Towards Probabilistic Logic Program Synthesis

Luc De Raedt

Dagstuhl 2015

DTAI
Machine Learning
KU LEUVEN

Probabilistic Programs

- Two questions:
  - can we synthesise them ?
  - can we use them during the search ?

Overview

- Intro to PLP
  - some part on continuous distributions
- Probabilistic ILP
  - rule learning for Probabilistic Logic Programs
- Affordances
  - learning with continuous distributions

PART 1: Intro PLP

Probabilistic Logic Programming

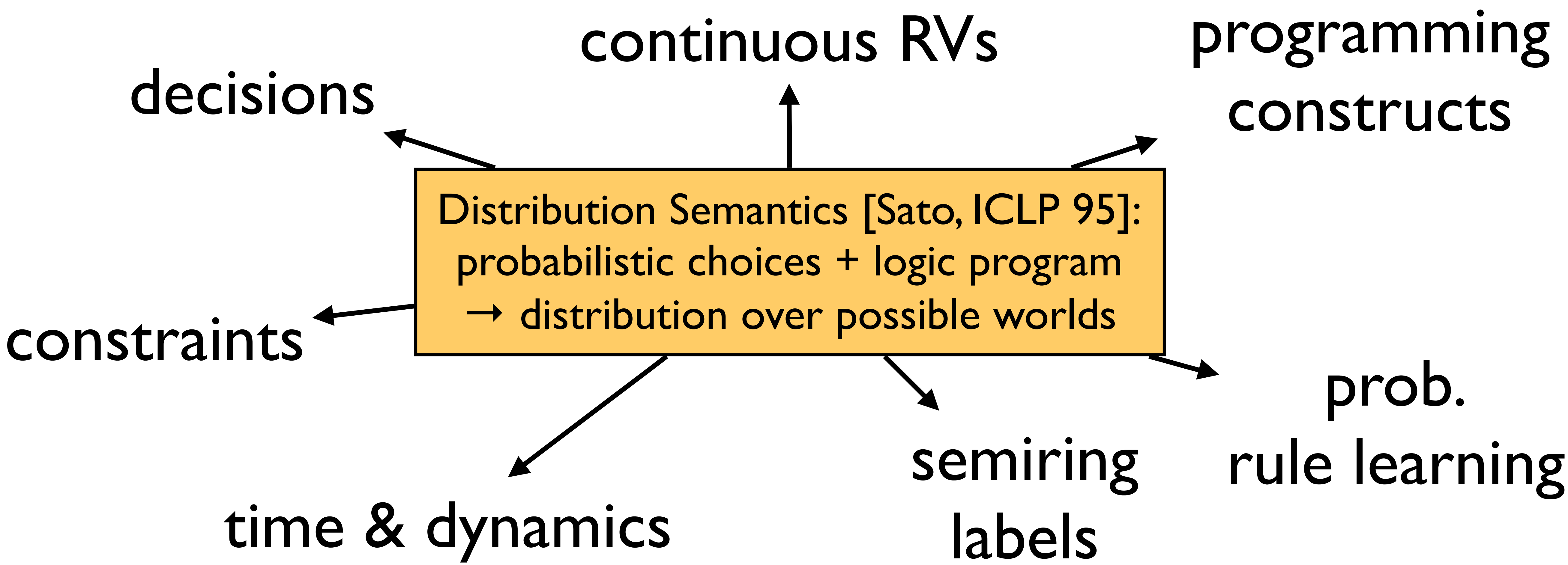
Distribution Semantics [Sato, ICLP 95]:
probabilistic choices + logic program
→ distribution over possible worlds

OVERVIEW paper [Kimmig, De Raedt, MLJ 15]

e.g., PRISM, ICL, ProbLog, LPADs, CP-logic, …

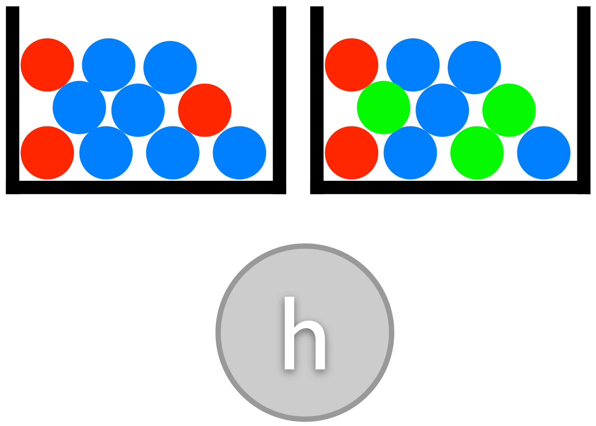
- PRISM → multi-valued switches
- ICL → probabilistic alternatives
- ProbLog → probabilistic facts
- ProbLog, LPADs → annotated disjunctions
- CP-logic → causal-probabilistic laws

Extensions of basic PLP



ProbLog by example:

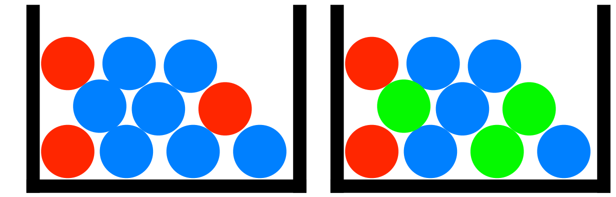
A bit of gambling



- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

ProbLog by example:

A bit of gambling



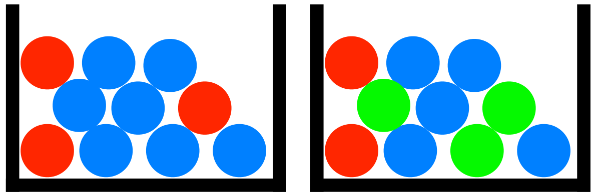
- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

```
0.4 :: heads.
```

probabilistic fact: heads is true with probability 0.4 (and false with 0.6)

ProbLog by example:

A bit of gambling



- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

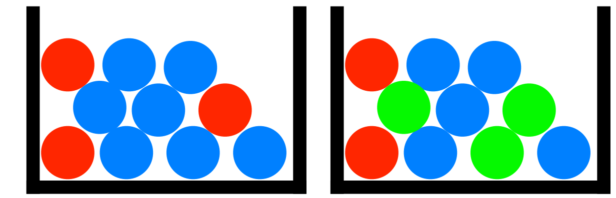
```
0.4 :: heads.
```

annotated disjunction: first ball is red with probability 0.3 and blue with 0.7

```
0.3 :: col(1,red); 0.7 :: col(1,blue) <- true.
```

ProbLog by example:

A bit of gambling



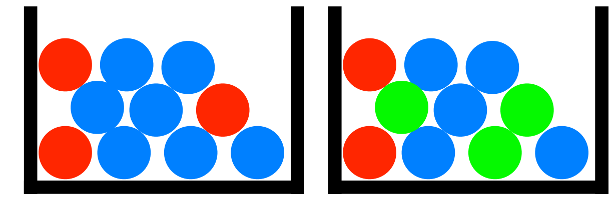
- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue) <- true.
0.2 :: col(2,red); 0.3 :: col(2,green);
                   0.5 :: col(2,blue) <- true.
```

annotated disjunction: second ball is red with probability 0.2, green with 0.3, and blue with 0.5

ProbLog by example:

A bit of gambling



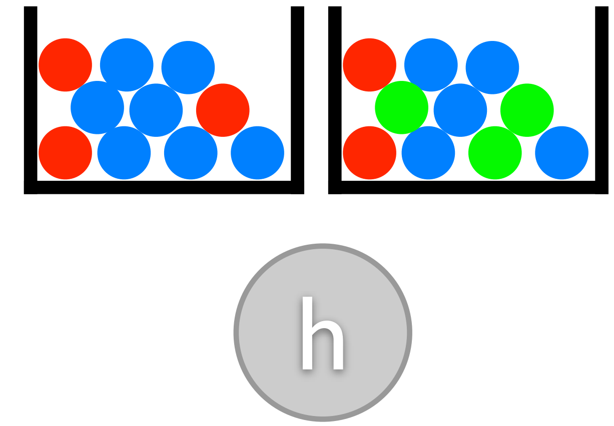
- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue) <- true.
0.2 :: col(2,red); 0.3 :: col(2,green);
                   0.5 :: col(2,blue) <- true.

win :- heads, col(_,red).
```

logical rule encoding background knowledge

ProbLog by example:

A bit of gambling



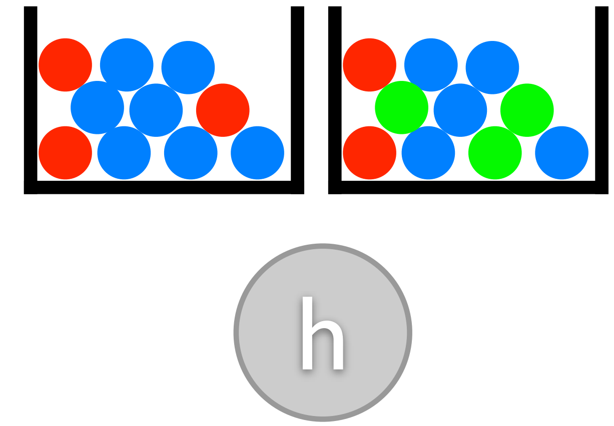
- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue) <- true.
0.2 :: col(2,red); 0.3 :: col(2,green);
                   0.5 :: col(2,blue) <- true.

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

logical rule encoding background knowledge

ProbLog by example:

A bit of gambling



- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue) <- true.
0.2 :: col(2,red); 0.3 :: col(2,green);
                   0.5 :: col(2,blue) <- true.
```

probabilistic choices

```
win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

consequences

Questions

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue) <- true.
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue) <- true.

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

marginal probability

- Probability of `win`?

conditional probability

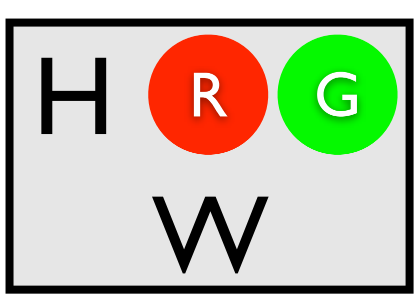
- Probability of `win` given `col(2,green)`?
- Most probable world where `win` is true?

MPE inference

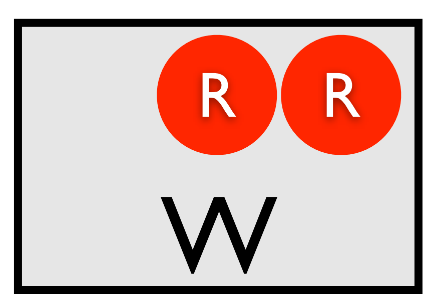
Possible Worlds

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue) <- true.
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue) <- true.

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

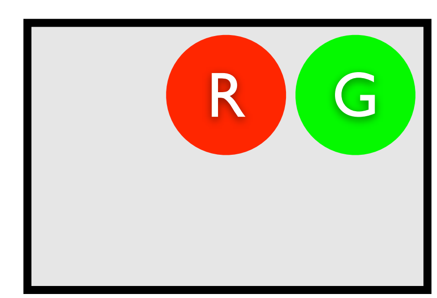
$0.4 \times 0.3 \times 0.3$



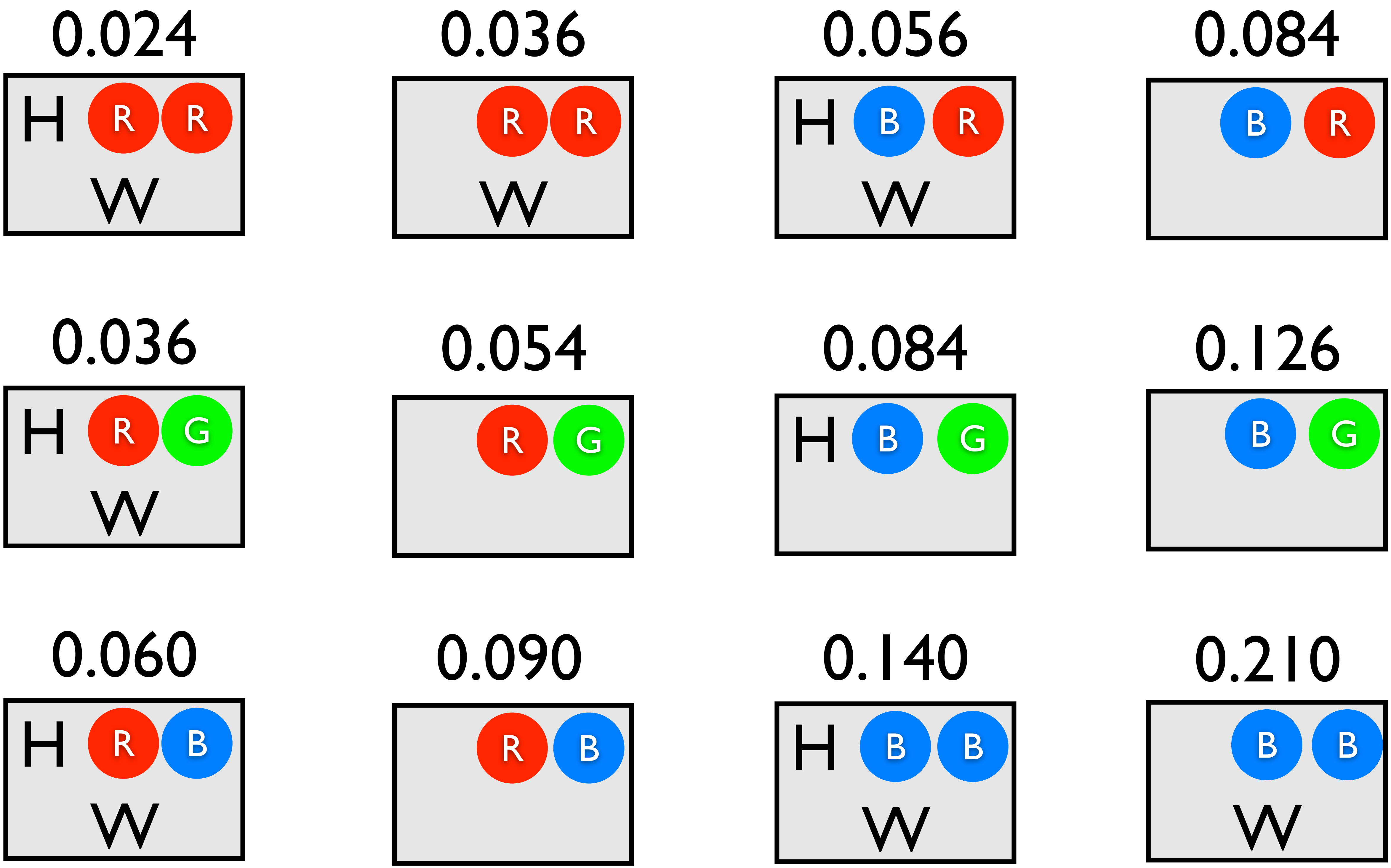
$(1-0.4)\times 0.3 \times 0.2$



$(1-0.4)\times 0.3 \times 0.3$



All Possible Worlds



Distribution Semantics

(with probabilistic facts)

[Sato, ICLP 95]

query

sum over possible worlds where Q is true

$$P(Q) = \sum_{F \cup R \models Q} \underbrace{\prod_{f \in F} p(f) \prod_{f \notin F} 1 - p(f)}_{\text{probability of possible world}}$$

subset of probabilistic facts

Prolog rules

probability of possible world

constraints

cProbLog: constraints on possible worlds

```
weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).

P::pack(Item) :-
  weight(Item,Weight),
  P is 1.0/Weight.

excess(Limit) :- ...

not excess(10).
pack(helmet) v pack(boots).
```


[Fierens et al, PP 12; Shterionov et al]

cProbLog: constraints on possible worlds

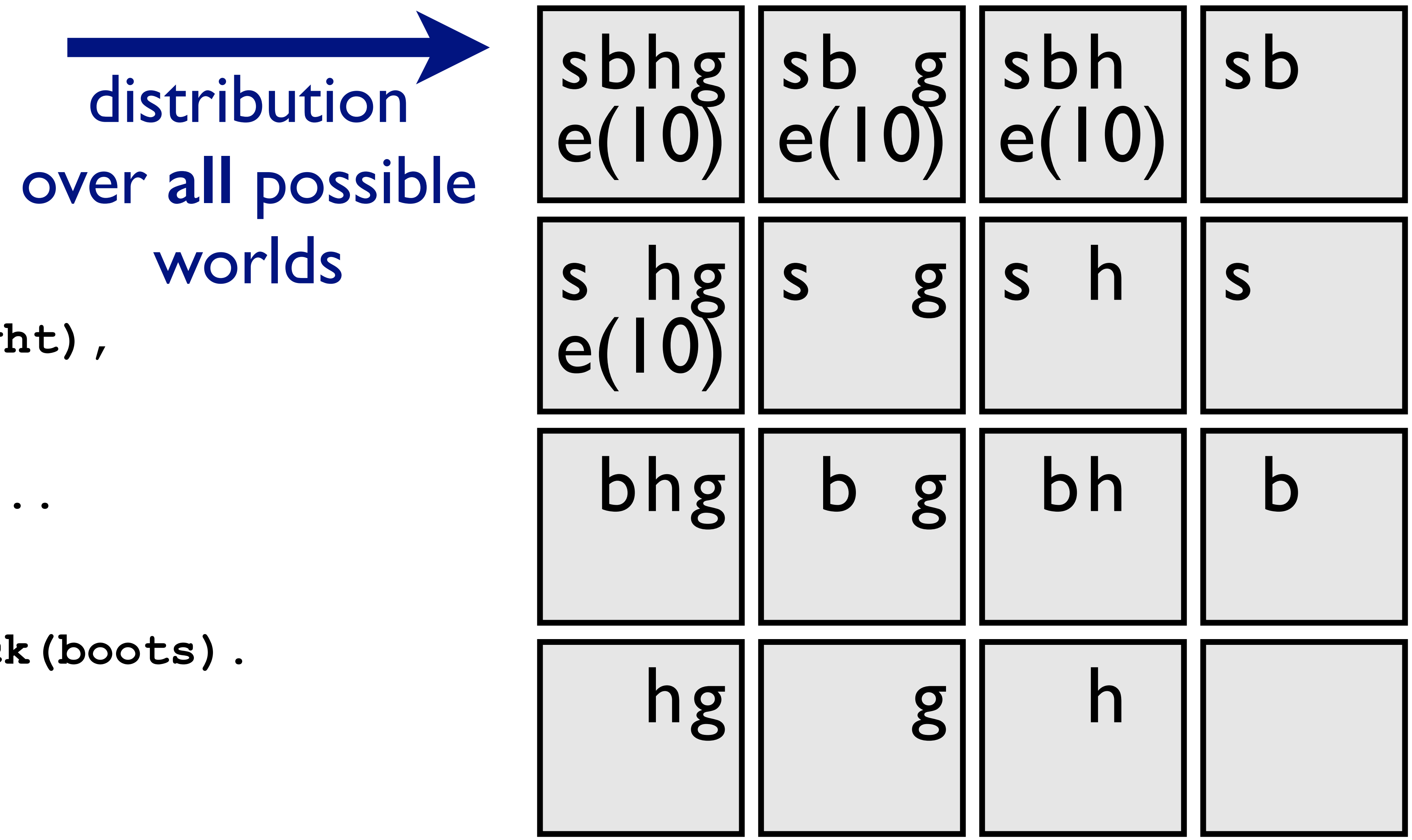
```
weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).

P::pack(Item) :-
  weight(Item,Weight),
  P is 1.0/Weight.

excess(Limit) :- ...

not excess(10).
pack(helmet) v pack(boots).
```

→ distribution over **all** possible worlds

[Fierens et al, PP 12; Shterionov et al]

cProbLog: constraints on possible worlds

```
weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).

P::pack(Item) :-
  weight(Item,Weight),
  P is 1.0/Weight.

excess(Limit) :- ...

not excess(10).
pack(helmet) v pack(boots).
```



[Fierens et al, PP 12; Shterionov et al]

cProbLog: constraints on possible worlds

```
weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).

P::pack(Item) :-
  weight(Item,Weight),
  P is 1.0/Weight.

excess(Limit) :- ...

not excess(10).
pack(helmet) v pack(boots).
```

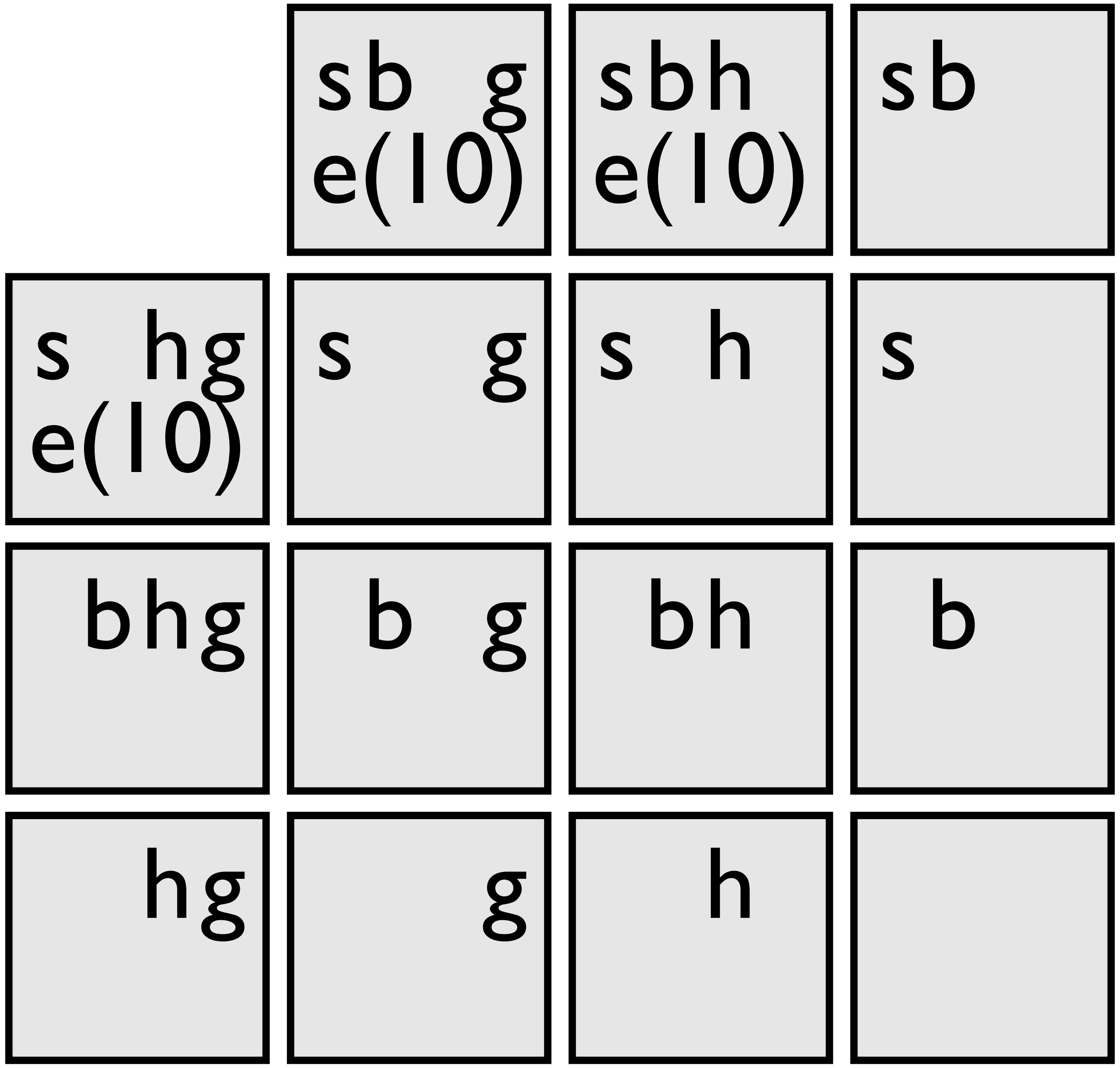
constraints
as FOL formulas
treat as evidence

sbhg e(10)	sb g e(10)	sbh e(10)	sb
s hg e(10)	s g	s h	s
bhg	b g	bh	b
hg	g	h	

[Fierens et al, PP 12; Shterionov et al]

cProbLog: constraints on possible worlds

```
weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).

P::pack(Item) :-
  weight(Item,Weight),
  P is 1.0/Weight.

excess(Limit) :- ...

not excess(10).
pack(helmet) v pack(boots).
```



constraints

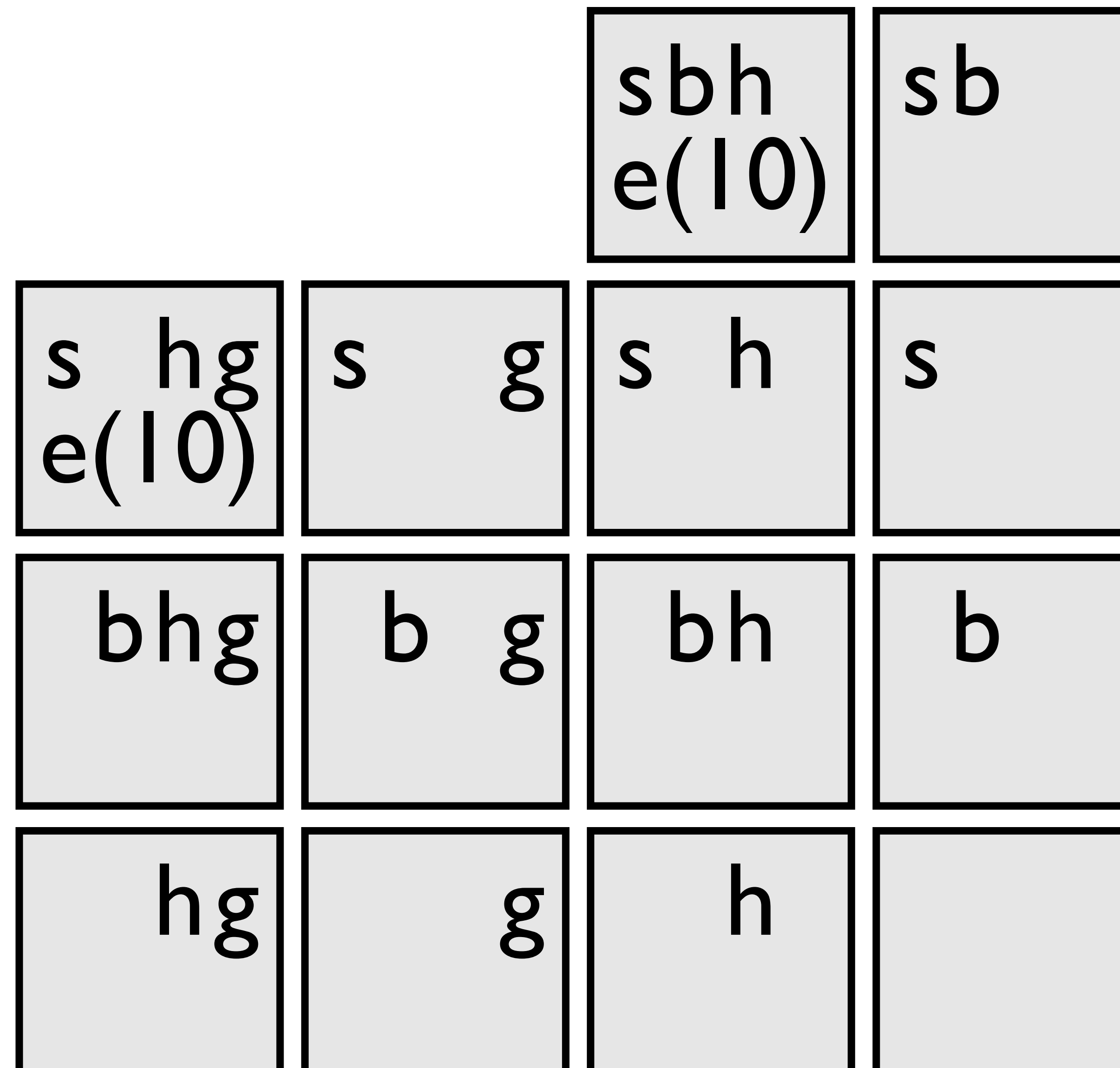
as FOL formulas

treat as evidence

[Fierens et al, PP 12; Shterionov et al]

cProbLog: constraints on possible worlds

```
weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).

P::pack(Item) :-
  weight(Item,Weight),
  P is 1.0/Weight.

excess(Limit) :- ...

not excess(10).
pack(helmet) v pack(boots).
```



constraints

as FOL formulas

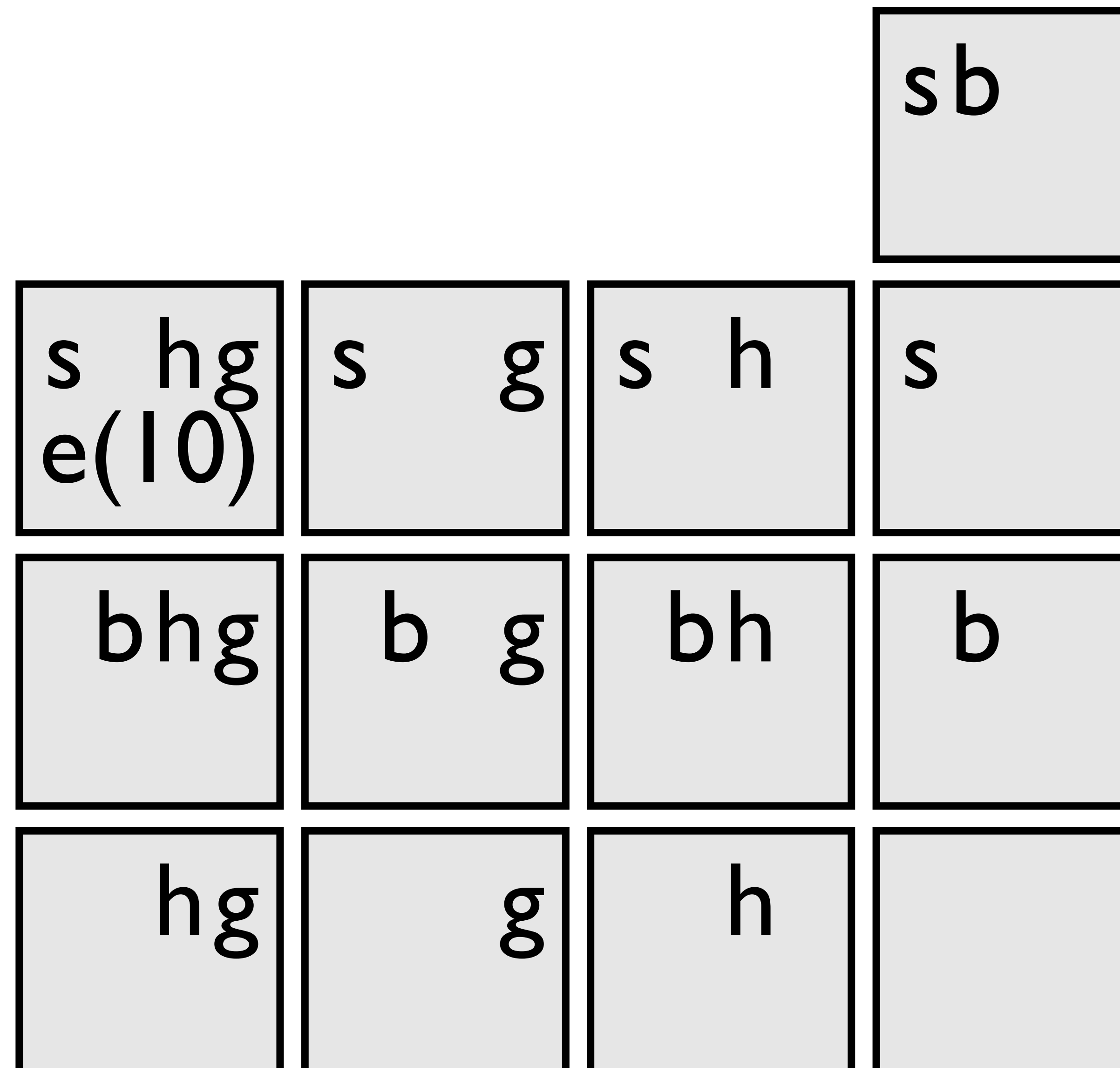
treat as evidence

[Fierens et al, PP 12; Shterionov et al]

cProbLog: constraints on possible worlds

```
weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).

P::pack(Item) :-
   weight(Item,Weight),
   P is 1.0/Weight.

excess(Limit) :- ...

not excess(10).
pack(helmet) v pack(boots).
```

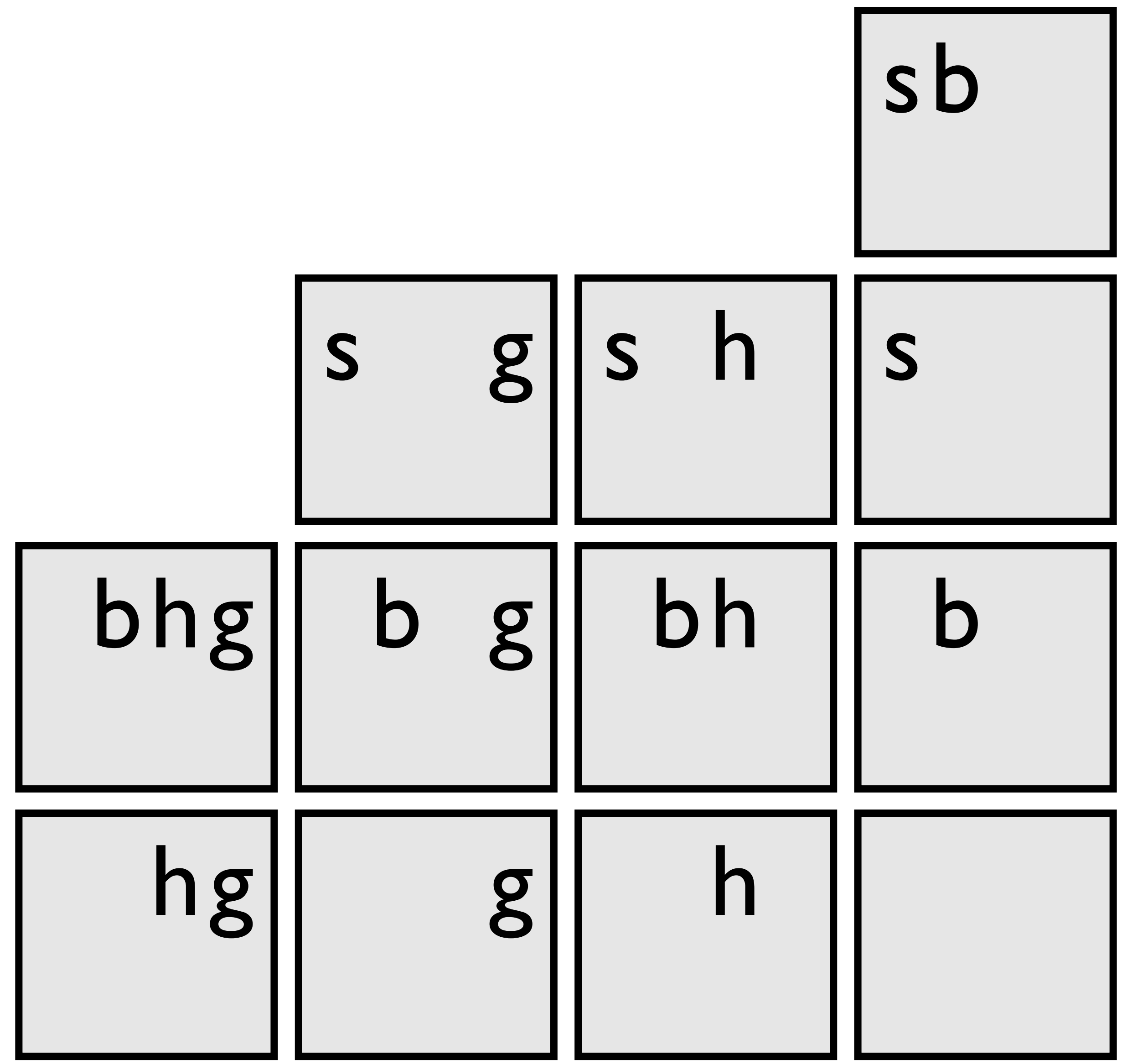
constraints
as FOL formulas
treat as evidence



[Fierens et al, PP 12; Shterionov et al]

cProbLog: constraints on possible worlds

```
weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).

P::pack(Item) :-
  weight(Item,Weight),
  P is 1.0/Weight.

excess(Limit) :- ...

not excess(10).
pack(helmet) v pack(boots).
```

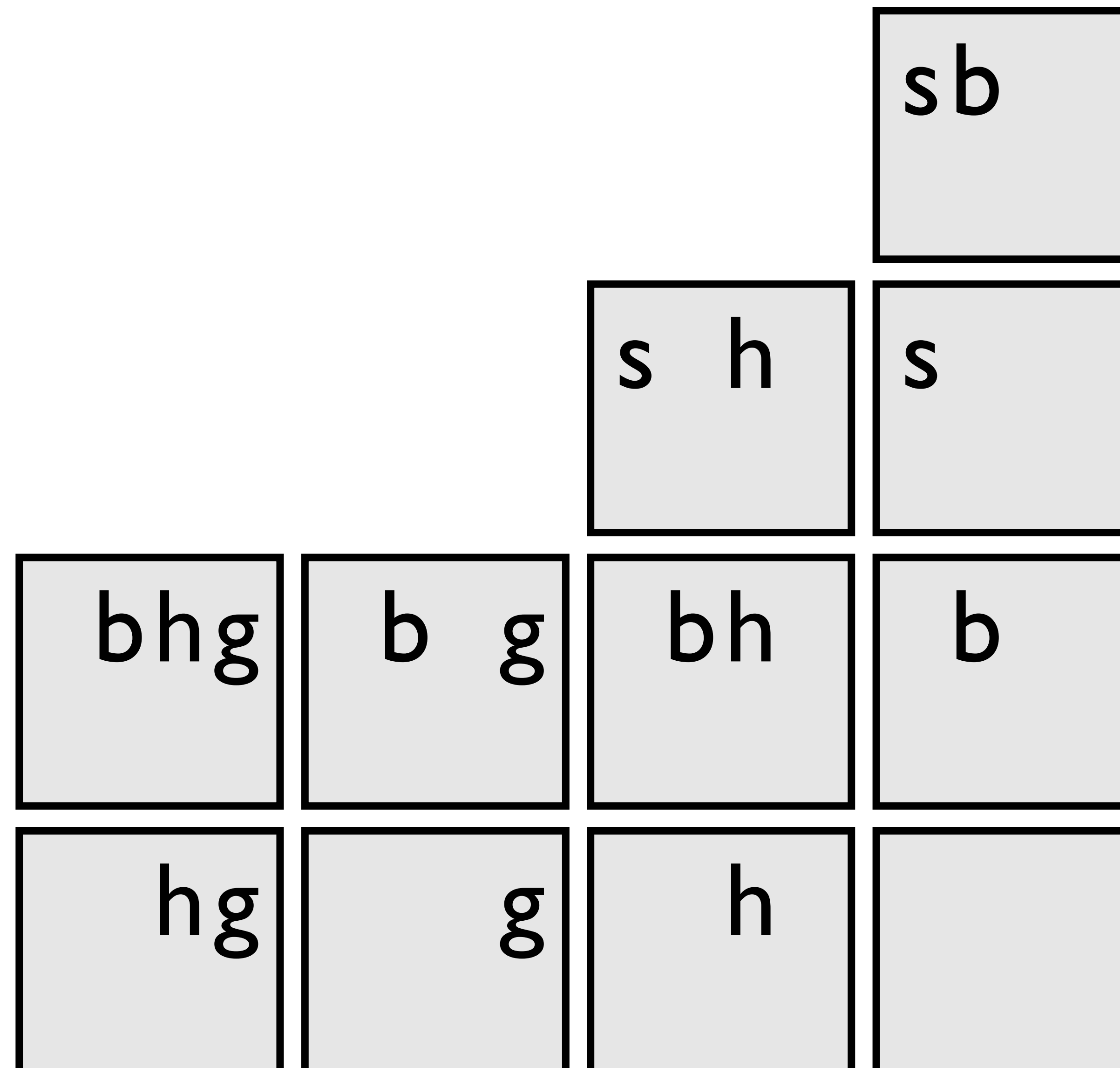


constraints
as FOL formulas
treat as evidence

[Fierens et al, PP 12; Shterionov et al]

cProbLog: constraints on possible worlds

```
weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).

P::pack(Item) :-
  weight(Item,Weight),
  P is 1.0/Weight.

excess(Limit) :- ...

not excess(10).
pack(helmet) v pack(boots).
```



constraints

as FOL formulas

treat as evidence

[Fierens et al, PP 12; Shterionov et al]

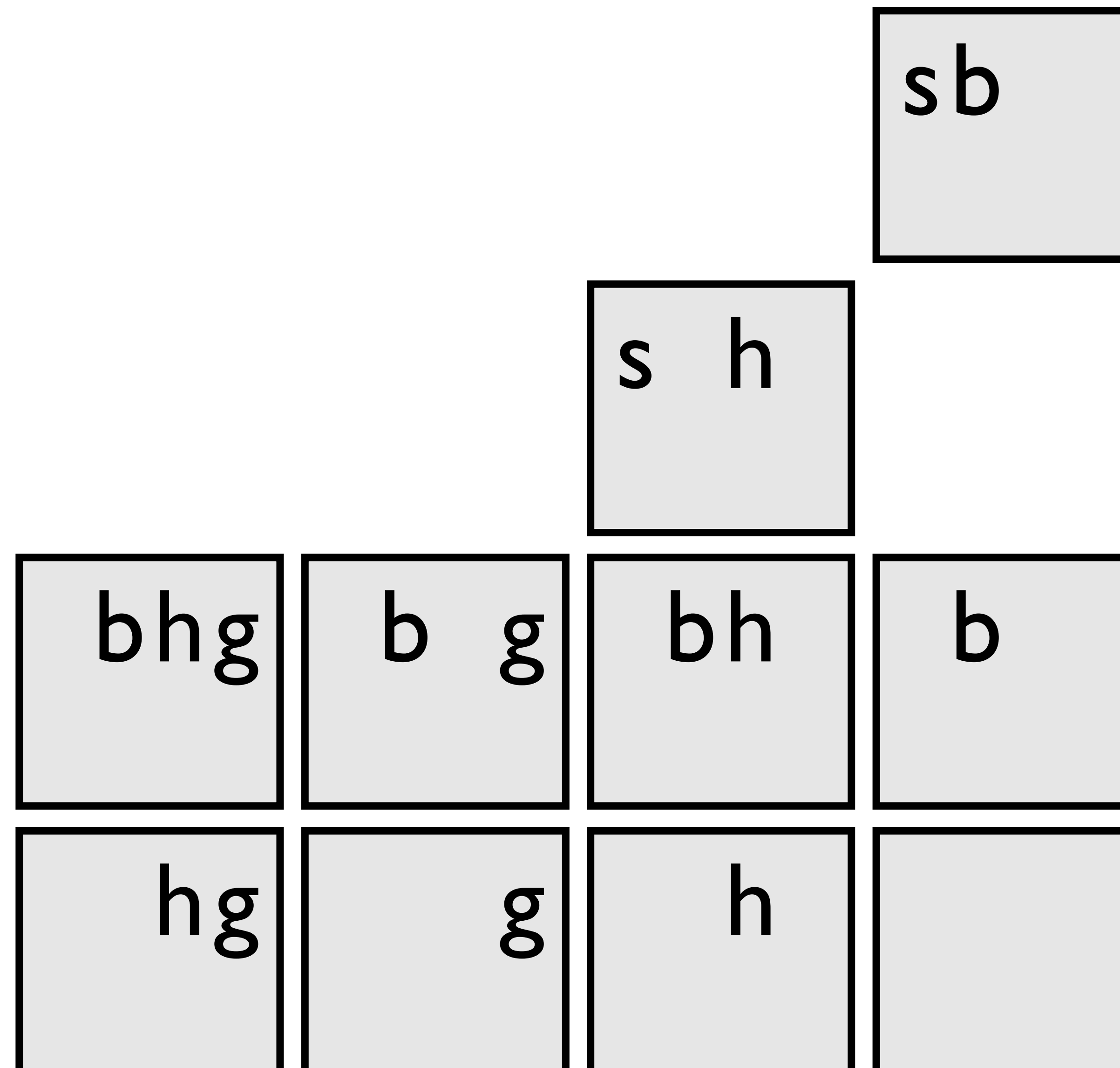
cProbLog: constraints on possible worlds

```
weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).

P::pack(Item) :-
  weight(Item,Weight),
  P is 1.0/Weight.

excess(Limit) :- ...

not excess(10).
pack(helmet) v pack(boots).
```

constraints

as FOL formulas

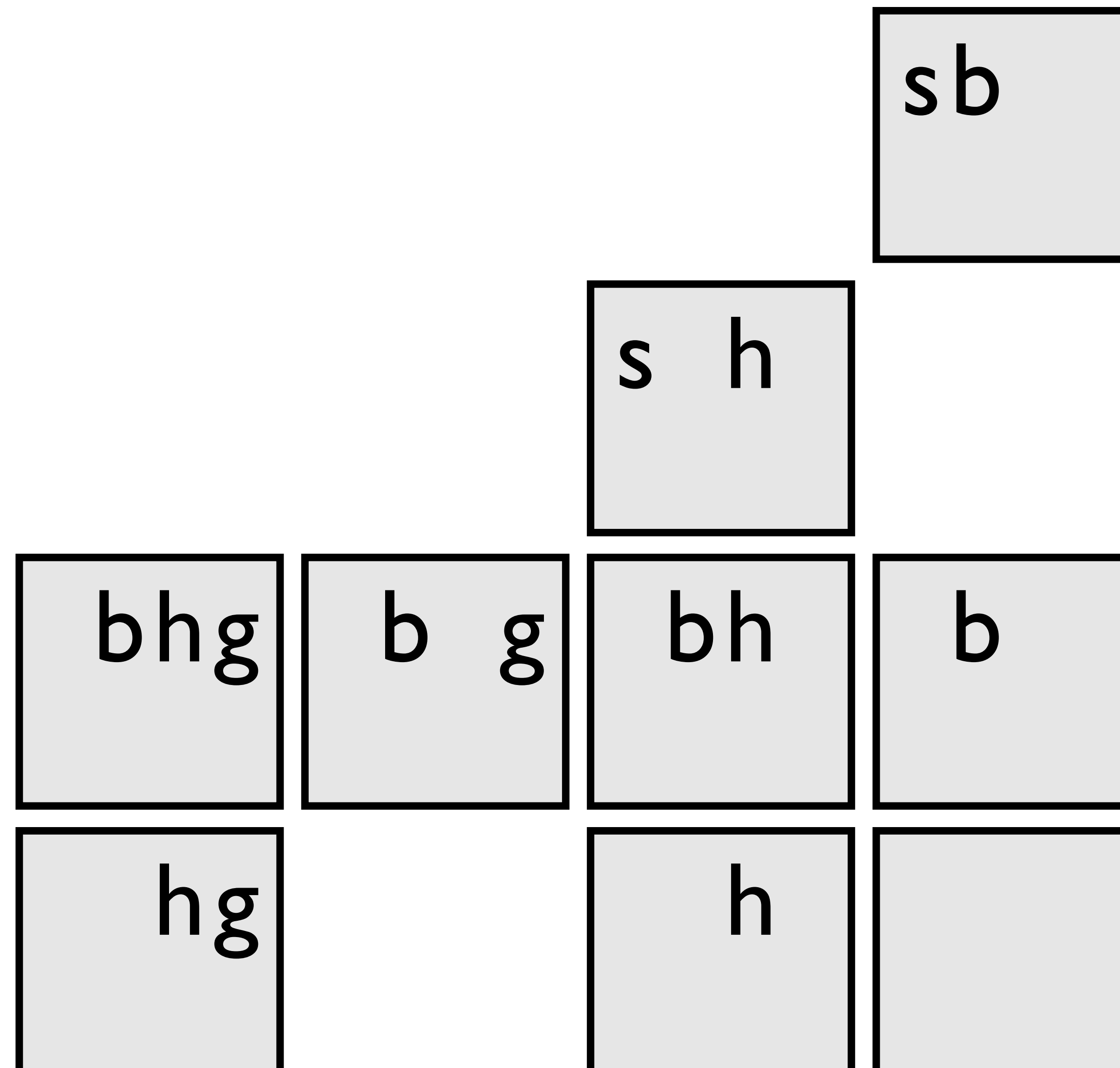
treat as evidence



[Fierens et al, PP 12; Shterionov et al]

cProbLog: constraints on possible worlds

```
weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).

P::pack(Item) :-
  weight(Item,Weight),
  P is 1.0/Weight.

excess(Limit) :- ...

not excess(10).
pack(helmet) v pack(boots).
```

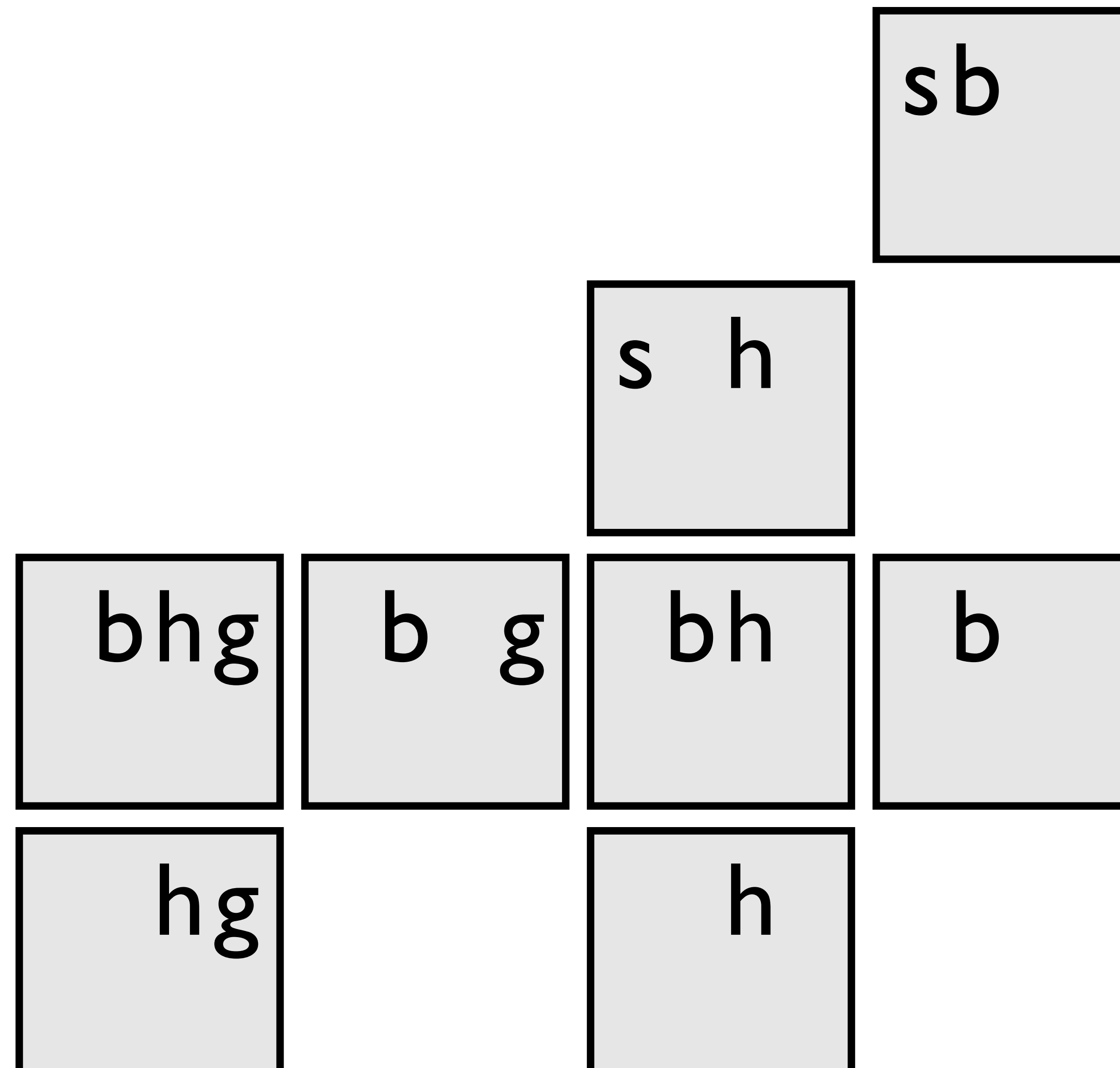


constraints
as FOL formulas
treat as evidence

[Fierens et al, PP 12; Shterionov et al]

cProbLog: constraints on possible worlds

```
weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).

P::pack(Item) :-
  weight(Item,Weight),
  P is 1.0/Weight.

excess(Limit) :- ...

not excess(10).
pack(helmet) v pack(boots).
```



constraints

as FOL formulas

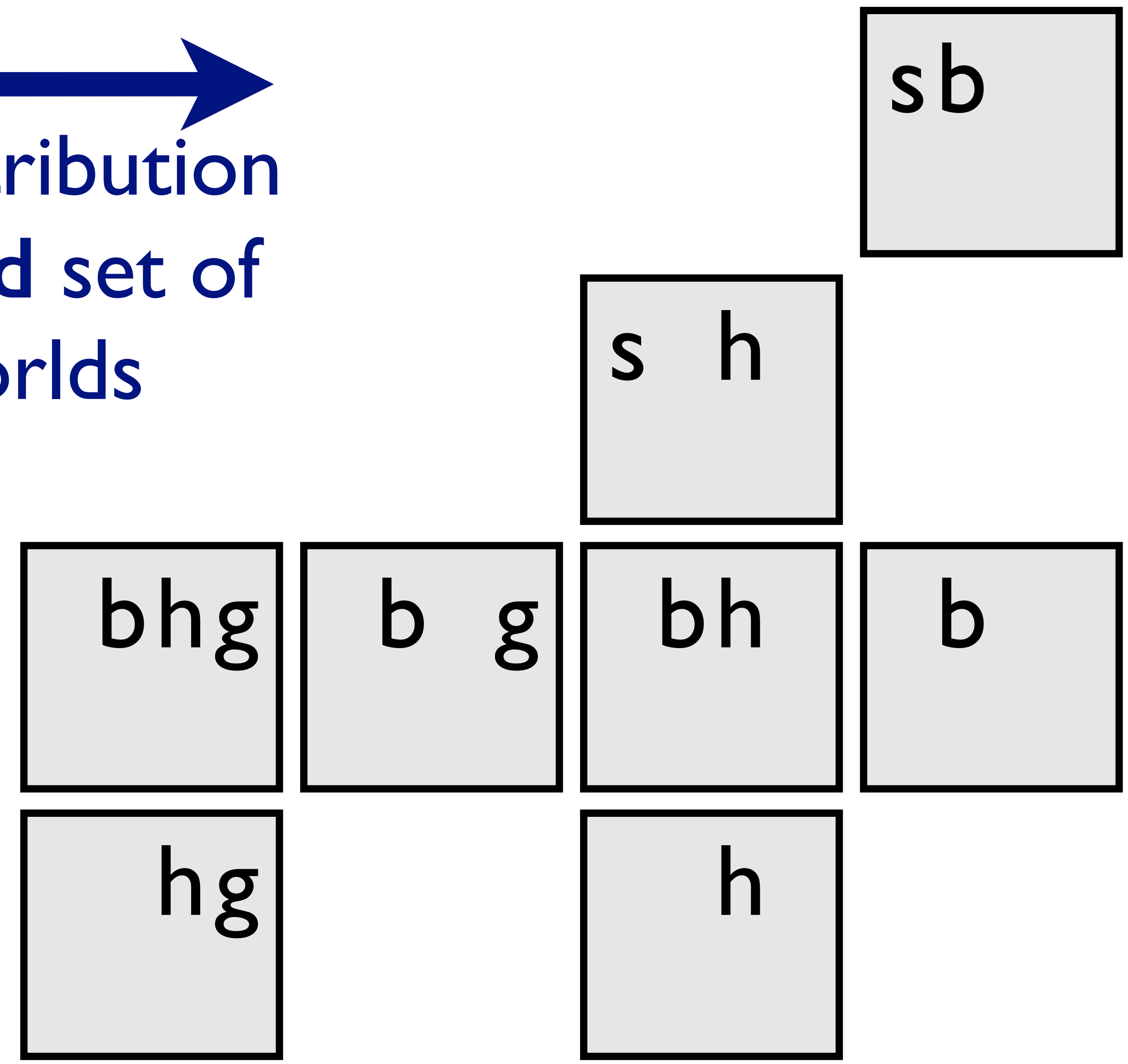
treat as evidence

[Fierens et al, PP 12; Shterionov et al]

cProbLog: constraints on possible worlds

```
weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).

P::pack(Item) :-
  weight(Item,Weight),
  P is 1.0/Weight.

excess(Limit) :- ...

not excess(10).
pack(helmet) v pack(boots).
```

normalized distribution over **restricted** set of possible worlds



constraints

as FOL formulas

treat as evidence

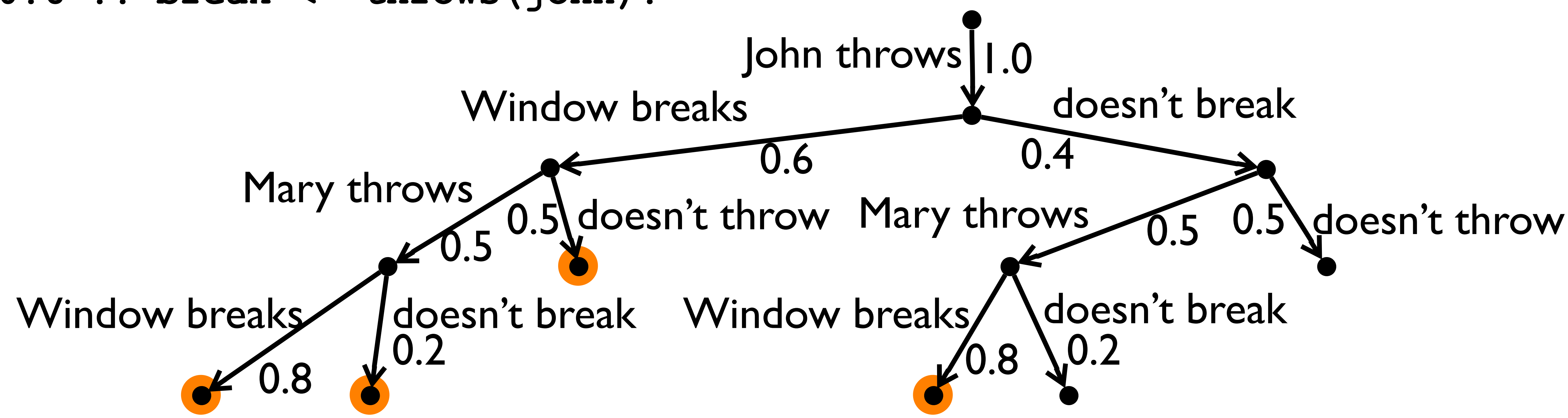
[Fierens et al, PP 12; Shterionov et al]

[Vennekens et al, ICLP 04]

Alternative view: CP-Logic

```
throws(john).
0.5::throws(mary).

0.8 :: break <- throws(mary).
0.6 :: break <- throws(john).
```

probabilistic causal laws



$$P(\text{break})=0.6\times0.5\times0.8+0.6\times0.5\times0.2+0.6\times0.5+0.4\times0.5\times0.8$$

CP-logic [Vennekens et al.]

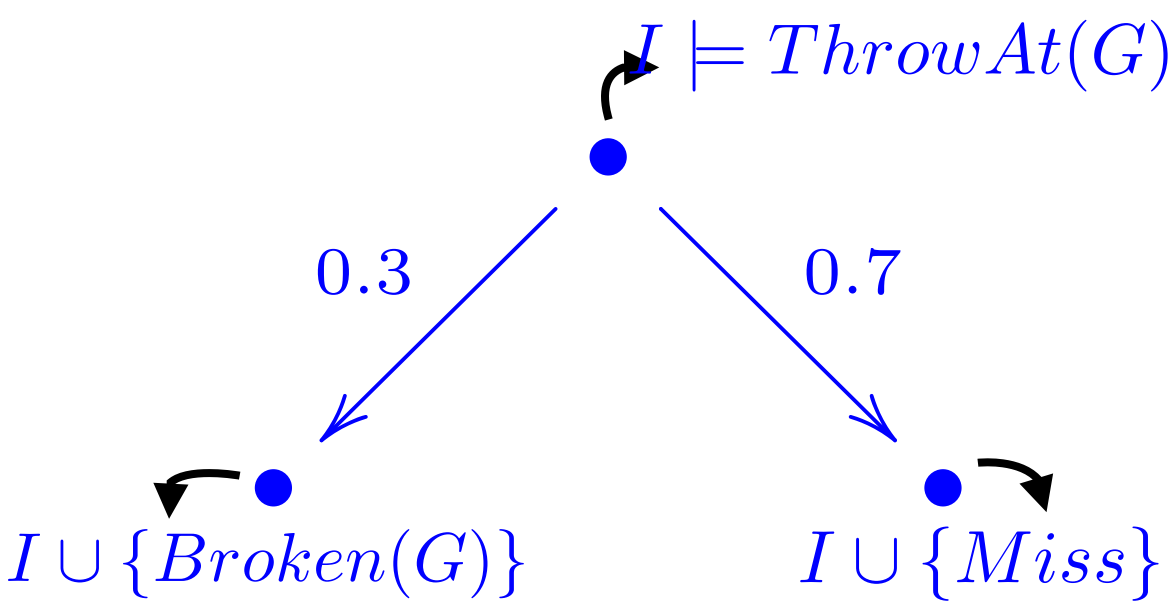
E.g., “throwing a rock at a glass breaks it with probability 0.3 and misses it with probability 0.7”

$$(Broken(G):0.3) \vee (Miss\ 0.7) \leftarrow ThrowAt(G).$$

ote that the actual non-deterministic event (“rock flying at glass”) is implicit

Slides CP-logic courtesy Joost Vennekens

Semantics



(Broken(G) 0.3) ∨ (Miss 0.7)
← ThrowAt(G)

Probability tree is an execution model of theory iff:

- Each tree-transition matches causal law
- The tree cannot be extended
- Each execution model defines the <u>same</u> probability distribution over final states

Slides CP-logic courtesy Joost Vennekens

Continuous Distributions

Distributional Clauses (DC)

- Discrete- and continuous-valued random variables

[Gutmann et al, TPLP 11; Nitti et al, IROS 13]

Distributional Clauses (DC)

- Discrete- and continuous-valued random variables

random variable with Gaussian distribution

```
length(Obj) ~ gaussian(6.0,0.45) :- type(Obj,glass).
```

[Gutmann et al, TPLP 11; Nitti et al, IROS 13]

Distributional Clauses (DC)

- Discrete- and continuous-valued random variables

```
length(Obj) ~ gaussian(6.0,0.45) :- type(Obj,glass).
stackable(OBot,OTop) :-
      ≃length(OBot) ≥ ≃length(OTop),
      ≃width(OBot) ≥ ≃width(OTop).
```

comparing values of random variables

[Gutmann et al, TPLP 11; Nitti et al, IROS 13]

Distributional Clauses (DC)

- Discrete- and continuous-valued random variables

```
length(Obj) ~ gaussian(6.0,0.45) :- type(Obj,glass).
stackable(OBot,OTop) :-
      ≃length(OBot) ≥ ≃length(OTop),
      ≃width(OBot) ≥ ≃width(OTop).
ontype(Obj,plate) ~ finite([0 : glass, 0.0024 : cup,
                            0 : pitcher, 0.8676 : plate,
                            0.0284 : bowl, 0 : serving,
                            0.1016 : none])
                        :- obj(Obj), on(Obj,O2), type(O2,plate).
```

random variable with discrete distribution

[Gutmann et al, TPLP 11; Nitti et al, IROS 13]

Distributional Clauses (DC)

- Discrete- and continuous-valued random variables

```
length(Obj) ~ gaussian(6.0,0.45) :- type(Obj,glass).
stackable(OBot,OTop) :-
      ≃length(OBot) ≥ ≃length(OTop),
      ≃width(OBot) ≥ ≃width(OTop).
ontype(Obj,plate) ~ finite([0 : glass, 0.0024 : cup,
                            0 : pitcher, 0.8676 : plate,
                            0.0284 : bowl, 0 : serving,
                            0.1016 : none])
                         :- obj(Obj), on(Obj,O2), type(O2,plate).
```

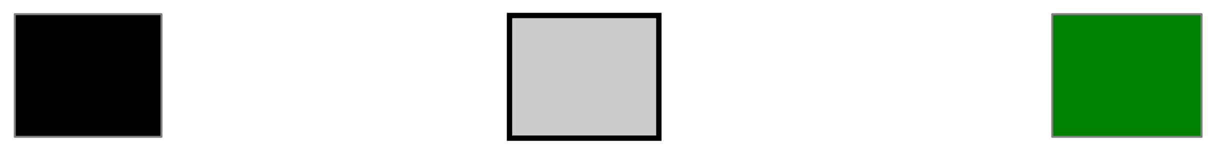
[Gutmann et al, TPLP 11; Nitti et al, IROS 13]

Distributional Clauses (DC)

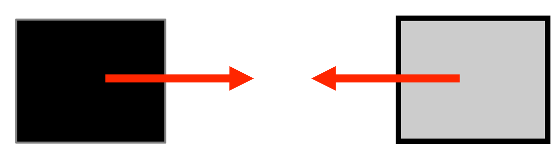
- Defines a generative process (as for CP-logic)
- Tree can become infinitely wide
  - Sampling
- Well-defined under reasonable assumptions
- See Gutmann et al TPLP 11, Nitti et al. 15

Magnetic scenario

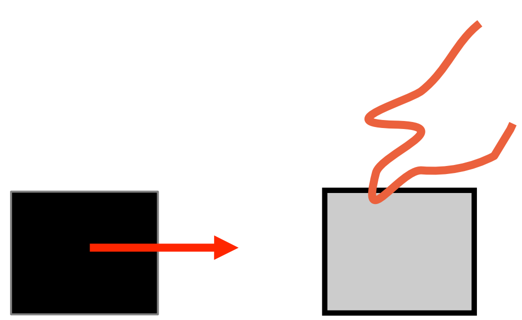
- 3 object types: magnetic, ferromagnetic, nonmagnetic

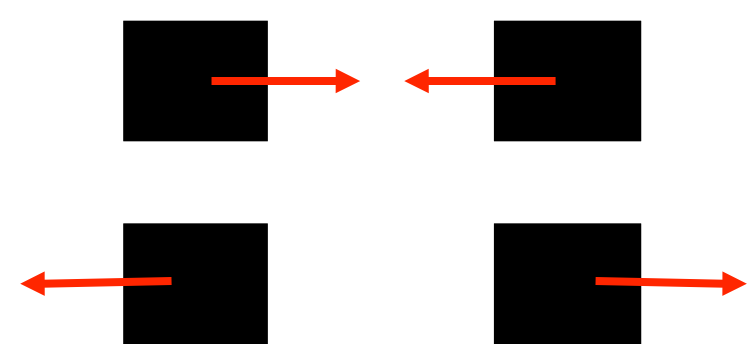


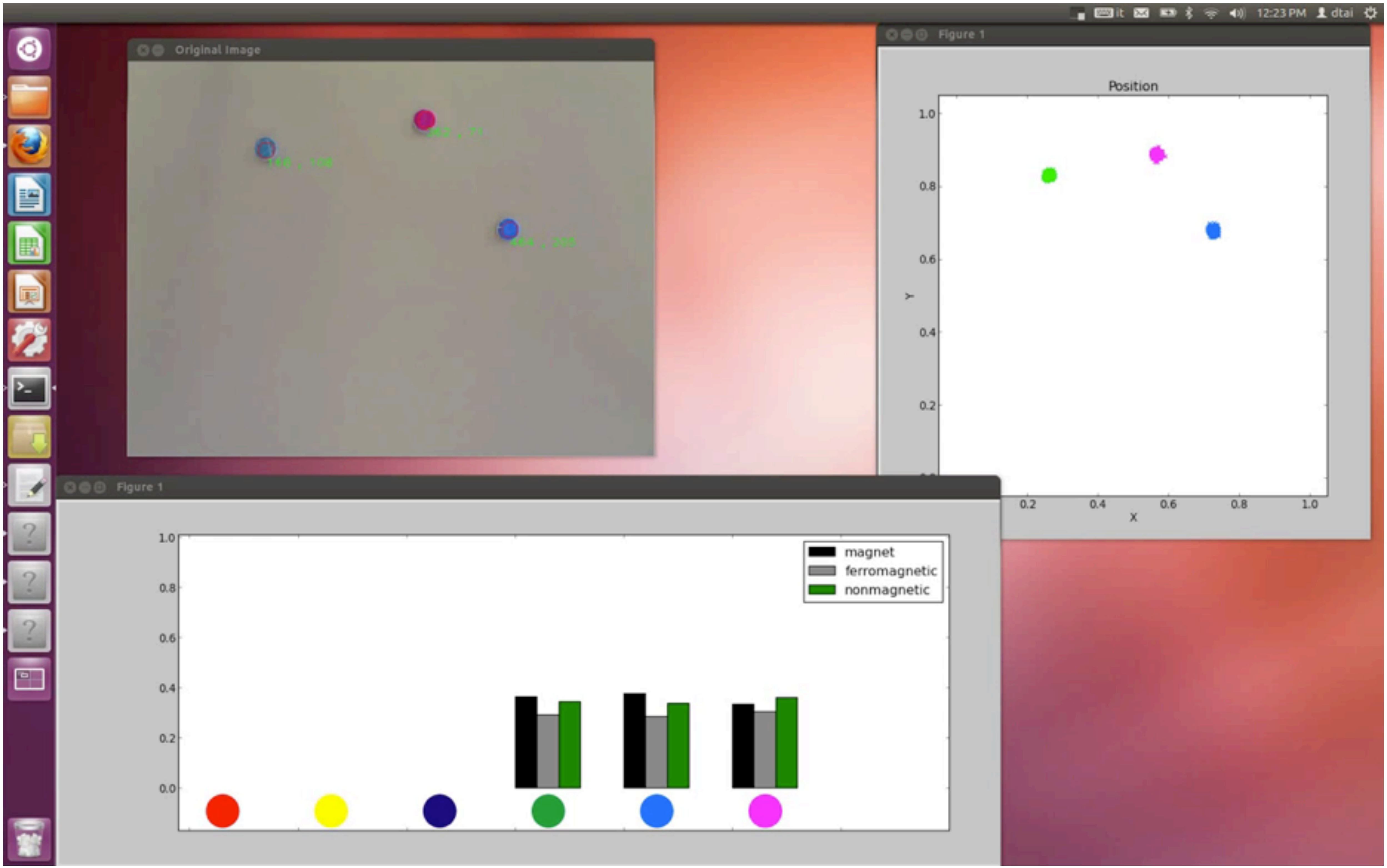
- Nonmagnetic objects do not interact
- A magnet and a ferromagnetic object attract each other



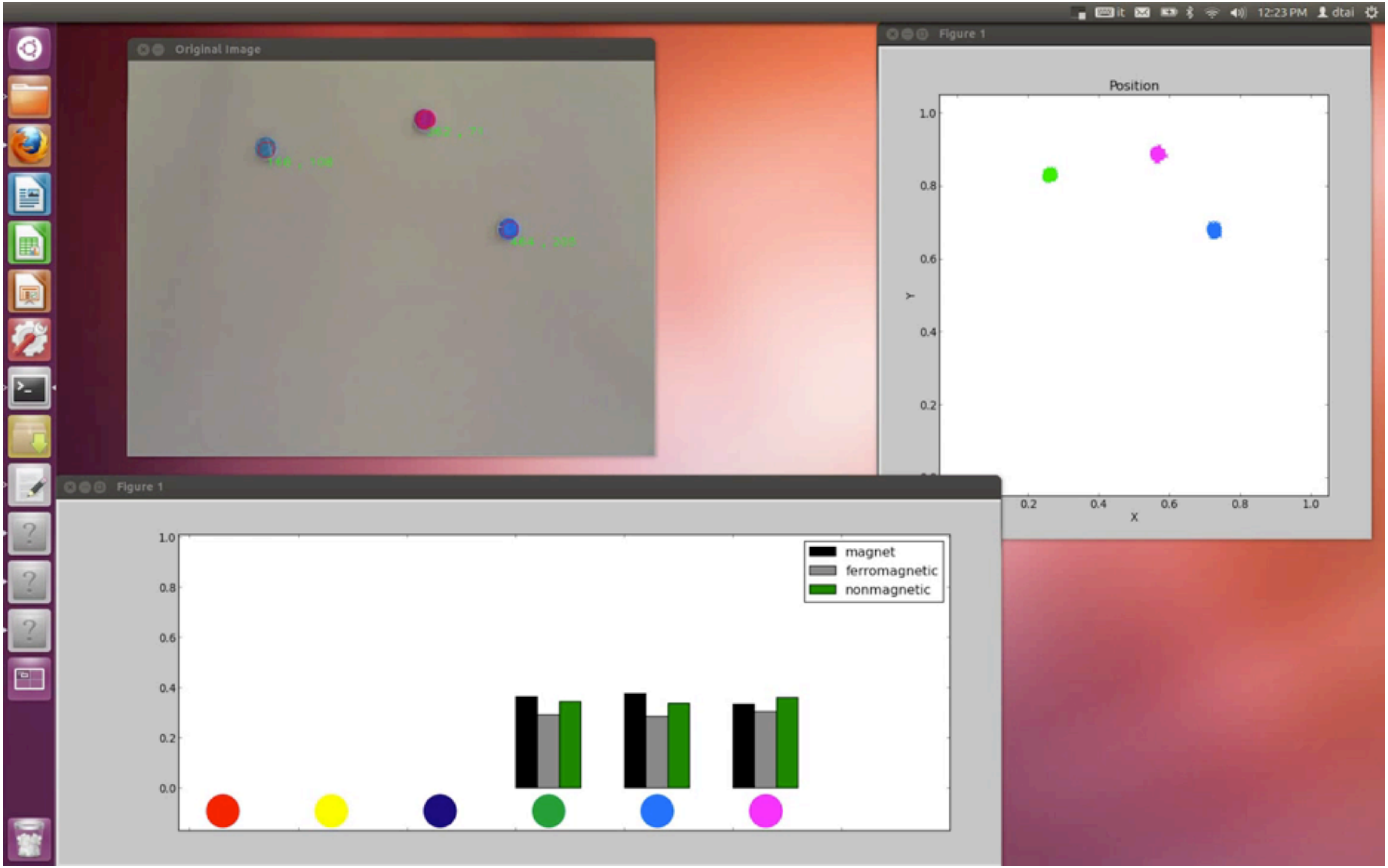
- Magnetic force that depends on the distance
- If an object is held magnetic force is compensated.







12:23 PM
dtai
Original Image
Figure 1
Position
1.0
0.8
0.6
0.4
0.2
Y
0.2
0.4
0.6
0.8
1.0
X
Figure 1
1.0
0.8
0.6
0.4
0.2
0.0
magnet
ferromagnetic
nonmagnetic


12:23 PM
dtai
Original Image
Figure 1
Position
1.0
0.8
0.6
0.4
0.2
0.2
0.4
0.6
0.8
1.0
X
Y
magnet
ferromagnetic
nonmagnetic
0.0

Magnetic scenario

- 3 object types: magnetic, ferromagnetic, nonmagnetic

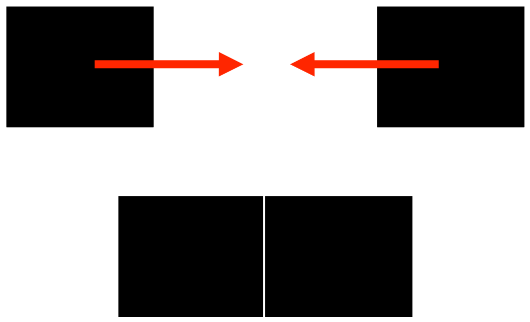
$$\text{type}(X)_t \sim \text{finite}([1/3:\text{magnet},1/3:\text{ferromagnetic},1/3:\text{nonmagnetic}]) \leftarrow \text{object}(X).$$

- 2 magnets attract or repulse

$$\text{interaction}(A,B)_t \sim \text{finite}([0.5:\text{attraction},0.5:\text{repulsion}]) \leftarrow \text{object}(A), \text{object}(B), A<B, \text{type}(A)_t = \text{magnet}, \text{type}(B)_t = \text{magnet}.$$

- Next position after attraction

$$\text{pos}(A)_{t+1} \sim \text{gaussian}(\text{middlepoint}(A,B)_t,\text{Cov}) \leftarrow \text{near}(A,B)_t, \text{not}(\text{held}(A)), \text{not}(\text{held}(B)), \text{interaction}(A,B)_t = \text{attr}, c/\text{dist}(A,B)_t^2 > \text{friction}(A)_t.$$



$$\text{pos}(A)_{t+1} \sim \text{gaussian}(\text{pos}(A)_t,\text{Cov}) \leftarrow \text{not}(\text{ attraction}(A,B)).$$

Probabilistic Programs

- Distributional clauses / PLP similar in spirit
  - to e.g. BLOG, ... but embedded in existing logic and programming language
  - to e.g. Church but use of logic instead of functional programming ...
    - natural possible world semantics and link with prob. databases.
    - somewhat harder to do meta-programming

Markov Logic

Key differences

- programming language
- Dist. Sem. uses least-fix point semantics
  - can express transitive closure of relation
  - this cannot be expressed in FOL (and Markov Logic), requires second order logic
  - p(X,Y) :- p(X,Z), p(Z,Y).

Inference in PLP

- As in Prolog and logic programming
  - proof-based
- As in Answer Set Programming
  - model based
- As in Probabilistic Programming
  - sampling

Logical Reasoning: Proofs in Prolog

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X),
     smokes(Y).
```

Logical Reasoning: Proofs in Prolog

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
      influences(Y,X),
      smokes(Y).
```

```
?- smokes(carl).
```

Logical Reasoning: Proofs in Prolog

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
      influences(Y,X),
      smokes(Y).
```

```
            ?- smokes(carl).

  ?- stress(carl).
```

Logical Reasoning: Proofs in Prolog

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X),
     smokes(Y).
```

```
?- smokes(carl).

?- stress(carl).        ?- influences(Y,carl),smokes(Y).
```

Logical Reasoning: Proofs in Prolog

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X),
     smokes(Y).
```

```
?- smokes(carl).

?- stress(carl).        ?- influences(Y,carl),smokes(Y).
```

Logical Reasoning: Proofs in Prolog

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X),
     smokes(Y).
```

```
?- smokes(carl).

?- stress(carl).

?- influences(Y,carl),smokes(Y).

?- smokes(bob).
```

Logical Reasoning: Proofs in Prolog

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
      influences(Y,X),
      smokes(Y).
```

```
?- smokes(carl).

?- stress(carl).        ?- influences(Y,carl),smokes(Y).

                            ?- smokes(bob).

      ?- stress(bob).
```

Logical Reasoning: Proofs in Prolog

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
      influences(Y,X),
      smokes(Y).
```

```
?- smokes(carl).

?- stress(carl).
?- influences(Y,carl),smokes(Y).

?- smokes(bob).

?- stress(bob).
?- influences(Y1,bob),smokes(Y1).
```

Logical Reasoning: Proofs in Prolog

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X),
     smokes(Y).
```

```
?- smokes(carl).
?- stress(carl).
?- influences(Y,carl),smokes(Y).
?- smokes(bob).
?- stress(bob).
?- influences(Y1,bob),smokes(Y1).
```

Logical Reasoning: Proofs in Prolog

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X),
     smokes(Y).
```

```
?- smokes(carl).
?- stress(carl).
?- influences(Y,carl),smokes(Y).
?- smokes(bob).
?- stress(bob).
?- influences(Y1,bob),smokes(Y1).
?- smokes(ann).
```

Logical Reasoning: Proofs in Prolog

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X),
     smokes(Y).
```

```
?- smokes(carl).

?- stress(carl).          ?- influences(Y,carl),smokes(Y).

?- smokes(bob).

?- stress(bob).           ?- influences(Y1,bob),smokes(Y1).

?- smokes(ann).

?- stress(ann).
```

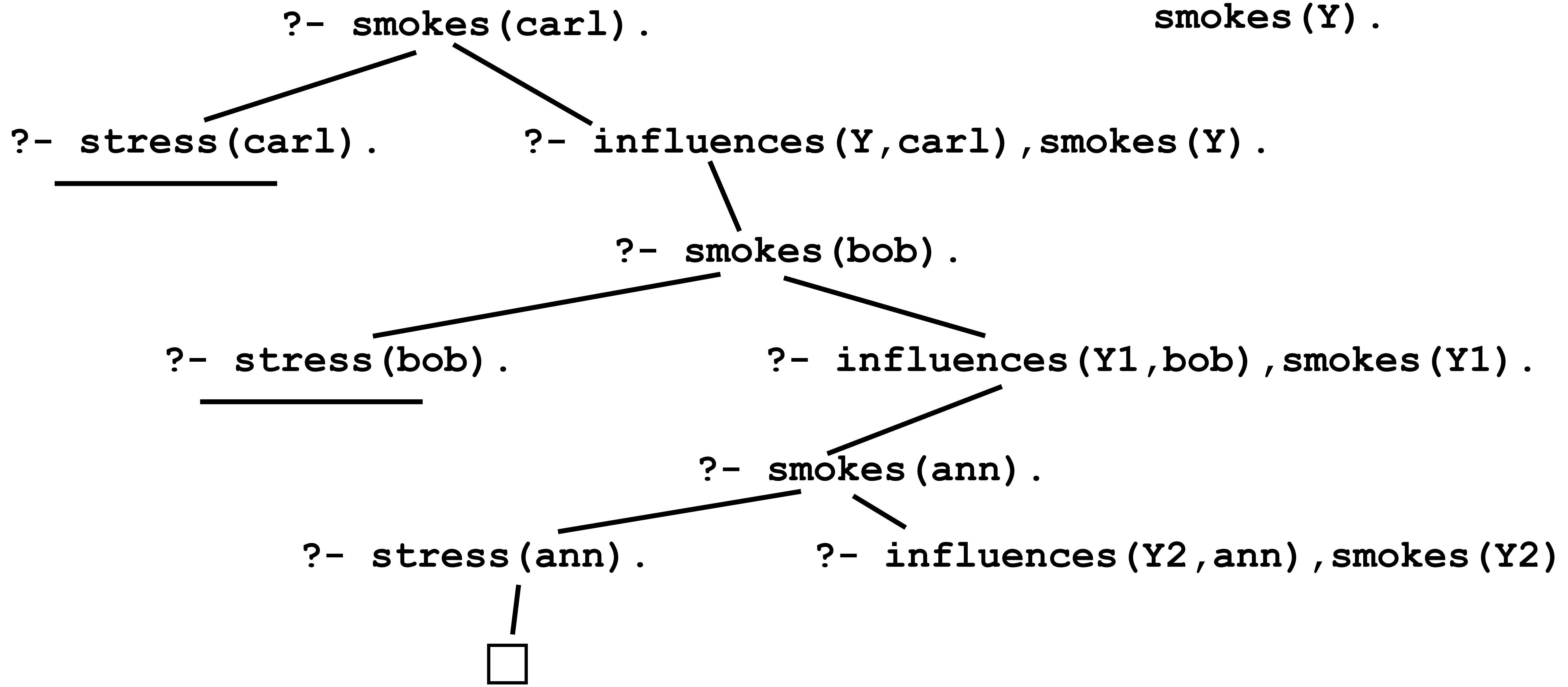
Logical Reasoning: Proofs in Prolog

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X),
     smokes(Y).
```

```
?- smokes(carl).

?- stress(carl).          ?- influences(Y,carl),smokes(Y).

?- smokes(bob).

?- stress(bob).          ?- influences(Y1,bob),smokes(Y1).

?- smokes(ann).

?- stress(ann).          ?- influences(Y2,ann),smokes(Y2).
```

Logical Reasoning: Proofs in Prolog

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X),
     smokes(Y).
```



Logical Reasoning: Proofs in Prolog

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X),
     smokes(Y).
```

- `?- smokes(carl).`
  - `?- stress(carl).`
  - `?- influences(Y,carl),smokes(Y).`
    - `?- smokes(bob).`
      - `?- stress(bob).`
      - `?- influences(Y1,bob),smokes(Y1).`
        - `?- smokes(ann).`
          - `?- stress(ann).`
            - □
          - `?- influences(Y2,ann),smokes(Y2).`

Logical Reasoning: Proofs in Prolog

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
      influences(Y,X),
      smokes(Y).
```

```
?- smokes(carl).
?- stress(carl).
?- influences(Y,carl),smokes(Y).
```

Y=bob

```
?- smokes(bob).
?- stress(bob).
?- influences(Y1,bob),smokes(Y1).
```

Y1=ann

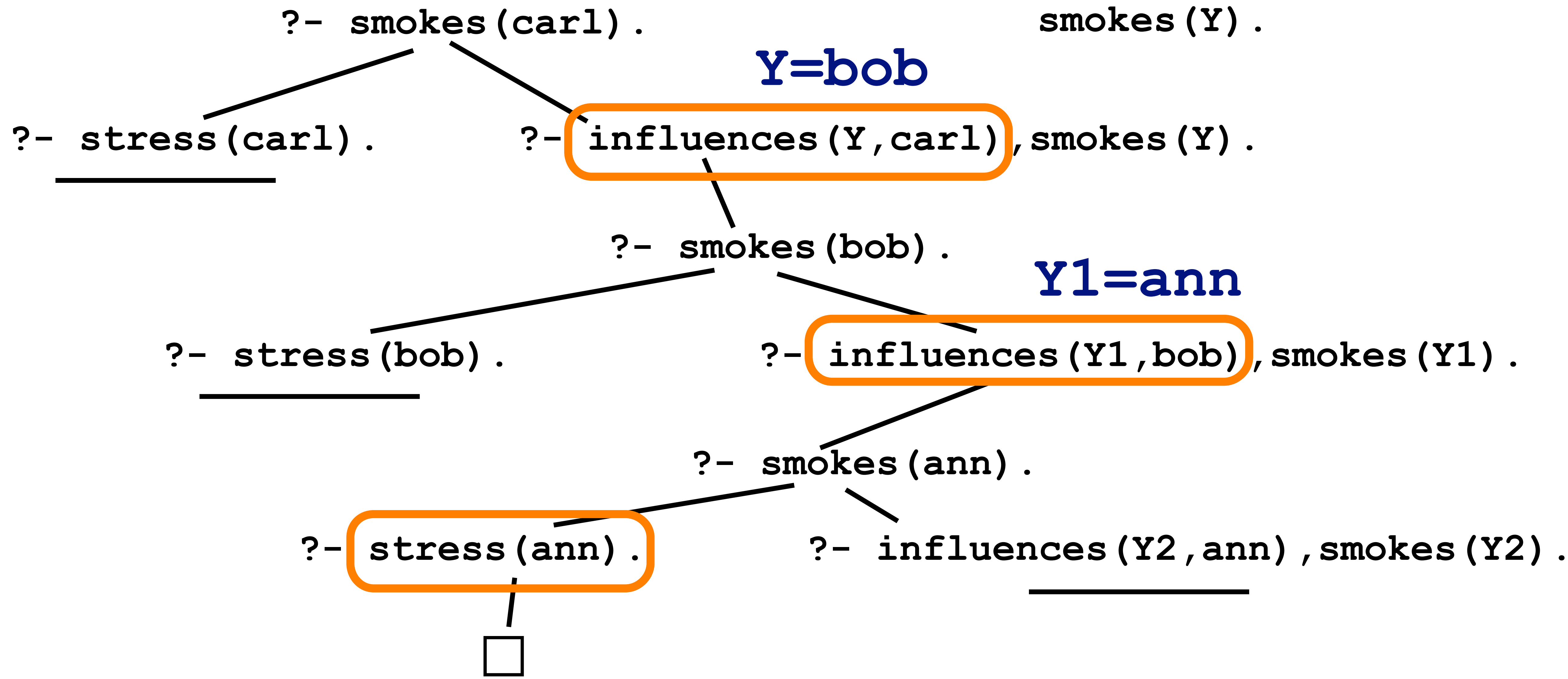
```
?- smokes(ann).
?- stress(ann).
?- influences(Y2,ann),smokes(Y2).
□
```

proof = facts used in successful derivation:

`influences(bob,carl)&influences(ann,bob)&stress(ann)`

Proofs in ProbLog

```
0.8::stress(ann).
0.6::influences(ann,bob).
0.2::influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
      influences(Y,X),
      smokes(Y).
```



influences(bob,carl)&influences(ann,bob)&stress(ann)

probability of proof = 0.2 × 0.6 × 0.8 = 0.096

Proofs in ProbLog

```
0.8::stress(ann).
0.4::stress(bob).
0.6::influences(ann,bob).
0.2::influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
      influences(Y,X),
      smokes(Y).
```

```
?- smokes(carl).
?- stress(carl).
?- influences(Y,carl),smokes(Y).
```

Y=bob

```
?- smokes(bob).
?- stress(bob).
?- influences(Y1,bob),smokes(Y1).
```

□

Y1=ann

```
?- smokes(ann).
?- stress(ann).
?- influences(Y2,ann),smokes(Y2).
```

□

```
influences(bob,carl)
& influences(ann,bob)
& stress(ann)
```

$0.2 \times 0.6 \times 0.8$

$= 0.096$

Proofs in ProbLog

```
0.8::stress(ann).
0.4::stress(bob).
0.6::influences(ann,bob).
0.2::influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
        influences(Y,X),
        smokes(Y).
```

?- smokes(carl).

?- stress(carl).

?- influences(Y,carl),smokes(Y).

Y=bob

?- smokes(bob).

?- stress(bob).

□

?- influences(Y1,bob),smokes(Y1).

Y1=ann

?- smokes(ann).

?- stress(ann).

?- influences(Y2,ann),smokes(Y2).

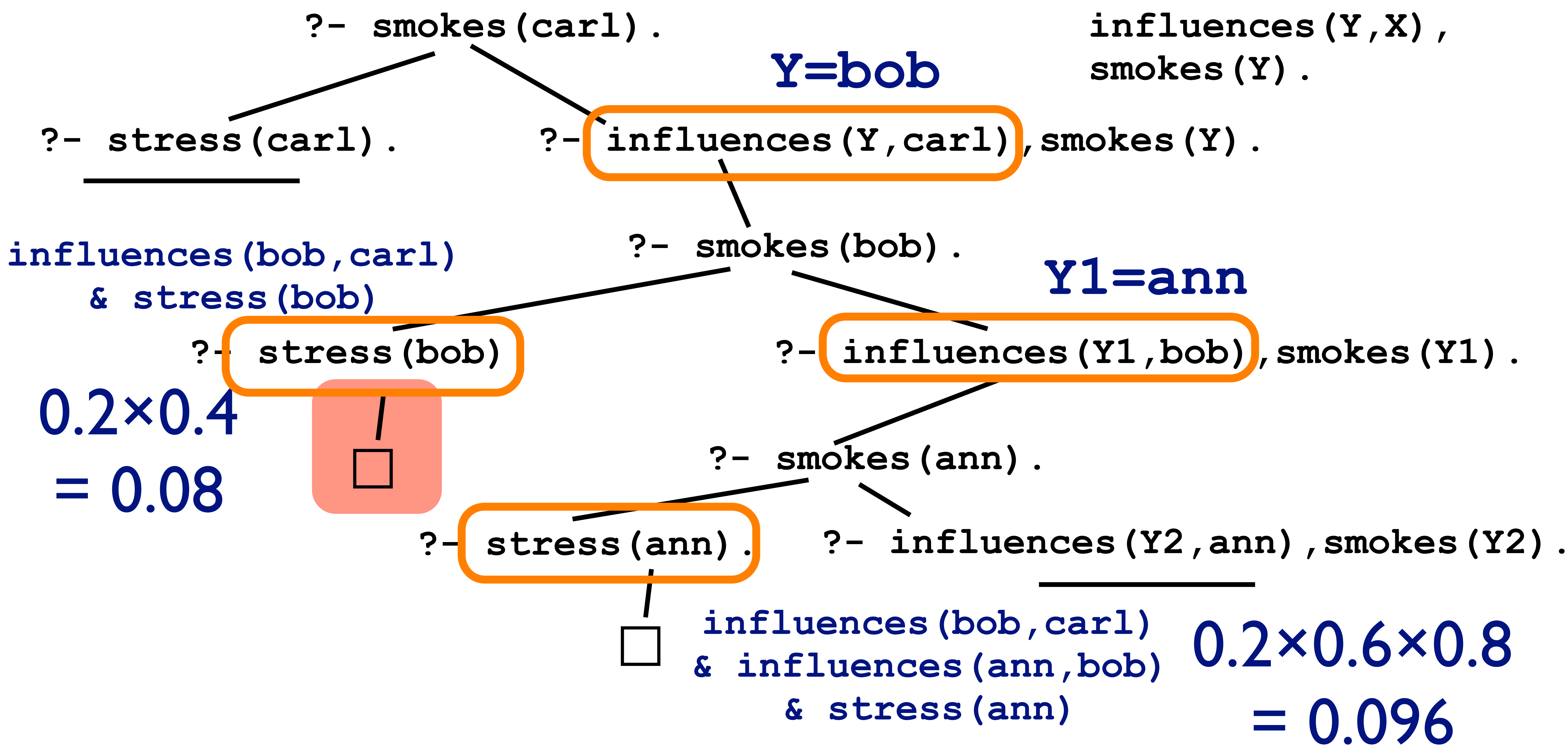
□

influences(bob,carl)
& influences(ann,bob)
& stress(ann)

$0.2 \times 0.6 \times 0.8$
$= 0.096$

Proofs in ProbLog

```
0.8::stress(ann).
0.4::stress(bob).
0.6::influences(ann,bob).
0.2::influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
        influences(Y,X),
        smokes(Y).
```



Proofs in ProbLog

```
0.8::stress(ann).
0.4::stress(bob).
0.6::influences(ann,bob).
0.2::influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
      influences(Y,X),
      smokes(Y).
```

?- smokes(carl).

?- stress(carl).

?- influences(Y,carl),smokes(Y).

Y=bob

?- smokes(bob).

influences(bob,carl)
& stress(bob)

?- stress(bob).

$\square$

0.2×0.4
= 0.08

?- influences(Y1,bob),smokes(Y1).

Y1=ann

?- smokes(ann).

?- stress(ann).

$\square$

?- influences(Y2,ann),smokes(Y2).

influences(bob,carl)
& influences(ann,bob)
& stress(ann)

0.2×0.6×0.8
= 0.096

proofs overlap!
cannot sum probabilities
(disjoint-sum-problem)

Part II : Synthesising Probabilistic Programs

Information Extraction in NELL

Recently-Learned Facts twitter

Refresh

instance	iteration	date learned	confidence
kelly_andrews is a female	826	29-mar-2014	98.7
investment_next_year is an economic sector	829	10-apr-2014	95.3
shibenik is a geopolitical entity that is an organization	829	10-apr-2014	97.2
quality_web_design_work is a character trait	826	29-mar-2014	91.0
mercedes_benz_cls_by_carlsson is an automobile manufacturer	829	10-apr-2014	95.2
social_work is an academic program at the university rutgers_university	827	02-apr-2014	93.8
dante wrote the book the_divine_comedy	826	29-mar-2014	93.8
willie_aames was born in the city los_angeles	831	16-apr-2014	100.0
kitt_peak is a mountain in the state or province arizona	831	16-apr-2014	96.9
greenwich is a park in the city london	831	16-apr-2014	100.0

instances for many different relations

degree of certainty

NELL: http://rtw.ml.cmu.edu/rtw/

Rule learning in NELL (1)

- Original approach
  - Make probabilistic data deterministic
  - run classic rule-learner (variant of FOIL)
  - re-introduce probabilities on learned rules and predict

Rule learning in NELL (2)

- Newer Page Rank Based Approach (Cohen et al.) -- ProPPR
    - Change the underlying model, from random graph / database to random walk one;
    - No longer "degree of belief" assigned to facts;
    - more like stochastic logic programs
    - Learn rules / parameters

Probabilistic Rule Learning

- Learn the rules directly in a PLP setting
- Generalize relational learning and inductive logic programming directly towards probabilistic setting
- Traditional rule learning/ILP as a special case
- Apply to probabilistic databases like NELL
- ILP 10, IJCAI 15

Pro Log

surfing(X) :- not pop(X) and windok(X).

H

surfing(X) :- not pop(X) and sunshine(X).

pop(e1).   windok(e1).   sunshine(e1).   **B**

?-surfing(e1). **e**
no

B U H |=\= e   (H does not cover e)

An ILP example

ProbLog

a probabilistic Prolog

p1:: surfing(X) :- not pop(X) and windok(X).

H

p2:: surfing(X) :- not pop(X) and sunshine(X).

0.2::pop(e1). 0.7::windok(e1). 0.6::sunshine(e1).

B

?-P(surfing(e1)).**e**

gives (1-0.2) x 0.7 x p1 + (1-0.2) x 0.6 x (1-0.7) x p2 = P(B U H |= e)

not pop x windok x p1 + not pop x sunshine x (not windok) x p1

probability that the example is covered

Inductive Probabilistic Logic Programs

Given

a set of example facts e ∈ E together with the probability p that they hold

a background theory B in ProbLog

a hypothesis space L (a set of clauses)

Find

$$\arg\min_H loss(H, B, E) = \arg\min_H \sum_{e_i \in E} |P_s(B \cup H \models e) - p_i|$$

Observations

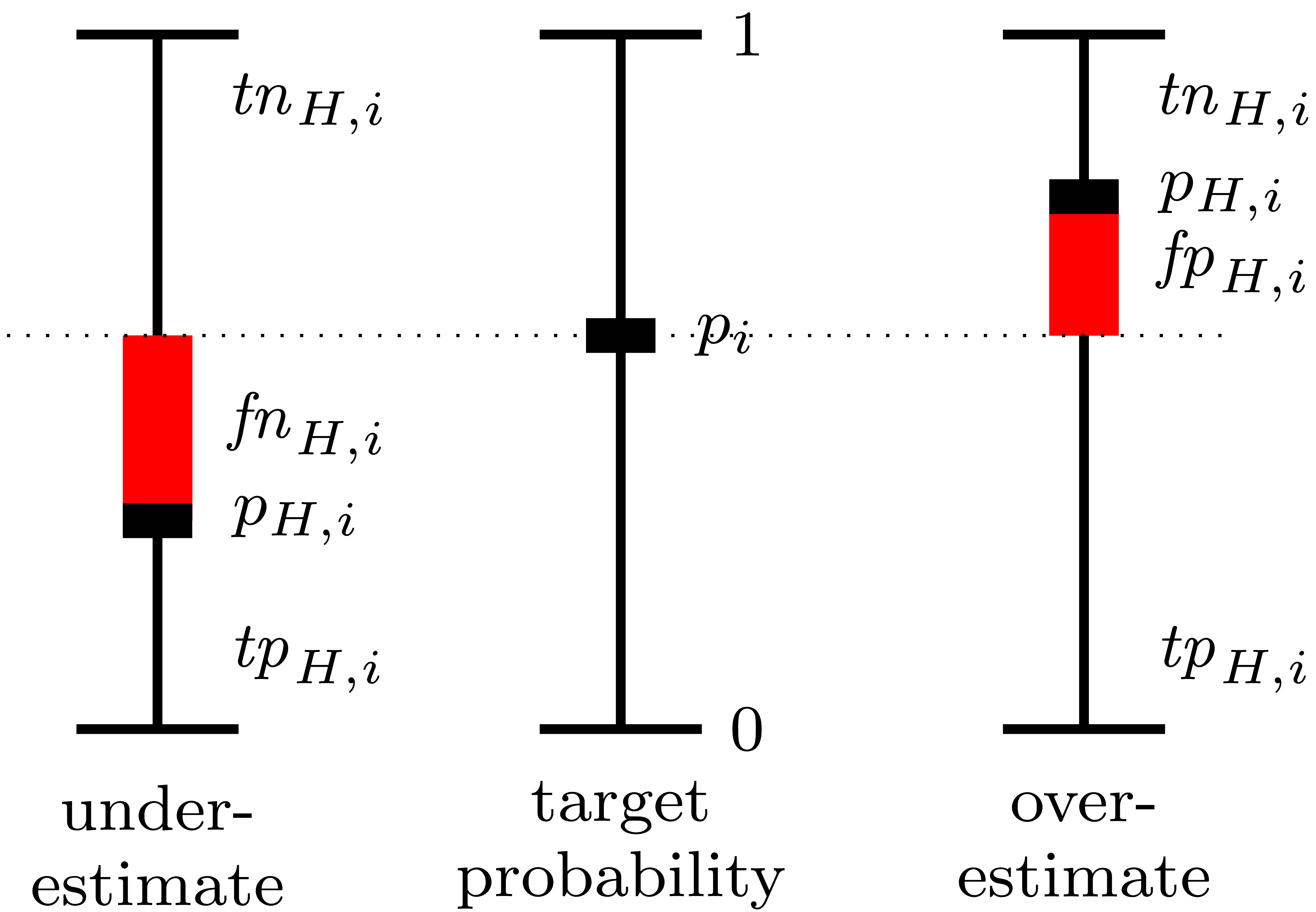
Propositional versus first order

- traditional rule learning = propositional
- inductive logic programming = first order

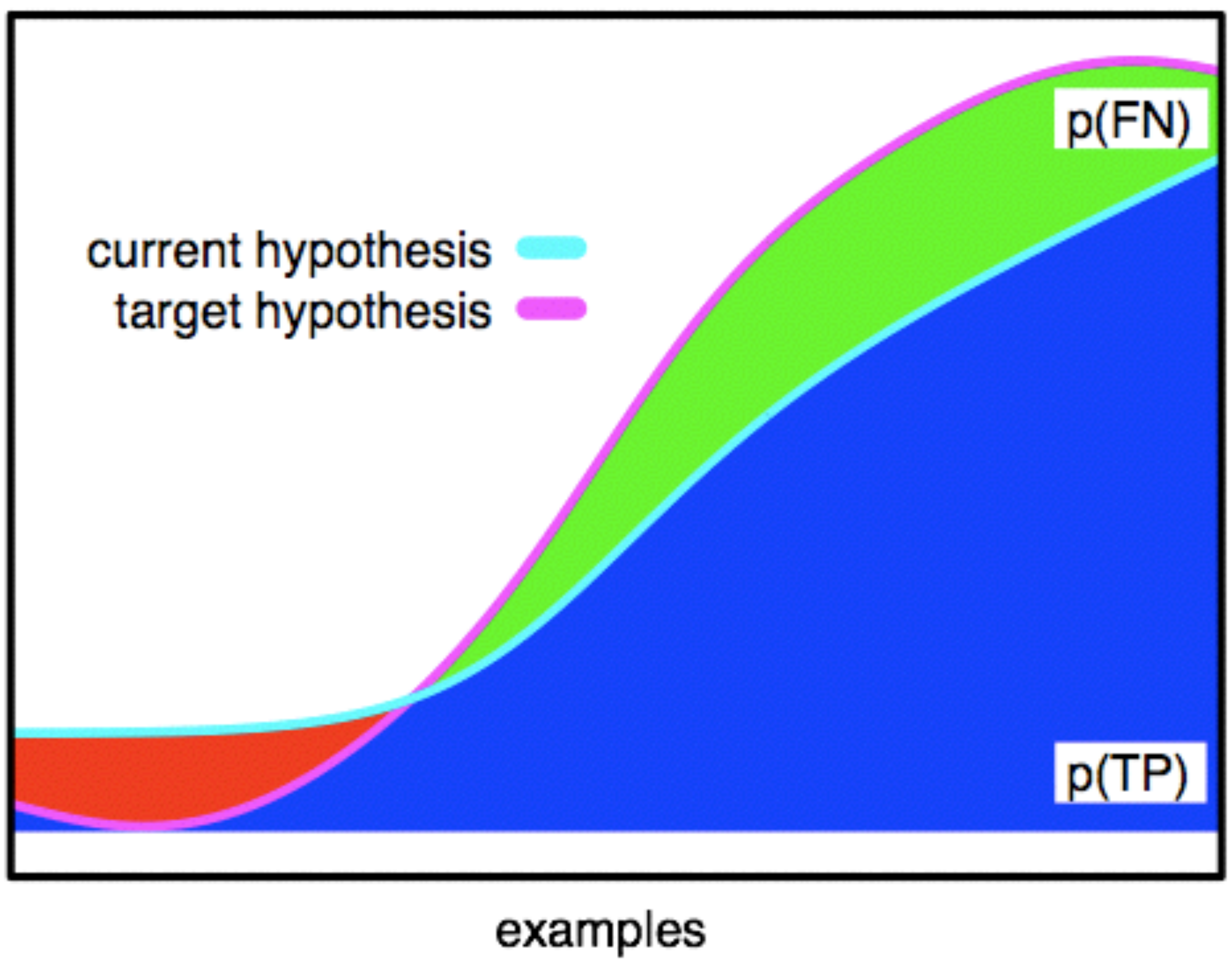
Deterministic case

- all probabilities 0 or 1
- traditional rule learning / ILP as special case

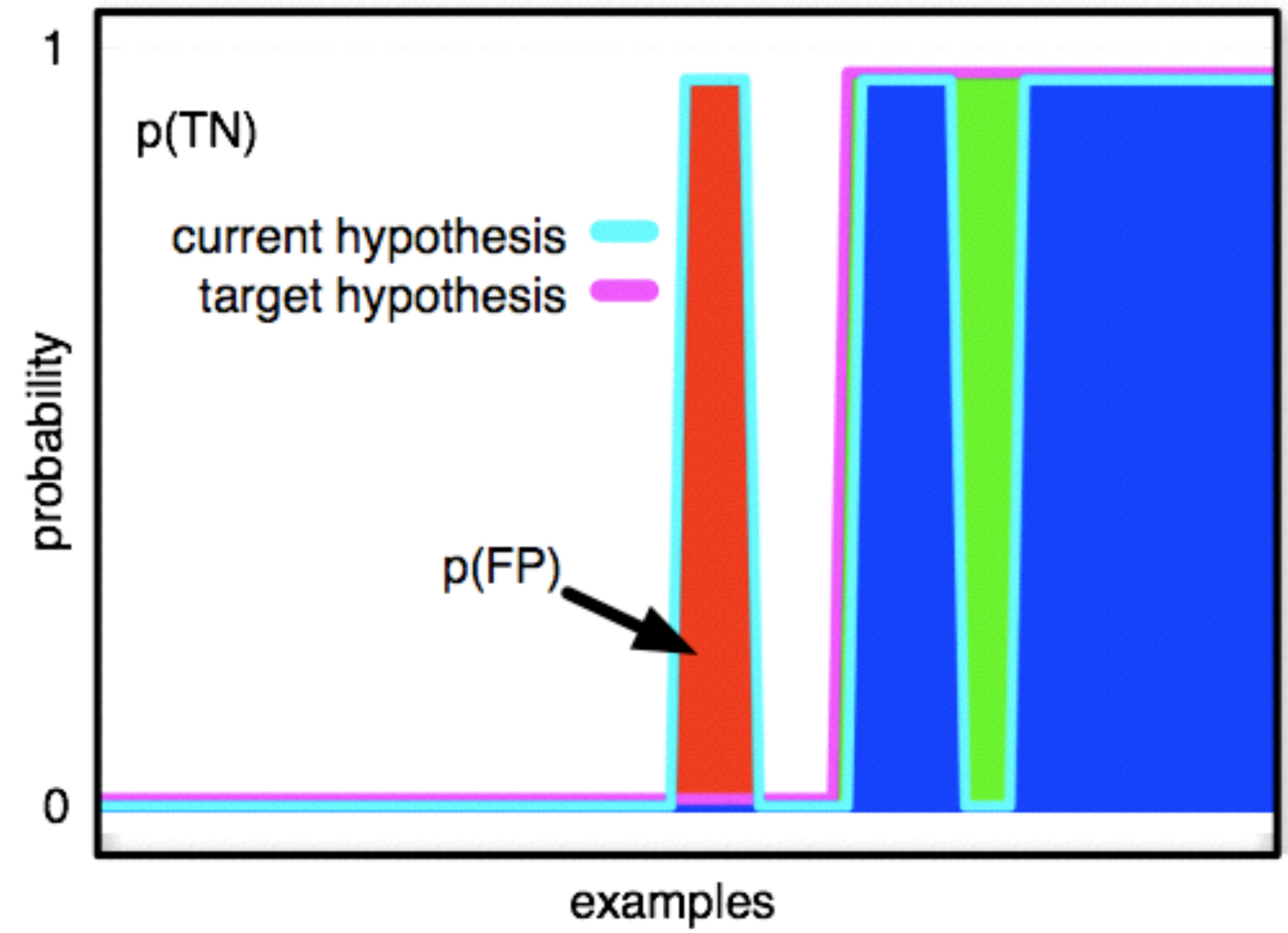
Analysis


1
$tn_{H,i}$
$fn_{H,i}$
$p_{H,i}$
$tp_{H,i}$
p_i
0
$tn_{H,i}$
$p_{H,i}$
$fp_{H,i}$
$tp_{H,i}$
under-
estimate
target
probability
over-
estimate

Analysis



p(FN)
current hypothesis
target hypothesis
p(TP)
examples



1
p(TN)
current hypothesis
target hypothesis
probability
p(FP)
0
examples

Rule learning

Interesting properties

- adding a rule is monotonic, this can only increase the probability of an example
- adding a condition to a rule is anti-monotonic, this can only decrease the probability of an example
- several rules may be needed to cover an example
    - use all examples all of the time (do not delete them while learning), do not forget the positives
    - disjoint sum problem

ProbFOIL

Quinlan's well-known FOIL algorithm combined with ProbLog and probabilistic examples and background knowledge

Essentially a vanilla sequential covering algorithm with m-estimate as local score and accuracy as global score.

Criteria

$$
\begin{aligned}
precision &= \frac{TP}{TP + FP} \\
m\text{-}estimate &= \frac{TP + m \cdot \frac{P}{N}}{TP + FP + m} \\
recall &= \frac{TP}{TP + FN} \\
accuracy &= \frac{TP + TN}{TP + TN + FP + FN}
\end{aligned}
$$

local score

global score

Avoiding overfitting using significance test

ProbFOIL

Algorithm 1 The ProbFOIL$^+$ learning algorithm

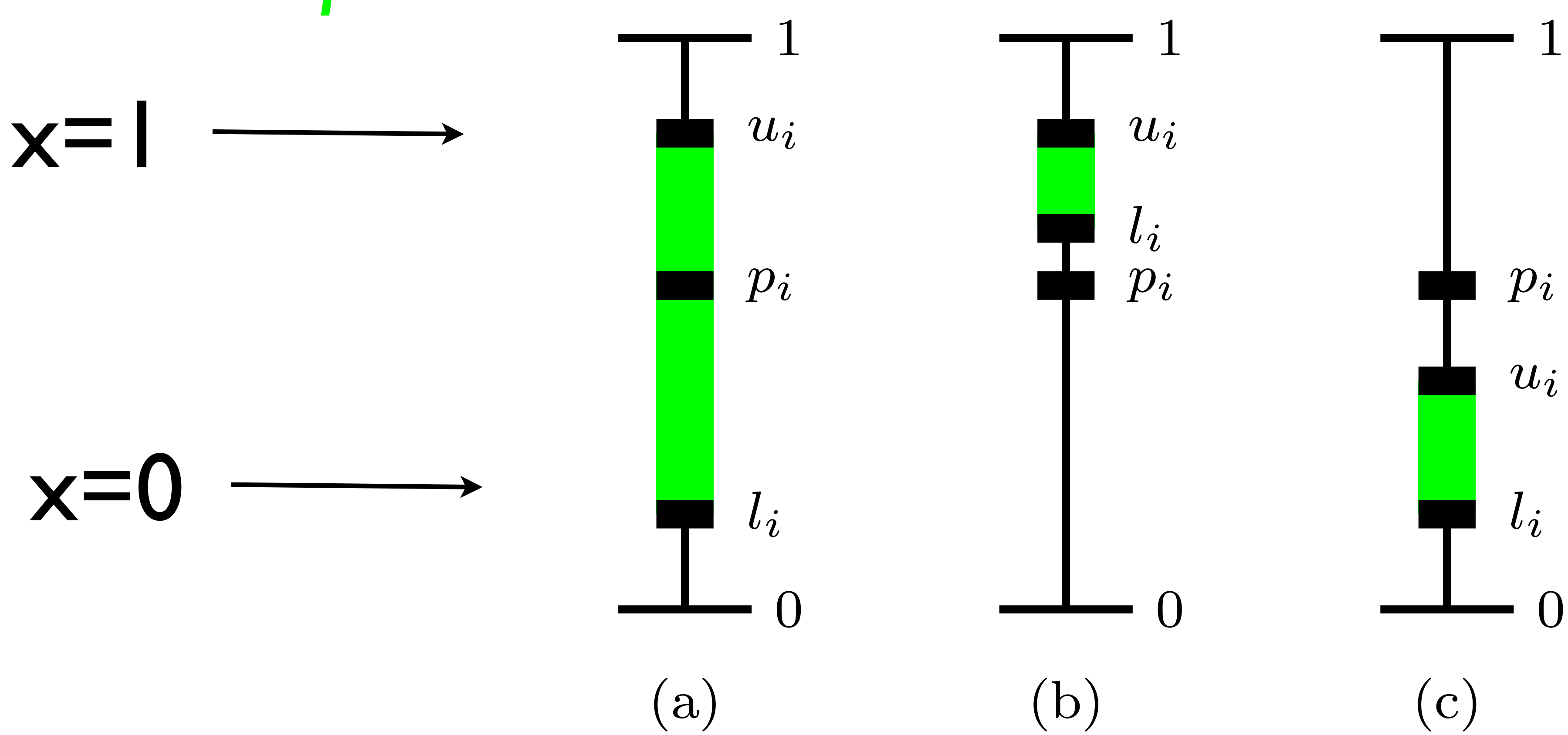
```
function ProbFOIL+(target)                                   ▷ target is the target predicate
    H := ∅
    while true do
        clause := LearnRule(H, target)
        if GlobalScore(H) < GlobalScore(H ∪ {clause}) then
            H := H ∪ {clause}
        else
            return H
function LearnRule(H, target)
    candidates := {x :: target ← true}                       ▷ Start with an empty (probabilistic) body
    bestrule := (x :: target ← true)
    while candidates ≠ ∅ do                                  ▷ Grow rule
        nextcandidates := ∅
        for all x :: target ← body ∈ candidates do
            for all literal ∈ ρ(target ← body) do             ▷ Generate all refinements
                if not RejectRefinement(H, bestrule, x :: target ← body) then    ▷ Reject unsuited refinements
                    nextcandidates := nextcandidates ∪ {x :: target ← body ∧ l}
                    if LocalScore (H, x :: target ← body ∧ literal) > LocalScore(H, bestrule) then
                        bestrule := (x :: target ← body ∧ literal)            ▷ Update best rule
        candidates := nextcandidates
    return bestrule
```

Extended rule learning

Learn rules with probability *x:: head :- body*

What changes ?

- value of *x* *determines prob. of coverage of example*



Extended rule learning

Express local score as a function of x

Compute optimal value of x

NELL

Table 5: Number of facts per predicate (NELL athlete dataset)

athletecoach(person,person)	18	athleteplaysforteam(person,team)	721
athleteplayssport(person,sport)	1921	teamplaysinleague(team,league)	1085
athleteplaysinleague(person,league)	872	athletealsoknownas(person,name)	17
coachesinleague(person,league)	93	coachesteam(person,team)	132
teamhomestadium(team,stadium)	198	teamplayssport(team,sport)	359
athleteplayssportsteamposition(person,position)	255	athletehomestadium(person,stadium)	187
athlete(person)	1909	attraction(stadium)	2
coach(person)	624	female(person)	2
male(person)	7	hobby(sport)	5
organization(league)	1	person(person)	2
personafrica(person)	1	personasia(person)	4
personaustralia(person)	22	personcanada(person)	1
personeurope(person)	1	personmexico(person)	108
personus(person)	6	sport(sport)	36
sportsleague(league)	18	sportsteam(team)	1330
sportsteamposition(position)	22	stadiumoreventvenue(stadium)	171

athleteplaysforteam

```
athleteplaysforteam(A,B) :- coachesteam(A,B).
0.875::athleteplaysforteam(A,B) :- teamhomestadium(B,C), athletehomestadium(A,C).
0.99080::athleteplaysforteam(A,B) :- teamhomestadium(B,_), male(A), athleteplayssport(A,_).
0.75::athleteplaysforteam(A,B) :- teamhomestadium(B,_), athleteplaysinleague(A,C), teamplaysinleague(B,C),
 athlete(A).
0.75::athleteplaysforteam(A,B) :- teamplayssport(B,C), athleteplayssport(A,C), coach(A), teamplaysinleague(B,_).
0.97555::athleteplaysforteam(A,B) :- personus(A), teamplayssport(B,_).
0.762::athleteplaysforteam(A,B) :- teamplayssport(B,C), athleteplayssport(A,C), personmexico(A),
 teamplaysinleague(B,_).
0.52571::athleteplaysforteam(A,B) :- teamplayssport(B,C), athleteplayssport(A,C), athleteplaysinleague(A,_),
 teamplaysinleague(B,_), athlete(A), teamplayssport(B,C).
0.50546::athleteplaysforteam(A,B) :- teamplayssport(B,_), teamplaysinleague(B,C), athleteplaysinleague(A,C),
 athleteplayssport(A,_).
0.50::athleteplaysforteam(A,B) :- teamplayssport(B,_), teamplaysinleague(B,C), athleteplaysinleague(A,C).
0.52941::athleteplaysforteam(A,B) :- teamplayssport(B,_), teamhomestadium(B,_), coach(A), teamplaysinleague(B,_).
0.55287::athleteplaysforteam(A,B) :- teamplayssport(B,_), teamplaysinleague(B,C), athleteplaysinleague(A,C),
 athlete(A).
0.46875::athleteplaysforteam(A,B) :- teamplayssport(B,_), teamplaysinleague(B,_), coach(A),
 teamhomestadium(B,_).
```

Experiments

Table 4: Precision for different experimental setups and parameters (A: $m = 1, p = 0.99$, B: $m = 1000, p = 0.90$).

Setting	athleteplaysforteam		athleteplayssport		teamplaysinleague		athleteplaysinleague		teamplaysagainstteam	
train/test/rule	**A**	**B**	**A**	**B**	**A**	**B**	**A**	**B**	**A**	**B**
1: det/det/det	74.00	69.36	94.14	93.47	96.29	82.15	80.95	74.14	73.40	73.86
2: det/prob/det	73.51	69.57	97.53	94.85	96.70	87.83	90.83	77.73	73.70	73.35
3: det/prob/prob	74.67	69.82	95.86	94.74	96.35	82.57	82.26	75.29	73.84	74.34
4: det/prob/prob	77.25	73.87	96.53	96.04	98.00	90.59	84.91	79.36	77.26	77.83
5: det/prob/prob	74.76	69.97	95.85	94.69	96.44	82.51	81.99	75.07	73.90	74.16
6: prob/prob/det	75.83	73.11	93.40	93.76	94.44	93.67	79.41	79.42	80.87	80.60
7: prob/prob/prob	78.31	73.72	95.62	95.10	98.84	91.86	96.94	79.49	85.78	81.81

Table 3: Learned relational rules for the different predicates (fold 1).

0.9375::athleteplaysforteam(A,B)	←	athleteledsportsteam(A,B).
0.9675::athleteplaysforteam(A,B)	←	athleteledsportsteam(A,V1), teamplaysagainstteam(B,V1).
0.9375::athleteplaysforteam(A,B)	←	athleteplayssport(A,V1), teamplayssport(B,V1).
0.5109::athleteplaysforteam(A,B)	←	athleteplaysinleague(A,V1), teamplaysinleague(B,V1).
0.9070::athleteplayssport(A,B)	←	athleteledsportsteam(A,V2), teamalsoknownas(V2,V1), teamplayssport(V1,B), teamplayssport(V2,B).
0.9070::athleteplayssport(A,B)	←	athleteplaysforteam(A,V2), teamalsoknownas(V2,V1), teamplayssport(V1,B), teamplayssport(V2,B),teamalsoknownas(V1,V2).
0.9070::athleteplayssport(A,B)	←	athleteplaysforteam(A,V1), teamplayssport(V1,B).
0.9286::athleteplaysinleague(A,B)	←	athleteledsportsteam(A,V1), teamplaysinleague(V1,B).
0.7868::athleteplaysinleague(A,B)	←	athleteplaysforteam(A,V2), teamalsoknownas(V2,V1), teamplaysinleague(V1,B).
0.9384::athleteplaysinleague(A,B)	←	athleteplayssport(A,V2), athleteplayssport(V1,V2), teamplaysinleague(V1,B).
0.9024::athleteplaysinleague(A,B)	←	athleteplaysforteam(A,V1), teamplaysinleague(V1,B).

Rule learning summary

Learning rules (or inducing logic programs) from uncertain/probabilistic data

A new problem formulation

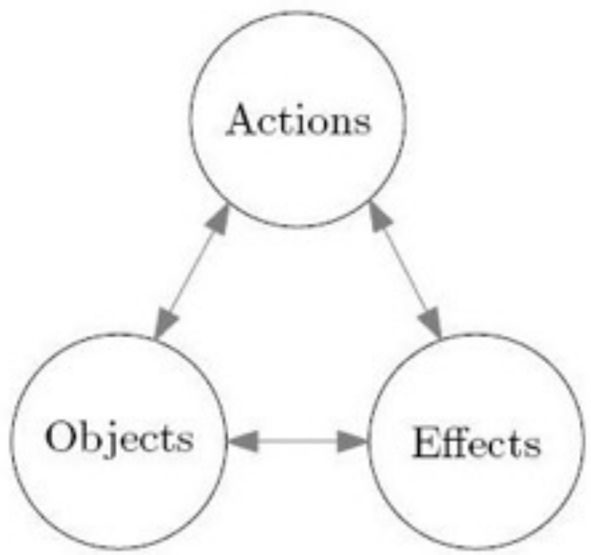
Traditional rule learning (ILP) is the deterministic special case

Traditional rule learning principles apply directly (including ROC analysis)

Affordances with DCs

Affordances

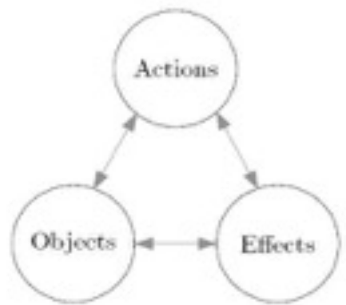
- Model captures action opportunities
  - What can one do with an object?
- Three main aspects:
  - Object (properties):
    - Measured from perceptual devices
    - shape, size, …
  - Action:
    - Applied physical manipulation
    - Tap, Push, Grab
  - Effects:
    - Measurable features after action
    - displacement, orientation, ...



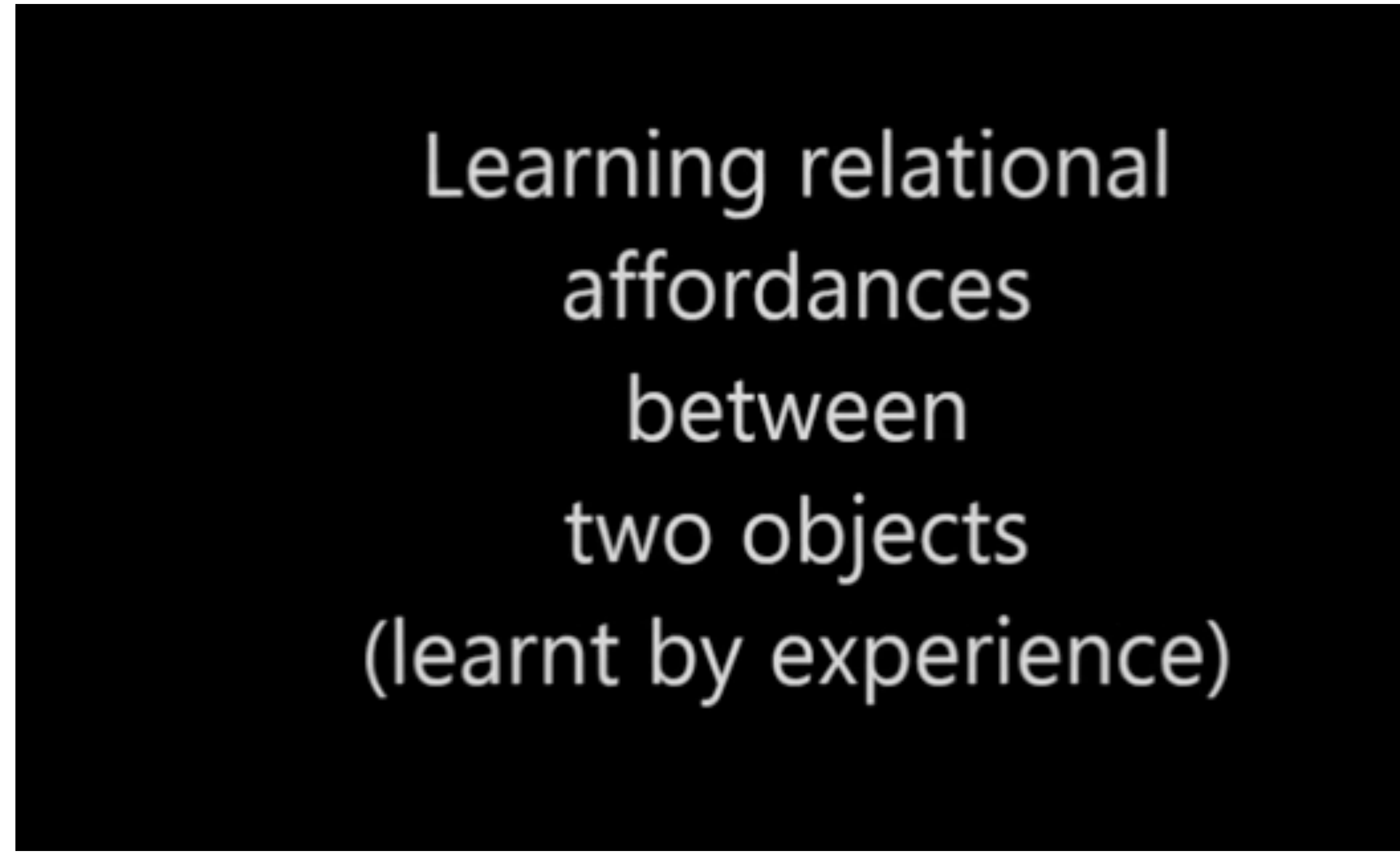
Inputs	Outputs	Function
(O, A)	E	Effect prediction
(O, E)	A	Action recognition/planning
(A, E)	O	Object recognition/selection

Learning relational affordances

Learn probabilistic model



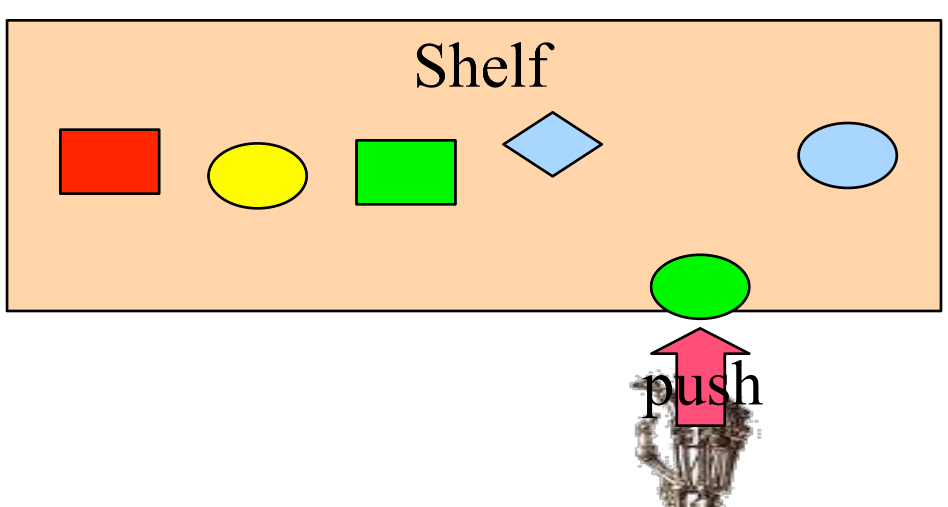
Inputs	Outputs	Function
(O, A)	E	Effect prediction
(O, E)	A	Action recognition/planning
(A, E)	O	Object recognition/selection

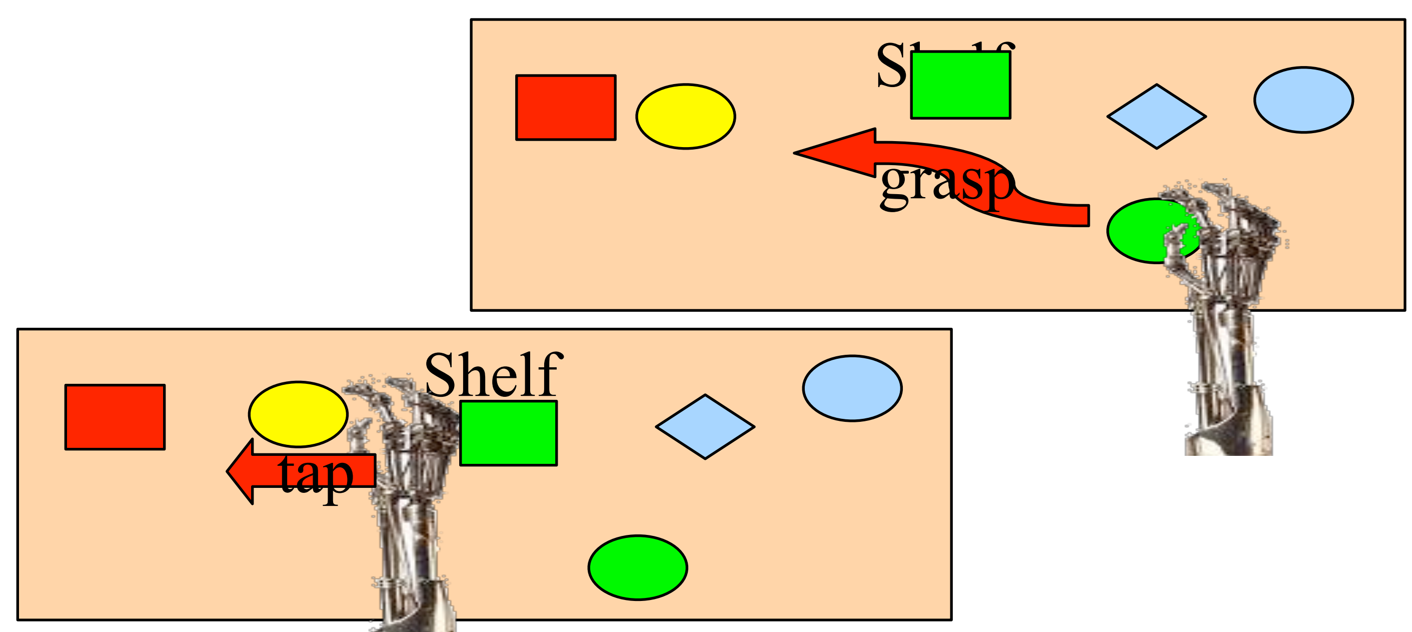


From two object interactions

Generalize to N

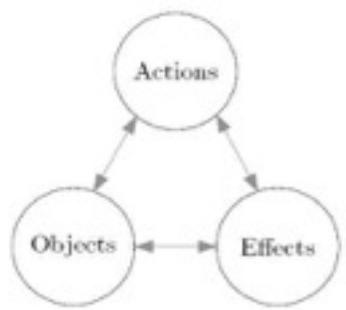
***Moldovan** et al. ICRA 12, 13, 14, PhD 15*



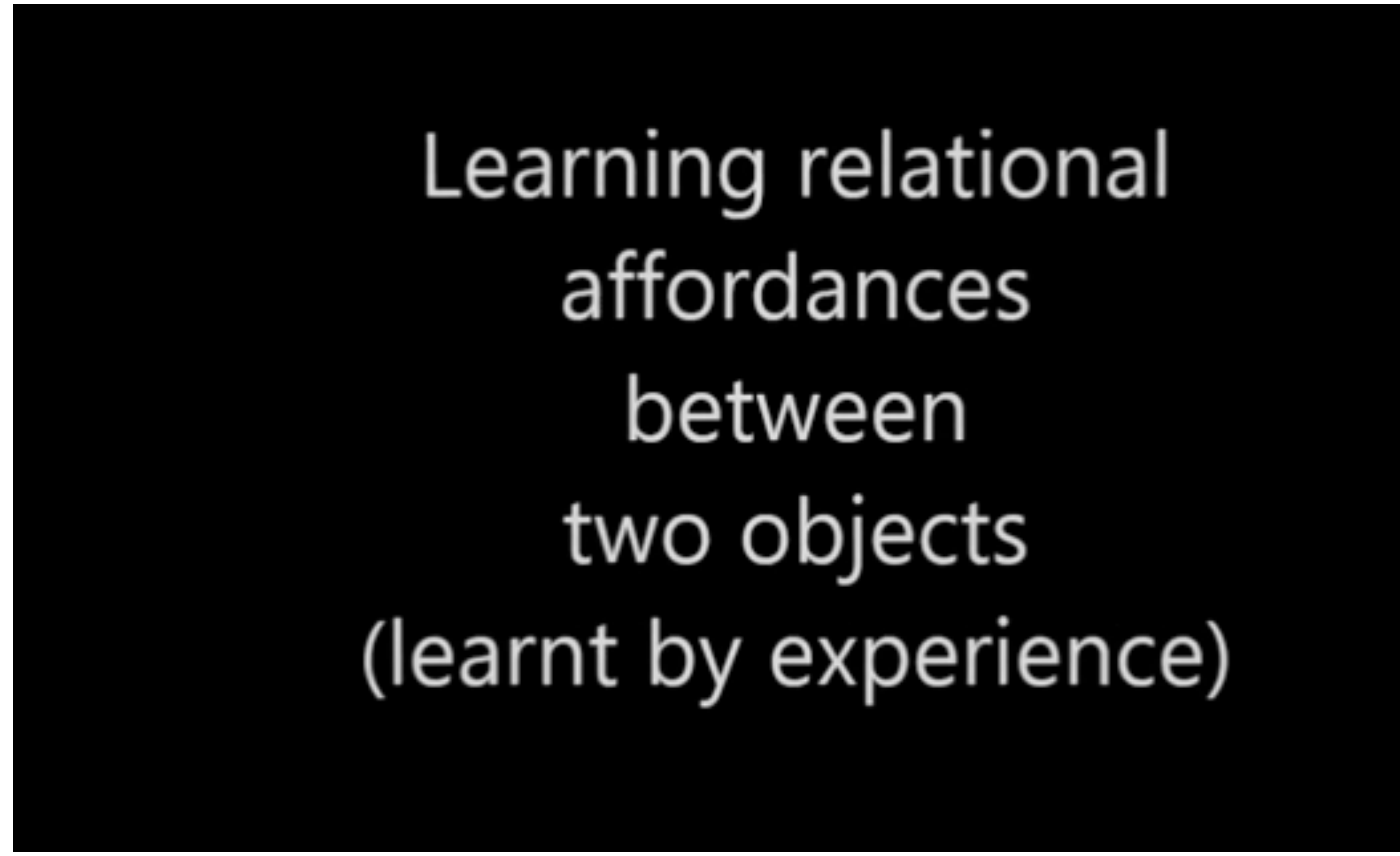


Learning relational affordances

Learn probabilistic model



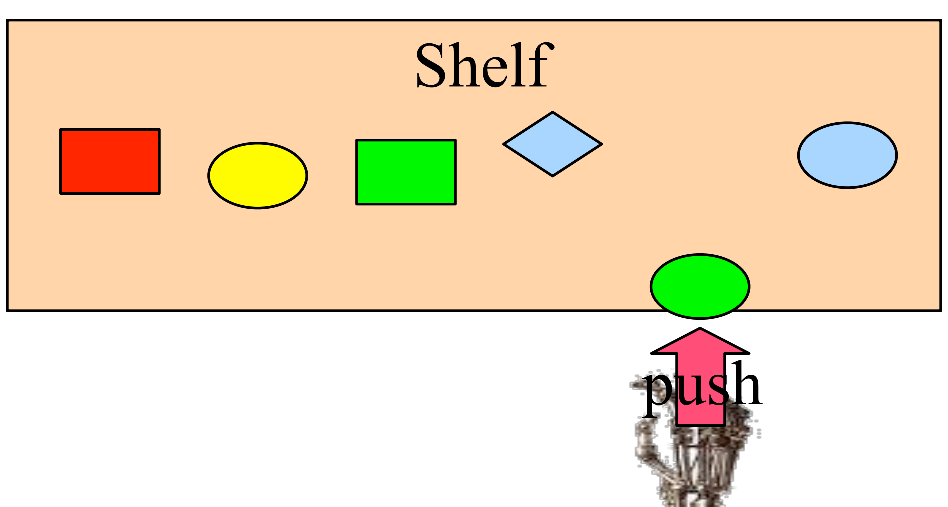
Inputs	Outputs	Function
(O, A)	E	Effect prediction
(O, E)	A	Action recognition/planning
(A, E)	O	Object recognition/selection

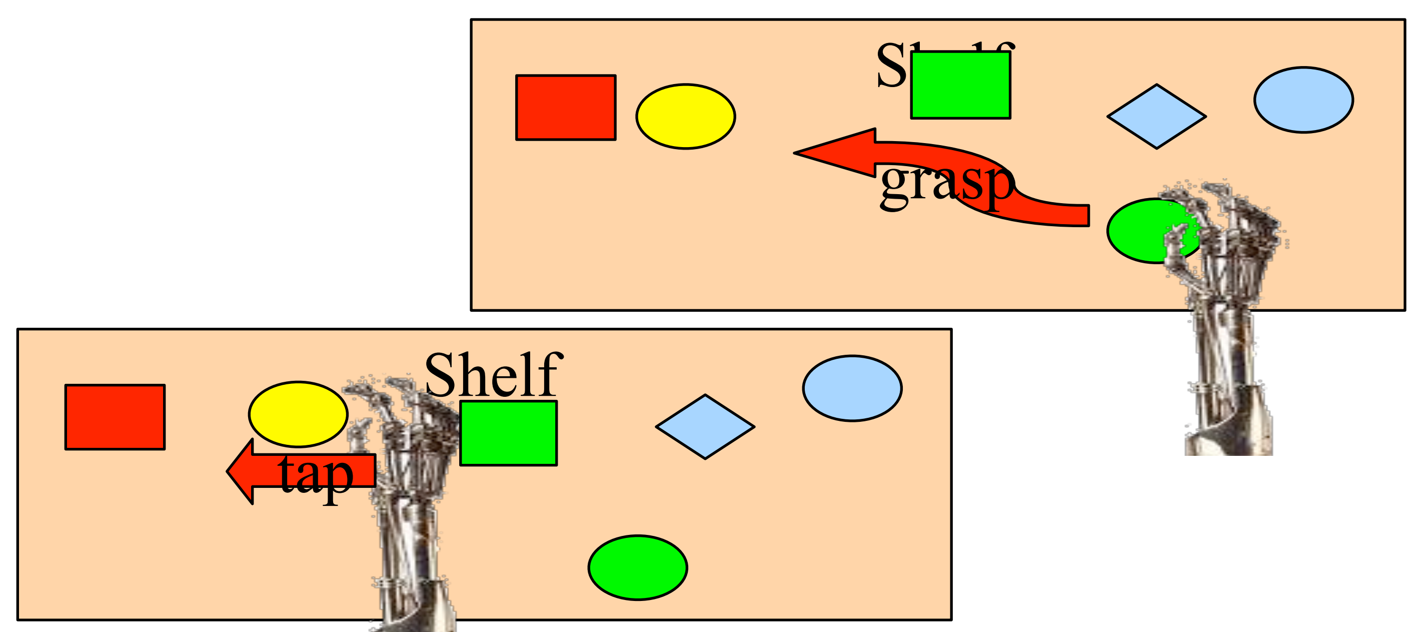


From two object interactions

Generalize to N

Moldovan et al. ICRA 12, 13, 14, PhD 15





Learning relational affordances between two objects (learnt by experience)

Right Arm

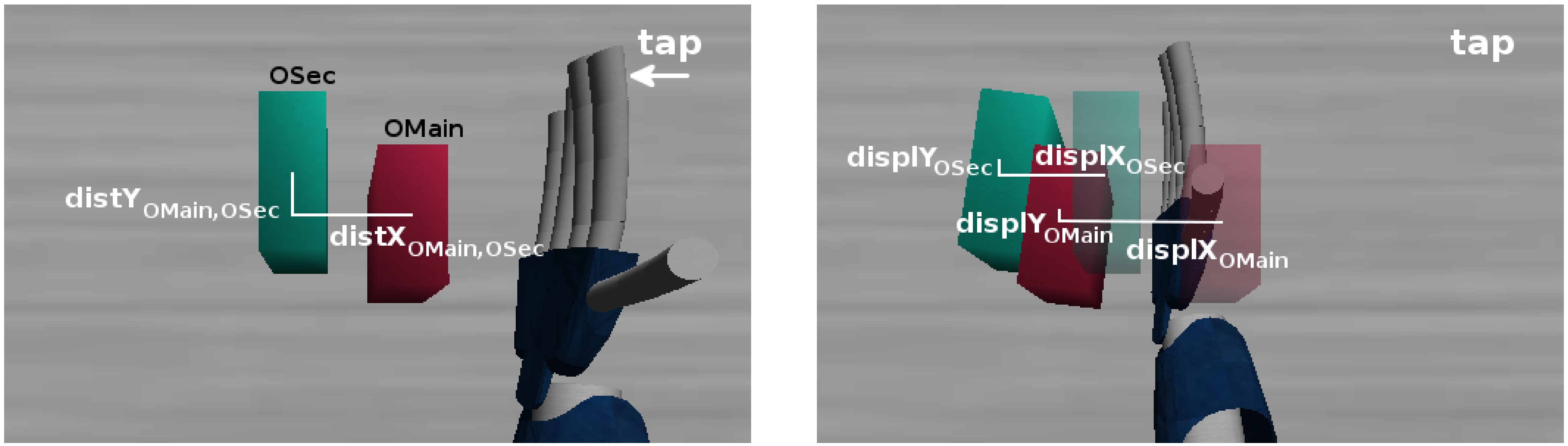
Examples

Learning relational affordances between two objects (learnt by experience)

Right Arm

Examples

What is an affordance ?



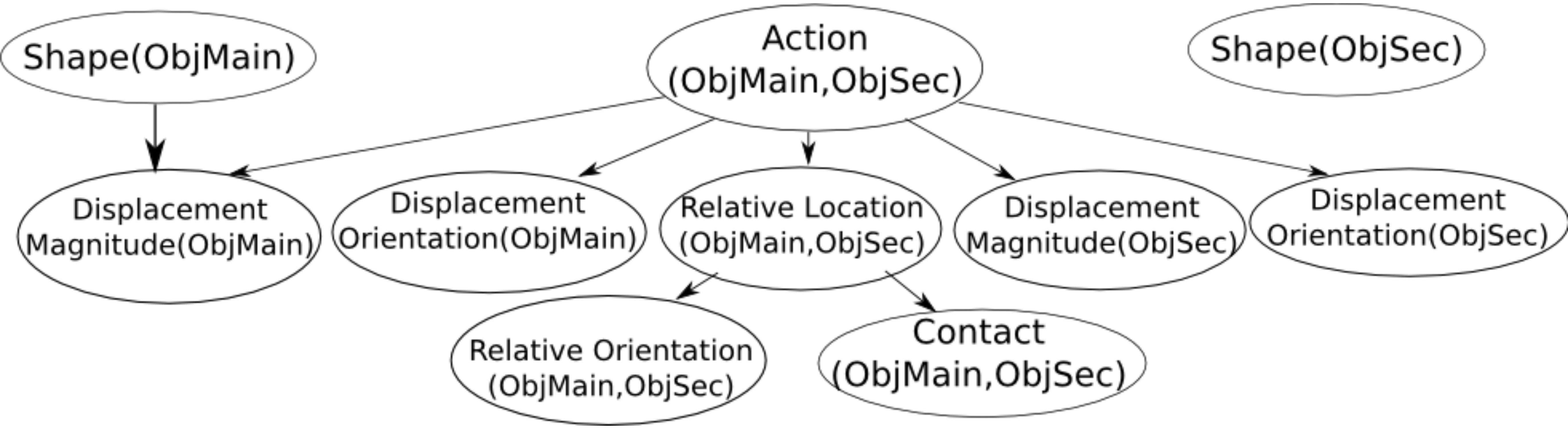
Clip 8: Relational O before (l), and E after the action execution (r).

Table 1: Example collected O, A, E data for action in Figure 8

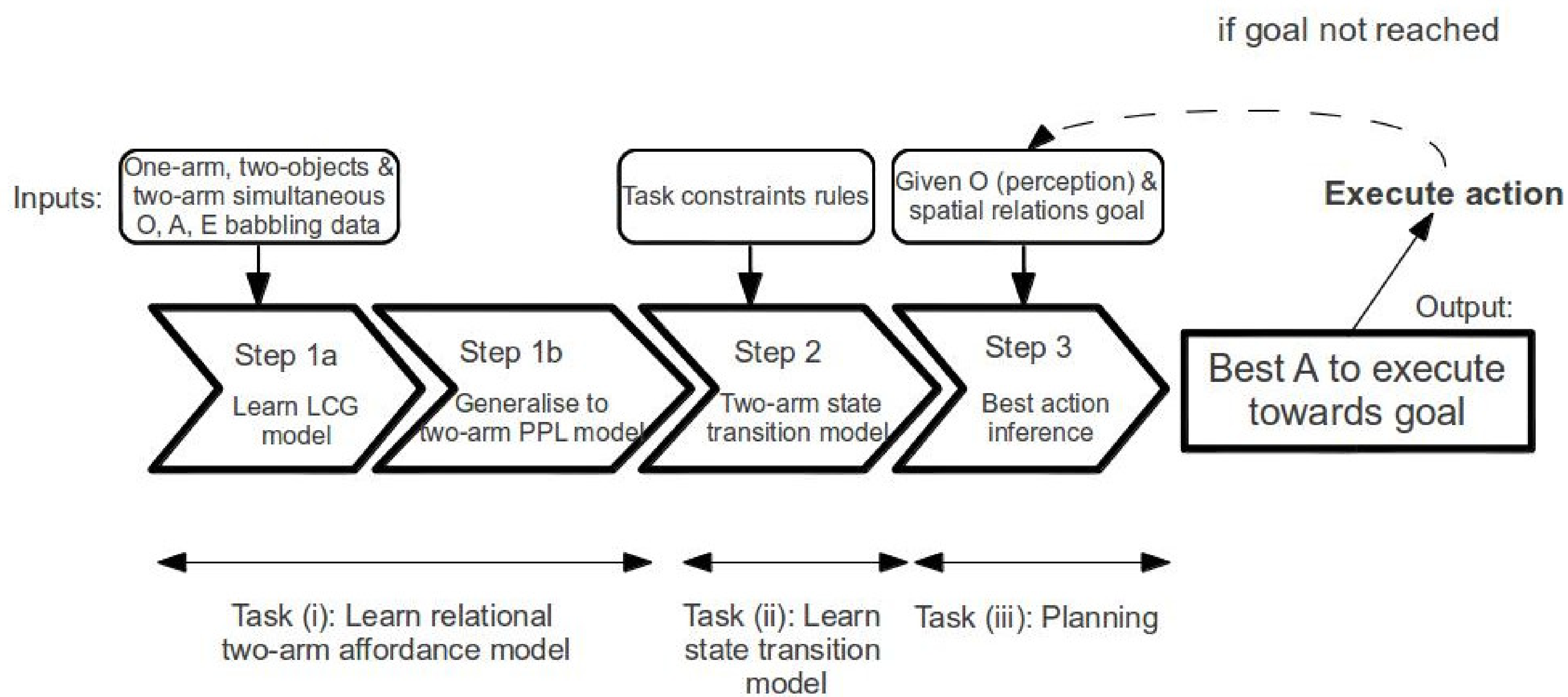
Object Properties	Action	Effects
$shape_{O_{Main}}$: $sprism$ $shape_{O_{Sec}}$: $sprism$ $distX_{O_{Main},O_{Sec}}$: $6.94cm$ $distY_{O_{Main},O_{Sec}}$: $1.90cm$	$tap(10)$	$displX_{O_{Main}}$: $10.33cm$ $displY_{O_{Main}}$: $-0.68cm$ $displX_{O_{Sec}}$: $7.43cm$ $displY_{O_{Sec}}$: $-1.31cm$

- Formalism — related to STRIPS but models delta
    - but also joint probability model over A, E, O

Bayesian Network



Learning relational affordances



Clip 4: Pipeline for table-top two-arm object manipulation.

- 1a) learn a Linear Continuous Gaussian (LCG) Bayesian Network (BN) from single arm and simultaneous two-arm exploratory data,
- 1b) from the LCG model, build the two-arm continuous domain relational affordance model in a PPL,
- 2) build a state transition model from the relational affordance model, and
- 3) infer best action to execute to reach goal (step repeated until goal reached).

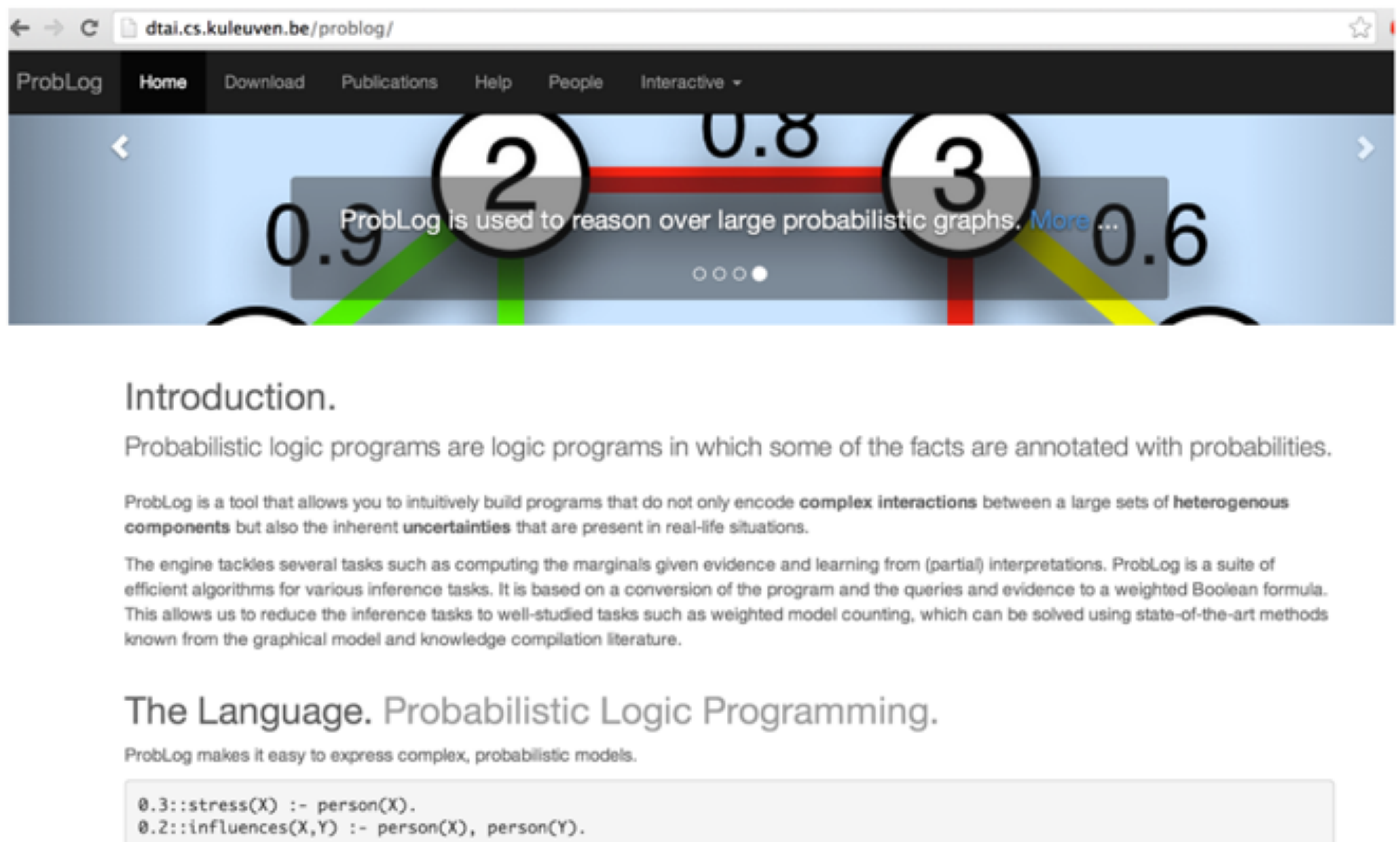
Remaining challenge

- Learn DC model directly
- Work on planning with DC (Nitti et al., ECML, EWRL 15)

Maurice Bruynooghe
Bart Demoen
Anton Dries
Daan Fierens
Jason Filippou
Bernd Gutmann
Manfred Jaeger
Gerda Janssens
Kristian Kersting
Angelika Kimmig
Theofrastos Mantadelis
Wannes Meert
Bogdan Moldovan
Siegfried Nijssen
Davide Nitti
Joris Renkens
Kate Revoredo
Ricardo Rocha
Vitor Santos Costa
Dimitar Shterionov
Ingo Thon
Hannu Toivonen
Guy Van den Broeck
Mathias Verbeke
Jonas Vlasselaer

Thanks !

http://dtai.cs.kuleuven.be/problog



PLP Systems

- **PRISM** http://sato-www.cs.titech.ac.jp/prism/
- **ProbLog2** http://dtai.cs.kuleuven.be/problog/
- Yap Prolog http://www.dcc.fc.up.pt/~vsc/Yap/ includes
  - **ProbLog1**
  - **cplint** https://sites.google.com/a/unife.it/ml/cplint
  - **CLP(BN)**
  - **LP2**
- **PITA** in XSB Prolog http://xsb.sourceforge.net/
- **AILog2** http://artint.info/code/ailog/ailog2.html
- **SLPs** http://stoics.org.uk/~nicos/sware/pepl
- **contdist** http://www.cs.sunysb.edu/~cram/contdist/
- **DC** https://code.google.com/p/distributional-clauses
- **WFOMC** http://dtai.cs.kuleuven.be/ml/systems/wfomc

Graphs & Randomness

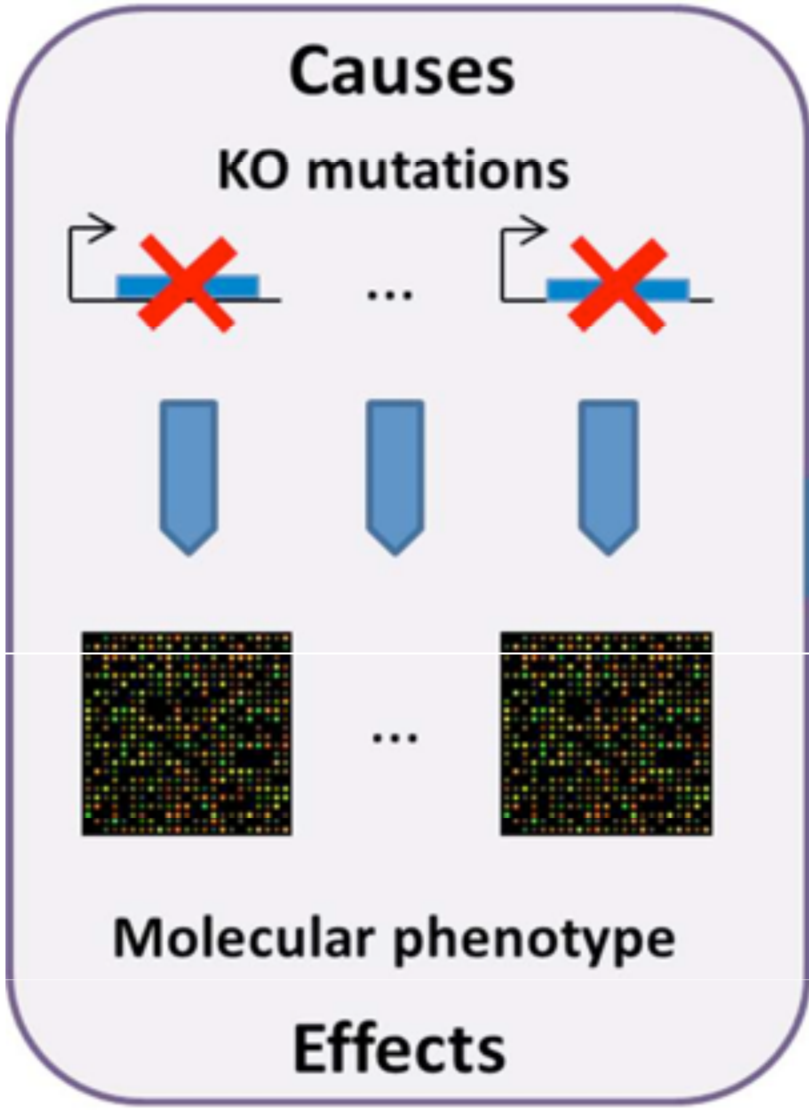
ProbLog, Phenetic, Prism, ICL, Probabilistic Databases, ...

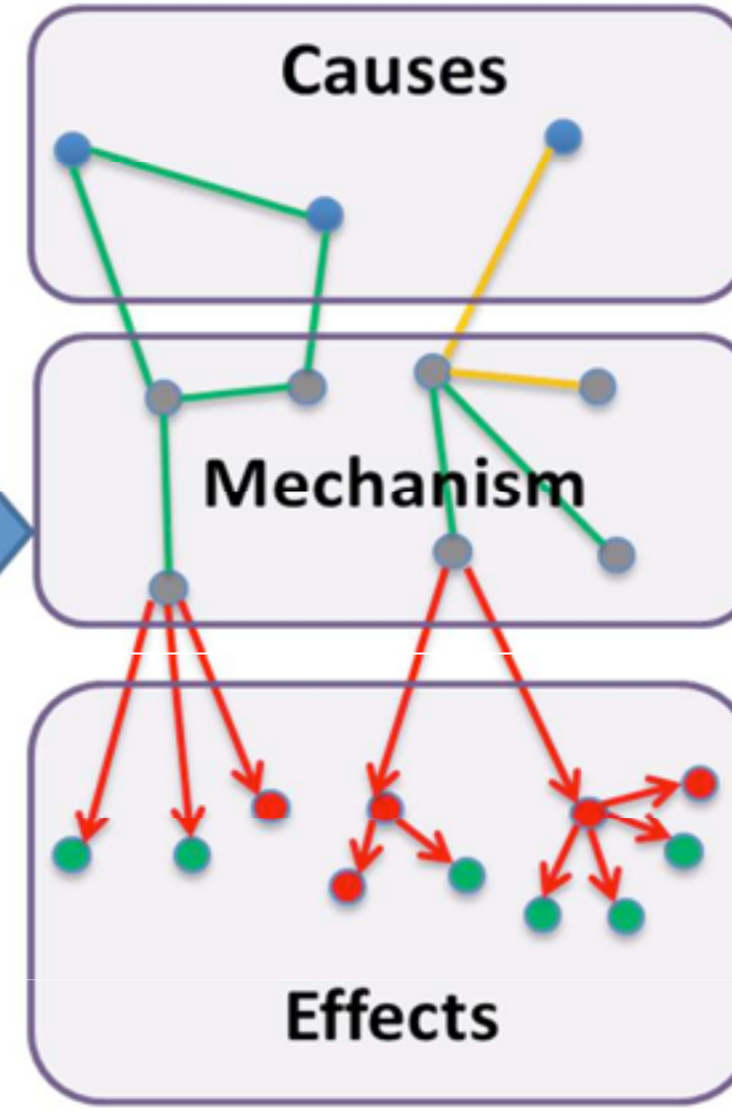
- all based on a “random graph” model

Stochastic Logic Programs, ProPPR, PCFGs, ...

- based on a “random walk” model
- connected to PageRank

Phenetic





DT-ProbLog
decision theoretic version

- Causes: Mutations
  - All related to similar phenotype
- Effects: Differentially expressed genes
- 27 000 cause effect pairs

- Interaction network:
  - 3063 nodes
    - Genes
    - Proteins
  - 16794 edges
    - Molecular interactions
    - Uncertain

- Goal: connect causes to effects through common subnetwork
  - = Find mechanism
- Techniques:
  - DTProbLog [Van den Broeck]
  - Approximate inference

Can we find the mechanism connecting causes to effects?

[De Maeyer et al., Molecular Biosystems 13, NAR 15]

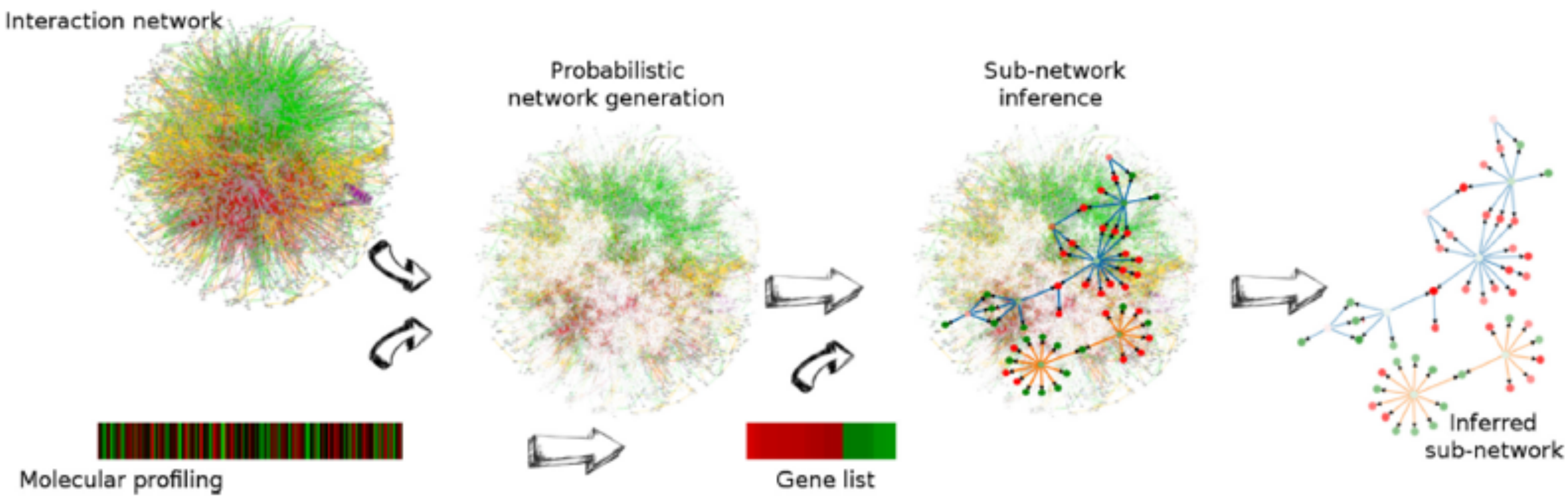
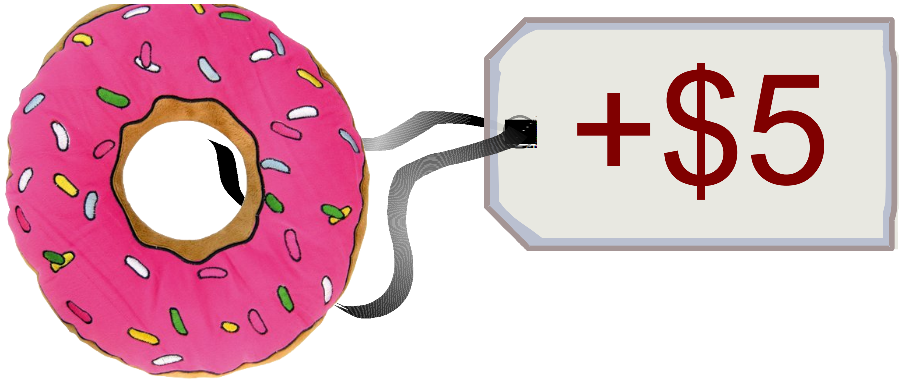


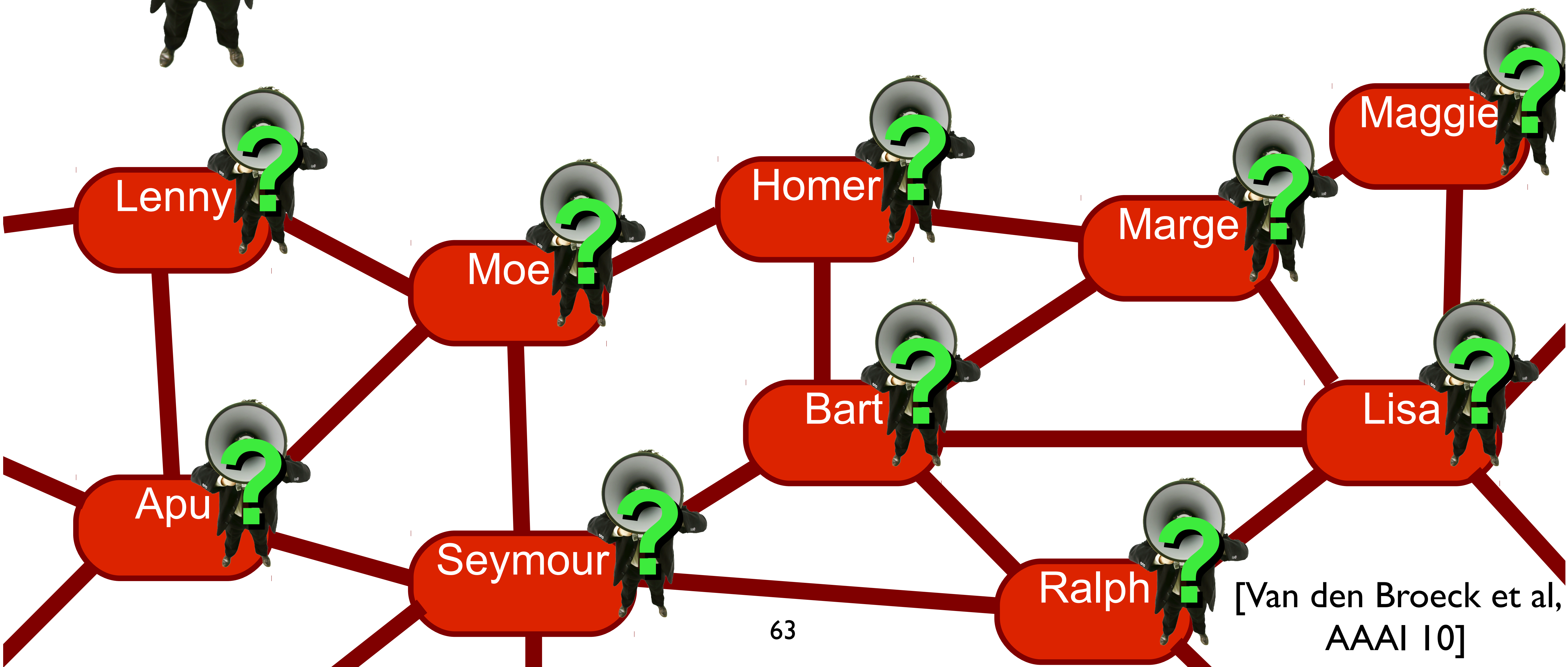
Figure 1. Overview of PheNetic, a web service for network-based interpretation of 'omics' data. The web service uses as input a genome wide interaction network for the organism of interest, a user generated molecular profiling data set and a gene list derived from these data. Interaction networks for a wide variety of organisms are readily available from the web server. Using the uploaded user-generated molecular data the interaction network is converted into a probabilistic network: edges receive a probability proportional to the levels measured for the terminal nodes in the molecular profiling data set. This probabilistic interaction network is used to infer the sub-network that best links the genes from the gene list. The inferred sub-network provides a trade-off between linking as many genes as possible from the gene list and selecting the least number of edges.

[De Mayer et al., NAR 15]

Viral Marketing

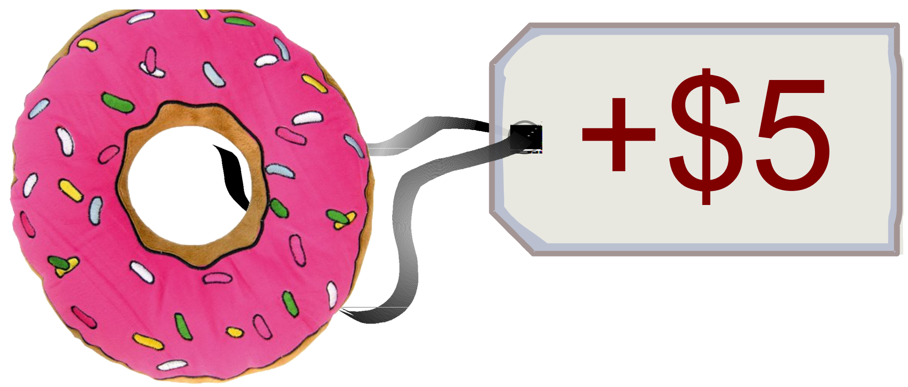


Which advertising strategy maximizes expected profit?

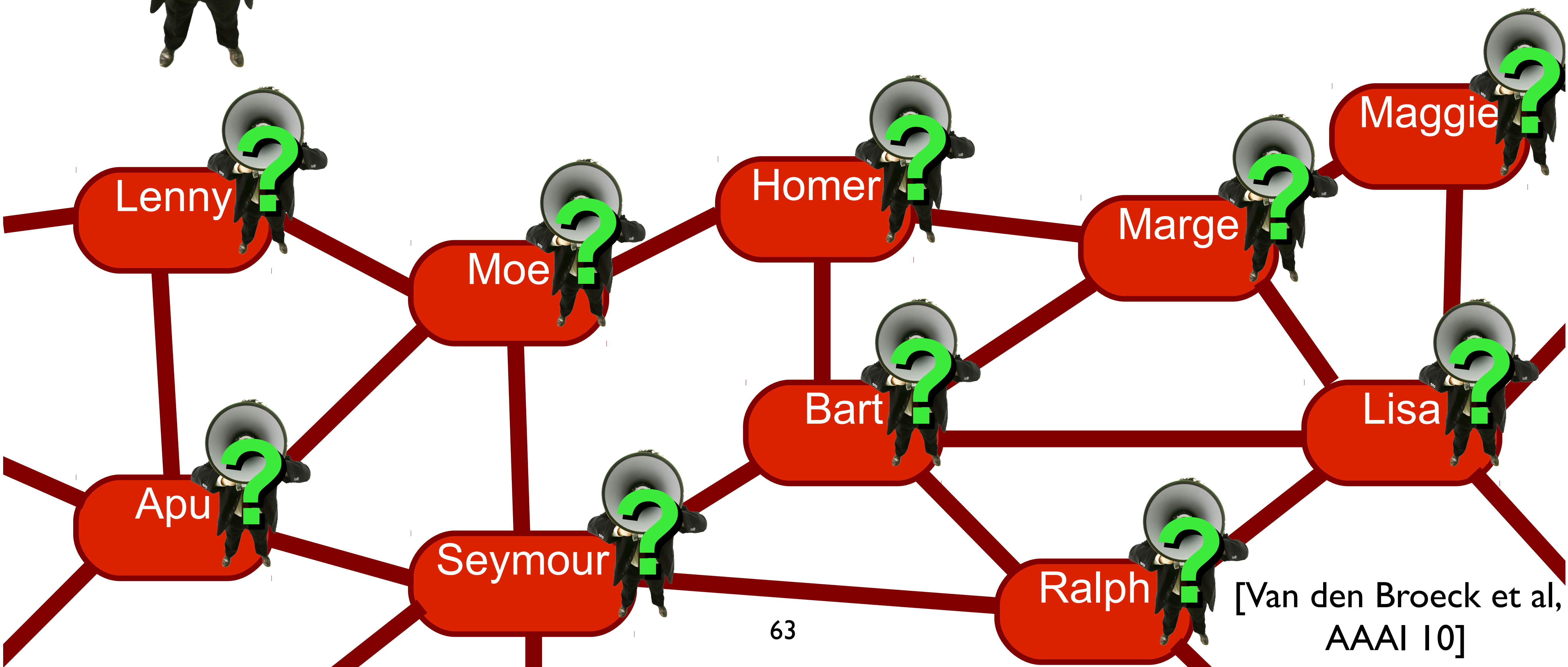


[Van den Broeck et al, AAAI 10]

Viral Marketing

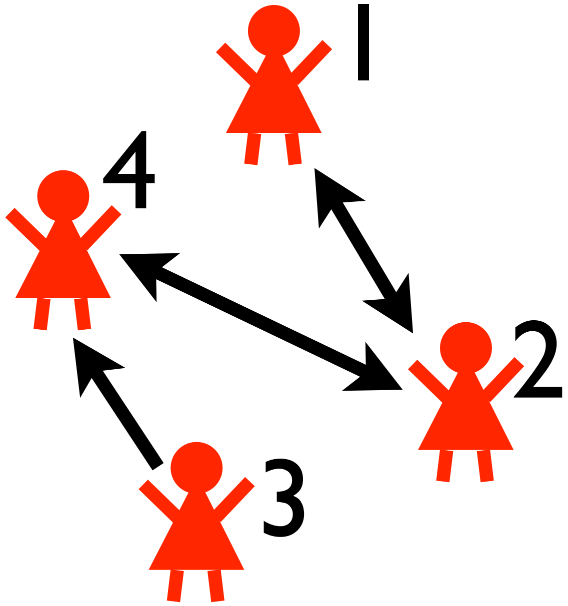


decide truth values of some atoms



[Van den Broeck et al, AAAI 10]

DTProbLog

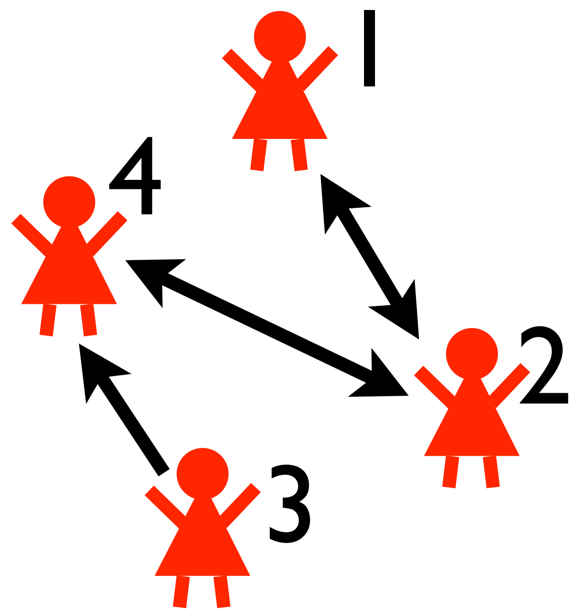



```
person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

DTProbLog

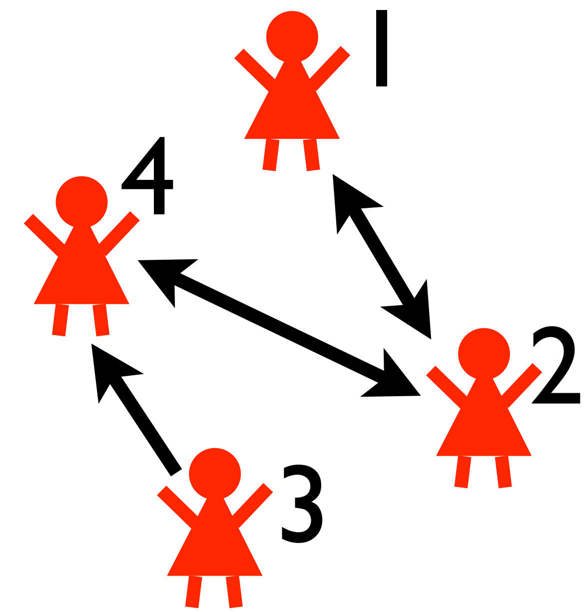
```
? :: marketed(P) :- person(P).
```

decision fact: true or false?




```
person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

DTProbLog

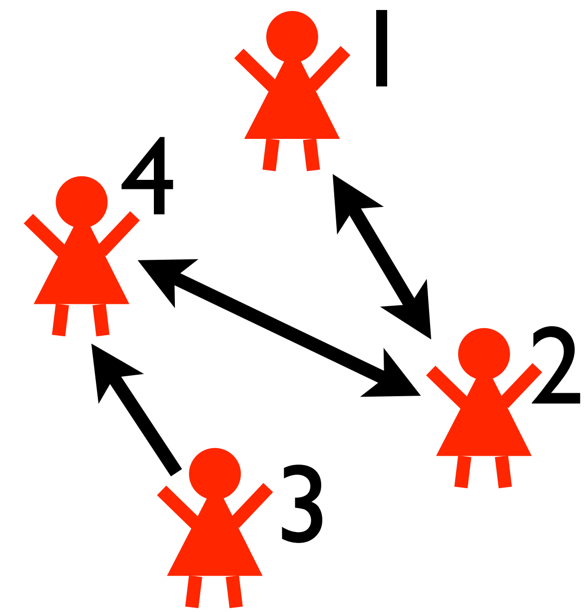



```
? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).
```

```
person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

probabilistic facts
+ logical rules

DTProbLog

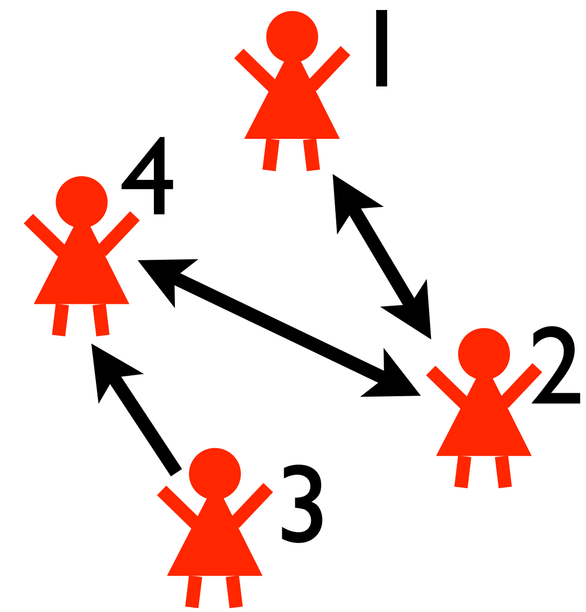



```
? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).


buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).
```

utility facts: cost/reward if true

```
person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

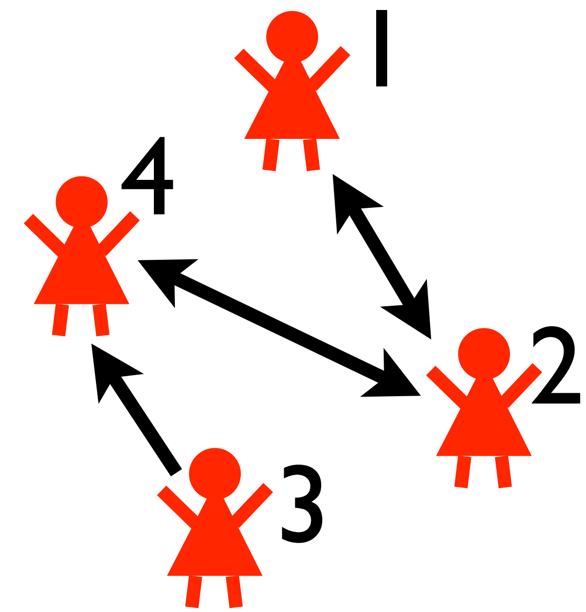
DTProbLog




```
? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).


buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).
```

```
person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

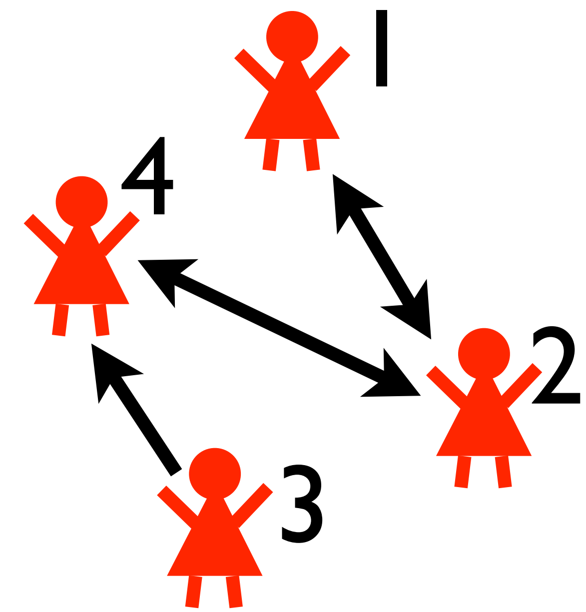
DTProbLog




```
? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).


buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).
```

```
person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

DTProbLog

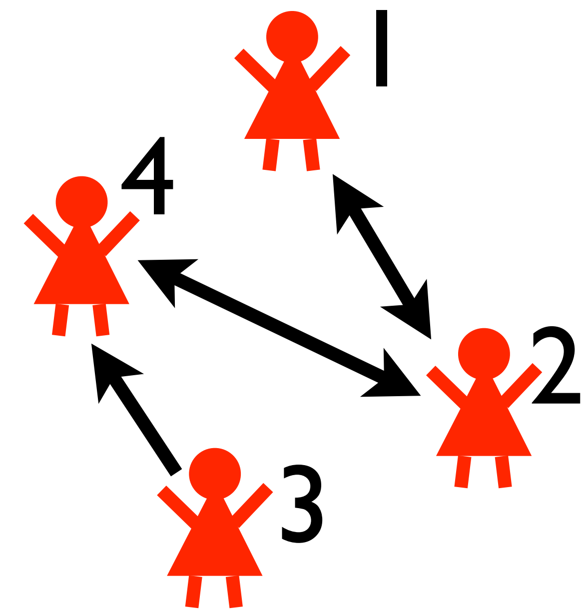



```
? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).


buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).
```

```
person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

```
marketed(1)        marketed(3)
```

DTProbLog

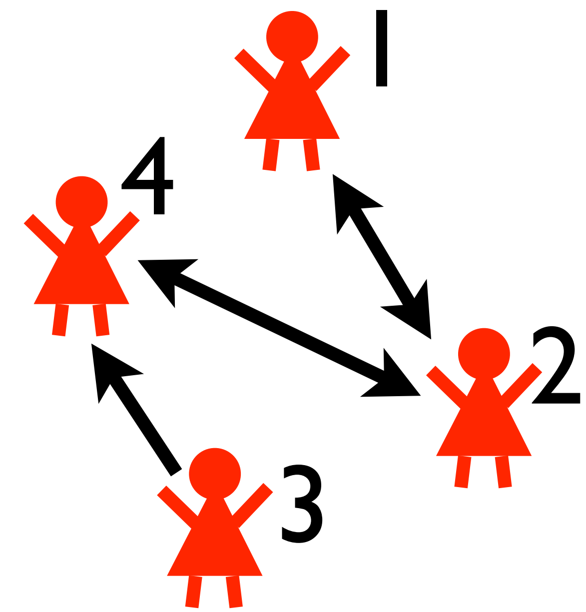



```
? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).


buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).
```

```
person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

```
marketed(1)          marketed(3)

    bt(2,1)    bt(2,4)          bm(1)
```

DTProbLog

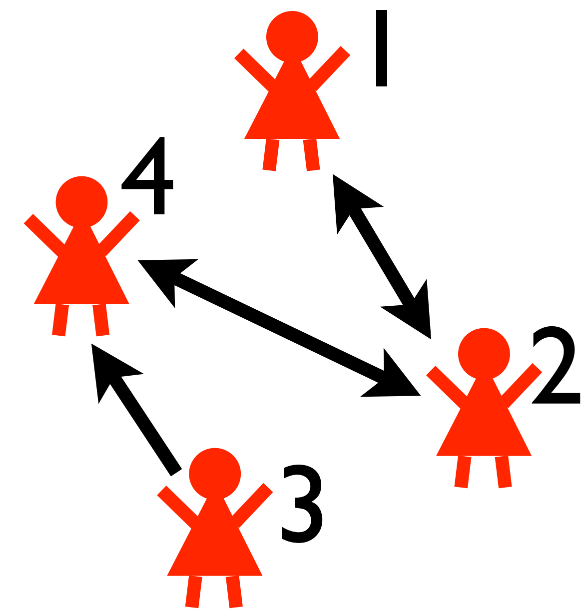



```
? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).


buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).
```

```
person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

```
marketed(1)        marketed(3)

    bt(2,1)    bt(2,4)        bm(1)

buys(1)      buys(2)
```

DTProbLog




```
? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).


buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).
```

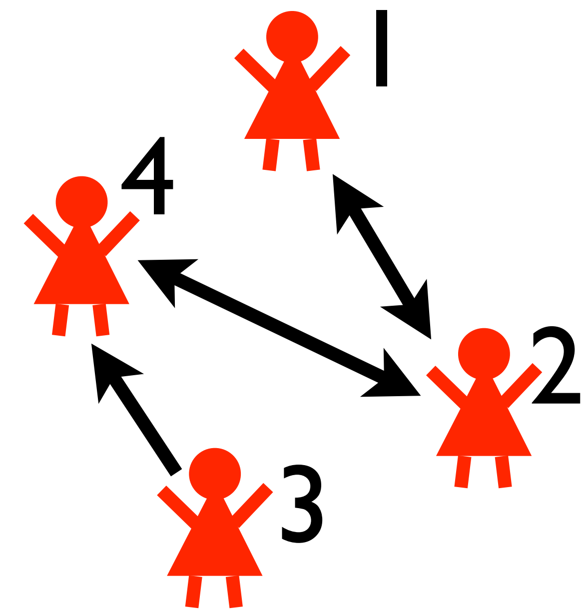
```
person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

utility = −3 + −3 + 5 + 5 = 4

probability = 0.0032

```
marketed(1)        marketed(3)

    bt(2,1)    bt(2,4)        bm(1)

buys(1)      buys(2)
```

DTProbLog




```
? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).

buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).
```

```
person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

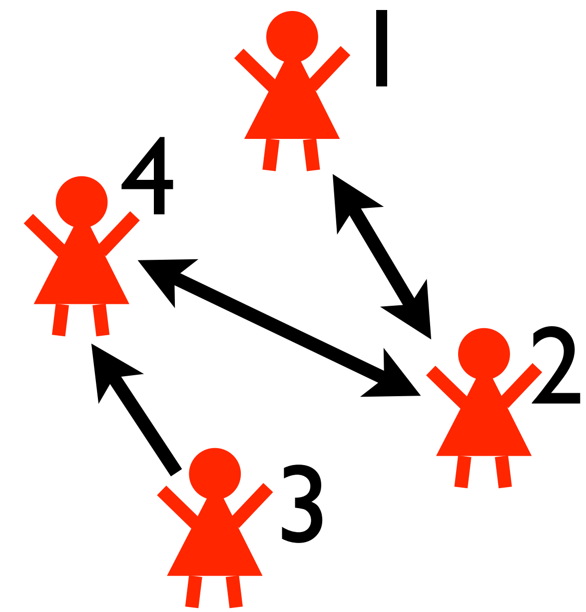
utility = −3 + −3 + 5 + 5 = 4

probability = 0.0032

```
marketed(1)        marketed(3)

    bt(2,1)    bt(2,4)          bm(1)

buys(1)     buys(2)
```

world contributes 0.0032×4 to expected utility of strategy

DTProbLog




```
? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).


buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).
```

```
person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

task: find strategy that maximizes expected utility
solution: using ProbLog technology

A true application

A tool for Computational Biology

Based on decision theoretic variation of ProbLog

ProbLog / Prob. Programming for prototyping

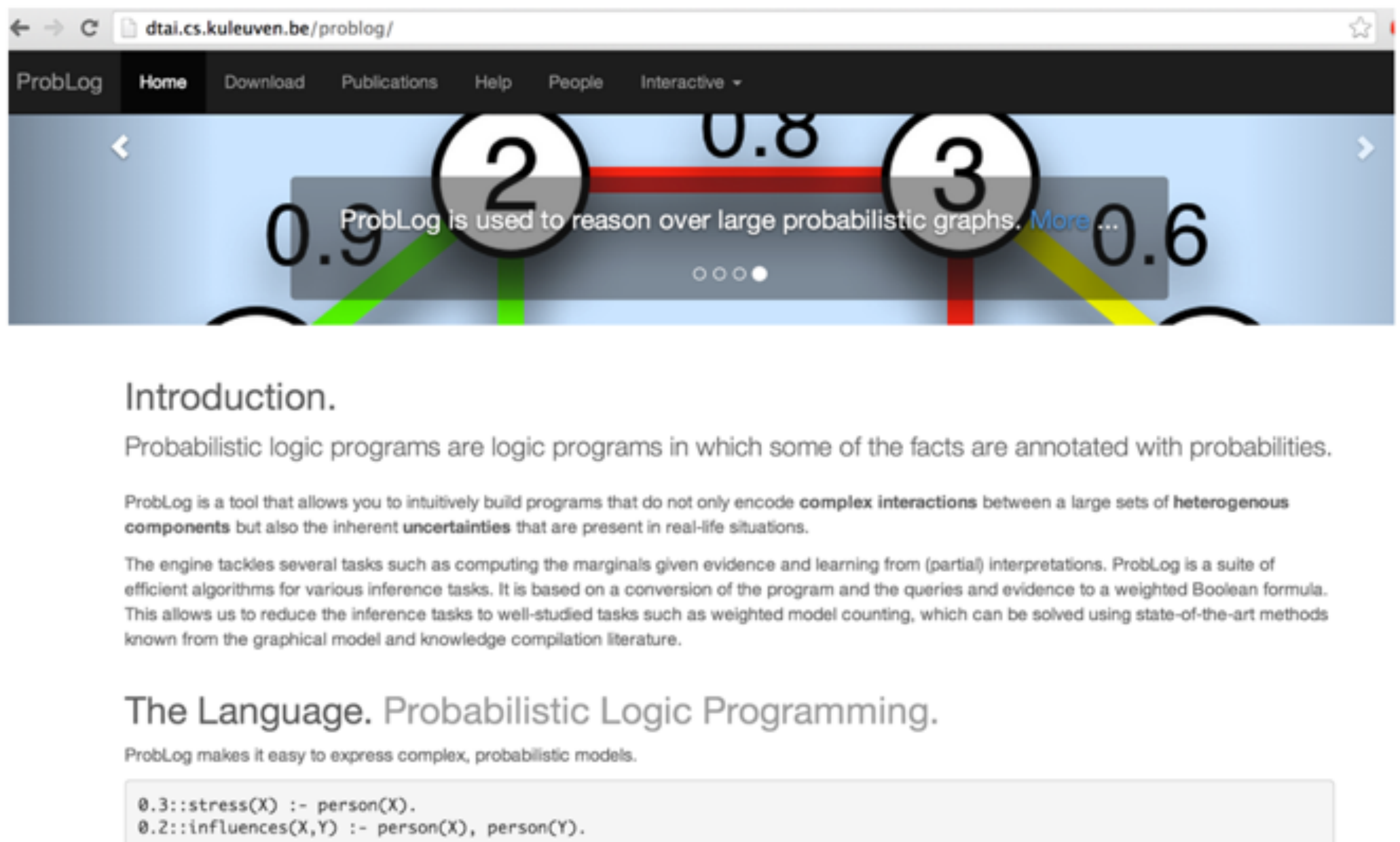
More specialised inference engine was needed

  also some special purpose approximations

Maurice Bruynooghe
Bart Demoen
Anton Dries
Daan Fierens
Jason Filippou
Bernd Gutmann
Manfred Jaeger
Gerda Janssens
Kristian Kersting
Angelika Kimmig
Theofrastos Mantadelis
Wannes Meert
Bogdan Moldovan
Siegfried Nijssen
Davide Nitti
Joris Renkens
Kate Revoredo
Ricardo Rocha
Vitor Santos Costa
Dimitar Shterionov
Ingo Thon
Hannu Toivonen
Guy Van den Broeck
Mathias Verbeke
Jonas Vlasselaer

Thanks !

http://dtai.cs.kuleuven.be/problog



PLP Systems

- **PRISM** http://sato-www.cs.titech.ac.jp/prism/
- **ProbLog2** http://dtai.cs.kuleuven.be/problog/
- Yap Prolog http://www.dcc.fc.up.pt/~vsc/Yap/ includes
  - **ProbLog1**
  - **cplint** https://sites.google.com/a/unife.it/ml/cplint
  - **CLP(BN)**
  - **LP2**
- **PITA** in XSB Prolog http://xsb.sourceforge.net/
- **AILog2** http://artint.info/code/ailog/ailog2.html
- **SLPs** http://stoics.org.uk/~nicos/sware/pepl
- **contdist** http://www.cs.sunysb.edu/~cram/contdist/
- **DC** https://code.google.com/p/distributional-clauses
- **WFOMC** http://dtai.cs.kuleuven.be/ml/systems/wfomc

References

Bach SH, Broecheler M, Getoor L, O'Leary DP (2012) Scaling MPE inference for constrained continuous Markov random fields with consensus optimization. In: Proceedings of the 26th Annual Conference on Neural Information Processing Systems (NIPS-12)

Broecheler M, Mihalkova L, Getoor L (2010) Probabilistic similarity logic. In: Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence (UAI-10)

Bryant RE (1986) Graph-based algorithms for Boolean function manipulation. IEEE Transactions on Computers 35(8):677–691

Cohen SB, Simmons RJ, Smith NA (2008) Dynamic programming algorithms as products of weighted logic programs. In: Proceedings of the 24th International Conference on Logic Programming (ICLP-08)

Cussens J (2001) Parameter estimation in stochastic logic programs. Machine Learning 44(3):245–271

De Maeyer D, Renkens J, Cloots L, De Raedt L, Marchal K (2013) Phenetic: network-based interpretation of unstructured gene lists in e. coli. Molecular BioSystems 9(7):1594–1603

De Raedt L, Kimmig A (2013) Probabilistic programming concepts. CoRR abs/1312.4328

De Raedt L, Kimmig A, Toivonen H (2007) ProbLog: A probabilistic Prolog and its application in link discovery. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)

De Raedt L, Frasconi P, Kersting K, Muggleton S (eds) (2008) Probabilistic Inductive Logic Programming — Theory and Applications, Lecture Notes in Artificial Intelligence, vol 4911. Springer

Eisner J, Goldlust E, Smith N (2005) Compiling Comp Ling: Weighted dynamic programming and the Dyna language. In: Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP-05)

Fierens D, Blockeel H, Bruynooghe M, Ramon J (2005) Logical Bayesian networks and their relation to other probabilistic logical models. In: Proceedings of the 15th International Conference on Inductive Logic Programming (ILP-05)

Fierens D, Van den Broeck G, Bruynooghe M, De Raedt L (2012) Constraints for probabilistic logic programming. In: Proceedings of the NIPS Probabilistic Programming Workshop

Fierens D, Van den Broeck G, Renkens J, Shterionov D, Gutmann B, Thon I, Janssens G, De Raedt L (2014) Inference and learning in probabilistic logic programs using weighted Boolean formulas. Theory and Practice of Logic Programming (TPLP) FirstView

Getoor L, Friedman N, Koller D, Pfeffer A, Taskar B (2007) Probabilistic relational models. In: Getoor L, Taskar B (eds) An Introduction to Statistical Relational Learning, MIT Press, pp 129–174

Goodman N, Mansinghka VK, Roy DM, Bonawitz K, Tenenbaum JB (2008) Church: a language for generative models. In: Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence (UAI-08)

Gutmann B, Thon I, De Raedt L (2011a) Learning the parameters of probabilistic logic programs from interpretations. In: Proceedings of the 22nd European Conference on Machine Learning (ECML-11)

Gutmann B, Thon I, Kimmig A, Bruynooghe M, De Raedt L (2011b) The magic of logical inference in probabilistic programming. Theory and Practice of Logic Programming (TPLP) 11((4–5)):663–680

Huang B, Kimmig A, Getoor L, Golbeck J (2013) A flexible framework for probabilistic models of social trust. In: Proceedings of the International Conference on Social Computing, Behavioral-Cultural Modeling, and Prediction (SBP-13)

Jaeger M (2002) Relational Bayesian networks: A survey. Linköping Electronic Articles in Computer and Information Science 7(015)

Kersting K, Raedt LD (2001) Bayesian logic programs. CoRR cs.AI/0111058

Kimmig A, Van den Broeck G, De Raedt L (2011a) An algebraic Prolog for reasoning about possible worlds. In: Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI-11)

Kimmig A, Demoen B, De Raedt L, Santos Costa V, Rocha R (2011b) On the implementation of the probabilistic logic programming language ProbLog. Theory and Practice of Logic Programming (TPLP) 11:235–262

Koller D, Pfeffer A (1998) Probabilistic frame-based systems. In: Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98)

McCallum A, Schultz K, Singh S (2009) FACTORIE: Probabilistic programming via imperatively defined factor graphs. In: Proceedings of the 23rd Annual Conference on Neural Information Processing Systems (NIPS-09)

Milch B, Marthi B, Russell SJ, Sontag D, Ong DL, Kolobov A (2005) Blog: Probabilistic models with unknown objects. In: Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI-05)

Moldovan B, De Raedt L (2014) Occluded object search by relational affordances. In: IEEE International Conference on Robotics and Automation (ICRA-14)

Moldovan B, Moreno P, van Otterlo M, Santos-Victor J, De Raedt L (2012) Learning relational affordance models for robots in multi-object manipulation tasks. In: IEEE International Conference on Robotics and Automation (ICRA-12)

Muggleton S (1996) Stochastic logic programs. In: De Raedt L (ed) Advances in Inductive Logic Programming, IOS Press, pp 254–264

Nitti D, De Laet T, De Raedt L (2013) A particle filter for hybrid relational domains. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-13)

Nitti D, De Laet T, De Raedt L (2014) Relational object tracking and learning. In: IEEE International Conference on Robotics and Automation (ICRA), June 2014

Pfeffer A (2001) IBAL: A probabilistic rational programming language. In: Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01)

Pfeffer A (2009) Figaro: An object-oriented probabilistic programming language. Tech. rep., Charles River Analytics

Poole D (2003) First-order probabilistic inference. In: Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03)

Richardson M, Domingos P (2006) Markov logic networks. Machine Learning 62(1-2):107–136

Santos Costa V, Page D, Cussens J (2008) CLP(BN): Constraint logic programming for probabilistic knowledge. In: De Raedt et al (2008), pp 156–188

Sato T (1995) A statistical learning method for logic programs with distribution semantics. In: Proceedings of the 12th International Conference on Logic Programming (ICLP-95)

Sato T, Kameya Y (2001) Parameter learning of logic programs for symbolic-statistical modeling. J Artif Intell Res (JAIR) 15:391–454

Sato T, Kameya Y (2008) New advances in logic-based probabilistic modeling by prism. In: Probabilistic Inductive Logic Programming, pp 118–155

Skarlatidis A, Artikis A, Filiopou J, Paliouras G (2014) A probabilistic logic programming event calculus. Theory and Practice of Logic Programming (TPLP) FirstView

Suciu D, Olteanu D, Ré C, Koch C (2011) Probabilistic Databases. Synthesis Lectures on Data Management, Morgan & Claypool Publishers

Taskar B, Abbeel P, Koller D (2002) Discriminative probabilistic models for relational data. In: Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence (UAI-02)

Thon I, Landwehr N, De Raedt L (2008) A simple model for sequences of relational state descriptions. In: Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD-08)

Thon I, Landwehr N, De Raedt L (2011) Stochastic relational processes: Efficient inference and applications. Machine Learning 82(2):239–272

Van den Broeck G, Thon I, van Otterlo M, De Raedt L (2010) DTProbLog: A decision-theoretic probabilistic Prolog. In: Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI-10)

Van den Broeck G, Taghipour N, Meert W, Davis J, De Raedt L (2011) Lifted probabilistic inference by first-order knowledge compilation. In: Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI-11)

Vennekens J, Verbaeten S, Bruynooghe M (2004) Logic programs with annotated disjunctions. In: Proceedings of the 20th International Conference on Logic Programming (ICLP-04)

Vennekens J, Denecker M, Bruynooghe M (2009) CP-logic: A language of causal probabilistic events and its relation to logic programming. Theory and Practice of Logic Programming (TPLP) 9(3):245–308

Wang WY, Mazaitis K, Cohen WW (2013) Programming with personalized pagerank: a locally groundable first-order probabilistic logic. In: Proceedings of the 22nd ACM International Conference on Information and Knowledge Management (CIKM-13)