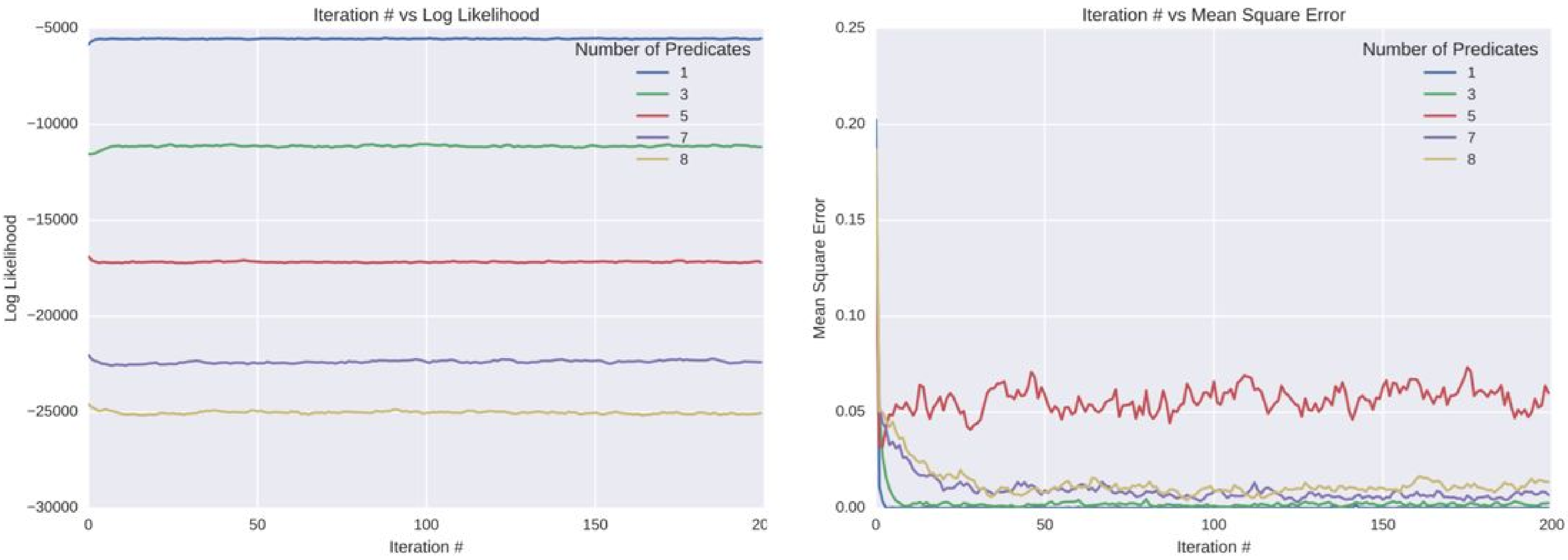
Abducibles:
use(0,0). use(0,1). use(0,2).
use(1,0). - *2 categories(!) with priors from probabilistic bias*

Observations:
Each partial interpretation

Output: Parameters for each abducible
use(0,0). [0.5, 0.5]
use(1,0). [0.8, 0.2]
use(0,1). [0.3, 0.7]
use(0,2). [0.4, 0.6]

a:0.5 :- b:0.4, c:0.6.
b:0.8.

Evaluation


Iteration # vs Log Likelihood
Log Likelihood
−5000
−10000
−15000
−20000
−25000
−30000
0
50
100
150
20
Iteration #
Number of Predicates
1
3
5
7
8
Iteration # vs Mean Square Error
Mean Square Error
0.25
0.20
0.15
0.10
0.05
0.00
0
50
100
150
200
Iteration #
Number of Predicates
1
3
5
7
8

Summary

1. Properties of ALP and PROBXHAIL:
   - Probabilistic bias: Uses a Bayesian prior to encode prior information on probabilities.
   - Uses stable model semantics - brave entailment

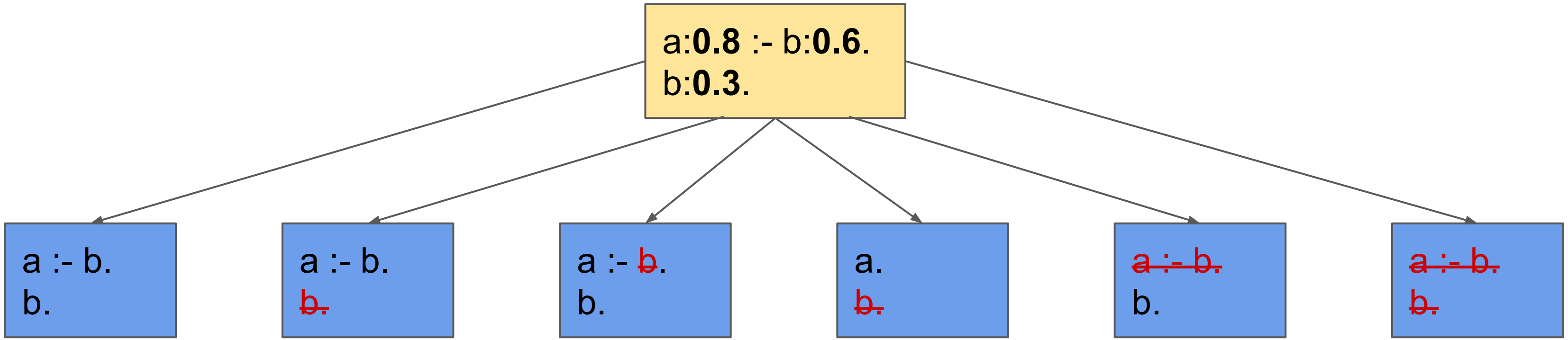
2. Further Work
   - Reduce output size of structural learning:
     - Data-Driven structure Learning
     - Fixed parameters
   - Break independence of abducibles using a more complex probability model
     - Joint probability use literals with Categorical Parameters

End

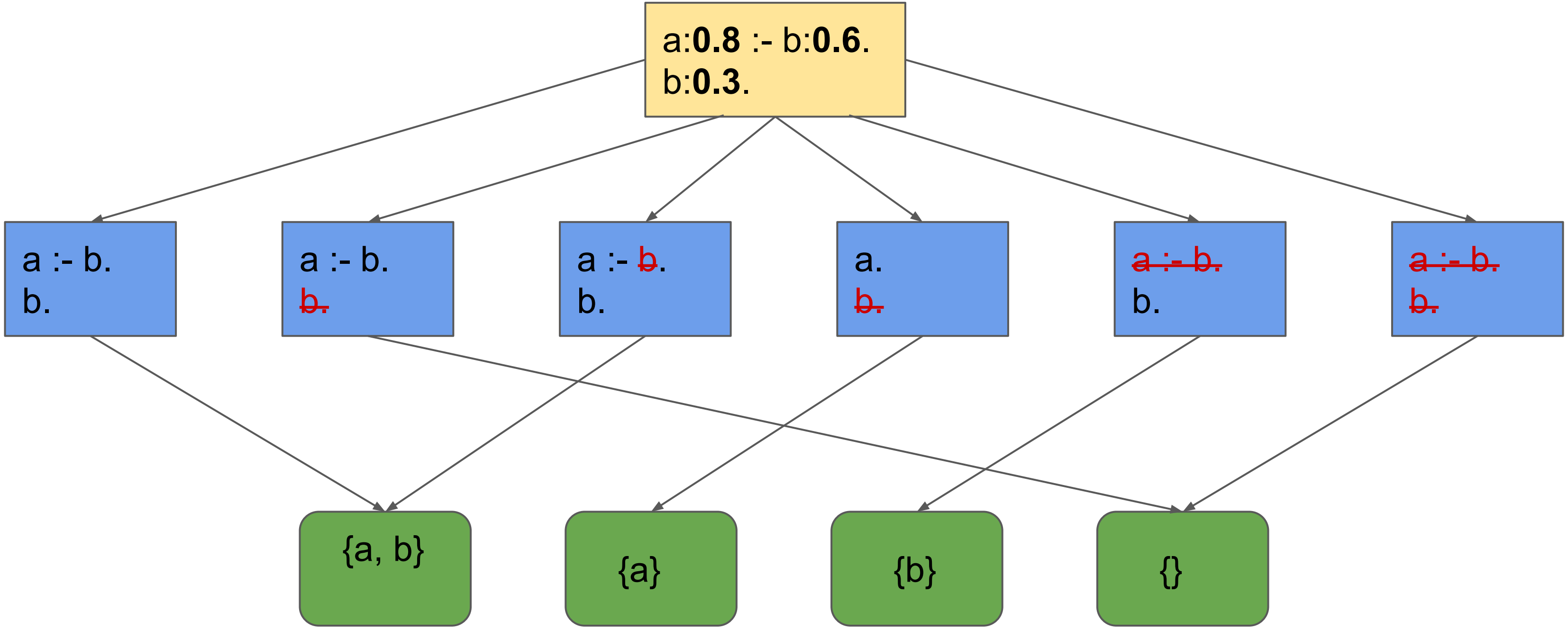
Questions?

Summary

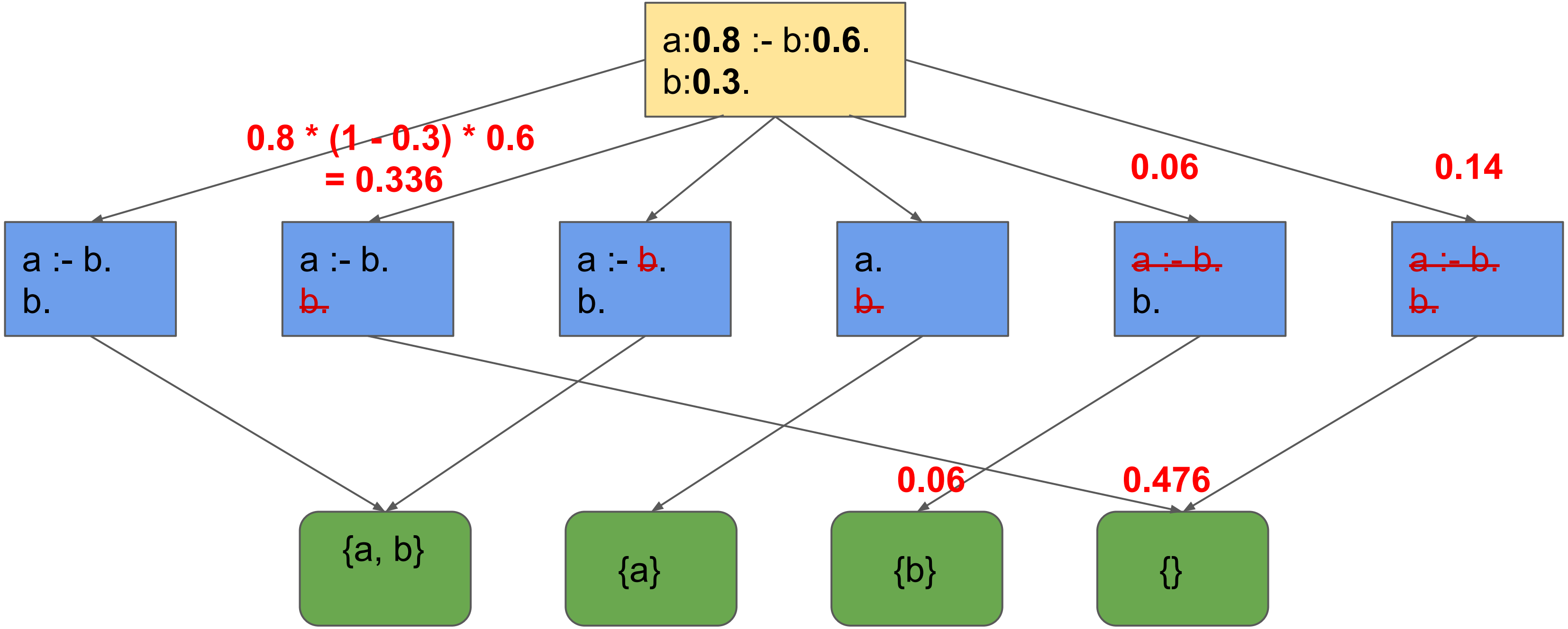
Annotated Literal Programs



Annotated Literal Programs

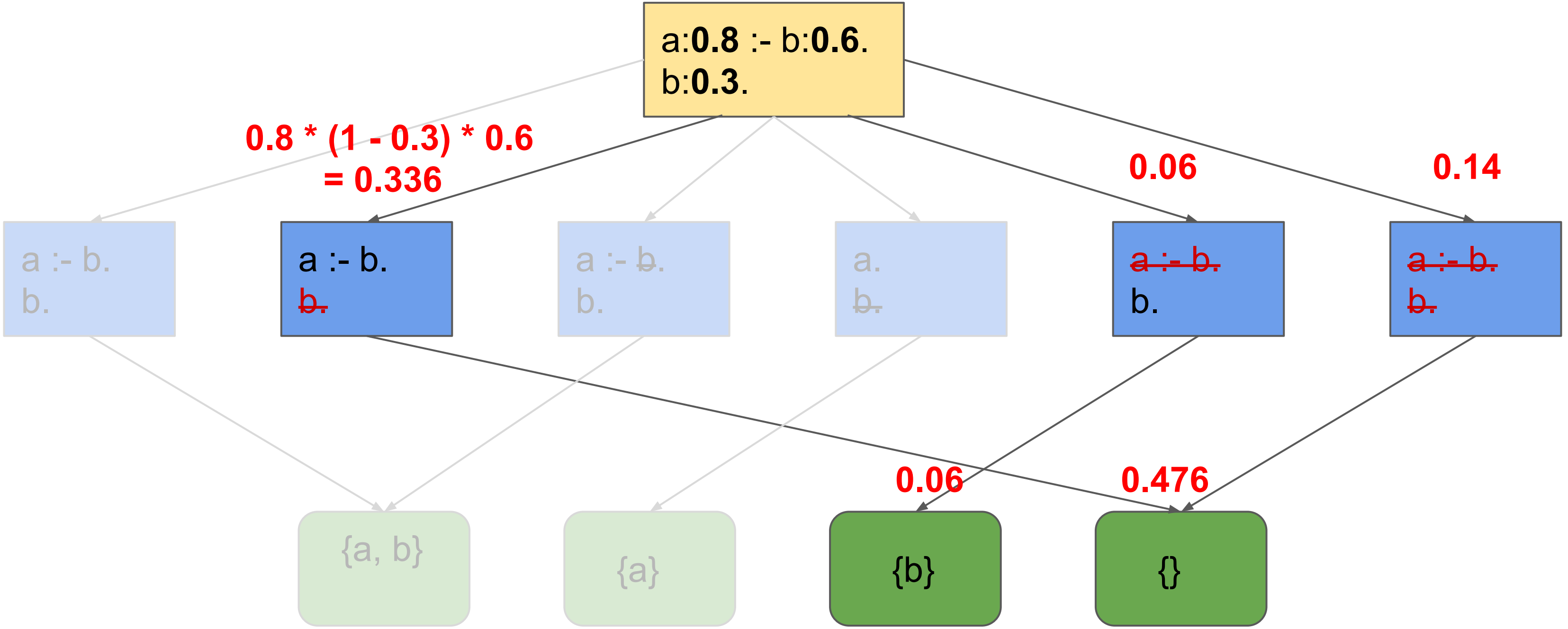


Annotated Literal Programs



`?- not a.` **0.482**

Annotated Literal Programs



`?- not a.` **0.482**

Structure Learning (using XHAIL)

- Example of a bias

- Describe the approach

Peircebayes output

Bayesian estimator vs Predictive distribution

Background - XHAIL Kernel Set Learning + Induction

Kernel Set Construction

Input: A set of examples E (positive/negative ground atoms), background knowledge B, mode bias M
Output: A logic program P such that $B \cup P \VDash E$

2 Steps:

- Abduce head of clauses Δ such that: $B \cup \Delta \VDash E$
- Deduce body of clauses Λ such that $B \cup \Delta \VDash \Lambda$

Induction

Expressed as an abductive task:

```
a :- b, c.
b.
```

```
a :- use(0,0), try(0,1), try(0,2).
try(0,1) :- b, use(0,1).
try(0,1) :- not use(0,1).
try(0,2) :- c, use(0,1).
try(0,2) :- not use(0,2).

b :- use(1,0).
```

Oliver Ray. Nonmonotonic abductive inductive learning. *Journal of Applied Logic, 7(3):329 – 340, 2009.*

Input/Output Summary

Input

===>

Output

=>>>>>>

Use parameter of classification on unseen examples.

Poster

3 parts:
General Idea

More Background and MORE Motivation and (Comparison Method Table)

Algorithm