Probabilistic (Logic) Programming and its Applications

Luc De Raedt

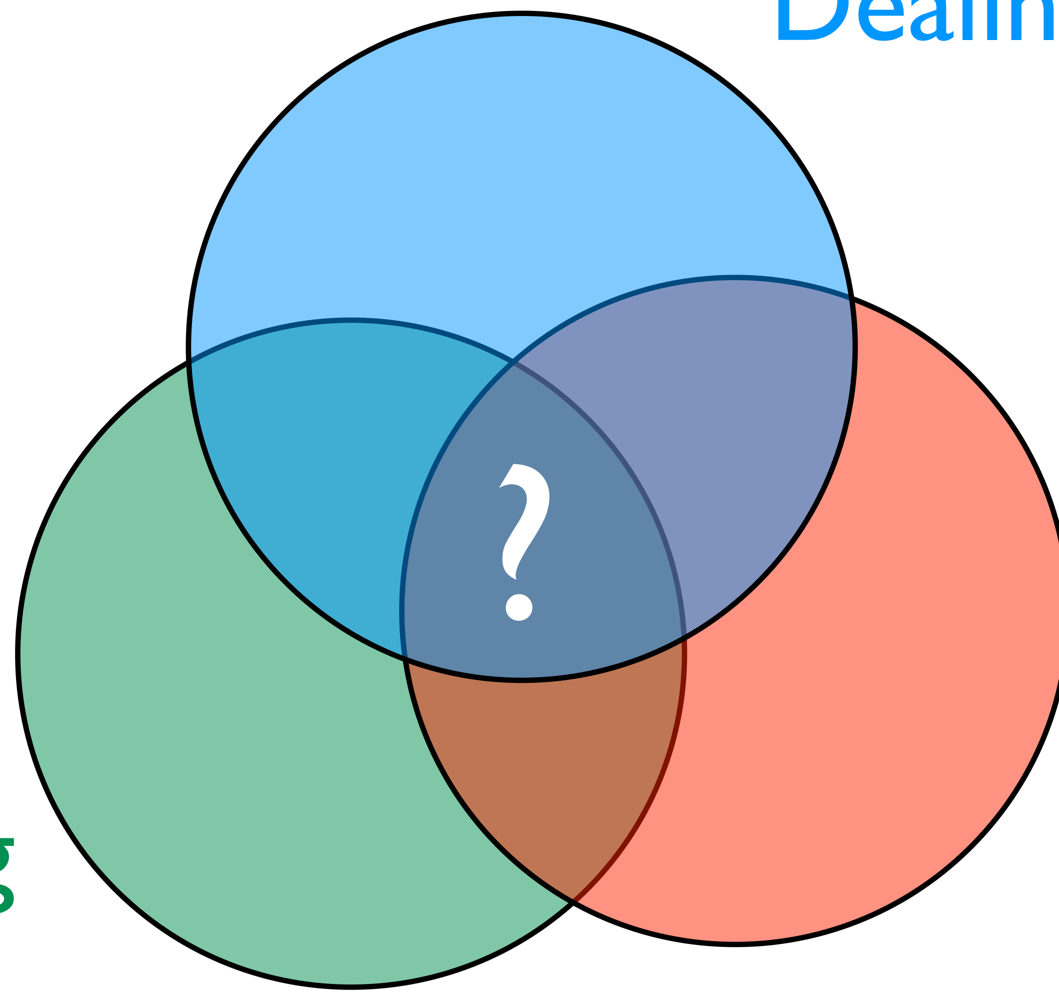
with many slides from Angelika Kimmmig

MACHINE LEARNING

A key question in AI:

Dealing with uncertainty
- probability theory
- graphical models
- ...

Reasoning with relational data
- logic
- databases
- programming
- ...

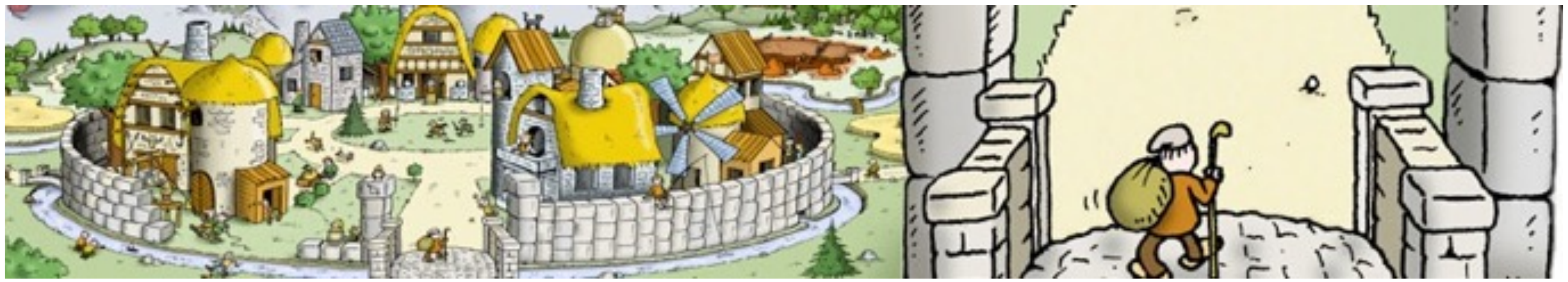


Learning
- parameters
- structure

Statistical relational learning
& Probabilistic Programming

The need for relations

Dynamics: Evolving Networks



- *Travian*: A massively multiplayer real-time strategy game
  - Commercial game run by TravianGames GmbH
  - ~3.000.000 players spread over different "worlds"
  - ~25.000 players in one world

[Thon et al. ECML 08]

World Dynamics

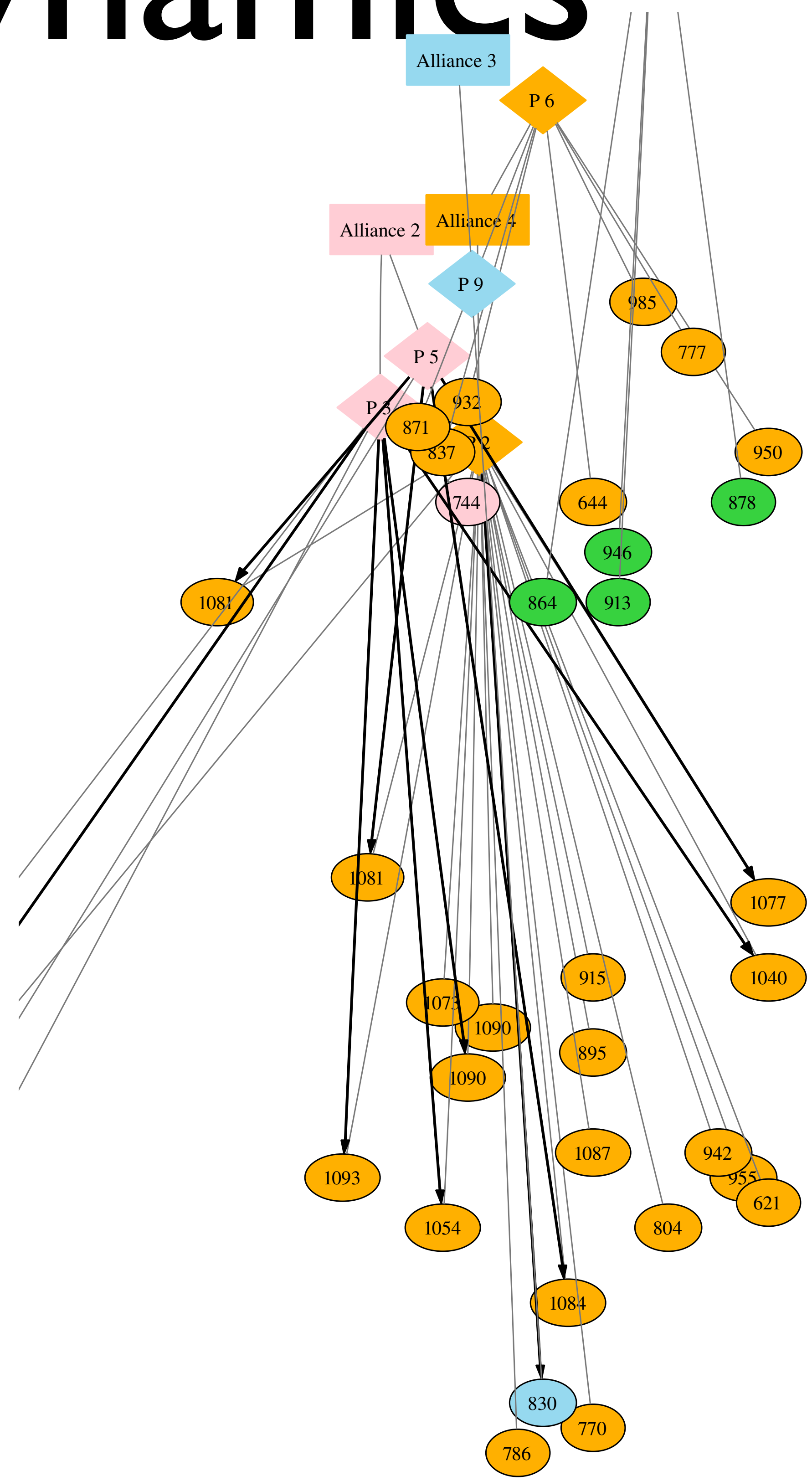
Fragment of world with

~10 alliances
~200 players
~600 cities

alliances color-coded

Can we build a model
of this world ?
Can we use it for playing
better ?

[Thon, Landwehr, De Raedt, ECML08]



World Dynamics

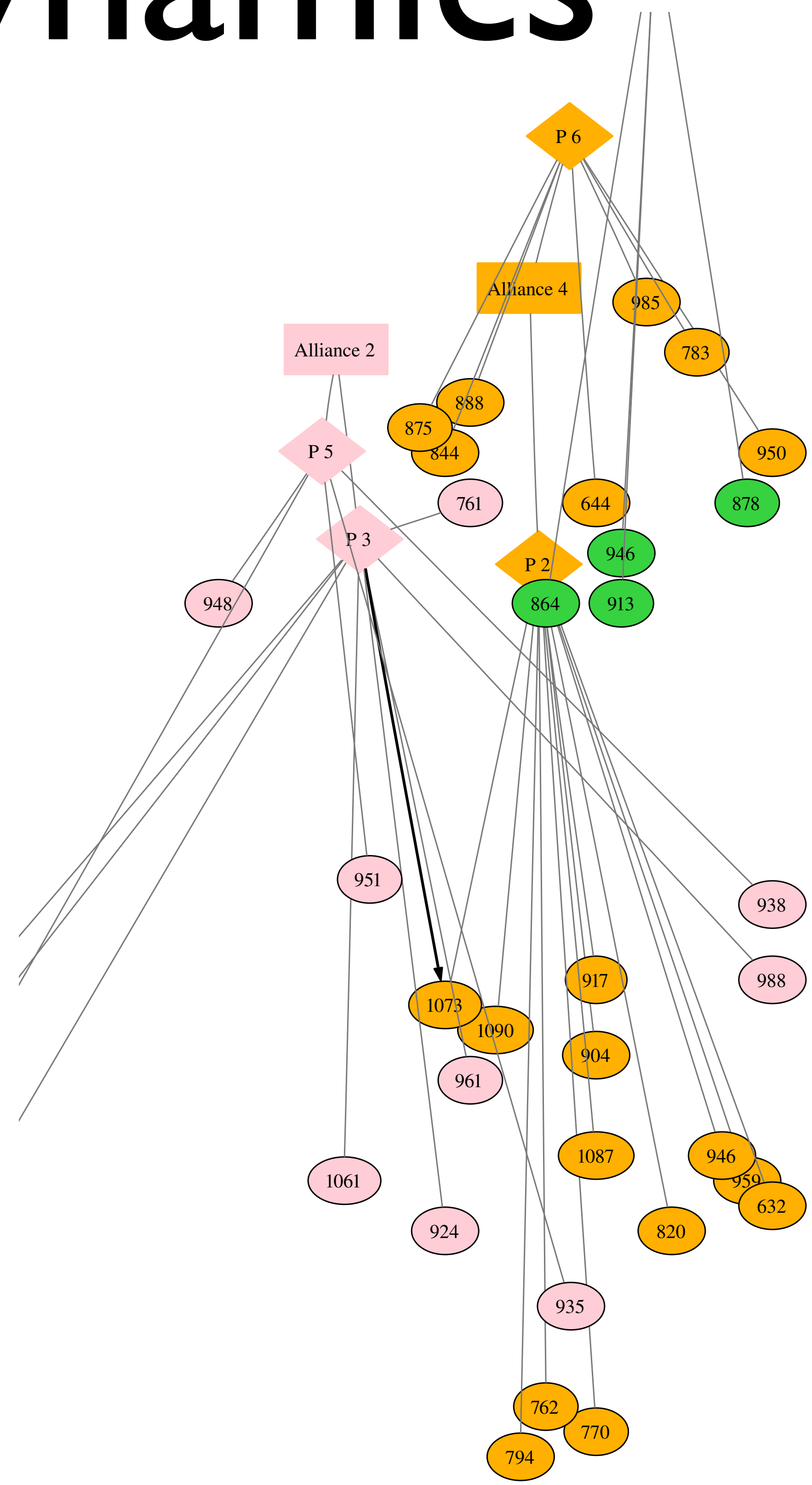
Fragment of world with

~10 alliances
~200 players
~600 cities

alliances color-coded

Can we build a model
of this world ?
Can we use it for playing
better ?

[Thon, Landwehr, De Raedt, ECML08]



World Dynamics

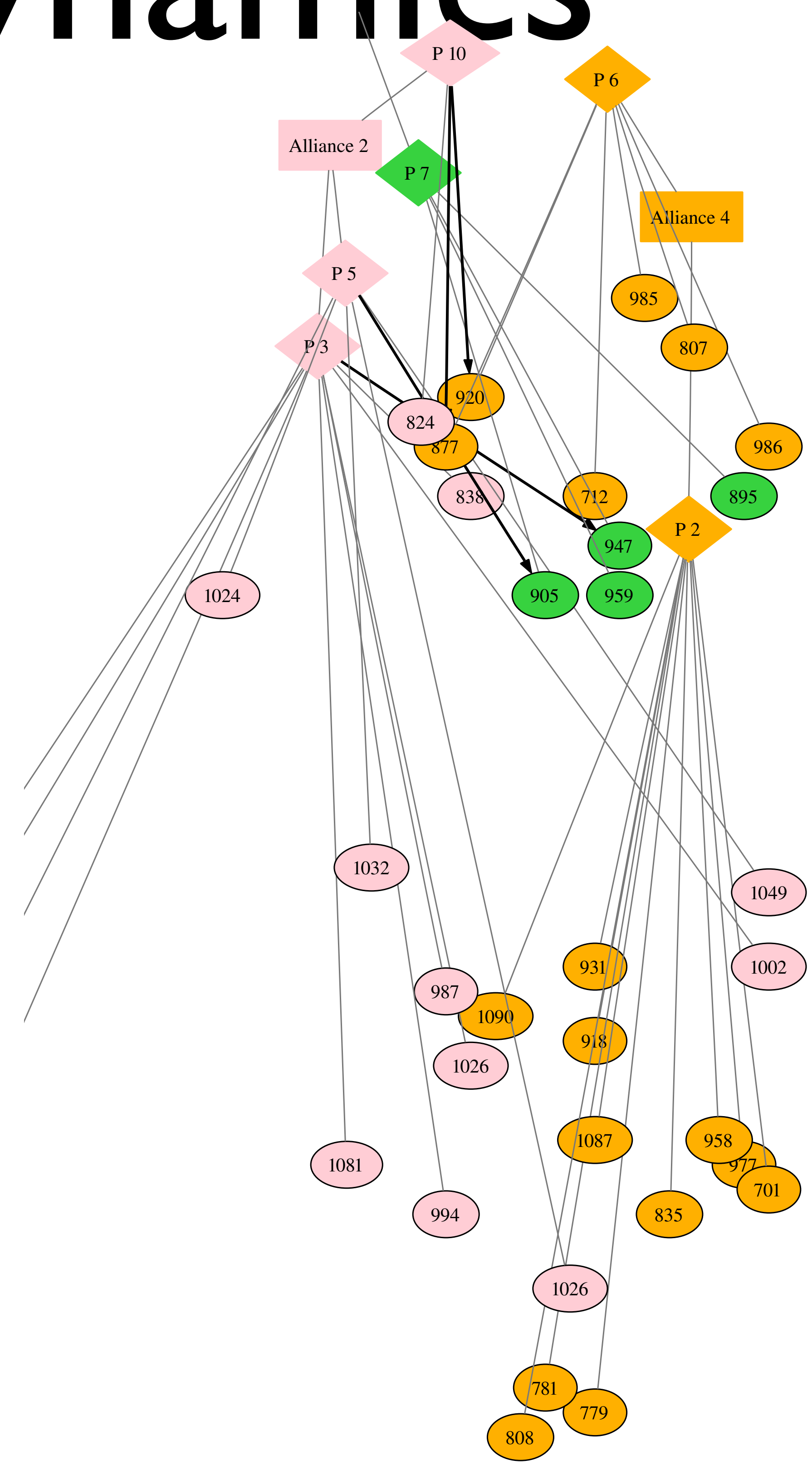
Fragment of world with

~10 alliances
~200 players
~600 cities

alliances color-coded

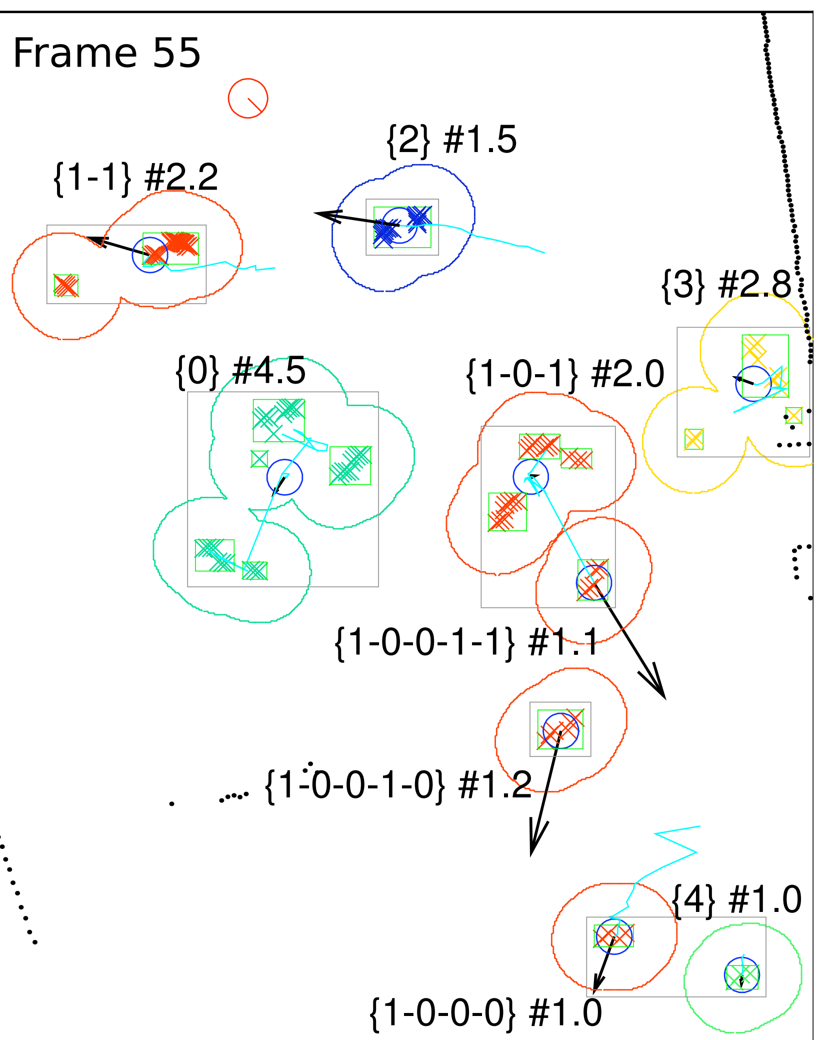
Can we build a model
of this world ?
Can we use it for playing
better ?

[Thon, Landwehr, De Raedt, ECML08]



Analyzing Video Data



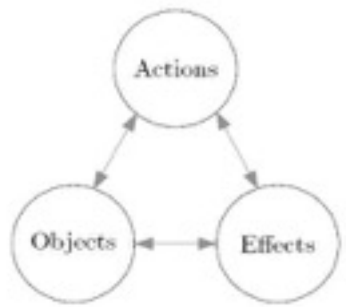


- Track people or objects over time? Even if temporarily hidden?
- Recognize activities?
- Infer object properties?

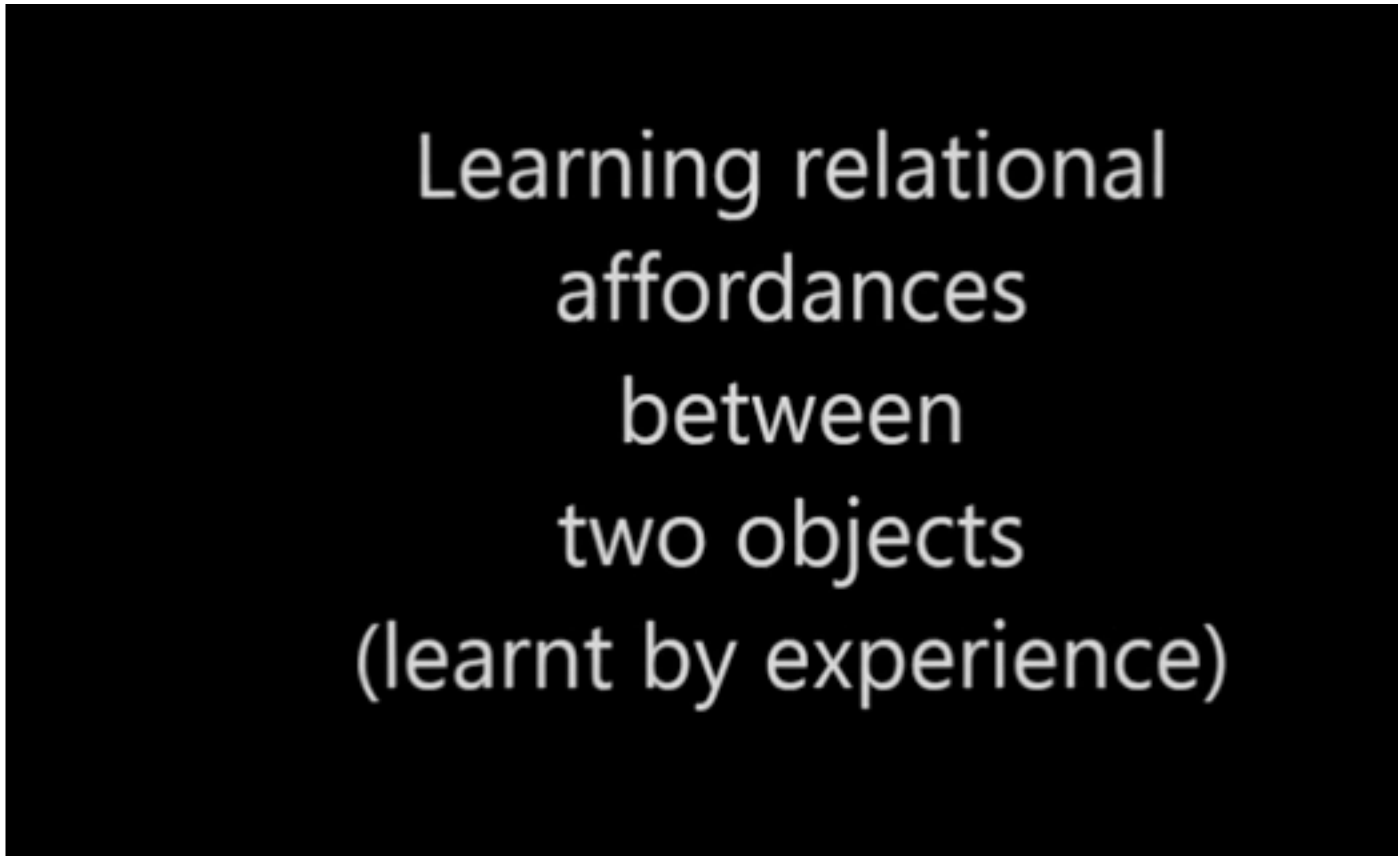
[Skarlatidis et al, TPLP 14; Nitti et al, IROS 13, ICRA 14]

Learning relational affordances

Learn probabilistic model



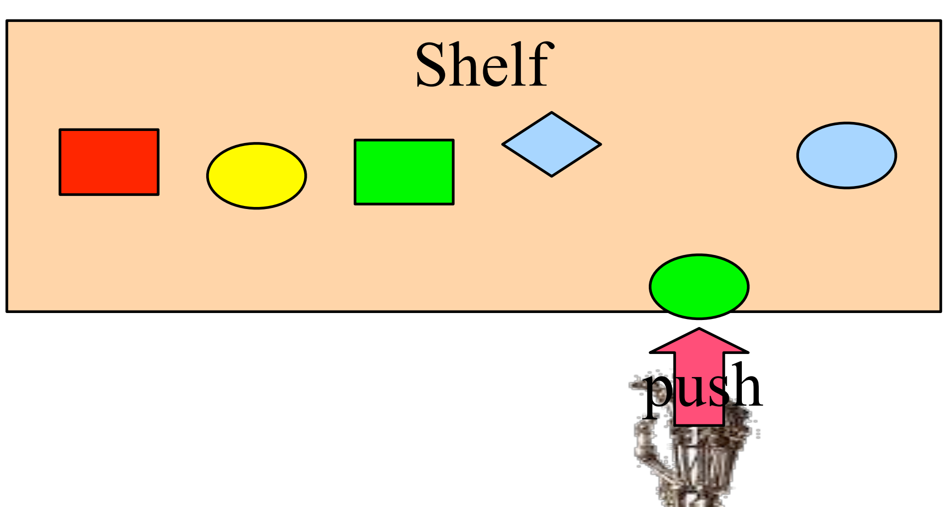
Inputs	Outputs	Function
(O, A)	E	Effect prediction
(O, E)	A	Action recognition/planning
(A, E)	O	Object recognition/selection

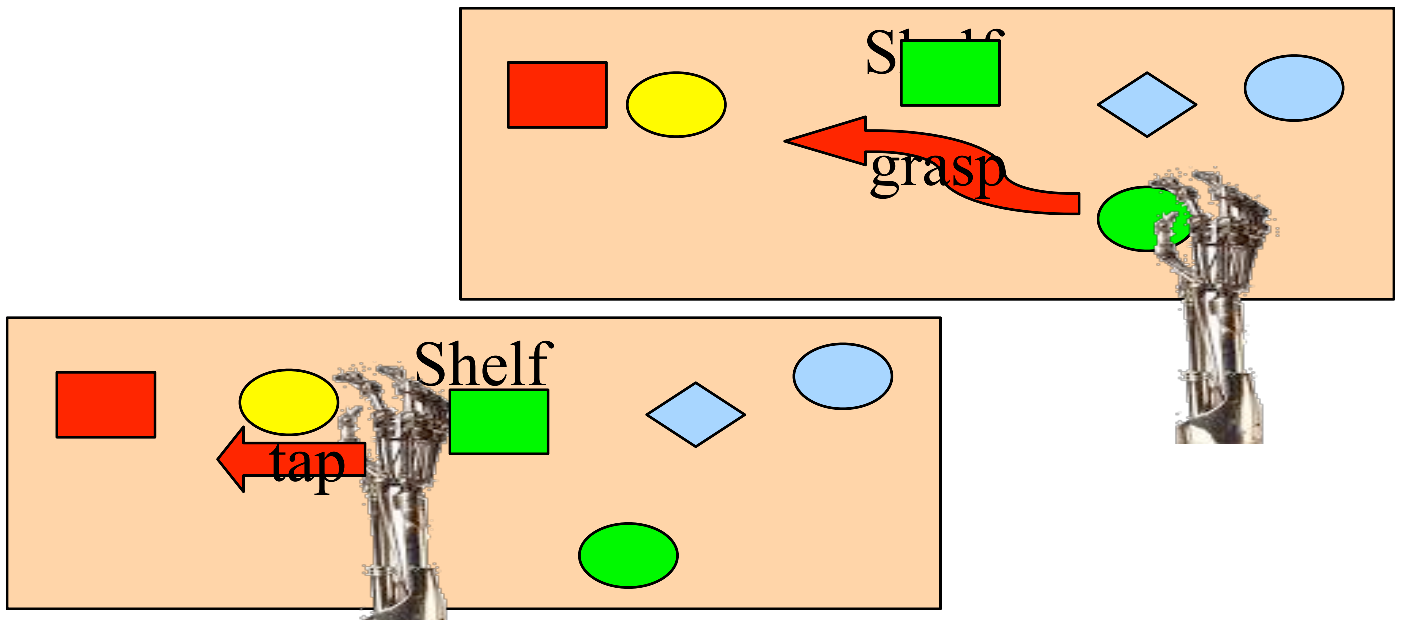


From two object interactions

Generalize to N

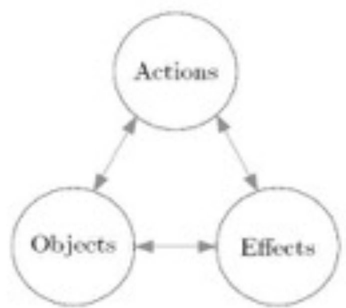
***Moldovan** et al. ICRA 12, 13, 14, PhD 15*



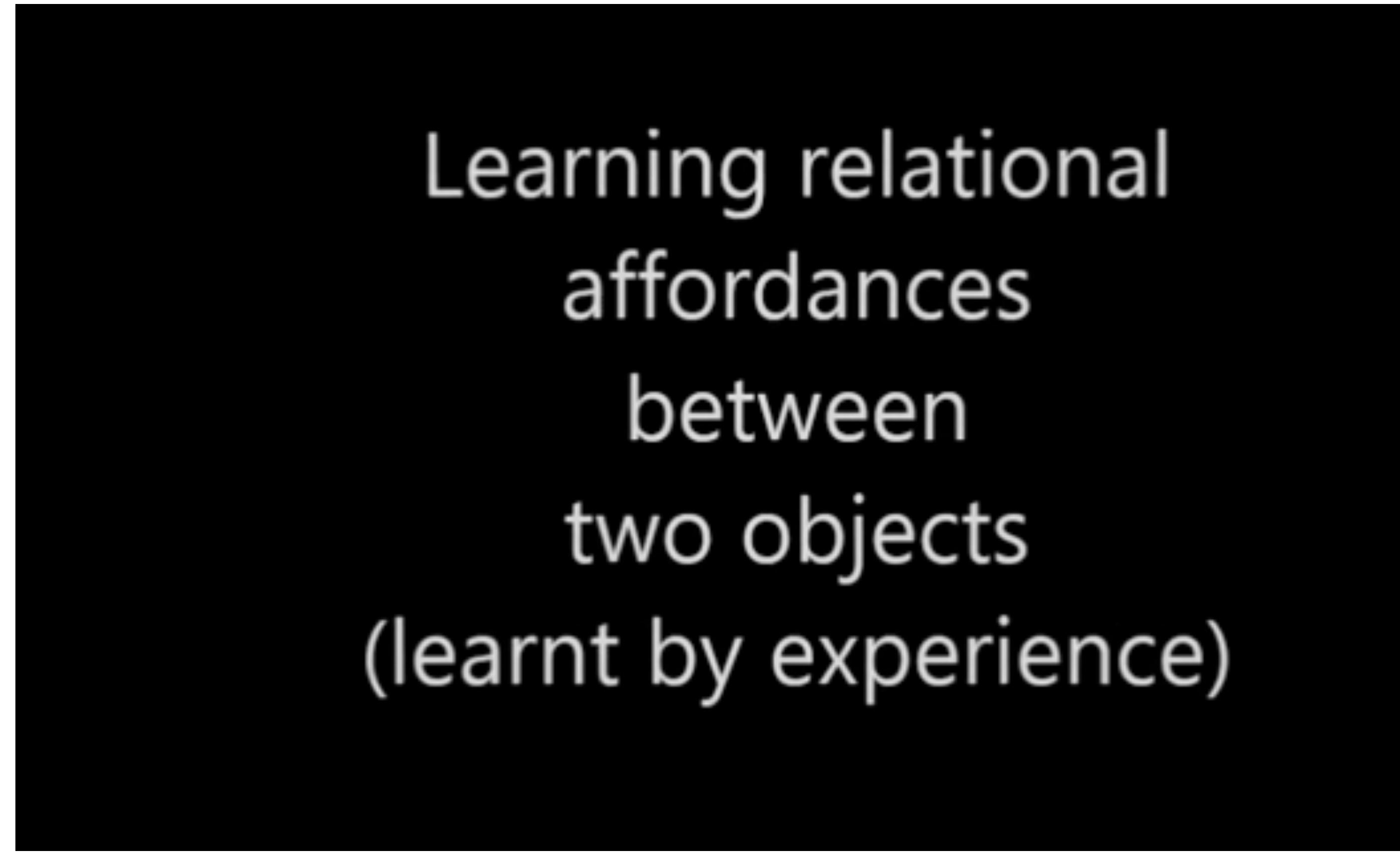


Learning relational affordances

Learn probabilistic model



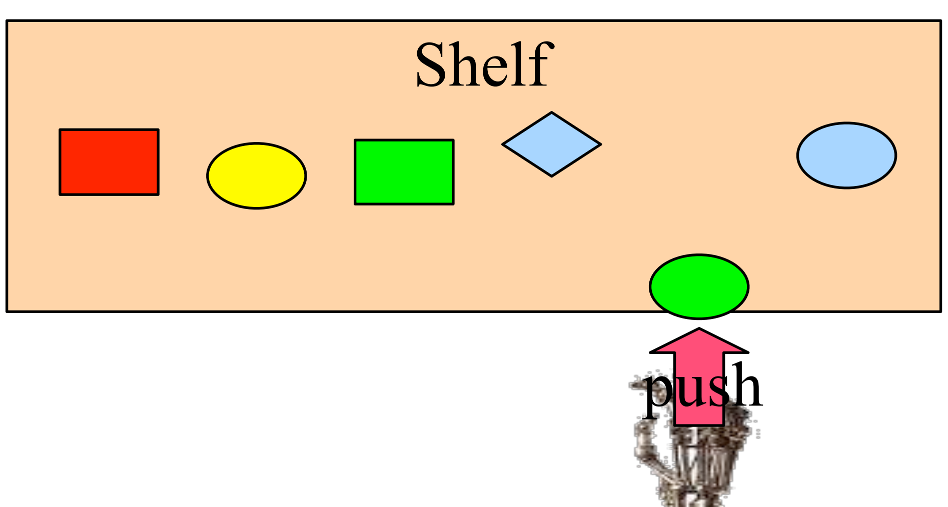
Inputs	Outputs	Function
(O, A)	E	Effect prediction
(O, E)	A	Action recognition/planning
(A, E)	O	Object recognition/selection

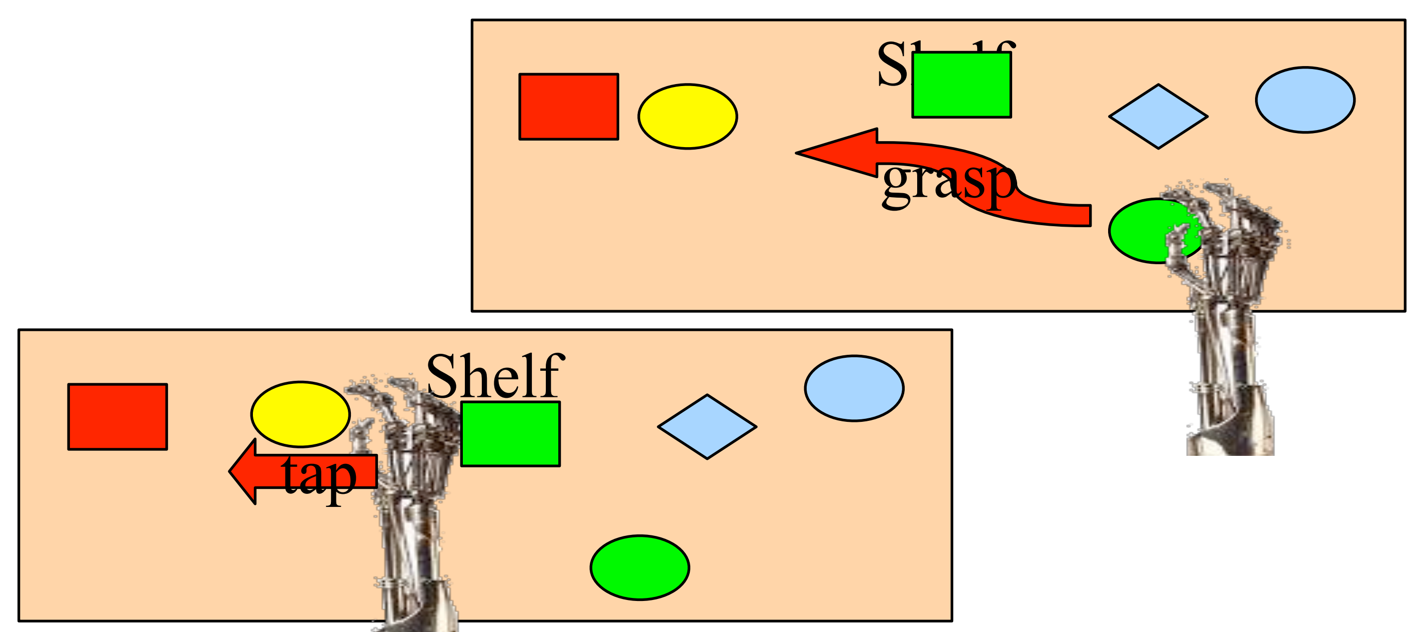


From two object interactions

Generalize to N

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





Example: Information Extraction

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

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

Example: Information Extraction

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

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

Example: Information Extraction

Recently-Learned Facts twitter

Refresh

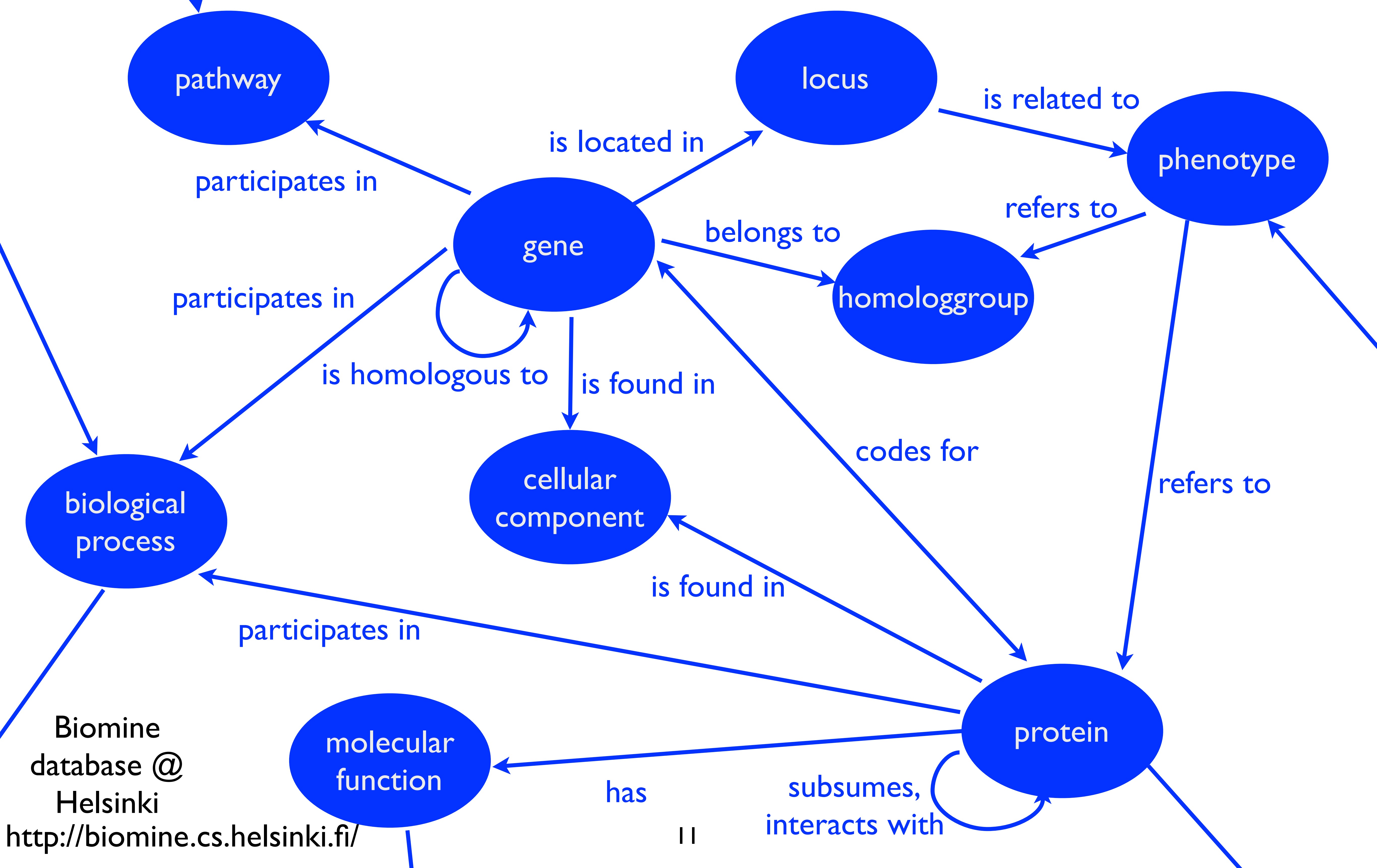
instance	iteration	date learned	confidence
kelly_andrews is a female	826	29-mar-2014	98.7
investment_next_year is an economic sector	829	10-apr-2014	95.3
shibenik is a geopolitical entity that is an organization	829	10-apr-2014	97.2
quality_web_design_work is a character trait	826	29-mar-2014	91.0
mercedes_benz_cls_by_carlsson 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/

Biological Networks



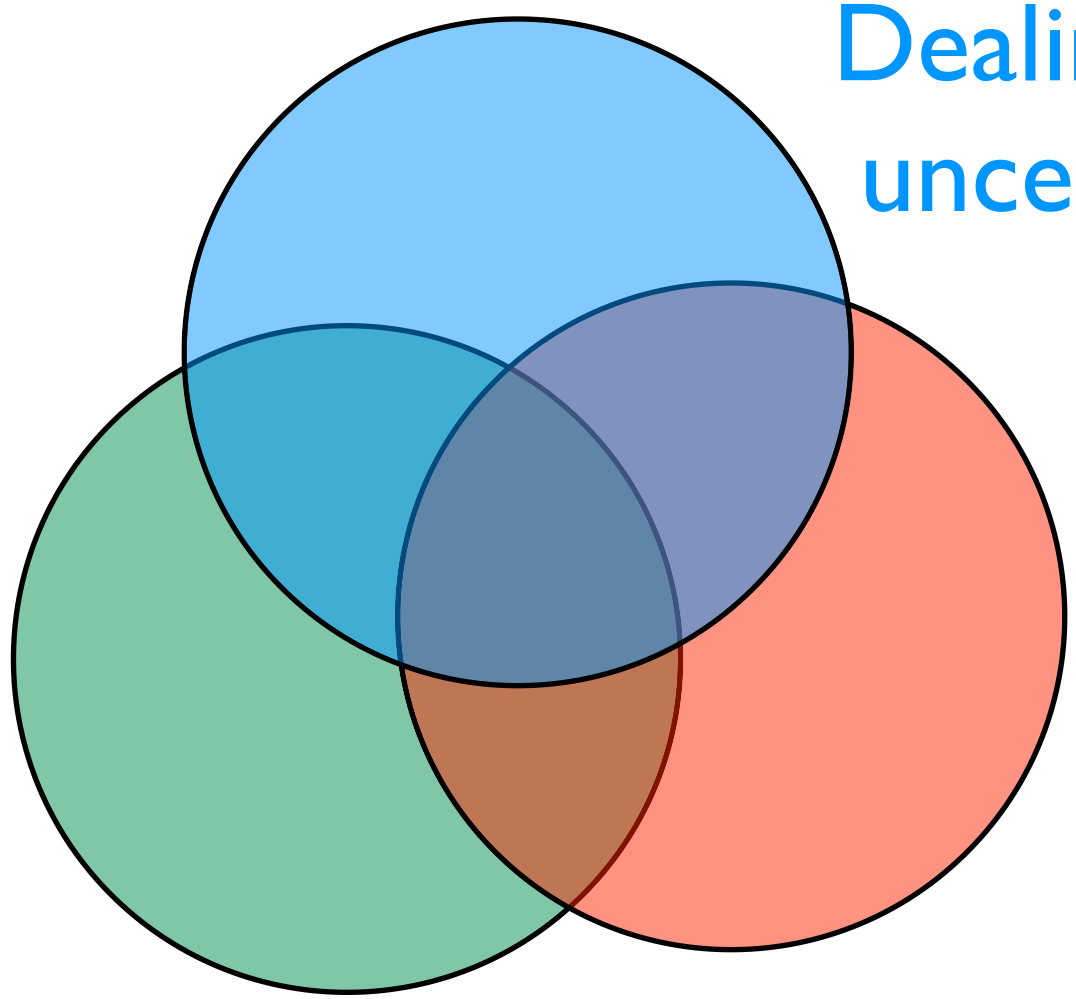
Biomine database @ Helsinki
http://biomine.cs.helsinki.fi/

This requires dealing with

- Structured environments
  - objects, and
  - relationships amongst them
- and possibly
  - using background knowledge
- cope with uncertainty
- learn from data

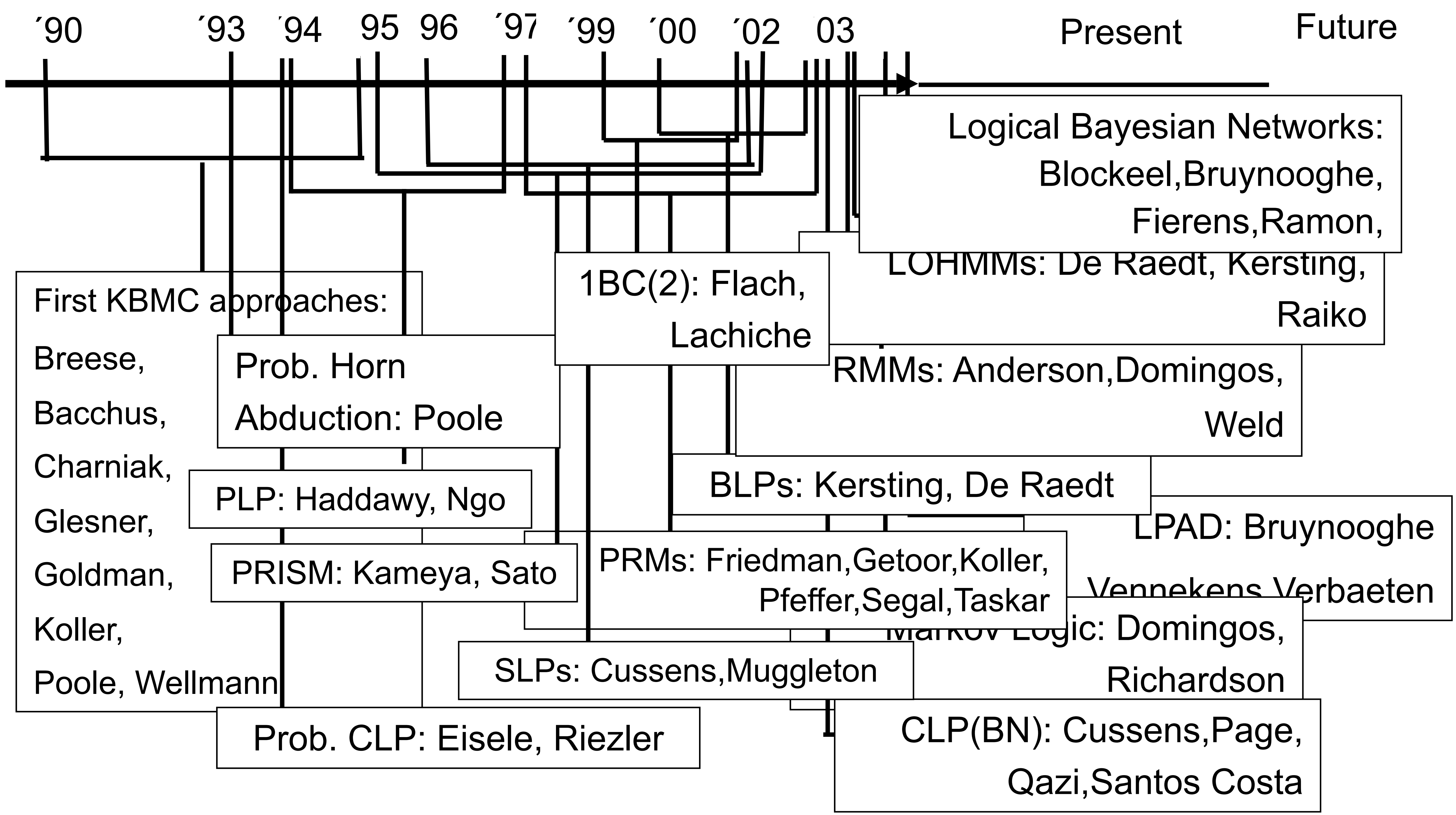
Statistical Relational Learning
Probabilistic Programming

Common theme

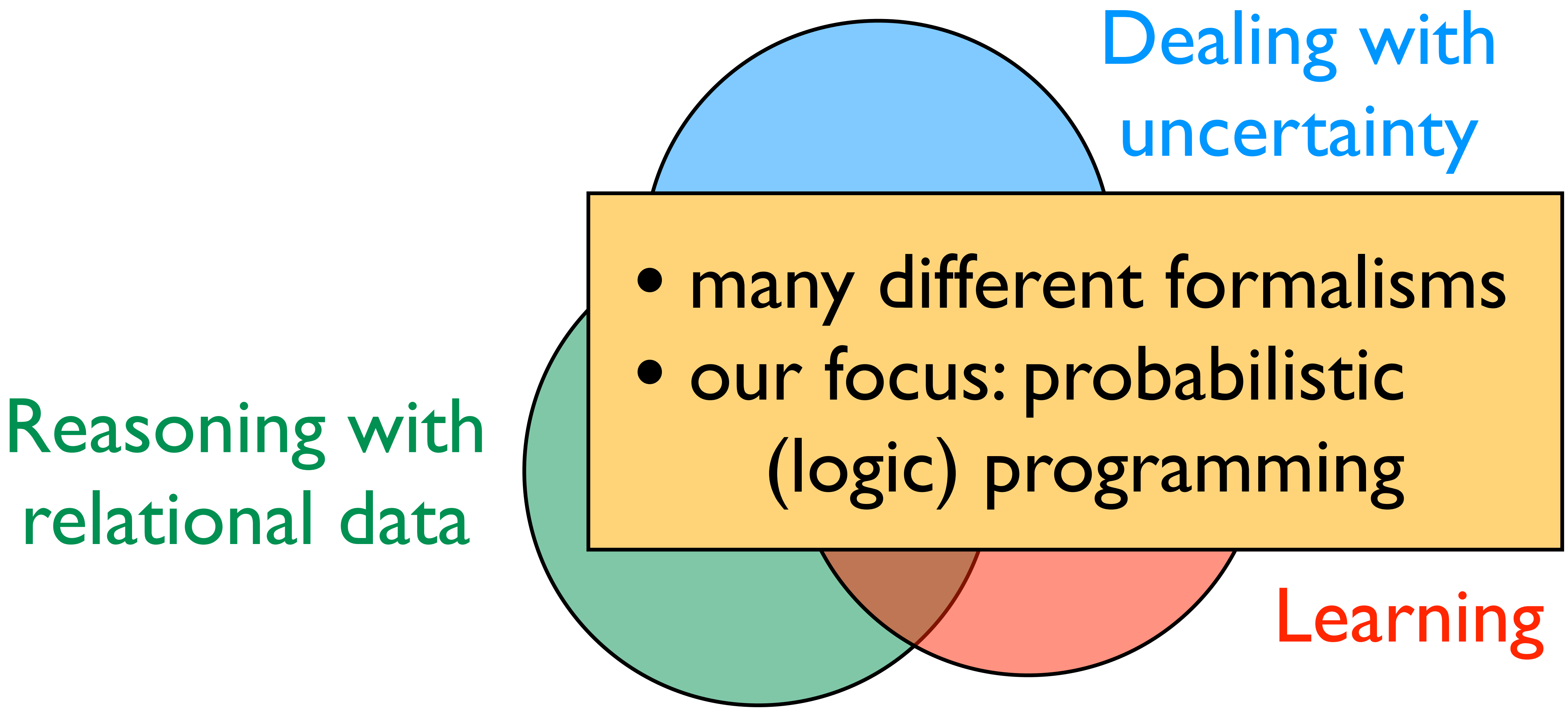


Statistical relational learning
& Probabilistic Programming, ...

Some formalisms



Common theme



Statistical relational learning
& Probabilistic Programming, ...

Probabilistic Logic Programming

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

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

multi-valued switches

probabilistic alternatives

probabilistic facts

annotated disjunctions

causal-probabilistic laws

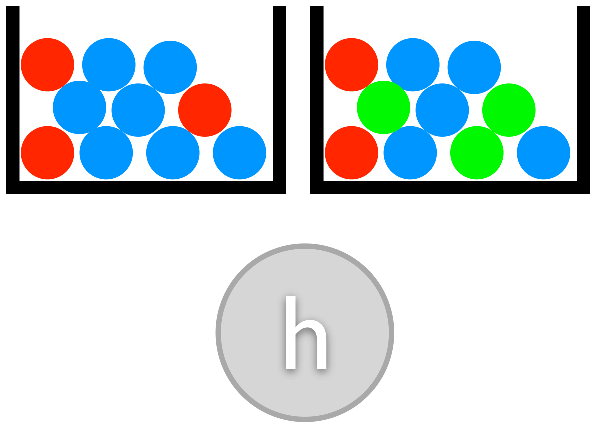
Roadmap

- Modeling (ProbLog and Church, another representative of PP)
- Inference
- Learning
- Dynamics and Decisions

... with some detours on the way

ProbLog by example:

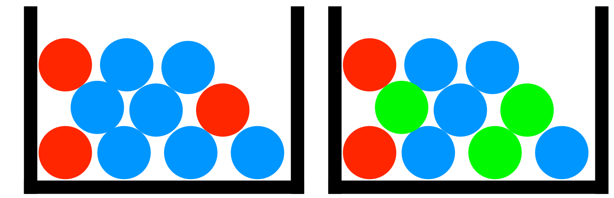
A bit of gambling



- 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



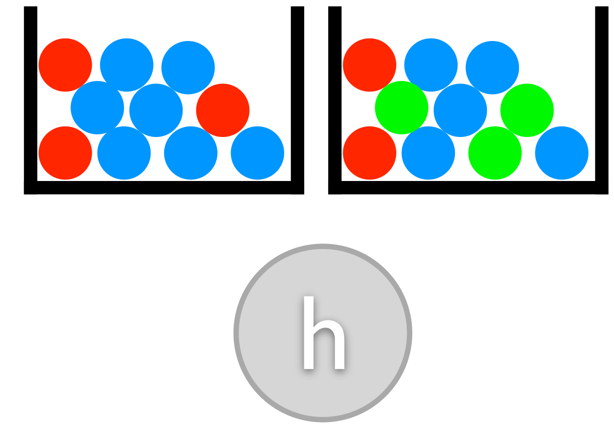
- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

```
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)

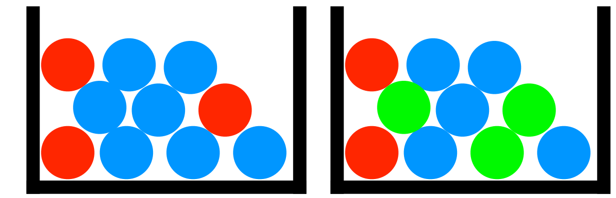
```
0.4 :: heads.
```

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).
```

ProbLog by example:

A bit of gambling



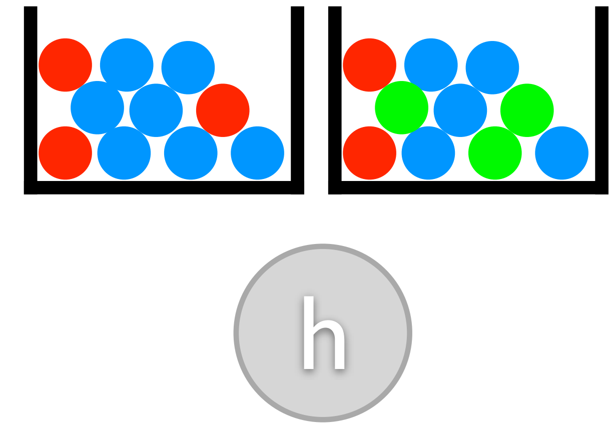
- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

```
0.4 :: heads.

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

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



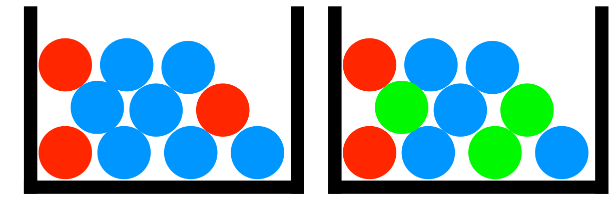
- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

```
0.4 :: heads.

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

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

logical rule encoding background knowledge

ProbLog by example:

A bit of gambling



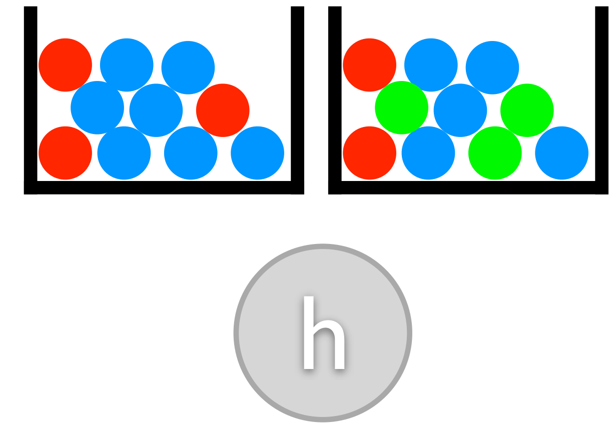
- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

```
0.4 :: heads.

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

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).
0.2 :: col(2,red); 0.3 :: col(2,green);
                   0.5 :: col(2,blue).
```

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).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

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

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

Questions

```
0.4 :: heads.

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

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

marginal probability

- Probability of `win`?

query

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

Questions

```
0.4 :: heads.

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

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)`?

evidence

- Most probable world where **`win`** is true?

Questions

```
0.4 :: heads.

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

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

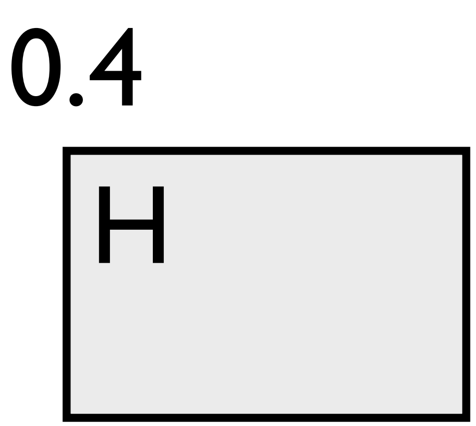
```
0.4 :: heads.

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

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

Possible Worlds

```
0.4 :: heads.

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

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

Possible Worlds

```
0.4 :: heads.

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

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



Possible Worlds

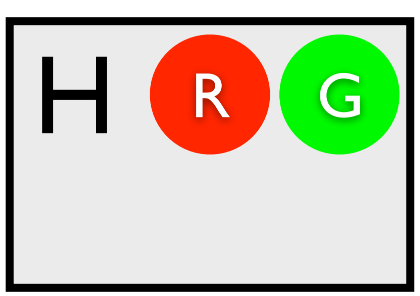
```
0.4 :: heads.

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

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

0.4×0.3

Possible Worlds

```
0.4 :: heads.

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

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

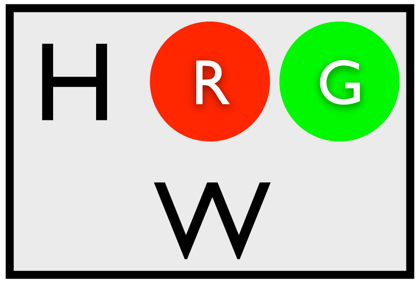
$0.4 \times 0.3 \times 0.3$



Possible Worlds

```
0.4 :: heads.

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

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

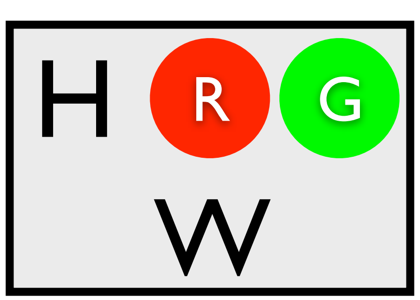
0.4 ×0.3 ×0.3



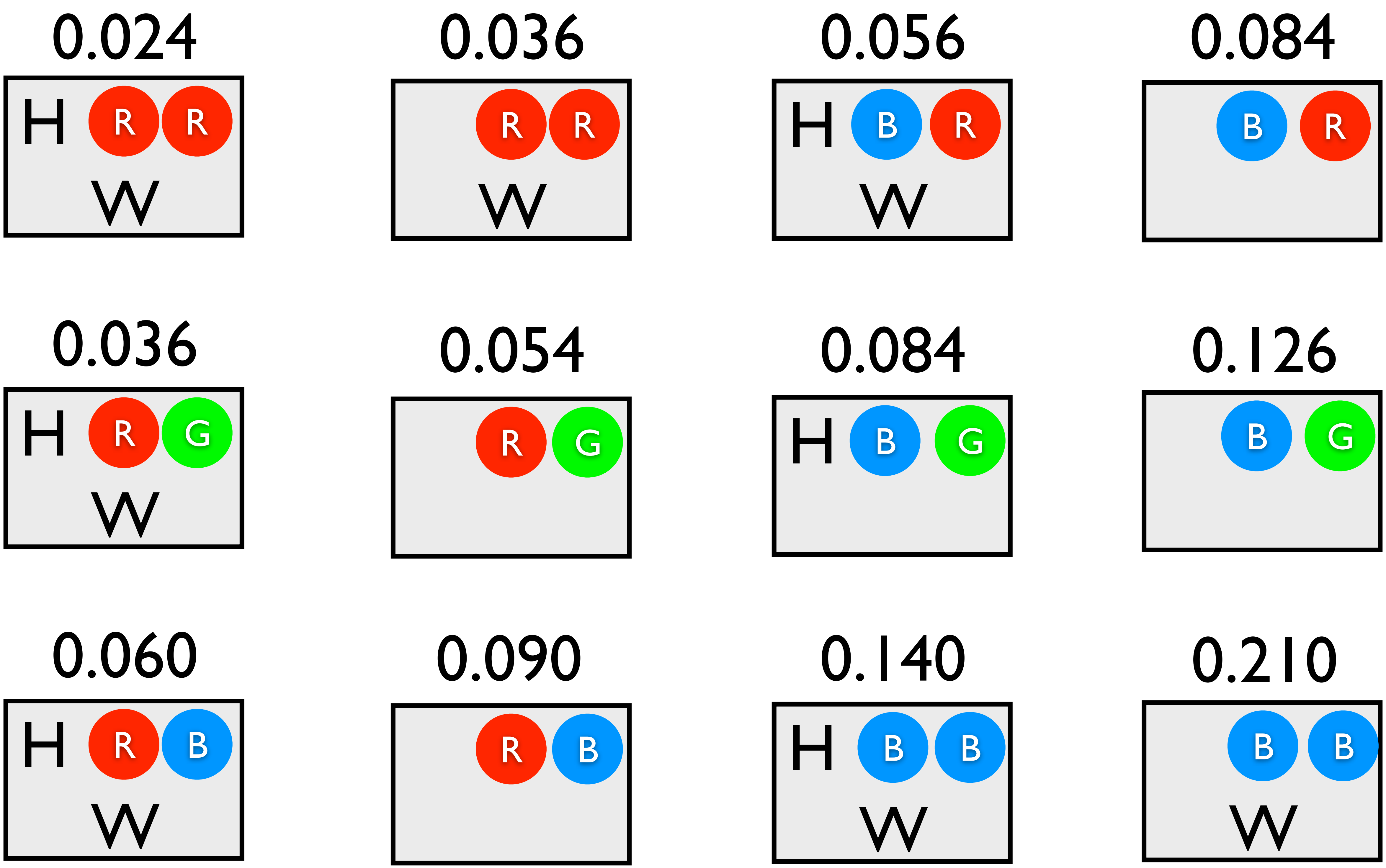
Possible Worlds

```
0.4 :: heads.

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

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

$0.4 \times 0.3 \times 0.3$

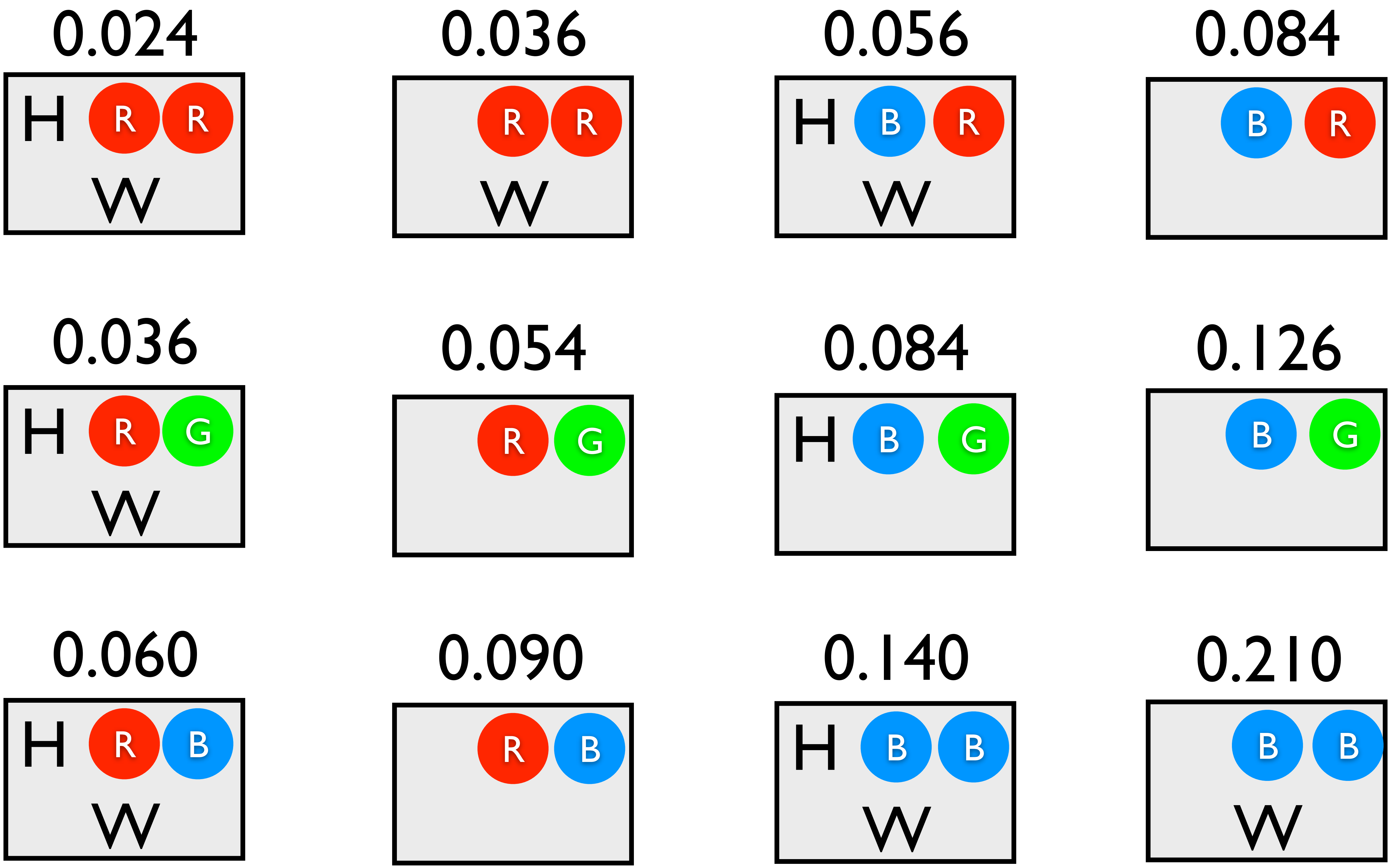


All Possible Worlds



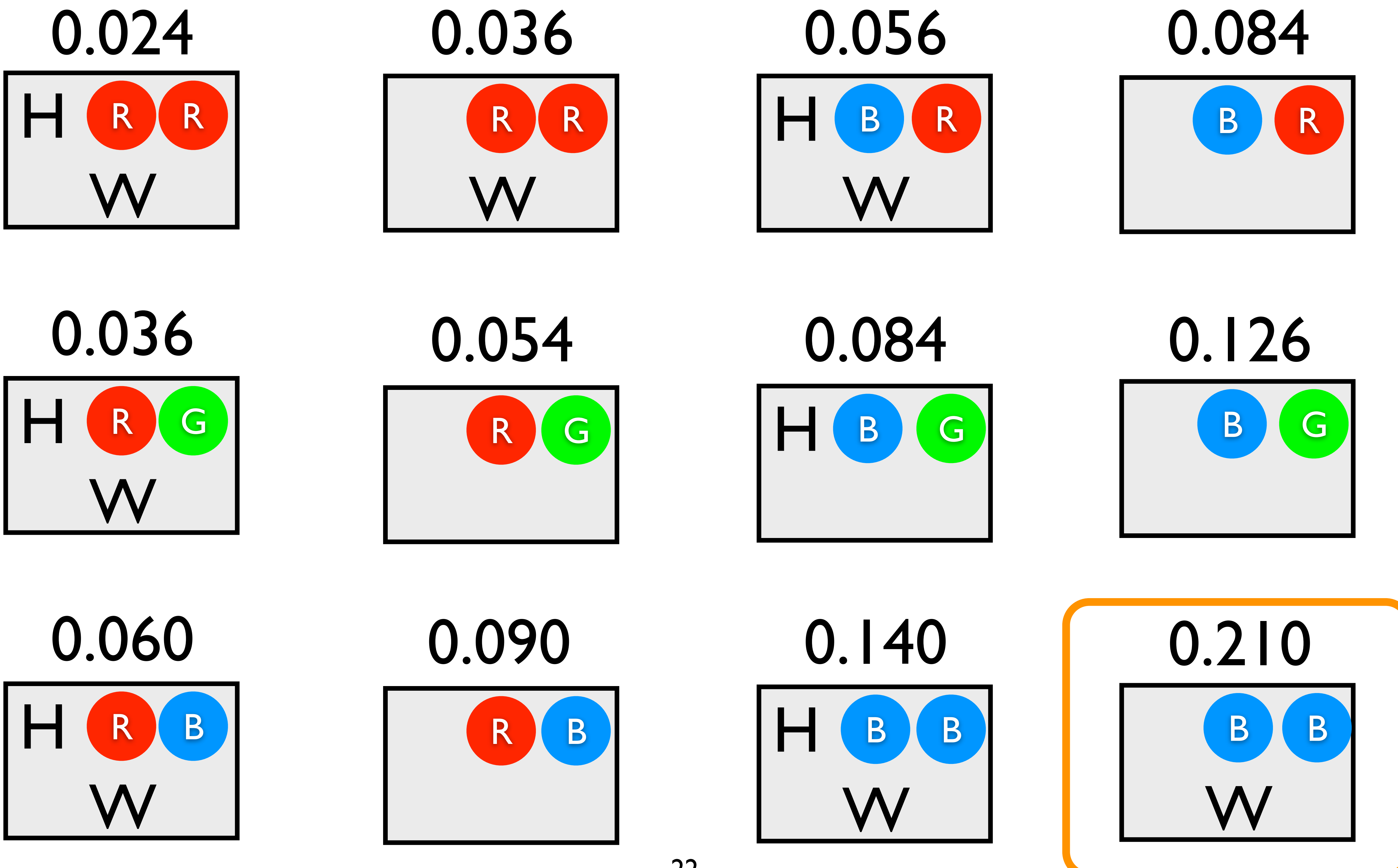
Most likely world where **win** is true?

MPE Inference



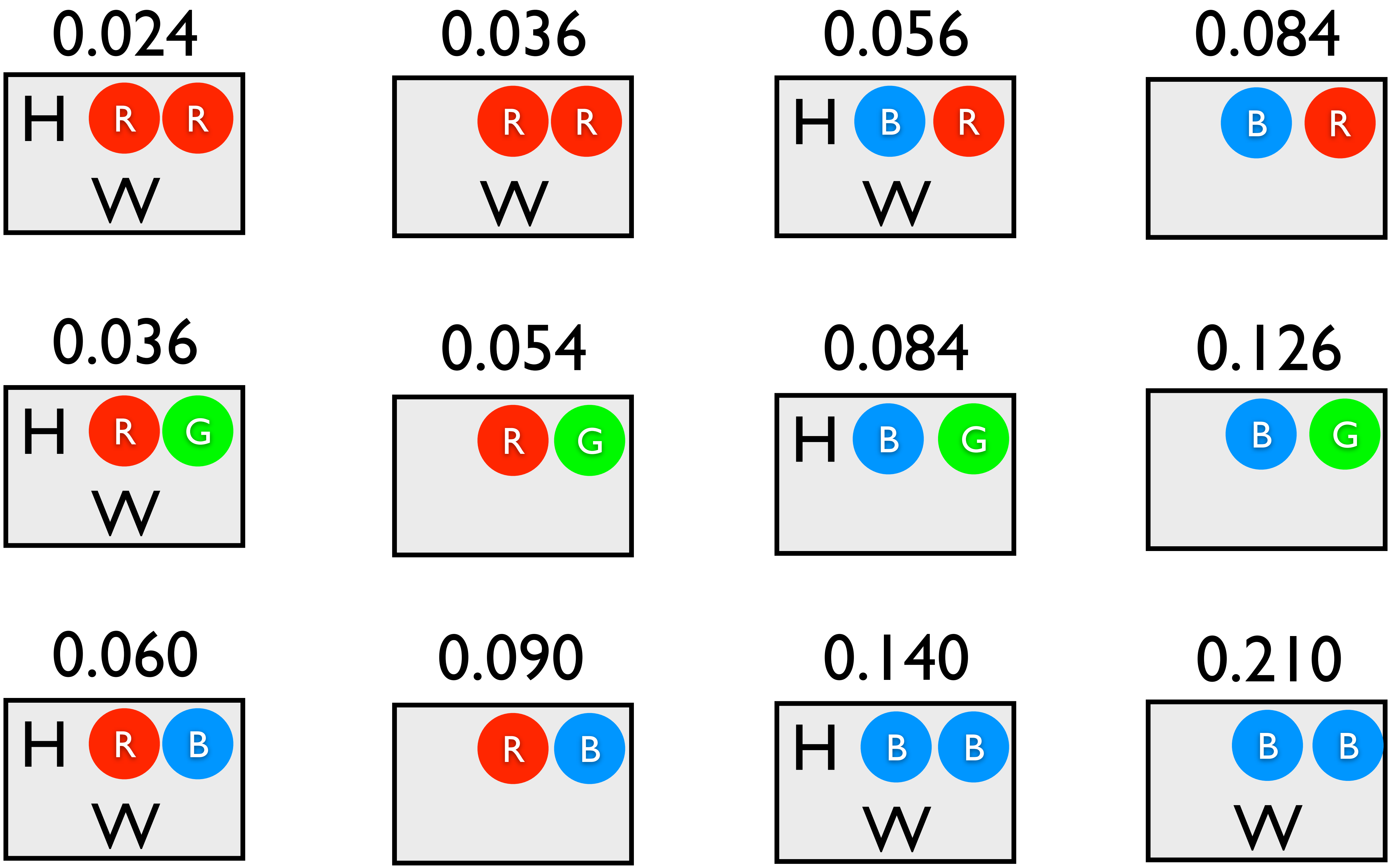
Most likely world where **`win`** is true?

MPE Inference



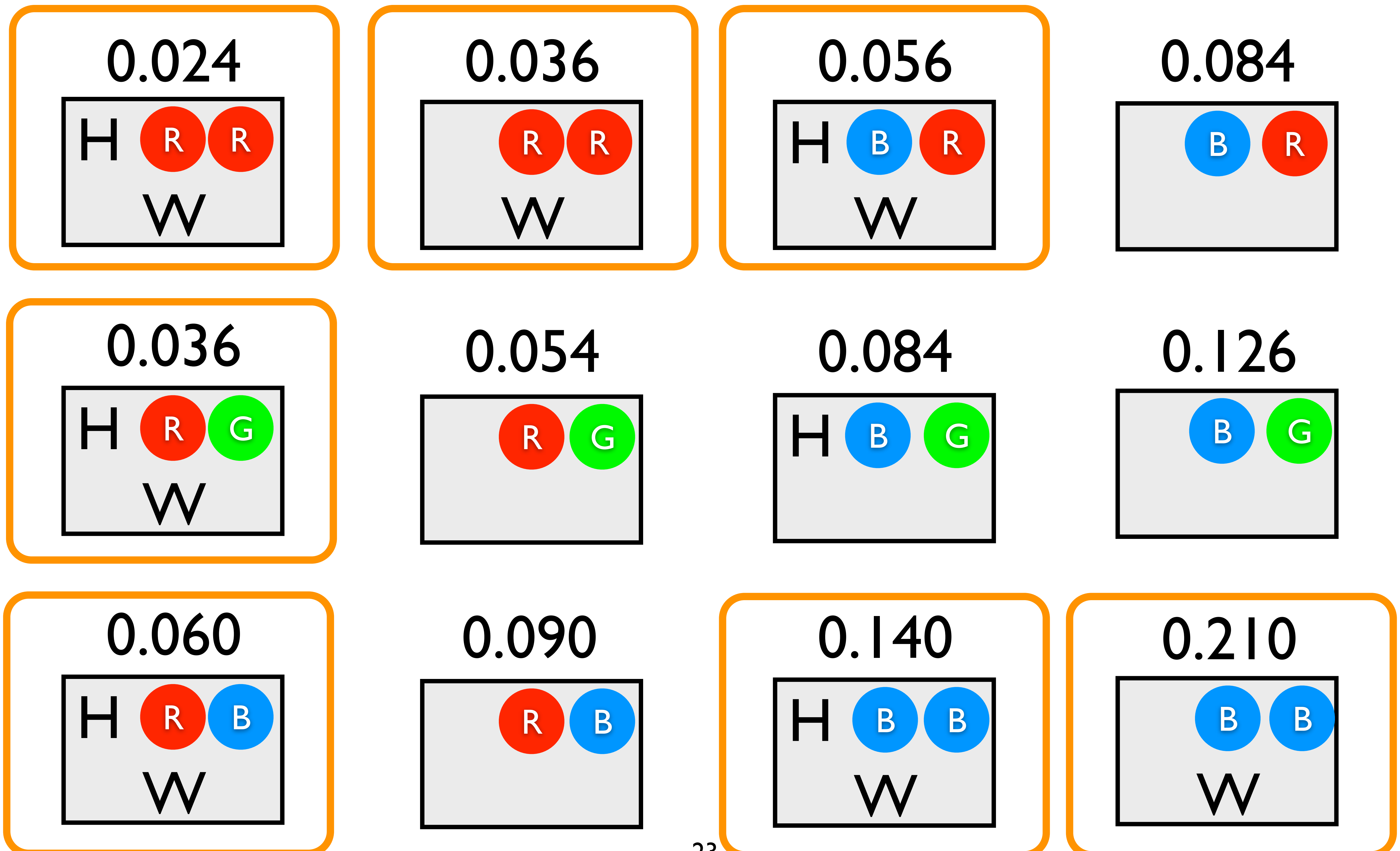
P(win)=?

Marginal Probability



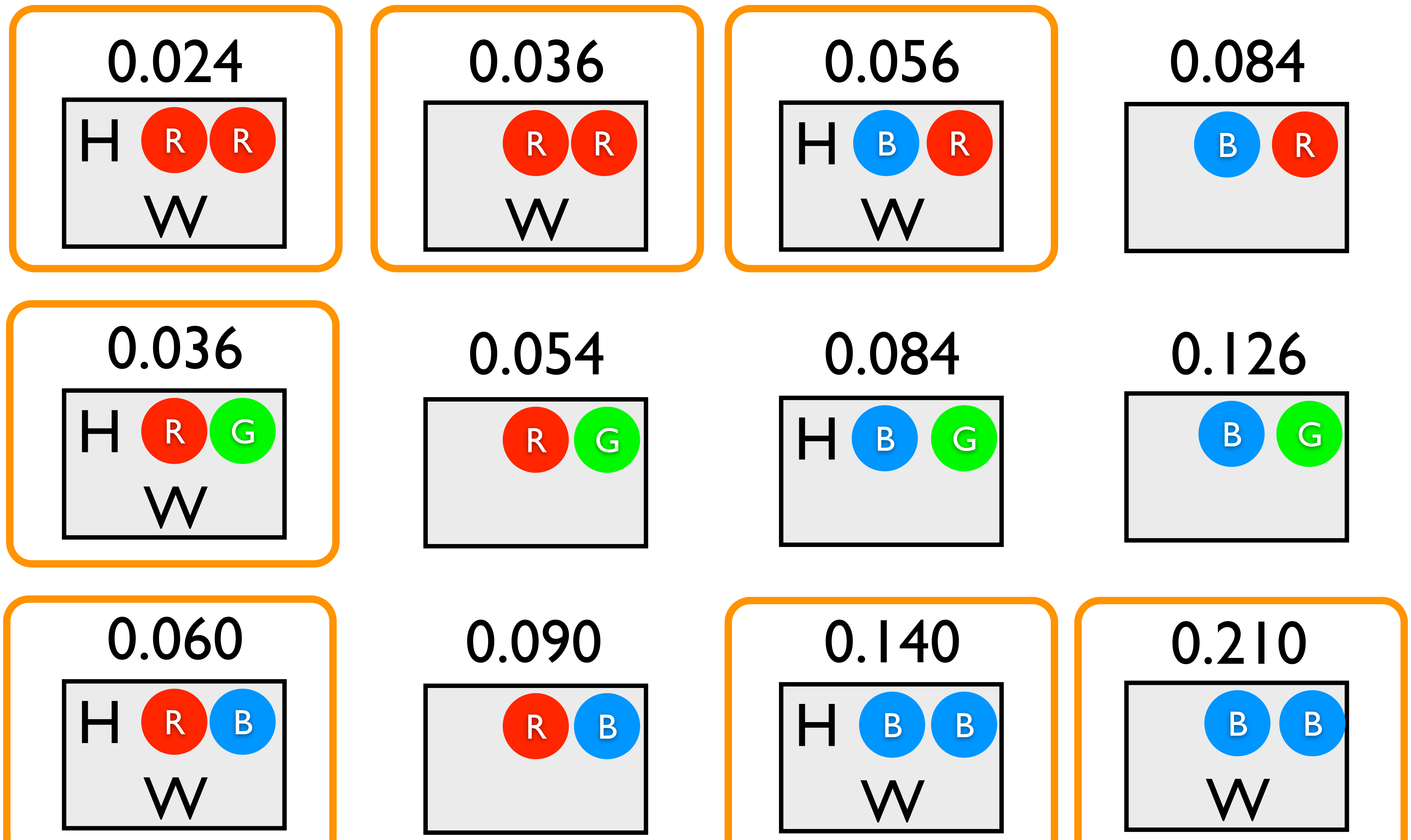
$$P(\underline{win}) = \Sigma$$

Marginal Probability



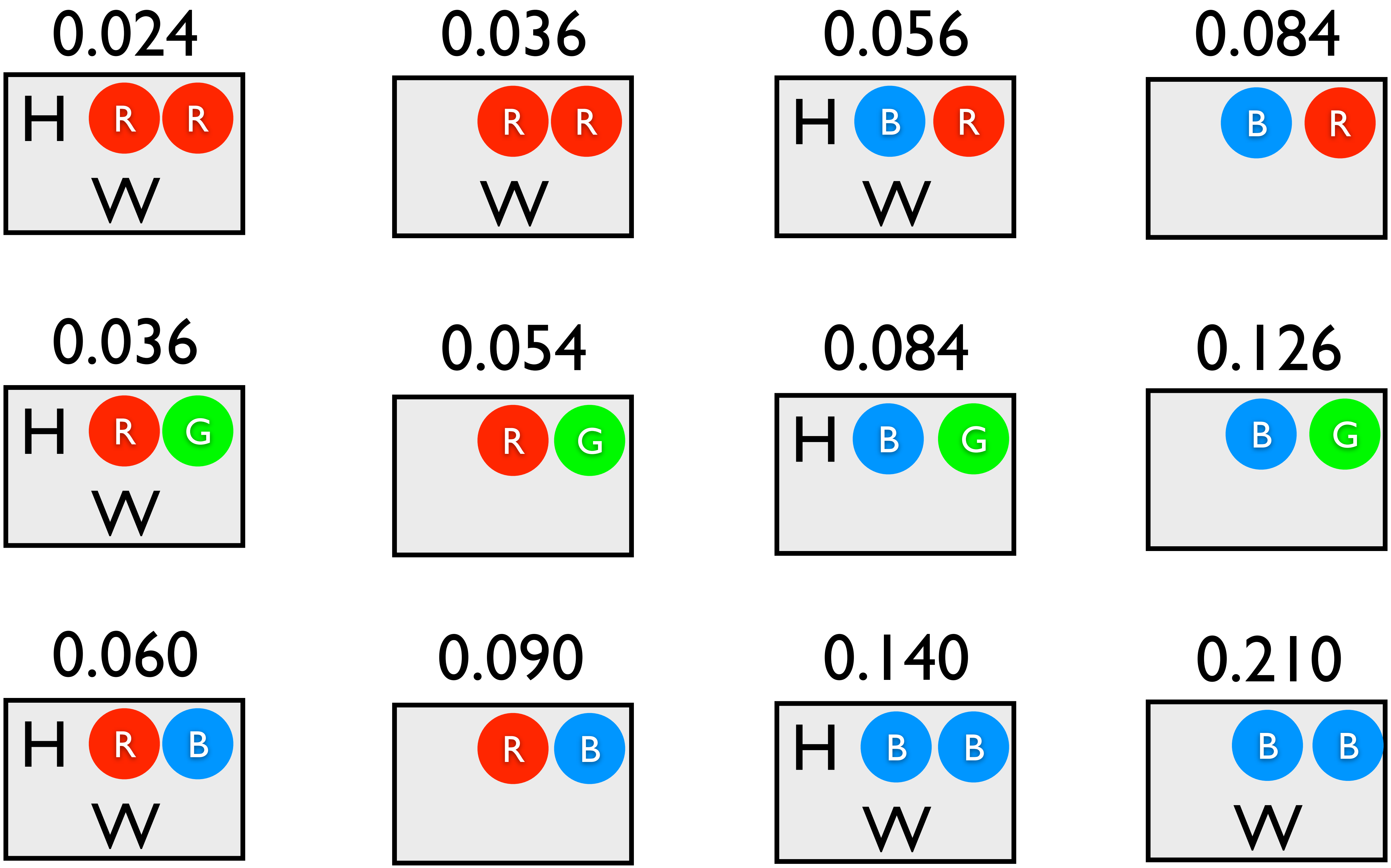
$P(\underline{win}) = \sum = 0.562$

Marginal Probability



P(win|col(2,green))= ?

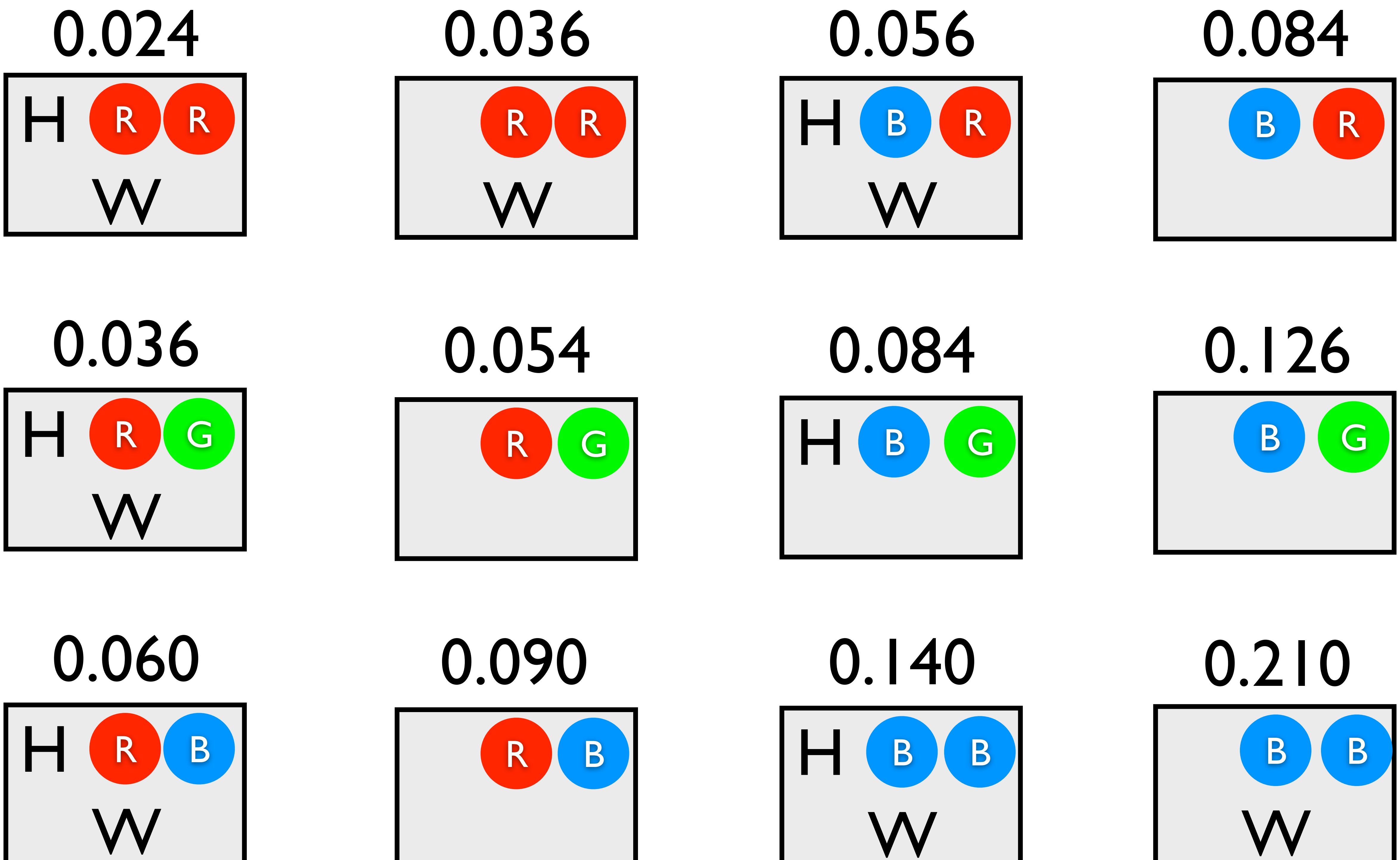
Conditional Probability



Conditional Probability

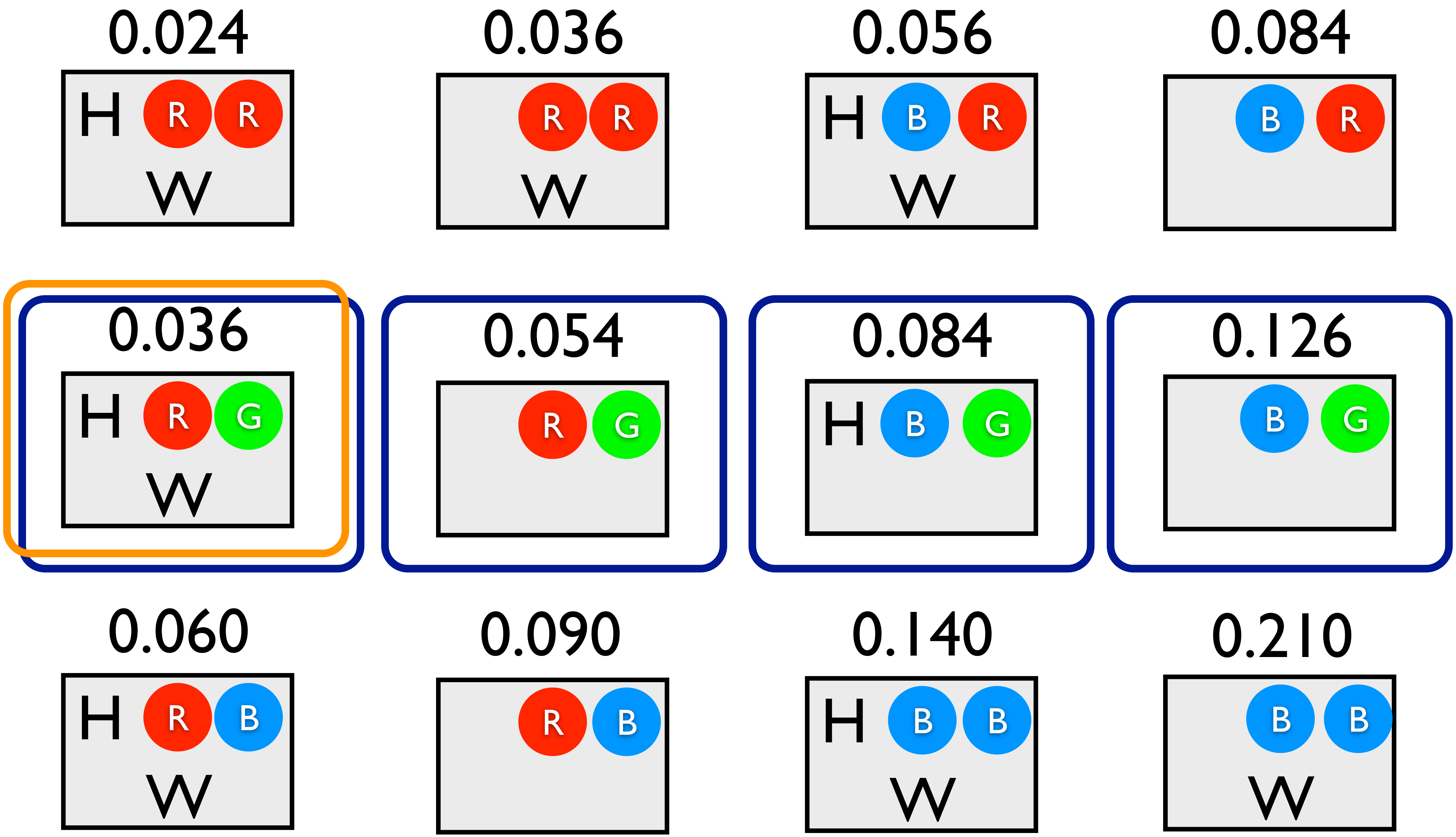
$$P(win|\underline{col(2,green)}) = \Sigma / \Sigma$$

$$= P(\underline{win \wedge col(2,green)}) / P(\underline{col(2,green)})$$



$$P(win \mid col(2,green)) = \Sigma / \Sigma$$
$$= P(win \wedge col(2,green)) / P(col(2,green))$$

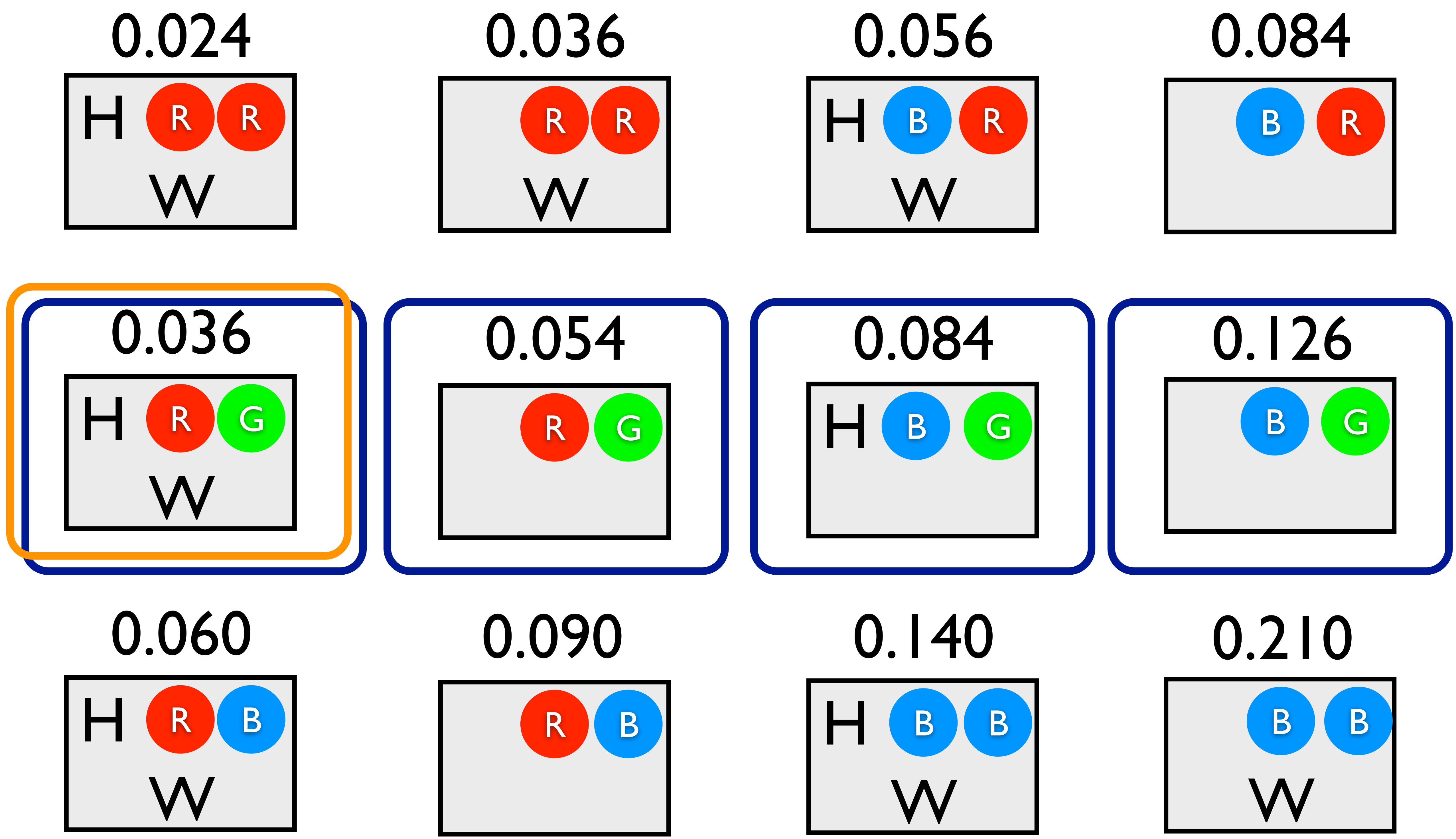
Conditional Probability



$P(win|col(2,green)) = \Sigma / \Sigma$

$= 0.036/0.3 = 0.12$

Conditional Probability

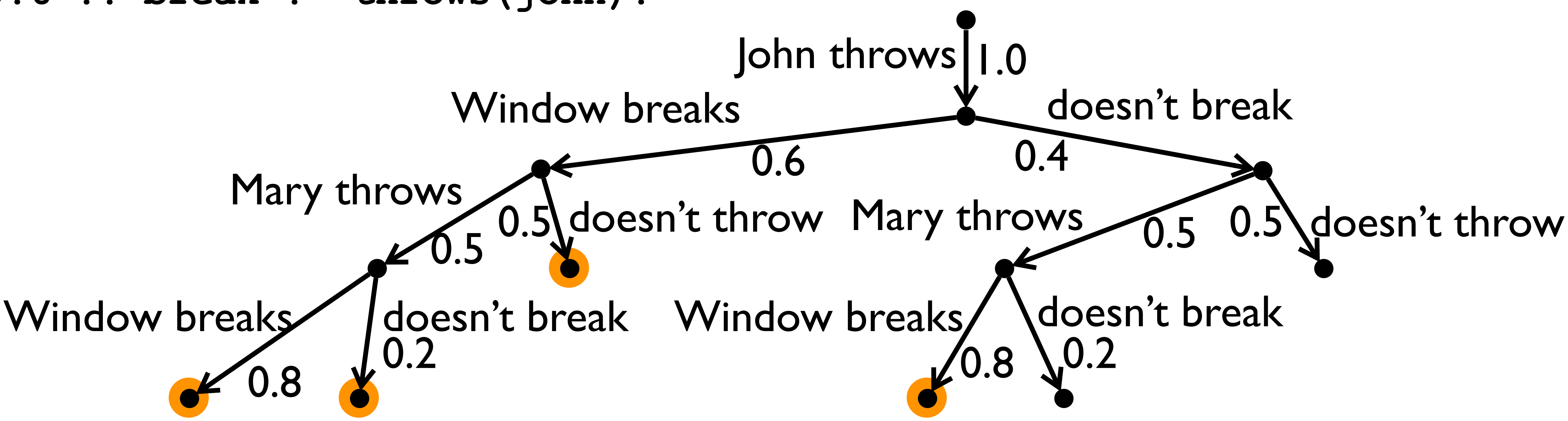


[Vennekens et al, ICLP 04]

Alternative view: CP-Logic

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

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

probabilistic causal laws



P(break)=0.6×0.5×0.8+0.6×0.5×0.2+0.6×0.5+0.4×0.5×0.8

Distributional Clauses (DC)

Closely related to BLOG [Russell et al.]

- Discrete- and continuous-valued random variables

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

Distributional Clauses (DC)

Closely related to BLOG [Russell et al.]

- 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)

Closely related to BLOG [Russell et al.]

- 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)

Closely related to BLOG [Russell et al.]

- 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)

Closely related to BLOG [Russell et al.]

- 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).
```

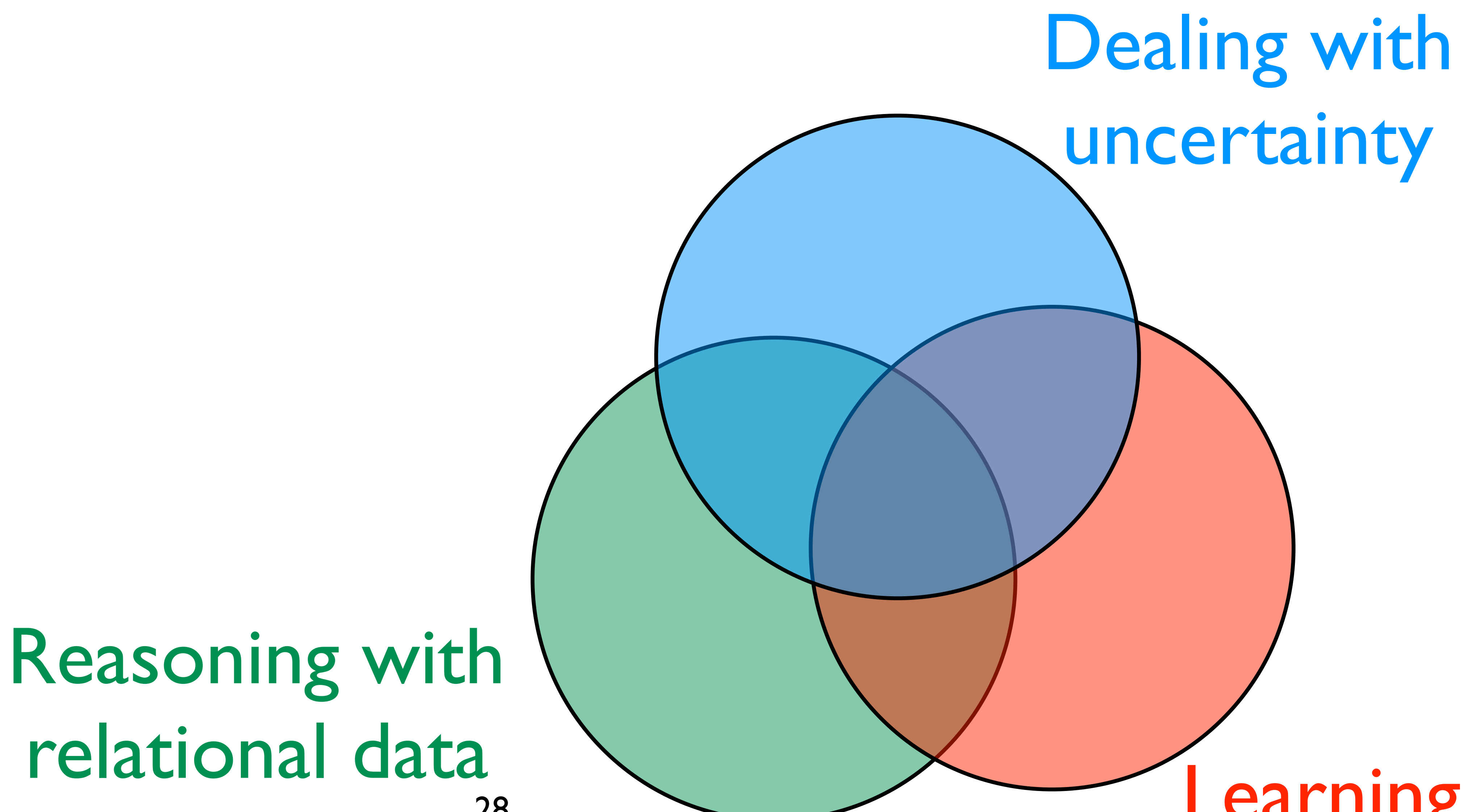
[Gutmann et al, TPLP 11; Nitti et al, IROS 13]

Distributional Clauses (DC)

- Defines a generative process (as for CP-logic)
- Logic programming variant of Blog
- Tree can become infinitely wide
  - Sampling
- Well-defined under reasonable assumptions

[Suciu et al 2011]

Probabilistic Databases



[Suciu et al 2011]

Probabilistic Databases

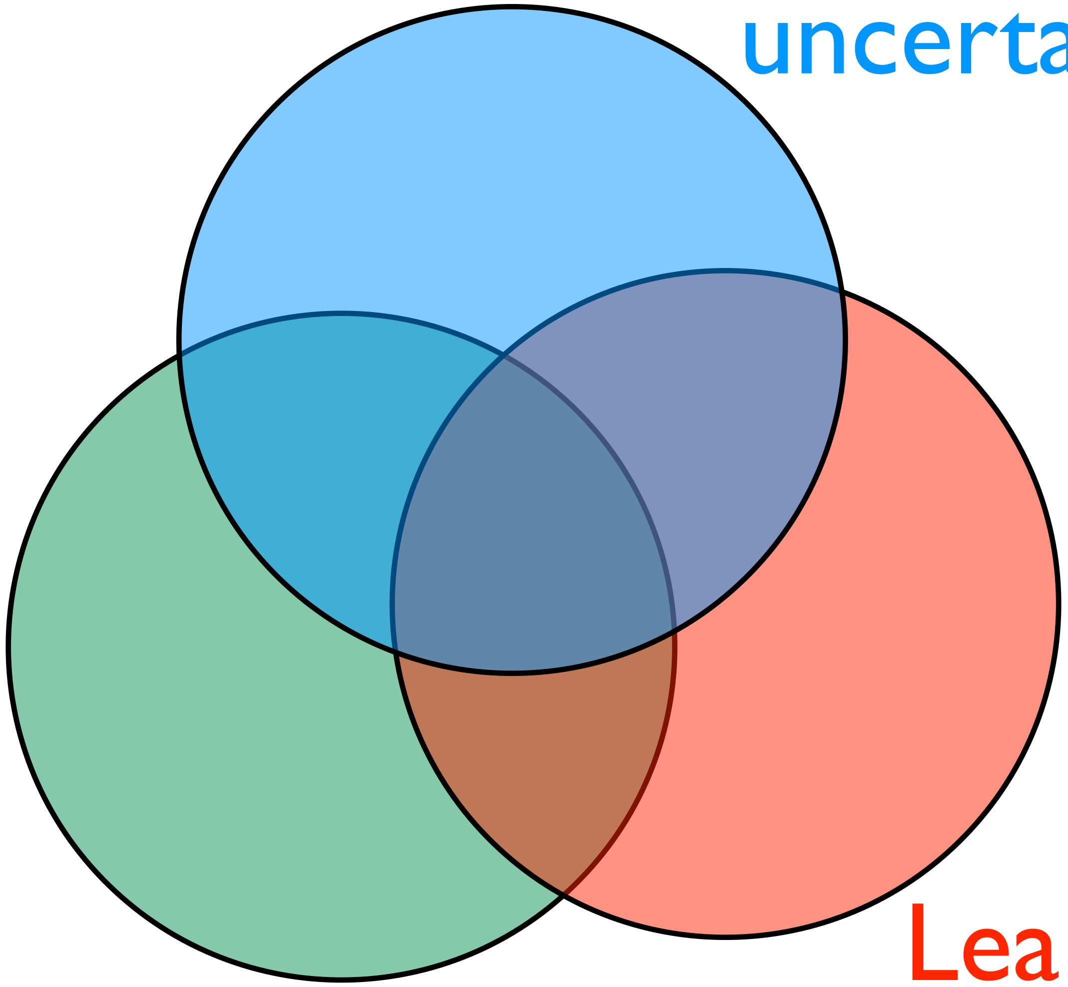
```
select x.person, y.country
from bornIn x, cityIn y
where x.city=y.city
```

bornIn

person	city
ann	london
bob	york
eve	new york
tom	paris

cityIn

city	country
london	uk
york	uk
paris	usa



[Suciu et al 2011]

Probabilistic Databases

```
select x.person, y.country
from bornIn x, cityIn y
where x.city=y.city
```

one world

bornIn

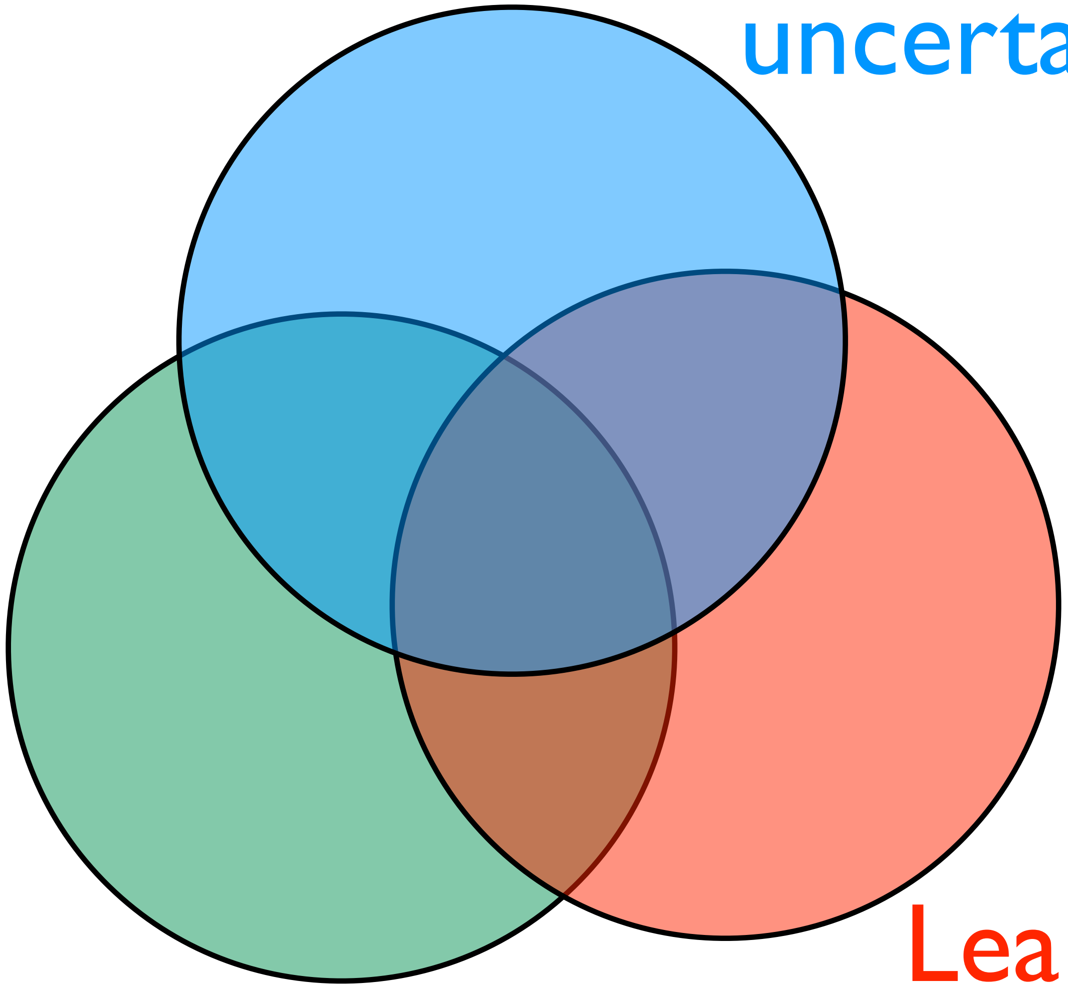
person	city
ann	london
bob	york
eve	new york
tom	paris

cityIn

city	country
london	uk
york	uk
paris	usa

relational database

Dealing with uncertainty



Learning

[Suciu et al 2011]

Probabilistic Databases

bornIn

person	city	P
ann	london	0,87
bob	york	0,95
eve	new york	0,9
tom	paris	0,56

cityIn

city	country	P
london	uk	0,99
york	uk	0,75
paris	usa	0,4

```
select x.person, y.country
from bornIn x, cityIn y
where x.city=y.city
```

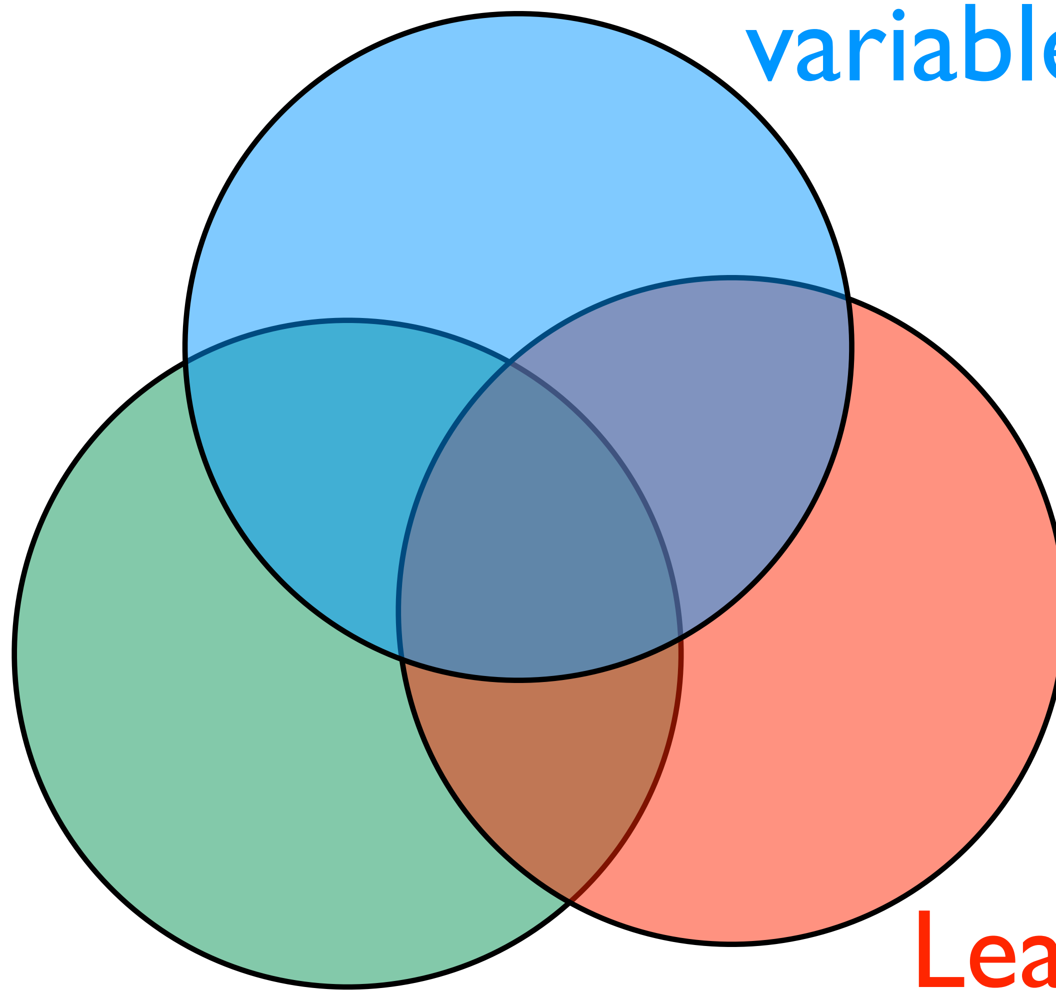
one world

bornIn

person	city
ann	london
bob	york
eve	new york
tom	paris

cityIn

city	country
london	uk
york	uk
paris	usa



[Suciu et al 2011]

Probabilistic Databases

several possible worlds

bornIn

person	city	P
ann	london	0,87
bob	york	0,95
eve	new york	0,9
tom	paris	0,56

cityIn

city	country	P
london	uk	0,99
york	uk	0,75
paris	usa	0,4

tuples as random variables

```
select x.person, y.country
from bornIn x, cityIn y
where x.city=y.city
```

one world

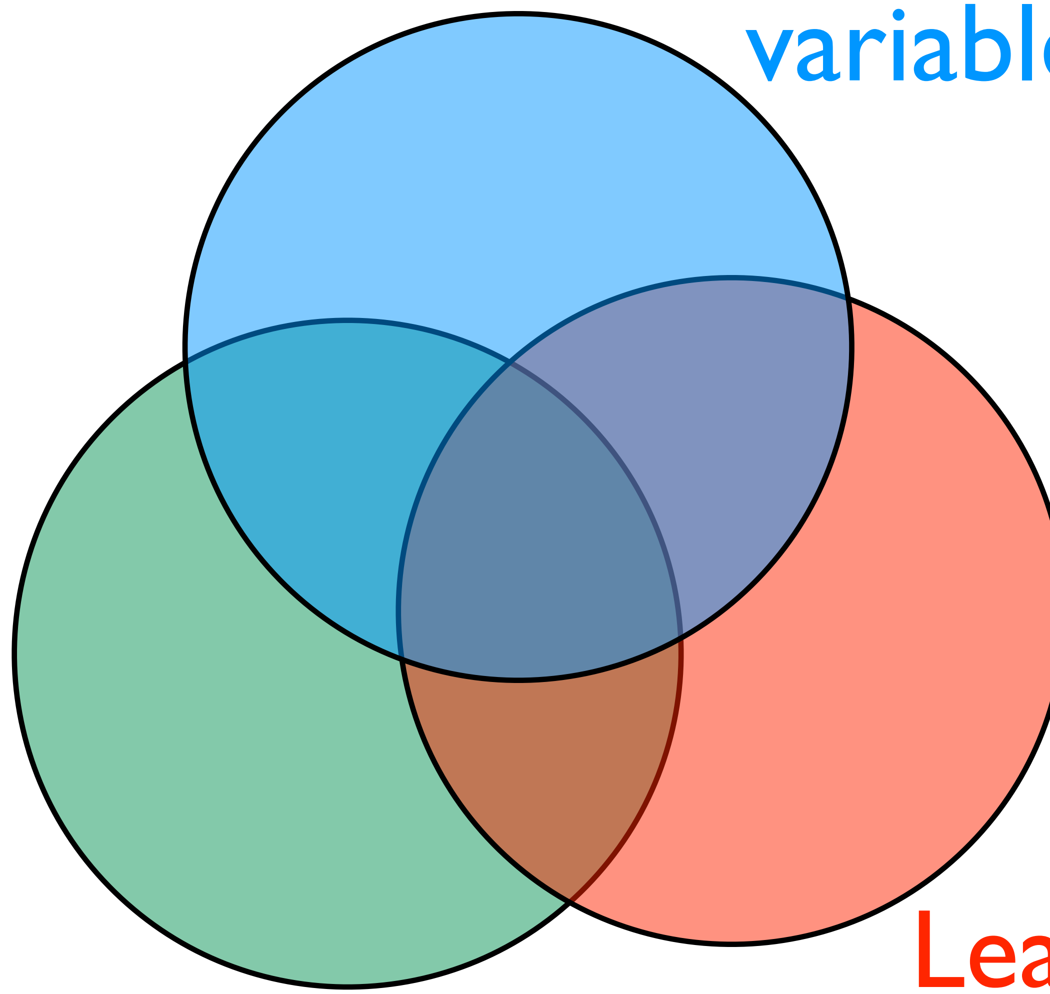
bornIn

person	city
ann	london
bob	york
eve	new york
tom	paris

cityIn

city	country
london	uk
york	uk
paris	usa

relational database



Learning

[Suciu et al 2011]

Probabilistic Databases

several possible worlds

bornIn

person	city	P
ann	london	0,87
bob	york	0,95
		0,9
		0,56

cityIn

city	country	P
london	uk	0,99
york	uk	0,75
paris	usa	0,4

probabilistic tables + database queries
→ distribution over possible worlds

```
select
from bornIn x, cityIn y
where x.city=y.city
```

one world

bornIn

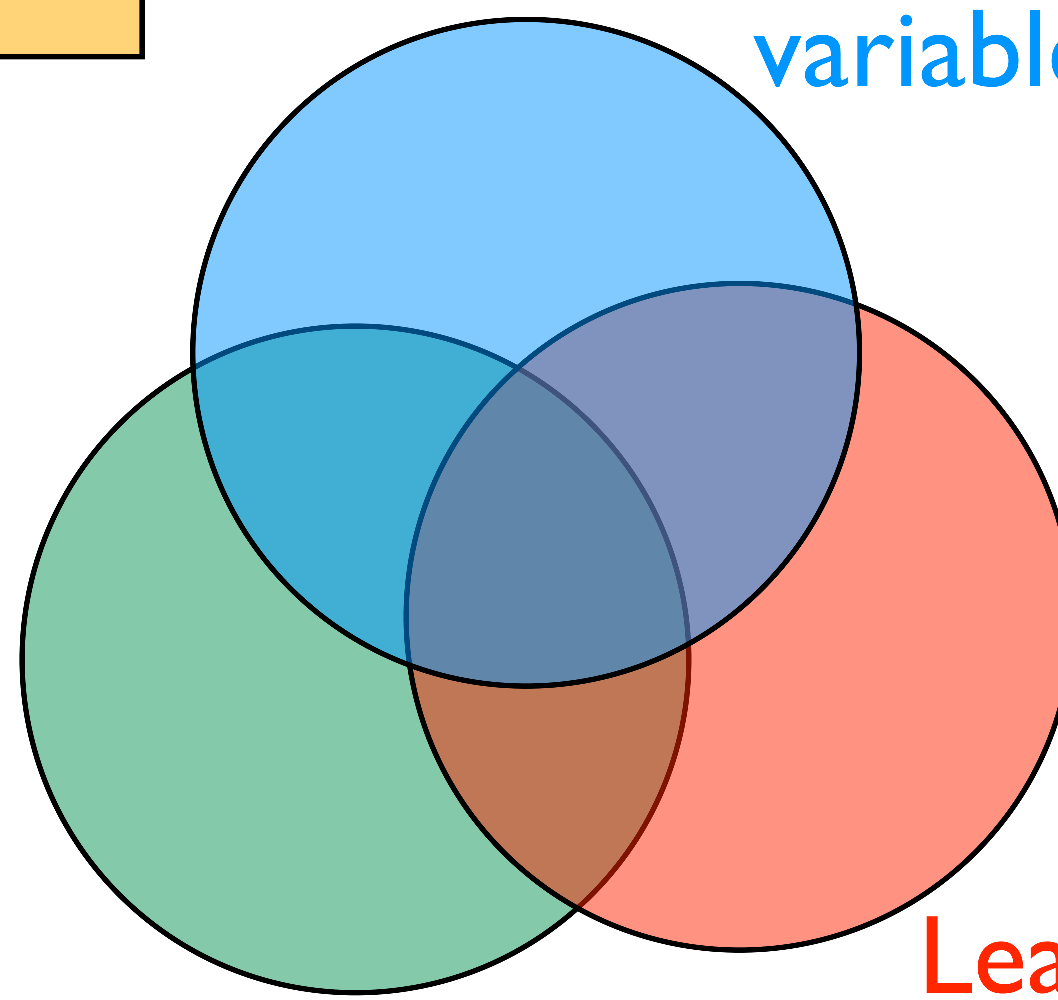
person	city
ann	london
bob	york
eve	new york
tom	paris

cityIn

city	country
london	uk
york	uk
paris	usa

relational database

tuples as random variables



Learning

Example: Information Extraction

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

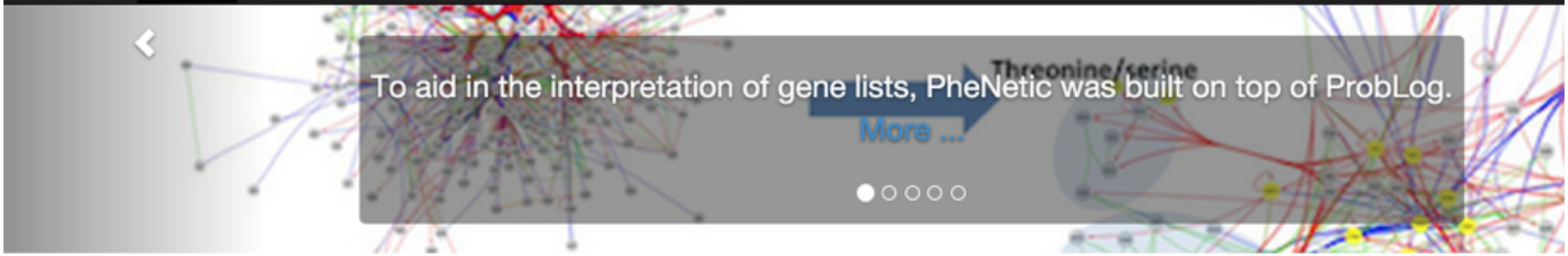
instances for many different relations

degree of certainty

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

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

ProbLog | Home | Download | Publications | Help | People | Interactive



Introduction.

Probabilistic logic programs are logic programs in which some of the facts are annotated with probabilities.

ProbLog is a tool that allows you to intuitively build programs that do not only encode **complex interactions** between a large sets of **heterogenous components** bu
uncertainties that are present in real-life situations.

The engine tackles several tasks such as computing the marginals given evidence and learning from (partial) interpretations. ProbLog is a suite of efficient algorithms
tasks. It is based on a conversion of the program and the queries and evidence to a weighted Boolean formula. This allows us to reduce the inference tasks to well-s
weighted model counting, which can be solved using state-of-the-art methods known from the graphical model and knowledge compilation literature.

The Language. Probabilistic Logic Programming.

ProbLog makes it easy to express complex, probabilistic models.

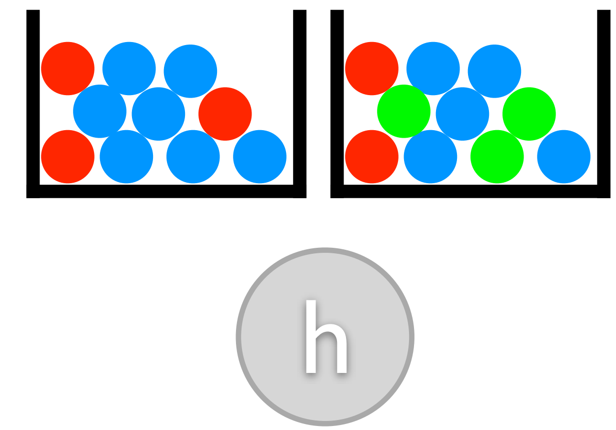
```
0.3::stress(X) :- person(X).
0.2::influences(X,Y) :- person(X), person(Y).

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

Some Probabilistic Programming Languages outside LP

- IBAL [Pfeffer 01]
- Figaro [Pfeffer 09]
- Church [Goodman et al 08]
- BLOG [Milch et al 05]
- Venture [Mansingha et al.]
- Anglican and Probabilistic-C [Wood et al].
- and many more appearing recently

Church 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)

```
(define heads (mem (lambda () (flip 0.4))))

(define color1 (mem (lambda () (if (flip 0.3) 'red 'blue))))

(define color2 (mem (lambda ()
                 (multinomial '(red green blue) '(0.2 0.3 0.5)))))

(define redball (or (equal? (color1) 'red) (equal? (color2) 'red)))

(define win1 (and (heads) redball))

(define win2 (equal? (color1) (color2)))

(define win (or win1 win2))
```

Probabilistic Programming Summary

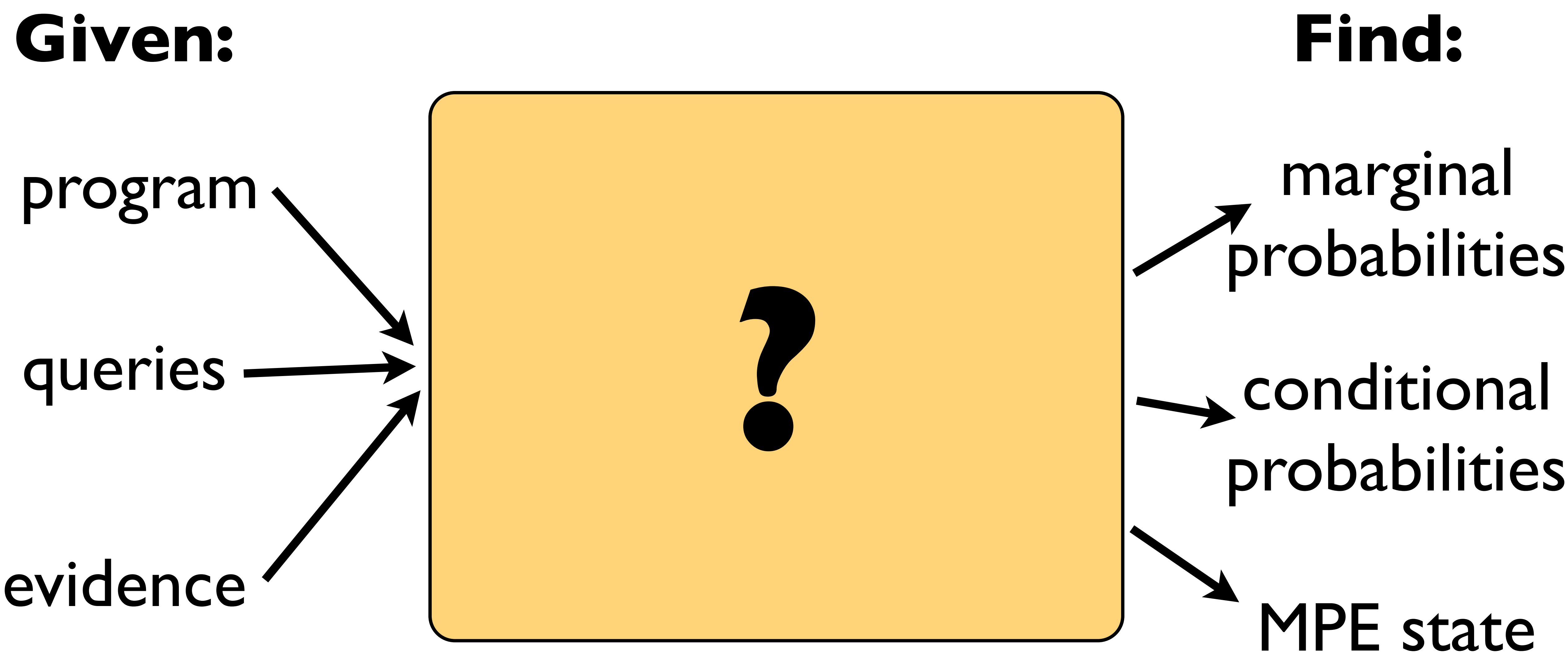
- Church: functional programming + random primitives
- probabilistic generative model
- stochastic memoization
- sampling
- increasing number of probabilistic programming languages using various underlying paradigms

Roadmap

- Modeling (ProbLog and Church, another representative of PP)
- **Inference**
- Learning
- Dynamics and Decisions

... with some detours on the way

Answering Questions



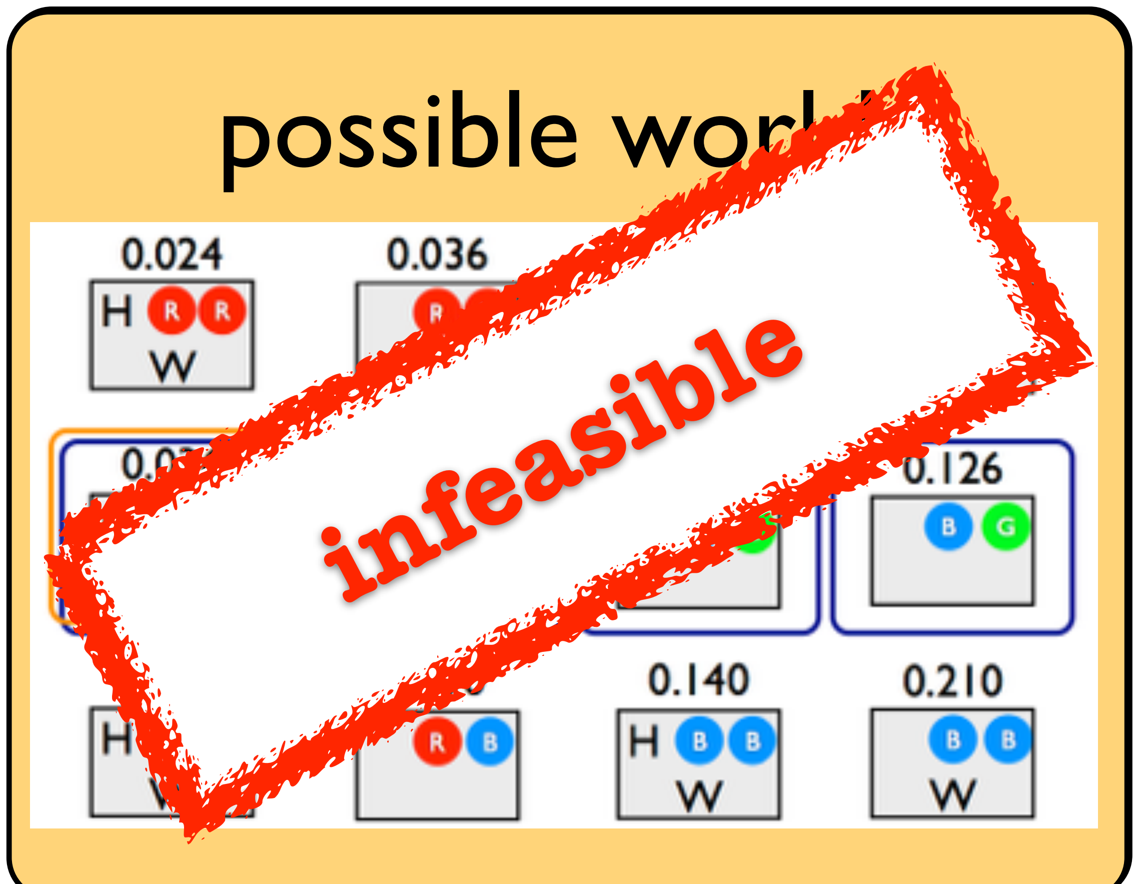
Answering Questions

Given:

program

queries

evidence



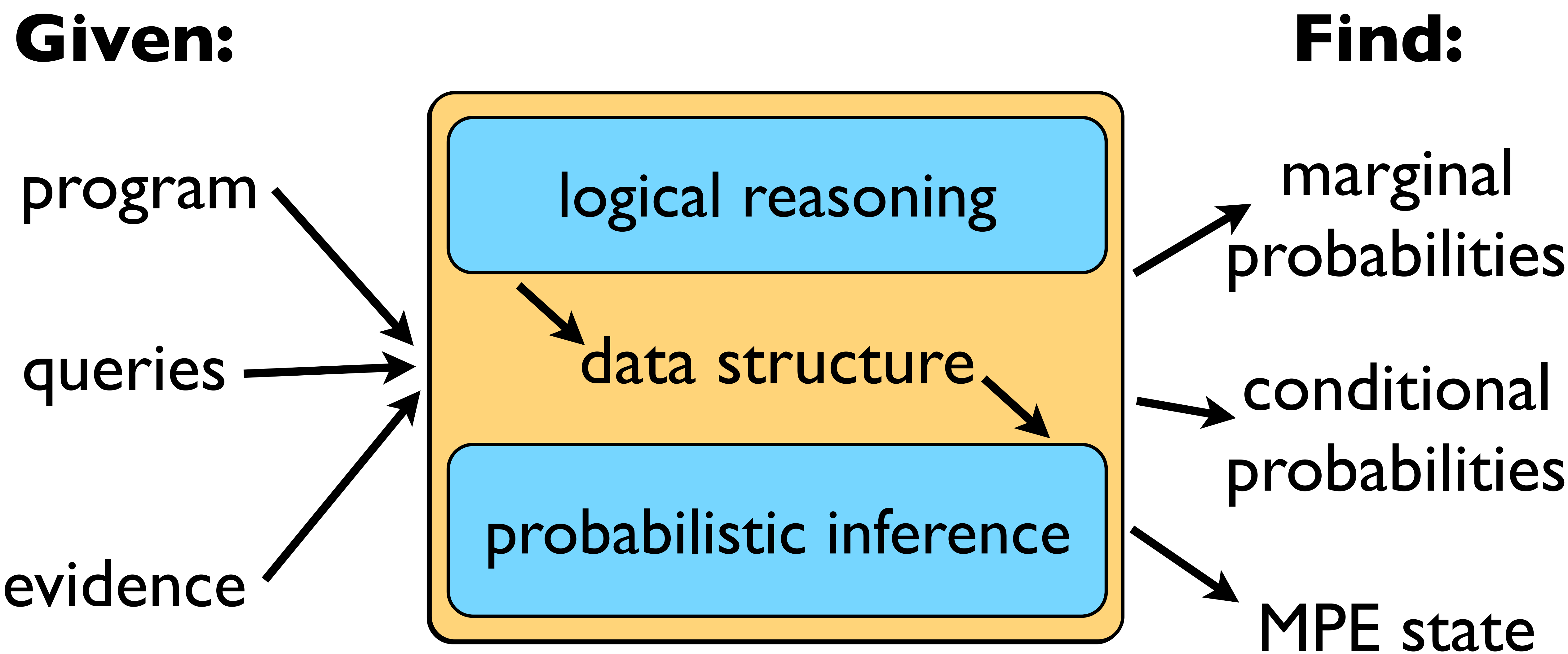
Find:

marginal probabilities

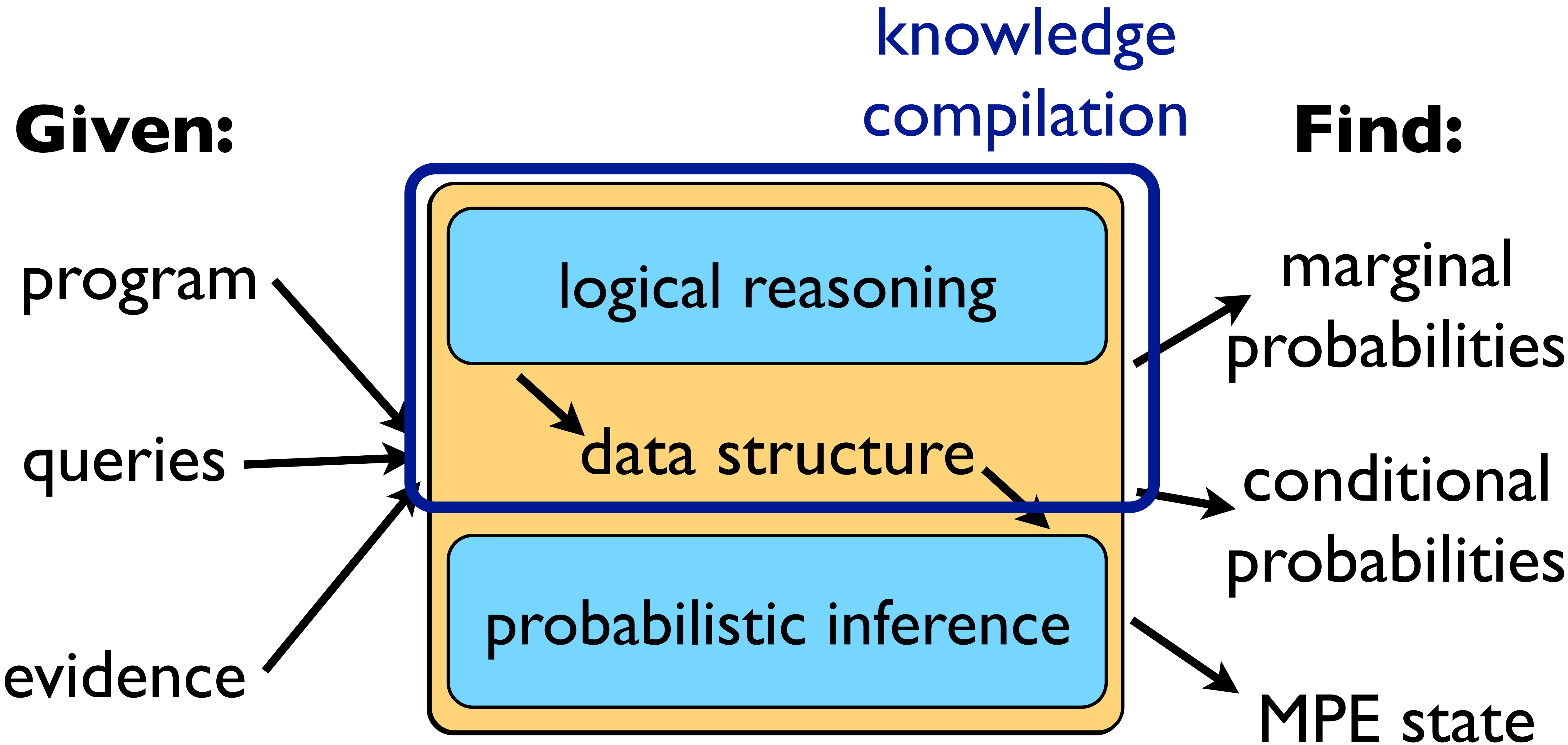
conditional probabilities

MPE state

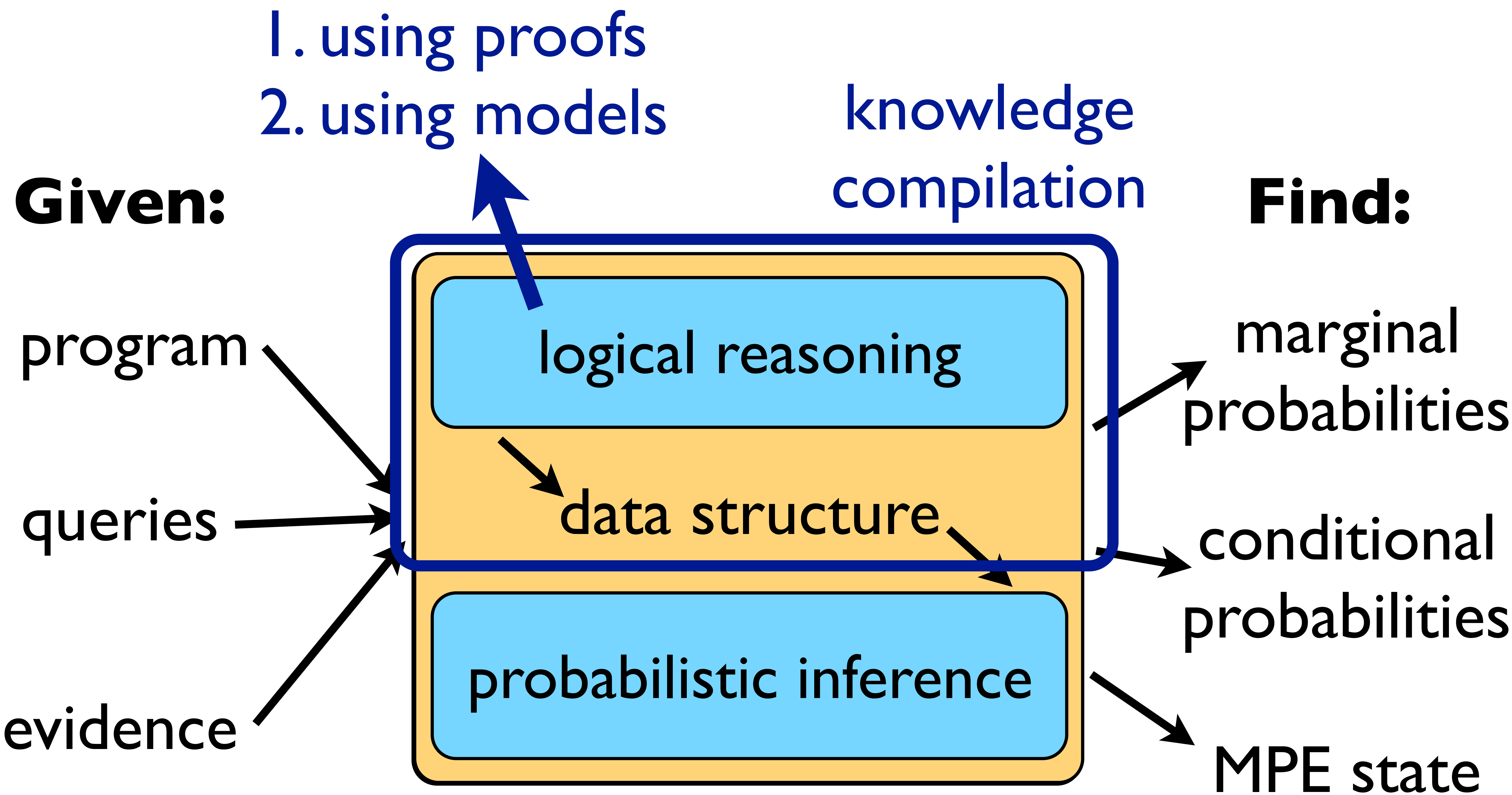
Answering Questions



Answering Questions



Answering Questions



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).
```

?- smokes(carl).

?- stress(carl).

Y=bob

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

?- smokes(bob).

?- stress(bob).

Y1=ann

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

?- smokes(ann).

?- stress(ann).

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

□

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

                                    ?- smokes(bob).

                                                       Y1=ann
?- stress(bob).                     ?- influences(Y1,bob),smokes(Y1).
   □
                                    ?- 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) & stress(bob)

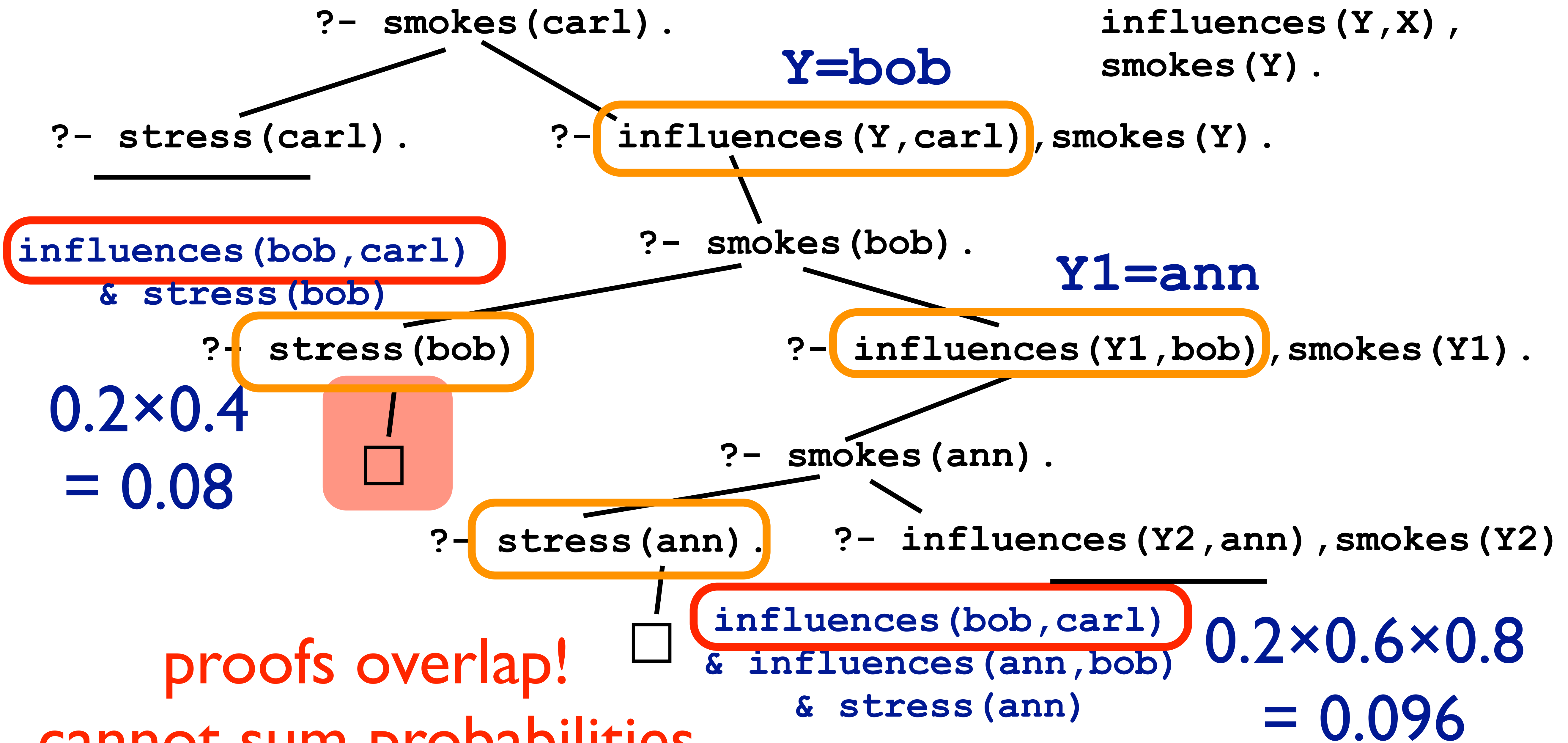
$0.2 \times 0.4 = 0.08$

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

Disjoint-Sum-Problem

possible worlds

```
infl(bob,carl) &   infl(ann,bob) &   st(ann) & \+st(bob)

infl(bob,carl) &   infl(ann,bob) &   st(ann) &   st(bob)

infl(bob,carl) & \+infl(ann,bob) &   st(ann) &   st(bob)

infl(bob,carl) &   infl(ann,bob) & \+st(ann) &   st(bob)

infl(bob,carl) & \+infl(ann,bob) & \+st(ann) &   st(bob)

...
```

Disjoint-Sum-Problem

possible worlds

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

```
infl(bob,carl) &   infl(ann,bob) &   st(ann) & \+st(bob)

infl(bob,carl) &   infl(ann,bob) &   st(ann) &   st(bob)

infl(bob,carl) & \+infl(ann,bob) &   st(ann) &   st(bob)

infl(bob,carl) &   infl(ann,bob) & \+st(ann) &   st(bob)

infl(bob,carl) & \+infl(ann,bob) & \+st(ann) &   st(bob)

...
```

Disjoint-Sum-Problem

possible worlds

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

```
infl(bob,carl) &   infl(ann,bob) &   st(ann) & \+st(bob)
infl(bob,carl) &   infl(ann,bob) &   st(ann) &   st(bob)
infl(bob,carl) & \+infl(ann,bob) &   st(ann) &   st(bob)
infl(bob,carl) &   infl(ann,bob) & \+st(ann) &   st(bob)
infl(bob,carl) & \+infl(ann,bob) & \+st(ann) &   st(bob)
...
```

influences(bob,carl) & stress(bob)

Disjoint-Sum-Problem

possible worlds

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

infl(bob,carl) &   infl(ann,bob) &   st(ann) & \+st(bob)
infl(bob,carl) &   infl(ann,bob) &   st(ann) &   st(bob)
infl(bob,carl) & \+infl(ann,bob) &   st(ann) &   st(bob)
infl(bob,carl) &   infl(ann,bob) & \+st(ann) &   st(bob)
infl(bob,carl) & \+infl(ann,bob) & \+st(ann) &   st(bob)
...        influences(bob,carl) & stress(bob)
```

sum of proof probabilities: 0.096+0.08 = 0.1760

Disjoint-Sum-Problem

possible worlds

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

```
infl(bob,carl) &   infl(ann,bob) &   st(ann) & \+st(bob)
infl(bob,carl) &   infl(ann,bob) &   st(ann) &   st(bob)
infl(bob,carl) & \+infl(ann,bob) &   st(ann) &   st(bob)
infl(bob,carl) &   infl(ann,bob) & \+st(ann) &   st(bob)
infl(bob,carl) & \+infl(ann,bob) & \+st(ann) &   st(bob)
...
```

0.0576
0.0384
0.0256
0.0096
0.0064

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

$\sum = 0.1376$

sum of proof probabilities: 0.096+0.08 = 0.1760

Disjoint-Sum-Problem

possible worlds

solution: knowledge compilation

```
infl(bob,carl) &   infl(ann,bob) &   st(ann) & \+st(bob)
infl(bob,carl) &   infl(ann,bob) &   st(ann) &   st(bob)
infl(bob,carl) & \+infl(ann,bob) &   st(ann) &   st(bob)
infl(bob,carl) &   infl(ann,bob) & \+st(ann) &   st(bob)
infl(bob,carl) & \+infl(ann,bob) & \+st(ann) &   st(bob)
...
```

0.0576
0.0384
0.0256
0.0096
0.0064

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

$\sum = 0.1376$

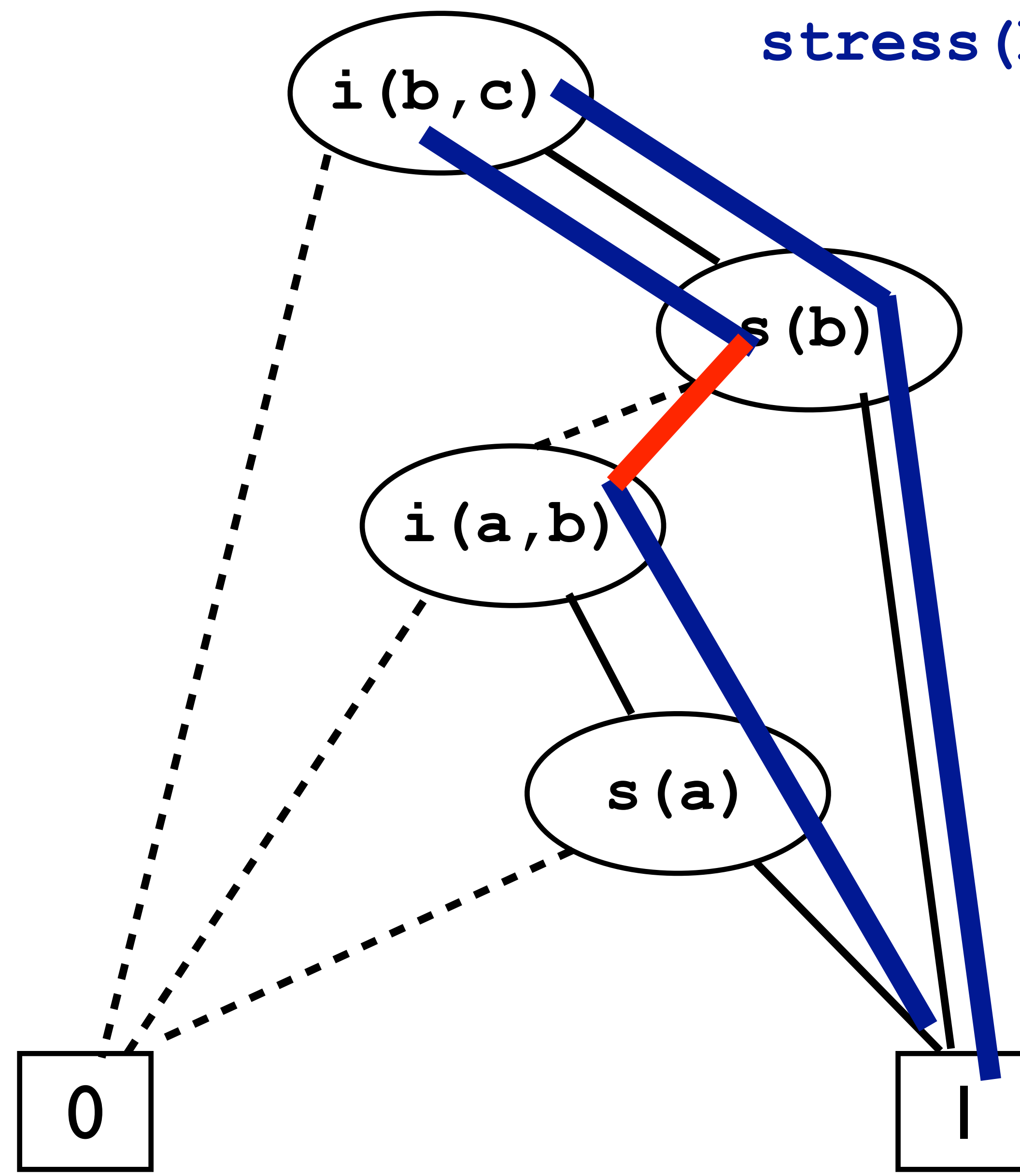
sum of proof probabilities: 0.096+0.08 = 0.1760

Binary Decision Diagrams

[Bryant 86]

- compact graphical representation of Boolean formula
- automatically disjoins proofs
- popular in many branches of CS

influences(bob,carl) & stress(bob)



influences(bob,carl) & influences(ann,bob) & stress(ann) & not stress(bob)

Markov Chain Monte Carlo (MCMC)

- Generate next sample by modifying current one
- Most common inference approach for PP languages such as Church, BLOG, ...
- Also considered for PRISM and ProbLog

Key challenges:
- how to propose next sample
- how to handle evidence

Roadmap

- Modeling (ProbLog and Church, another representative of PP)
- Inference
- **Learning**
- Dynamics and Decisions

... with some detours on the way

Parameter Learning

e.g., webpage classification model

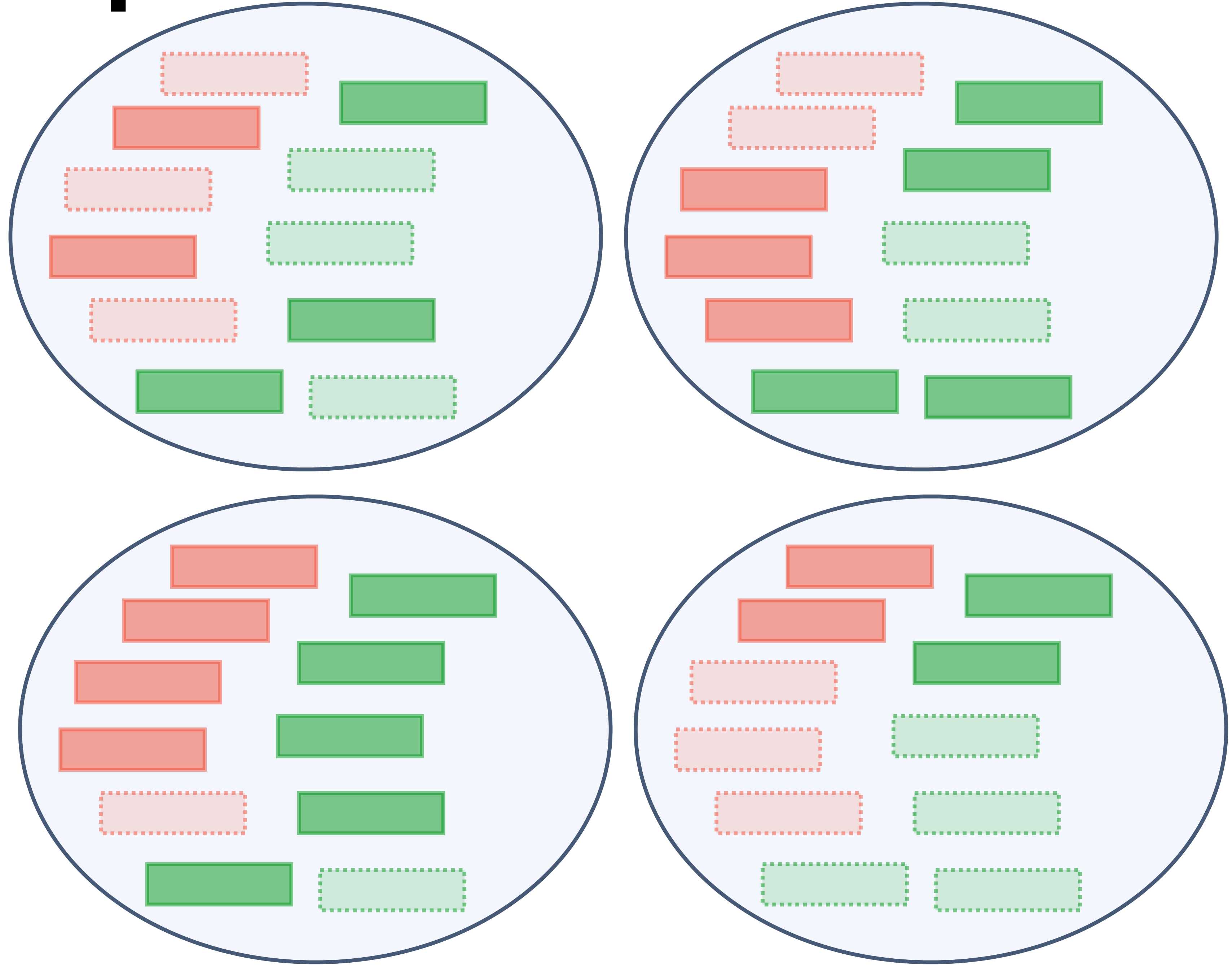
for each *CLASS1*, *CLASS2* and each *WORD*

?? :: link_class(Source,Target,*CLASS1*,*CLASS2*).
?? :: word_class(*WORD*,*CLASS*).

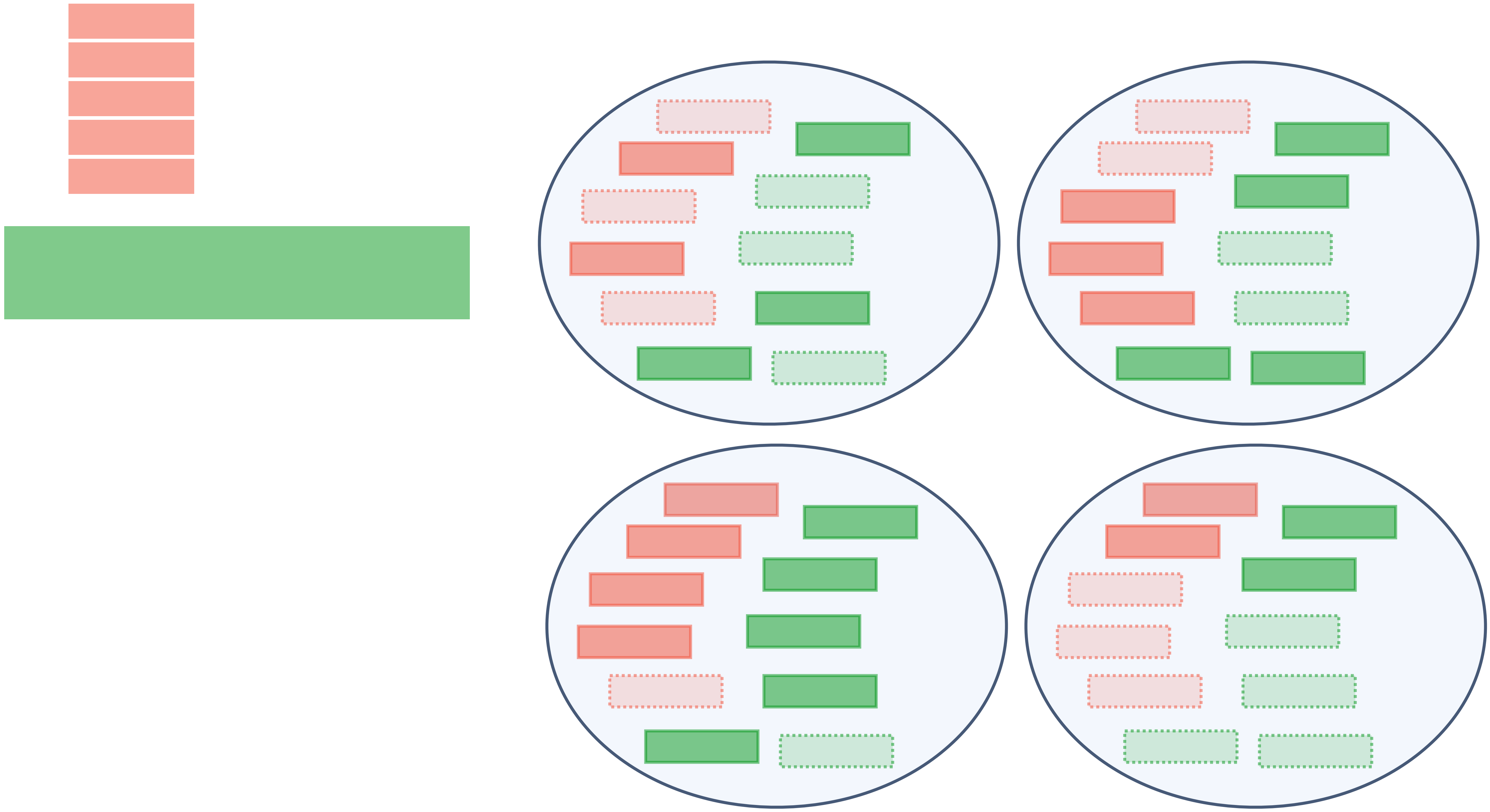
```
class(Page,C)    :-   has_word(Page,W), word_class(W,C).

class(Page,C)          :-   links_to(OtherPage,Page),
    class(OtherPage,OtherClass),
                                   link_class(OtherPage,Page,OtherClass,C).
```

Sampling Interpretations

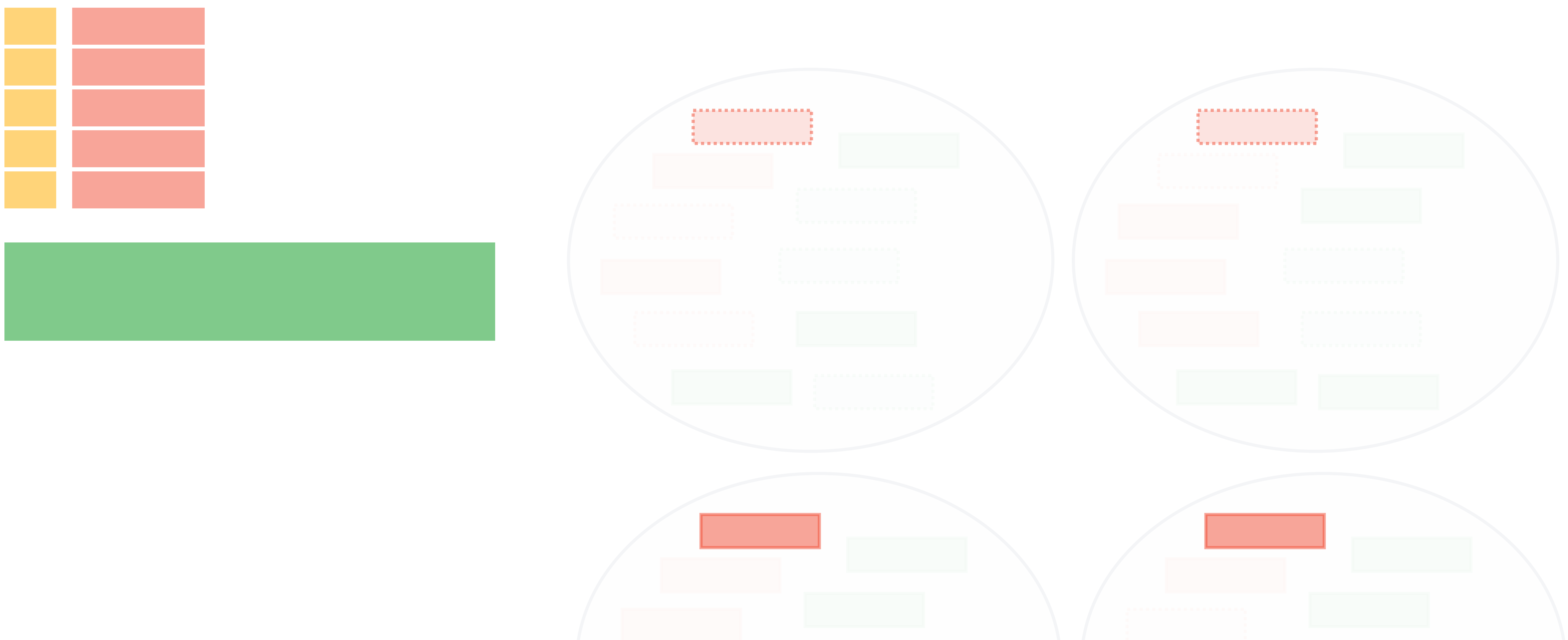
Sampling Interpretations



Parameter Estimation

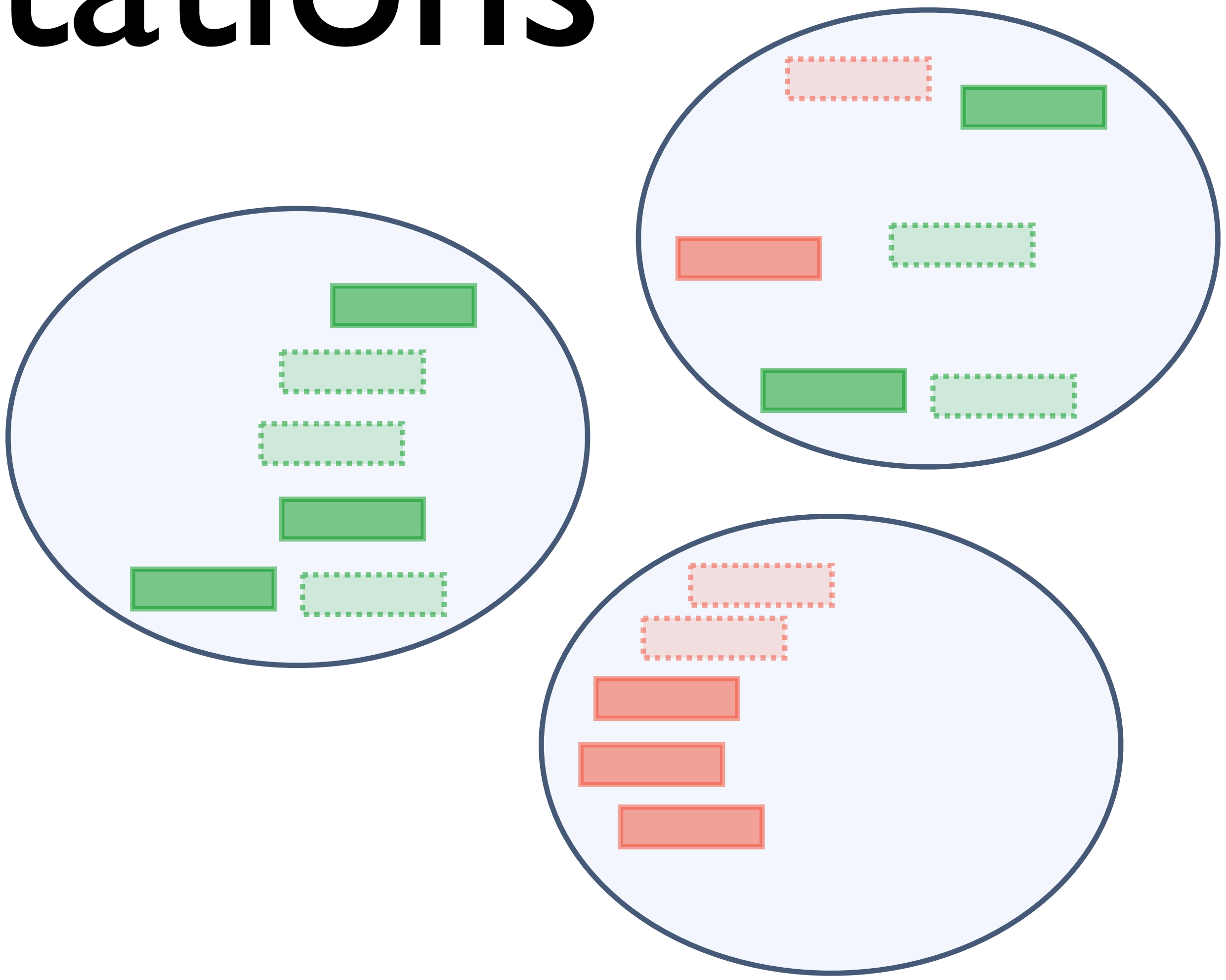


Parameter Estimation



$$p(\textbf{fact}) = \frac{\text{count}(\textbf{fact} \text{ is true})}{\text{Number of interpretations}}$$

Learning from partial interpretations



- Not all facts observed
- Soft-EM
- use **expected count** instead of **count**
- **P(Q |E) -- conditional queries !**

 [Gutmann et al, ECML 11; Fierens et al, TPLP 14]

Rule learning — 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

Adaptation of standard rule learning and
inductive logic programming setting
[De Raedt et al IJCAI 15]

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).

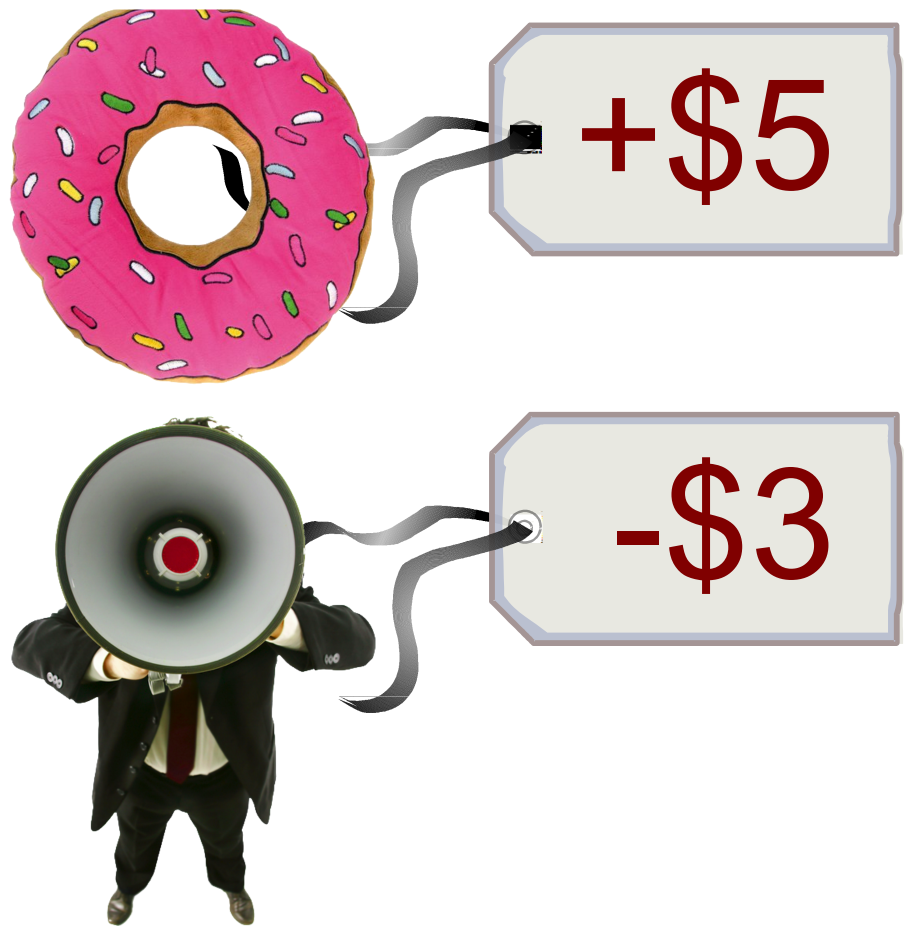
0.9375::athleteplaysforteam(A,B)	←	athleteledsportsteam(A,B).
0.9675::athleteplaysforteam(A,B)	←	athleteledsportsteam(A,V1), teamplaysagainstteam(B,V1).
0.9375::athleteplaysforteam(A,B)	←	athleteplayssport(A,V1), teamplayssport(B,V1).
0.5109::athleteplaysforteam(A,B)	←	athleteplaysinleague(A,V1), teamplaysinleague(B,V1).
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).

Roadmap

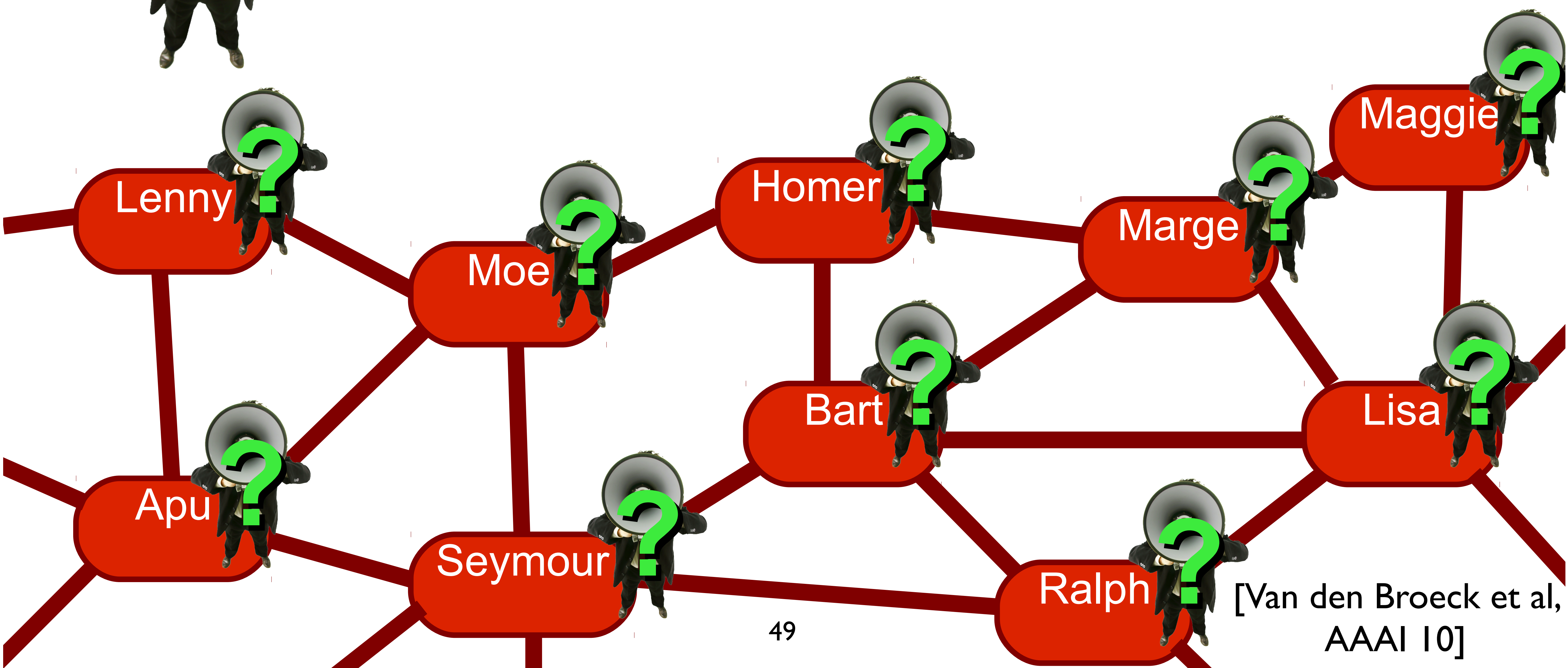
- Modeling (ProbLog and Church, another representative of PP)
- Inference
- Learning
- **Dynamics and Decisions**

... with some detours on the way

Viral Marketing

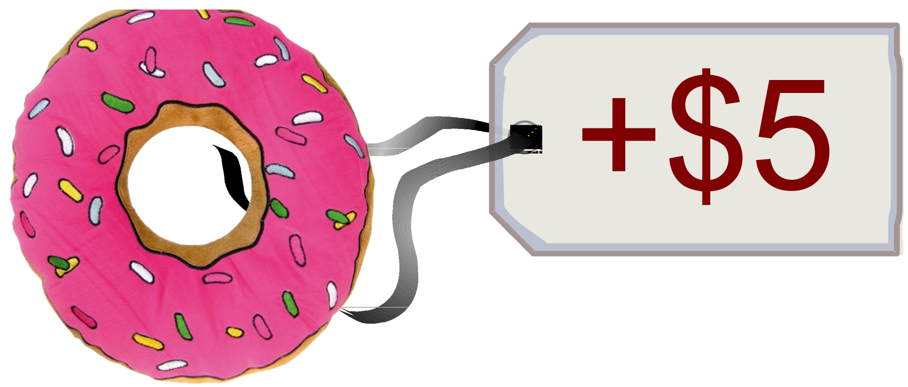


Which advertising strategy maximizes expected profit?

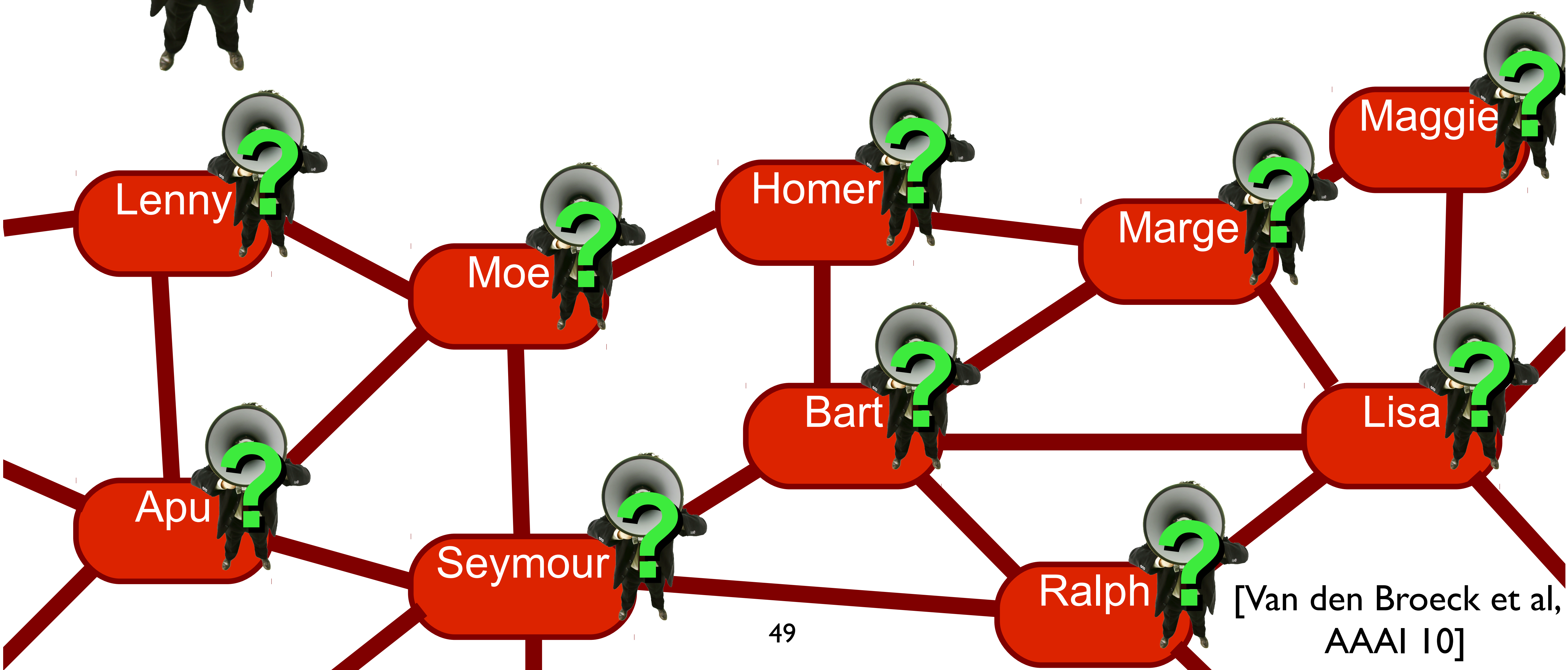


[Van den Broeck et al, AAAI 10]

Viral Marketing

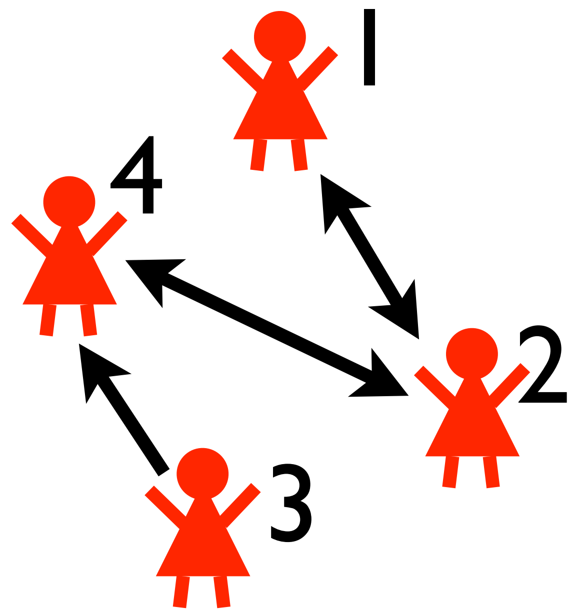


decide truth values of some atoms



[Van den Broeck et al, AAAI 10]

DTProbLog

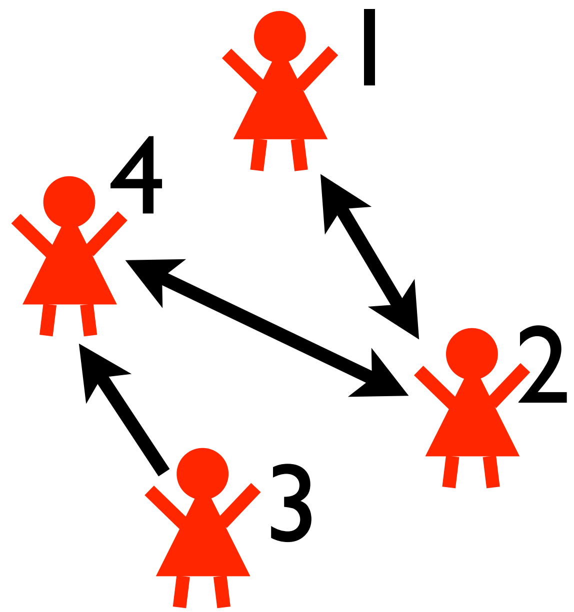



```
person(1).
person(2).
person(3).
person(4).

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

DTProbLog

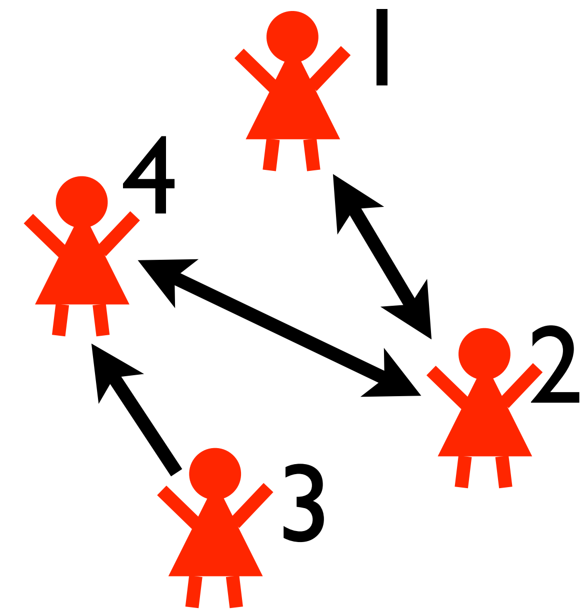
```
? :: marketed(P) :- person(P).
```

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).
```

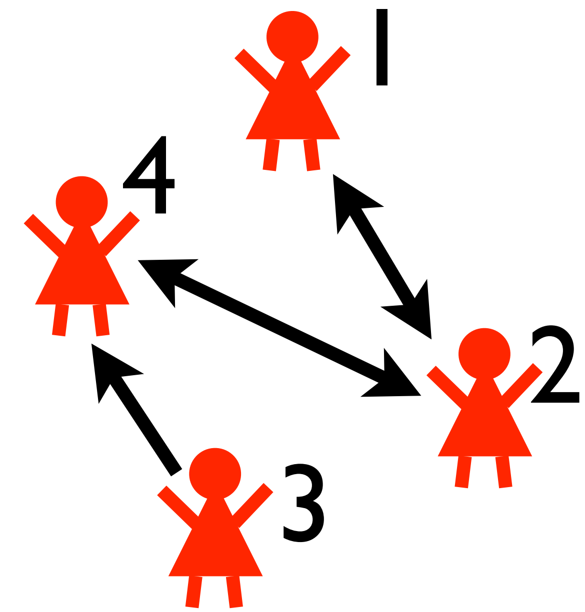
```
person(1).
person(2).
person(3).
person(4).

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

probabilistic facts
+ logical rules

DTProbLog

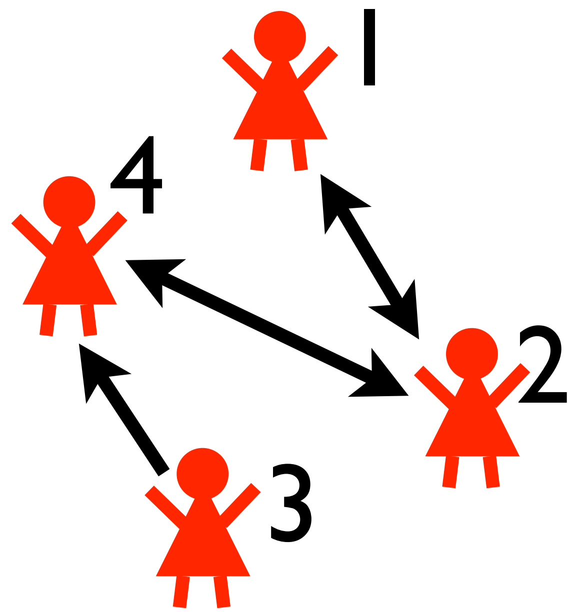
```
? :: marketed(P) :- person(P).

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).
```

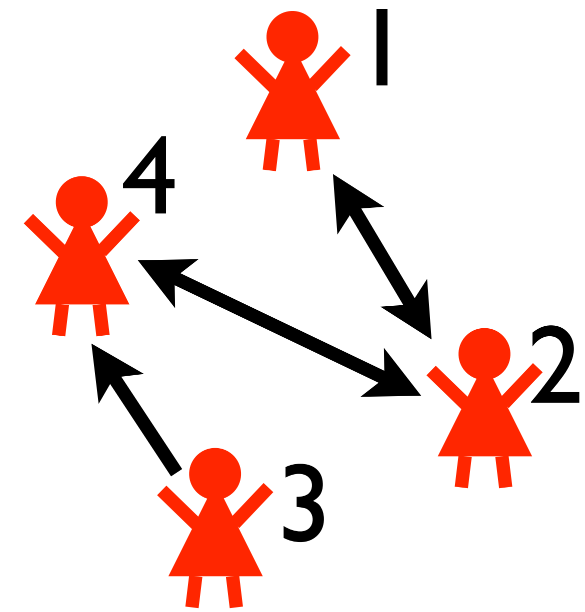
DTProbLog




```
? :: 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).
```

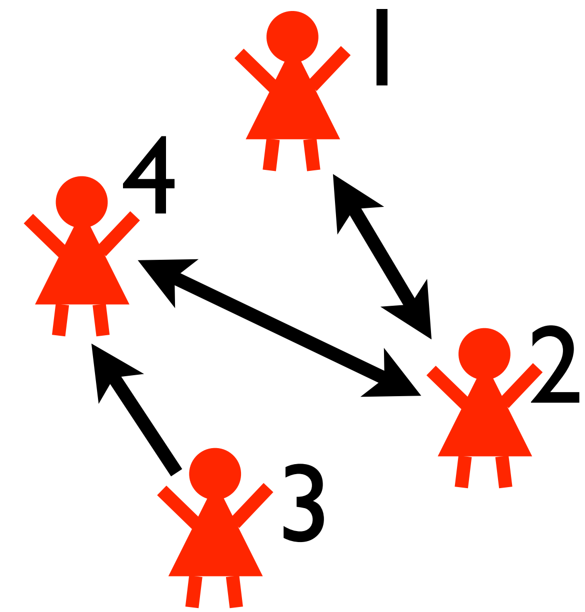
DTProbLog




```
? :: 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

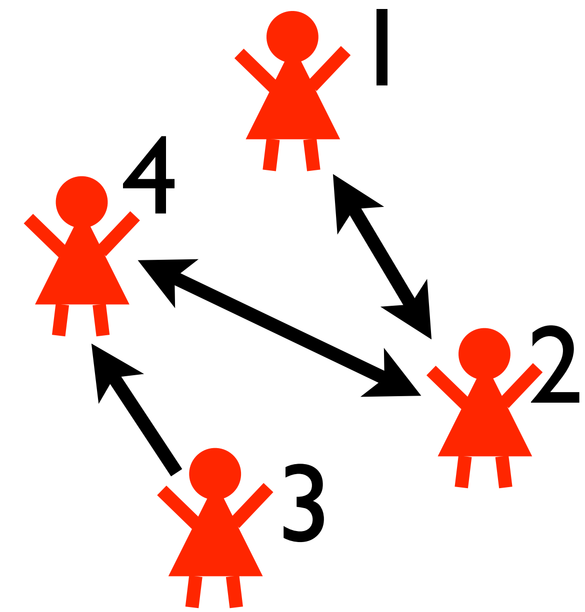



```
? :: marketed(P) :- person(P).

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

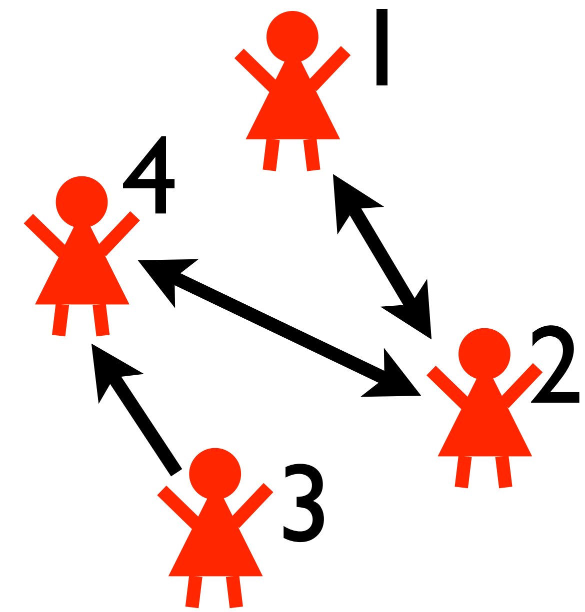



```
? :: marketed(P) :- person(P).

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

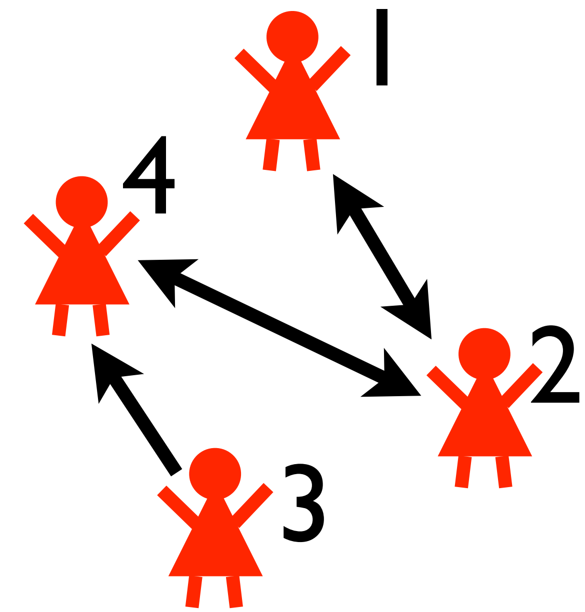



```
? :: marketed(P) :- person(P).

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).
```

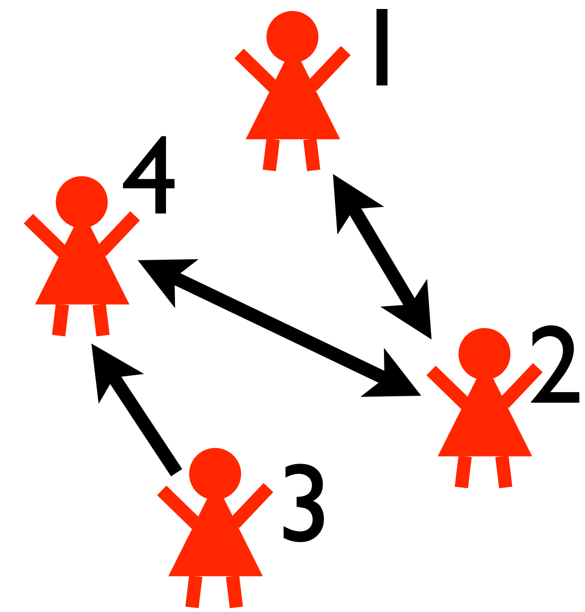
```
person(1).
person(2).
person(3).
person(4).

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).
```

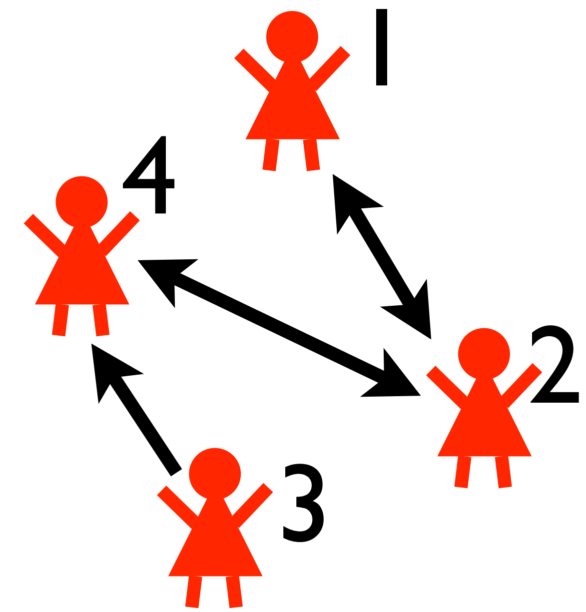
utility = −3 + −3 + 5 + 5 = 4

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

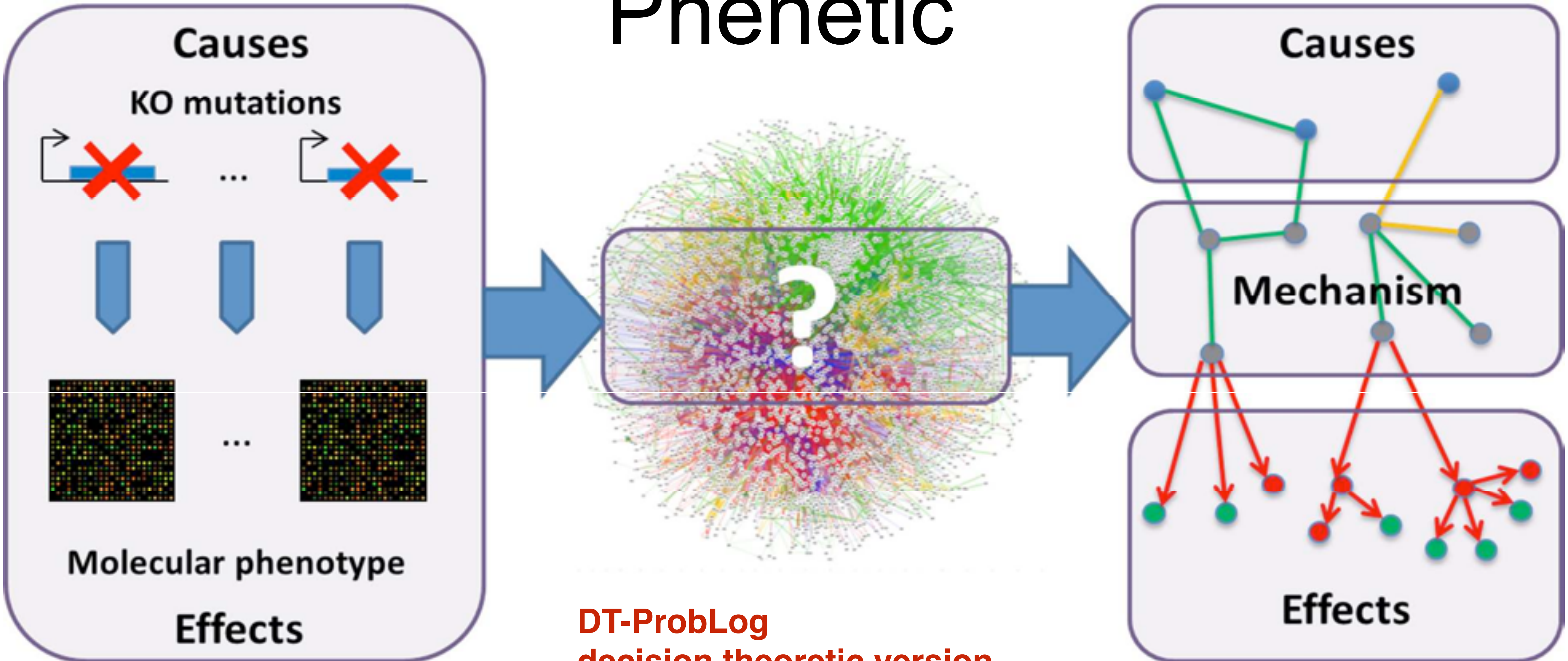



```
? :: marketed(P) :- person(P).

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

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

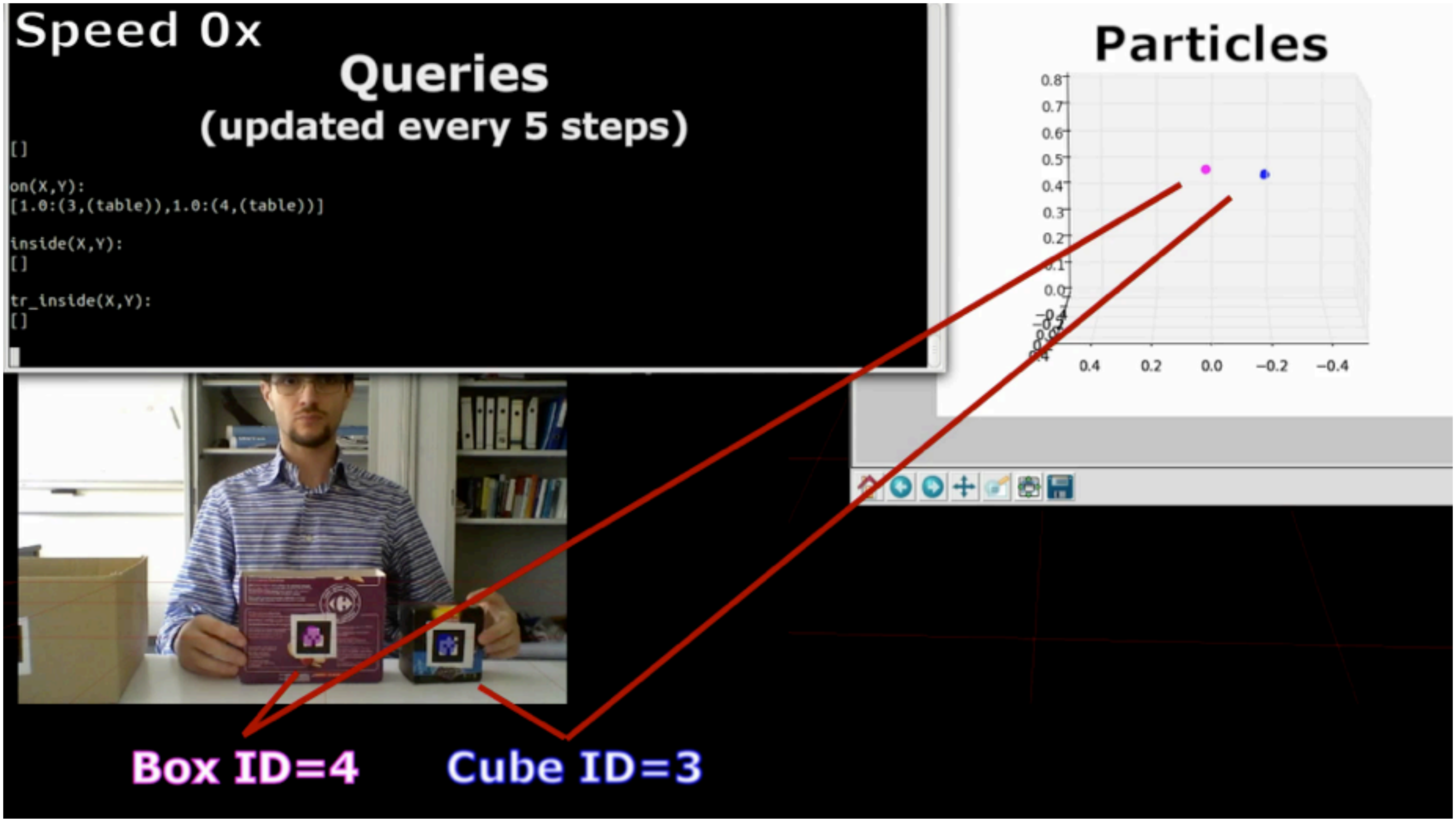
- Goal: connect causes to effects through common subnetwork
  - = Find mechanism
- Techniques:
  - DTProbLog [Van den Broeck]
  - Approximate inference

Can we find the mechanism connecting causes to effects?

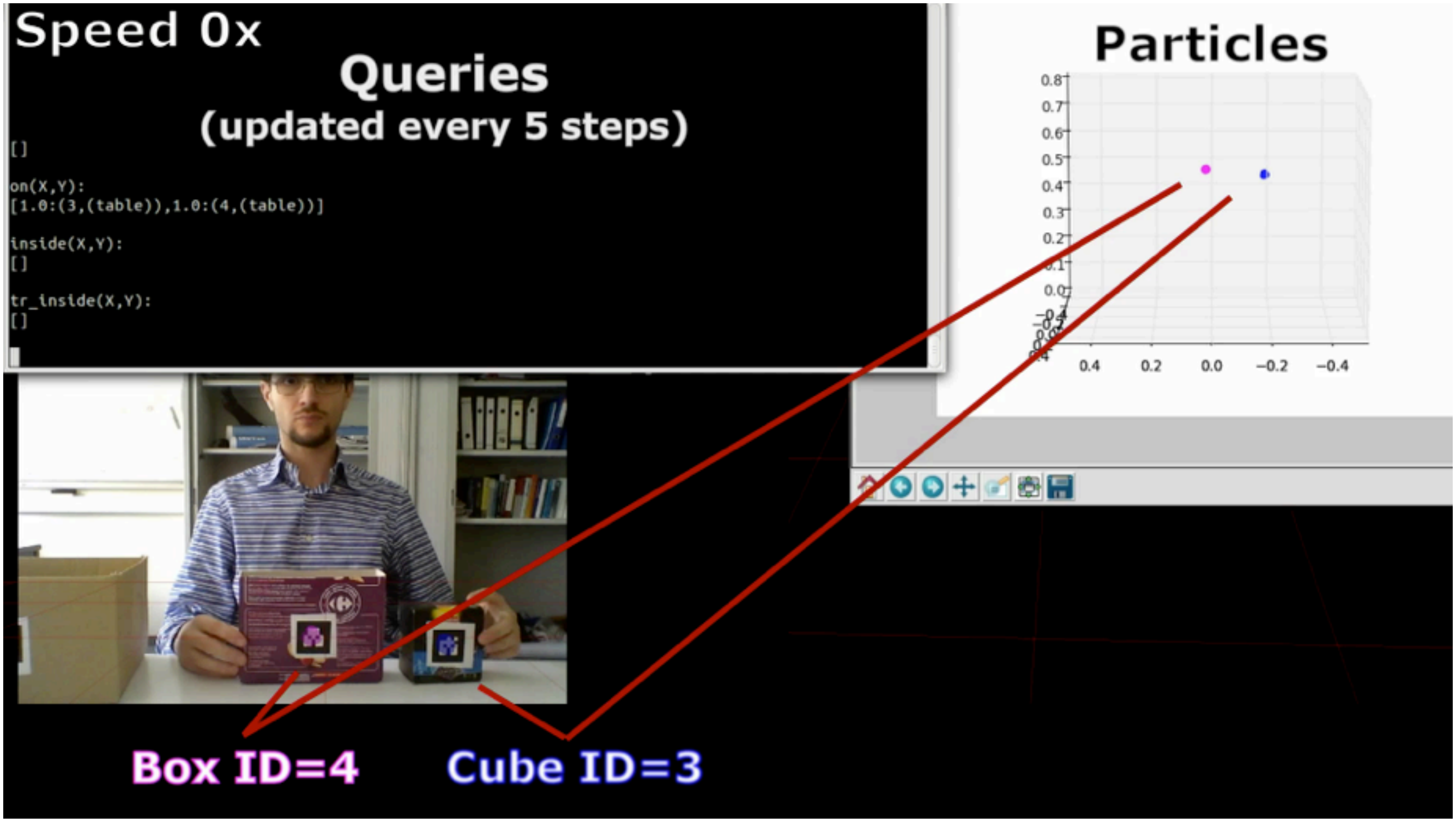
[De Maeyer et al., Molecular Biosystems 13, NAR 15]

Distributional Clauses (DC)

- A probabilistic logic language
- Logic (relational): a template to define random variables
- MDP representation in Dynamic DC:
  - Transition model: $\text{Head}_{t+1} \sim \text{Distribution} \leftarrow \text{Conditions}_t$
  - Applicable actions: $\text{applicable(Action)}_t \leftarrow \text{Conditions}_t$
  - Reward: $\text{reward(R)}_t \leftarrow \text{Conditions}_t$
  - Terminal state: $\text{stop}_t \leftarrow \text{Conditions}_t$
- The state can contain:
  - Discrete, continuous variables
  - The number of variables in the state can change over time


Speed 0x
Queries
(updated every 5 steps)
[]
on(X,Y):
[1.0:(3,(table)),1.0:(4,(table))]
inside(X,Y):
[]
tr_inside(X,Y):
[]
Particles
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.0
0.4
0.2
0.0
−0.2
−0.4
Box ID=4
Cube ID=3

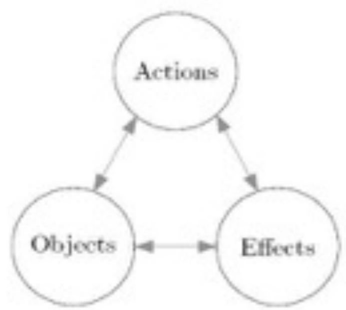
IROS 13


Speed 0x
Queries
(updated every 5 steps)
[]
on(X,Y):
[1.0:(3,(table)),1.0:(4,(table))]
inside(X,Y):
[]
tr_inside(X,Y):
[]
Particles
Box ID=4
Cube ID=3

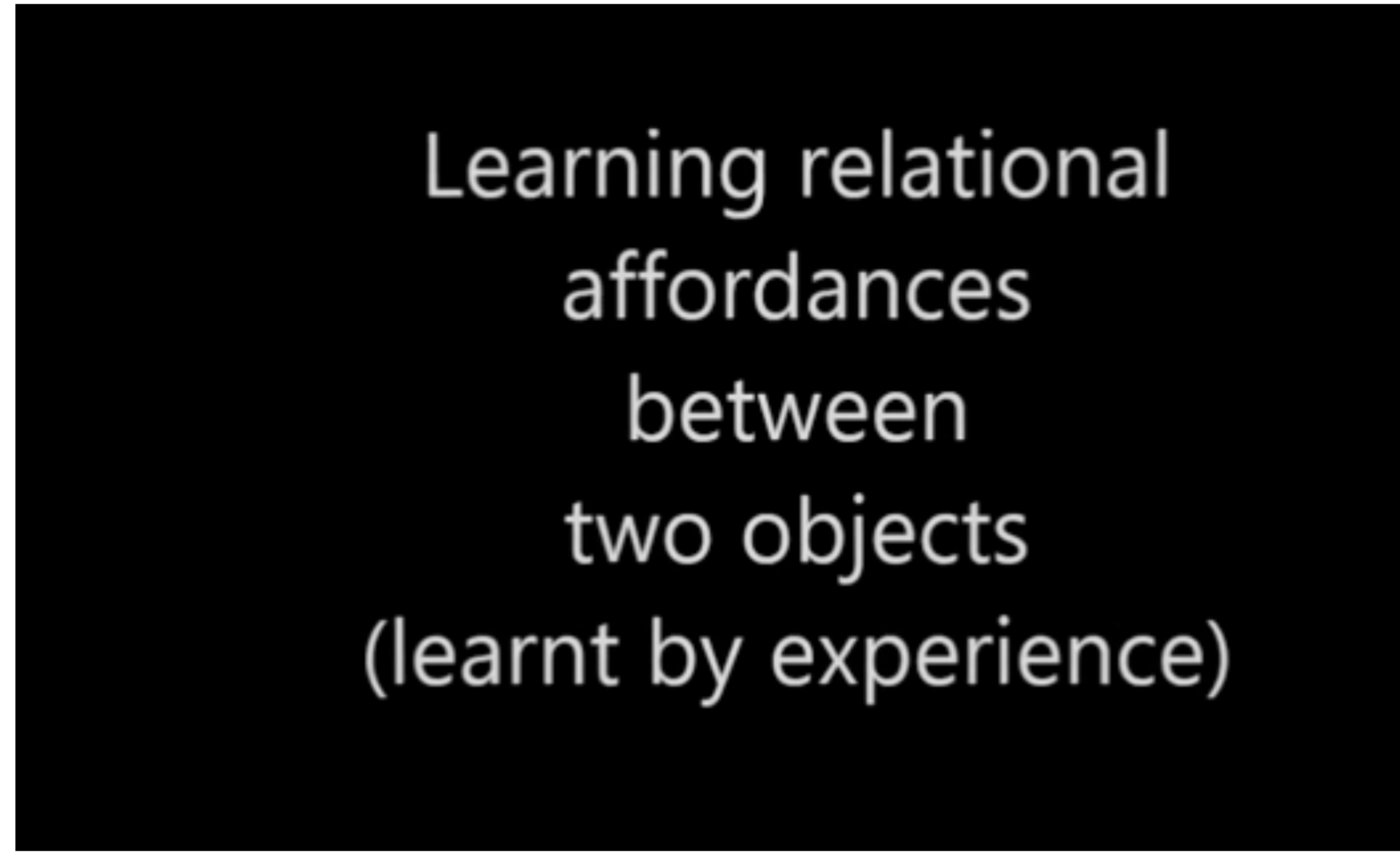
IROS 13

Learning relational affordances

Learn probabilistic model



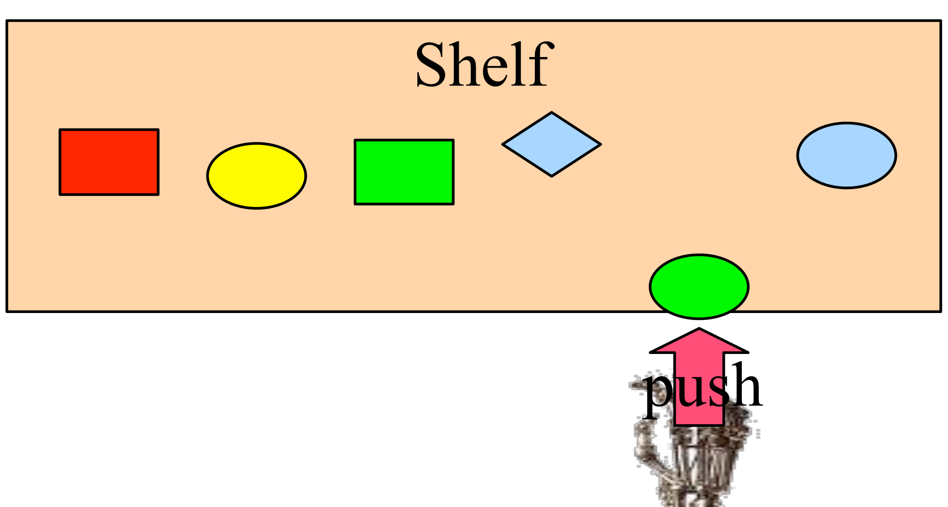
Inputs	Outputs	Function
(O, A)	E	Effect prediction
(O, E)	A	Action recognition/planning
(A, E)	O	Object recognition/selection

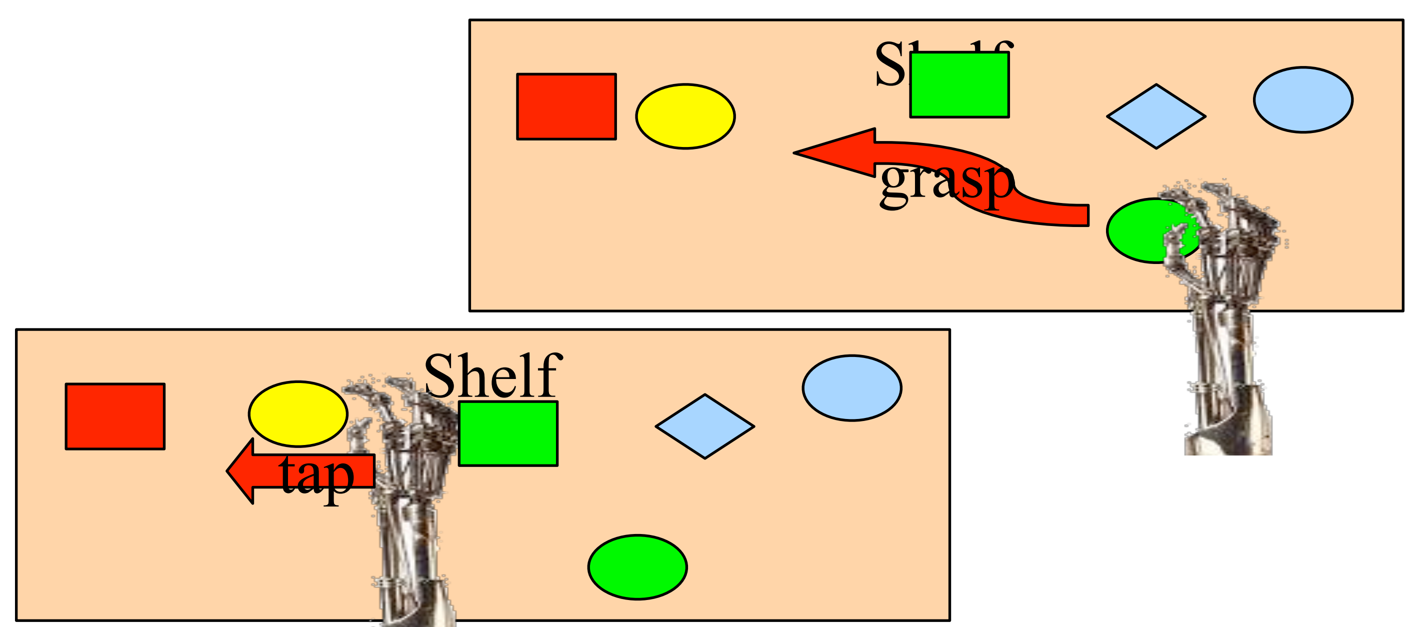


From two object interactions

Generalize to N

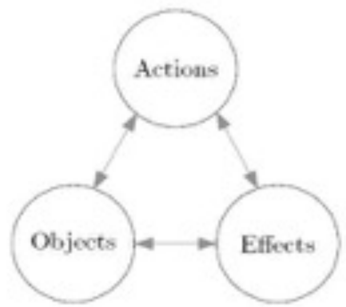
Moldovan et al. ICRA 12, 13, 14, PhD 15



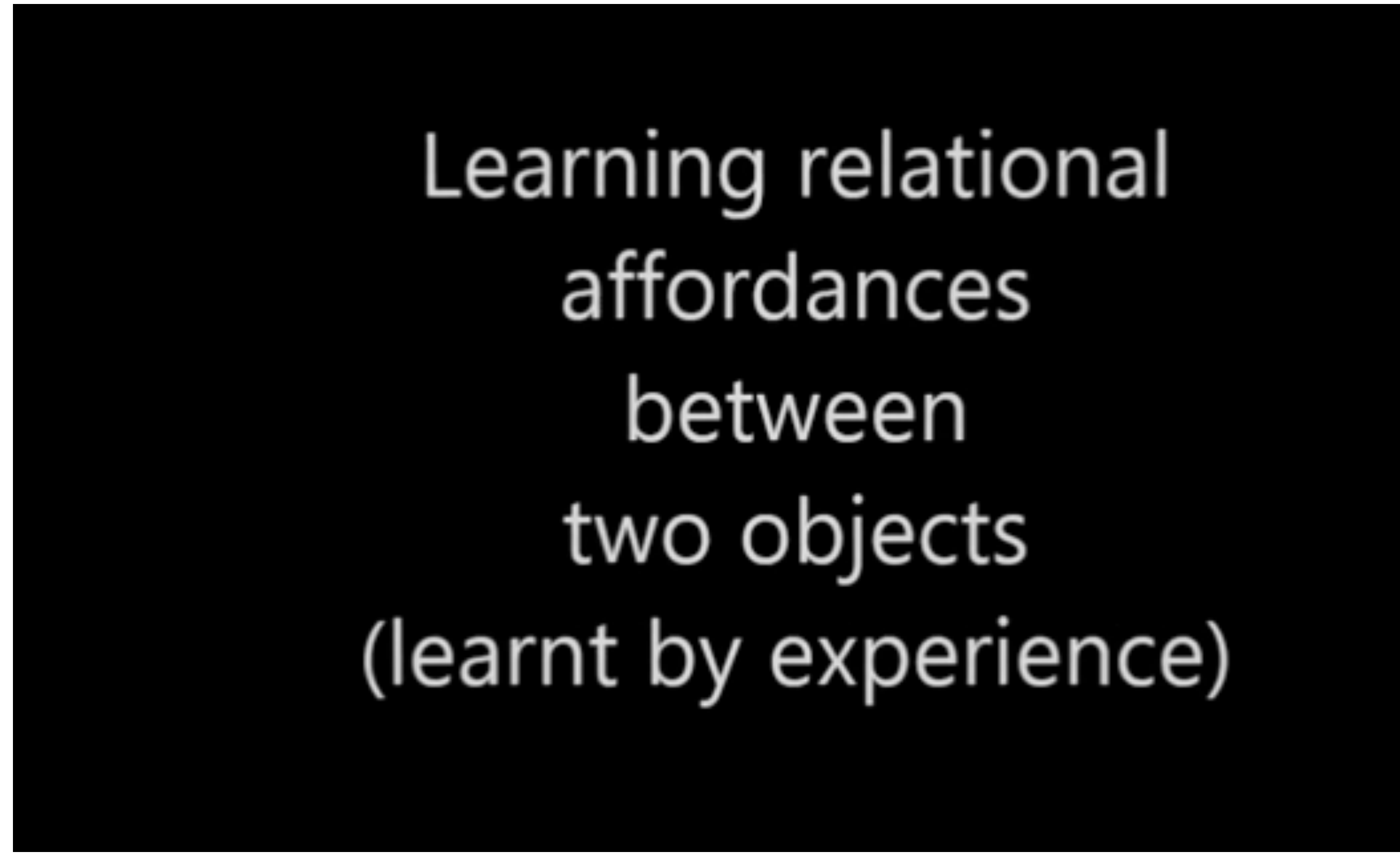


Learning relational affordances

Learn probabilistic model



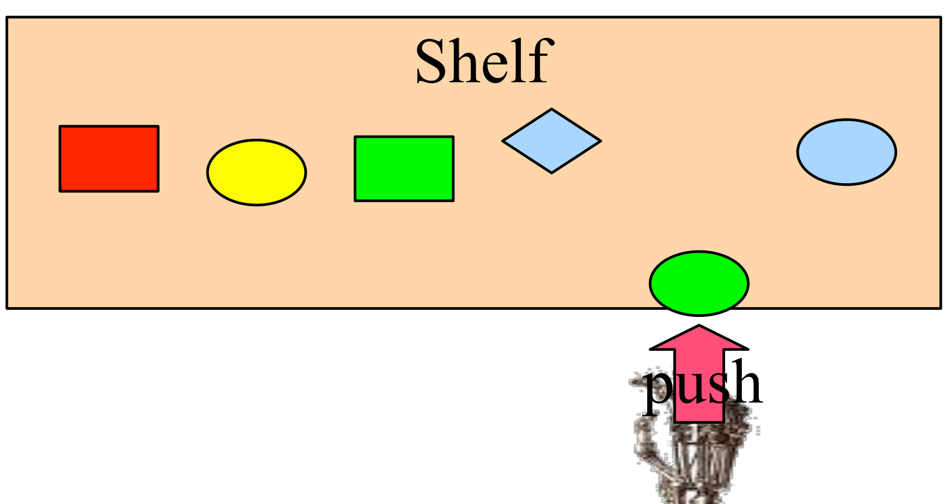
Inputs	Outputs	Function
(O, A)	E	Effect prediction
(O, E)	A	Action recognition/planning
(A, E)	O	Object recognition/selection

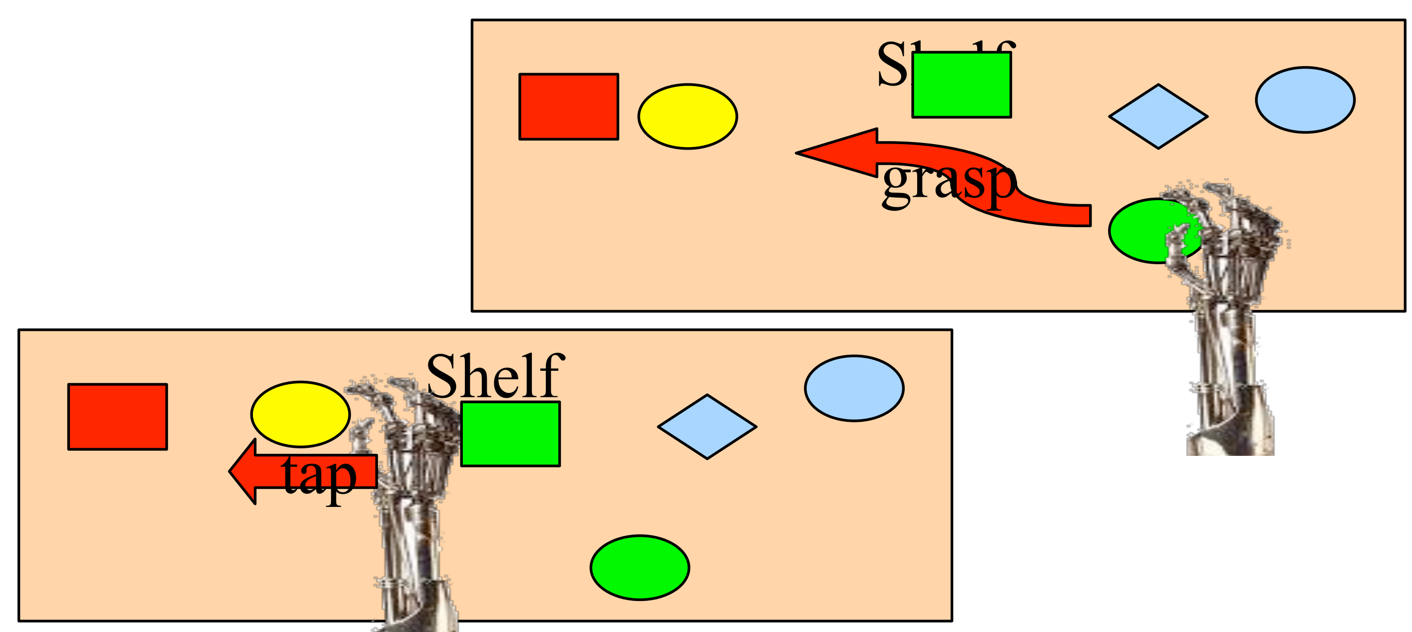


From two object interactions

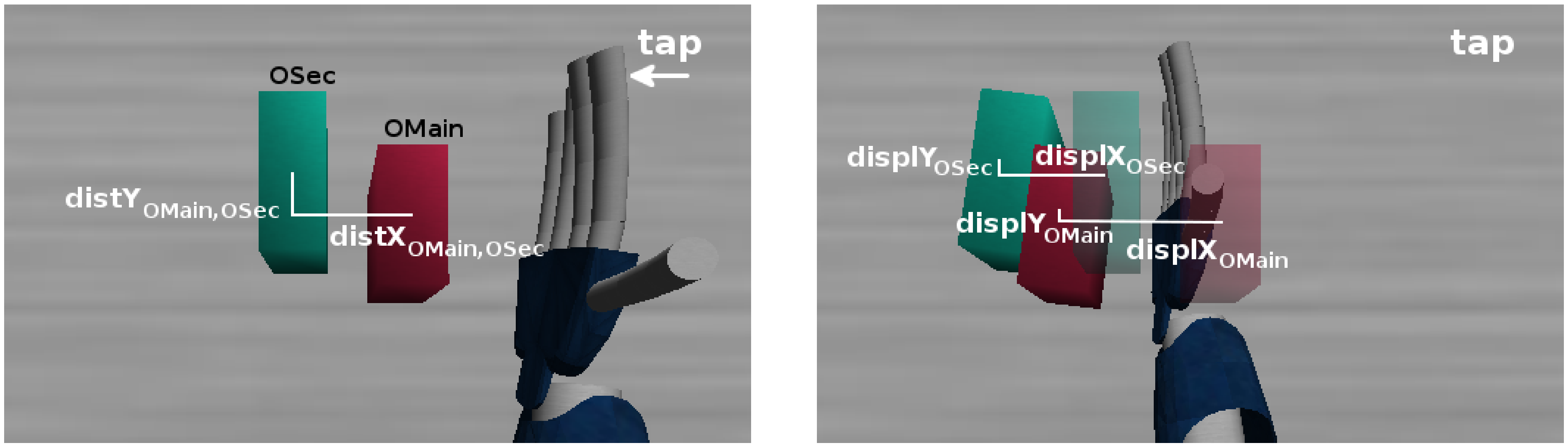
Generalize to N

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





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

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

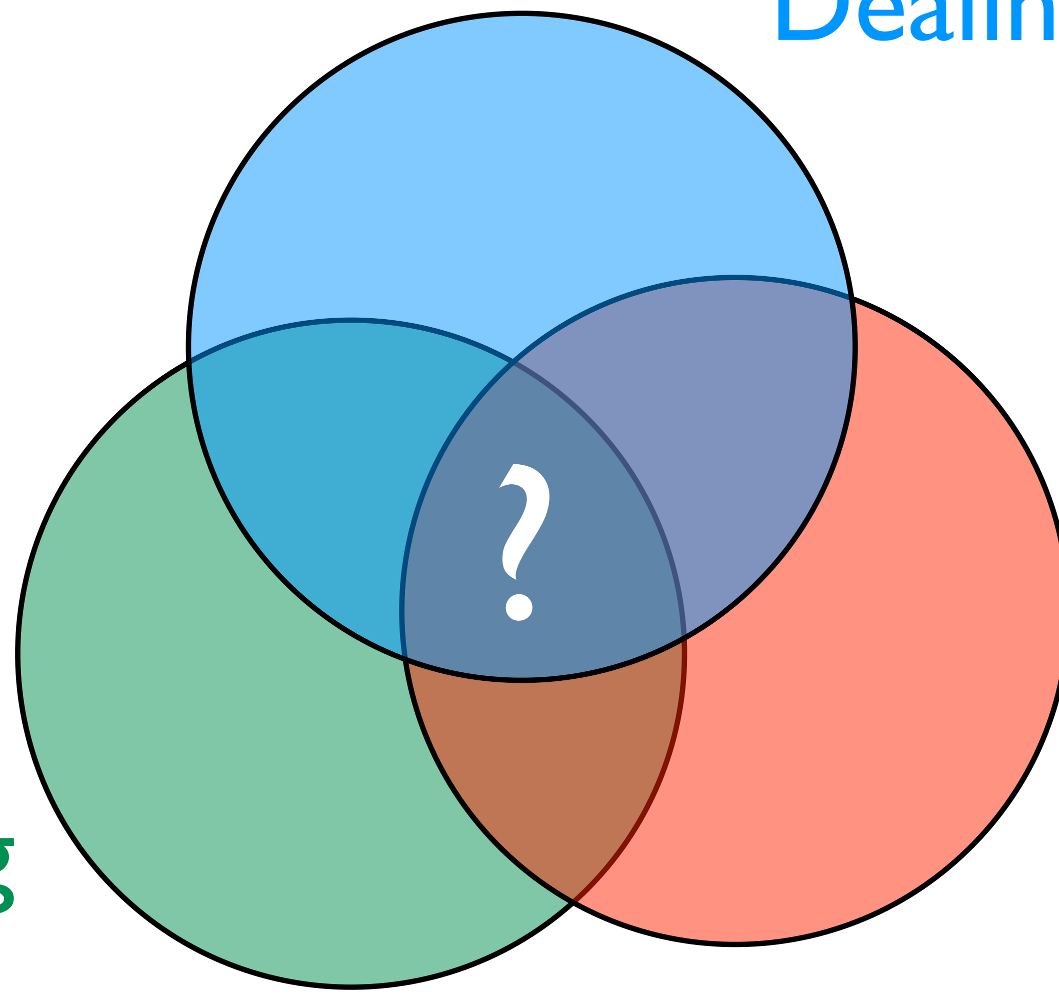
A key question in AI:

Dealing with uncertainty

- probability theory
- graphical models
- ...

Reasoning with relational data

- logic
- databases
- programming
- ...



Learning

- parameters
- structure

Statistical relational learning
Probabilistic programming, ...

A key question in AI:

Dealing with uncertainty
- probability theory
- dels

Reas
relat
- log
- da
- pr
- ...

- Many languages, systems, applications, ...
- not yet a technology ! but a lot of progress
- and a lot more to do !
- … excellent area for PhDs …

s
structure

Statistical relational learning
Probabilistic programming, ...

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