

Capita Selecta AI: Probabilistic Programming

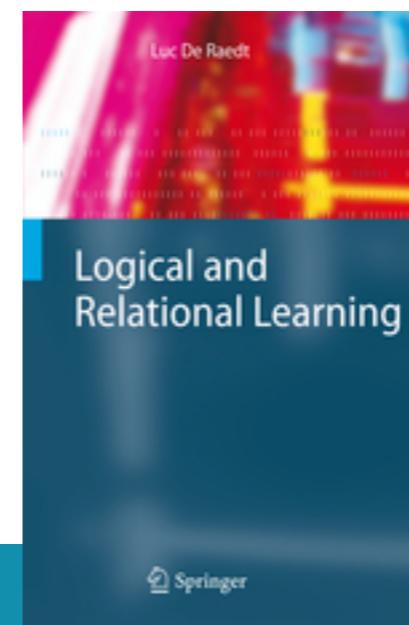
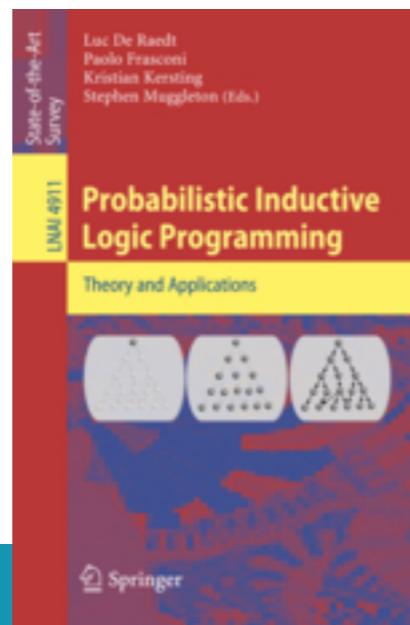
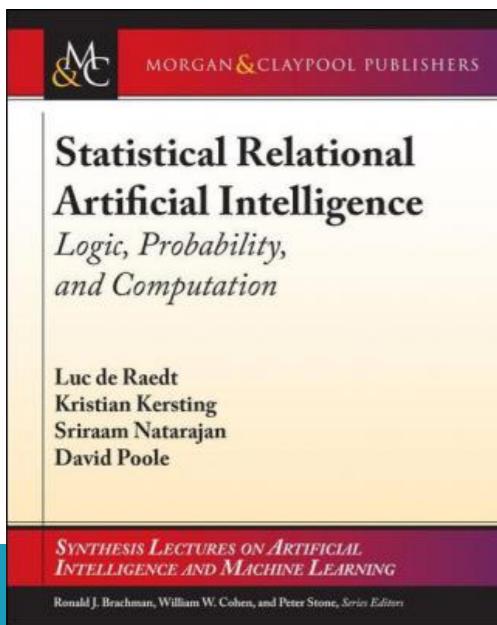
Wannes Meert

18/10/2017

Slides courtesy of Angelika Kimmig, Luc De Raedt and Guy Van den Broeck

What?

- Probabilistic (logic) learning & related fields
- A core research topic in the DTAI research group
- Lectures
 - One lectures in first semester
 - Second (interactive) lecture in second semester
 - Additional Q&A sessions possible
- Assignment: Inference for ProbLog hands-on
 - Evaluation based on short written report



Today

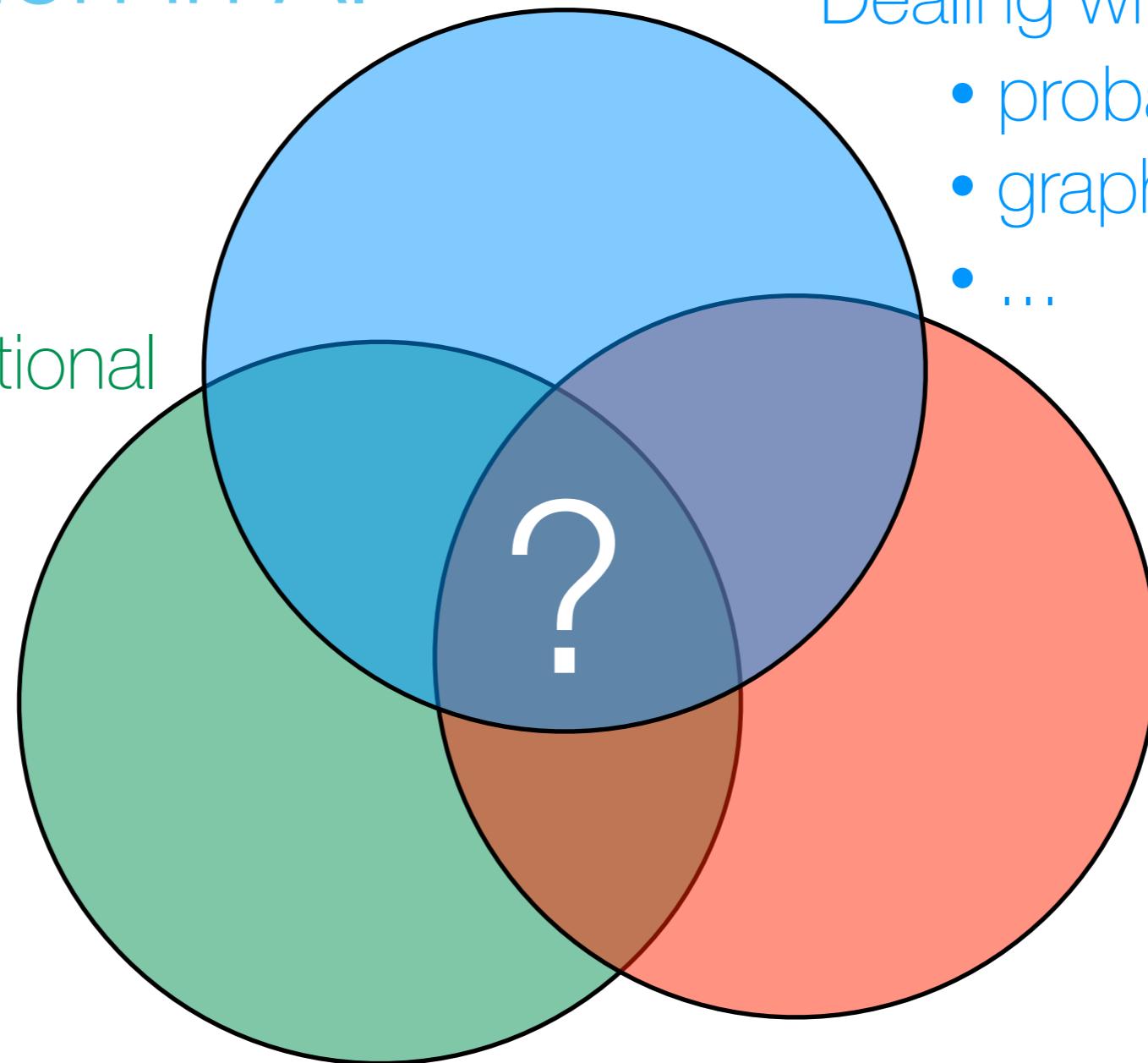
- What is Probabilistic Programming?
- ProbLog: introduction and inner workings
- Learning: parameters and structure
- Language Extensions and Variants
- Assignment

Probabilistic Programming

A key question in AI

Reasoning with relational data

- logic
- code
- databases
- programming
- ...



Dealing with uncertainty

- probability theory
- graphical models
- ...

Learning

- parameters
- structure

Statistical relational learning, probabilistic logic learning,
probabilistic programming, ...

Many formalisms exist

~Par-factor based

FACTORIE
ProbLog
Markov Logic
PRISM
Bayesian Logic Programs
Probabilistic Soft Logic
CLP(BN)
ICL
Relational Bayesian networks
Probabilistic Relational Models
...

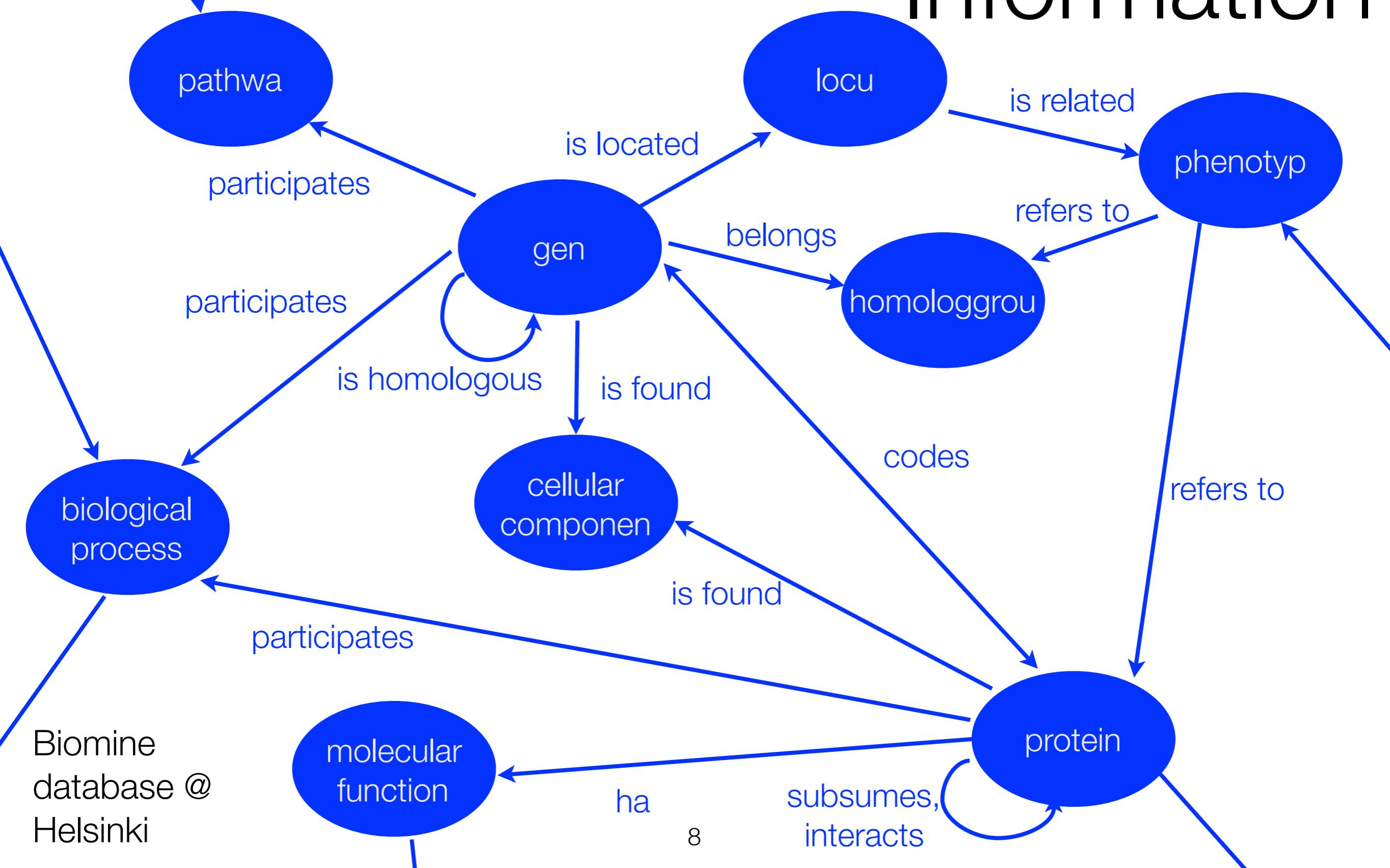
~MCMC based

Church
Stan-MC
PyMC
Anglican
Venture
BayesDB
Bayesian Logic (BLOG)
...

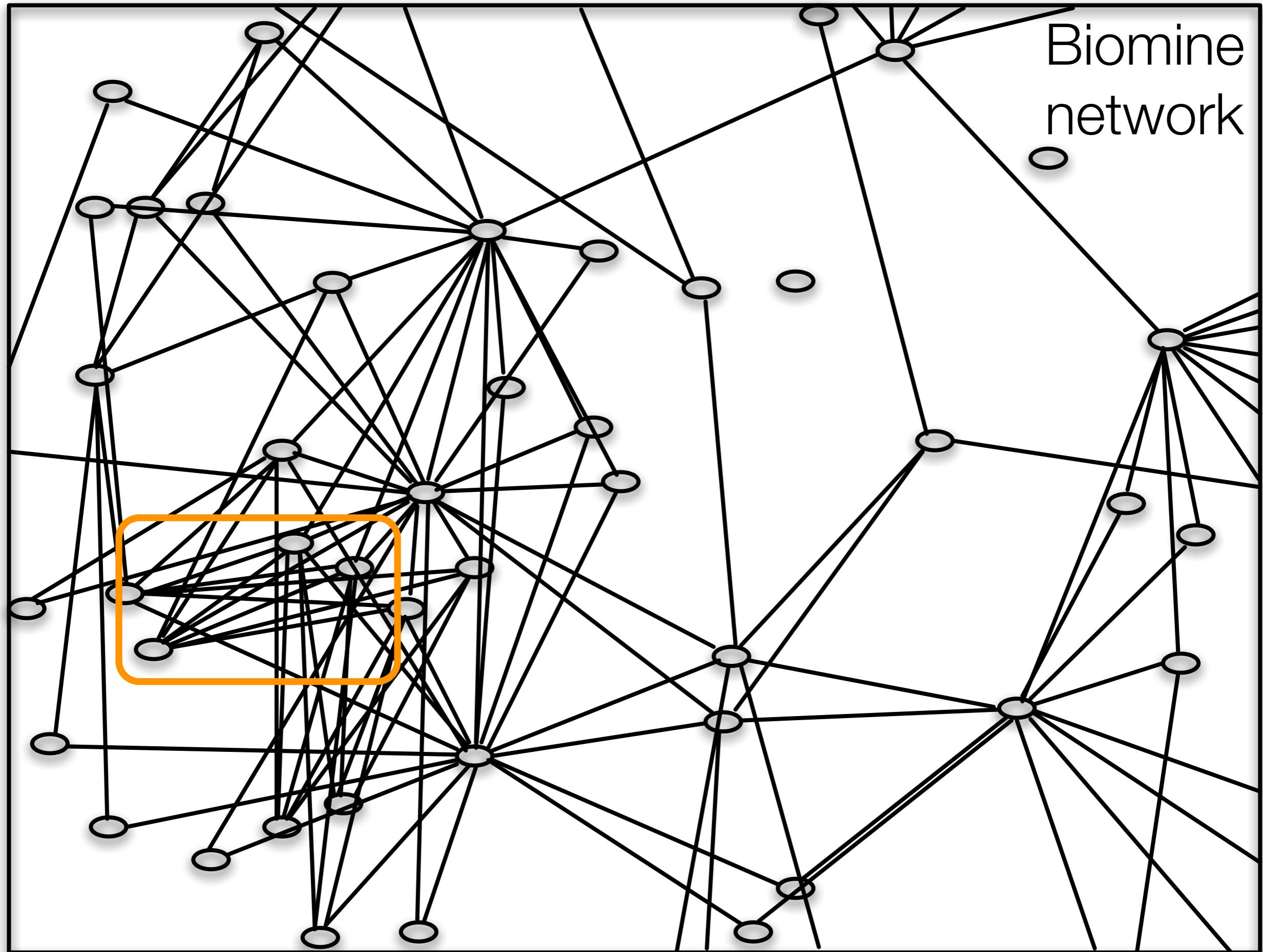
Why Relational?

- Our data is already relational!
 - Companies run relational databases
 - Scientific data is relational:
 - Large Hadron Collider generated 25PB in 2012
 - LSST Telescope will produce 30TB per night
 - Here: relational \neq (normalized) structured
 - Relations link heterogeneous data sources (tables/db/...)
 - Big data and databases are big business:
 - Oracle: \$7.1BN in sales
 - IBM: \$3.2BN in sales
 - Microsoft: \$2.6BN in sales
- ≈ GDP of
Iceland!

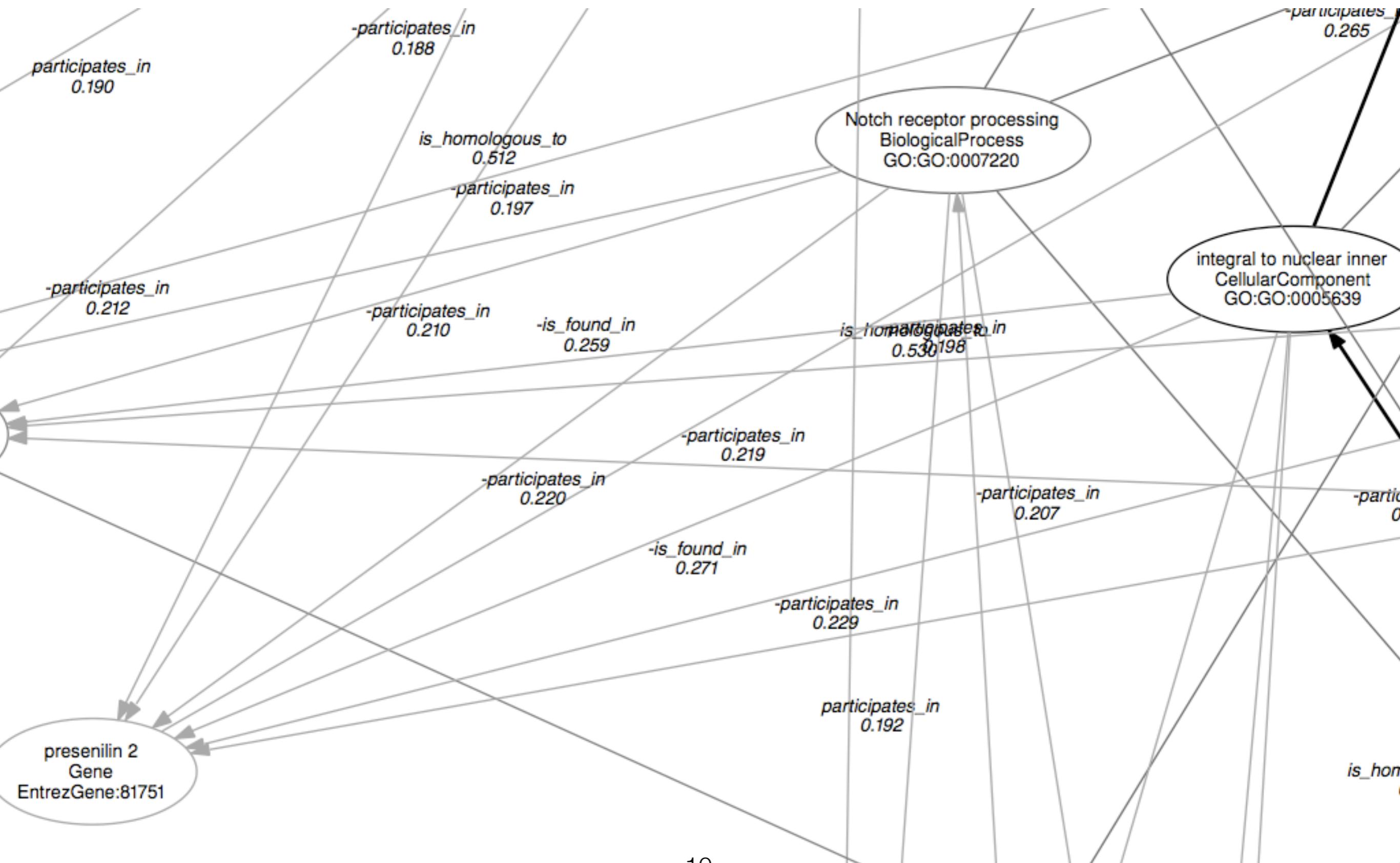
Networks of Uncertain Information



Biomine
network

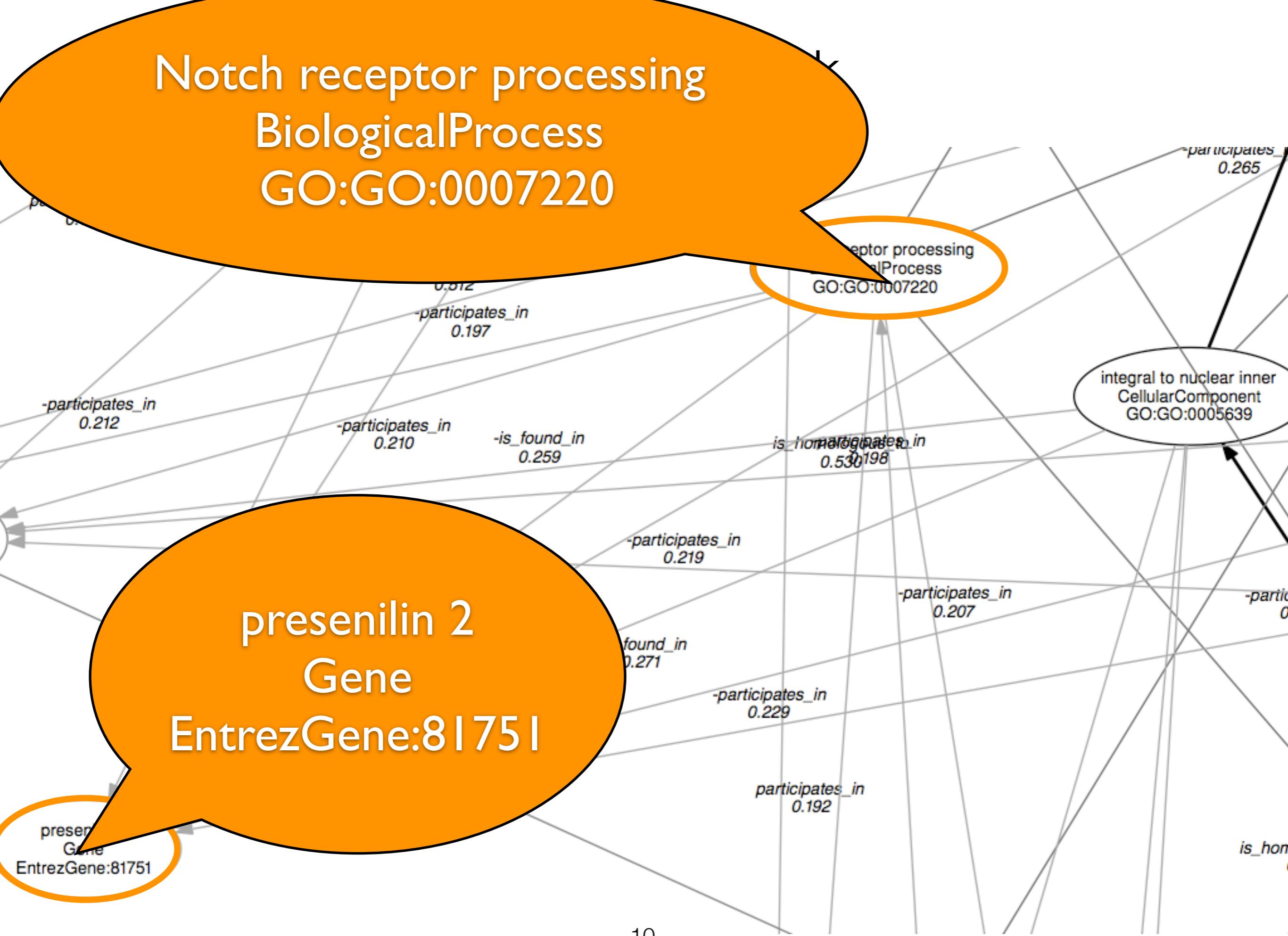


Biomine Network

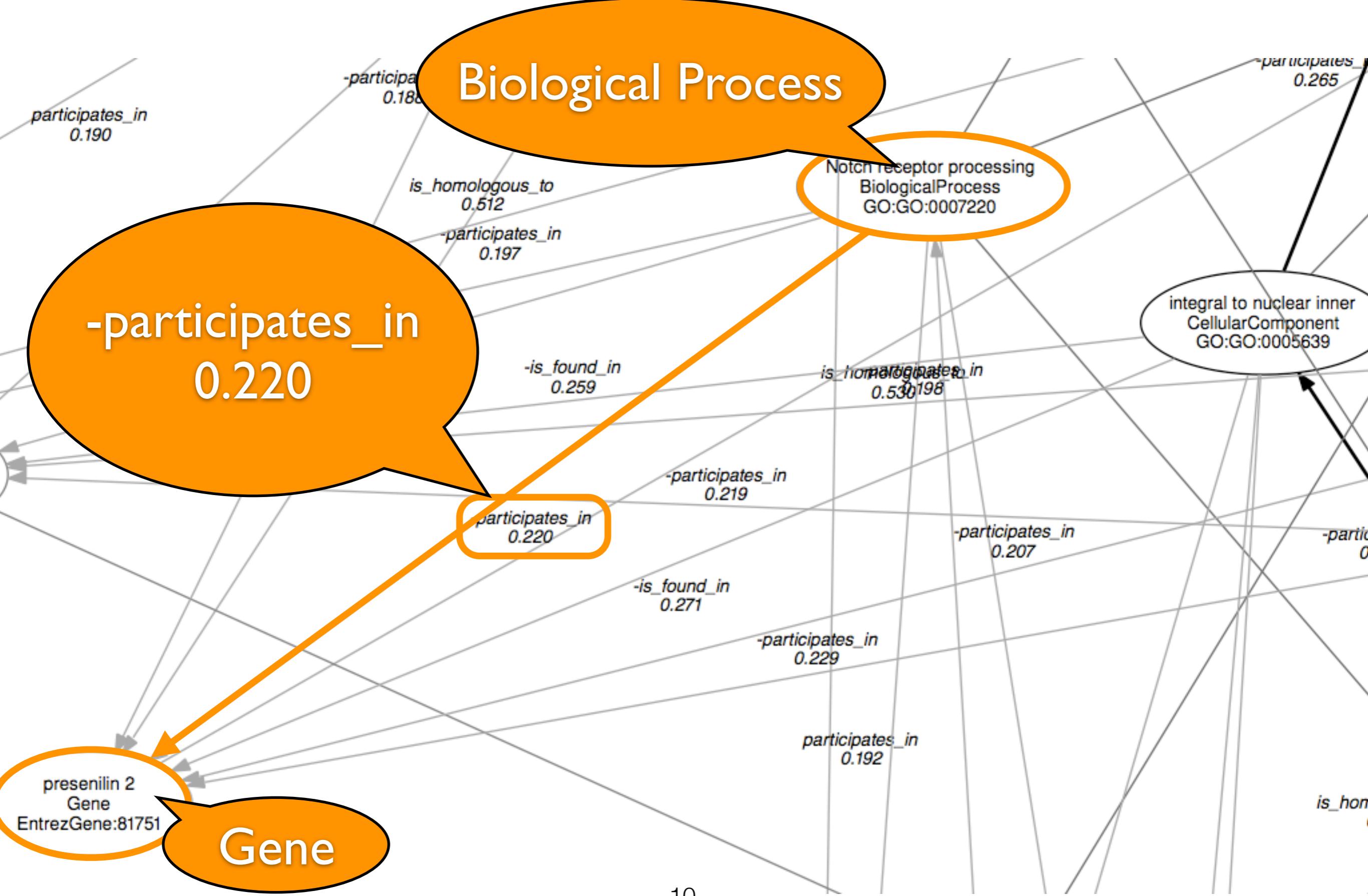


Notch receptor processing
BiologicalProcess
GO:GO:0007220

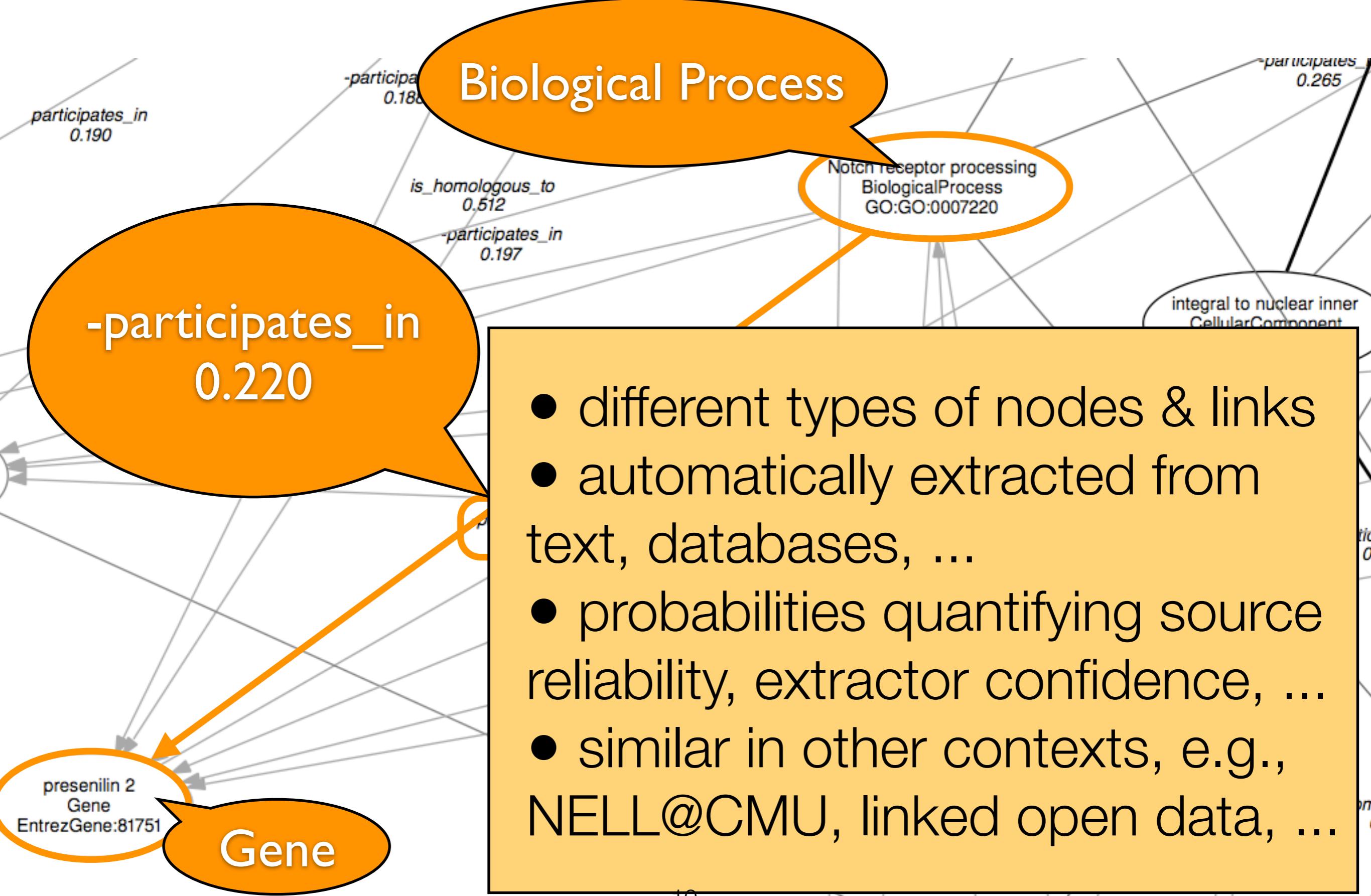
presenilin 2
Gene
EntrezGene:81751

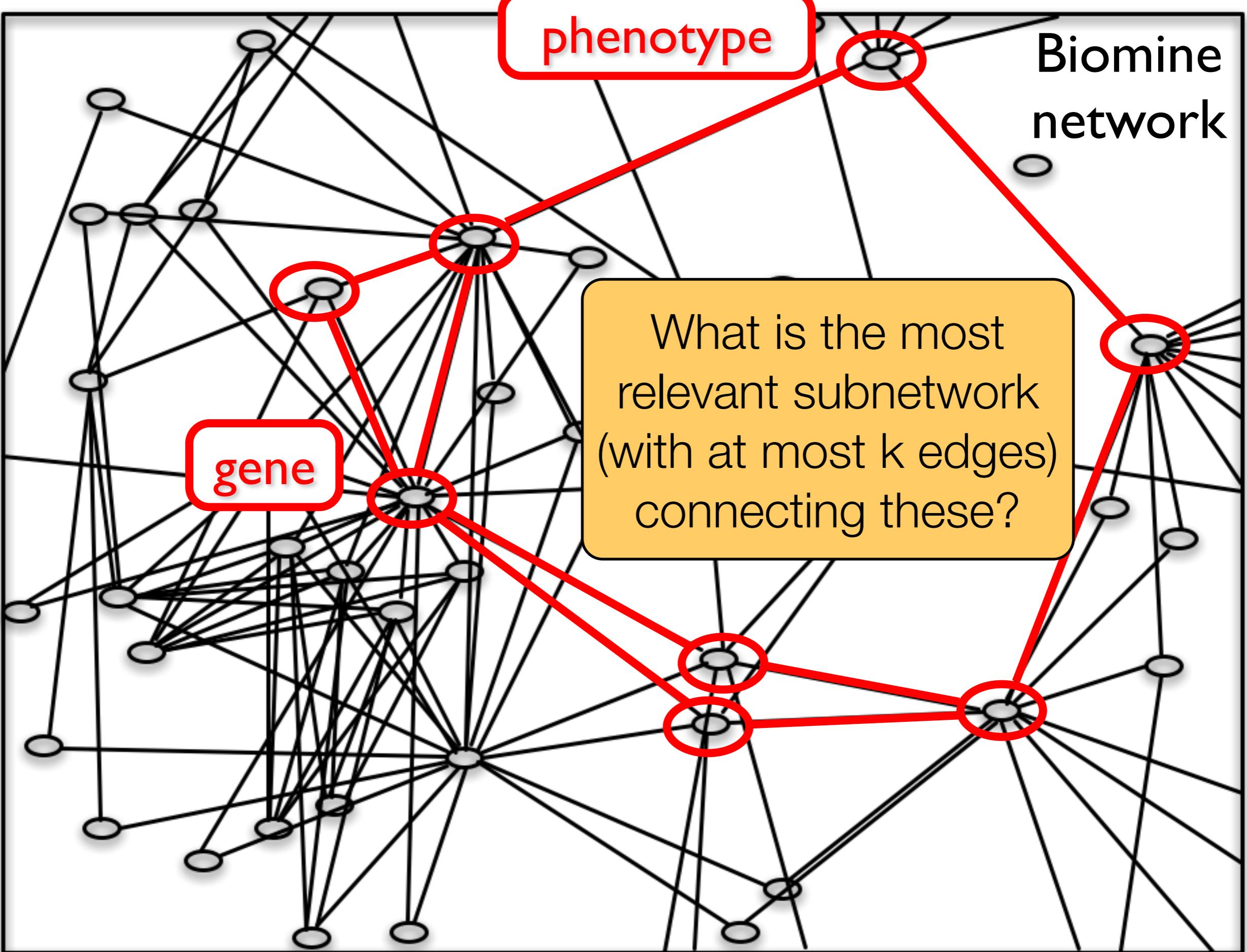


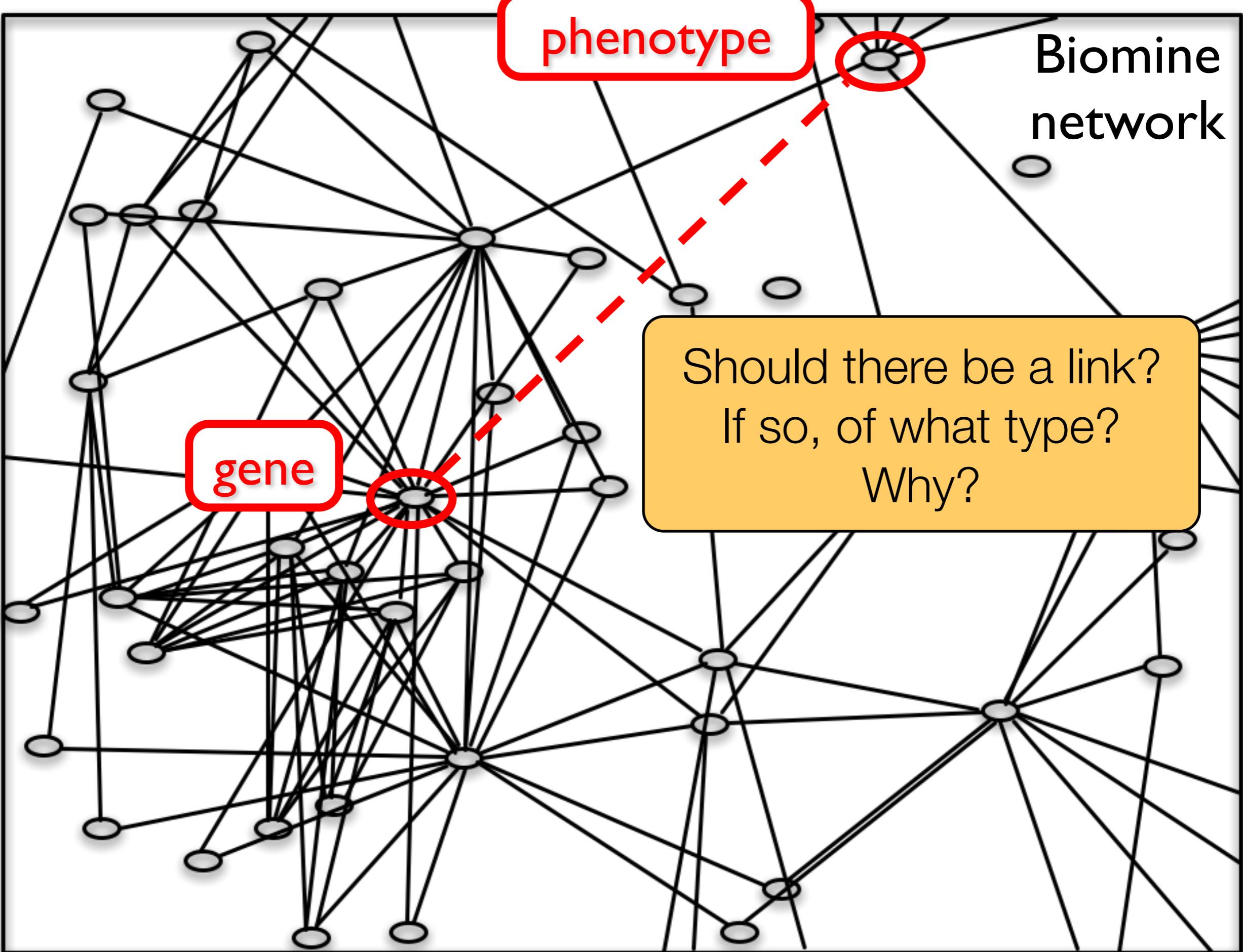
Biomine Network



Biomine Network







Google Knowledge Graph

+You Search Images Maps Play YouTube News Gmail Drive Calendar More .

Google Larry Page

Web Images Maps Shopping News More Search tools

About 350 000 000 results (0.24 seconds)

>570 million objects
>18 billion facts

Ubergizmo - 3 days ago
Android 4.4 KitKat marks a milestone for Google as they have named their mobile operating system after a branded chocolate – although ...

Larry Page - Forbes
www.forbes.com/profile/larry-page/ ▾
Larry Page on Forbes - #20 Billionaires, #20 Powerful People, #13 Forbes 400.

Larry Page - Google+
 <https://plus.google.com/+LarryPage> ▾
by Larry Page - in 6,606,272 Google+ circles
Dear Google users— You may be aware of press reports alleging that Internet companies have joined a secret U.S. government program called PRISM to give ...

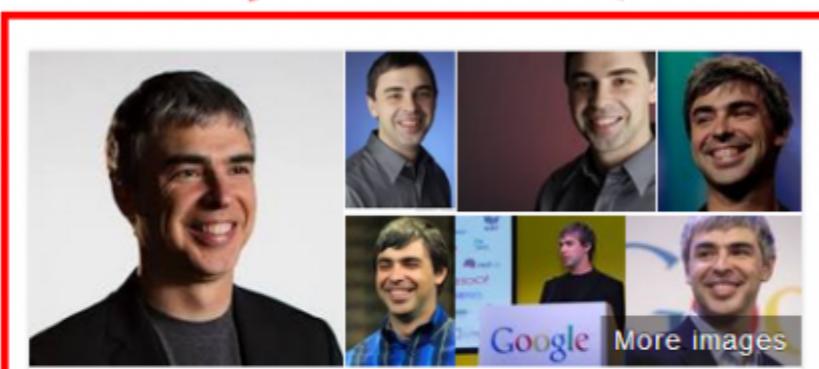
Management team – Company – Google
www.google.com/about/company/facts/management/ ▾
Larry Page and Sergey Brin founded Google in September 1998. Since then, the company has grown to more than 30,000 employees worldwide, with a ...

Larry Page Biography - Facts, Birthday, Life Story - Biography.com
[www.biography.com](http://www.biography.com/people/larry-page-9303121) ▾ People ▾
You don't need a search engine to find out all there is to know about Larry Page, co-founder of Google. Just come to Biography.com!

Larry Page | CrunchBase Profile
[www.crunchbase.com](http://www.crunchbase.com/people/larry-page) ▾ People ▾
Larry Page was Google's founding CEO and grew the company to more than 200 employees and profitability before moving into.

Oracle's Larry Ellison: Google's Larry Page acted 'evil' | Internet ...
 news.cnet.com/.../oracles-larry-ellison-googles-larry-page-acted-... ▾
by Dan Farber - in 3,346 Google+ circles
Aug 13, 2013 - In an interview with Charlie Rose, Ellison accuses Google's CEO of pursuing evilness by violating Oracle patents to develop Android.

Knowledge Graph



Larry Page
6,606,633 followers on Google+
Lawrence "Larry" Page is an American computer scientist and Internet entrepreneur who is the co-founder of Google, alongside Sergey Brin. On April 4, 2011, Page succeeded Eric Schmidt as the chief executive officer of Google. Wikipedia

Born: March 26, 1973 (age 40), [East Lansing, MI](#)
Height: 5' 11" (1.80 m)
Spouse: [Lucinda Southworth](#) (m. 2007)
Siblings: [Carl Victor Page, Jr.](#)
Education: [East Lansing High School](#) (1987–1991), [More](#)
Awards: Marconi Prize, TR100

Recent posts
Just opened the new Android release. KitKat! Sep 3, 2013

People also search for


Sergey Brin


Eric Schmidt


Larry Ellison


Marissa Mayer


Bill Gates

[Feedback / More info](#)

Example: Information Extraction

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

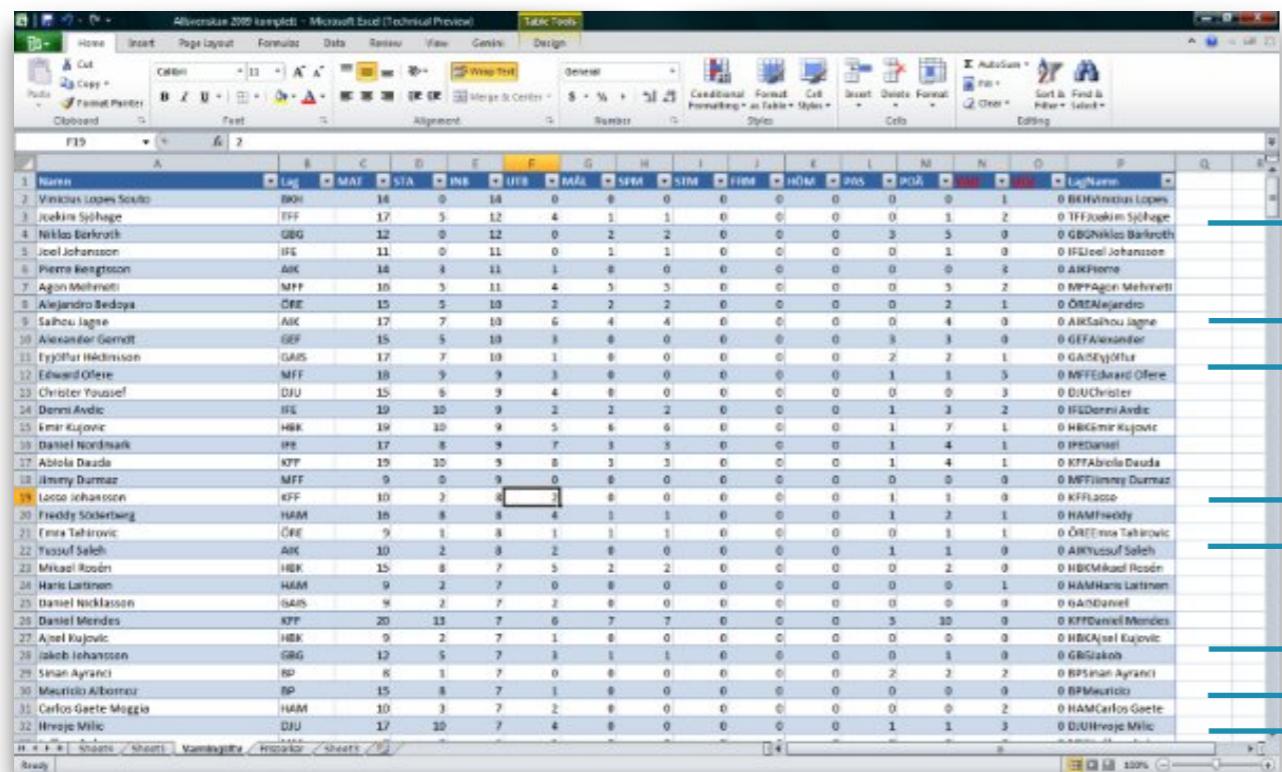
Why Statistical Relational Data?

- Relational data is increasingly probabilistic
 - NELL machine reading (>50M facts)
 - Google Knowledge Vault (>2BN facts)
 - DeepDive (>7M facts)
 - ...
- Data is inferred from unformatted information using statistical models
 - Learned from the web, large text corpora, ontologies, etc.
 - The learned/extracted data is relational
- Many applications require a combination of relational/logical and statistical models. Eg, autonomous cars that need to reason about the law but also contain open-textured or vague terms.

One table is not enough

Traditional data format

Knowledge about world



A screenshot of Microsoft Excel showing a table of player statistics. The table has columns for Name, Log, MAT, STA, INB, UTR, HAM, SPW, SHM, FHM, HOM, PWS, POA, and LagName. The data includes names like Vinicius Lopes Soárez, Joakim Sjöberg, Niklas Barkroth, Joel Johansson, Pierre Bengtsson, Agon Mehmeti, Alejandro Bedoya, Salhou Jagne, Alexander Gerdt, Trygði Héðinsson, Edward Otere, Christer Yousif, Denni Andic, Emir Kujovic, Daniel Nordmark, Abiola Dauda, Jimmy Durmaz, Leos Johansson, Freddy Söderberg, Enes Tahirovic, Yusuf Saleh, Mikael Rosén, Harri Latvala, Daniel Nicklasson, Daniel Mendes, Aysel Kujovic, Jakob Johansson, Sinan Ayrancı, Mehtap Albomuz, Carlos Góete Moggia, and Hrvoje Milic. The table is set against a background of four blue callout boxes pointing to the right, each containing a relationship type: Teammate, Family, Stands in between, and Previous training.

Name	Log	MAT	STA	INB	UTR	HAM	SPW	SHM	FHM	HOM	PWS	POA	LagName
Vinicius Lopes Soárez	BKÖ	16	0	18	0	0	0	0	0	0	0	0	0 BKÖVinicius Lopes
Joakim Sjöberg	TFF	17	5	12	4	1	1	0	0	0	1	2	0 TFFJoakim Sjöberg
Niklas Barkroth	GBG	12	0	12	0	2	2	0	0	0	3	5	0 GBGWiktor Barkroth
Joel Johansson	IFB	11	0	11	0	1	1	0	0	0	0	1	0 IFBJoel Johansson
Pierre Bengtsson	AIK	14	3	11	1	8	0	0	0	0	0	0	0 IFBPIerre Bengtsson
Agon Mehmeti	MFF	10	3	11	4	3	3	0	0	0	0	5	2 MFFAgon Mehmeti
Alejandro Bedoya	CDE	15	5	10	2	2	2	0	0	0	0	2	1 MFFAlejandro Bedoya
Salhou Jagne	AIK	17	7	10	6	4	4	0	0	0	0	4	0 AIKSalhou Jagne
Alexander Gerdt	GEF	15	5	10	8	6	0	0	0	0	3	3	0 GEFAlexander Gerdt
Trygði Héðinsson	GAR	17	7	10	1	8	0	0	0	0	2	3	1 GEFTrygði Héðinsson
Edward Otere	MFF	18	9	9	3	6	0	0	0	0	1	3	5 MFFEdward Otere
Christer Yousif	DUU	15	6	9	4	0	0	0	0	0	0	0	3 DUUChrister Yousif
Denni Andic	IFB	19	10	9	2	2	2	0	0	0	1	3	2 IFBDenni Andic
Emir Kujovic	HBK	19	10	9	5	6	6	0	0	0	1	3	1 HBKEmir Kujovic
Daniel Nordmark	IFB	17	8	9	7	3	3	0	0	0	1	4	1 IFBDaniel Nordmark
Abiola Dauda	KTF	19	10	9	8	1	3	0	0	0	1	4	1 KTFAbiola Dauda
Jimmy Durmaz	MFF	9	0	9	0	0	0	0	0	0	0	0	0 MFFJimmy Durmaz
Leos Johansson	KFF	10	2	0	2	0	0	0	0	0	1	1	0 KFFLeos Johansson
Freddy Söderberg	HAM	16	8	8	4	1	1	0	0	0	1	2	1 HAMFreddy Söderberg
Enes Tahirovic	CDE	9	1	8	1	1	1	0	0	0	0	1	0 CDEEnes Tahirovic
Yusuf Saleh	AIK	10	2	8	2	0	0	0	0	0	1	1	0 AIKYusuf Saleh
Mikael Rosén	HBK	15	8	7	5	2	2	0	0	0	0	2	0 HBKMikael Rosén
Harri Latvala	HAM	9	2	7	0	0	0	0	0	0	0	1	0 HAMHarri Latvala
Daniel Nicklasson	GAR	9	2	7	2	0	0	0	0	0	0	0	0 DANDaniel Nicklasson
Daniel Mendes	KTF	20	13	7	0	7	7	0	0	0	3	10	0 KTFDaniel Mendes
Aysel Kujovic	HBK	9	2	7	1	0	0	0	0	0	0	0	0 HBKAysel Kujovic
Jakob Johansson	GBG	12	5	7	3	1	1	0	0	0	1	0 GBGJakob Johansson	
Sinan Ayrancı	BS	6	1	7	0	0	0	0	0	2	2	2 BSSinan Ayrancı	
Mehtap Albomuz	BS	15	8	7	1	0	0	0	0	0	0	0	0 BSMehtap Albomuz
Carlos Góete Moggia	HAM	10	3	7	2	0	0	0	0	0	0	2	0 HAMCarlos Góete Moggia
Hrvoje Milic	DUU	17	10	7	4	0	0	0	0	1	1	3	0 DUUHrvoje Milic

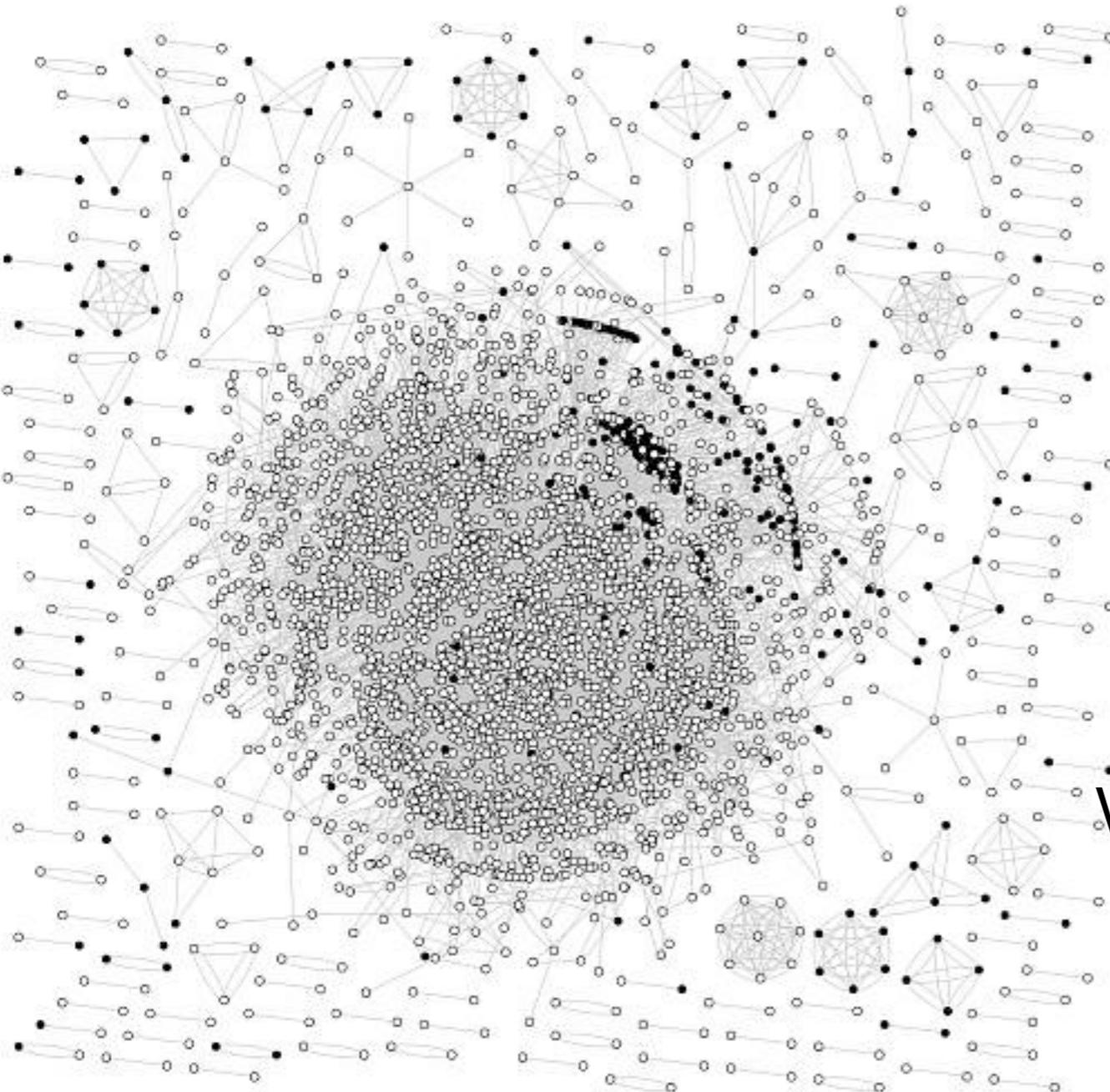
Teammate Relations

Family Hierarchical

Stands in between Relative position

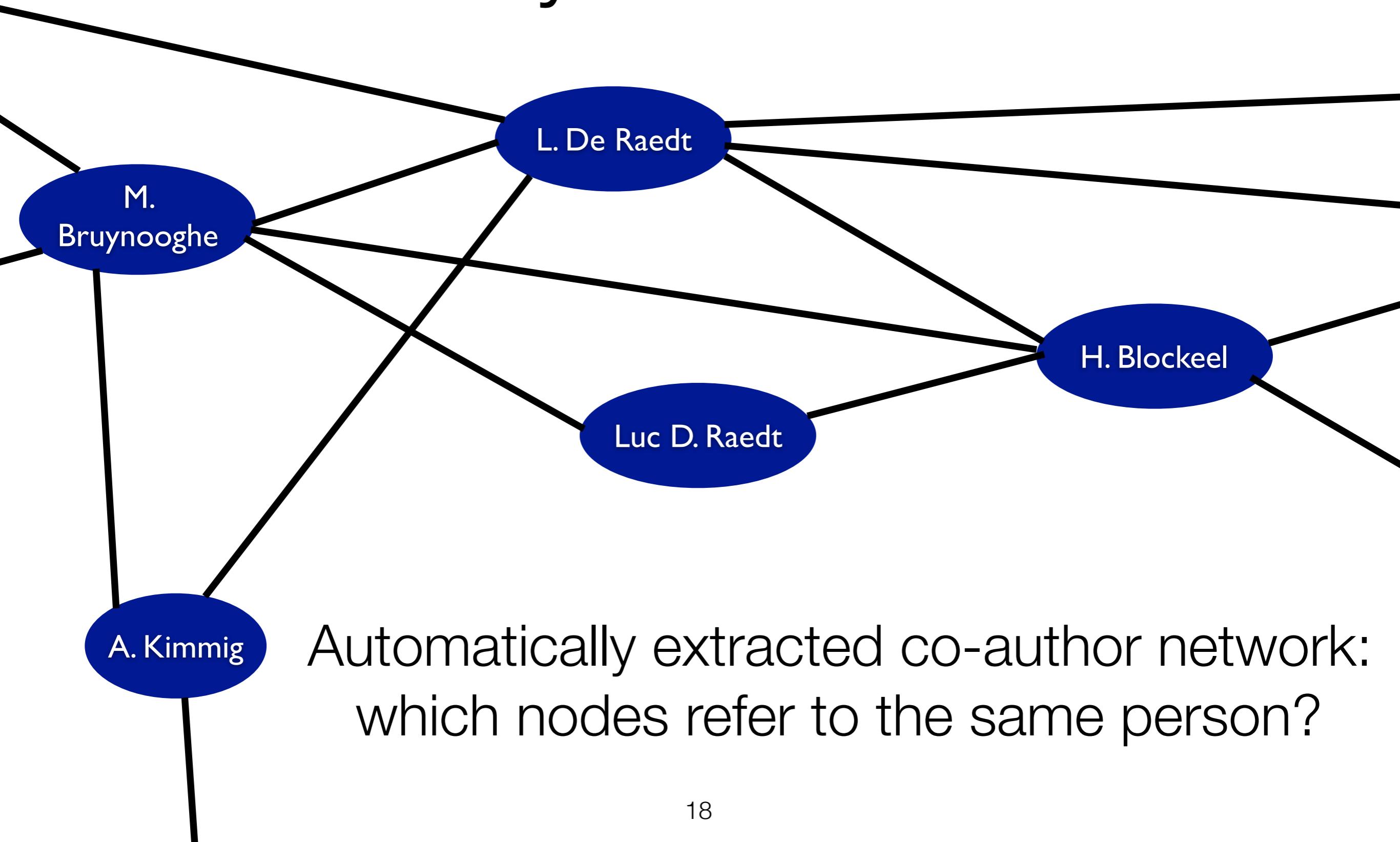
Previous training Temporal

Node Classification

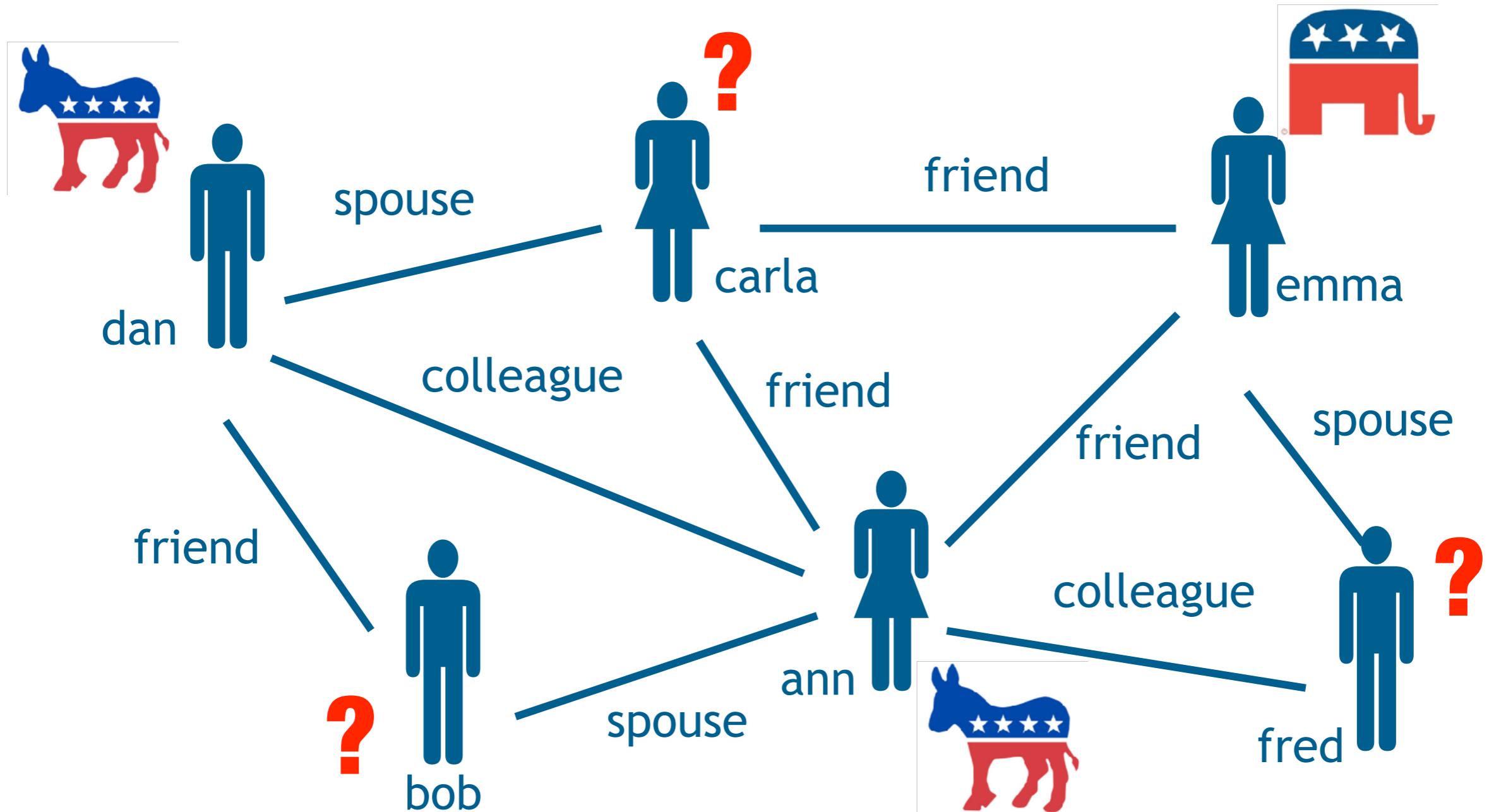


Can we predict
the type of a
node given
information on its
neighbors?
e.g., the type of a
webpage given its links
and the words on the
page?

Entity Resolution

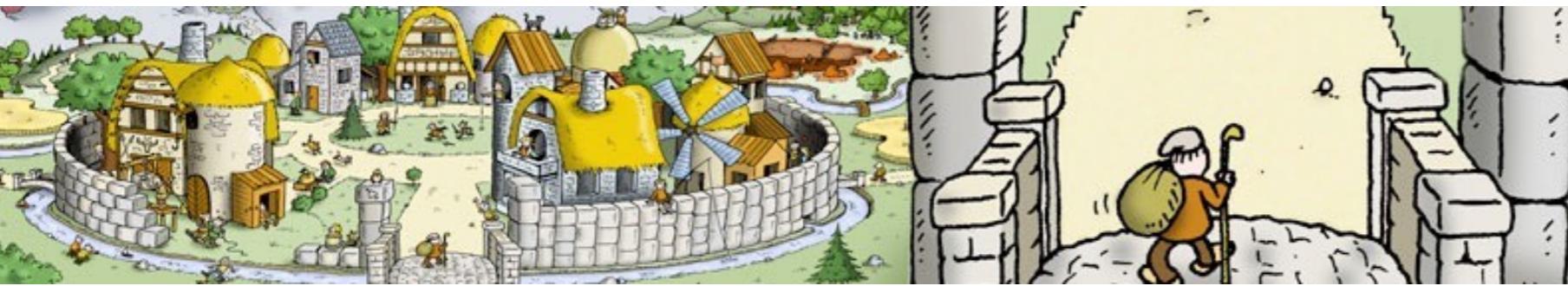


Voter Opinion Modeling



Can we predict preferences?

Dynamic networks



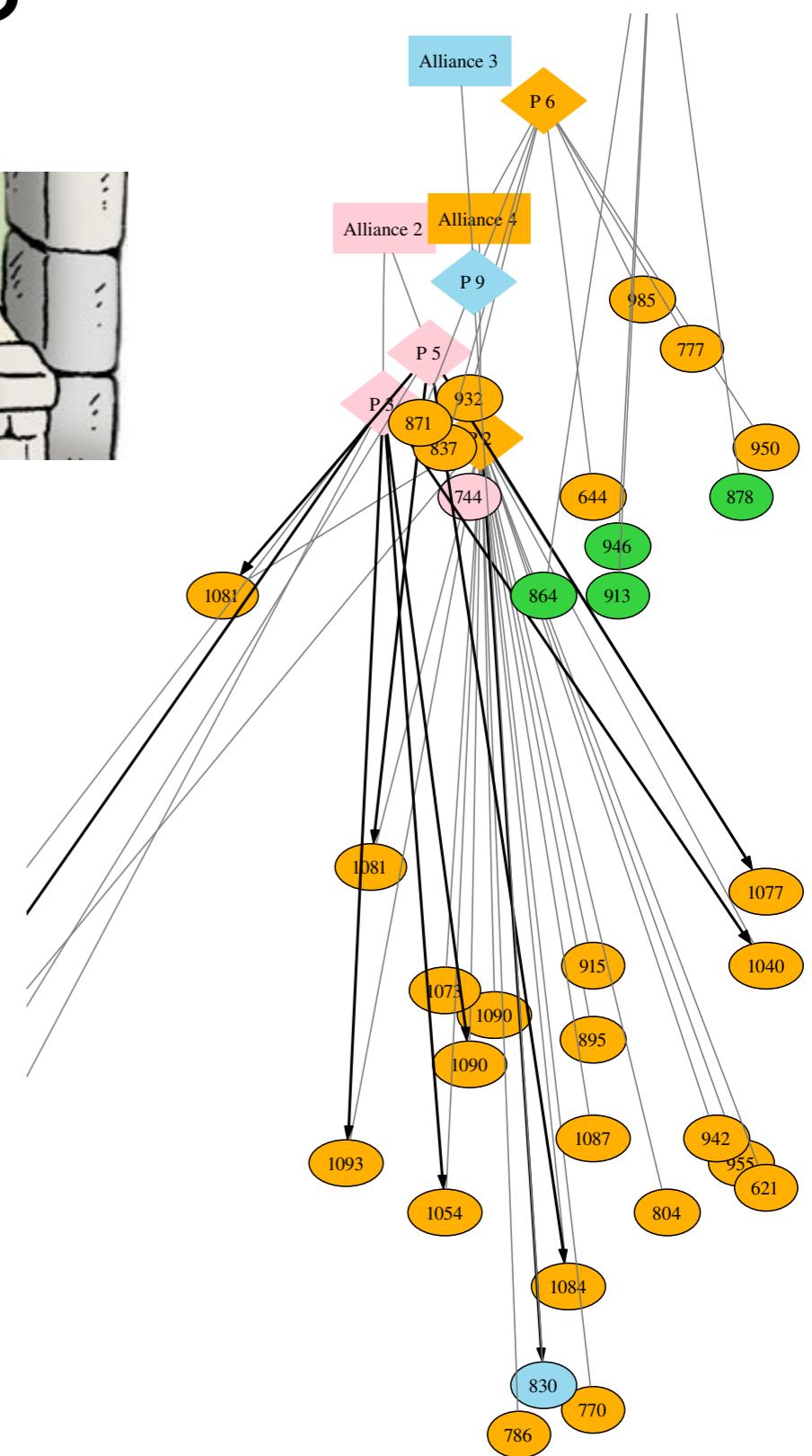
Travian: A massively multiplayer
real-time strategy game

Can we build a model
of this world ?

Can we use it for playing
better ?

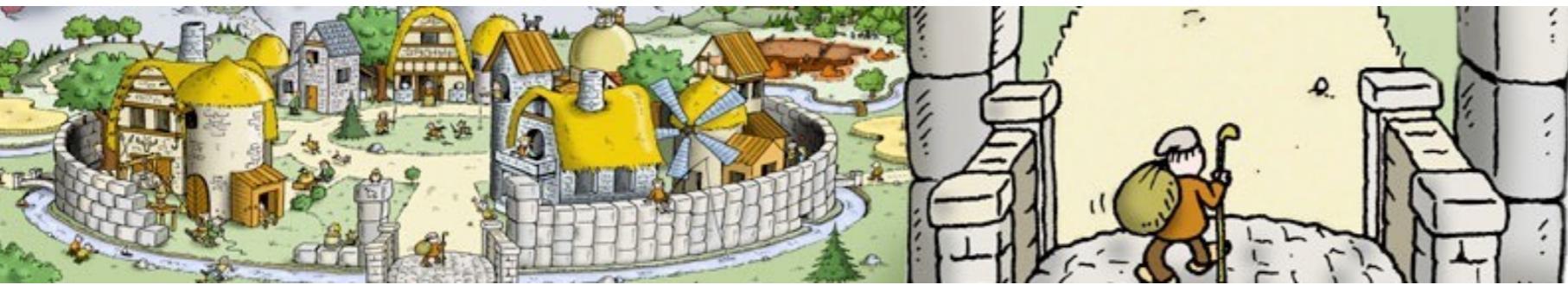


20



[Thon et al, MLJ 11]

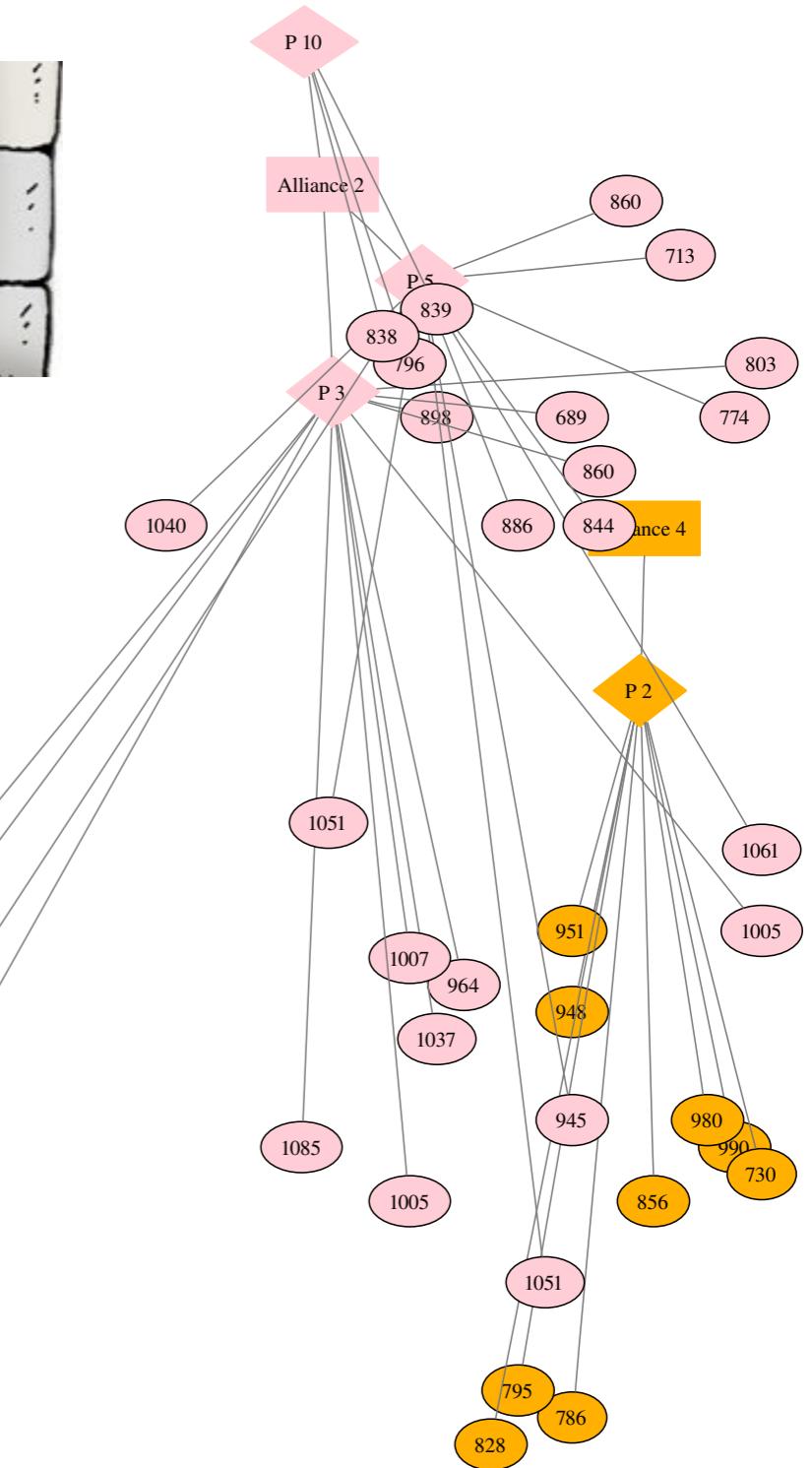
Dynamic networks



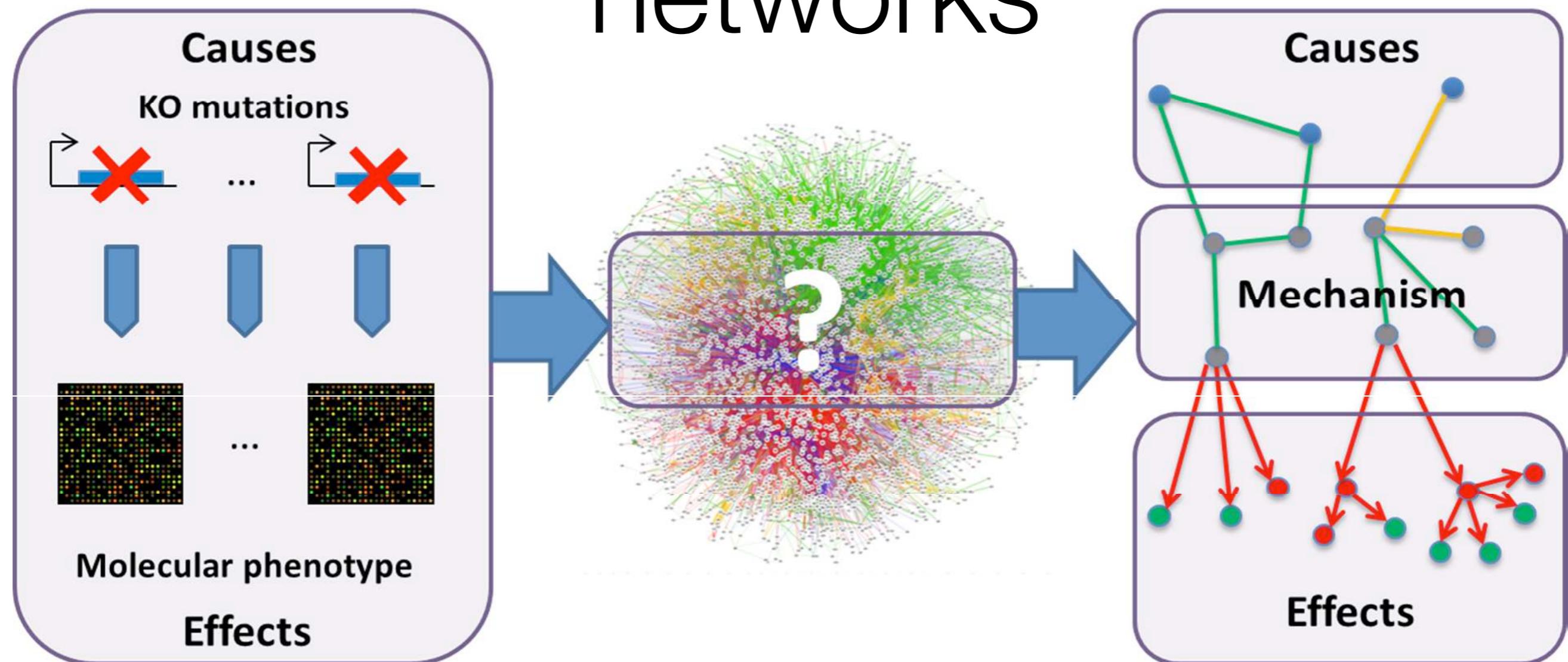
Travian: A massively multiplayer
real-time strategy game

Can we build a model
of this world ?

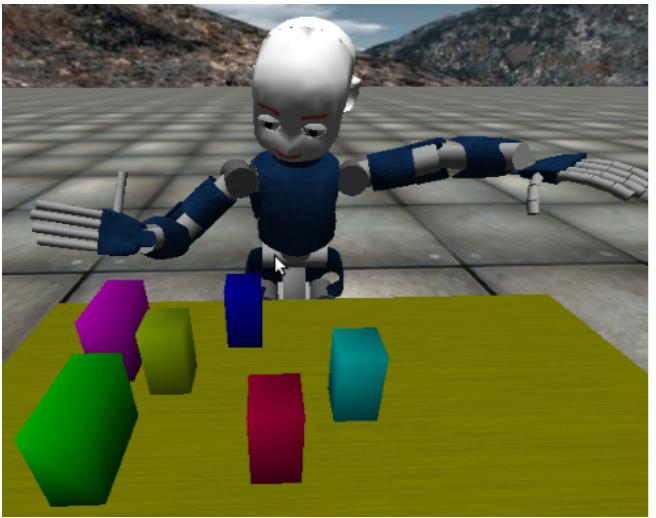
Can we use it for playing
better ?



Molecular interaction networks

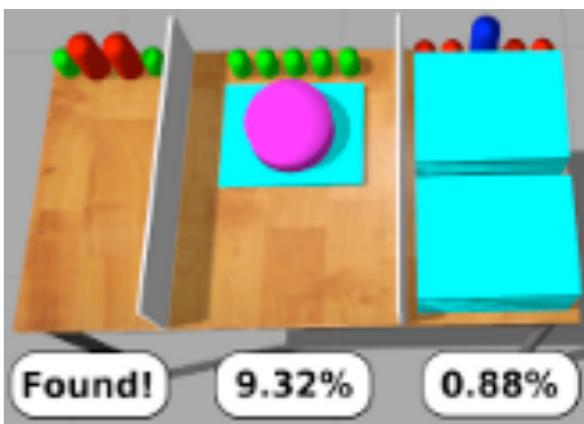
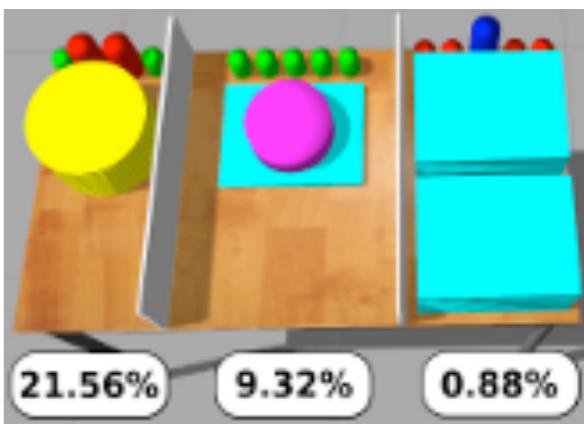
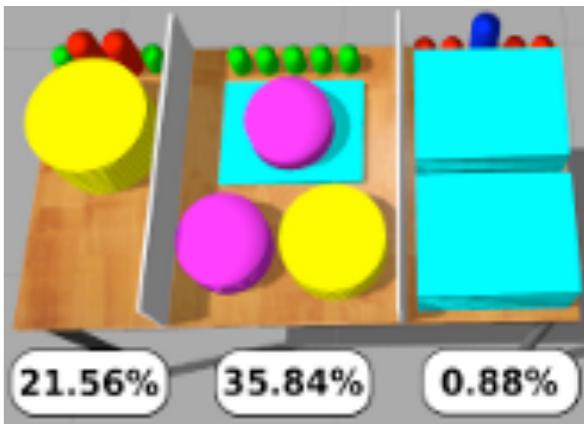
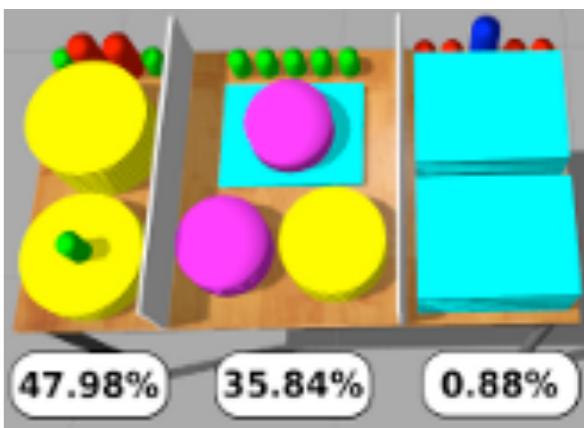


Can we find the mechanism connecting causes to effects?



Robotics

- How to achieve a specific configuration of objects on the shelf?
- Where's the orange mug?
- Where's something to serve soup in?



Answering SAT Questions



Mike has a bag of marbles with 4 white, 8 blue, and 6 red marbles. He pulls out one marble from the bag and it is red. What is the probability that the second marble he pulls out of the bag is white?

The answer is 0.235941.



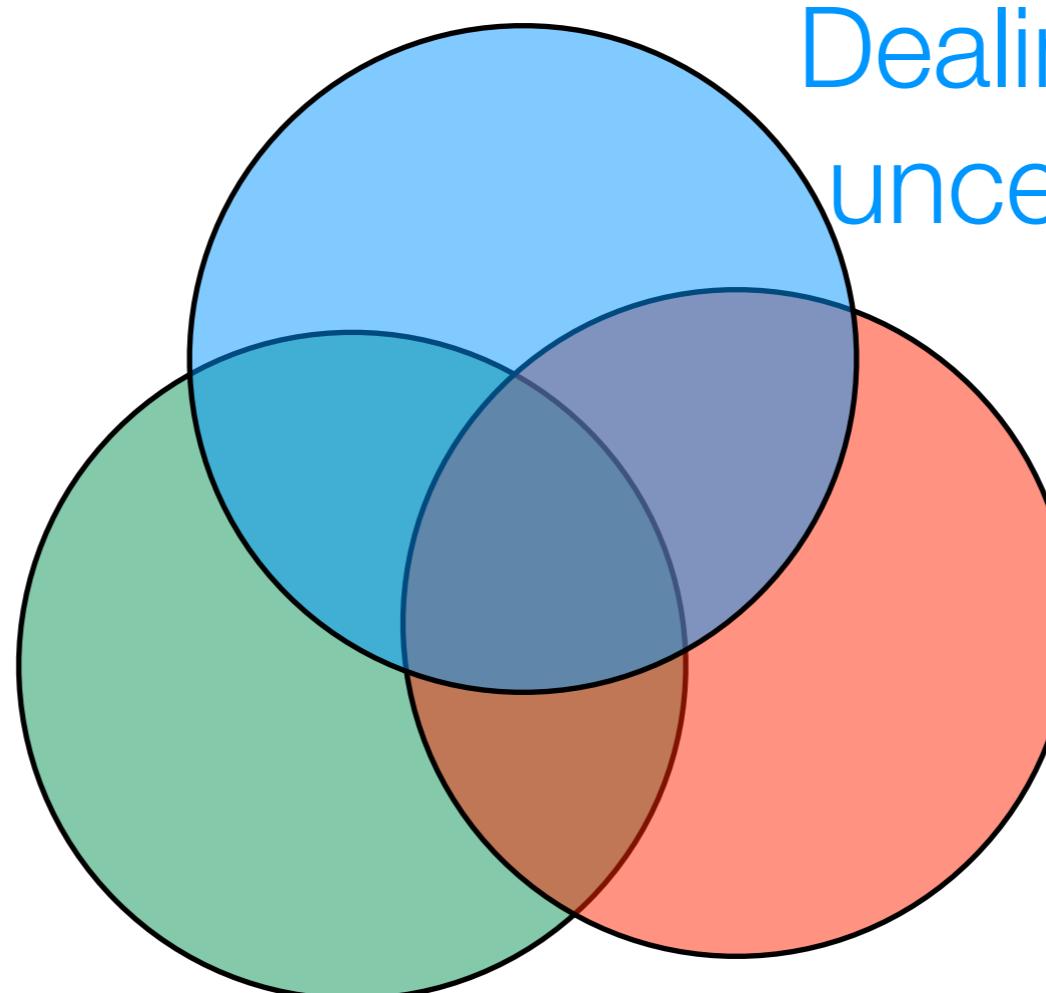
ProbLog

- Semantics by example
- Strategies for inference

ProbLog

probabilistic Prolog

Reasoning with
relational data



Dealing with
uncertainty

Learning

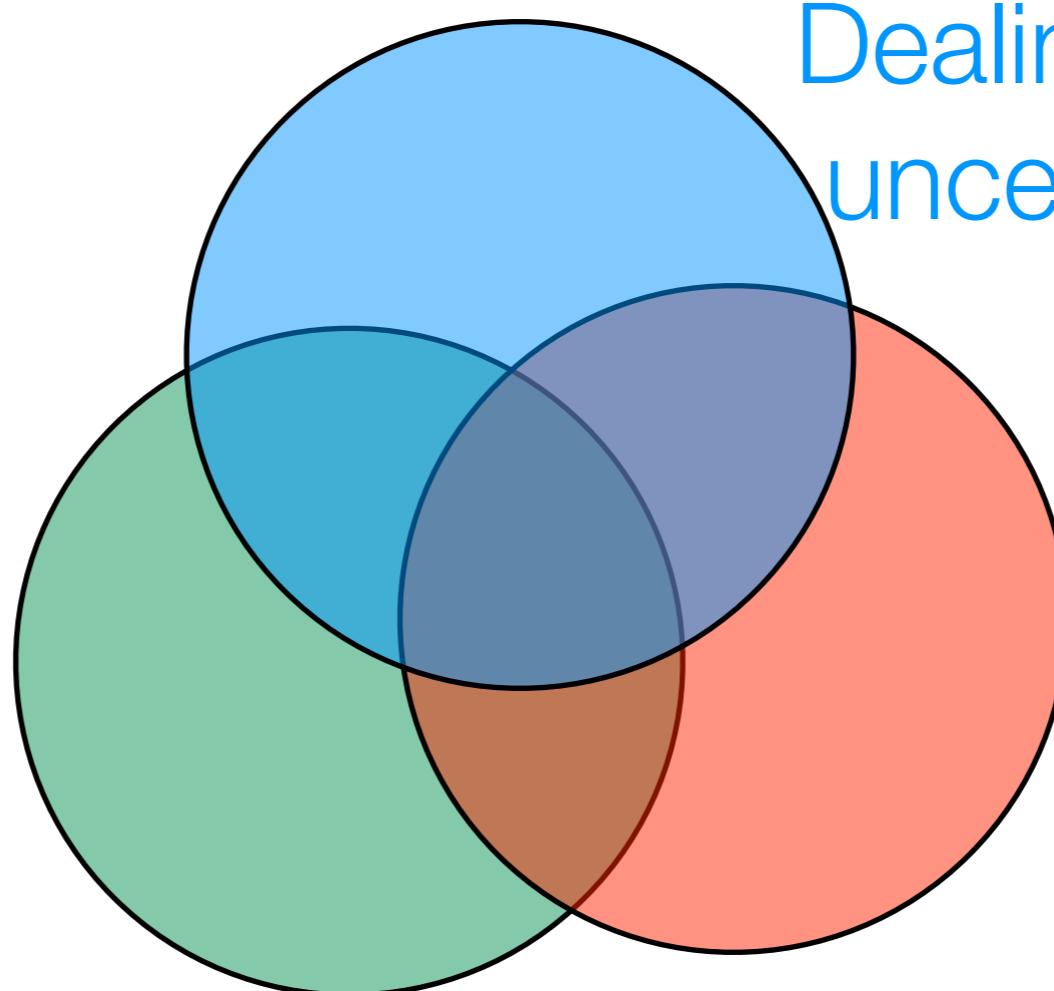
ProbLog

probabilistic Prolog

Prolog / logic
programming

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

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



Dealing with
uncertainty

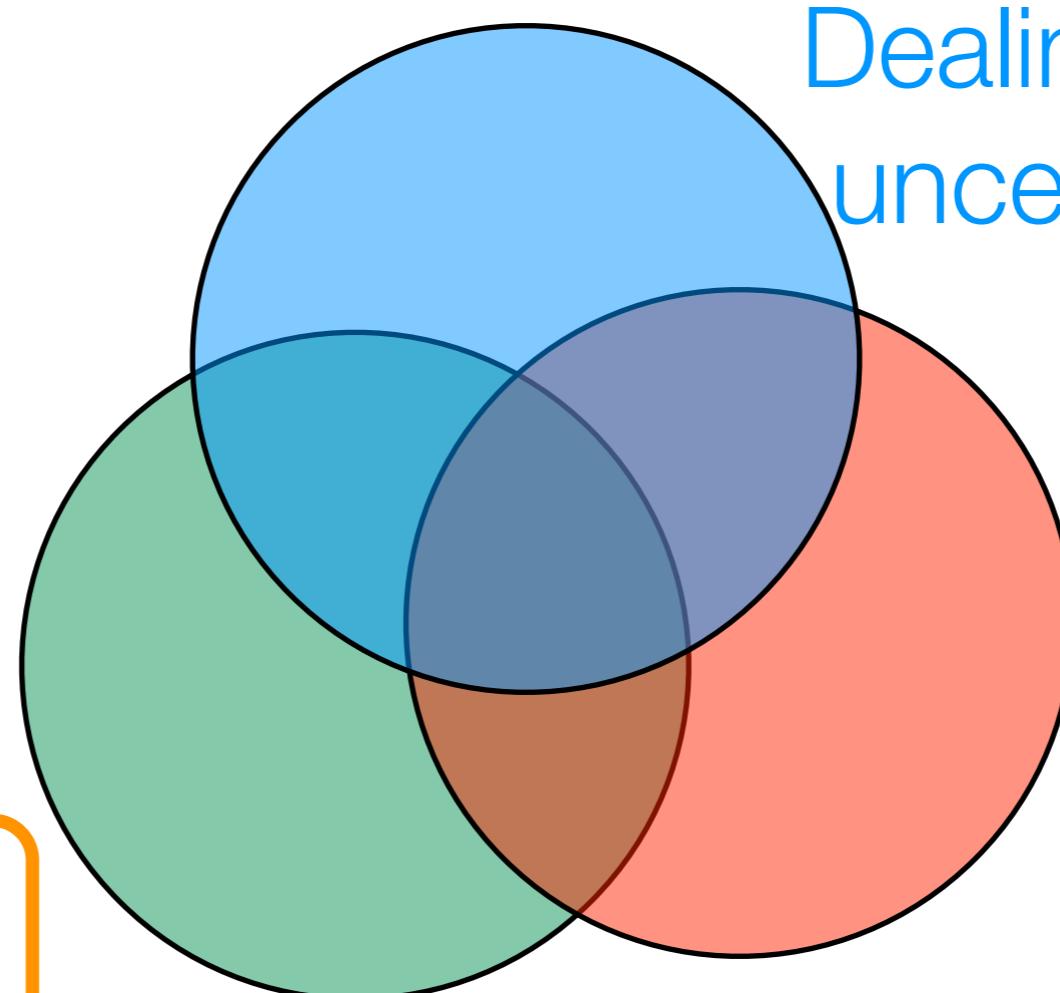
Learning

ProbLog

probabilistic Prolog

Prolog / logic
programming

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



one world

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

Dealing with
uncertainty

Learning

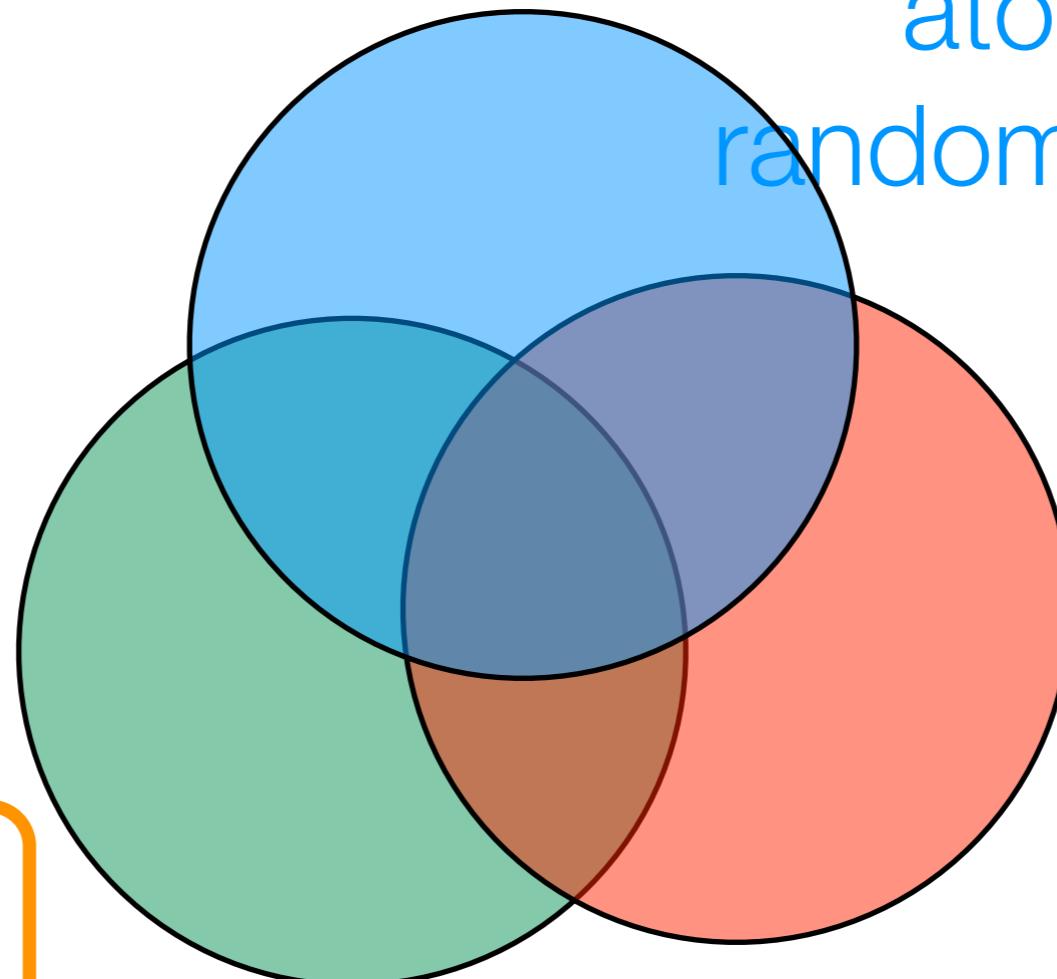
ProbLog

probabilistic Prolog

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

Prolog / logic
programming

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



one world

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

atoms as
random variables

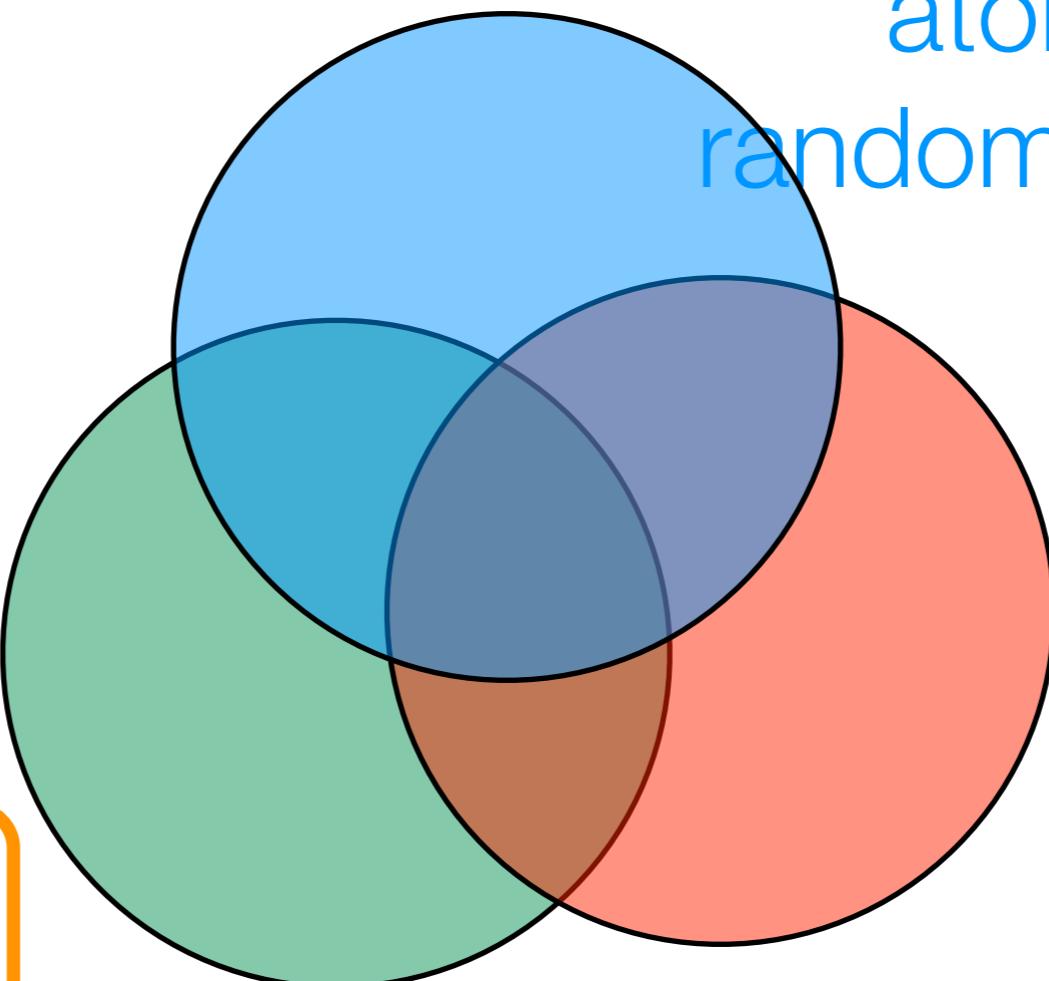
Learning

ProbLog

probabilistic Prolog

Prolog / logic
programming

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



one world

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

several possible worlds

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

atoms as
random variables

Learning

ProbLog

probabilistic Prolog

several possible worlds

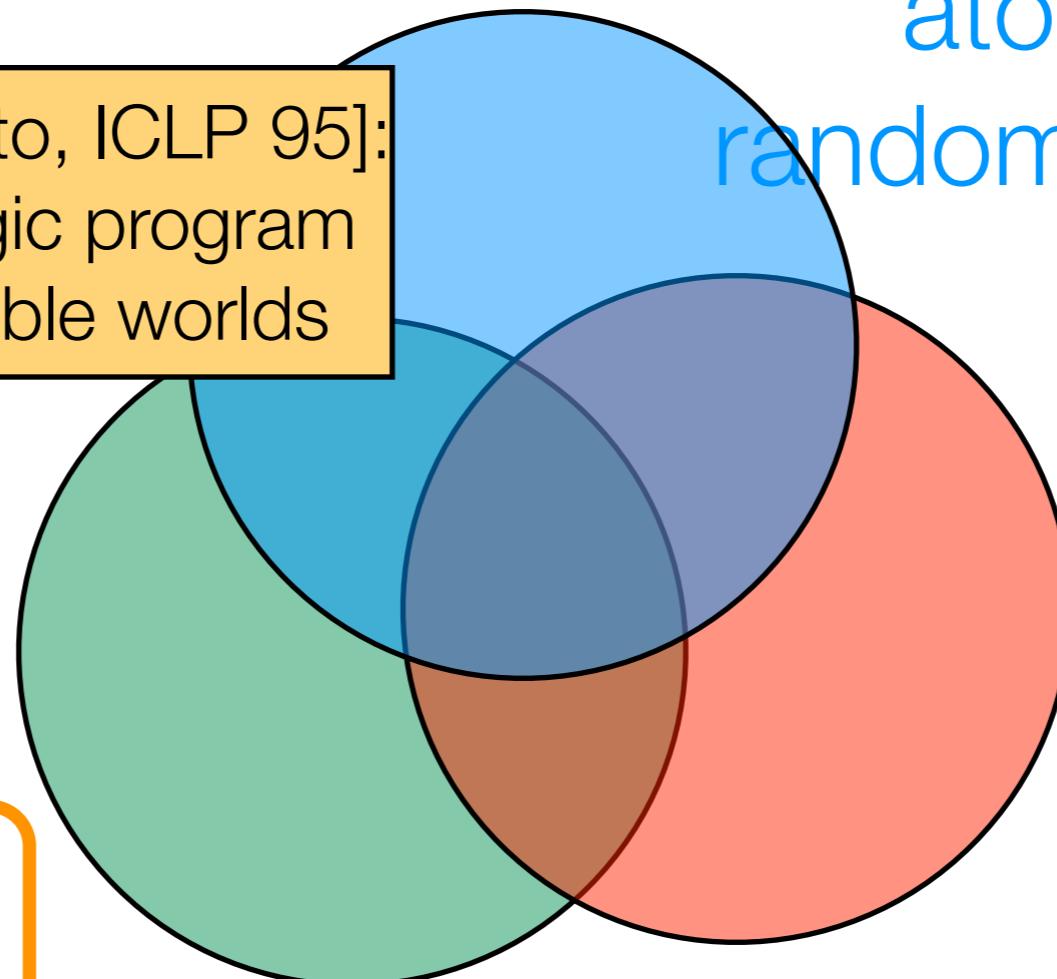
```
0.8::stress(ann).  
0.6::influences(ann,bob).  
0.2::influences(bob,carl).
```

atoms as
random variables

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

Prolog / logic
programming

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



one world

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

Learning

ProbLog

probabilistic Prolog

several possible worlds

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

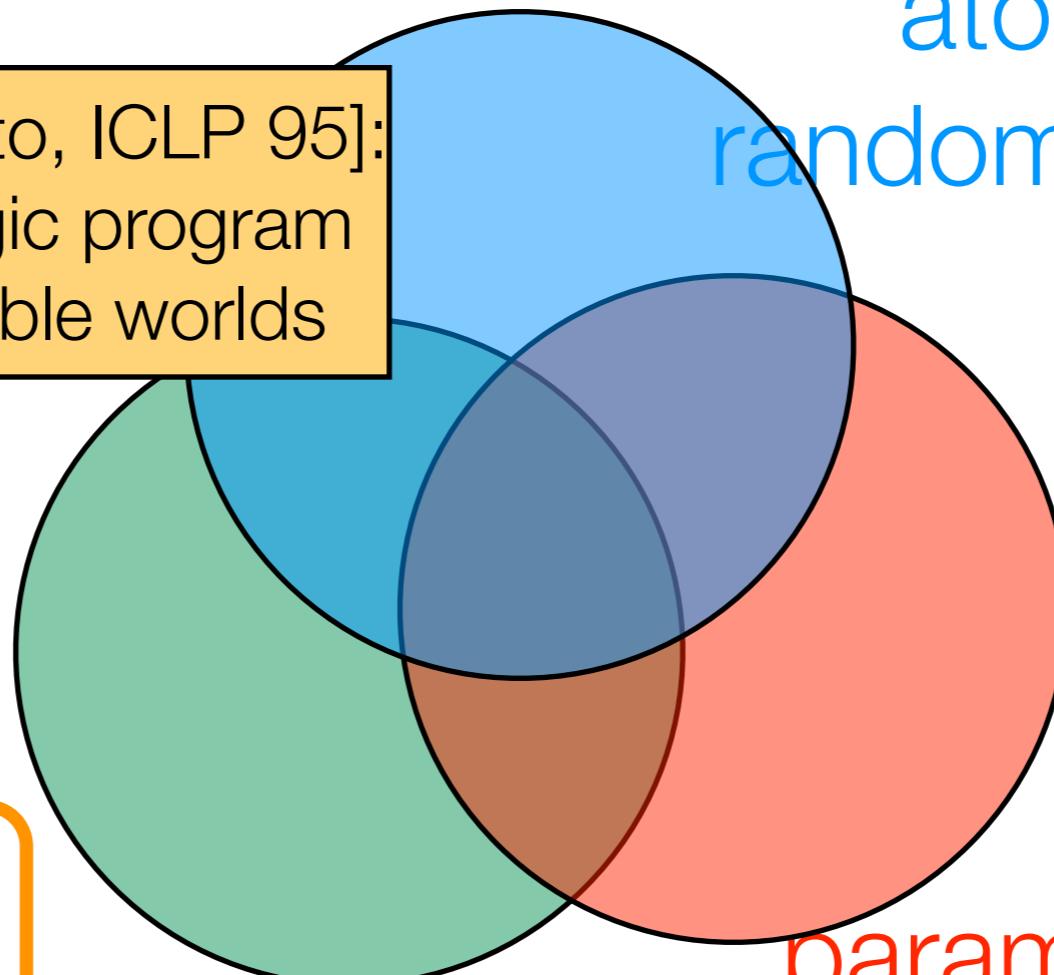
atoms as

random variables

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

Prolog / logic
programming

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

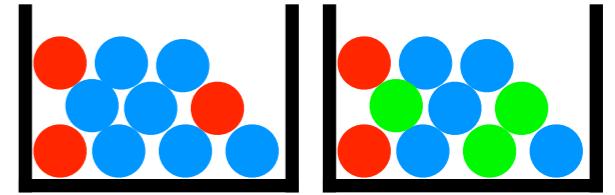


one world

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

parameter learning,
adapted relational
learning techniques

ProbLog by example:

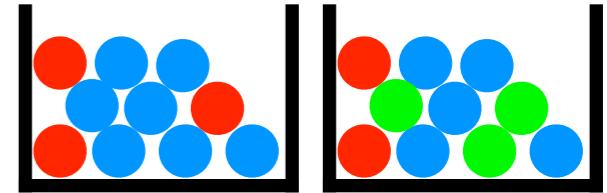


A bit of gambling

h

- 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

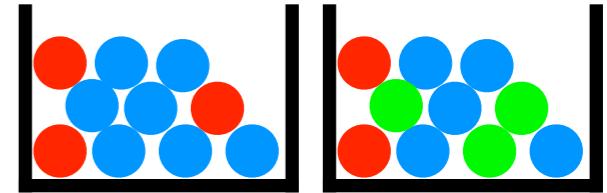


- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

0.4 :: heads .

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

ProbLog by example:



A bit of gambling

h

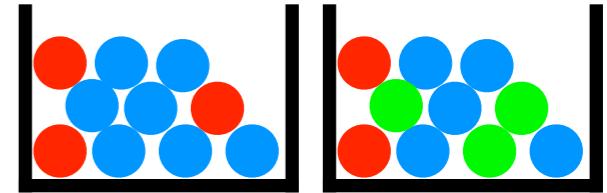
- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

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

h

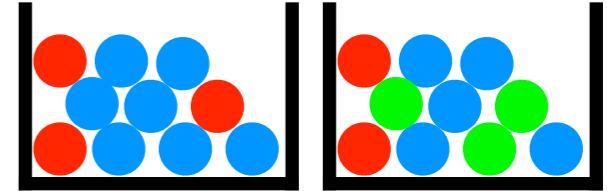
- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

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)

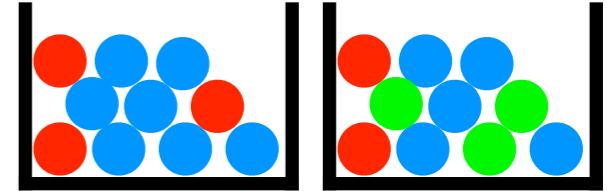
0.4 :: heads .

```
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) .
```

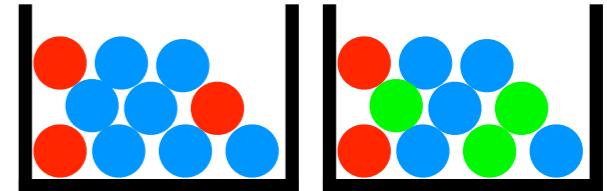
win :- **heads** , **col**(_,**red**) .

logical rule encoding

win :- **col**(1,C) , **col**(2,C) .

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 .
```

probabilistic choices

```
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) .
```

consequences

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

Questions

$0.4::\text{heads}.$

$0.3::\text{col}(1,\text{red}); 0.7::\text{col}(1,\text{blue}).$

$0.2::\text{col}(2,\text{red}); 0.3::\text{col}(2,\text{green}); 0.5::\text{col}(2,\text{blue}).$

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

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

- Probability of win?
query

Marginal probability

- Probability of win given $\text{col}(2,\text{green})$?
evidence

Conditional probability

- Most probable world where win is true?

MPE inference

Pythonic representation

```
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).
```

ProbLog

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

```
def heads():  
    return memchoice(0.4)
```

Python

How would you implement memchoice?

Stochastic variables

```
colors = {  
    1: {'red': 0.3, 'blue': 0.7}  
    2: {'red': 0.2, 'green': 0.3, 'blue': 0.5}  
}
```

```
def color(u):  
    return memchoice(colors[u])
```

```
def win():  
    return heads() and any([color[u] == 'red' for u in colors.keys()]) or \  
          color(1) == color(2)
```

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).
```



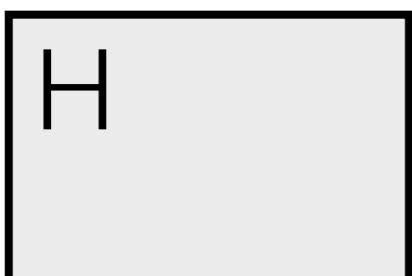
Possible Worlds

```
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



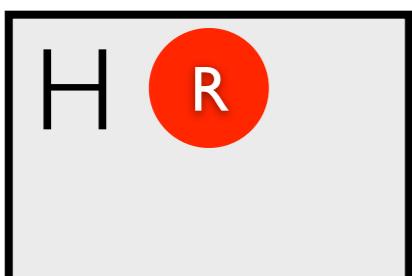
Possible Worlds

```
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



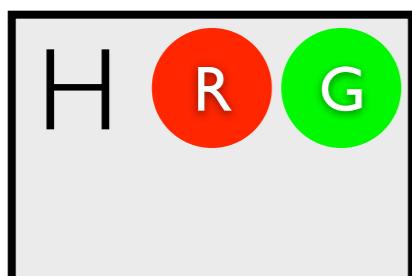
Possible Worlds

```
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

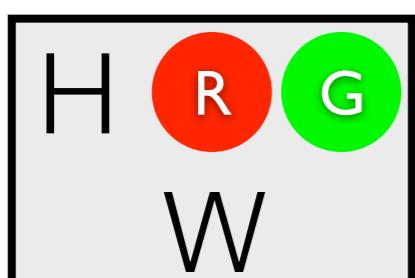
```
0.4 :: heads.
```

```
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).
```

$0.4 \times 0.3 \times 0.3$



Possible Worlds

```
0.4 :: heads.
```

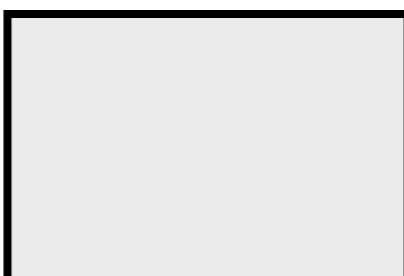
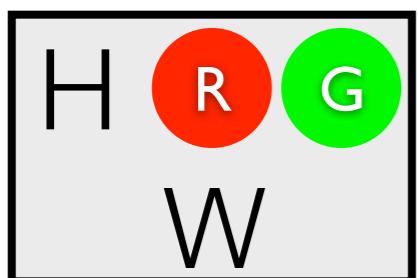
```
0.3 :: col(1,red); 0.7 :: col(1,blue)..
```

```
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).
```

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



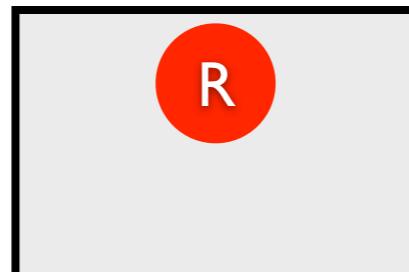
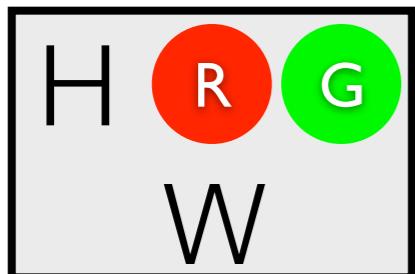
Possible Worlds

```
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).
```

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



Possible Worlds

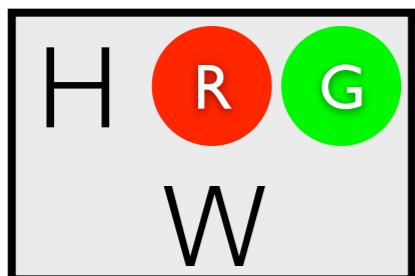
```
0.4 :: heads.
```

```
0.3 :: col(1,red) ; 0.7 :: col(1,blue)
```

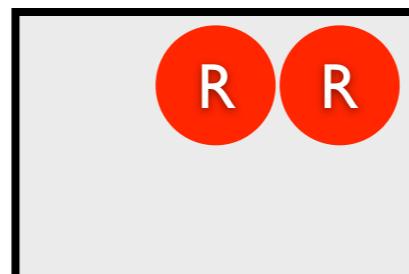
```
0.2 :: col(2,red) ; 0.3 :: col(2,green) ; 0.5 :: col(2,blue).
```

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

$$0.4 \times 0.3 \times 0.3$$



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



Possible Worlds

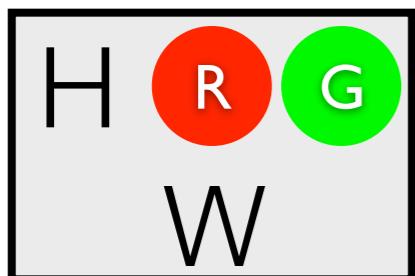
```
0.4 :: heads.
```

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

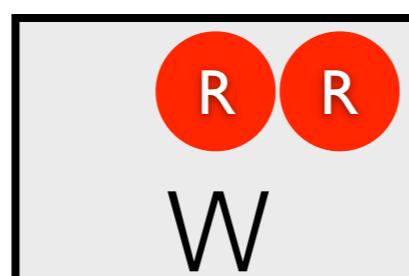
```
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).
```

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

$$0.4 \times 0.3 \times 0.3$$



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



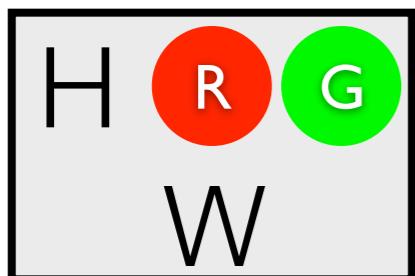
Possible Worlds

```
0.4 :: heads.
```

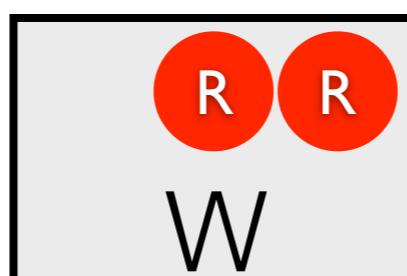
```
0.3 :: col(1,red); 0.7 :: col(1,blue)..  
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).
```

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

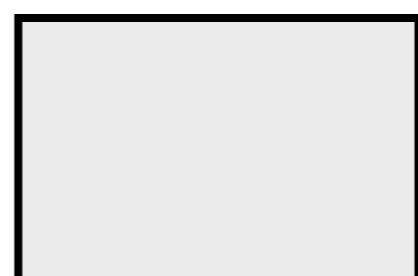
$0.4 \times 0.3 \times 0.3$



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



$(1 - 0.4)$



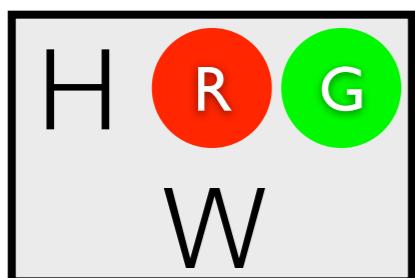
Possible Worlds

```
0.4 :: heads.
```

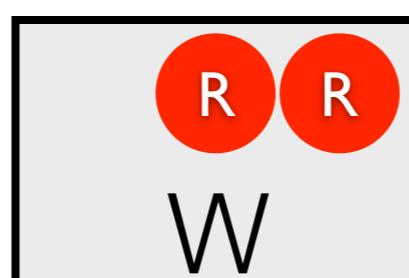
```
0.3 :: col(1,red) ; 0.7 :: col(1,blue)..  
0.2 :: col(2,red) ; 0.3 :: col(2,green) ; 0.5 :: col(2,blue).
```

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

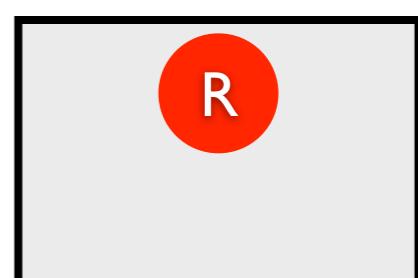
$0.4 \times 0.3 \times 0.3$



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



$(1 - 0.4) \times 0.3$



Possible Worlds

```
0.4 :: heads.
```

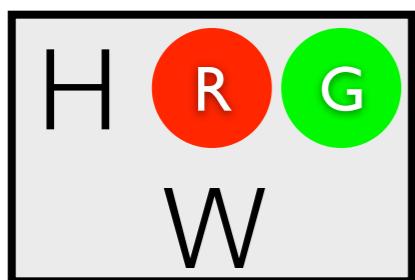
```
0.3 :: col(1, red) ; 0.7 :: col(1, blue)
```

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

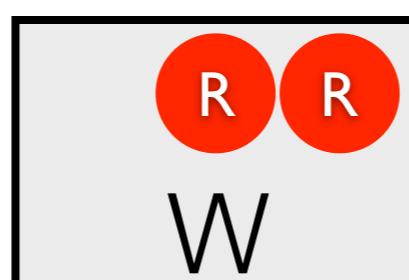
```
win :- heads, col(_, red).
```

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

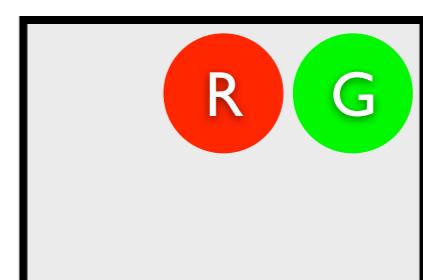
$0.4 \times 0.3 \times 0.3$



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



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



Possible Worlds

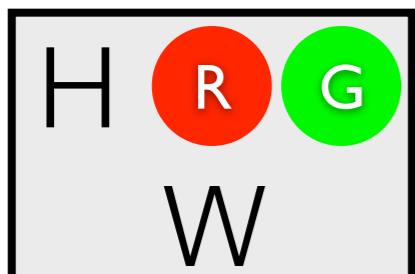
```
0.4 :: heads.
```

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

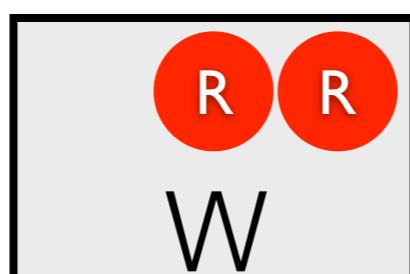
```
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).
```

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

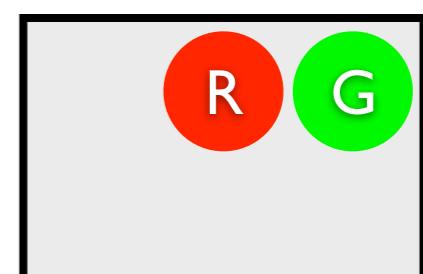
$0.4 \times 0.3 \times 0.3$



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



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



Possible Worlds

```
0.4 :: heads.
```

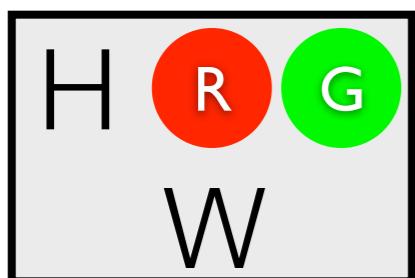
```
0.3 :: col(1,red) ; 0.7 :: col(1,blue)..
```

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

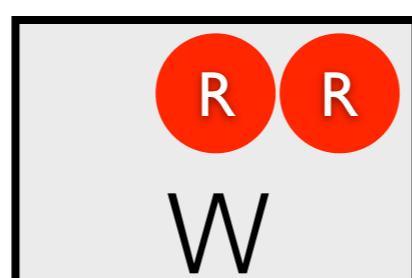
```
win :- heads, col(_,red).
```

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

$0.4 \times 0.3 \times 0.3$



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

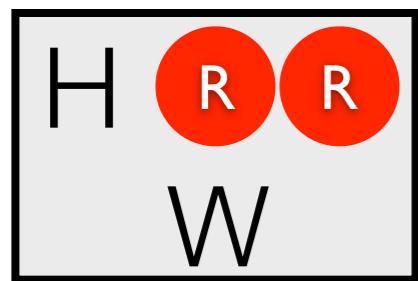


$(1 - 0.4) \times 0.3 \times 0.3$

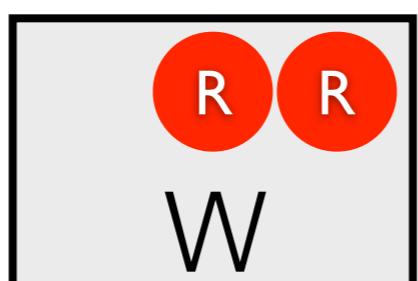


All Possible Worlds

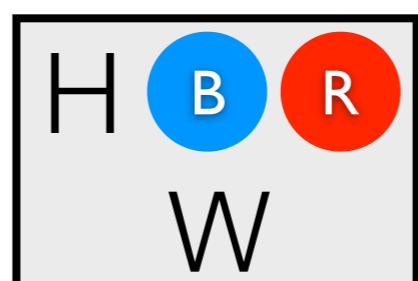
0.024



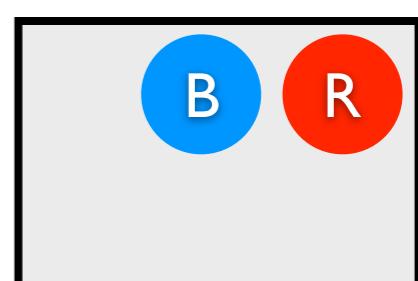
0.036



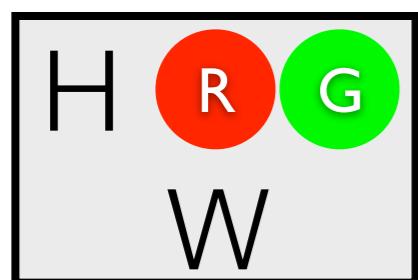
0.056



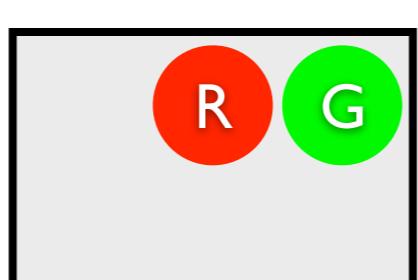
0.084



0.036



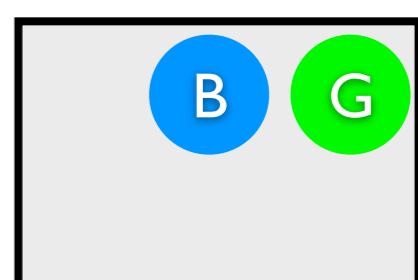
0.054



0.084



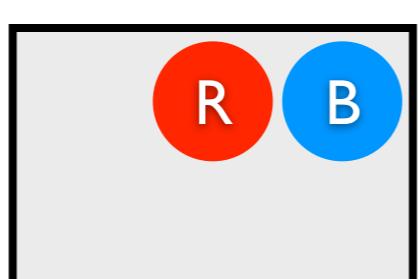
0.126



0.060



0.090



0.140



0.210

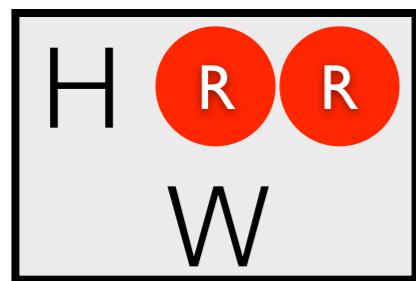


Most likely world

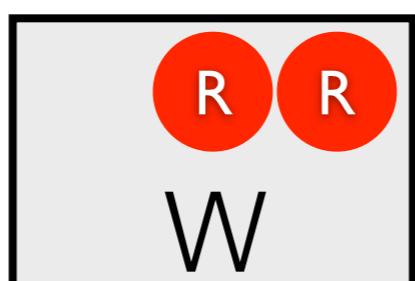
MPE Inference

where **win** is

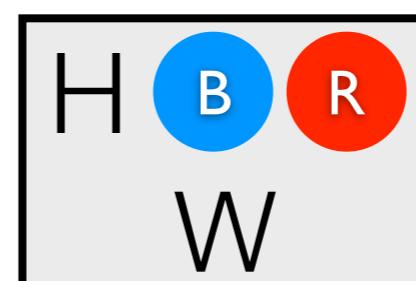
0.024



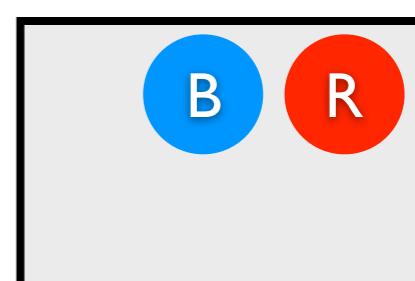
0.036



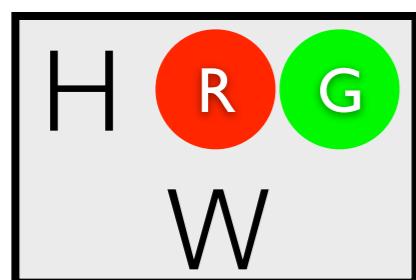
0.056



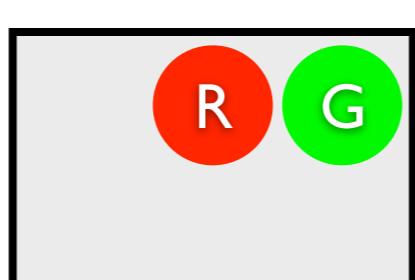
0.084



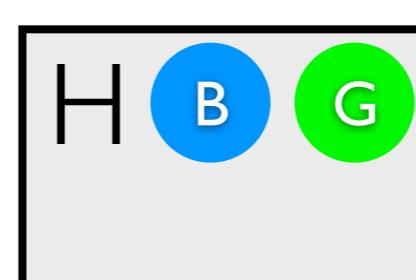
0.036



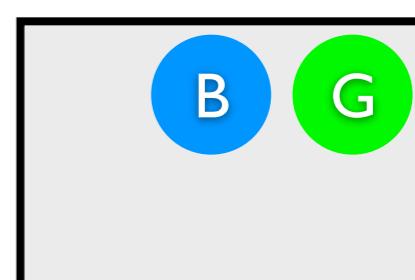
0.054



0.084



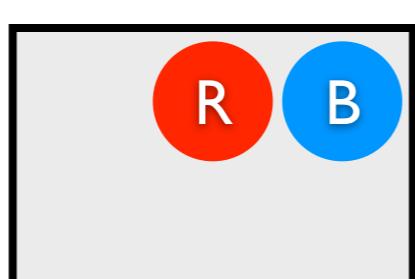
0.126



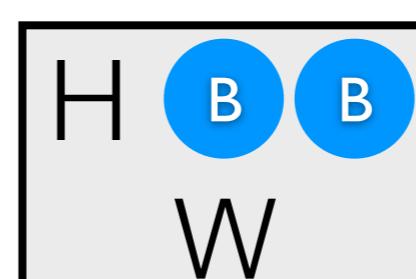
0.060



0.090



0.140



0.210

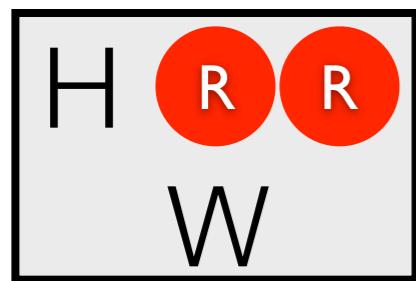


Most likely world

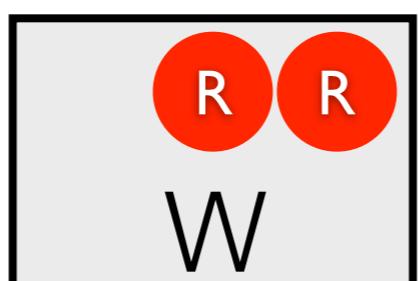
MPE Inference

where **win** is

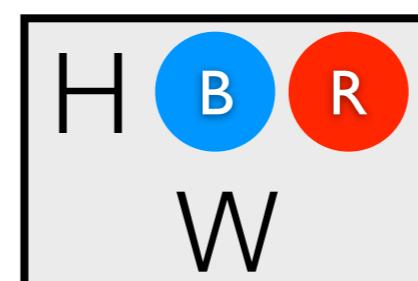
0.024



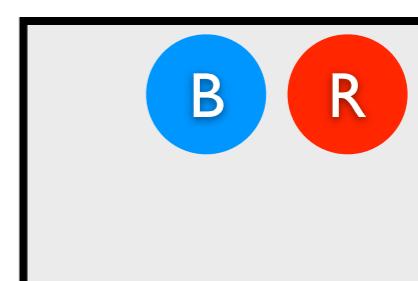
0.036



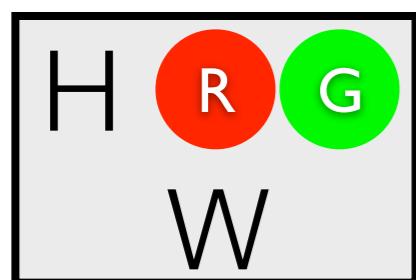
0.056



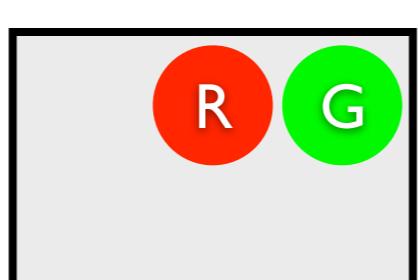
0.084



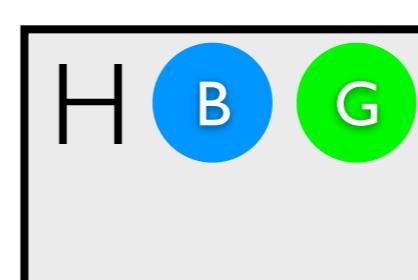
0.036



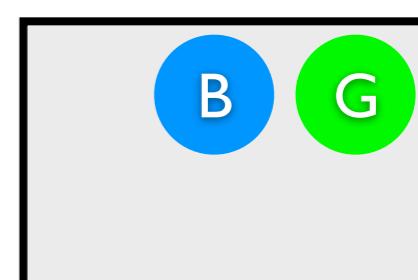
0.054



0.084



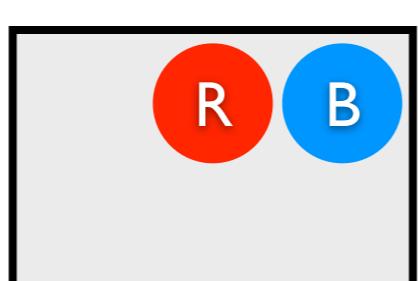
0.126



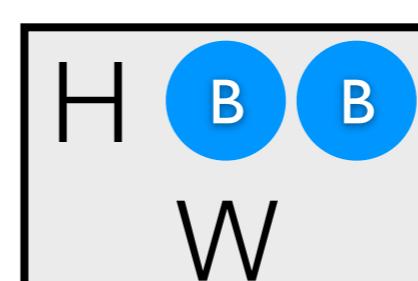
0.060



0.090



0.140

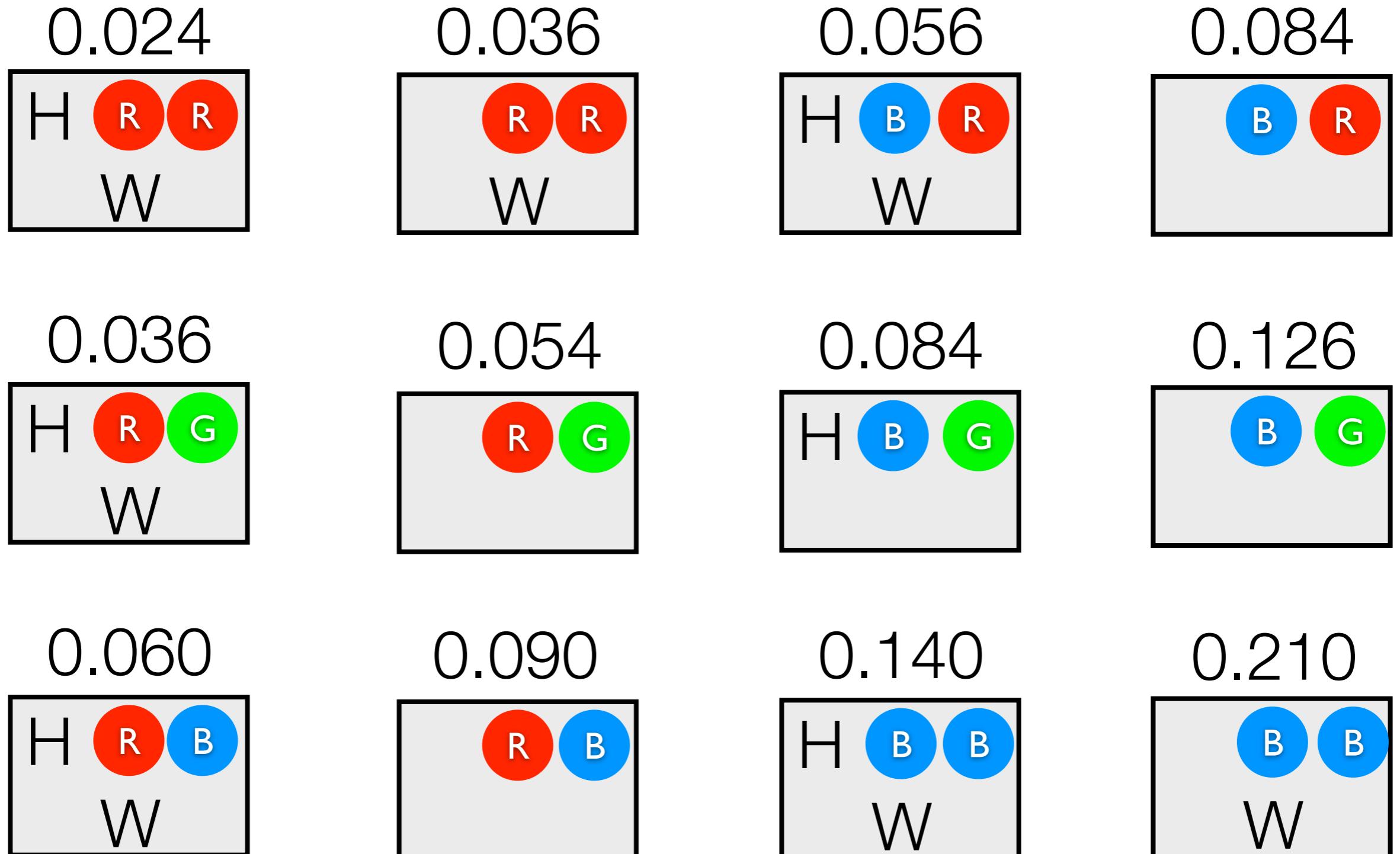


0.210



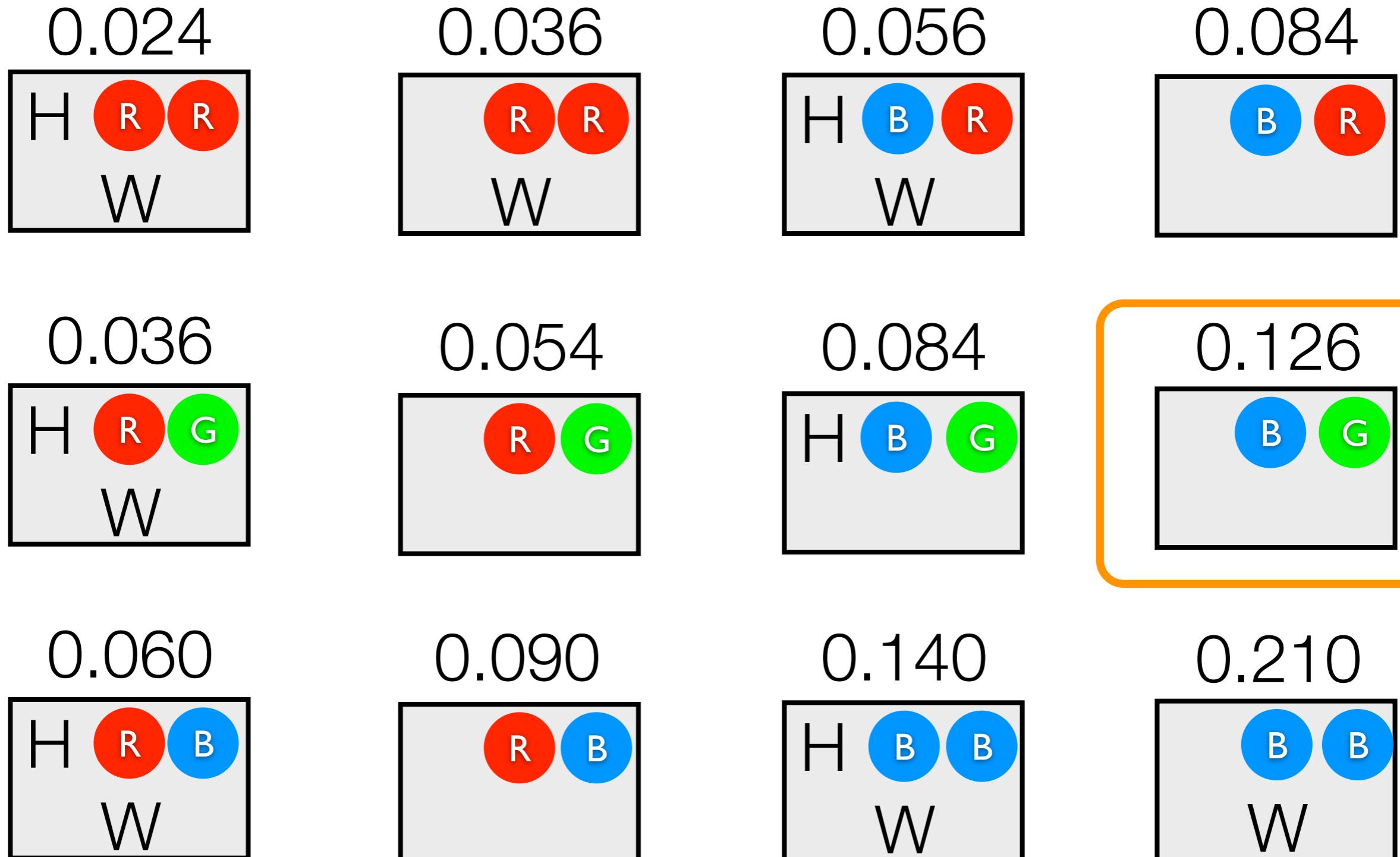
Most likely world where col(2,blue) is false?

MPE Inference



Most likely world where col(2,blue) is false?

MPE Inference



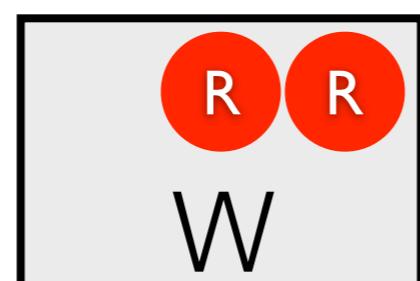
$P(\text{win}) = ?$

Marginal
Probability

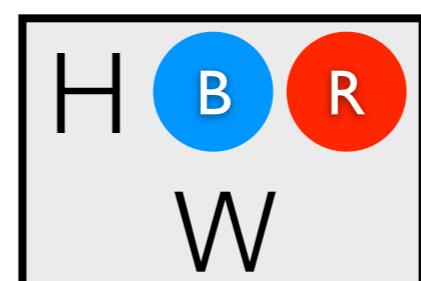
0.024



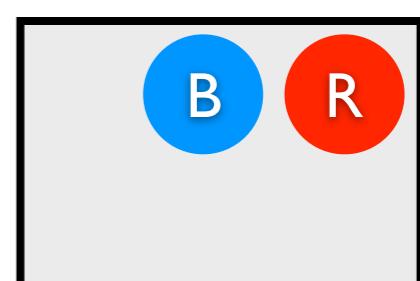
0.036



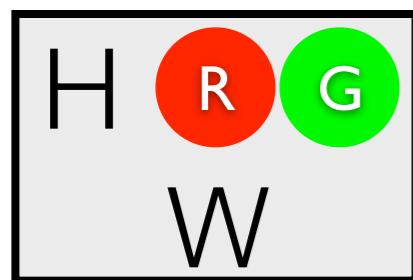
0.056



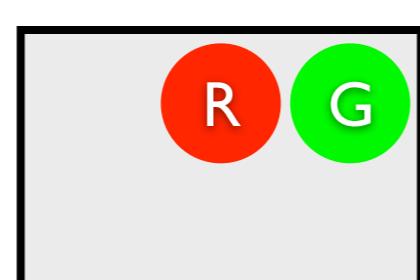
0.084



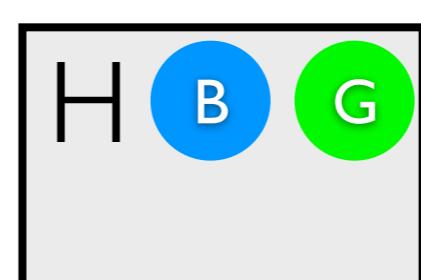
0.036



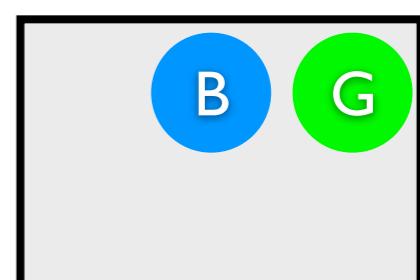
0.054



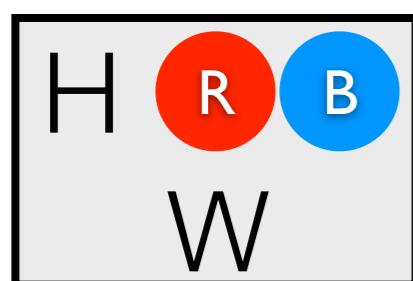
0.084



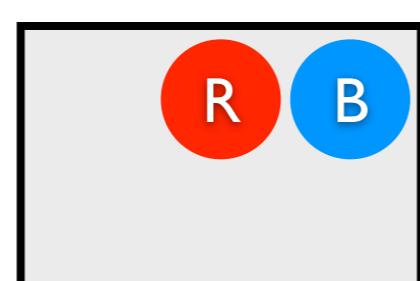
0.126



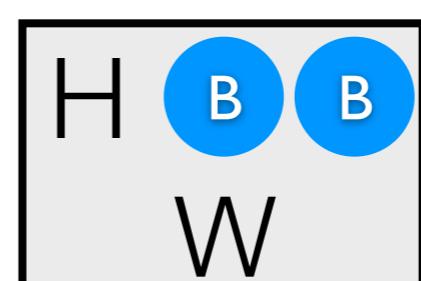
0.060



0.090



0.140



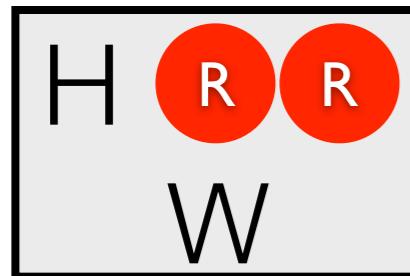
0.210



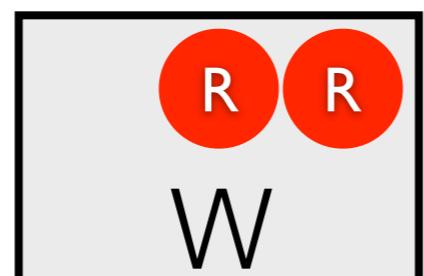
$$P(\underline{\text{win}}) = \sum$$

Marginal
Probability

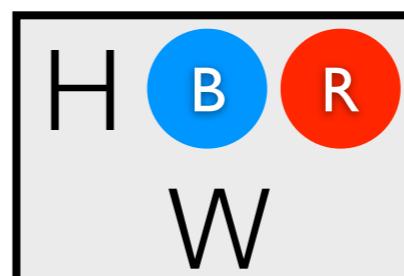
0.024



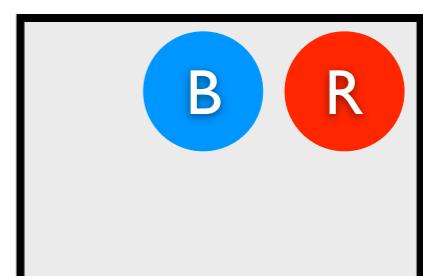
0.036



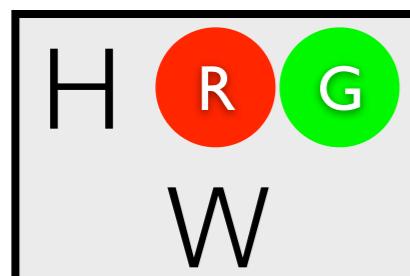
0.056



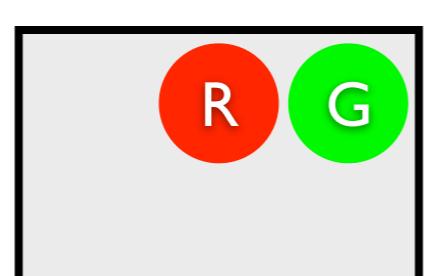
0.084



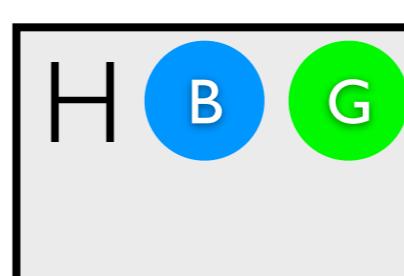
0.036



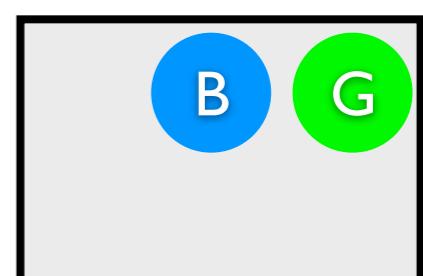
0.054



0.084



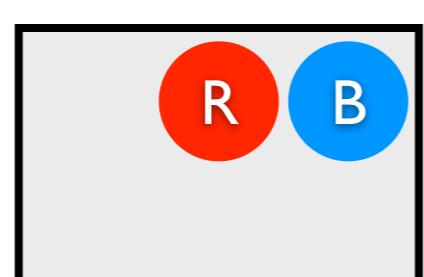
0.126



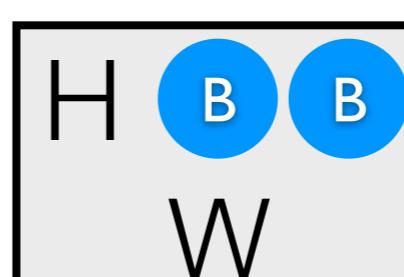
0.060



0.090



0.140



0.210



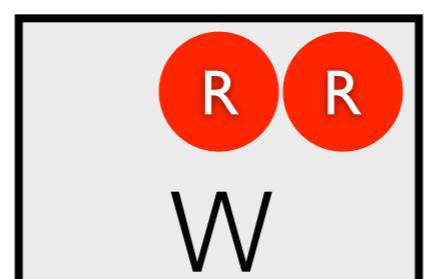
$$\underline{P(\text{win})} = \sum = 0.562$$

Marginal
Probability

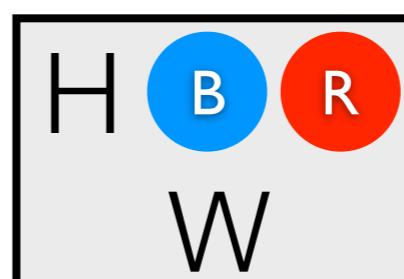
0.024



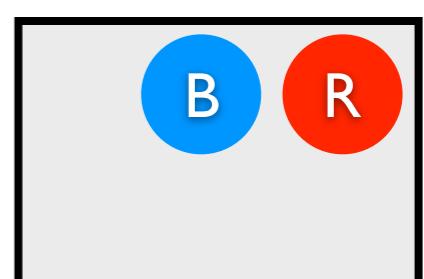
0.036



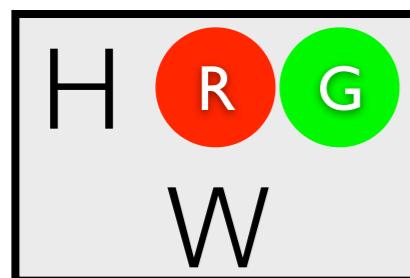
0.056



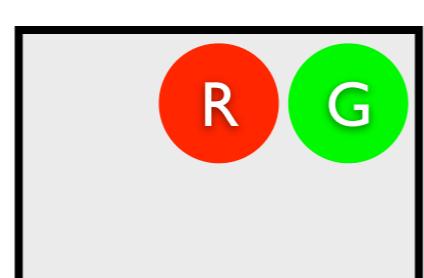
0.084



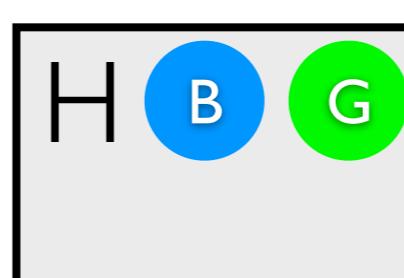
0.036



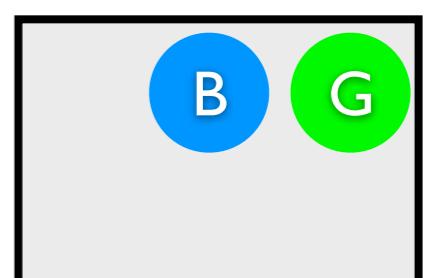
0.054



0.084



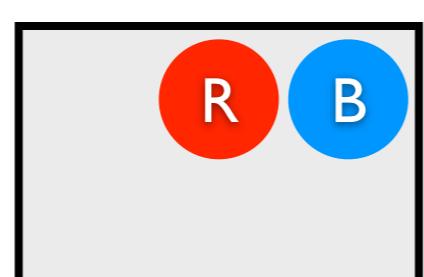
0.126



0.060



0.090



0.140

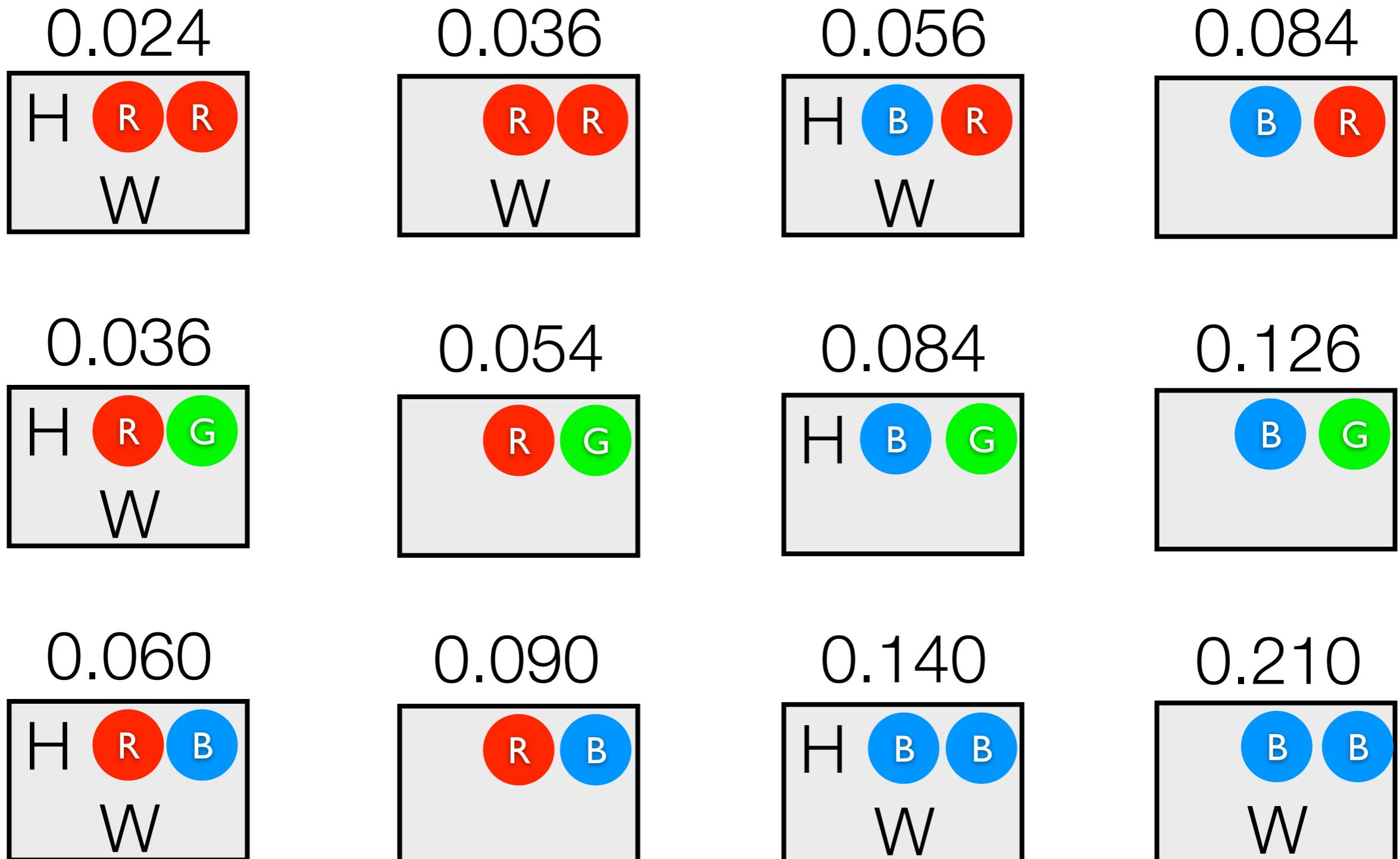


0.210



$P(\text{win}|\text{col}(2,\text{green}))=?$

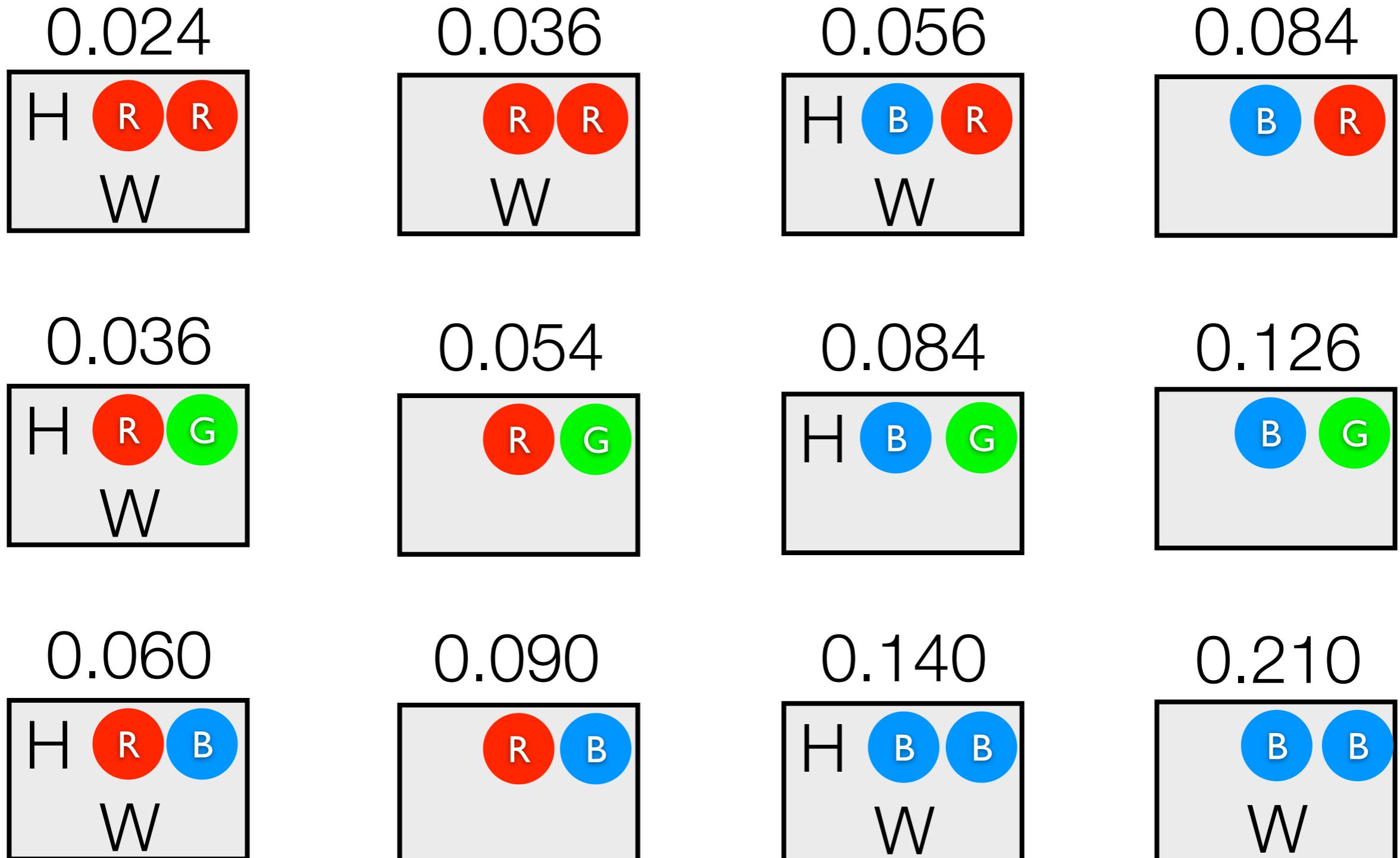
Conditional
Probability



$$P(\text{win} | \underline{\text{col}(2, \text{green})}) = \frac{\sum}{\sum}$$

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

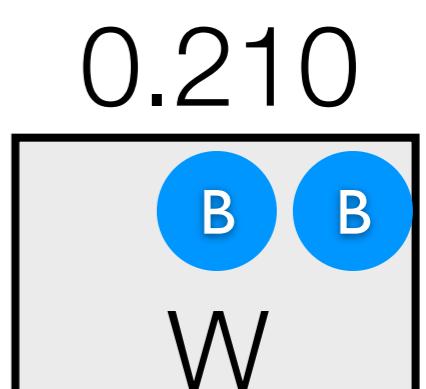
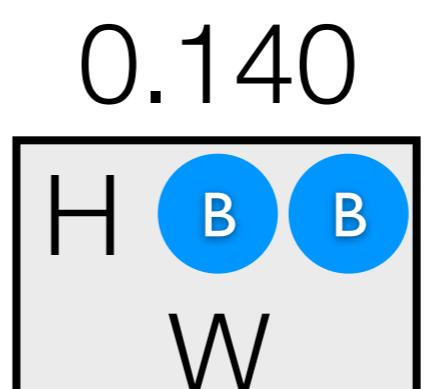
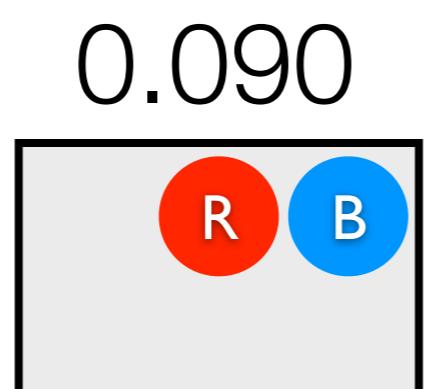
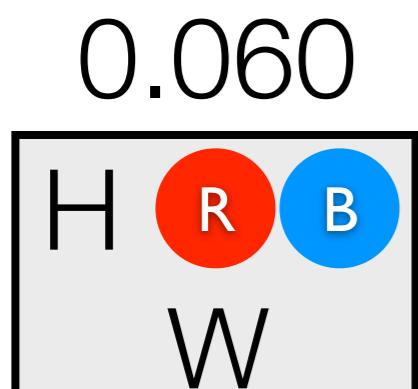
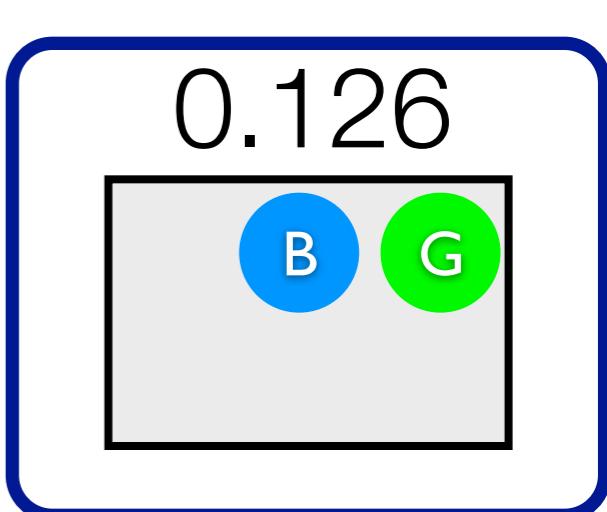
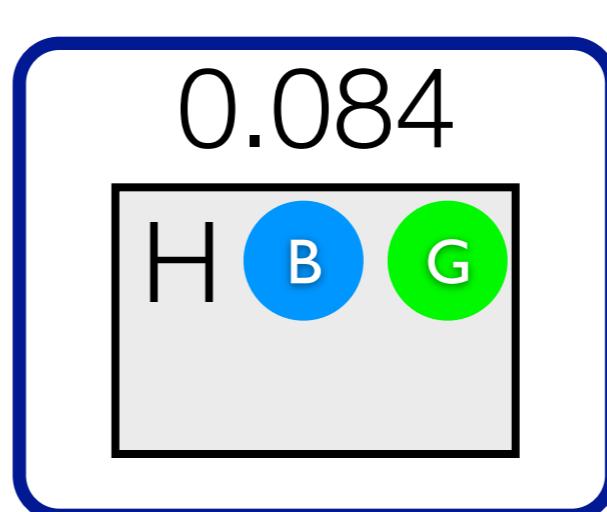
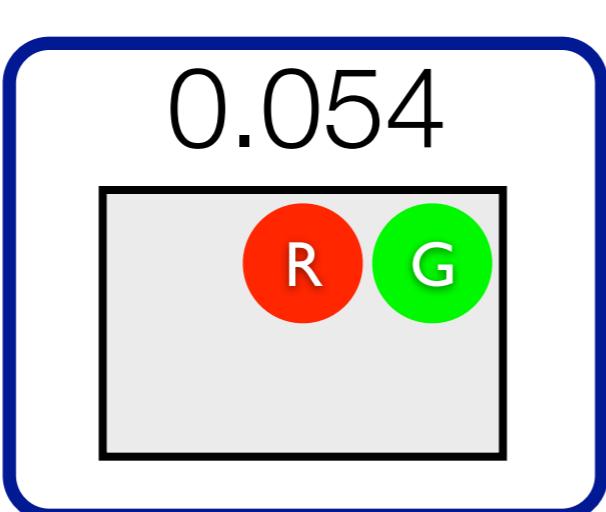
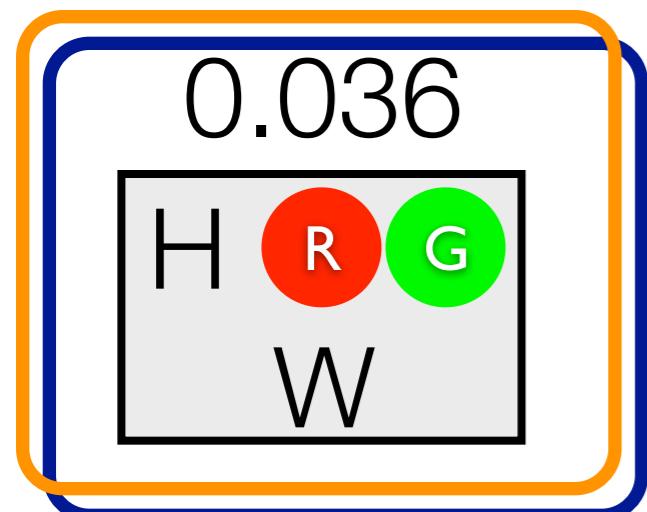
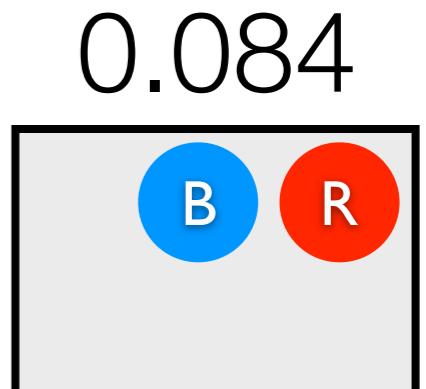
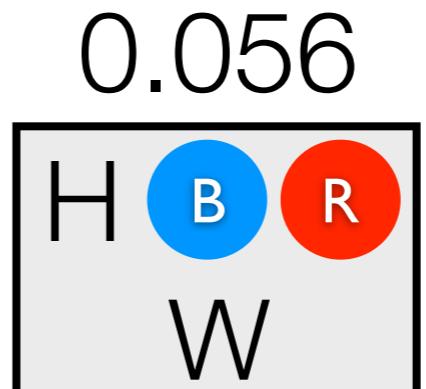
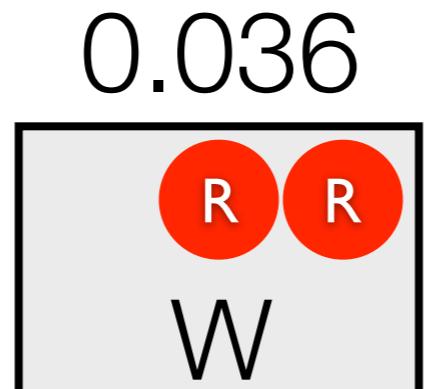
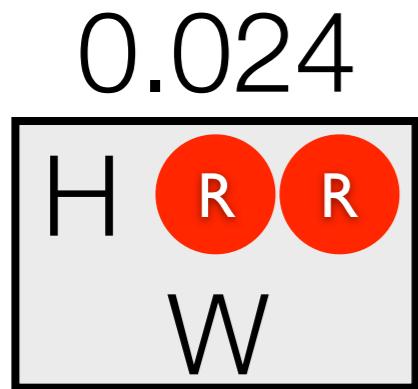
Conditional
Probability



$$P(\text{win} | \underline{\text{col}(2, \text{green})}) = \frac{\sum}{\sum}$$

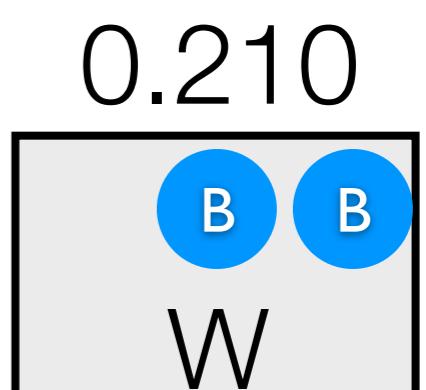
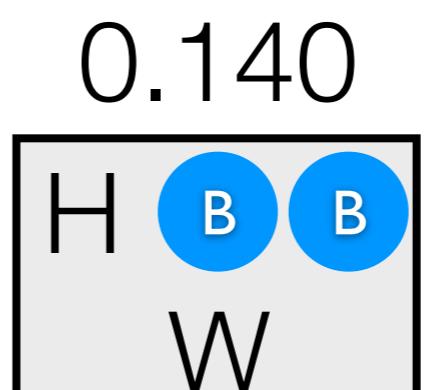
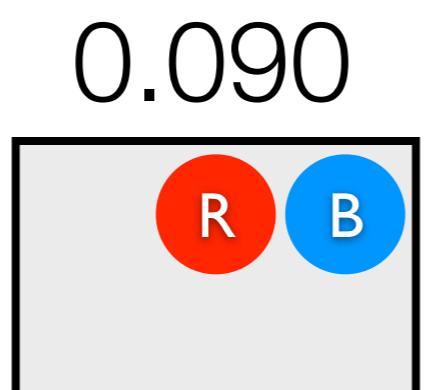
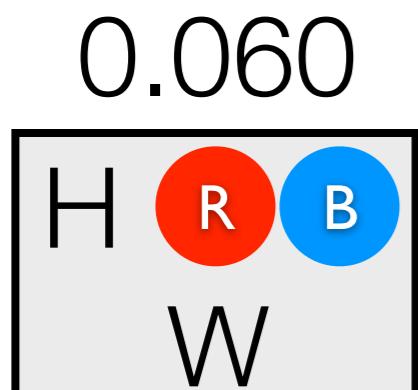
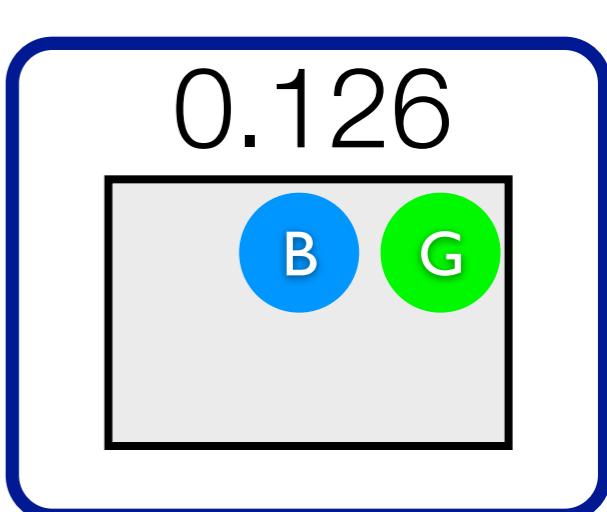
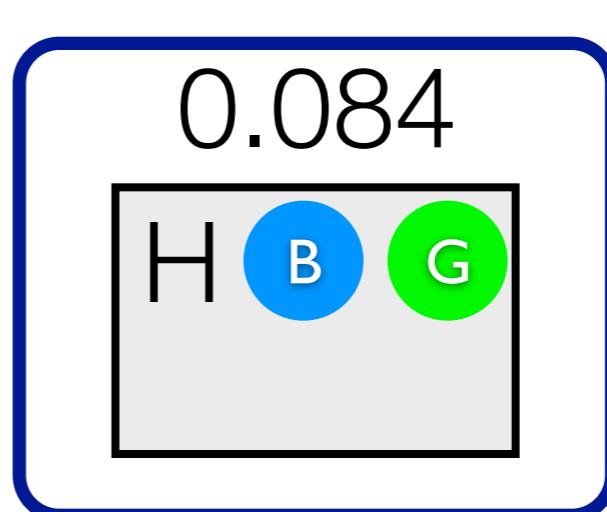
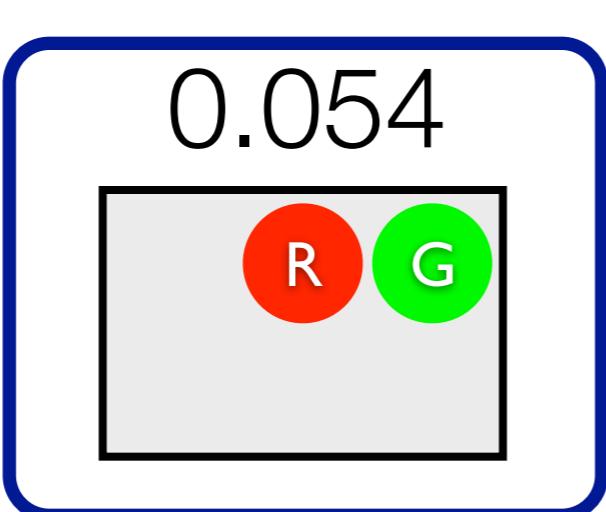
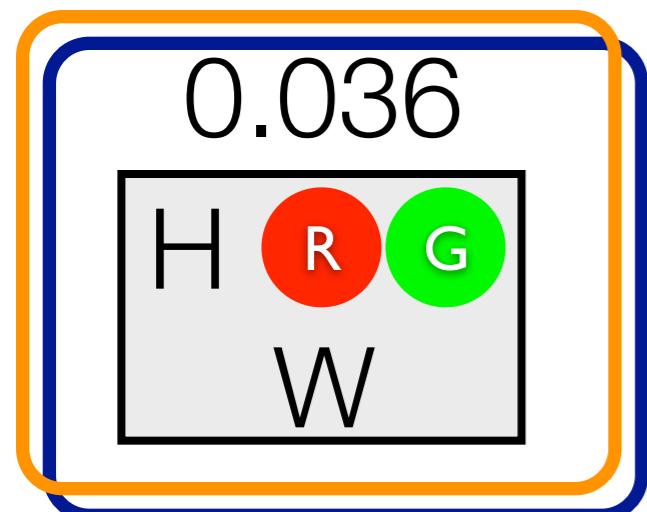
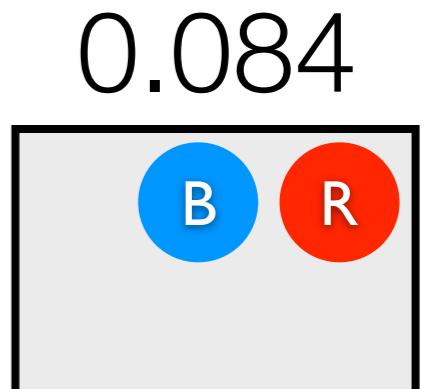
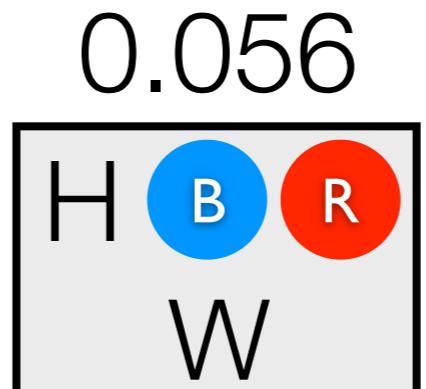
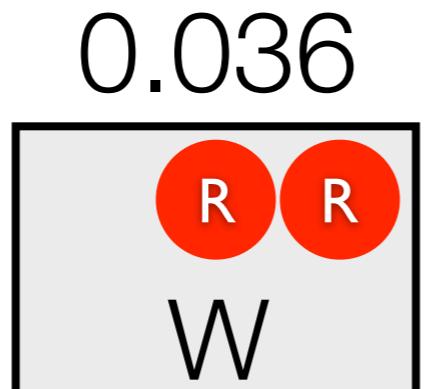
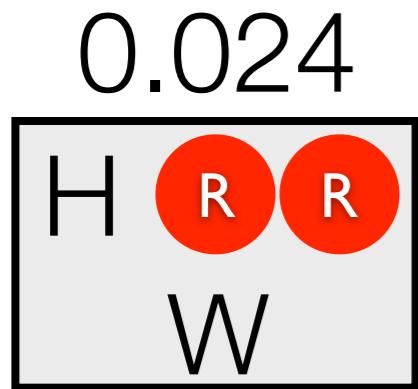
$$= P(\underline{\text{win} \wedge \text{col}(2, \text{green})}) / P(\underline{\text{col}(2, \text{green})}),$$

Conditional
Probability



$$\text{P}(\text{win}|\underline{\text{col}(2,\text{green})}) = \frac{\sum/\Sigma}{\sum} = 0.036/0.3 = 0.12$$

Conditional
Probability

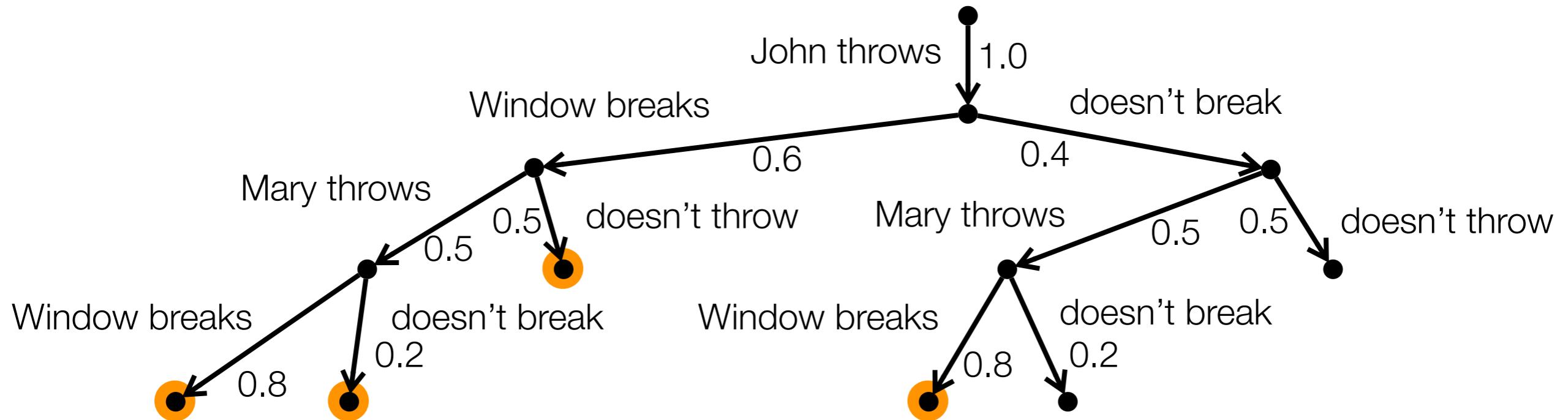


Alternative view: CP-logic

probabilistic causal laws

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

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



$$P(\text{break}) = 0.6 \times 0.5 \times 0.8 + 0.6 \times 0.5 \times 0.2 + 0.6 \times 0.5 + 0.4 \times 0.5 \times 0.8$$

[Vennekens et al 2003, Meert and Vennekens 2014]

KU LEUVEN

Sato's Distribution semantics

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

query

subset of probabilistic facts

Prolog rules

sum over possible worlds where Q is true

Examples from Tutorial

Try yourself:

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

```
$ pip install problog
```

The screenshot shows a web browser window for the ProbLog system at dtai.cs.kuleuven.be. The page title is "Higher-order functions / Meta-predicates — ProbLog: Probabilistic Programming". The navigation bar includes links for ProbLog, Home, Download, Publications, Help, People, Tutorial, and Online.

The main content area contains a code snippet for calculating the total probability of coin flips:

```
1 :- use_module(library(apply)).  
2 :- use_module(library(lists)).  
3 :-  
4 PH::make_coin(C,PH).  
5 coin(C) :- make_coin(C,0.8).  
6 cumsum(C, In, Out) :- (coin(C), Out is In+1; \+coin(C), Out=In).  
7 total(S) :-  
8 .... findall(X, between(1,10,X), L),  
9 .... foldl(cumsum, L, 0, S).  
10 :-  
11 query(total(_)).
```

A blue "Evaluate" button is located to the right of the code editor.

Below the code editor is a table titled "Query ▾" showing the results of the query:

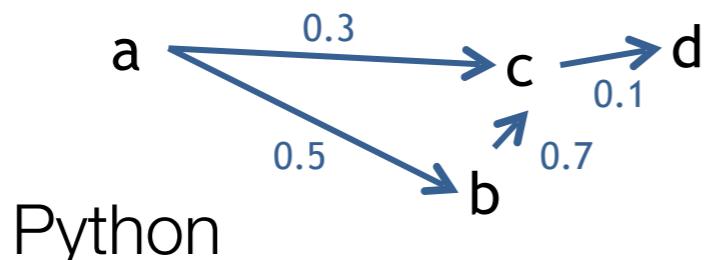
Query	Location	Probability
total(0)	11:7	1.024e-
total(1)	11:7	4.096e-
total(2)	11:7	7.3728e-
total(3)	11:7	0.000786432
total(4)	11:7	0.005505024
total(5)	11:7	0.026424115

Probabilistic Graph

Discrete probabilistic reachability program

Logic Programming

```
path(X, Y) :- edge(X, Y).  
path(X, Y) :- edge(X, Z),  
             path(Z, Y).  
P::edge(X, Y) :- ...random vars...
```



Python

```
edges = {  
    'a': [(0.9, 'b'), (0.3, 'c')], 'b': [(0.4, 'c')], 'c': [(0.5, 'd')]  
}
```

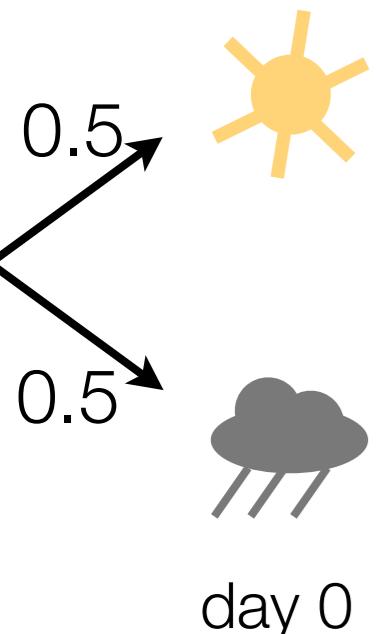
```
def path(start, end, visited=()):  
    if start == end: return True  
    if start in visited: return False  
    return any((path(nxt, end, visited+(start,)) for pr, nxt in  
               edges[start] if memchoice((pr, (start, nxt)))))
```

Functional Program (Scala-like)

```
def path(start, end, visited=List())={  
    if(start == end)  
        return true  
    if(visited.contains(start))  
        return false  
    return start.neighbors.exists{  
        path(_,end,(visited+start))  
    }  
}  
nodeA.neighbors = pr(...random vars...)  
nodeB.neighbors = pr(...random vars...)
```

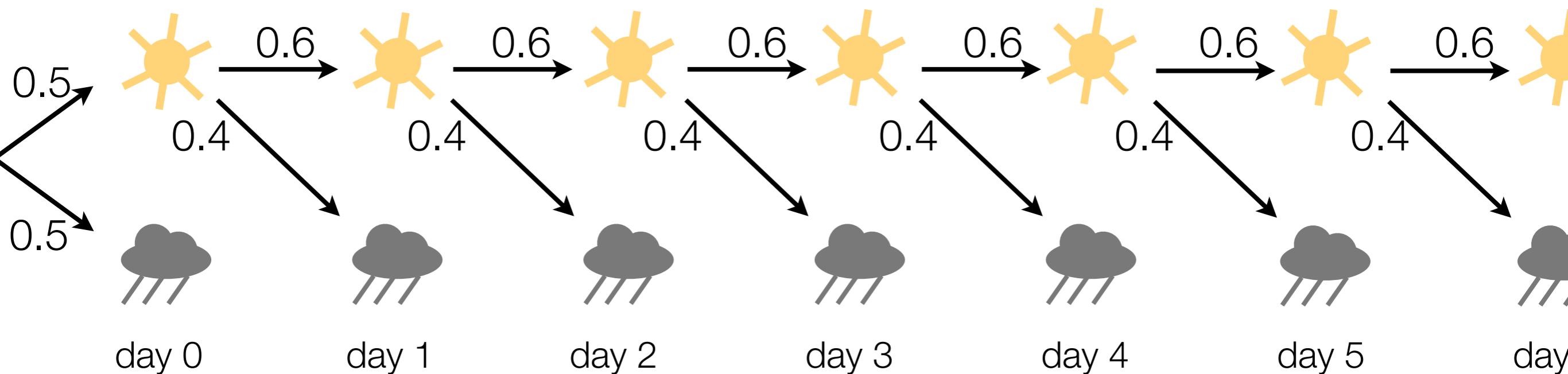
ProbLog by example: Rain or sun?

ProbLog by example: Rain or sun?



```
0.5::weather(sun,0) ; 0.5::weather(rain,0) :- true.
```

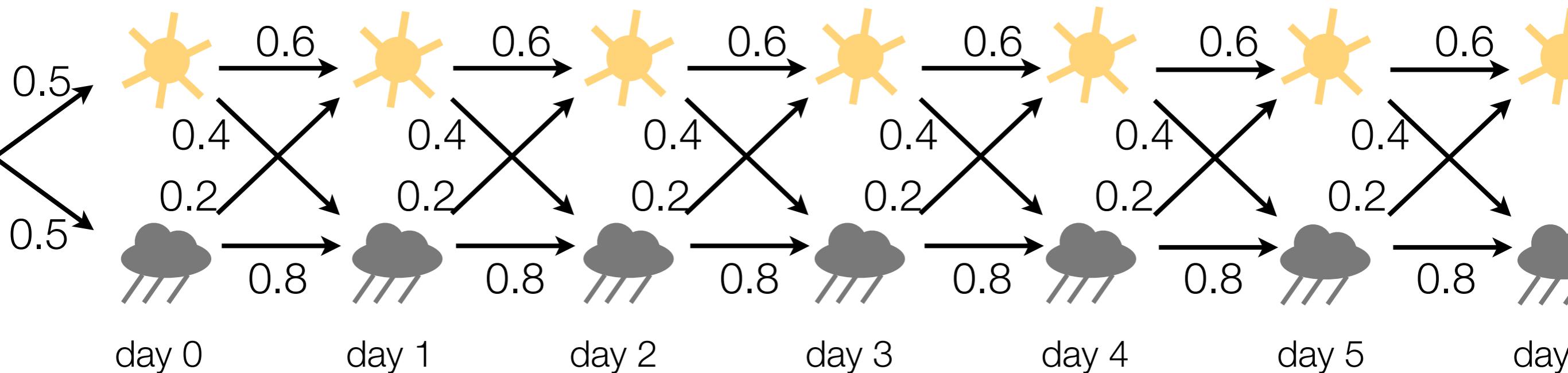
ProbLog by example: Rain or sun?



```
0.5::weather(sun,0) ; 0.5::weather(rain,0) :- true.
```

```
0.6::weather(sun,T) ; 0.4::weather(rain,T)  
:- T>0, Tprev is T-1, weather(sun,Tprev).
```

ProbLog by example: Rain or sun?

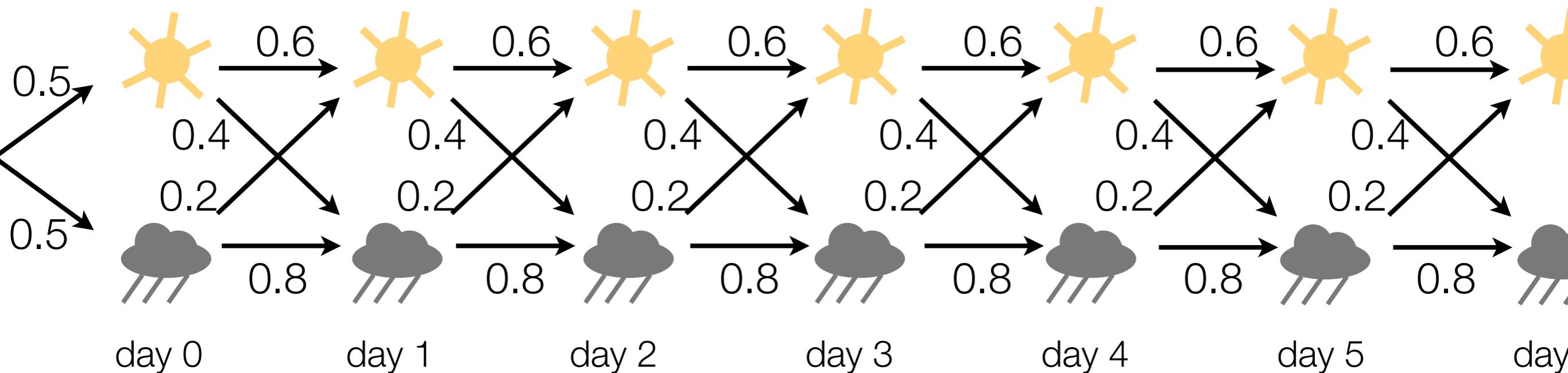


```
0.5::weather(sun,0) ; 0.5::weather(rain,0) :- true.
```

```
0.6::weather(sun,T) ; 0.4::weather(rain,T)  
:- T>0, Tprev is T-1, weather(sun,Tprev).
```

```
0.2::weather(sun,T) ; 0.8::weather(rain,T)  
:- T>0, Tprev is T-1, weather(rain,Tprev).
```

ProbLog by example: Rain or sun?



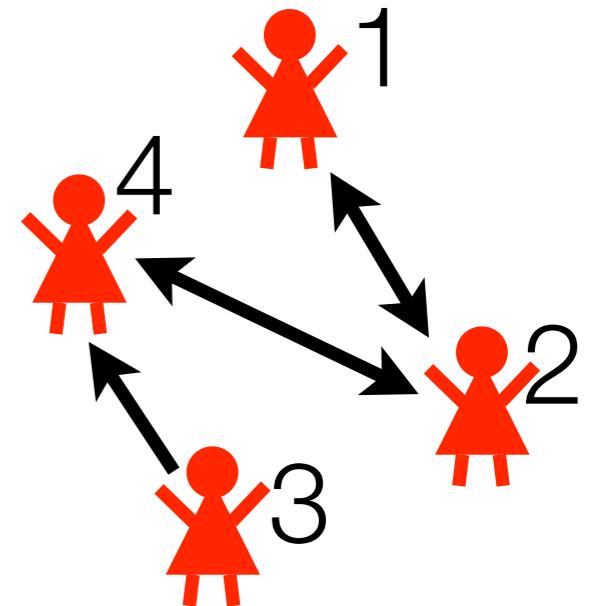
```
0.5::weather(sun,0) ; 0.5::weather(rain,0) :- true.
```

```
0.6::weather(sun,T) ; 0.4::weather(rain,T)
    :- T>0, Tprev is T-1, weather(sun,Tprev).
```

```
0.2::weather(sun,T) ; 0.8::weather(rain,T)
    :- T>0, Tprev is T-1, weather(rain,Tprev).
```

infinite possible worlds! BUT: finitely many partial worlds suffice to answer any given ground query

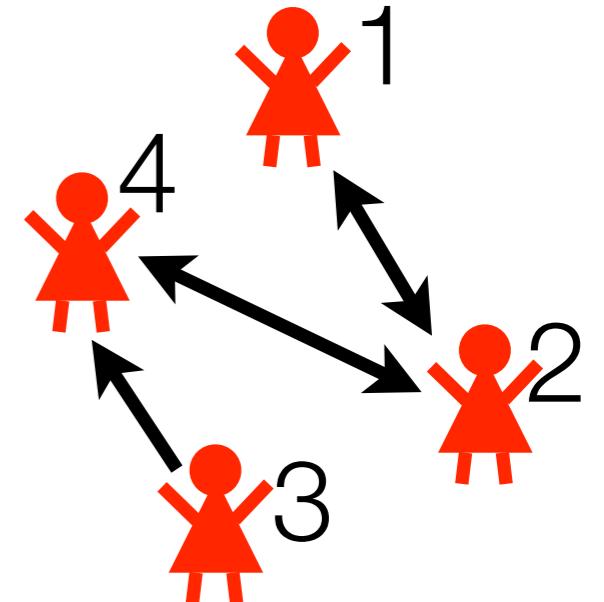
ProbLog by example: Friends & smokers



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

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

ProbLog by example: Friends & smokers



typed probabilistic facts

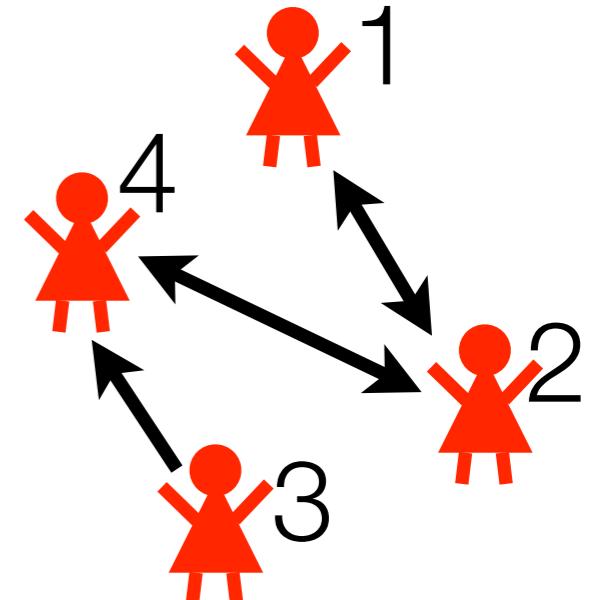
= a probabilistic fact for each grounding

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

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

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

ProbLog by example: Friends & smokers



typed probabilistic facts

= a probabilistic fact for each grounding

0.3::stress(X) :- person(X) .

0.2::influences(X,Y) :-
 person(X), person(Y) .

0.3::stress(1) .

0.3::stress(2) .

0.3::stress(3) .

0.3::stress(4) .

0.2::influences(1,1) .

0.2::influences(1,2) .

0.2::influences(1,3) .

0.2::influences(1,4) .

0.2::influences(2,1) .

...

0.2::influences(4,2) .

0.2::influences(4,3) .

0.2::influences(4,4) .

person(1) .

person(2) .

person(3) .

person(4) .

friend(1,2) .

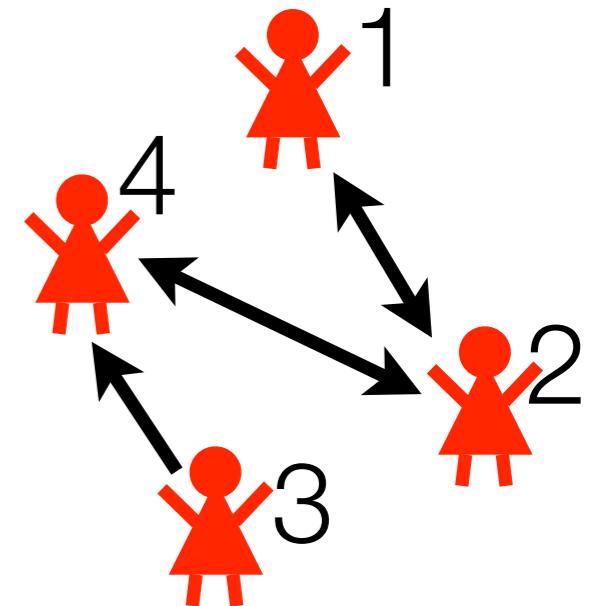
friend(2,1) .

friend(2,4) .

friend(3,4) .

friend(4,2) .

ProbLog by example: Friends & smokers

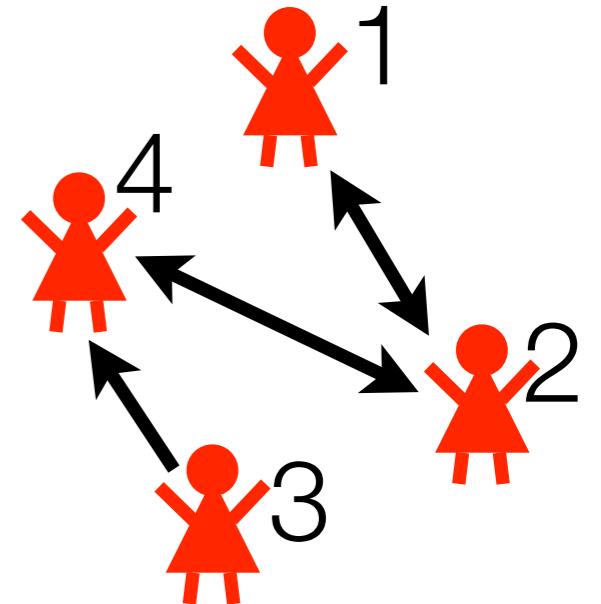


```
0.3::stress(X) :- person(X).  
0.2::influences(X,Y) :-  
    person(X), person(Y).
```

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

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

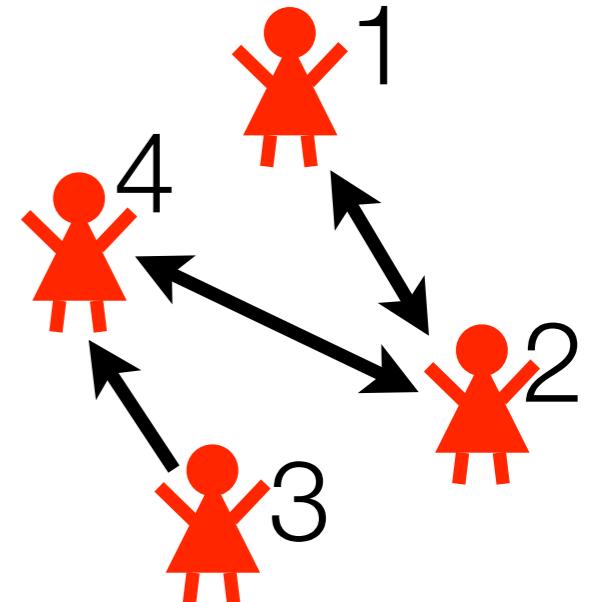
ProbLog by example: Friends & smokers



```
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).
```

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

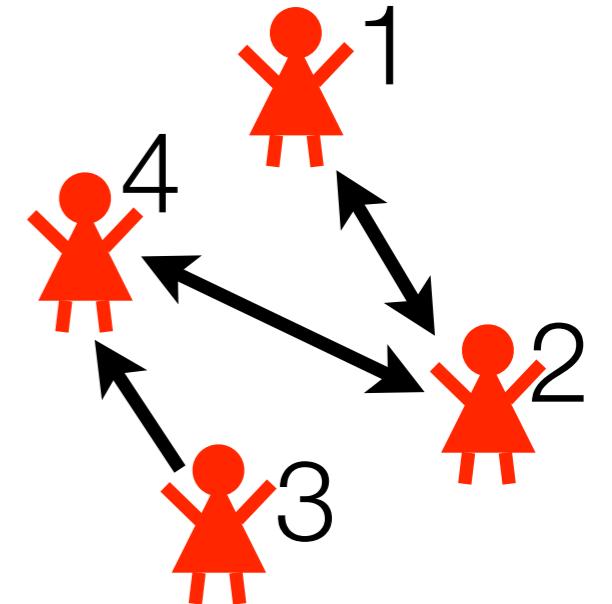
ProbLog by example: Friends & smokers



```
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).
```

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

ProbLog by example: Friends & smokers

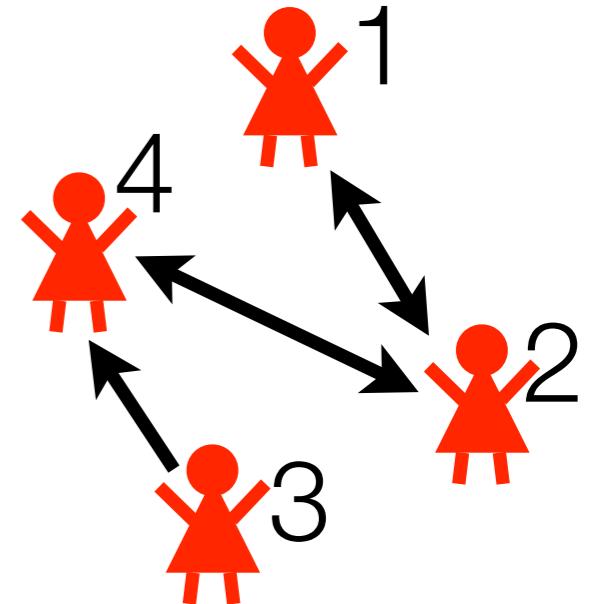


```
0.3::stress(X) :- person(X).  
0.2::influences(X,Y) :-  
    person(X), person(Y).  
  
smokes(X) :- stress(X).  
smokes(X) :-  
    friend(X,Y), influences(Y,X), smokes(Y).  
  
0.4::asthma(X) :- smokes(X).
```

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

annotated disjunction with implicit head atom: with probability 0.6, nothing happens

ProbLog by example: Friends & smokers



```
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).  
  
0.4::asthma(X) :- smokes(X).
```

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

Higher-order functions

```
: - use_module(library(apply)).  
: - use_module(library(lists)).  
  
PH::make_coin(C, PH).  
coin(C) :- make_coin(C, 0.8).  
cumsum(C, In, Out) :- (coin(C), Out is In+1; \+coin(C), Out=In).  
total(S) :-  
    findall(X, between(1,10,X), L),  
    foldl(cumsum, L, 0, S).  
  
query(total(_)).
```

Higher-order functions

```
: - use_module(library(lists)).  
:- use_module(library(functions)).  
  
PH::make_coin(C).  
coin(C) :- make_coin(C).  
cumsum([H|T], In, Out) :- cumsum(T, In, Out), !, Out is H + Out.  
cumsum([], In, Out) :- !, Out is In.  
total(S) :- findall(X, coin(X), L), foldl(cumsum, L, 0, S).  
  
query(total(_))
```

Query ▼	Location	Probability
total(0)	11:7	1.024e-
total(1)	11:7	4.096e-
total(2)	11:7	7.3728e-
total(3)	11:7	0.000786432
total(4)	11:7	0.005505024
total(5)	11:7	0.026424115
total(6)	11:7	0.088080384
total(7)	11:7	0.20132659
total(8)	11:7	0.30198989
total(9)	11:7	0.26843546
total(10)	11:7	0.10737418

In).

ProbLog

- **probabilistic choices** + their **consequences**
- probability distribution over **possible worlds**
- how to efficiently answer **questions**?
 - most probable world (MPE inference)
 - probability of query (computing marginals)
 - probability of query given evidence

Summary: ProbLog Syntax

- input database: ground facts
`person(bob) .`
- probabilistic facts
`0.5::stress(bob) .`
- flexible probabilities
`P::pack(Item) :- weight(Item,W),
P is 1.0/W.`
- annotated disjunctions
`0.4::asthma(X) :- smokes(X) .`
- Prolog clauses
`0.5::weather(sun,0) ; 0.5::weather(rain,0) :- true.`
- Prolog clauses
`smokes(X) :- influences(Y,X), smokes(Y) .`
- Prolog clauses
`excess([I|R],Limit) :- \+pack(I), excess(R,Limit) .`

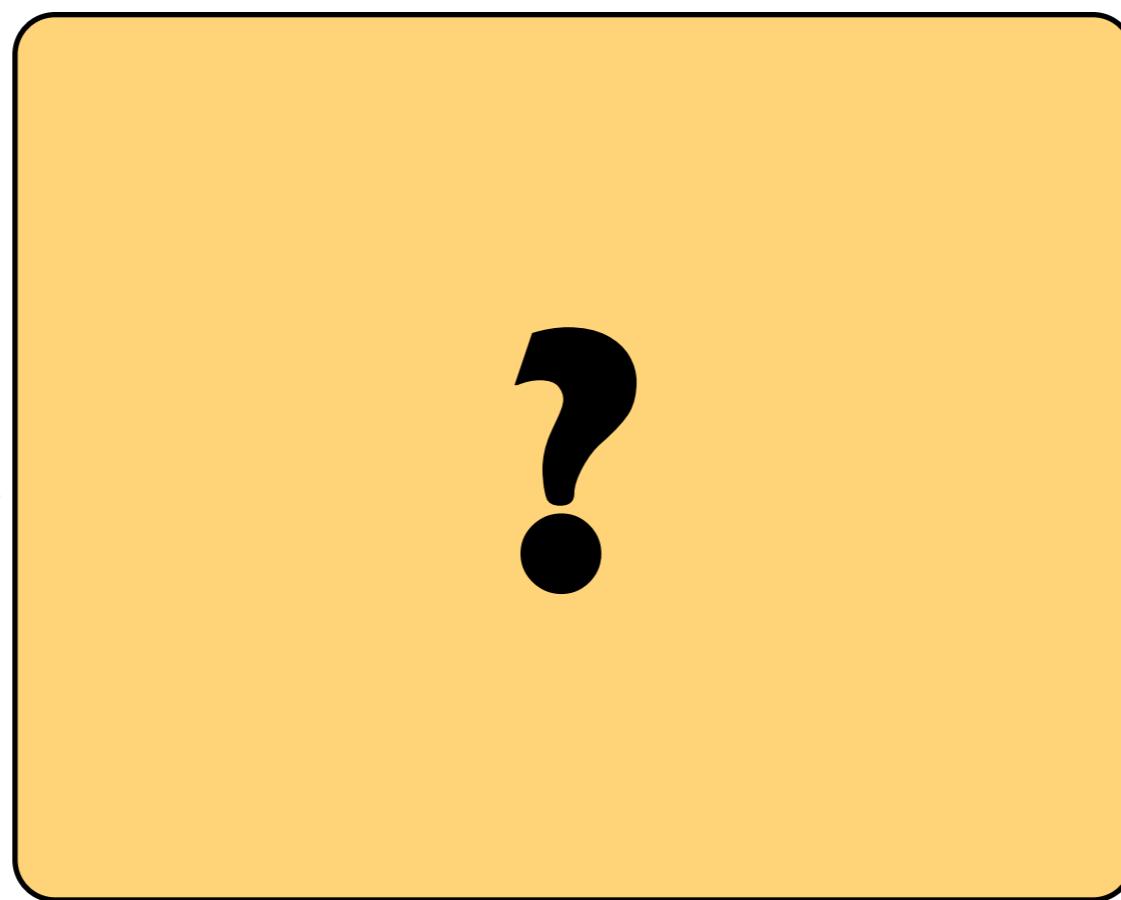
Answering Questions

Given:

program

queries

evidence



Find:

marginal
probabilities

conditional
probabilities

MPE state

Answering Questions

Given:

program

queries

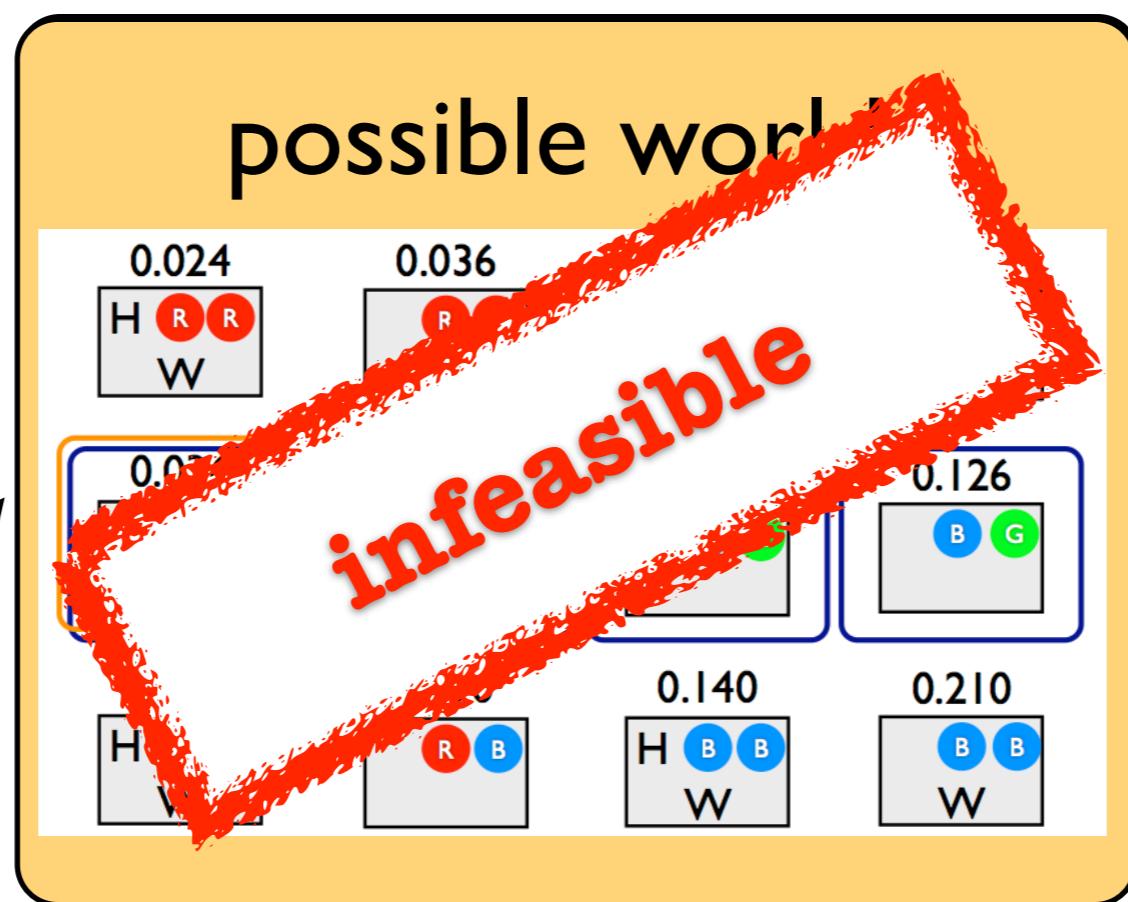
evidence

Find:

marginal probabilities

conditional probabilities

MPE state



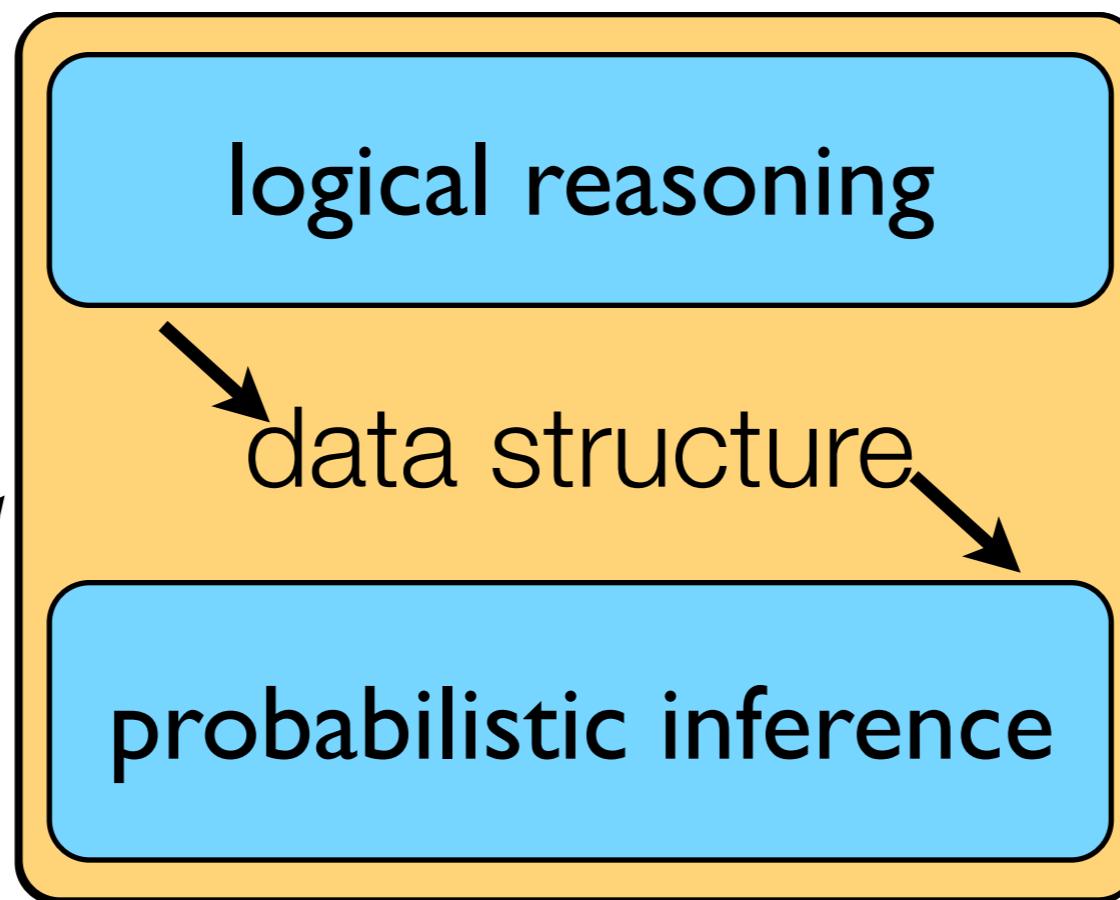
Answering Questions

Given:

program

queries

evidence



Find:

marginal
probabilities

conditional
probabilities

MPE state

Answering Questions

1. using proofs
2. using models

Given:

program

queries

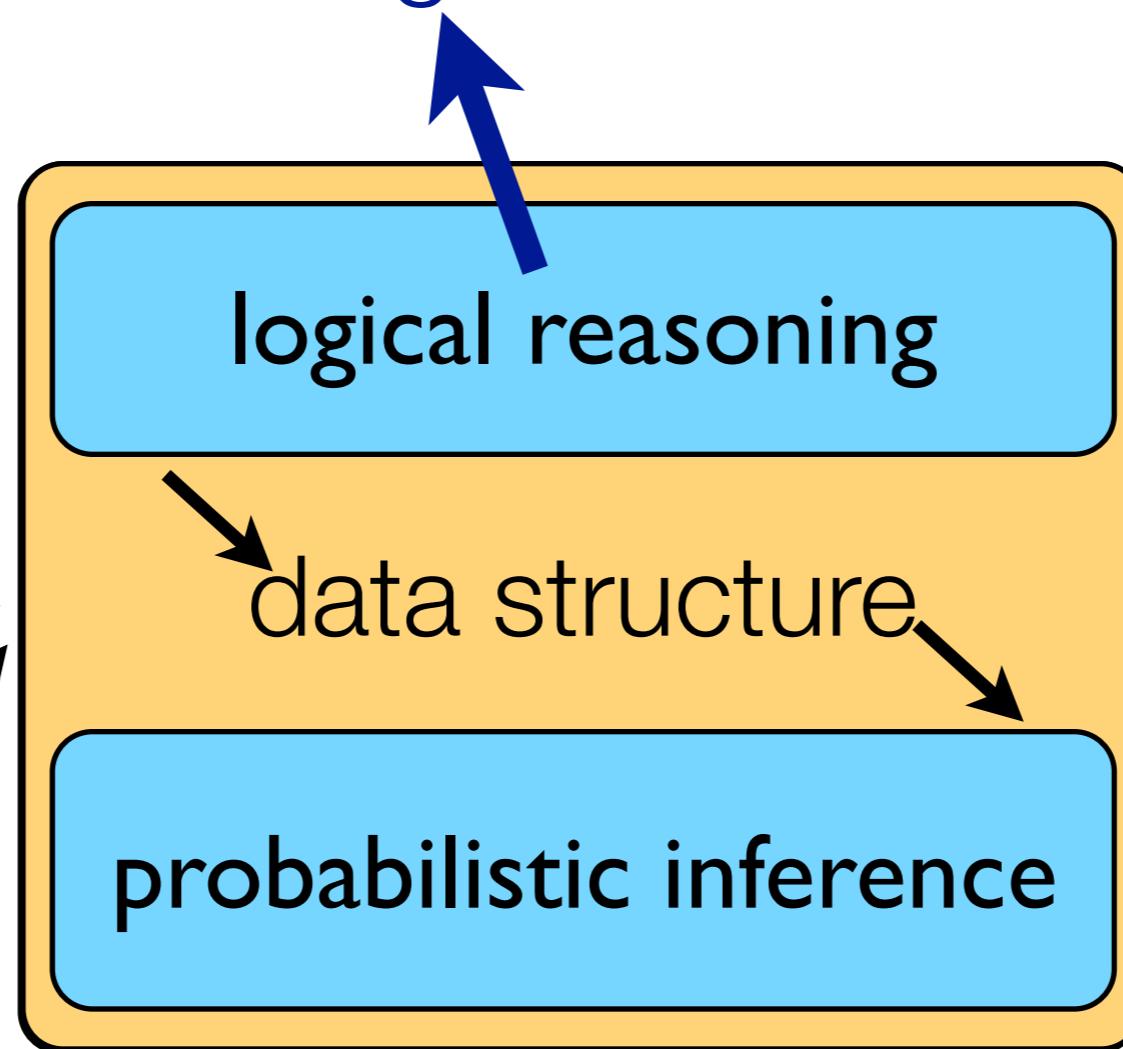
evidence

Find:

marginal
probabilities

conditional
probabilities

MPE state



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

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

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

Logical Reasoning: Proofs in Prolog

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

```
?- stress(carl) .
```

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

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

Logical Reasoning: Proofs in Prolog

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

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

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

Logical Reasoning: Proofs in Prolog

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

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

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

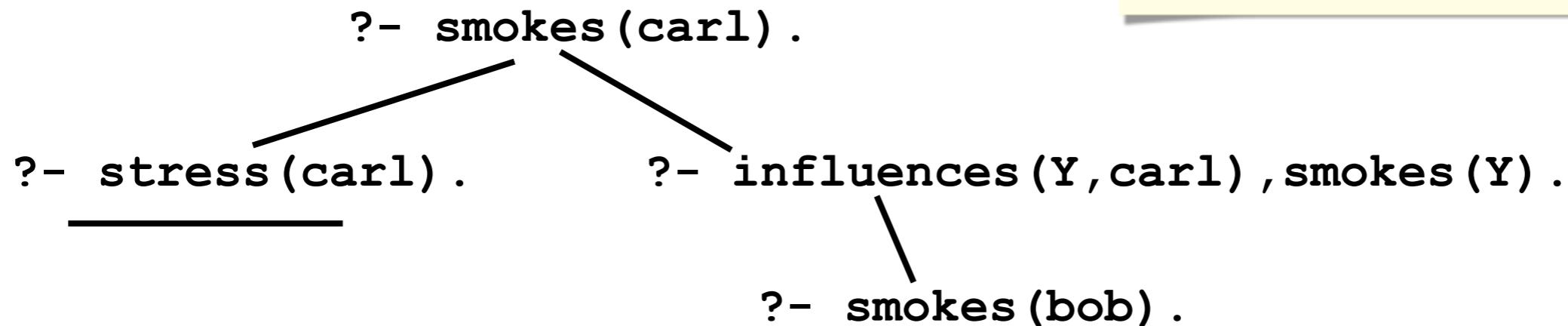
Logical Reasoning: Proofs in Prolog

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

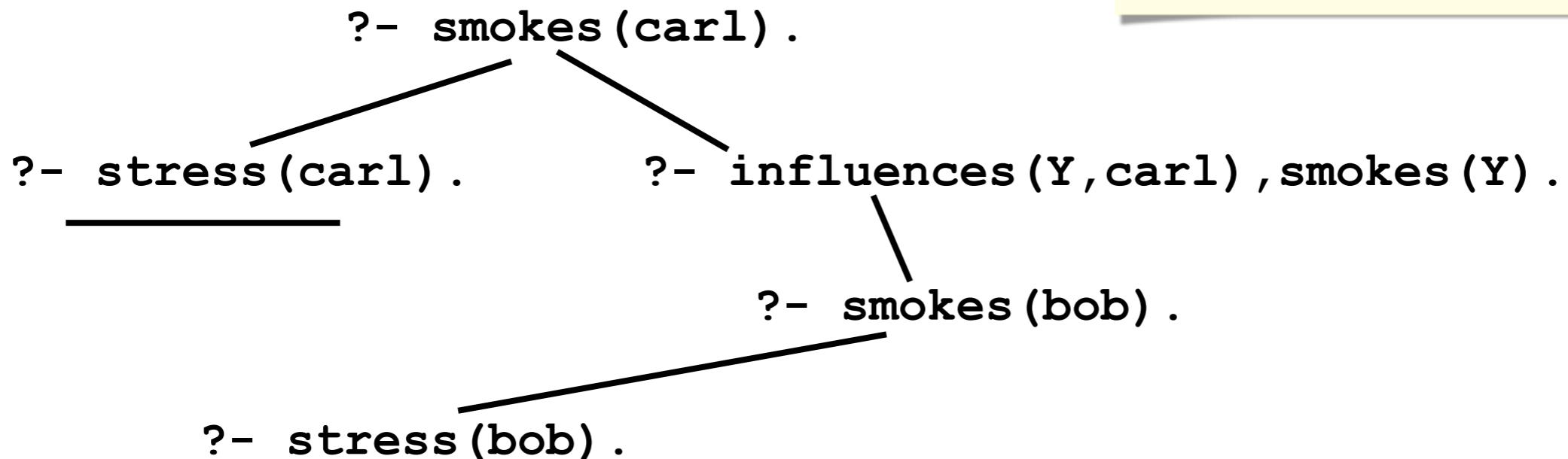
smokes(X) :- stress(X).

smokes(X) :-

influences(Y,X), smokes(Y).



Logical Reasoning: Proofs in Prolog



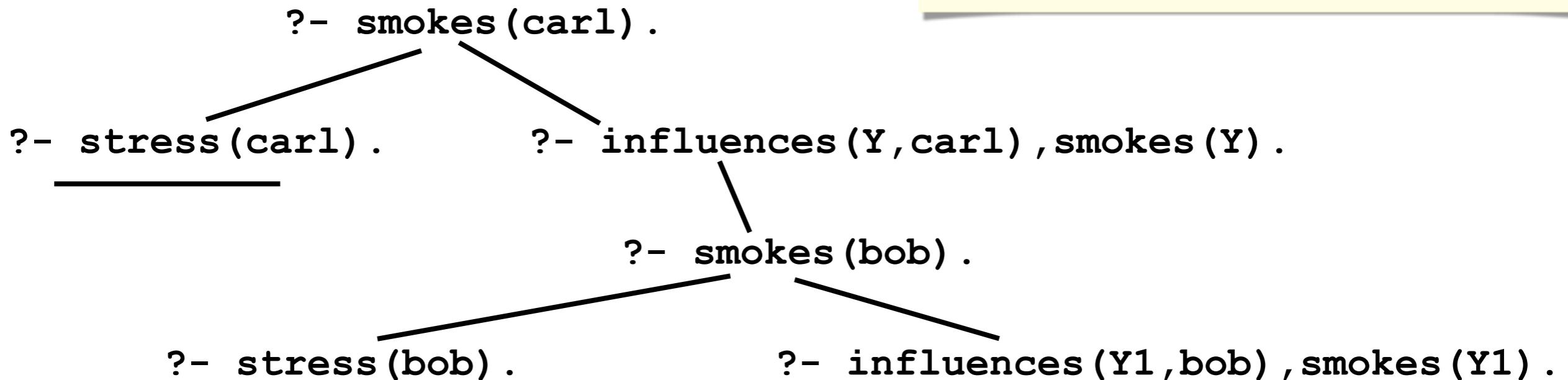
stress(ann).
influences(ann,bob).
influences(bob,carl).

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

Logical Reasoning: Proofs in Prolog

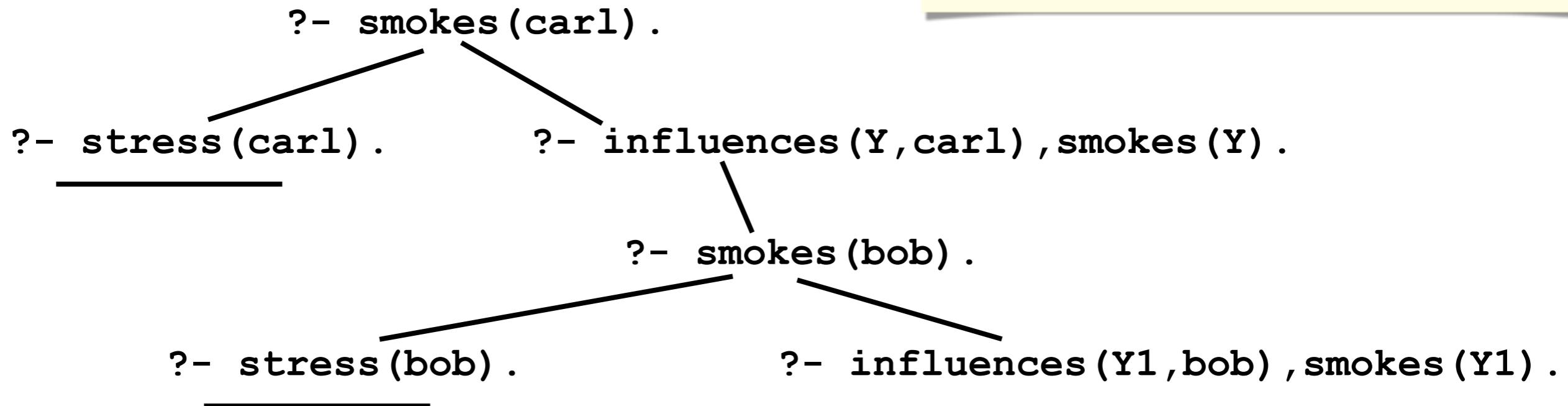
```
stress(ann).  
influences(ann,bob).  
influences(bob,carl).
```

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



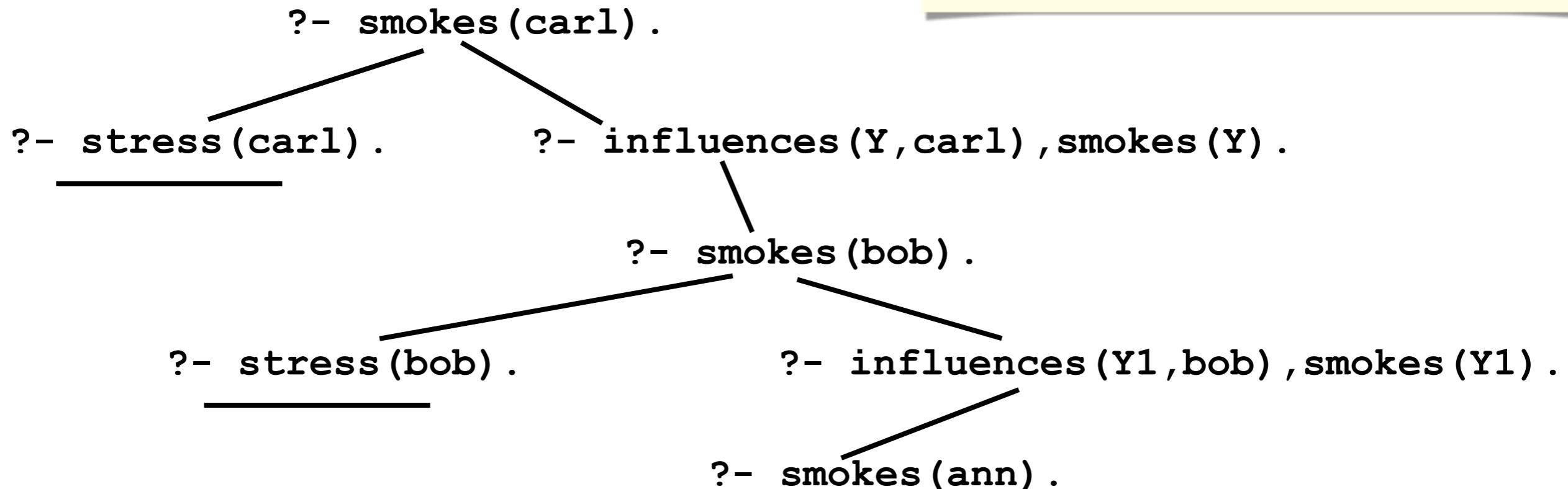
Logical Reasoning: Proofs in Prolog

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



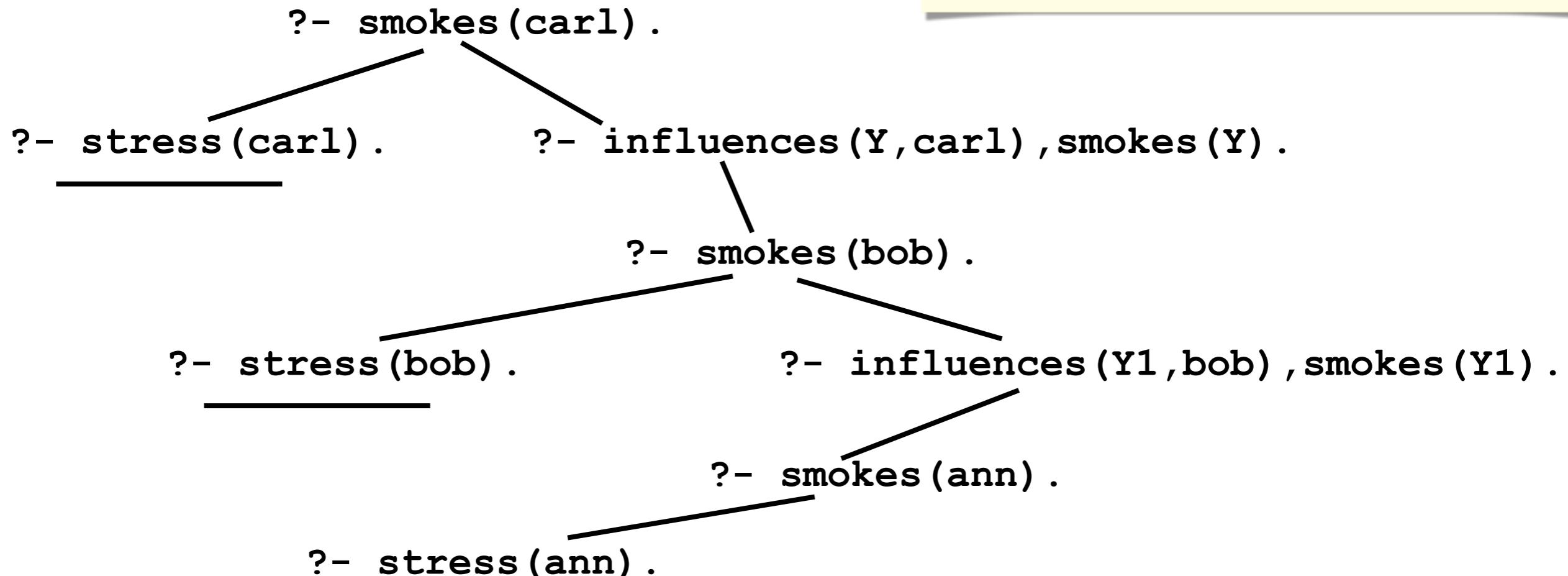
Logical Reasoning: Proofs in Prolog

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



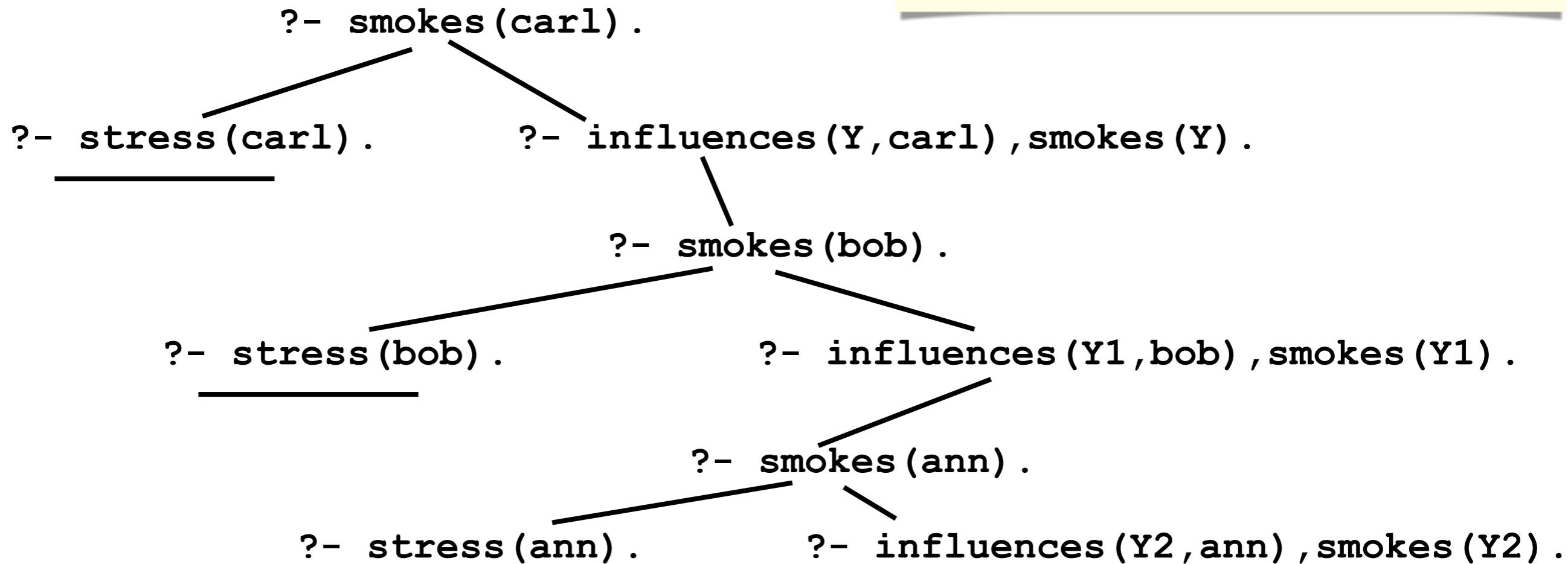
Logical Reasoning: Proofs in Prolog

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



Logical Reasoning: Proofs in Prolog

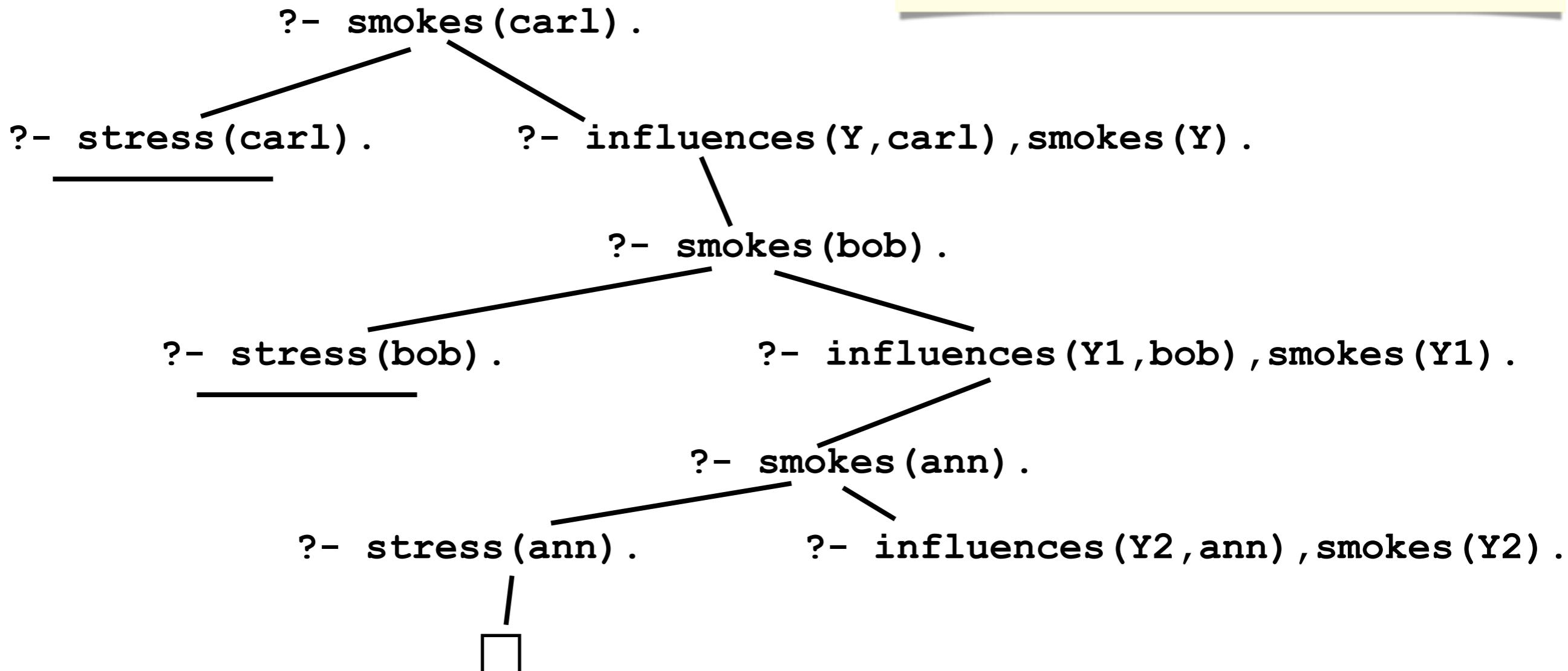
```
stress(ann).  
influences(ann,bob).  
influences(bob,carl).  
  
smokes(X) :- stress(X).  
smokes(X) :-  
    influences(Y,X), smokes(Y).
```



Logical Reasoning: Proofs in Prolog

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

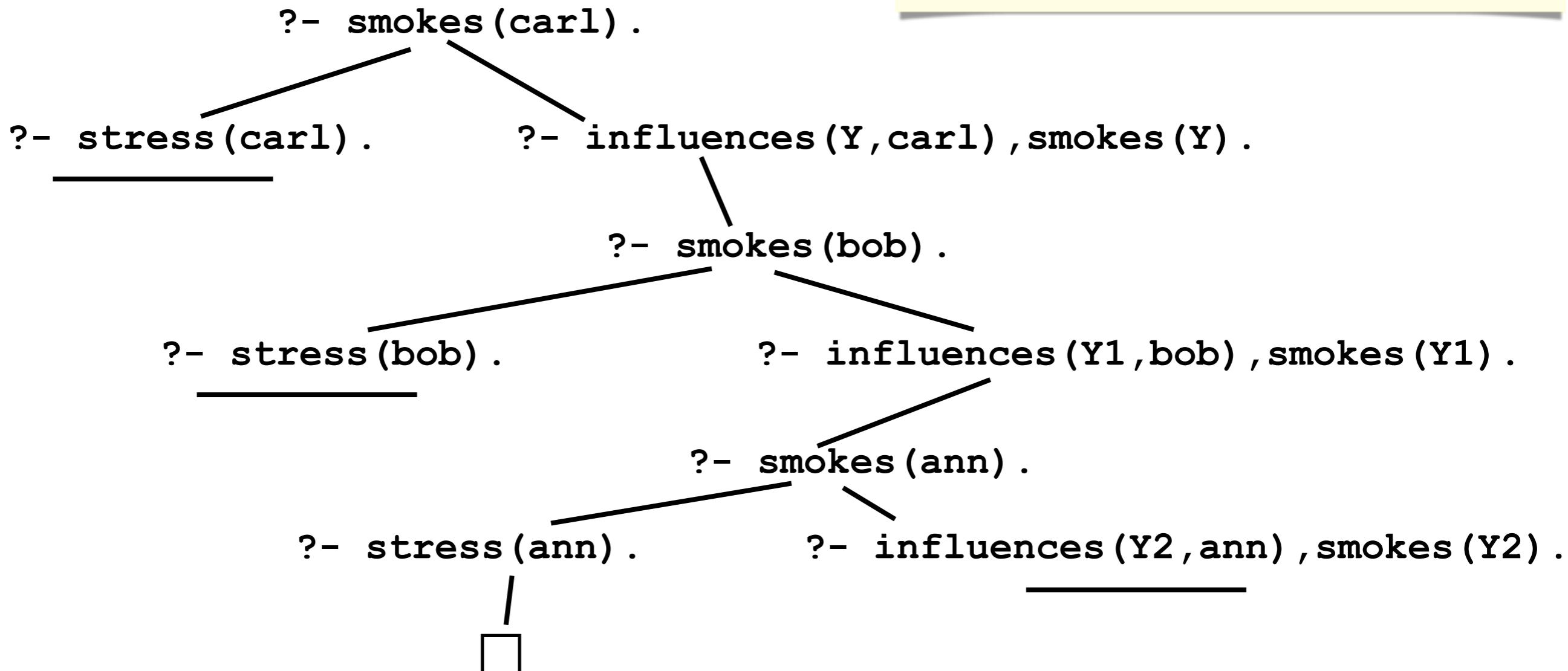
```
smokes(X) :- stress(X).  
smokes(X) :-  
    influences(Y,X), smokes(Y).
```



Logical Reasoning: Proofs in Prolog

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

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



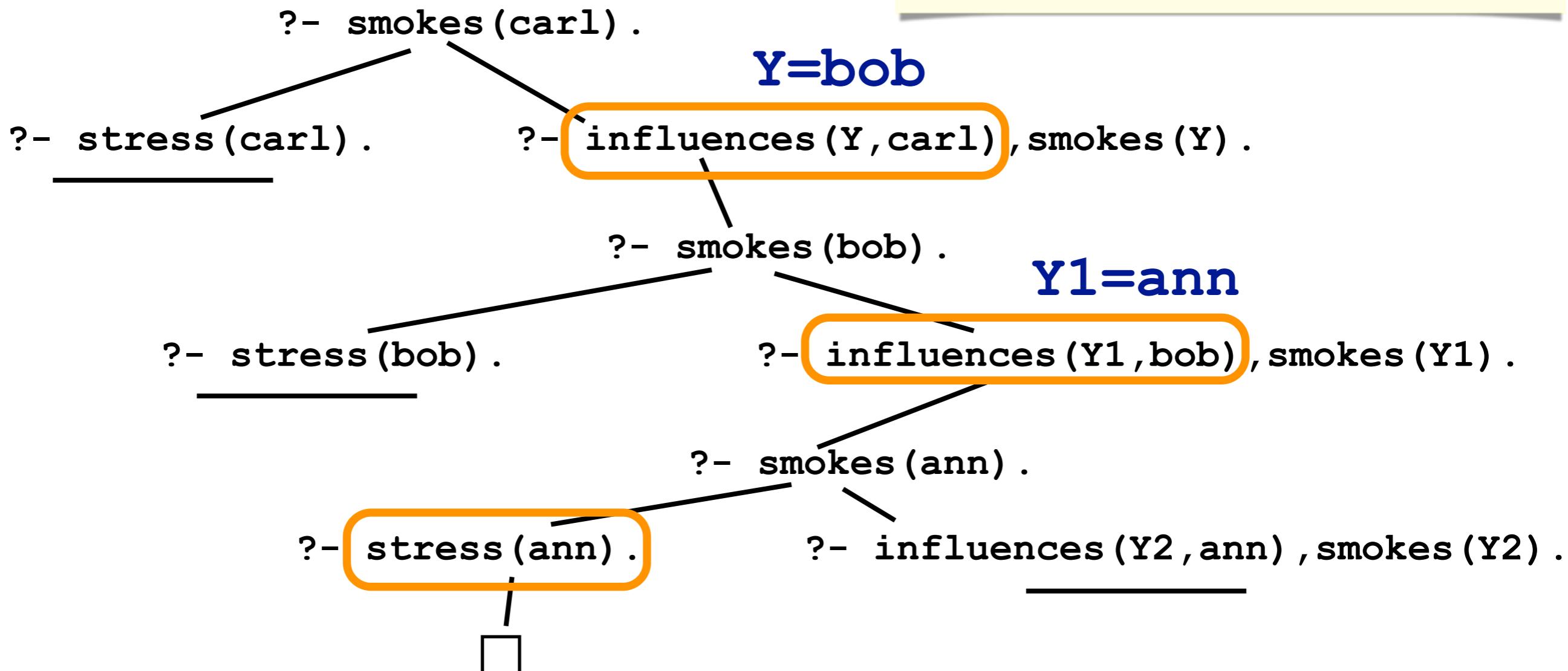
Logical Reasoning: Proofs in Prolog

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

```
smokes(X) :- stress(X).
```

smokes(X) :-

`influences(Y,X), smokes(Y).`



proof = facts used in successful derivation:
influences (bob, carl) & influences (ann, bob) & stress (ann)

Proofs in ProbLog

?- stress(bob) .

? - **stress (ann)**.

?- **smokes**(ann).

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

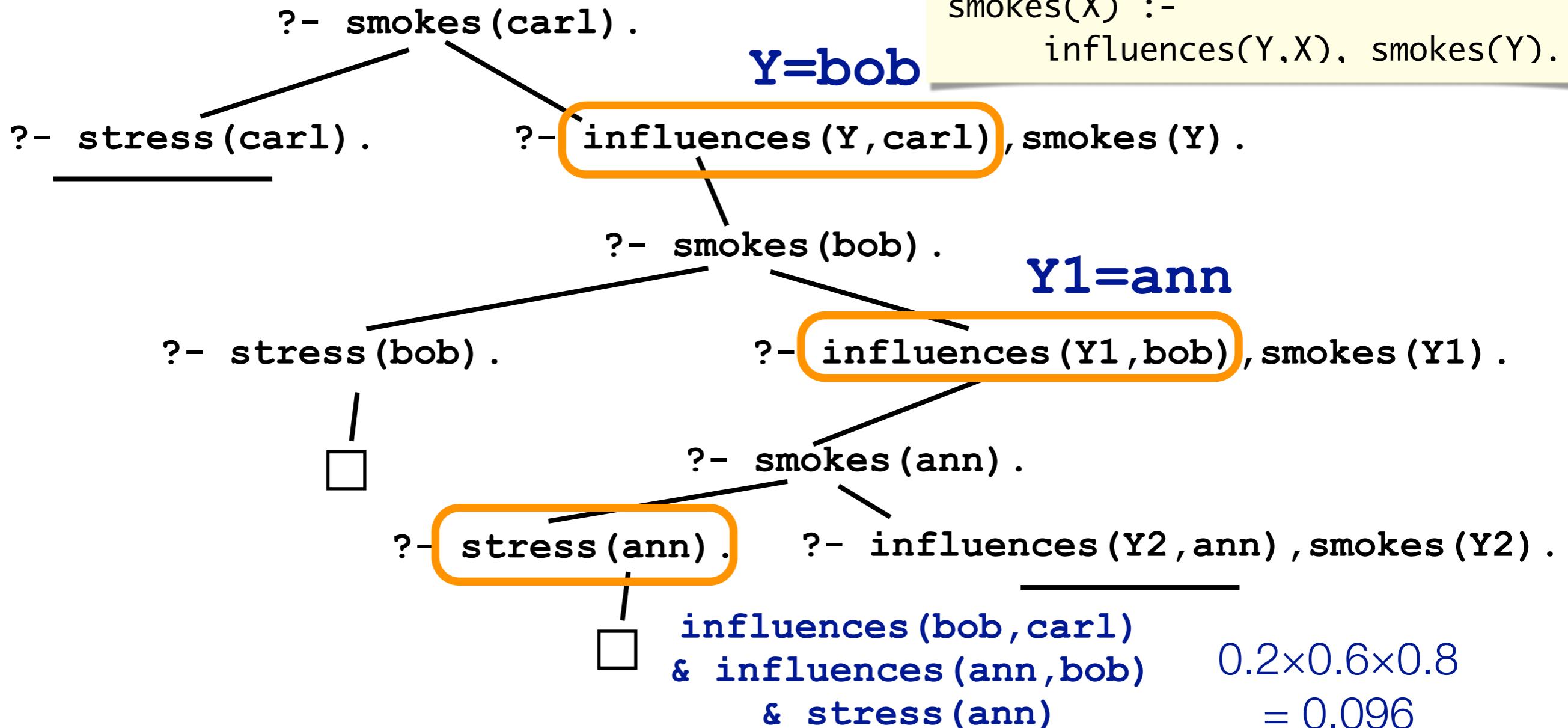
influences(bob, carl) & influences(ann, bob) & stress(ann)

probability of proof = $0.2 \times 0.6 \times 0.8 = 0.096$

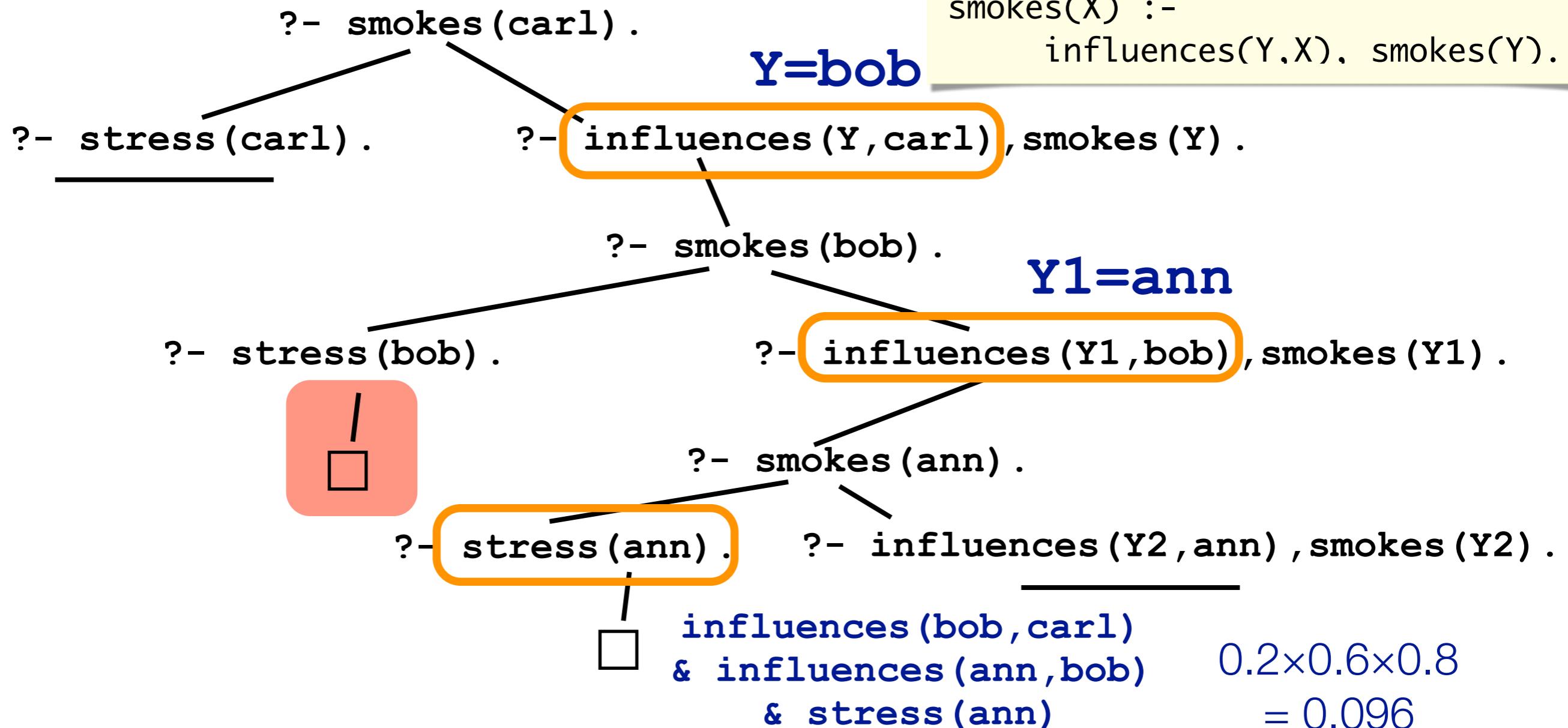
```
stress(ann).  
influences(ann,bob).  
influences(bob,carl).
```

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

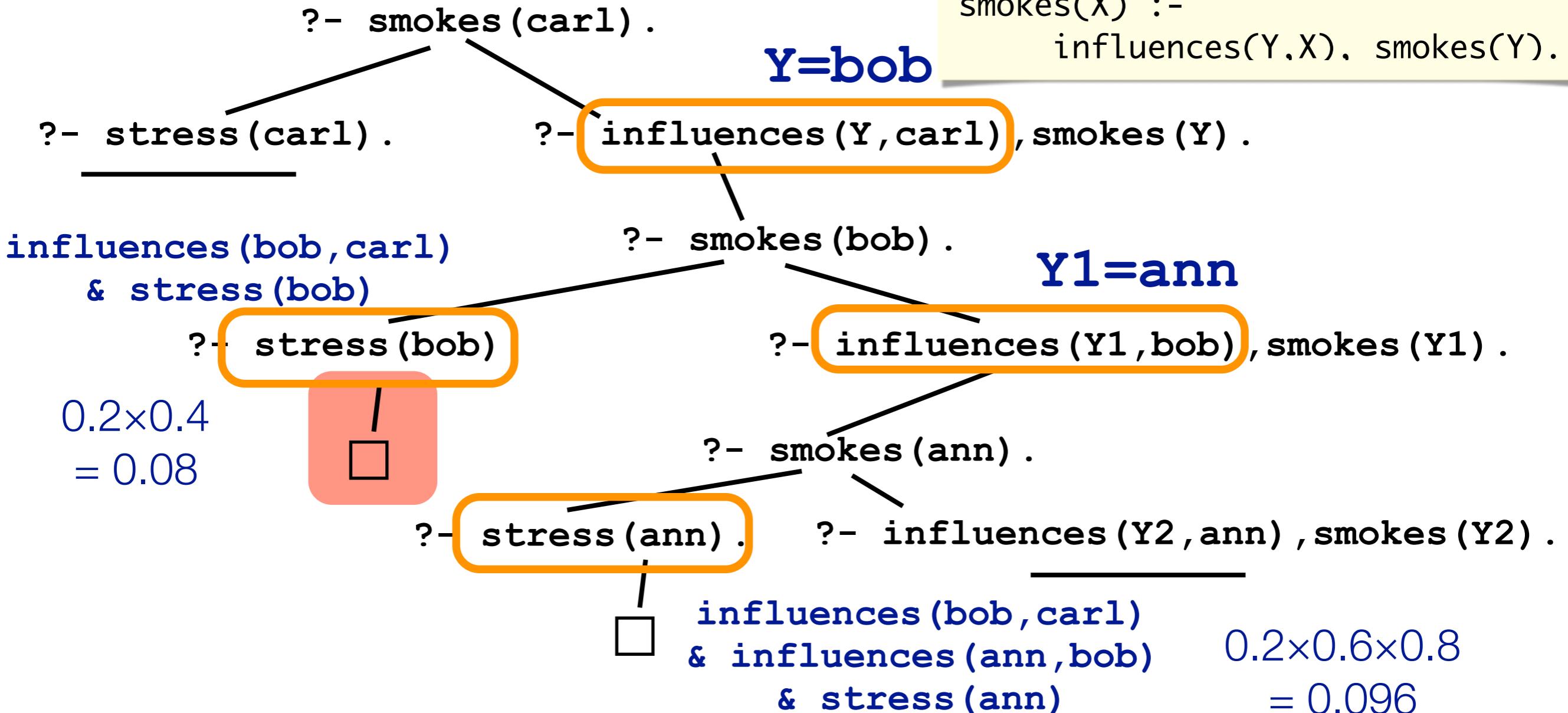
Proofs in ProbLog



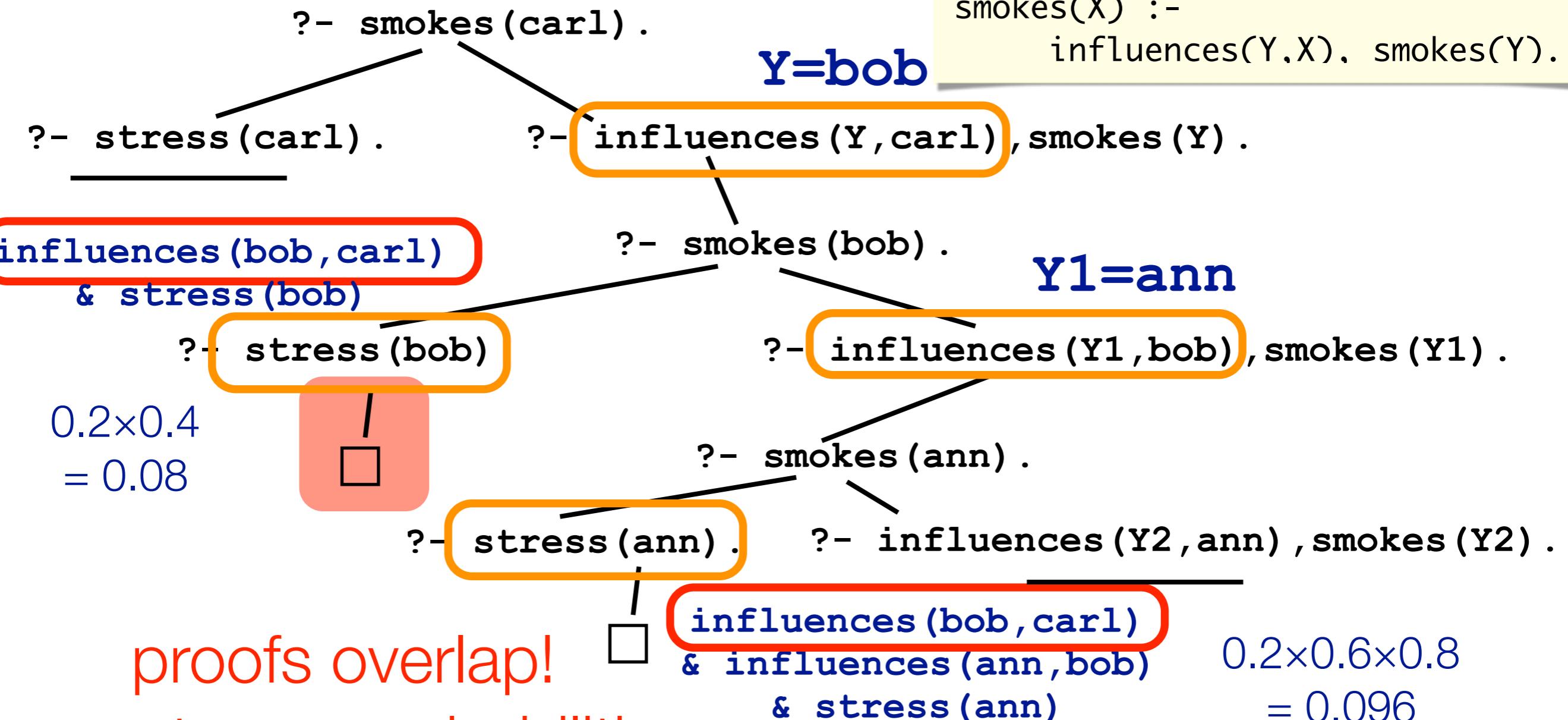
Proofs in ProbLog



Proofs in ProbLog



Proofs in ProbLog



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

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

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

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

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 =

48 0.1760

Disjoint-Sum-Problem

possible worlds

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

<code>infl(bob,carl) & infl(ann,bob) & st(ann) & \+st(bob)</code>	0.0576
<code>infl(bob,carl) & infl(ann,bob) & st(ann) & st(bob)</code>	0.0384
<code>infl(bob,carl) & \+infl(ann,bob) & st(ann) & st(bob)</code>	0.0256
<code>infl(bob,carl) & infl(ann,bob) & \+st(ann) & st(bob)</code>	0.0096
<code>infl(bob,carl) & \+infl(ann,bob) & \+st(ann) & st(bob)</code>	0.0064
<code>...</code>	
<code>influences(bob,carl) & stress(bob)</code>	$\sum = 0.1376$

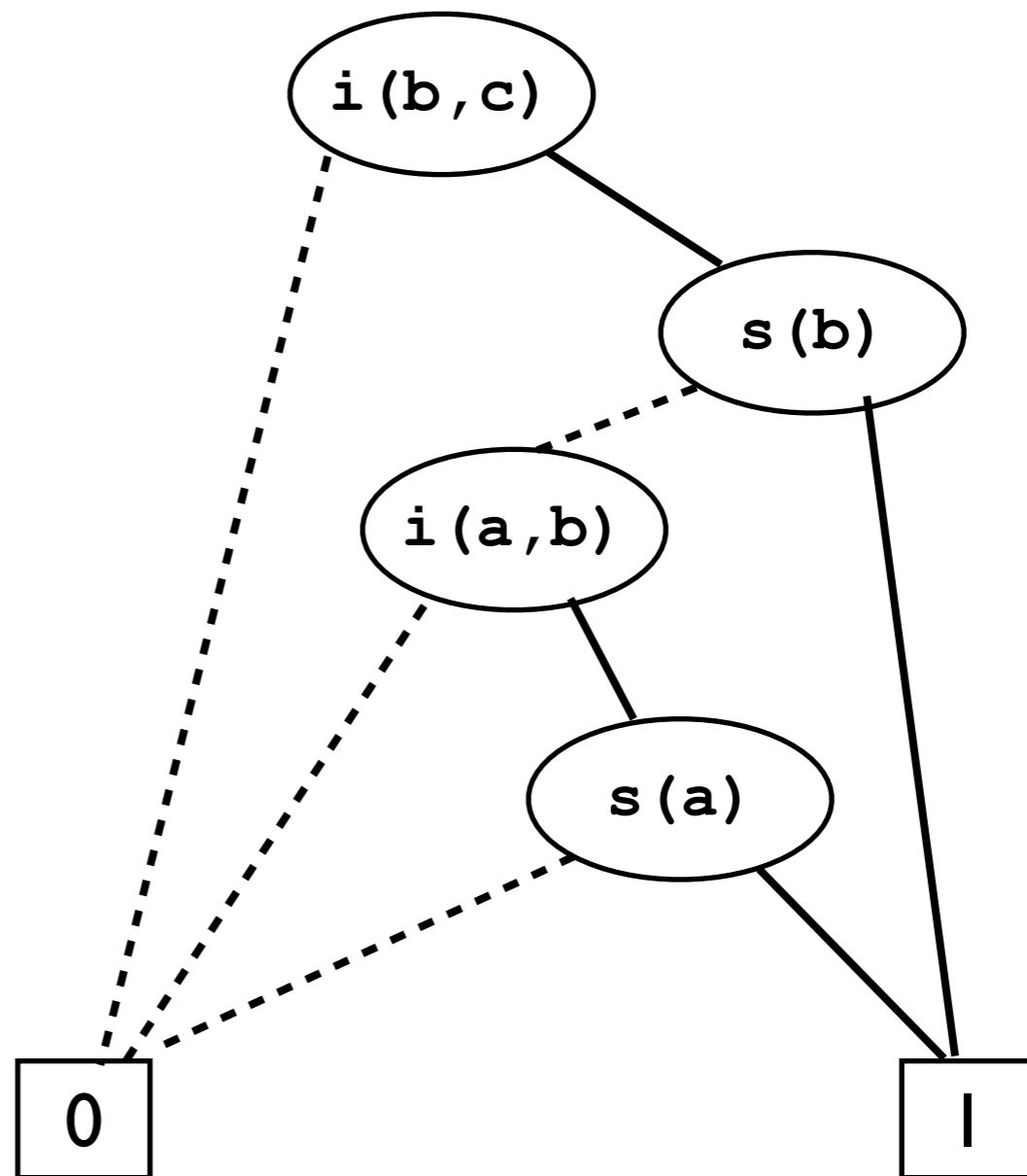
sum of proof probabilities: $0.096 + 0.08 =$

0.1760

Binary Decision Diagrams

[Bryant 86]

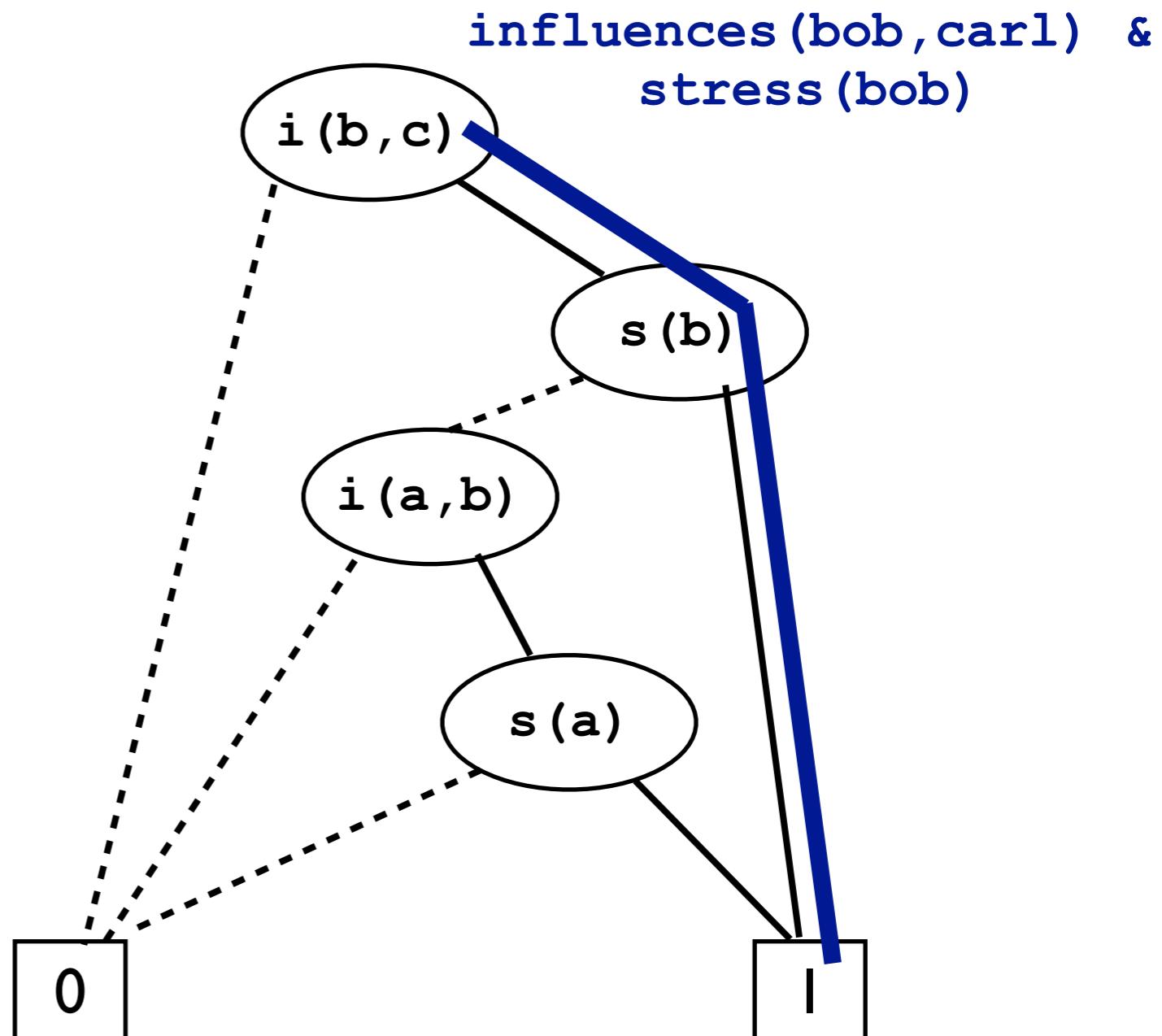
- compact graphical representation of Boolean formula
- automatically disjoins proofs
- popular in many branches of CS



Binary Decision Diagrams

[Bryant 86]

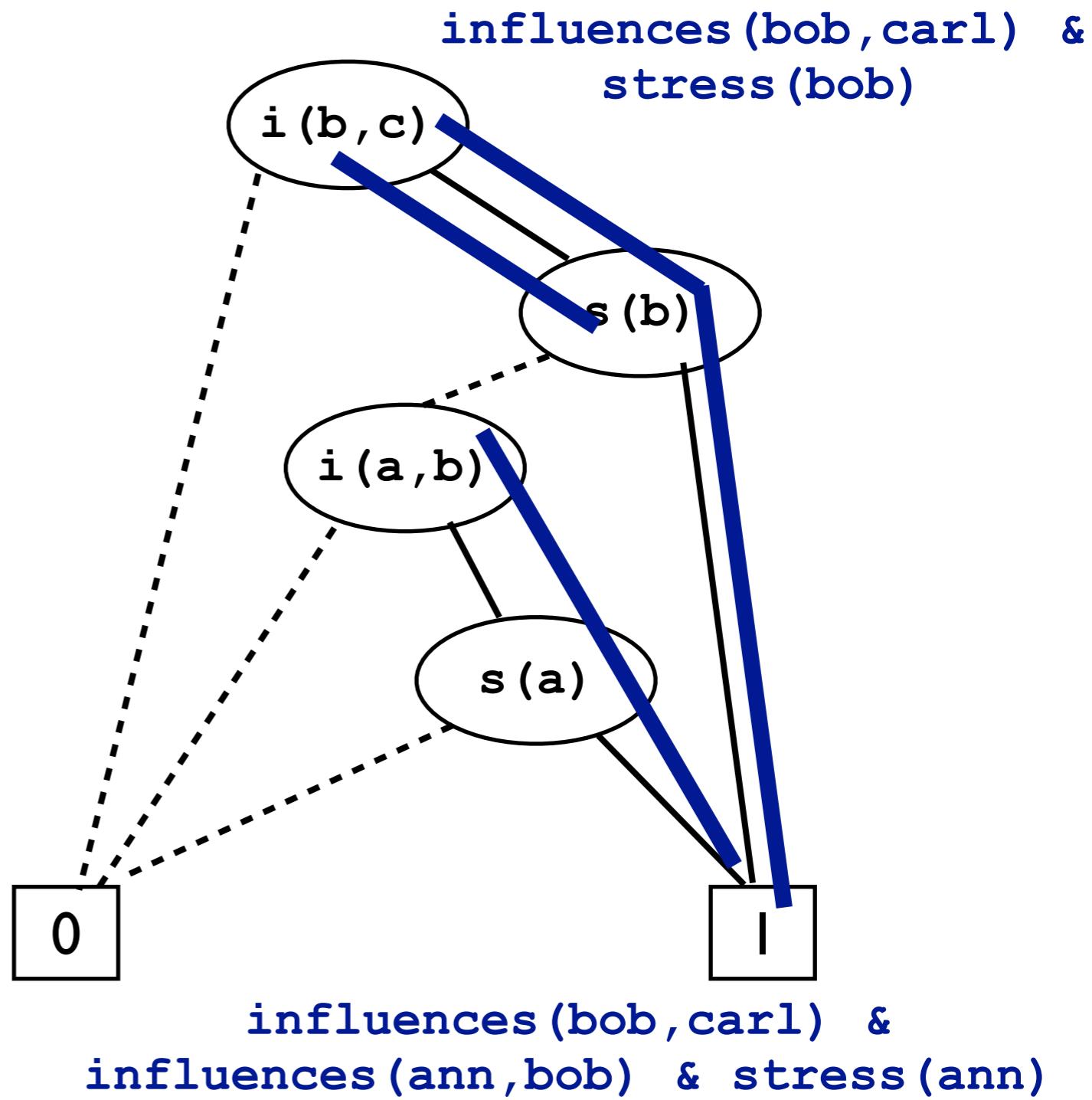
- compact graphical representation of Boolean formula
- automatically disjoins proofs
- popular in many branches of CS



Binary Decision Diagrams

[Bryant 86]

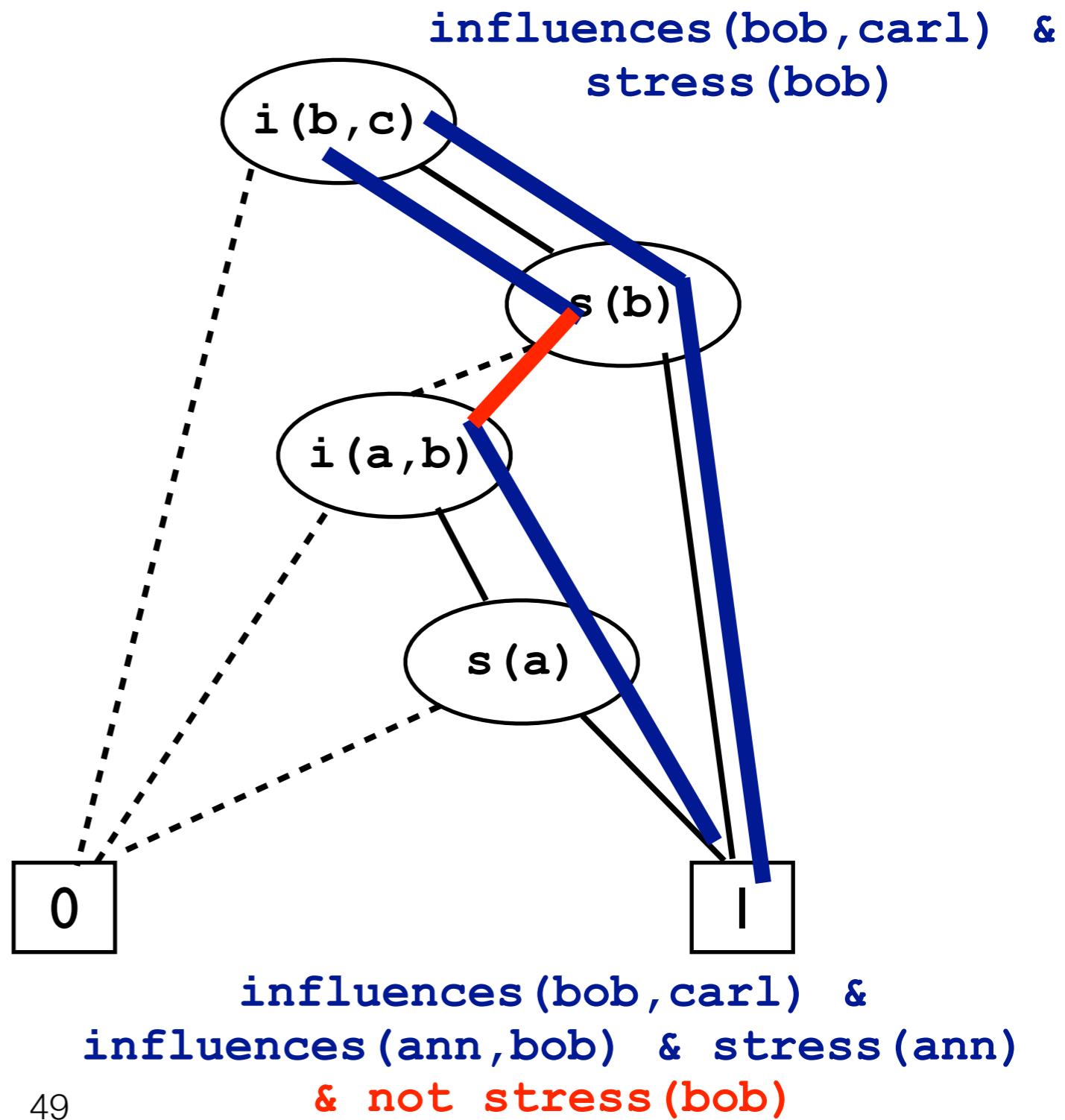
- compact graphical representation of Boolean formula
- automatically disjoins proofs
- popular in many branches of CS



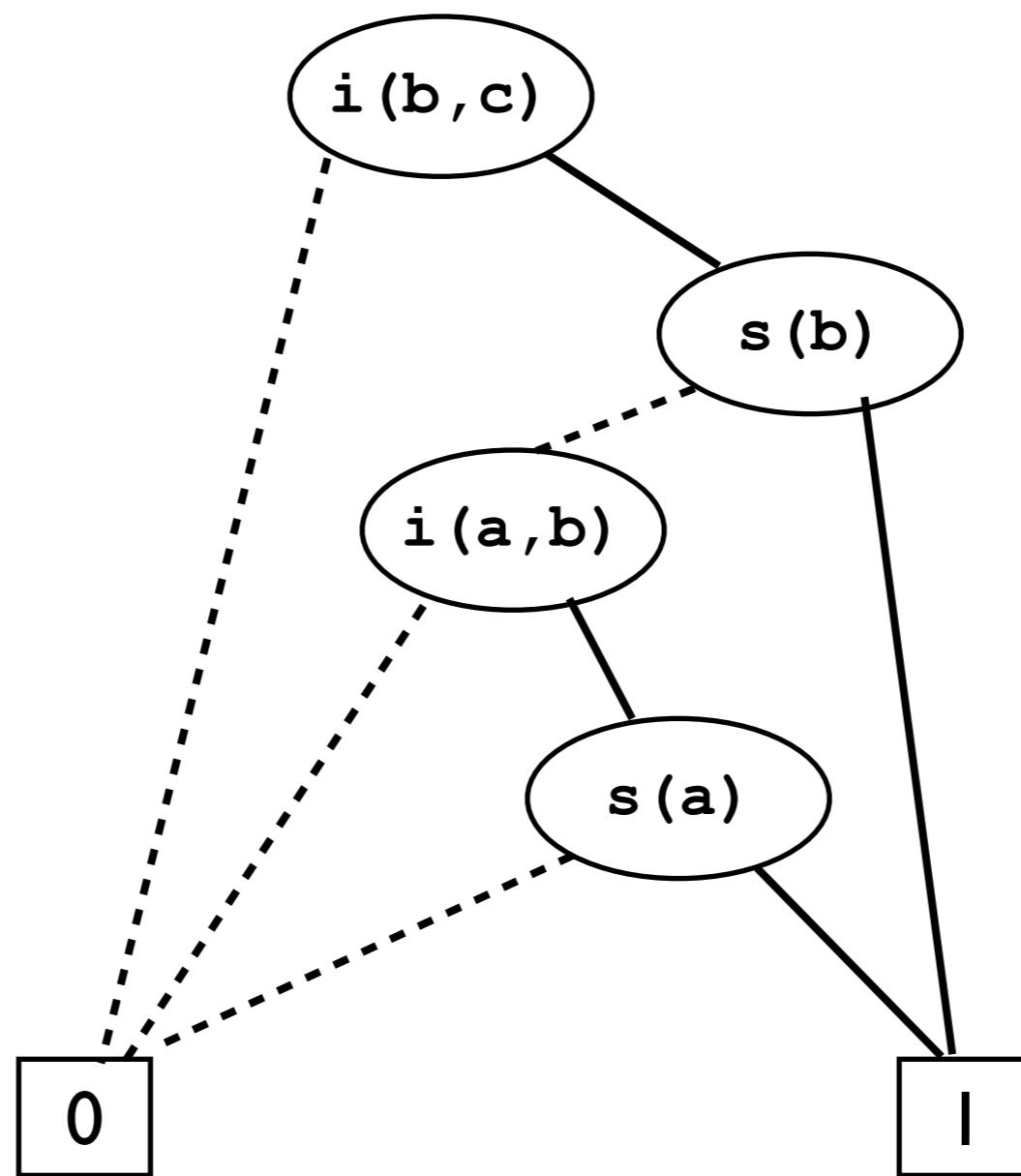
Binary Decision Diagrams

[Bryant 86]

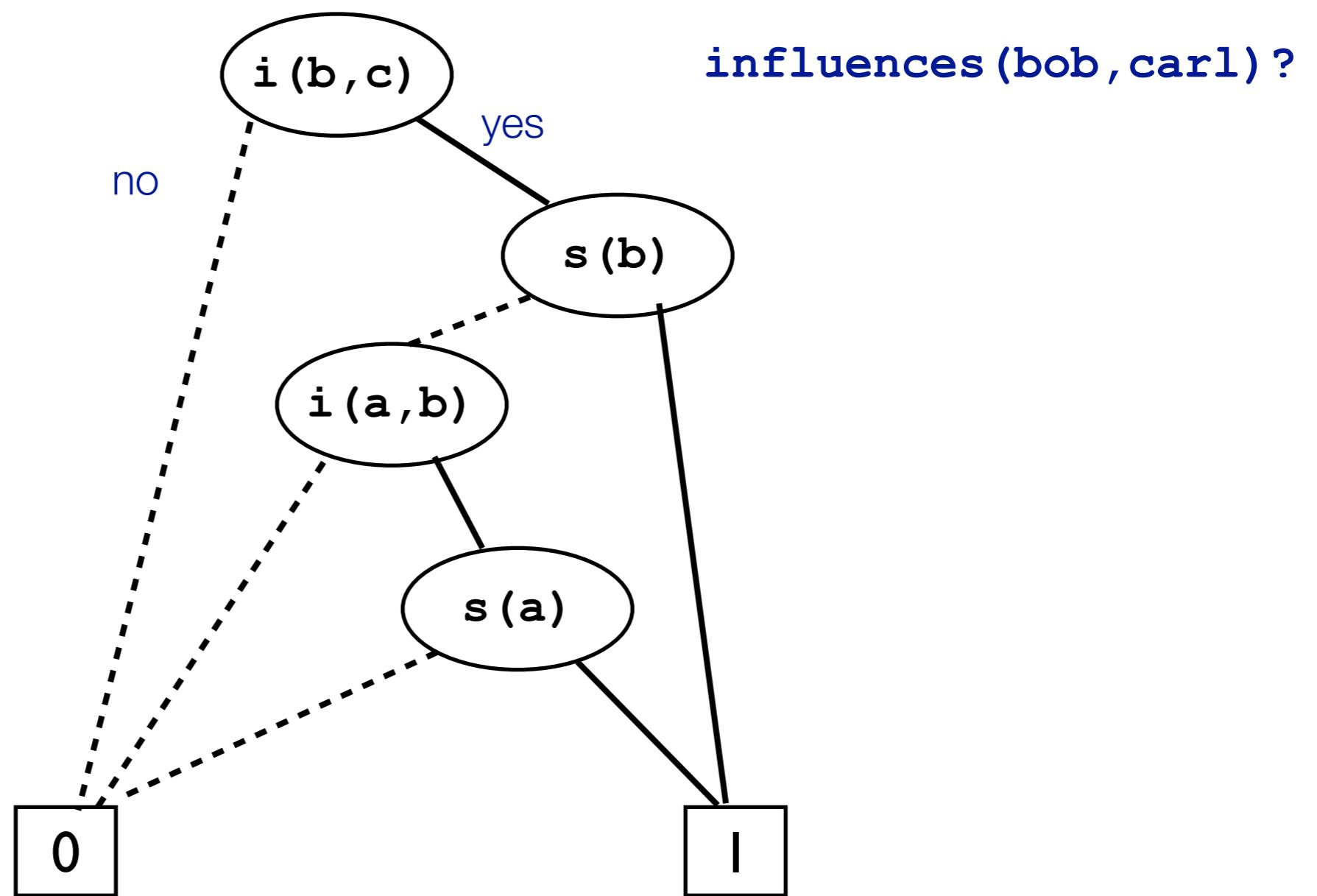
- compact graphical representation of Boolean formula
- automatically disjoins proofs
- popular in many branches of CS



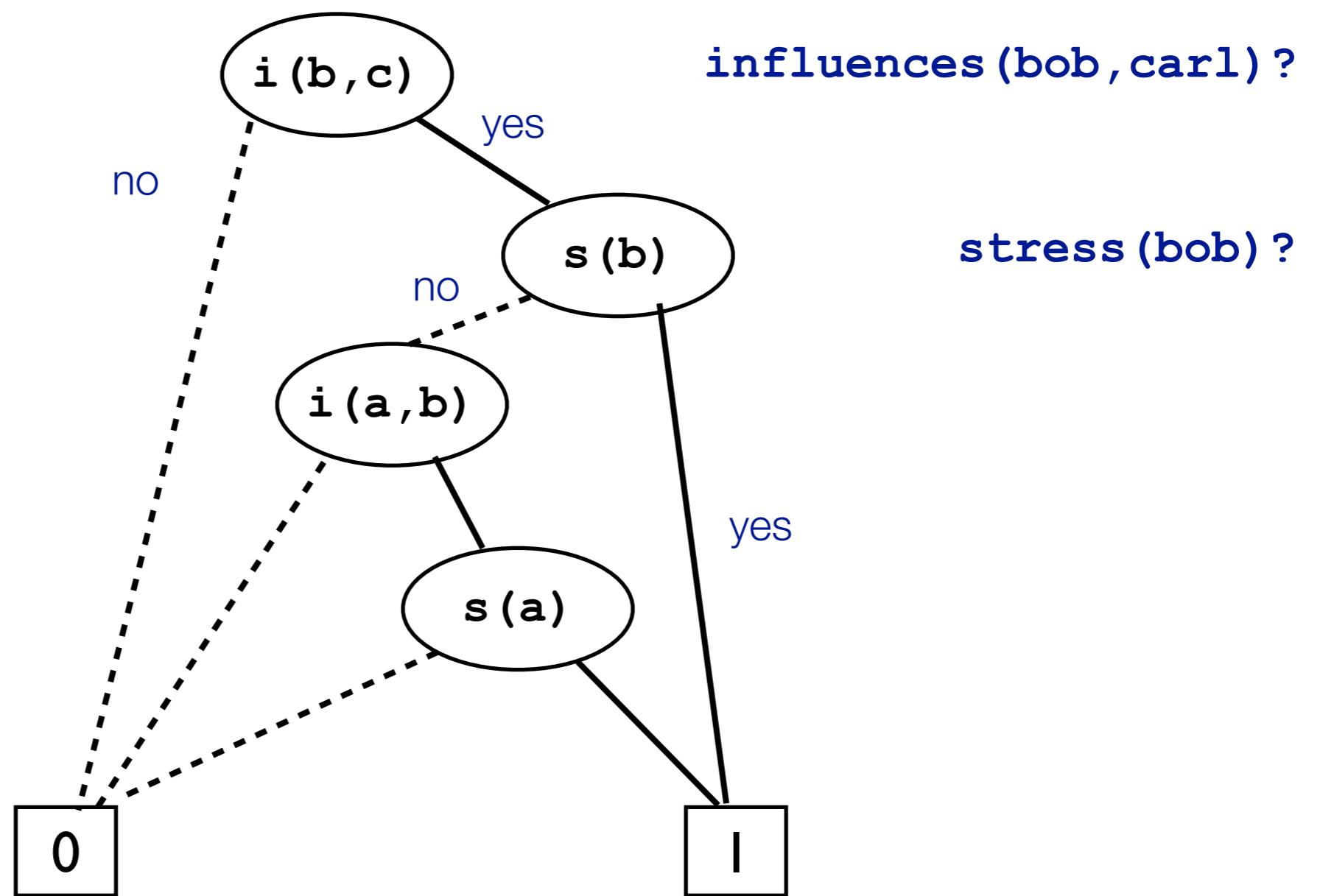
Binary Decision Diagrams



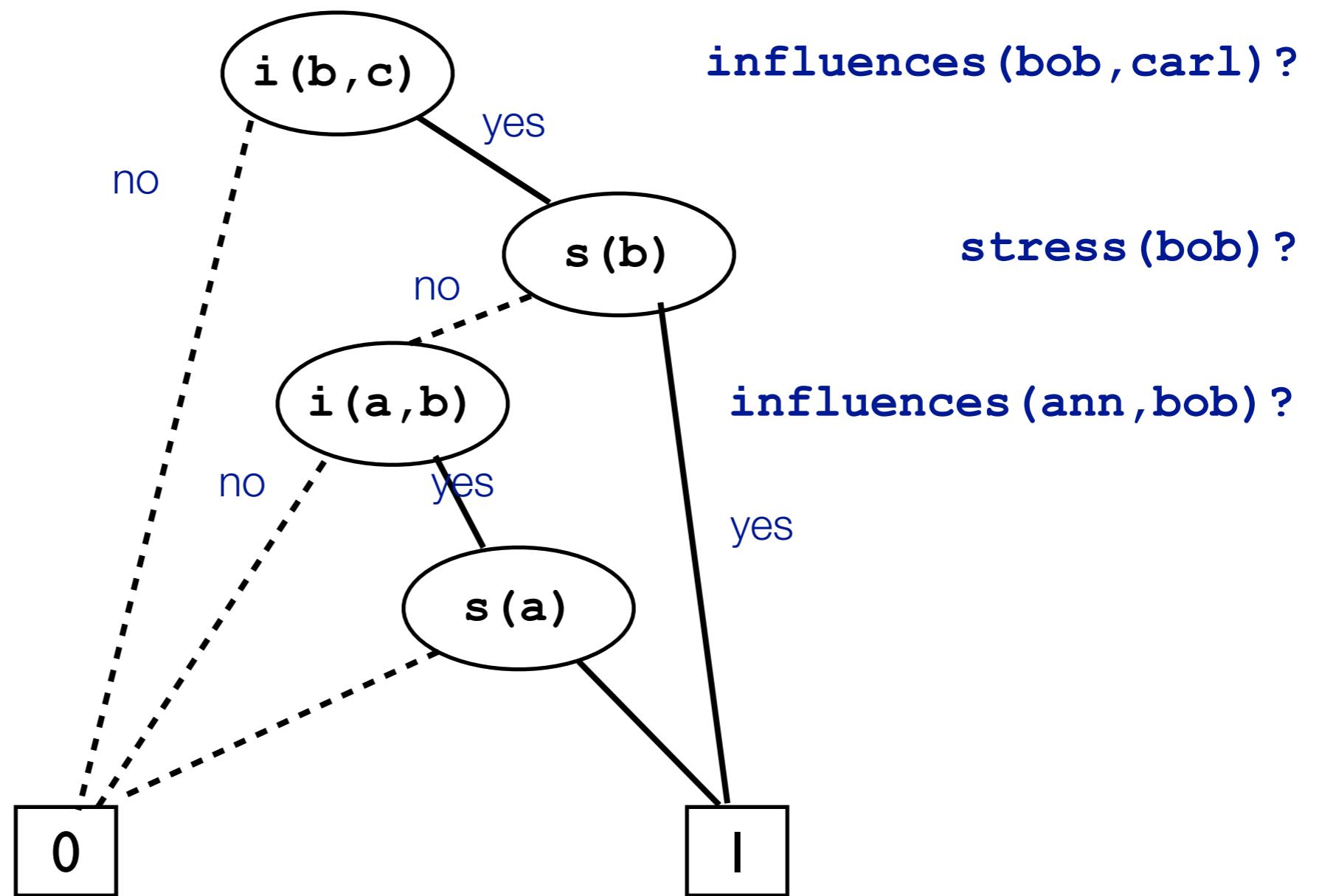
Binary Decision Diagrams



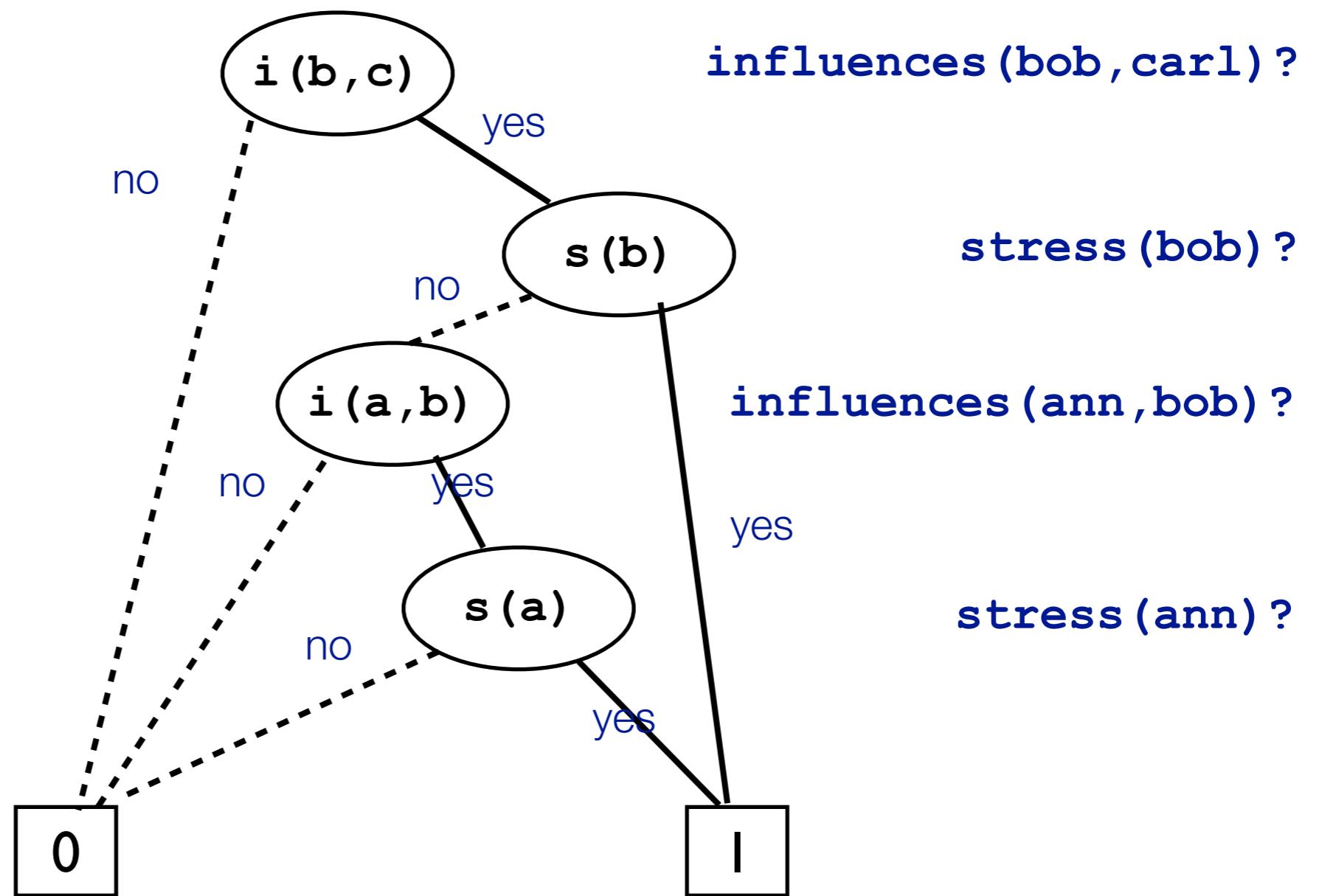
Binary Decision Diagrams



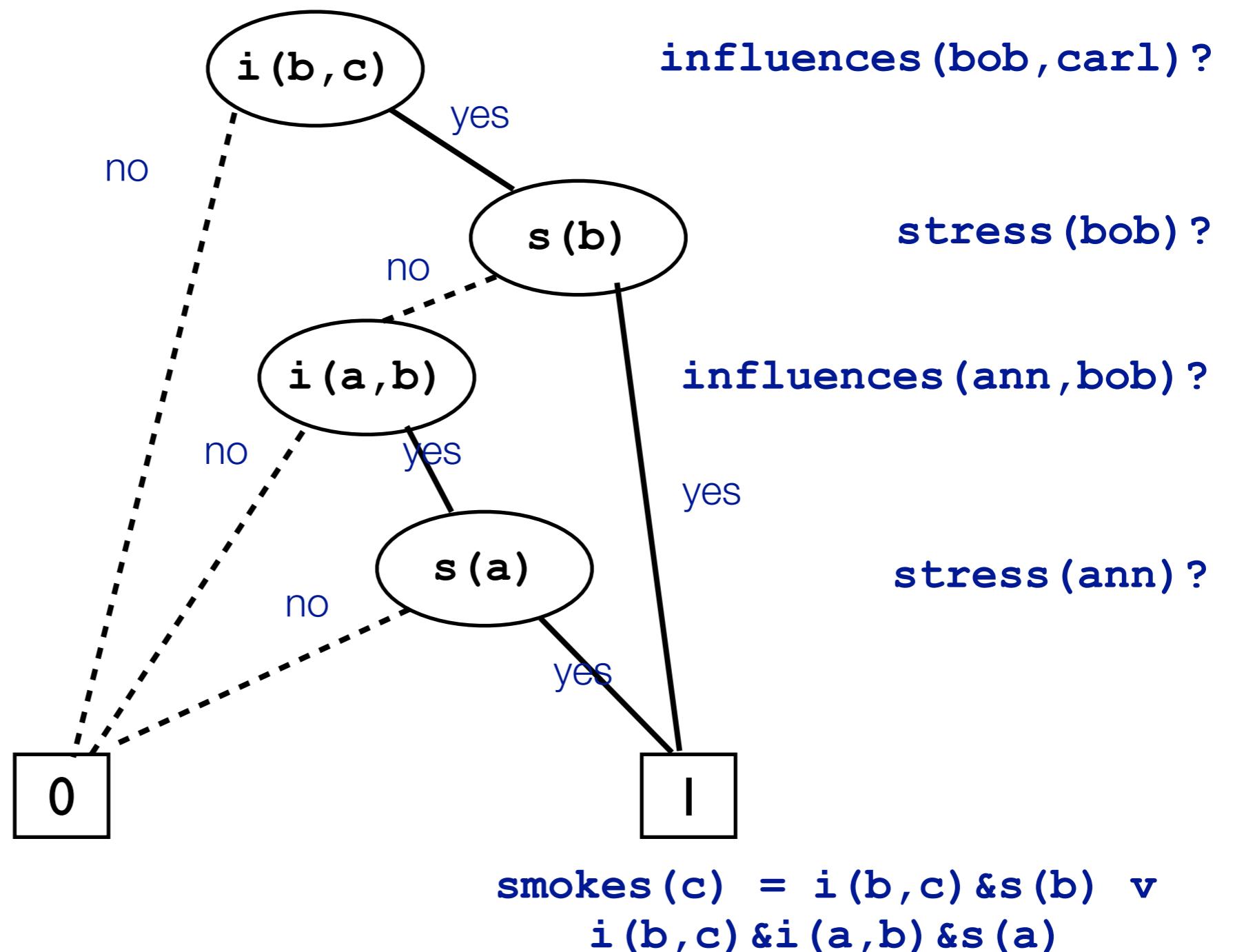
Binary Decision Diagrams



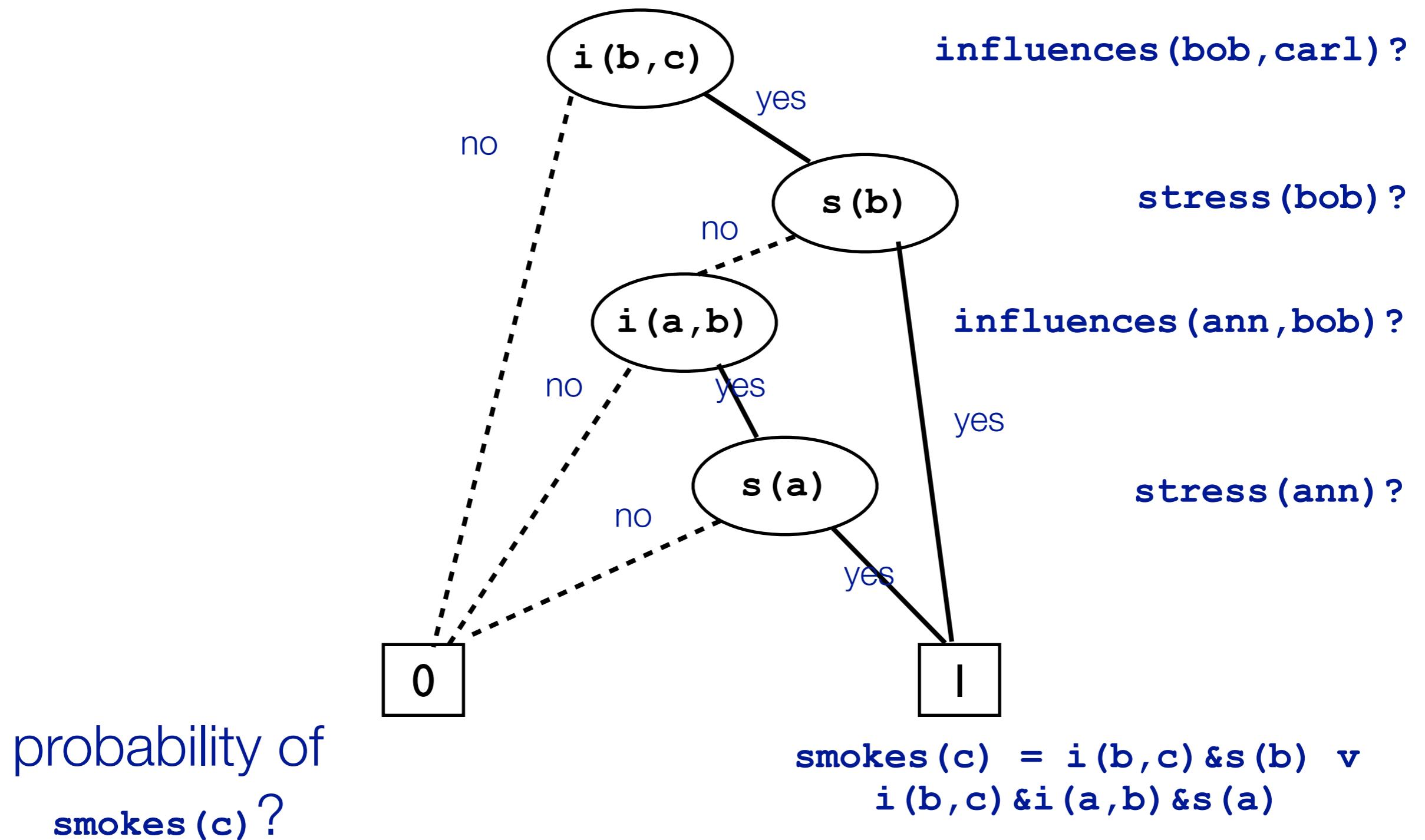
Binary Decision Diagrams



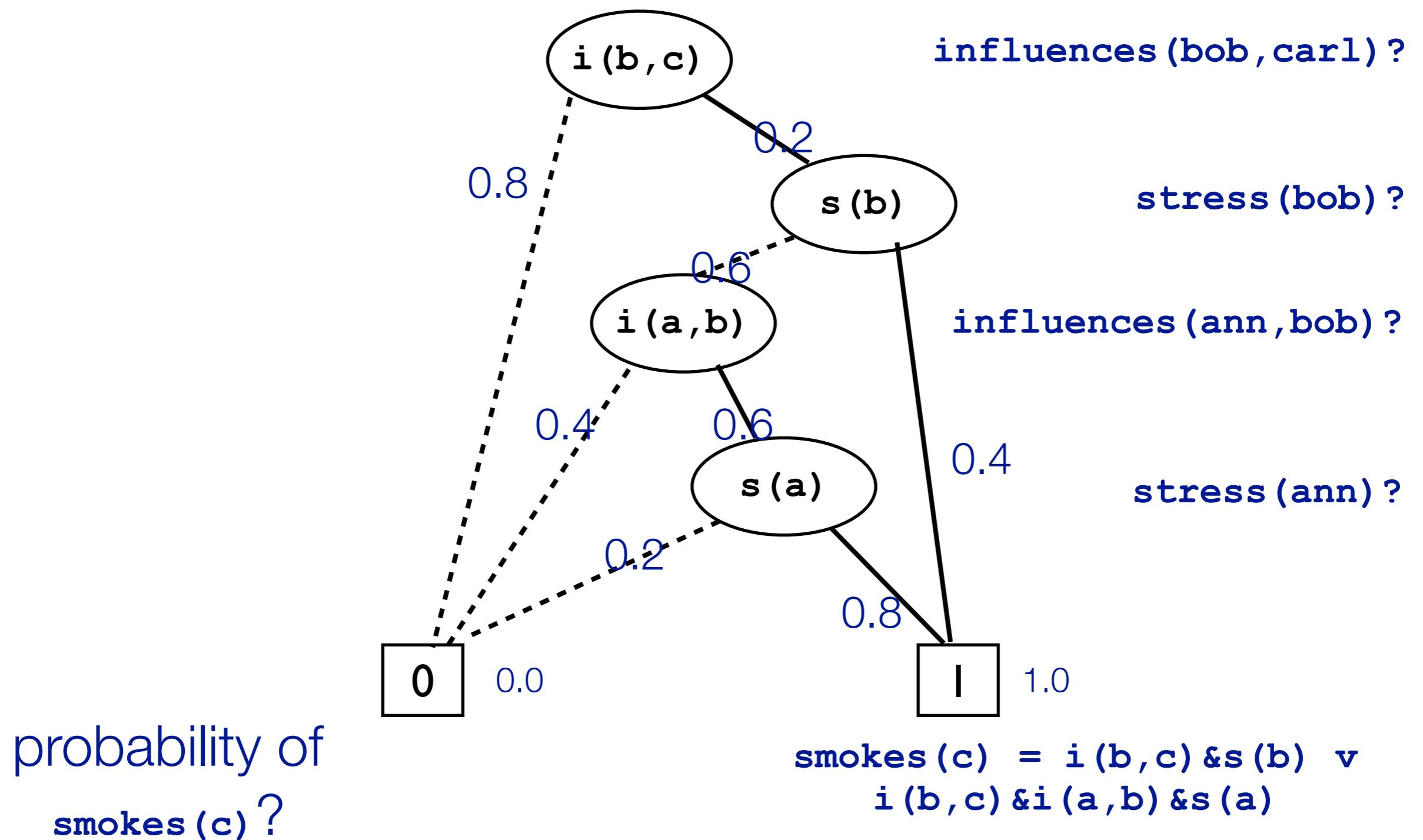
Binary Decision Diagrams



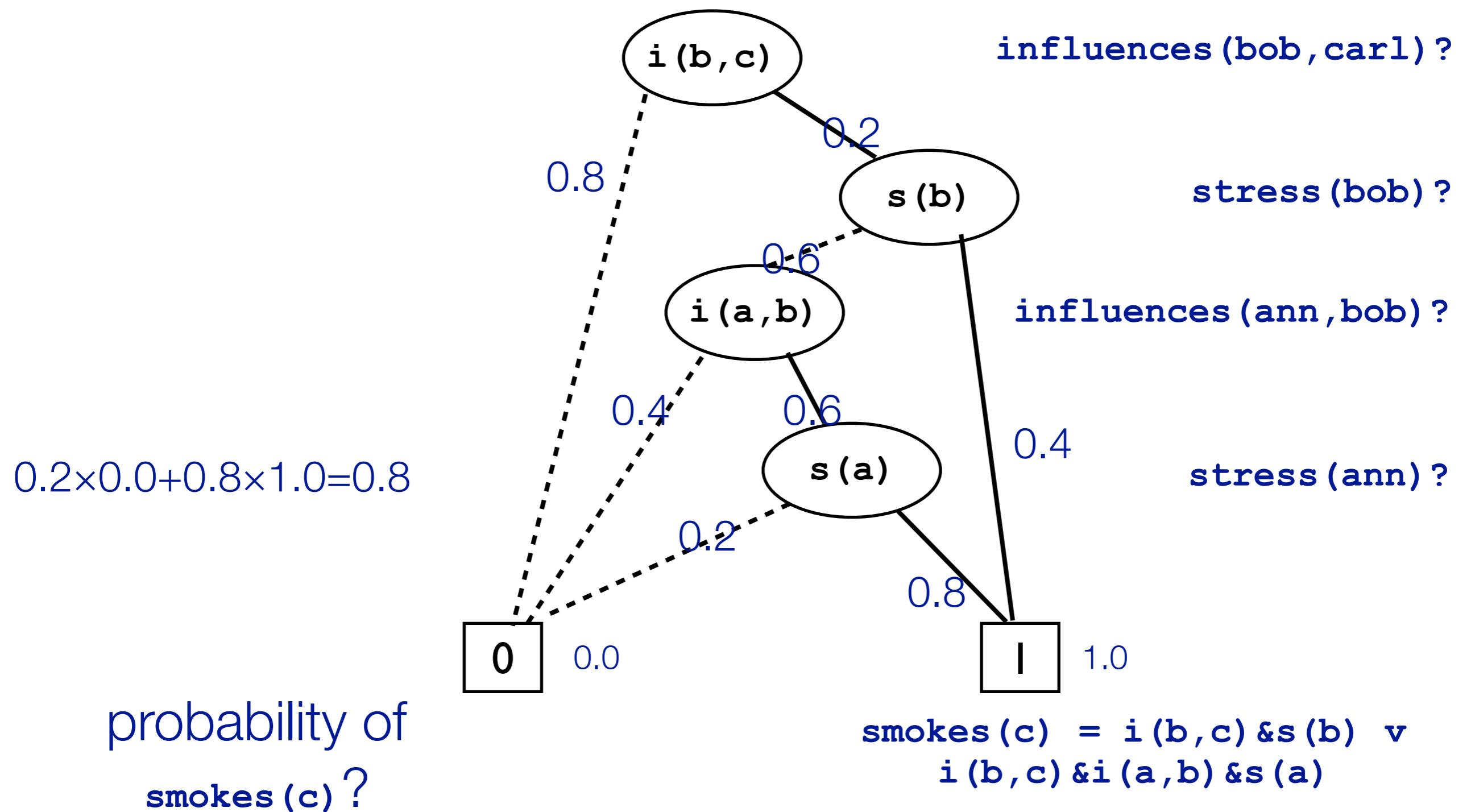
Binary Decision Diagrams



Binary Decision Diagrams



Binary Decision Diagrams

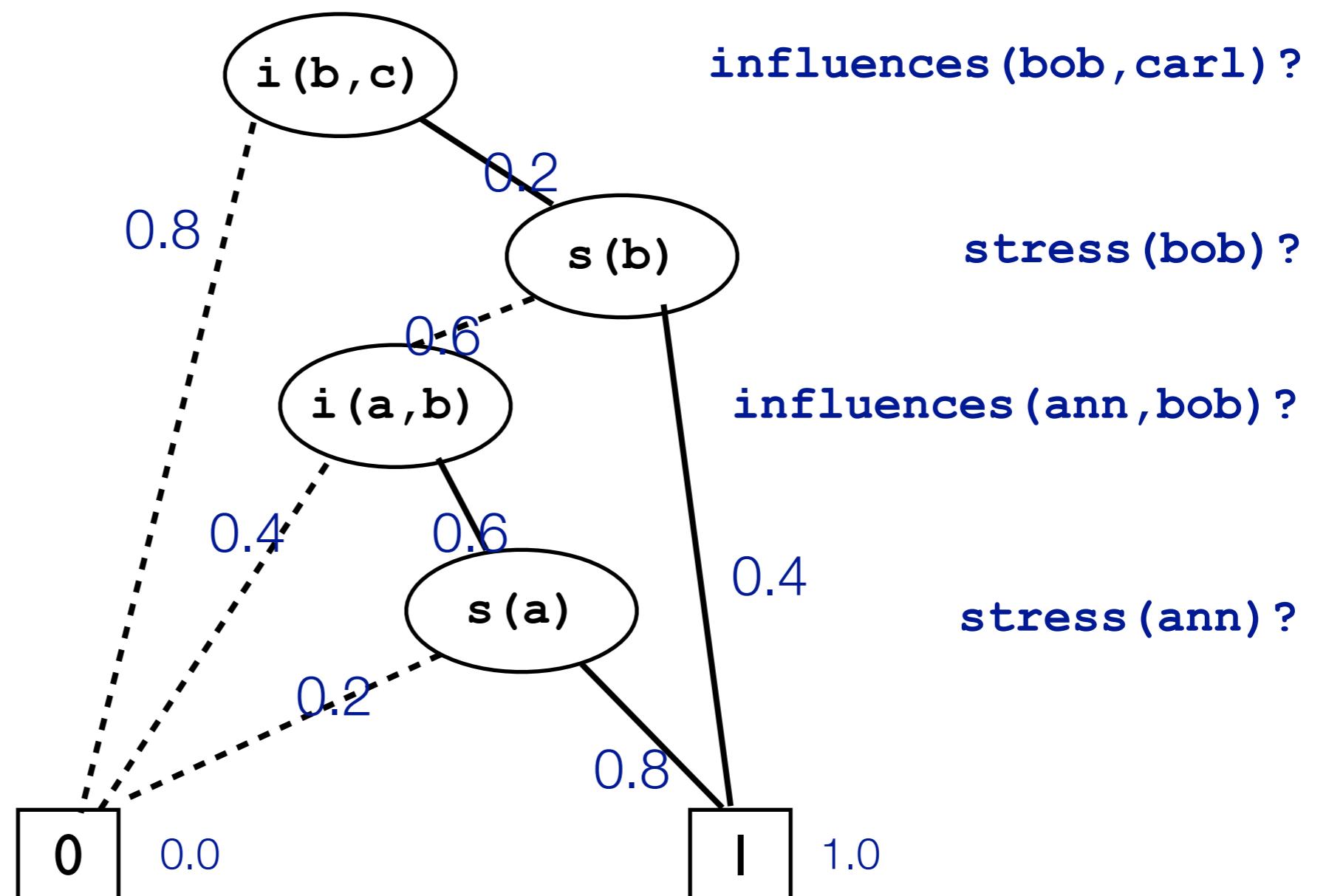


Binary Decision Diagrams

$$0.4 \times 0.0 + 0.6 \times 0.8 = 0.48$$

$$0.2 \times 0.0 + 0.8 \times 1.0 = 0.8$$

probability of
smokes (c) ?



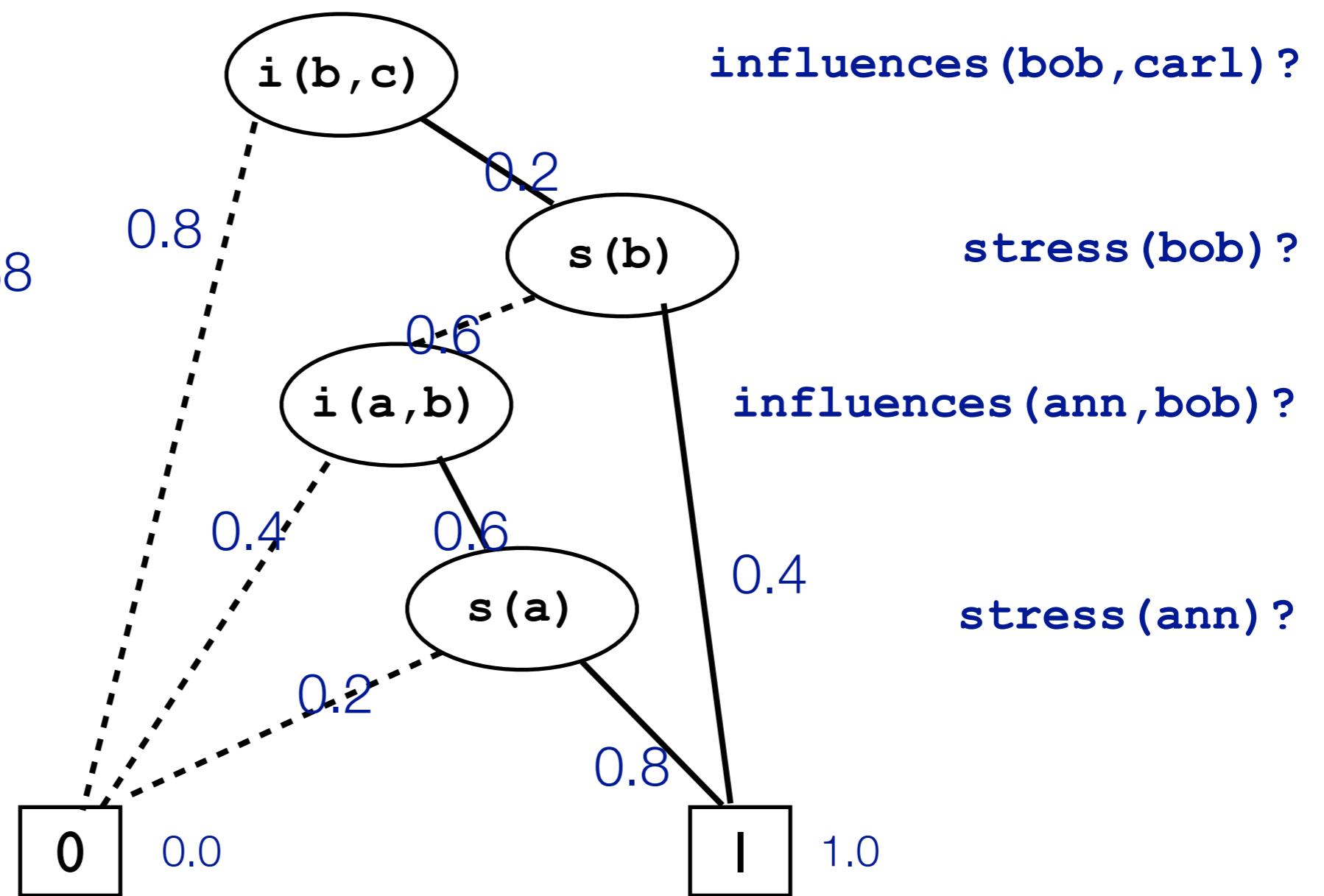
Binary Decision Diagrams

$$0.6 \times 0.48 + 0.4 \times 1.0 = 0.688$$

$$0.4 \times 0.0 + 0.6 \times 0.8 = 0.48$$

$$0.2 \times 0.0 + 0.8 \times 1.0 = 0.8$$

probability of
smokes (c) ?



Binary Decision Diagrams

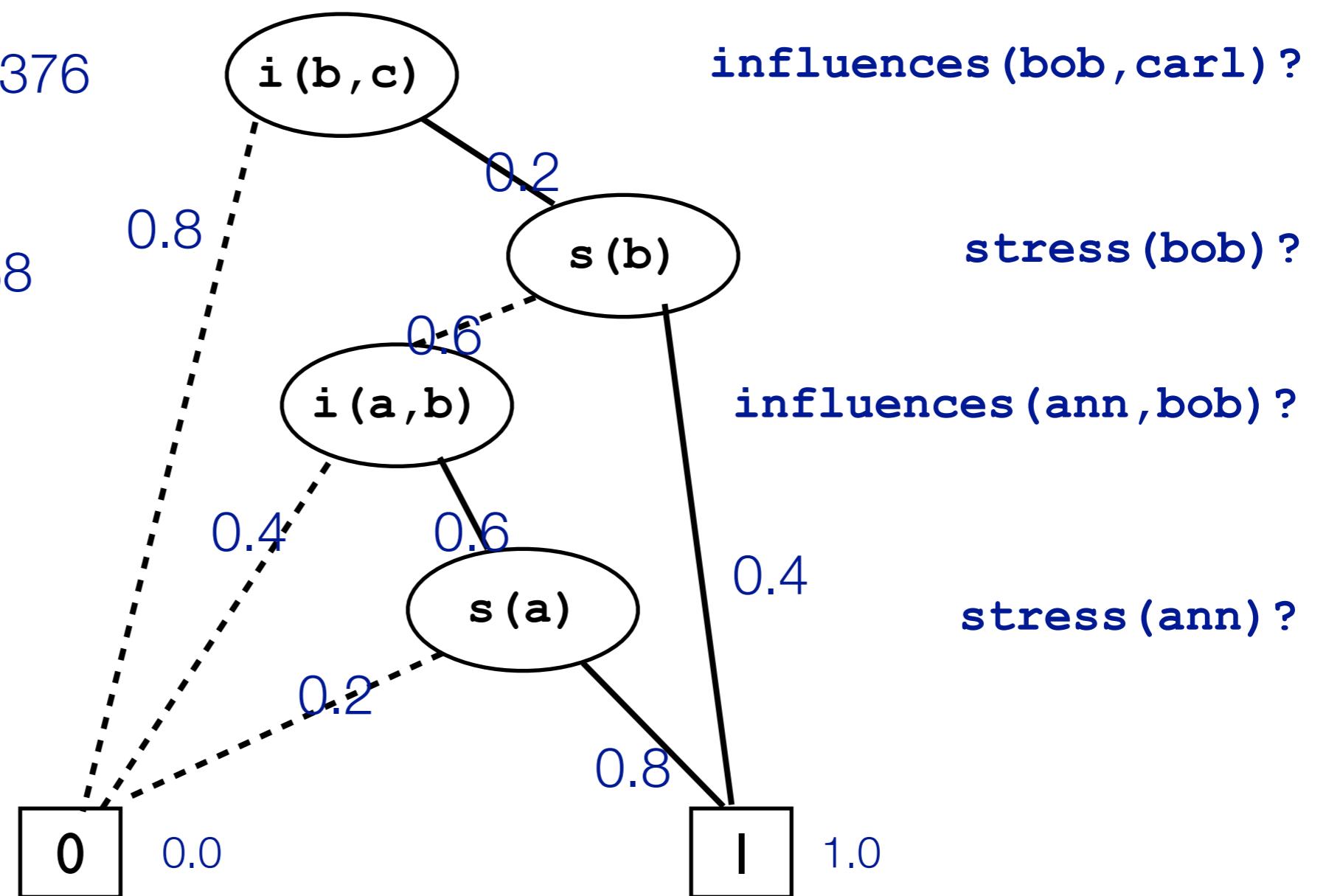
$$0.8 \times 0.0 + 0.2 \times 0.688 = 0.1376$$

$$0.6 \times 0.48 + 0.4 \times 1.0 = 0.688$$

$$0.4 \times 0.0 + 0.6 \times 0.8 = 0.48$$

$$0.2 \times 0.0 + 0.8 \times 1.0 = 0.8$$

probability of
smokes (c) ?



influences (bob , carl) ?

stress (bob) ?

influences (ann , bob) ?

stress (ann) ?

smokes (c) = $i(b,c) \& s(b) \vee i(b,c) \& i(a,b) \& s(a)$

Initial Approach

(ProbLog1)

Find all proofs of query

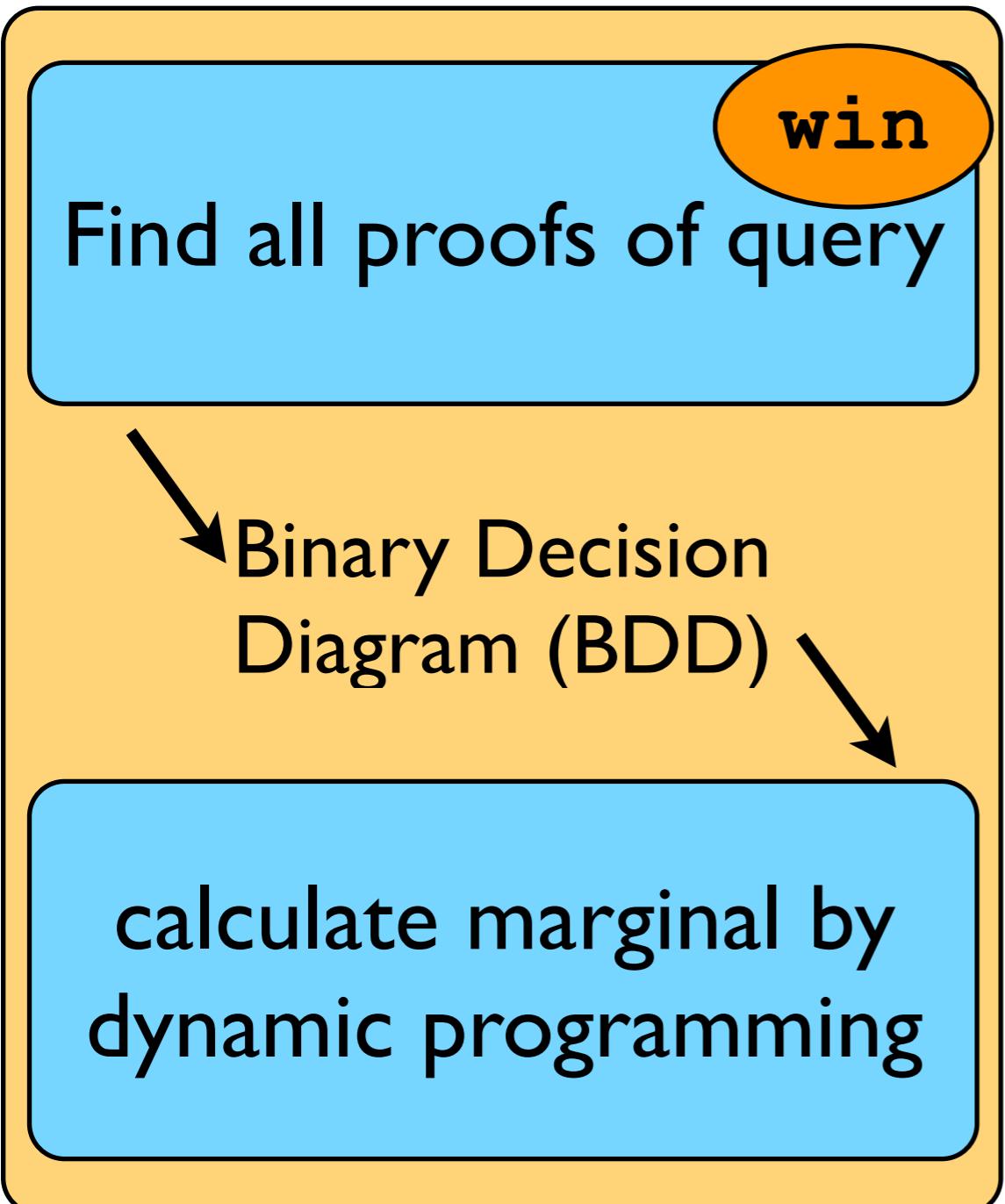
Binary Decision
Diagram (BDD)

calculate marginal by
dynamic programming

Initial Approach

(ProbLog1)

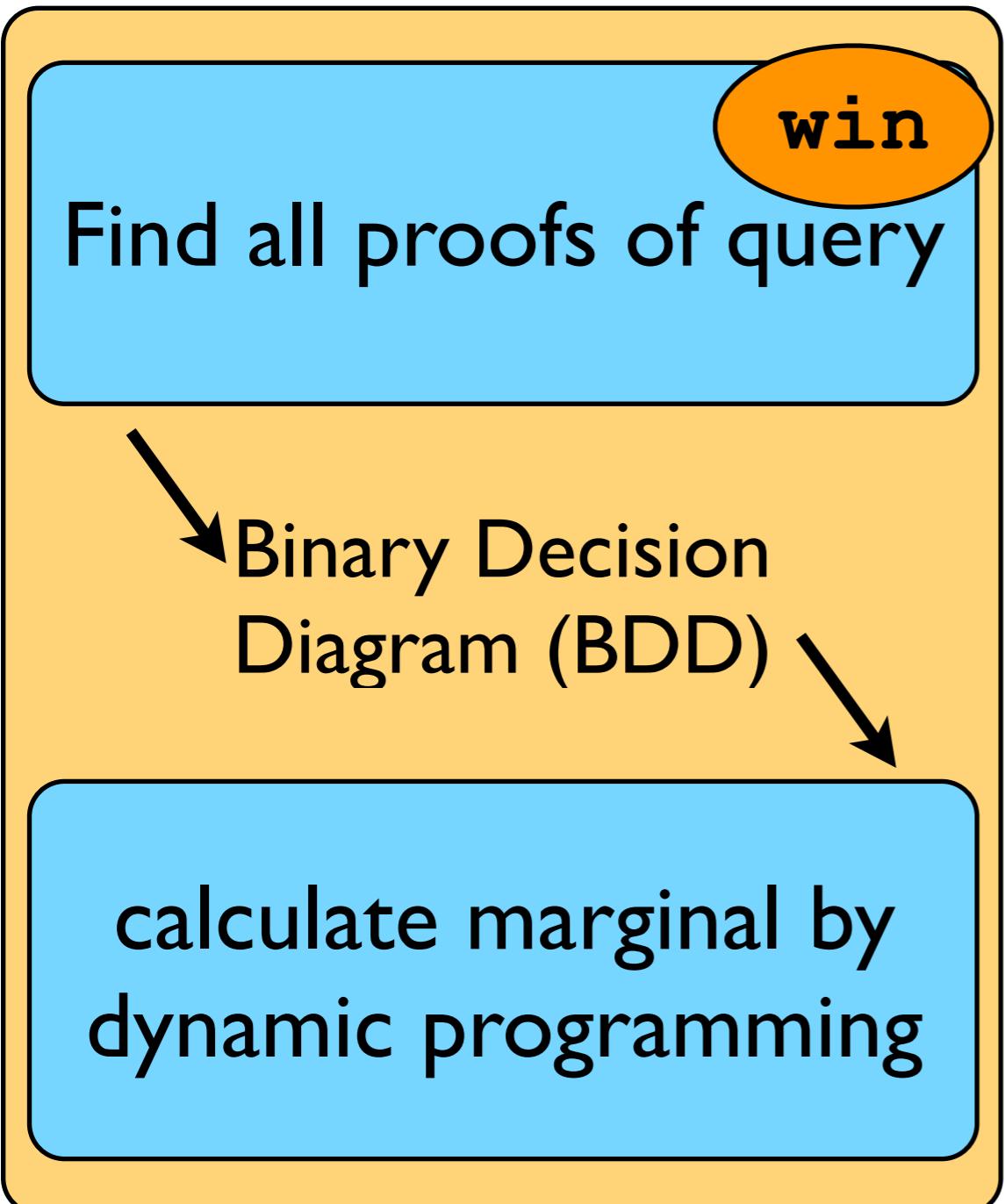
```
0.4 :: heads(1).  
0.7 :: heads(2).  
0.5 :: heads(3).  
win :- heads(1).  
win :- heads(2),heads(3).
```



Initial Approach

(ProbLog1)

```
0.4 :: heads(1).  
0.7 :: heads(2).  
0.5 :: heads(3).  
win :- heads(1).  
win :- heads(2), heads(3).
```

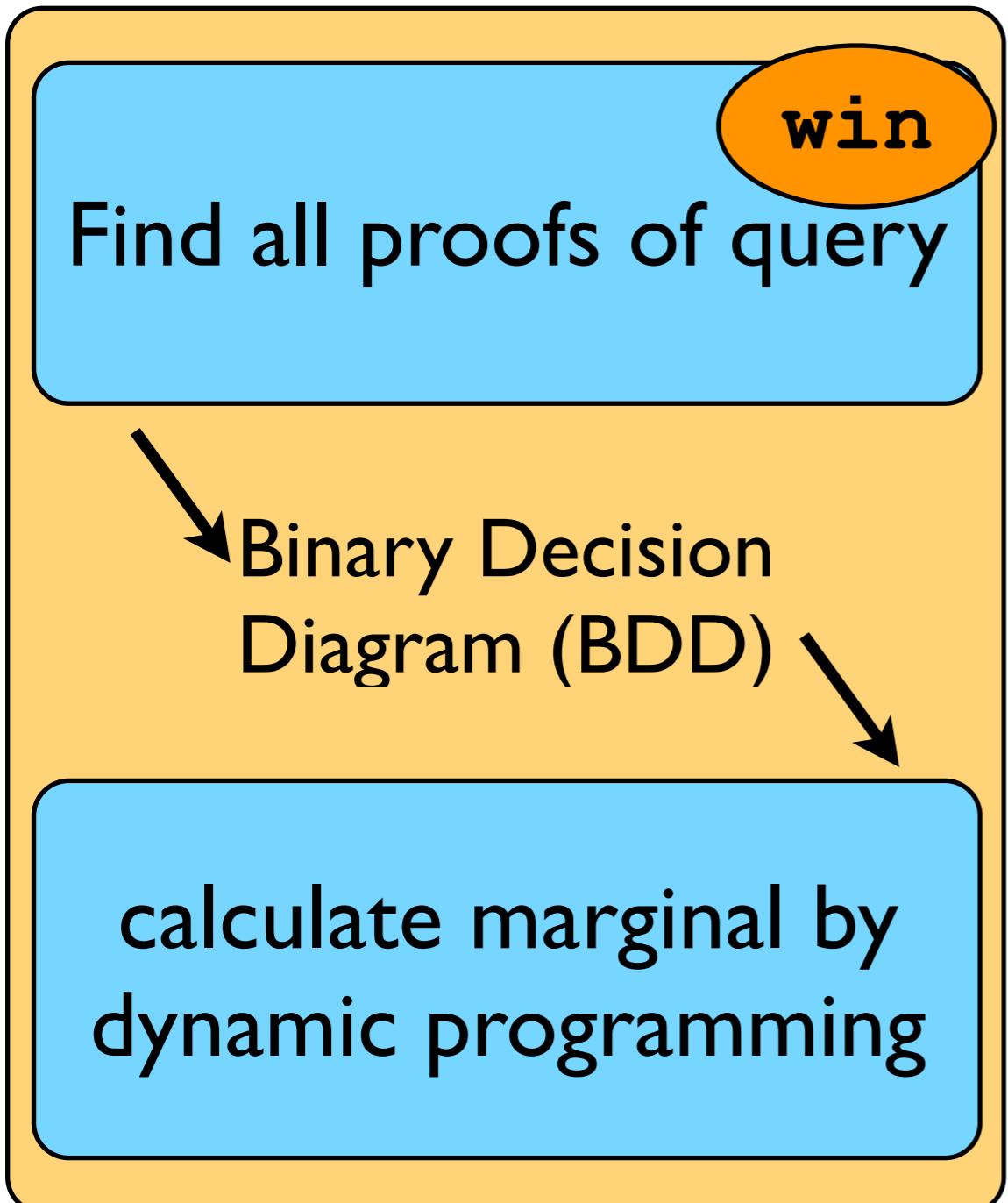


heads(1)
heads(2) & heads(3)

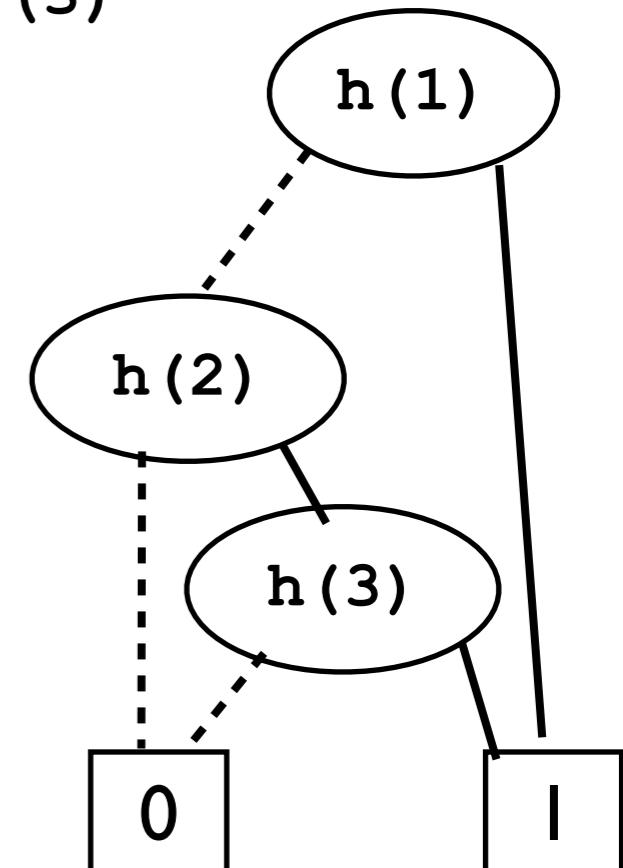
Initial Approach

(ProbLog1)

```
0.4 :: heads(1).  
0.7 :: heads(2).  
0.5 :: heads(3).  
win :- heads(1).  
win :- heads(2), heads(3).
```



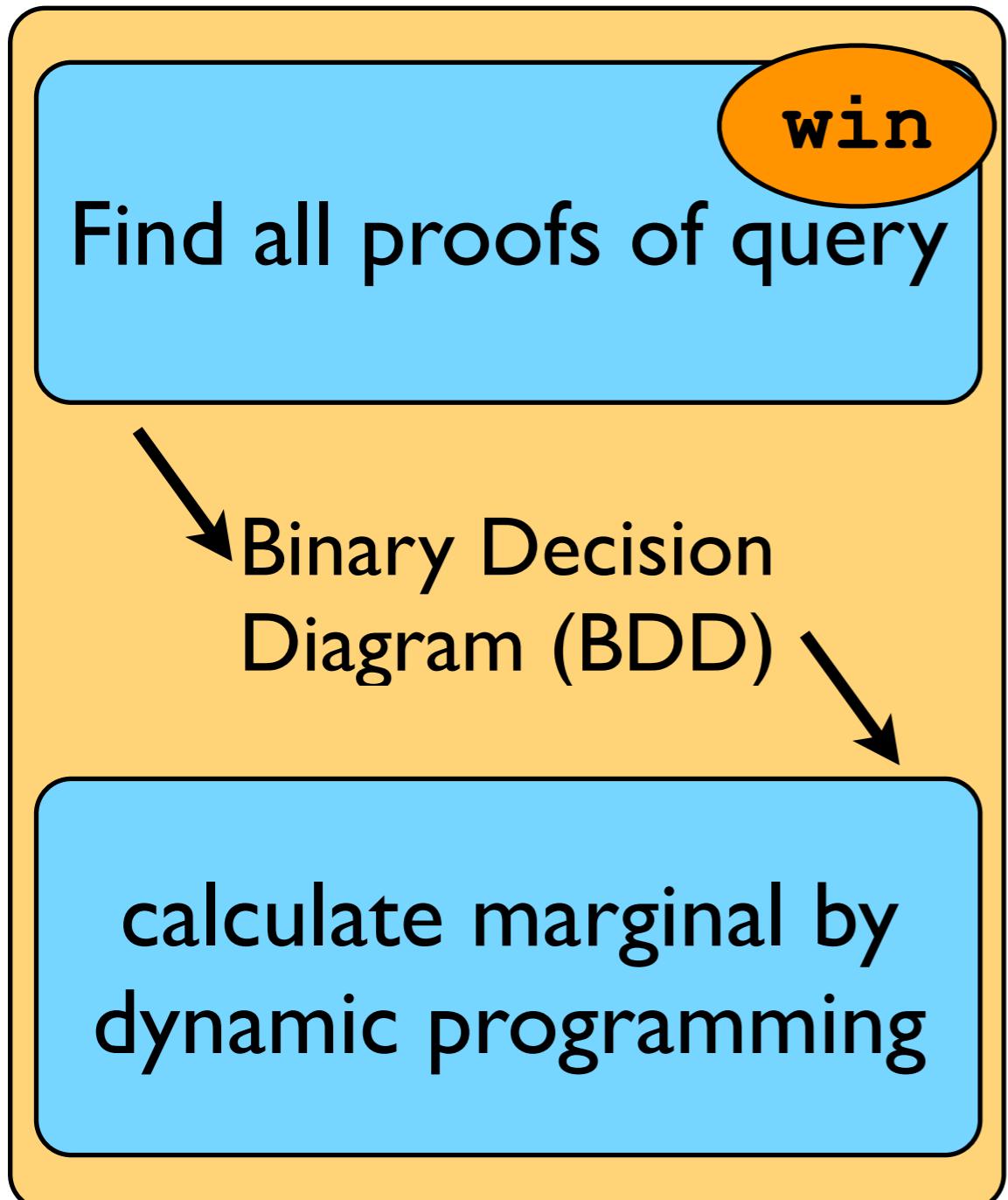
heads(1)
heads(2) & heads(3)



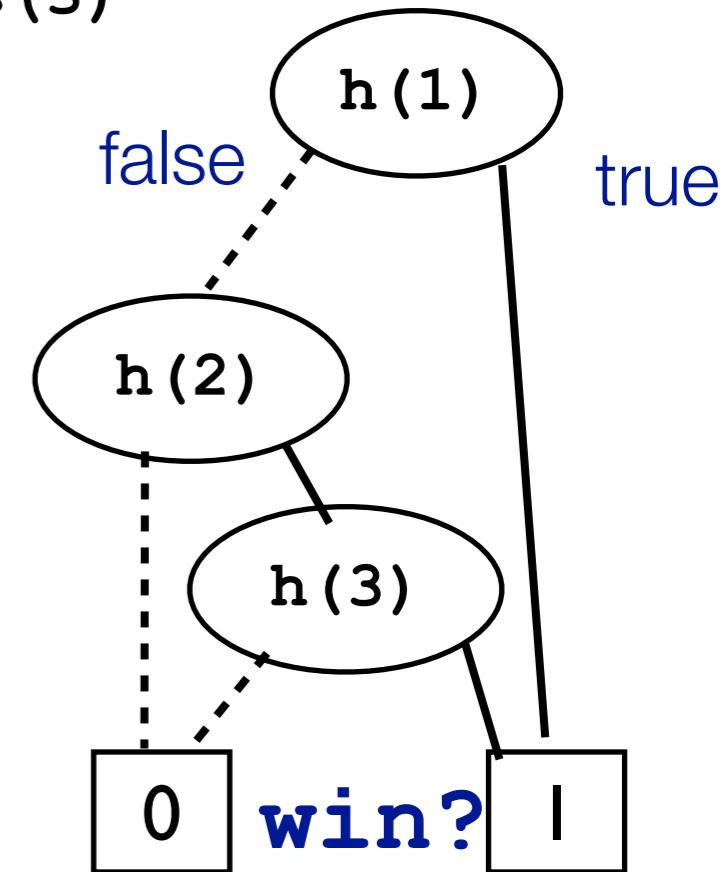
Initial Approach

(ProbLog1)

```
0.4 :: heads(1).  
0.7 :: heads(2).  
0.5 :: heads(3).  
win :- heads(1).  
win :- heads(2), heads(3).
```



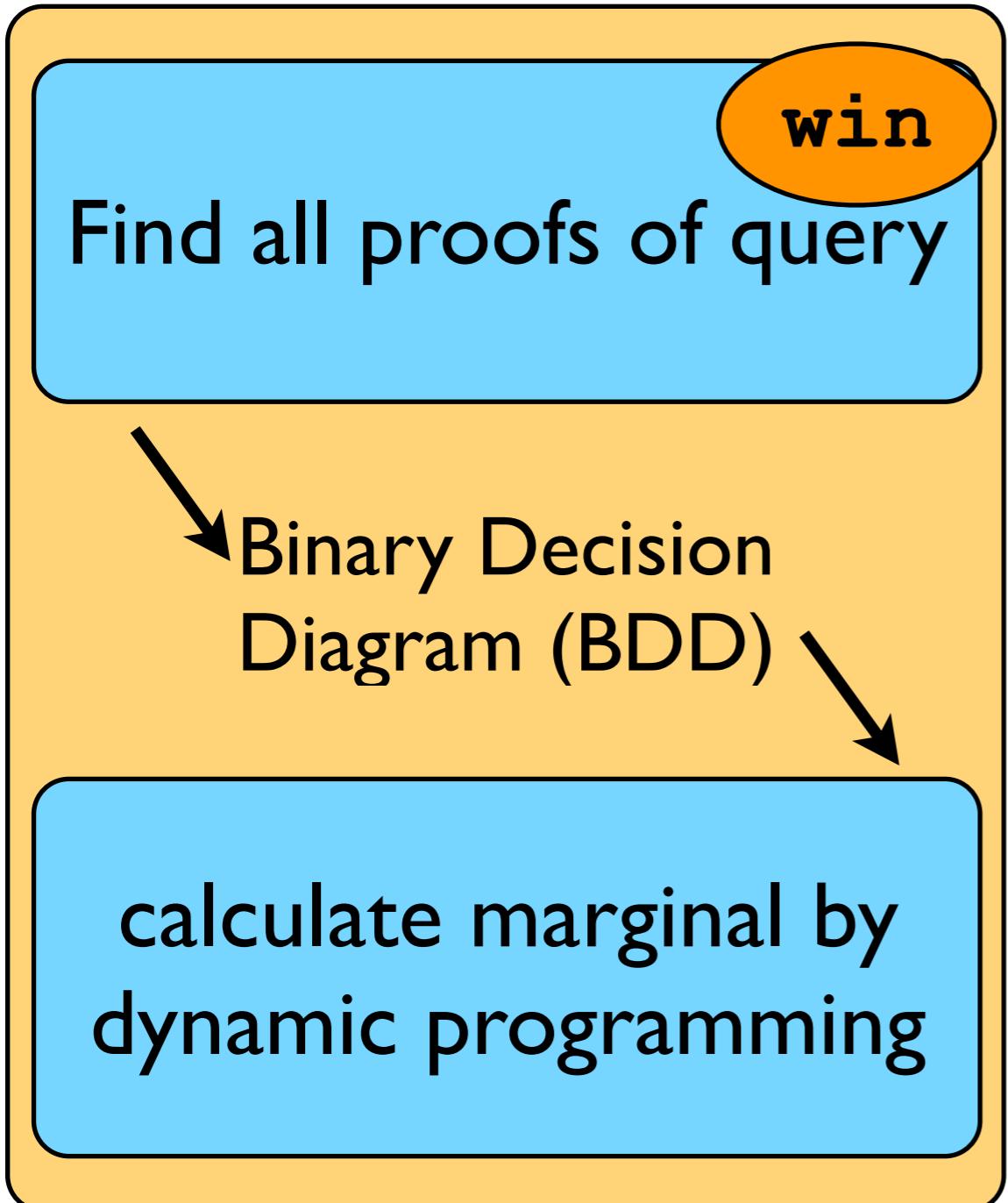
heads(1)
heads(2) & heads(3)



Initial Approach

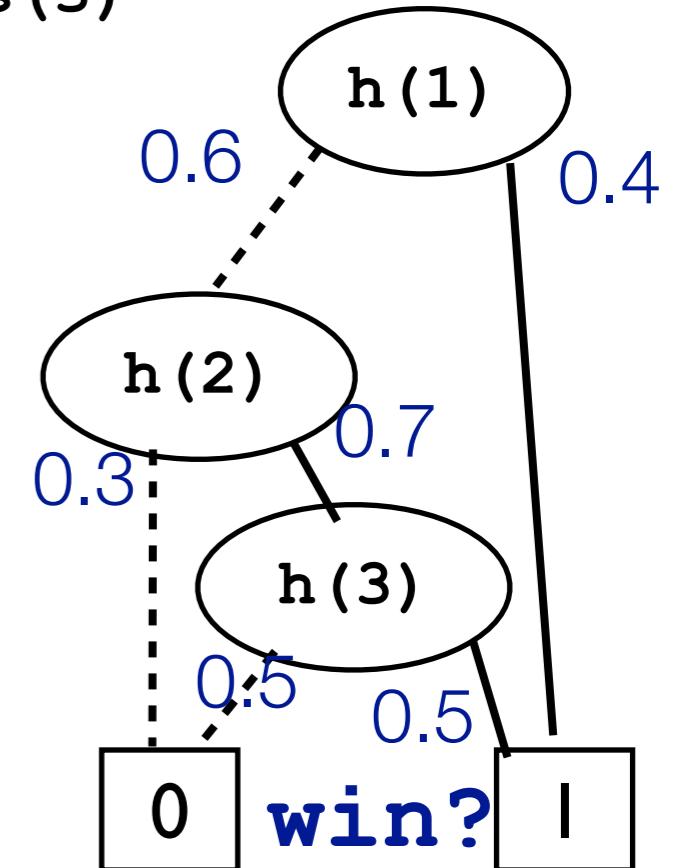
(ProbLog1)

```
0.4 :: heads(1).  
0.7 :: heads(2).  
0.5 :: heads(3).  
win :- heads(1).  
win :- heads(2), heads(3).
```



heads(1)
heads(2) & heads(3)

$P(\text{win}) =$
probability of
reaching 1-leaf



Answering Questions

1. using proofs
2. using models

Given:

program

queries

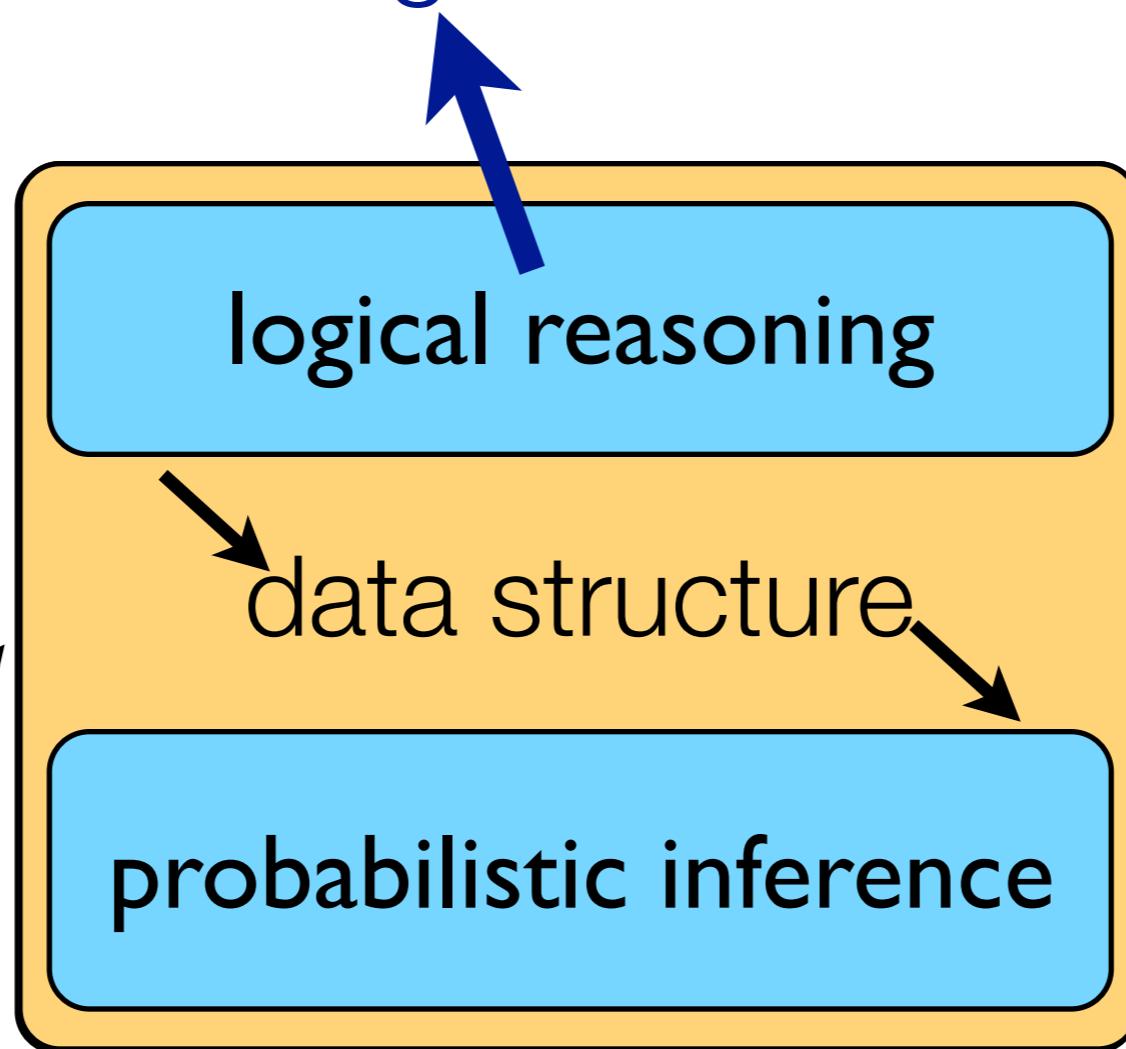
evidence

Find:

marginal probabilities

conditional probabilities

MPE state



Logical Reasoning: Models in Prolog

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

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

- Forward reasoning to construct unique model:

Logical Reasoning: Models in Prolog

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

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

- Forward reasoning to construct unique model:
 - Start with database facts

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

Logical Reasoning: Models in Prolog

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

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

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

- Forward reasoning to construct unique model:

- Start with database facts
- Use rules to add more facts

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

```
smokes (ann) .
```

Logical Reasoning: Models in Prolog

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

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

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

- Forward reasoning to construct unique model:

- Start with database facts
- Use rules to add more facts

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

```
smokes (ann) .  
smokes (bob) .
```

Logical Reasoning: Models in Prolog

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

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

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

- Forward reasoning to construct unique model:

- Start with database facts
- Use rules to add more facts

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

```
smokes (ann) .  
smokes (bob) .  
smokes (carl) .
```

Logical Reasoning: Models in Prolog

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

- Forward reasoning to construct unique model:
 - Start with database facts
 - Use rules to add more facts
- Query true iff in model

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

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

```
smokes(ann) .  
smokes(bob) .  
smokes(carl) .
```

Logical Reasoning: Models in Prolog

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

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

- Forward reasoning to construct unique model:
 - Start with database facts
 - Use rules to add more facts
- Query true iff in model
- **ProbLog**: each possible world is a model, probability of query is sum over models where query is true

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

```
smokes(ann).  
smokes(bob).  
smokes(carl).
```

Logical Reasoning: Models in Prolog

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

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

- Forward reasoning to construct unique model:
 - Start with database facts
 - Use rules to add more facts
- Query true iff in model
- **ProbLog**: each possible world is a model, probability of query is sum over models where query is true

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

```
smokes(ann).  
smokes(bob).  
smokes(carl).
```

→ weighted model counting

Weighted Model Counting

$$WMC(\phi) = \sum_{I_V \models \phi} \prod_{l \in I} w(l)$$

Weighted Model Counting

propositional formula in conjunctive normal form (CNF)

$$WMC(\phi) = \sum_{I_V \models \phi} \prod_{l \in I} w(l)$$

Weighted Model Counting

propositional formula in conjunctive normal form (CNF)

$$WMC(\phi) = \sum_{I_V \models \phi} \prod_{l \in I} w(l)$$

interpretations (truth
value assignments) of
propositional variables

Weighted Model Counting

propositional formula in conjunctive normal form (CNF)

$$WMC(\phi) = \sum_{I_V \models \phi} \prod_{l \in I} w(l)$$

interpretations (truth value assignments) of propositional variables

weight of literal

Weighted Model Counting

propositional formula in conjunctive normal form (CNF)

given by ProbLog program & query

$$WMC(\phi) = \sum_{I_V \models \phi} \prod_{l \in I} w(l)$$

interpretations (truth value assignments) of propositional variables

weight of literal

Weighted Model Counting

propositional formula in conjunctive normal form (CNF)

given by ProbLog program & query

$$WMC(\phi) = \sum_{I_V \models \phi} \prod_{l \in I} w(l)$$

interpretations (truth value assignments) of propositional variables

possible worlds

weight of literal

Weighted Model Counting

propositional formula in conjunctive normal form (CNF)

given by ProbLog program & query

$$WMC(\phi) = \sum_{I_V \models \phi} \prod_{l \in I} w(l)$$

interpretations (truth value assignments) of propositional variables
possible worlds

weight of literal
for p::f,
 $w(f) = p$
 $w(\text{not } f) = 1 - p$

Weight

$$P(Q) = \sum_{F \cup R \models Q} \prod_{f \in F} p(f) \prod_{f \notin F} 1 - p(f)$$

propositional formula in conjunctive normal form (CNF)

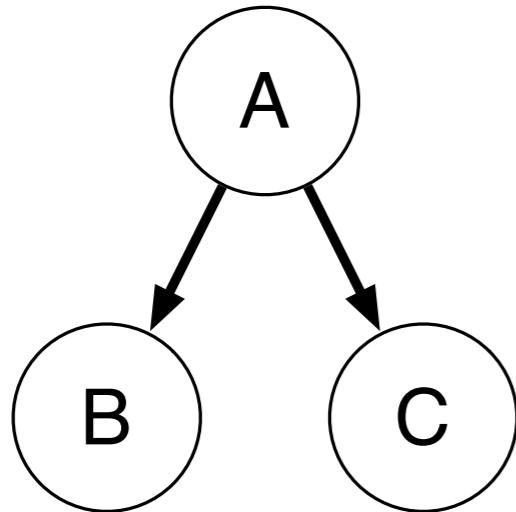
given by ProbLog program & query

$$WMC(\phi) = \sum_{I_V \models \phi} \prod_{l \in I} w(l)$$

interpretations (truth
value assignments) of
propositional variables
possible worlds

weight
of literal
for p::f,
 $w(f) = p$
 $w(\text{not } f) = 1 - p$

Bayesian net to WMC



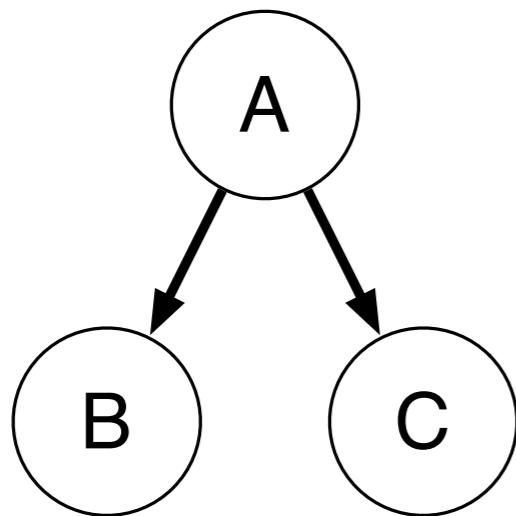
		A	C	Pr
		A	B	Pr
A				Pr
a ₁		a ₁	c ₁	0.1
		a ₁	b ₁	0.1
a ₂		a ₁	c ₂	0.2
		a ₁	b ₂	0.9
a ₁		a ₂	c ₃	0.7
		a ₂	b ₁	0.2
a ₂		a ₂	c ₁	0.01
		a ₂	b ₂	0.8
a ₂		a ₂	c ₂	0.09
		a ₂	c ₃	0.9



$$\begin{aligned}
 \Pr(B=1, C=1) &= \Pr(A=1, B=1, C=1) + \Pr(A=2, B=1, C=1) \\
 &= 0.1 \cdot 0.1 \cdot 0.1 + 0.9 \cdot 0.2 \cdot 0.01 \\
 &\quad \text{2 models}
 \end{aligned}$$

Variable A:	$\lambda_{a_1} \vee \lambda_{a_2}$	$\neg\lambda_{a_1} \vee \neg\lambda_{a_2}$
Variable B:	$\lambda_{b_1} \vee \lambda_{b_2}$	$\neg\lambda_{b_1} \vee \neg\lambda_{b_2}$
Variable C:	$\lambda_{c_1} \vee \lambda_{c_2} \vee \lambda_{c_3}$	$\neg\lambda_{c_1} \vee \neg\lambda_{c_2}$
		$\neg\lambda_{c_1} \vee \neg\lambda_{c_3}$
		$\neg\lambda_{c_2} \vee \neg\lambda_{c_3}$
CPT 1:	$\lambda_{a_1} \Leftrightarrow \theta_{a_1}$	$\lambda_{a_2} \Leftrightarrow \theta_{a_2}$
CPT 2:	$\lambda_{a_1} \wedge \lambda_{b_1} \Leftrightarrow \theta_{b_1 a_1}$	$\lambda_{a_2} \wedge \lambda_{b_1} \Leftrightarrow \theta_{b_1 a_2}$
	$\lambda_{a_1} \wedge \lambda_{b_2} \Leftrightarrow \theta_{b_2 a_1}$	$\lambda_{a_2} \wedge \lambda_{b_2} \Leftrightarrow \theta_{b_2 a_2}$
CPT 3:	$\lambda_{a_1} \wedge \lambda_{c_1} \Leftrightarrow \theta_{c_1 a_1}$	$\lambda_{a_2} \wedge \lambda_{c_1} \Leftrightarrow \theta_{c_1 a_2}$
	$\lambda_{a_1} \wedge \lambda_{c_2} \Leftrightarrow \theta_{c_2 a_1}$	$\lambda_{a_2} \wedge \lambda_{c_2} \Leftrightarrow \theta_{c_2 a_2}$
	$\lambda_{a_1} \wedge \lambda_{c_3} \Leftrightarrow \theta_{c_3 a_1}$	$\lambda_{a_2} \wedge \lambda_{c_3} \Leftrightarrow \theta_{c_3 a_2}$
	$W(\theta_{a_1}) = 0.1$	$W(\theta_{a_2}) = 0.9$
	$W(\theta_{b_1 a_1}) = 0.1$	$W(\theta_{b_2 a_1}) = 0.9$
	$W(\theta_{b_1 a_2}) = 0.2$	$W(\theta_{b_2 a_2}) = 0.8$
	$W(\theta_{c_1 a_1}) = 0.1$	$W(\theta_{c_2 a_1}) = 0.2$
	$W(\theta_{c_1 a_2}) = 0.01$	$W(\theta_{c_2 a_2}) = 0.09$
	$W(\theta_{c_3 a_1}) = 0.7$	$W(\theta_{c_3 a_2}) = 0.9$
	$W(\cdot) = 1$	

Bayesian net to WMC



		A	C	Pr
		A	B	Pr
		a ₁	b ₁	0.1
A	Pr			
	a ₁		a ₁	0.1
a ₂	0.9		a ₁	0.9
		a ₂	b ₁	0.2
		a ₂	b ₂	0.8
			a ₁	0.1
			c ₁	0.1
		a ₁	c ₂	0.2
		a ₁	c ₃	0.7
		a ₂	c ₁	0.01
		a ₂	c ₂	0.09
		a ₂	c ₃	0.9

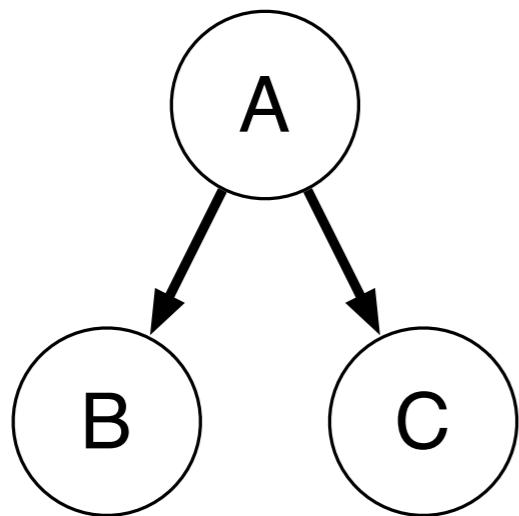
$$\begin{aligned}
 \Pr(B=1, C=1) &= \Pr(A=1, B=1, C=1) + \Pr(A=2, B=1, C=1) \\
 &= 0.1 \cdot 0.1 \cdot 0.1 + 0.9 \cdot 0.2 \cdot 0.01 \\
 &\quad \text{2 models}
 \end{aligned}$$



Variable A:	$\lambda_{a_1} \vee \lambda_{a_2}$	$\neg \lambda_{a_1} \vee \neg \lambda_{a_2}$
Variable B:	$\lambda_{b_1} \vee \lambda_{b_2}$	$\neg \lambda_{b_1} \vee \neg \lambda_{b_2}$
Variable C:	$\lambda_{c_1} \vee \lambda_{c_2} \vee \lambda_{c_3}$	$\neg \lambda_{c_1} \vee \neg \lambda_{c_2}$
		$\neg \lambda_{c_1} \vee \neg \lambda_{c_3}$
		$\neg \lambda_{c_2} \vee \neg \lambda_{c_3}$
CPT 1:	$\lambda_{a_1} \Leftrightarrow \theta_{a_1}$	$\lambda_{a_2} \Leftrightarrow \theta_{a_2}$
CPT 2:	$\lambda_{a_1} \wedge \lambda_{b_1} \Leftrightarrow \theta_{b_1 a_1}$	$\lambda_{a_2} \wedge \lambda_{b_1} \Leftrightarrow \theta_{b_1 a_2}$
	$\lambda_{a_1} \wedge \lambda_{b_2} \Leftrightarrow \theta_{b_2 a_1}$	$\lambda_{a_2} \wedge \lambda_{b_2} \Leftrightarrow \theta_{b_2 a_2}$
CPT 3:	$\lambda_{a_1} \wedge \lambda_{c_1} \Leftrightarrow \theta_{c_1 a_1}$	$\lambda_{a_2} \wedge \lambda_{c_1} \Leftrightarrow \theta_{c_1 a_2}$
	$\lambda_{a_1} \wedge \lambda_{c_2} \Leftrightarrow \theta_{c_2 a_1}$	$\lambda_{a_2} \wedge \lambda_{c_2} \Leftrightarrow \theta_{c_2 a_2}$
	$\lambda_{a_1} \wedge \lambda_{c_3} \Leftrightarrow \theta_{c_3 a_1}$	$\lambda_{a_2} \wedge \lambda_{c_3} \Leftrightarrow \theta_{c_3 a_2}$
	$W(\theta_{a_1}) = 0.1$	$W(\theta_{a_2}) = 0.9$
	$W(\theta_{b_1 a_1}) = 0.1$	$W(\theta_{b_2 a_1}) = 0.9$
	$W(\theta_{b_1 a_2}) = 0.2$	$W(\theta_{b_2 a_2}) = 0.8$
	$W(\theta_{c_1 a_1}) = 0.1$	$W(\theta_{c_2 a_1}) = 0.2$
	$W(\theta_{c_1 a_2}) = 0.01$	$W(\theta_{c_2 a_2}) = 0.09$
	$W(\theta_{c_3 a_1}) = 0.7$	$W(\theta_{c_3 a_2}) = 0.9$
	$W(\cdot) = 1$	

M. Chavira and A. Darwiche. On probabilistic inference by weighted model counting. Artificial Intelligence, 172(6):772–799, 2008.

Bayesian net to WMC



		A	C	Pr
		A	B	Pr
A				Pr
a ₁		a ₁	c ₁	0.1
		a ₁	b ₁	0.1
a ₂		a ₁	b ₂	0.9
		a ₂	b ₁	0.2
		a ₂	b ₂	0.8
		a ₁	c ₂	0.2
		a ₁	c ₃	0.7
		a ₂	c ₁	0.01
		a ₂	c ₂	0.09
		a ₂	c ₃	0.9

$$\begin{aligned}
 \Pr(B=1, C=1) &= \Pr(A=1, B=1, C=1) + \Pr(A=2, B=1, C=1) \\
 &= 0.1 \cdot 0.1 \cdot 0.1 + 0.9 \cdot 0.2 \cdot 0.01 \\
 &\quad \text{2 models}
 \end{aligned}$$



Variable A: $\lambda_{a_1} \vee \lambda_{a_2}$

$\neg\lambda_{a_1} \vee \neg\lambda_{a_2}$

Variable B: $\lambda_{b_1} \vee \lambda_{b_2}$

$\neg\lambda_{b_1} \vee \neg\lambda_{b_2}$

Variable C: $\lambda_{c_1} \vee \lambda_{c_2} \vee \lambda_{c_3}$

$\neg\lambda_{c_1} \vee \neg\lambda_{c_2}$

$\neg\lambda_{c_1} \vee \neg\lambda_{c_3}$

$\neg\lambda_{c_2} \vee \neg\lambda_{c_3}$

$\lambda_{a_2} \Leftrightarrow \theta_{a_2}$

CPT 1: $\lambda_{a_1} \Leftrightarrow \theta_{a_1}$

$\lambda_{a_2} \wedge \lambda_{b_1} \Leftrightarrow \theta_{b_1|a_1}$

CPT 2: $\lambda_{a_1} \wedge \lambda_{b_2} \Leftrightarrow \theta_{b_2|a_1}$

$\lambda_{a_2} \wedge \lambda_{b_2} \Leftrightarrow \theta_{b_2|a_2}$

CPT 3: $\lambda_{a_1} \wedge \lambda_{c_1} \Leftrightarrow \theta_{c_1|a_1}$

$\lambda_{a_2} \wedge \lambda_{c_1} \Leftrightarrow \theta_{c_1|a_2}$

$\lambda_{a_1} \wedge \lambda_{c_2} \Leftrightarrow \theta_{c_2|a_1}$

$\lambda_{a_2} \wedge \lambda_{c_2} \Leftrightarrow \theta_{c_2|a_2}$

$\lambda_{a_1} \wedge \lambda_{c_3} \Leftrightarrow \theta_{c_3|a_1}$

$\lambda_{a_2} \wedge \lambda_{c_3} \Leftrightarrow \theta_{c_3|a_2}$

$$W(\theta_{a_1}) = 0.1$$

$$W(\theta_{a_2}) = 0.9$$

$$W(\cdot) = 1$$

$$W(\theta_{b_1|a_1}) = 0.1$$

$$W(\theta_{b_2|a_1}) = 0.9$$

$$W(\theta_{b_1|a_2}) = 0.2$$

$$W(\theta_{b_2|a_2}) = 0.8$$

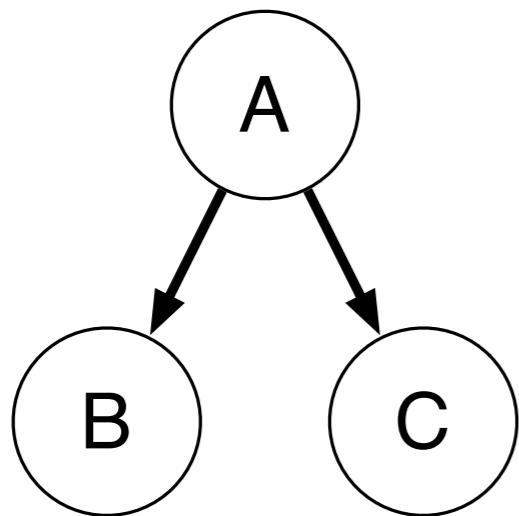
$$W(\theta_{c_1|a_1}) = 0.1$$

$$W(\theta_{c_2|a_1}) = 0.2 \quad W(\theta_{c_3|a_1}) = 0.7$$

$$W(\theta_{c_1|a_2}) = 0.01$$

$$W(\theta_{c_2|a_2}) = 0.09 \quad W(\theta_{c_3|a_2}) = 0.9$$

Bayesian net to WMC



		A	C	Pr
		A	B	Pr
A				Pr
a ₁		a ₁	c ₁	0.1
		a ₁	b ₁	0.1
a ₂		a ₁	b ₂	0.9
		a ₂	b ₁	0.2
		a ₂	b ₂	0.8
a ₁		a ₁	c ₂	0.2
		a ₂	c ₂	0.09
a ₂		a ₁	c ₃	0.7
		a ₂	c ₁	0.01
		a ₂	c ₃	0.9

$$\begin{aligned}
 \Pr(B=1, C=1) &= \Pr(A=1, B=1, C=1) + \Pr(A=2, B=1, C=1) \\
 &= 0.1 \cdot 0.1 \cdot 0.1 + 0.9 \cdot 0.2 \cdot 0.01 \\
 &\quad \text{2 models}
 \end{aligned}$$



Variable A: $\lambda_{a_1} \vee \lambda_{a_2}$

$\neg\lambda_{a_1} \vee \neg\lambda_{a_2}$

Variable B: $\lambda_{b_1} \vee \lambda_{b_2}$

$\neg\lambda_{b_1} \vee \neg\lambda_{b_2}$

Variable C: $\lambda_{c_1} \vee \lambda_{c_2} \vee \lambda_{c_3}$

$\neg\lambda_{c_1} \vee \neg\lambda_{c_2}$

$\neg\lambda_{c_1} \vee \neg\lambda_{c_3}$

$\neg\lambda_{c_2} \vee \neg\lambda_{c_3}$

$\lambda_{a_2} \Leftrightarrow \theta_{a_2}$

CPT 1: $\lambda_{a_1} \Leftrightarrow \theta_{a_1}$

$\lambda_{a_2} \wedge \lambda_{b_1} \Leftrightarrow \theta_{b_1|a_1}$

CPT 2: $\lambda_{a_1} \wedge \lambda_{b_1} \Leftrightarrow \theta_{b_1|a_1}$

$\lambda_{a_2} \wedge \lambda_{b_2} \Leftrightarrow \theta_{b_2|a_2}$

CPT 3: $\lambda_{a_1} \wedge \lambda_{c_1} \Leftrightarrow \theta_{c_1|a_1}$

$\lambda_{a_2} \wedge \lambda_{c_1} \Leftrightarrow \theta_{c_1|a_2}$

$\lambda_{a_1} \wedge \lambda_{c_2} \Leftrightarrow \theta_{c_2|a_1}$

$\lambda_{a_2} \wedge \lambda_{c_2} \Leftrightarrow \theta_{c_2|a_2}$

$\lambda_{a_1} \wedge \lambda_{c_3} \Leftrightarrow \theta_{c_3|a_1}$

$\lambda_{a_2} \wedge \lambda_{c_3} \Leftrightarrow \theta_{c_3|a_2}$

$$W(\theta_{a_1}) = 0.1$$

$$W(\theta_{a_2}) = 0.9$$

$$W(\cdot) = 1$$

$$W(\theta_{b_1|a_1}) = 0.1$$

$$W(\theta_{b_2|a_1}) = 0.9$$

$$W(\theta_{b_1|a_2}) = 0.2$$

$$W(\theta_{b_2|a_2}) = 0.8$$

$$W(\theta_{c_1|a_1}) = 0.1$$

$$W(\theta_{c_2|a_1}) = 0.2$$

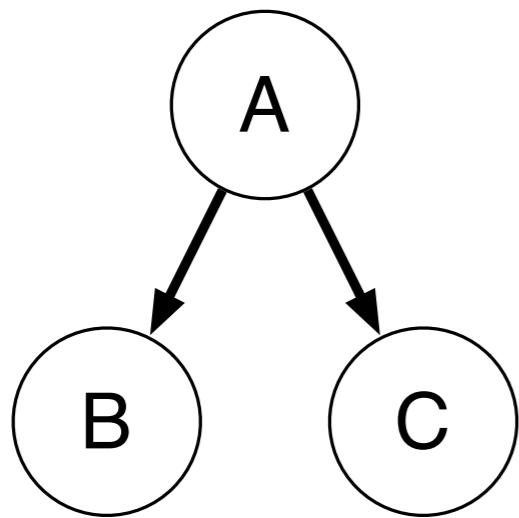
$$W(\theta_{c_3|a_1}) = 0.7$$

$$W(\theta_{c_1|a_2}) = 0.01$$

$$W(\theta_{c_2|a_2}) = 0.09$$

$$W(\theta_{c_3|a_2}) = 0.9$$

Bayesian net to WMC



		A	C	Pr
		A	B	Pr
		a_1	c_1	0.1
		a_1	b_1	0.1
		a_1	c_2	0.2
		a_1	c_3	0.7
	a_1			
	0.1			
	a_2			
	0.9			
		a_2	b_1	0.2
		a_2	c_1	0.01
		a_2	b_2	0.8
		a_2	c_2	0.09
		a_2	c_3	0.9

$$\begin{aligned}
 \Pr(B=1, C=1) &= \Pr(A=1, B=1, C=1) + \Pr(A=2, B=1, C=1) \\
 &= 0.1 \cdot 0.1 \cdot 0.1 + 0.9 \cdot 0.2 \cdot 0.01 \\
 &\quad \text{2 models}
 \end{aligned}$$



Variable A: $\lambda_{a_1} \vee \lambda_{a_2}$

$\neg\lambda_{a_1} \vee \neg\lambda_{a_2}$

Variable B: $\lambda_{b_1} \vee \cancel{\lambda_{b_2}}$

$\cancel{\lambda_{b_1}} \vee \neg\lambda_{b_2}$

Variable C: $\lambda_{c_1} \vee \cancel{\lambda_{c_2}} \vee \cancel{\lambda_{c_3}}$

$\cancel{\lambda_{c_1}} \vee \neg\lambda_{c_2}$

$\cancel{\lambda_{c_1}} \vee \neg\lambda_{c_3}$

$\neg\lambda_{c_2} \vee \neg\lambda_{c_3}$

$\lambda_{a_2} \Leftrightarrow \theta_{a_2}$

CPT 1: $\lambda_{a_1} \Leftrightarrow \theta_{a_1}$

$\lambda_{a_2} \wedge \lambda_{b_1} \Leftrightarrow \theta_{b_1|a_1}$

$\lambda_{a_1} \wedge \cancel{\lambda_{b_2}} \Leftrightarrow \theta_{b_2|a_1}$

$\lambda_{a_2} \wedge \cancel{\lambda_{b_2}} \Leftrightarrow \theta_{b_2|a_2}$

$\lambda_{a_1} \wedge \lambda_{c_1} \Leftrightarrow \theta_{c_1|a_1}$

$\lambda_{a_2} \wedge \lambda_{c_1} \Leftrightarrow \theta_{c_1|a_2}$

$\lambda_{a_1} \wedge \cancel{\lambda_{c_2}} \Leftrightarrow \theta_{c_2|a_1}$

$\lambda_{a_2} \wedge \cancel{\lambda_{c_2}} \Leftrightarrow \theta_{c_2|a_2}$

$\lambda_{a_1} \wedge \cancel{\lambda_{c_3}} \Leftrightarrow \theta_{c_3|a_1}$

$\lambda_{a_2} \wedge \cancel{\lambda_{c_3}} \Leftrightarrow \theta_{c_3|a_2}$

$$W(\theta_{a_1}) = 0.1$$

$$W(\theta_{a_2}) = 0.9$$

$$W(\cdot) = 1$$

$$W(\theta_{b_1|a_1}) = 0.1$$

$$W(\theta_{b_2|a_1}) = 0.9$$

$$W(\theta_{b_1|a_2}) = 0.2$$

$$W(\theta_{b_2|a_2}) = 0.8$$

$$W(\theta_{c_1|a_1}) = 0.1$$

$$W(\theta_{c_2|a_1}) = 0.2$$

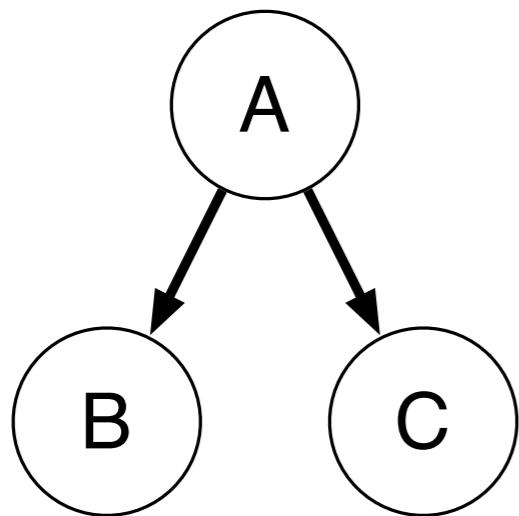
$$W(\theta_{c_3|a_1}) = 0.7$$

$$W(\theta_{c_1|a_2}) = 0.01$$

$$W(\theta_{c_2|a_2}) = 0.09$$

$$W(\theta_{c_3|a_2}) = 0.9$$

Bayesian net to WMC



		A	C	Pr
		A	B	Pr
		a_1	c_1	0.1
		a_1	b_1	0.1
		a_1	c_2	0.2
		a_1	c_3	0.7
		a_2	b_1	0.2
		a_2	c_1	0.01
		a_2	b_2	0.8
		a_2	c_2	0.09
		a_2	c_3	0.9



$$\begin{aligned}
 \Pr(B=1, C=1) &= \Pr(A=1, B=1, C=1) + \Pr(A=2, B=1, C=1) \\
 &= 0.1 \cdot 0.1 \cdot 0.1 + 0.9 \cdot 0.2 \cdot 0.01 \\
 &\quad \text{2 models}
 \end{aligned}$$

Variable A:	$\lambda_{a_1} \vee \lambda_{a_2}$	$\neg \lambda_{a_1} \vee \neg \lambda_{a_2}$
Variable B:	$\lambda_{b_1} \vee \lambda_{b_2}$	$\neg \lambda_{b_1} \vee \neg \lambda_{b_2}$
Variable C:	$\lambda_{c_1} \vee \lambda_{c_2} \vee \lambda_{c_3}$	$\neg \lambda_{c_1} \vee \neg \lambda_{c_2}$
	$\neg \lambda_{c_1} \vee \neg \lambda_{c_3}$	$\neg \lambda_{c_2} \vee \neg \lambda_{c_3}$
	$\neg \lambda_{c_2} \vee \neg \lambda_{c_1}$	$\lambda_{a_2} \Leftrightarrow \theta_{a_1}$
CPT 1:	$\lambda_{a_1} \Leftrightarrow \theta_{a_1}$	$\lambda_{a_2} \Leftrightarrow \theta_{a_2}$
CPT 2:	$\lambda_{a_1} \wedge \lambda_{b_1} \Leftrightarrow \theta_{b_1 a_1}$	$\lambda_{a_2} \wedge \lambda_{b_1} \Leftrightarrow \theta_{b_1 a_2}$
	$\lambda_{a_1} \wedge \lambda_{b_2} \Leftrightarrow \theta_{b_2 a_1}$	$\lambda_{a_2} \wedge \lambda_{b_2} \Leftrightarrow \theta_{b_2 a_2}$
CPT 3:	$\lambda_{a_1} \wedge \lambda_{c_1} \Leftrightarrow \theta_{c_1 a_1}$	$\lambda_{a_2} \wedge \lambda_{c_1} \Leftrightarrow \theta_{c_1 a_2}$
	$\lambda_{a_1} \wedge \lambda_{c_2} \Leftrightarrow \theta_{c_2 a_1}$	$\lambda_{a_2} \wedge \lambda_{c_2} \Leftrightarrow \theta_{c_2 a_2}$
	$\lambda_{a_1} \wedge \lambda_{c_3} \Leftrightarrow \theta_{c_3 a_1}$	$\lambda_{a_2} \wedge \lambda_{c_3} \Leftrightarrow \theta_{c_3 a_2}$
		$W(\theta_{a_1}) = 0.1$
		$W(\theta_{a_2}) = 0.9$
		$W(\cdot) = 1$
	$W(\theta_{b_1 a_1}) = 0.1$	$W(\theta_{b_2 a_1}) = 0.9$
	$W(\theta_{b_1 a_2}) = 0.2$	$W(\theta_{b_2 a_2}) = 0.8$
	$W(\theta_{c_1 a_1}) = 0.1$	$W(\theta_{c_2 a_1}) = 0.2$
	$W(\theta_{c_1 a_2}) = 0.01$	$W(\theta_{c_2 a_2}) = 0.09$
	$W(\theta_{c_3 a_1}) = 0.7$	$W(\theta_{c_3 a_2}) = 0.9$

M. Chavira and A. Darwiche. On probabilistic inference by weighted model counting. Artificial Intelligence, 172(6):772–799, 2008.

ProbLog → CNF

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

```
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).
```

ProbLog → CNF

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

```
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).
```

- Find relevant ground rules by backward reasoning

ProbLog → CNF

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

```
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) .
```

- Find relevant ground rules by backward reasoning

```
smokes(carl) :- influences(bob,carl) , smokes(bob) .  
smokes(bob) :- stress(bob) .  
smokes(bob) :- influences(ann,bob) , smokes(ann) .  
smokes(ann) :- stress(ann) .
```

ProbLog → CNF

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

```
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) .
```

- Find relevant ground rules by backward reasoning

```
smokes(carl) :- influences(bob,carl) , smokes(bob) .  
smokes(bob) :- stress(bob) .  
smokes(bob) :- influences(ann,bob) , smokes(ann) .  
smokes(ann) :- stress(ann) .
```

- Convert to propositional logic formula

ProbLog → CNF

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

```
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) .
```

- Find relevant ground rules by backward reasoning

```
smokes(carl) :- influences(bob,carl) , smokes(bob) .  
smokes(bob) :- stress(bob) .  
smokes(bob) :- influences(ann,bob) , smokes(ann) .  
smokes(ann) :- stress(ann) .
```

- Convert to propositional logic formula

$$\begin{aligned} \text{sm}(c) &\leftrightarrow (\text{i}(b,c) \wedge \text{sm}(b)) \\ \wedge \text{sm}(b) &\leftrightarrow (\text{st}(b) \vee (\text{i}(a,b) \wedge \text{sm}(a))) \\ \wedge \text{sm}(a) &\leftrightarrow \text{st}(a) \end{aligned}$$

ProbLog → CNF

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

```
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) .
```

- Find relevant ground rules by backward reasoning

```
smokes(carl) :- influences(bob,carl) , smokes(bob) .  
smokes(bob) :- stress(bob) .  
smokes(bob) :- influences(ann,bob) , smokes(ann) .  
smokes(ann) :- stress(ann) .
```

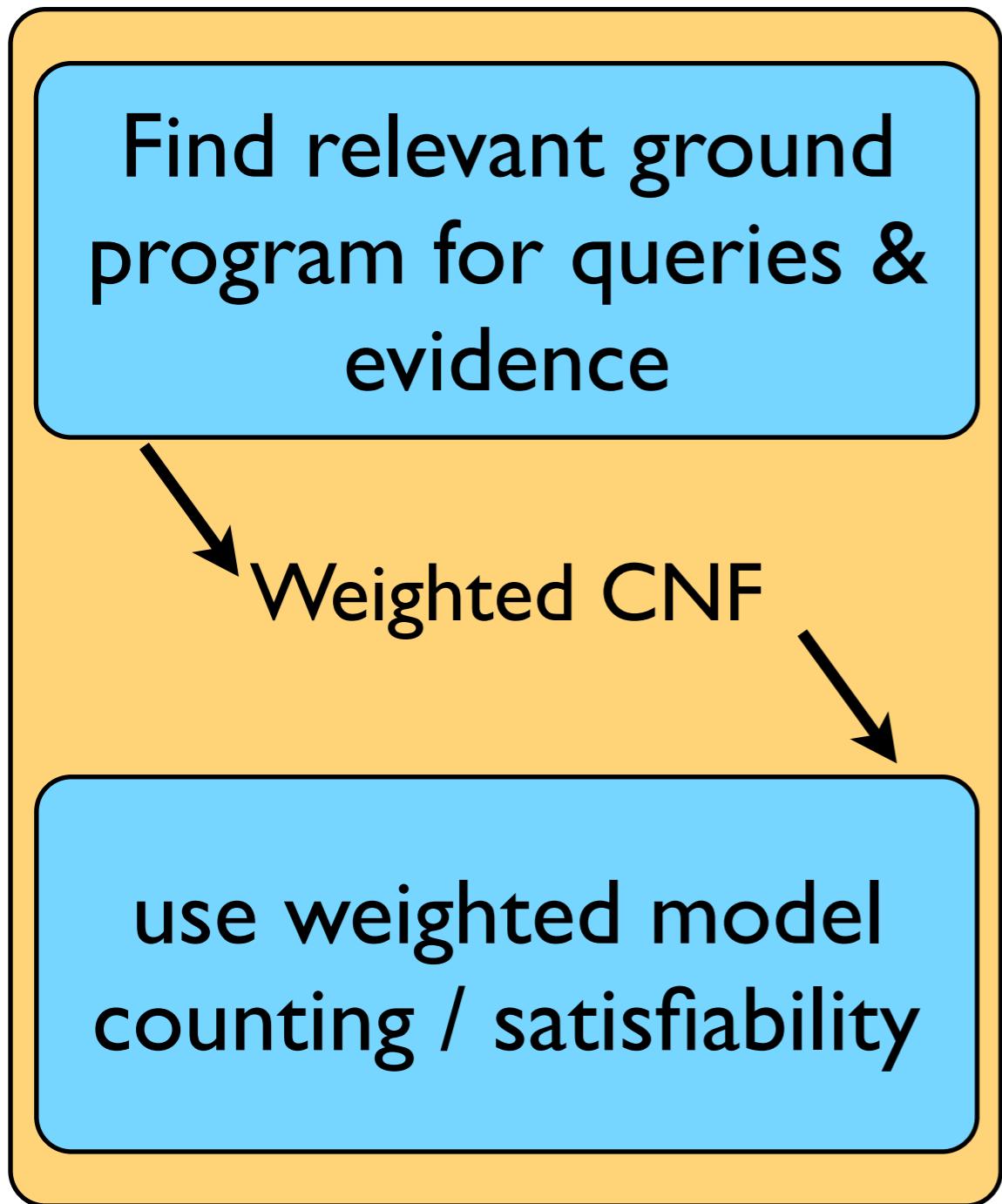
- Convert to propositional logic formula

$$\begin{aligned} \text{sm}(c) &\leftrightarrow (\text{i}(b,c) \wedge \text{sm}(b)) \\ \wedge \text{sm}(b) &\leftrightarrow (\text{st}(b) \vee (\text{i}(a,b) \wedge \text{sm}(a))) \\ \wedge \text{sm}(a) &\leftrightarrow \text{st}(a) \end{aligned}$$

- Rewrite in CNF (as usual)

Current Approach

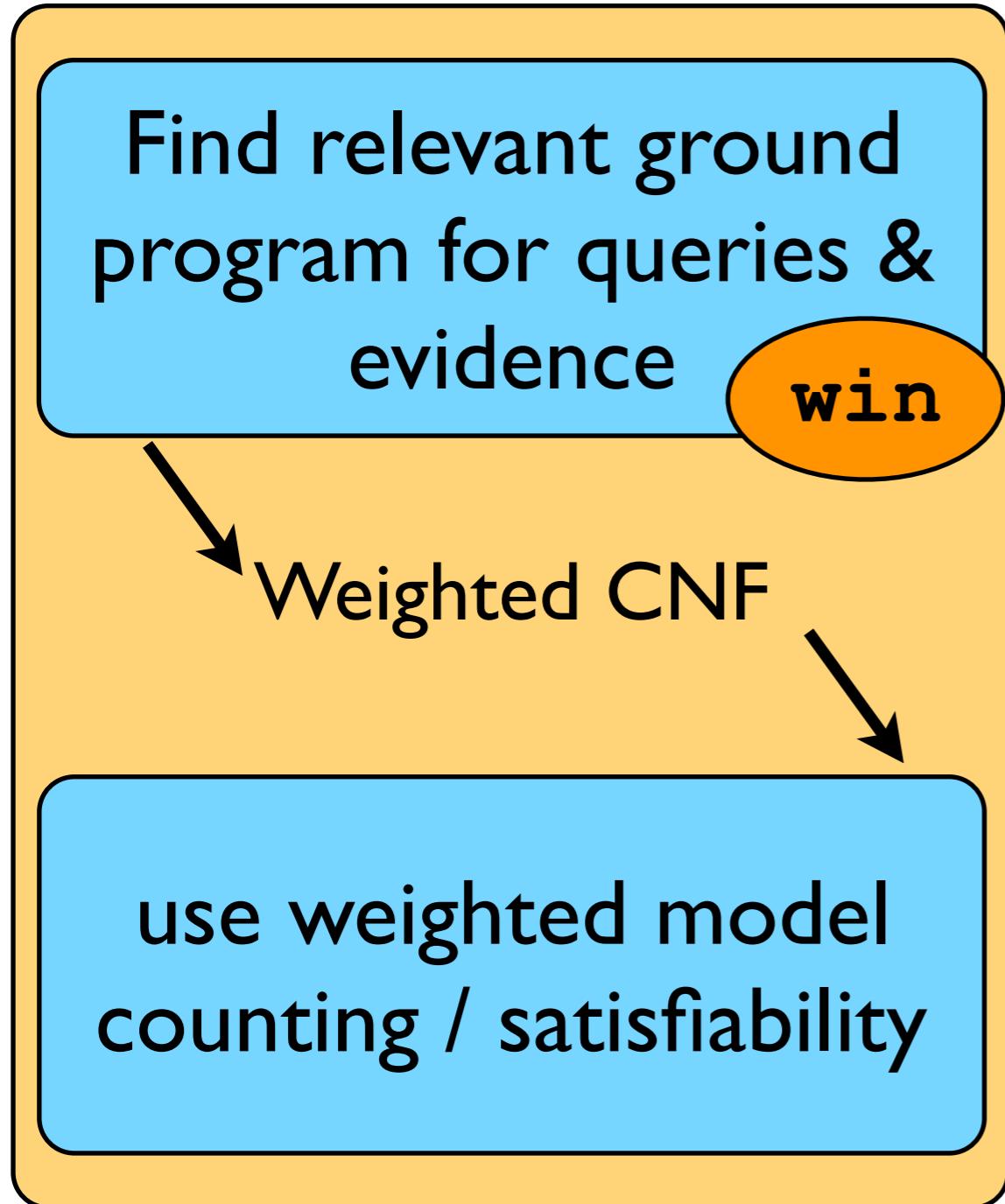
(ProbLog2)



Current Approach

(ProbLog2)

```
0.4 :: heads(1).  
0.7 :: heads(2).  
0.5 :: heads(3).  
win :- heads(1).  
win :- heads(2),  
      heads(3).
```



Current Approach

(ProbLog2)

```
0.4 :: heads(1).  
0.7 :: heads(2).  
0.5 :: heads(3).  
win :- heads(1).  
win :- heads(2),  
      heads(3).
```

Find relevant ground
program for queries &
evidence

win

Weighted CNF

use weighted model
counting / satisfiability

```
win :- heads(1).  
win :- heads(2), heads(3).
```

Current Approach

(ProbLog2)

```
0.4 :: heads(1).  
0.7 :: heads(2).  
0.5 :: heads(3).  
win :- heads(1).  
win :- heads(2),  
      heads(3).
```

Find relevant ground
program for queries &
evidence

win

Weighted CNF

use weighted model
counting / satisfiability

win :- heads(1).
win :- heads(2), heads(3).



$\text{win} \leftrightarrow h(1) \vee (h(2) \wedge h(3))$

Current Approach

(ProbLog2)

```
0.4 :: heads(1).  
0.7 :: heads(2).  
0.5 :: heads(3).  
win :- heads(1).  
win :- heads(2),  
      heads(3).
```

Find relevant ground
program for queries &
evidence

win

Weighted CNF

use weighted model
counting / satisfiability

```
win :- heads(1).  
win :- heads(2), heads(3).  
  
↓  
win ↔ h(1) ∨ (h(2) ∧ h(3))  
  
↓  
(¬win ∨ h(1) ∨ h(2))  
∧ (¬win ∨ h(1) ∨ h(3))  
    ∧ (win ∨ ¬h(1))  
    ∧ (win ∨ ¬h(2) ∨ ¬h(3))
```

Current Approach

(ProbLog2)

```
0.4 :: heads(1).  
0.7 :: heads(2).  
0.5 :: heads(3).  
win :- heads(1).  
win :- heads(2),  
      heads(3).
```

Find relevant ground
program for queries &
evidence

win

Weighted CNF

use weighted model
counting / satisfiability

win :- heads(1).
win :- heads(2), heads(3).



win \leftrightarrow h(1) \vee (h(2) \wedge h(3))



(\neg win \vee h(1) \vee h(2))
 \wedge (\neg win \vee h(1) \vee h(3))
 \wedge (win \vee \neg h(1))
 \wedge (win \vee \neg h(2) \vee \neg h(3))

h(1) \rightarrow 0.4

\neg h(1) \rightarrow 0.6

h(2) \rightarrow 0.7

\neg h(2) \rightarrow 0.3

h(3) \rightarrow 0.5

\neg h(3) \rightarrow 0.5

Current Approach

(ProbLog2)

```
0.4 :: heads(1).  
0.7 :: heads(2).  
0.5 :: heads(3).  
win :- heads(1).  
win :- heads(2),  
      heads(3).
```

Find relevant ground
program for queries &
evidence

win

Weighted CNF

use weighted model
counting / satisfiability

win :- heads(1).
win :- heads(2), heads(3).

win \leftrightarrow h(1) \vee (h(2) \wedge h(3))

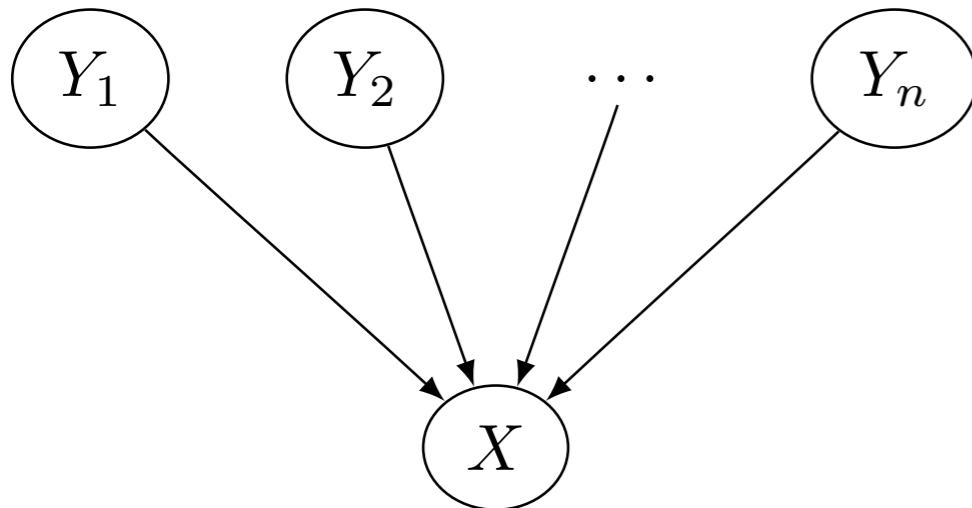
$(\neg \text{win} \vee \text{h}(1) \vee \text{h}(2))$
 $\wedge (\neg \text{win} \vee \text{h}(1) \vee \text{h}(3))$
 $\wedge (\text{win} \vee \neg \text{h}(1))$
 $\wedge (\text{win} \vee \neg \text{h}(2) \vee \neg \text{h}(3))$

use
standard
tool

$\text{h}(1) \rightarrow 0.4$ $\text{h}(2) \rightarrow 0.7$ $\text{h}(3) \rightarrow 0.5$
 $\neg \text{h}(1) \rightarrow 0.6$ $\neg \text{h}(2) \rightarrow 0.3$ $\neg \text{h}(3) \rightarrow 0.5$

Example: Encoding

Bayesian Network



Probabilistic Logic

0.4::y(1).
0.3::y(2).
...
0.5::y(n).
0.8::x :- y(N).

Propositional Logic

$$y(1) \Leftrightarrow p(1,1)$$

$$y(2) \Leftrightarrow p(1,2)$$

...

$$y(n) \Leftrightarrow p(1,n)$$

$$\begin{aligned} x \Leftrightarrow & (y(1) \wedge p(2,1)) \vee \dots \\ & \vee (y(n) \wedge p(2,n)) \end{aligned}$$

$$w(p(1,1)) = 0.4$$

$$w(p(1,2)) = 0.3$$

...

$$w(p(1,n)) = 0.5$$

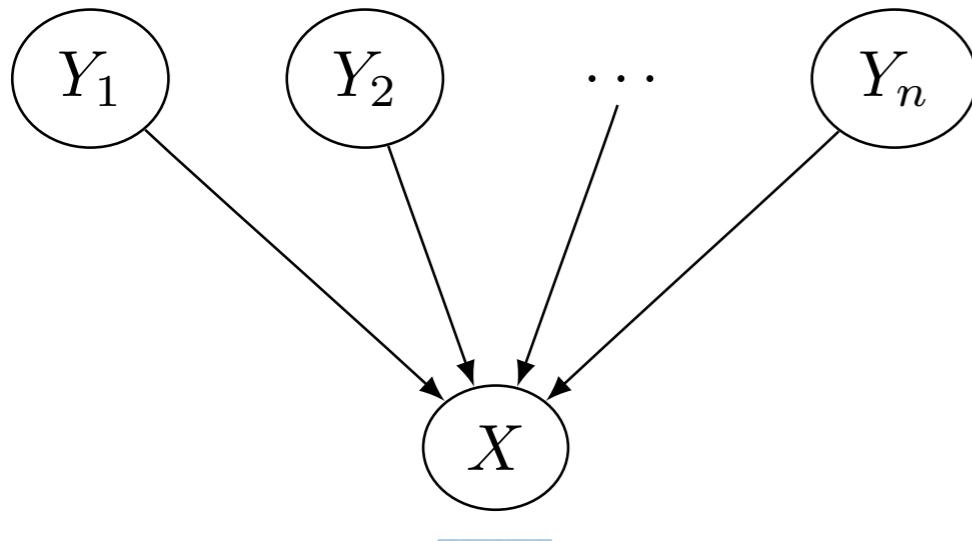
$$w(p(2,1)) = 0.8$$

...

$$w(p(2,n)) = 0.8$$

Example: Encoding

Bayesian Network



Probabilistic Logic

0.4::y(1).
0.3::y(2).
...
0.5::y(n).
0.8::x :- y(N).

Propositional Logic

$$y(1) \Leftrightarrow p(1,1)$$

$$y(2) \Leftrightarrow p(1,2)$$

...

$$y(n) \Leftrightarrow p(1,n)$$

$$\begin{aligned} x \Leftrightarrow & (y(1) \wedge p(2,1)) \vee \dots \\ & \vee (y(n) \wedge p(2,n)) \end{aligned}$$

Compact representation of full CPT

$$w(p(1,1)) = 0.4$$

$$w(p(1,2)) = 0.3$$

...

$$w(p(1,n)) = 0.5$$

$$w(p(2,1)) = 0.8$$

...

$$w(p(2,n)) = 0.8$$

Example: Weighted Model Counting

$$y(1) \Leftrightarrow p(1,1)$$

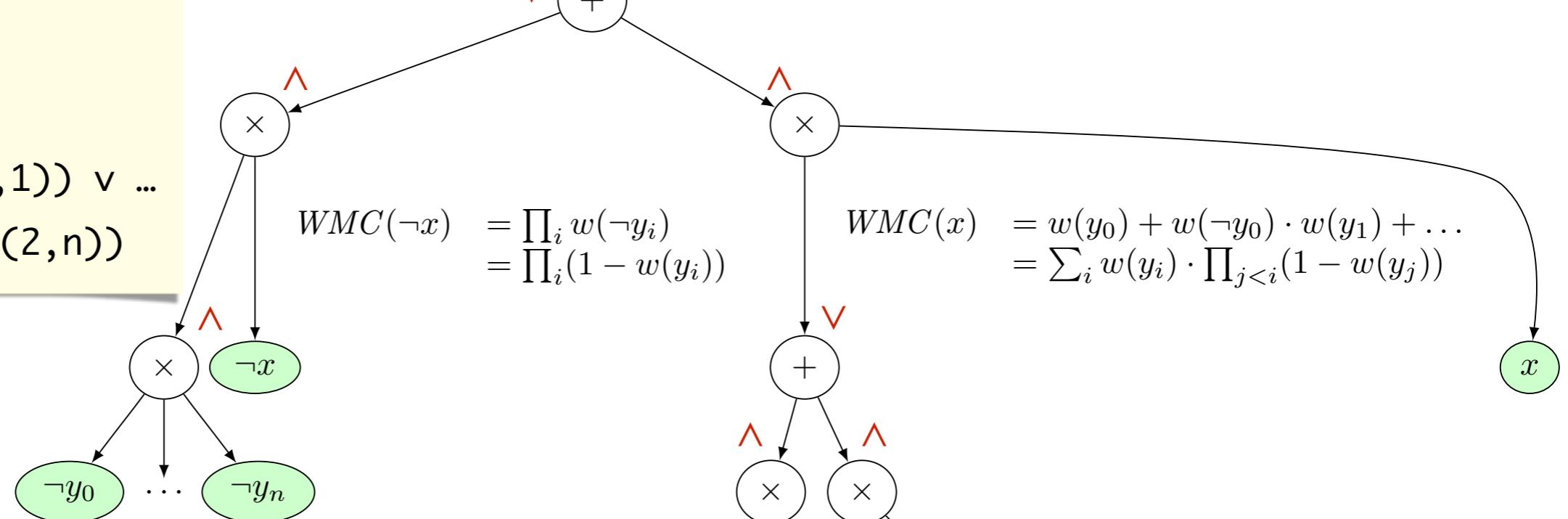
$$y(2) \Leftrightarrow p(1,2)$$

...

$$y(n) \Leftrightarrow p(1,n)$$

$$x \Leftrightarrow (y(1) \wedge p(2,1)) \vee \dots$$

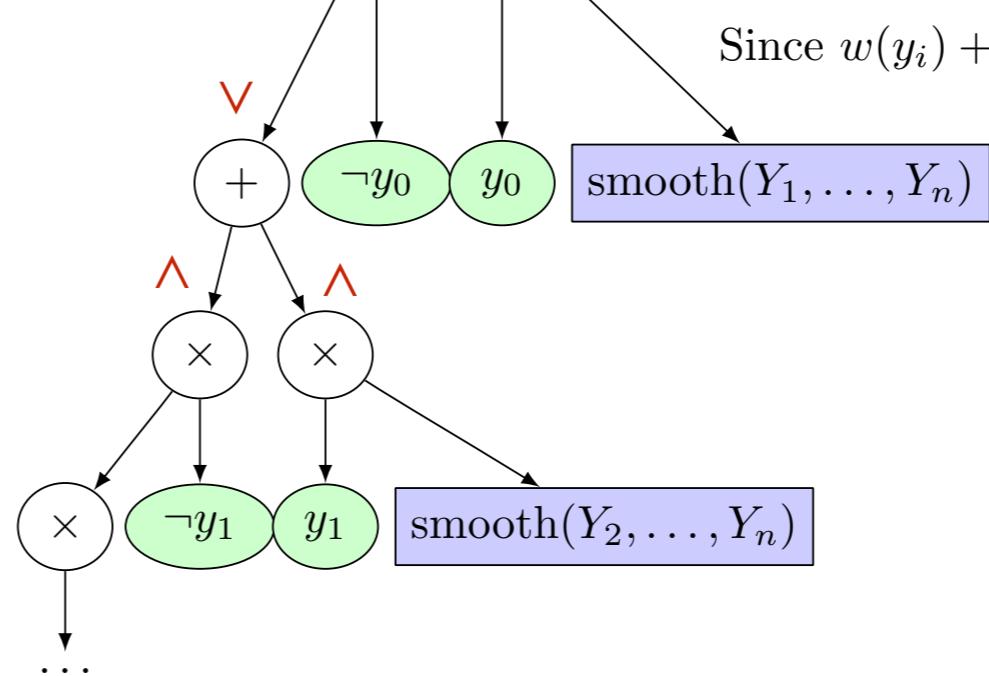
$$\vee (y(n) \wedge p(2,n))$$



$$WMC(\neg x) = \prod_i w(\neg y_i) = \prod_i (1 - w(y_i))$$

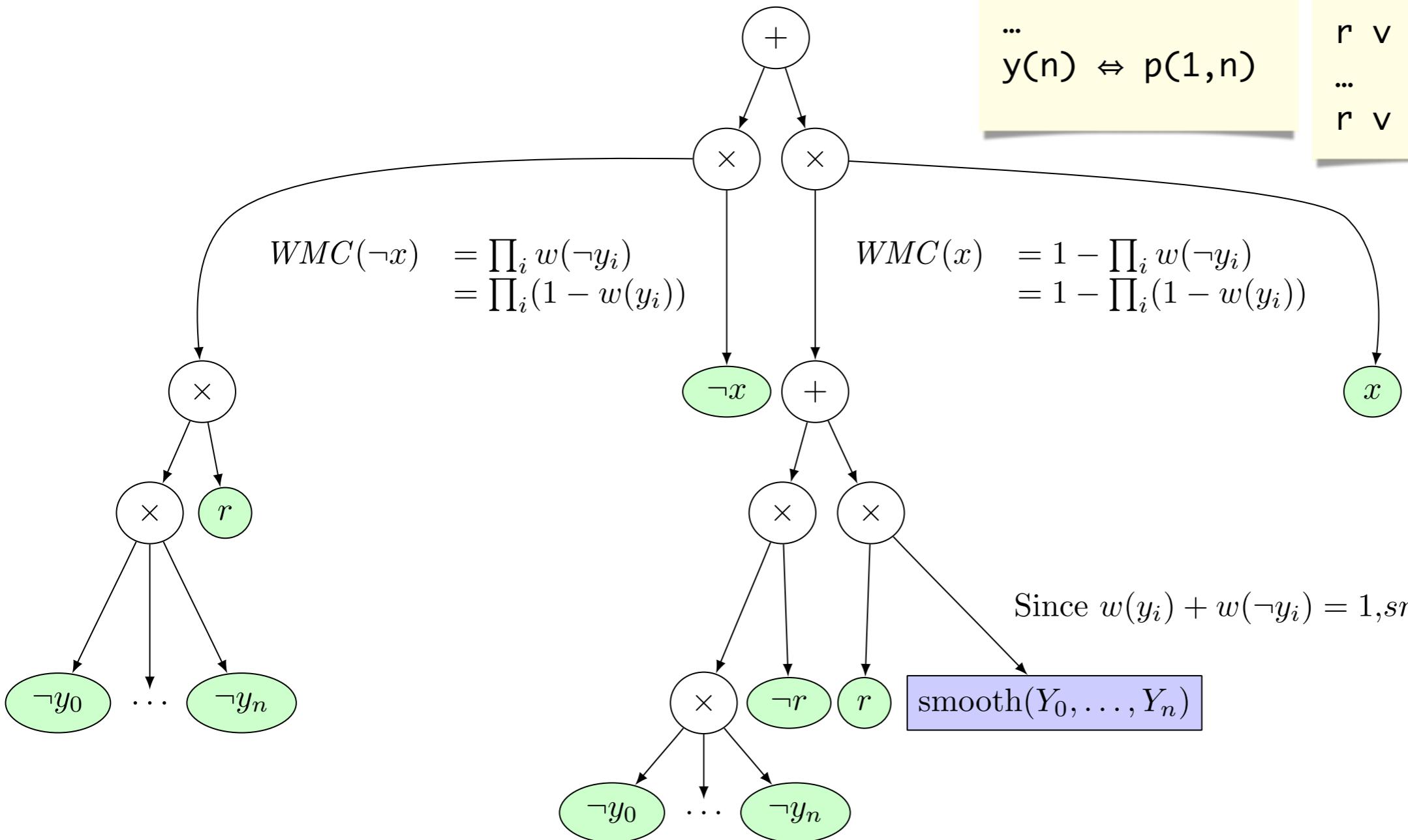
$$WMC(x) = w(y_0) + w(\neg y_0) \cdot w(y_1) + \dots = \sum_i w(y_i) \cdot \prod_{j < i} (1 - w(y_j))$$

An arithmetic circuit is the result of knowledge compilation. It is a (compact) mathematical formula to perform weighted counting.



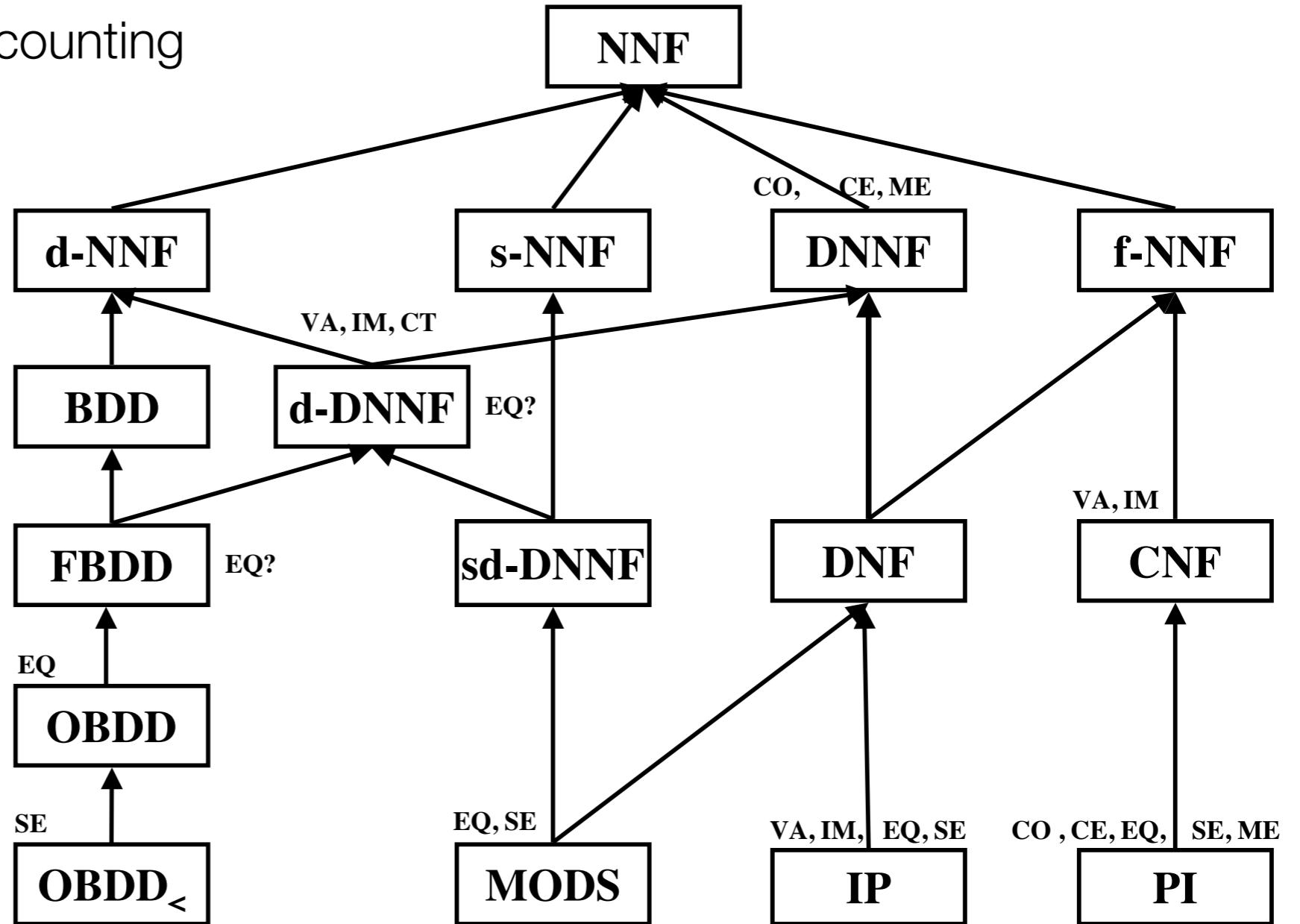
Example: Alternative Encoding

$x \Leftrightarrow z$
 $z \vee \neg y(1) \vee \neg p(2,1)$
 \dots
 $z \vee \neg y(n) \vee \neg p(2,n)$
 $r \vee z$
 $r \vee \neg y(1) \vee \neg p(2,1)$
 \dots
 $r \vee \neg y(n) \vee \neg p(2,n)$



Knowledge Compilation

Translate a propositional formula into a compact representation that allows for tractable weighted model counting



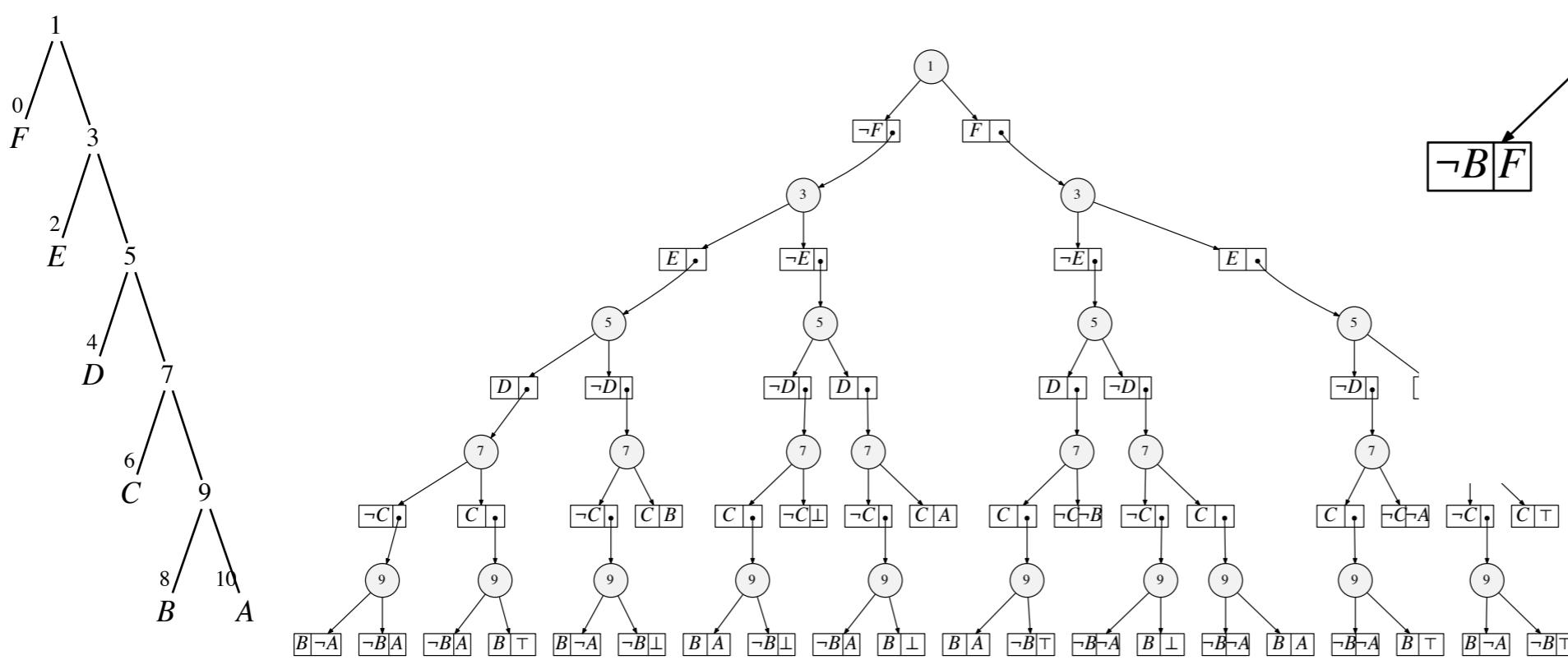
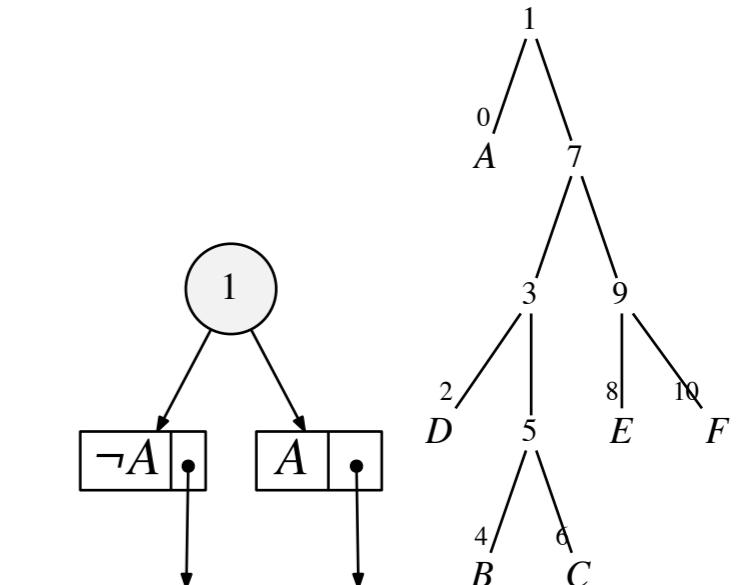
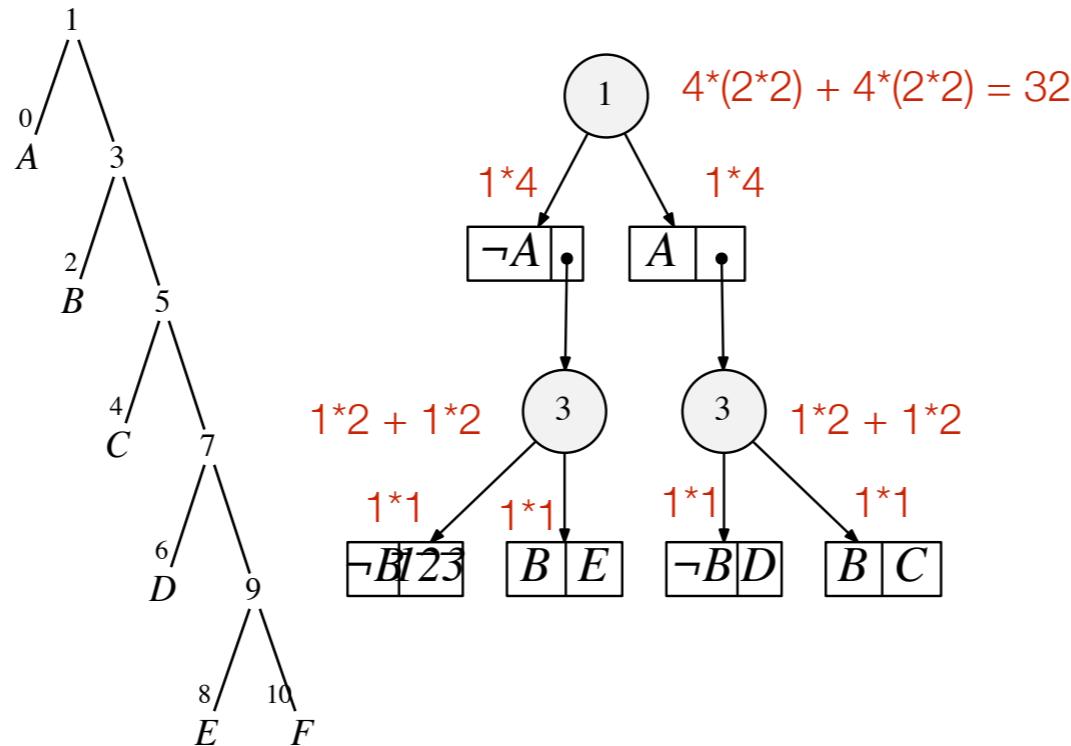
A. Darwiche and P. Marquis. A knowledge compilation map. Journal of Artificial Intelligence Research, 17:229–264, 2002.

Example: Effect of Ordering (vtree)

CNF

```

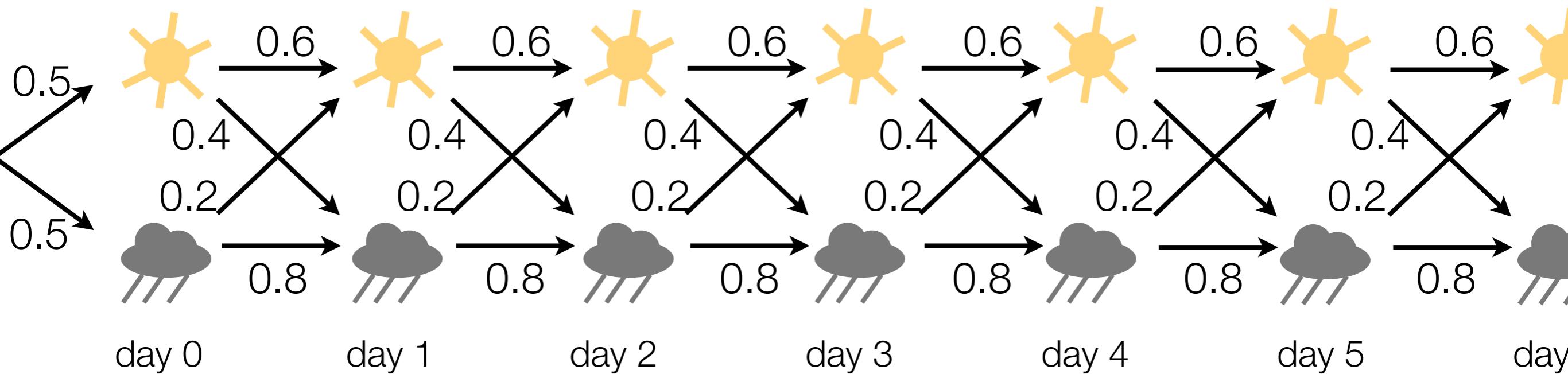
-A v -B v C
-A v B v D
A v -B v E
A v B v F
  
```



ProbLog Inference

- reduction to propositional formula
- addresses disjoint-sum-problem
- **but:** not all probabilistic logic programs face this problem! e.g., weather

ProbLog by example: Rain or sun?



```
0.5::weather(sun,0) ; 0.5::weather(rain,0) :- true.
```

```
0.6::weather(sun,T) ; 0.4::weather(rain,T)
    :- T>0, Tprev is T-1, weather(sun,Tprev).
```

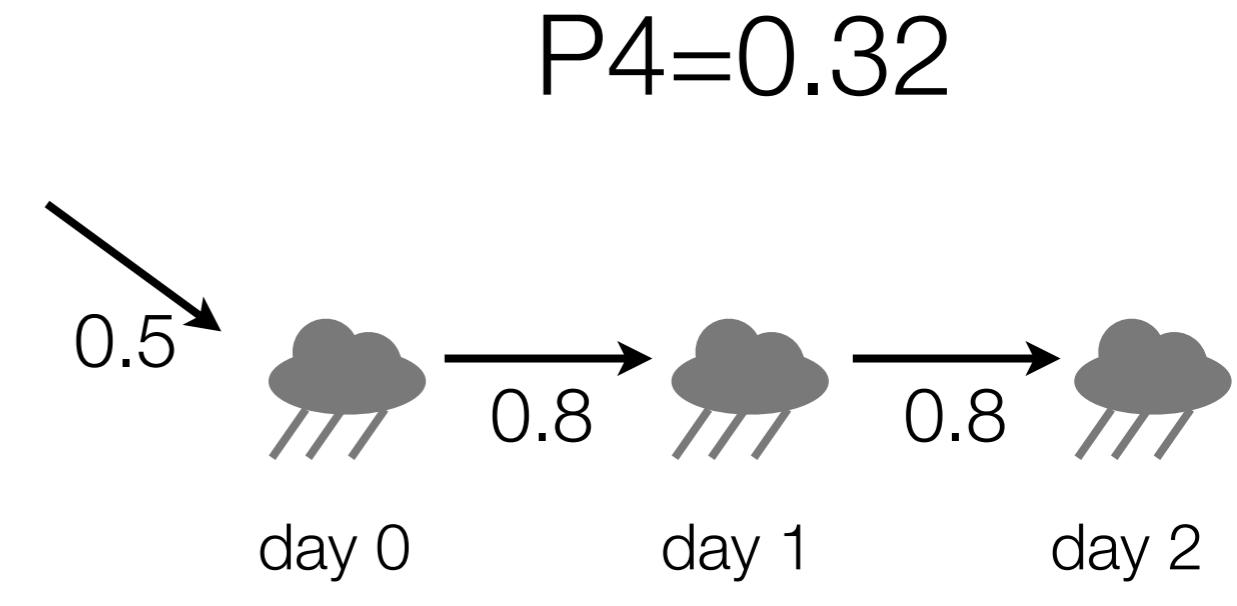
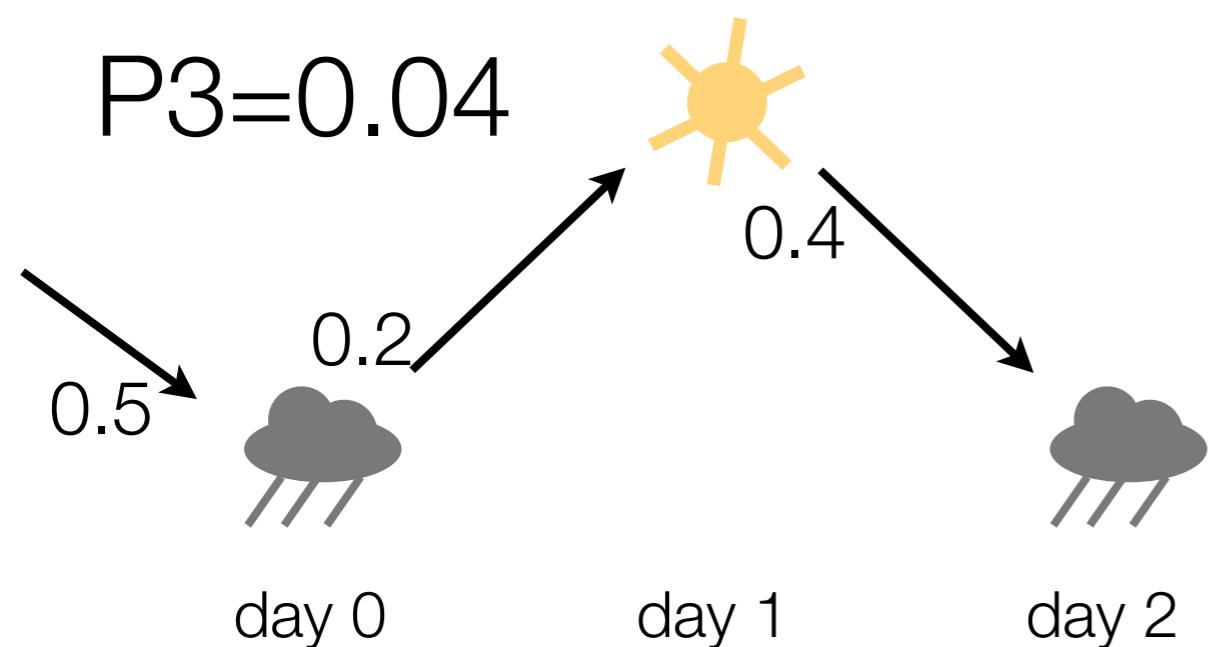
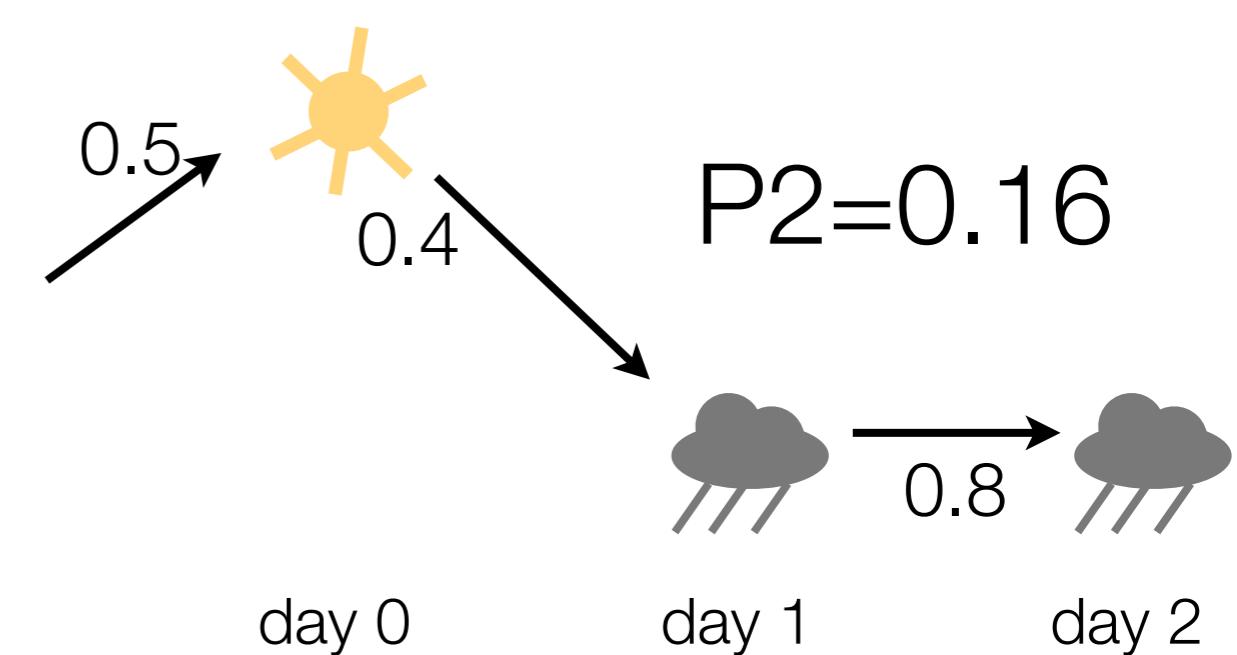
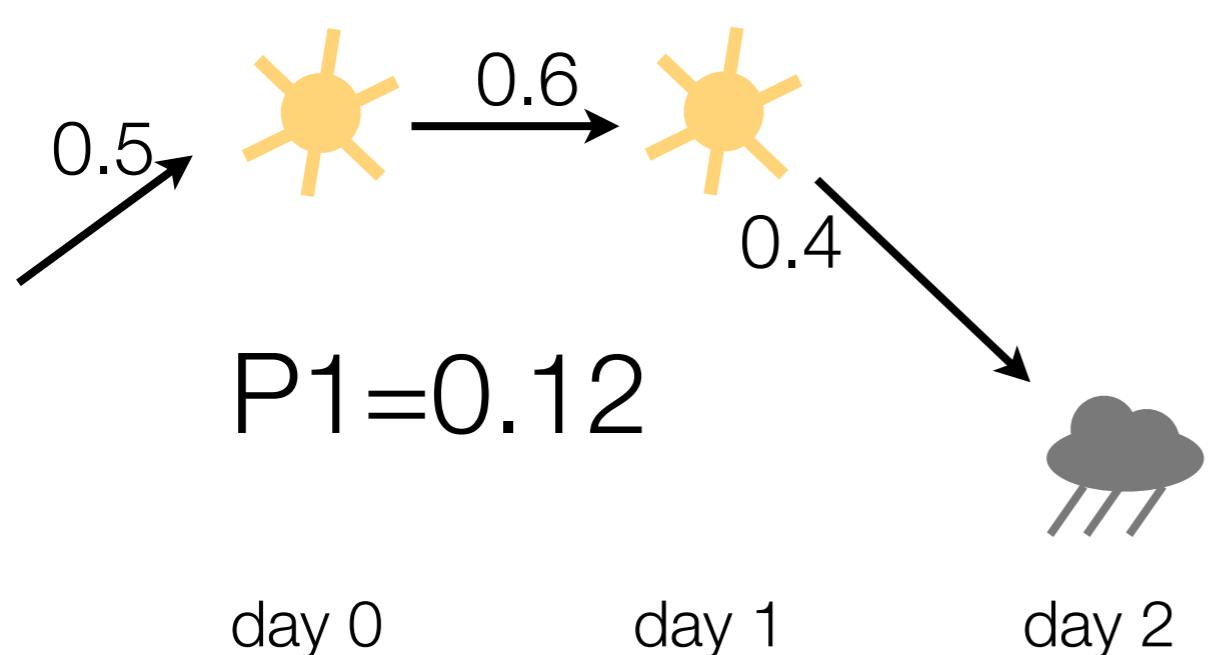
```
0.2::weather(sun,T) ; 0.8::weather(rain,T)
    :- T>0, Tprev is T-1, weather(rain,Tprev).
```

in every possible world, at most one proof for a ground query → summing proof probabilities is correct!

Possible worlds

?- `weather(rain,2)` .

$$P = P_1 + P_2 + P_3 + P_4$$



Mutually Exclusive Rules:

no two rules apply simultaneously

```
0.5::weather(sun,0) ; 0.5::weather(rain,0) :- true.
```

```
0.6::weather(sun,T) ; 0.4::weather(rain,T)
:- T>0, Tprev is T-1, weather(sun,Tprev).
```

```
0.2::weather(sun,T) ; 0.8::weather(rain,T)
:- T>0, Tprev is T-1, weather(rain,Tprev).
```

Mutually Exclusive Rules:

no two rules apply simultaneously

first rule for day 0, others for later days

```
0.5::weather(sun,0) ; 0.5::weather(rain,0) :- true.  
  
0.6::weather(sun,T) ; 0.4::weather(rain,T)  
:- T>0, Tprev is T-1, weather(sun,Tprev).  
  
0.2::weather(sun,T) ; 0.8::weather(rain,T)  
:- T>0, Tprev is T-1, weather(rain,Tprev).
```

Mutually Exclusive Rules:

no two rules apply simultaneously

first rule for day 0, others for later days

day 0: either sun or rain

```
0.5::weather(sun,0) ; 0.5::weather(rain,0) :- true.
```

```
0.6::weather(sun,T) ; 0.4::weather(rain,T)
:- T>0, Tprev is T-1, weather(sun,Tprev).
```

```
0.2::weather(sun,T) ; 0.8::weather(rain,T)
:- T>0, Tprev is T-1, weather(rain,Tprev).
```

Mutually Exclusive Rules:

no two rules apply simultaneously

first rule for day 0, others for later days

day 0: either sun or rain

```
0.5::weather(sun,0) ; 0.5::weather(rain,0) :- true.
```

```
0.6::weather(sun,T) ; 0.4::weather(rain,T)
:- T>0, Tprev is T-1, weather(sun,Tprev).
```

```
0.2::weather(sun,T) ; 0.8::weather(rain,T)
:- T>0, Tprev is T-1, weather(rain,Tprev).
```

rules for $T > 0$ cover mutually exclusive
cases on previous day

Learning

- Parameter learning
- Upgrading relational learning

Parameter Learning

e.g., webpage classification model

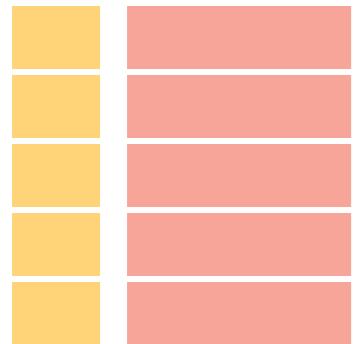
for each **CLASS1**, **CLASS2** and each **WORD**

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

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

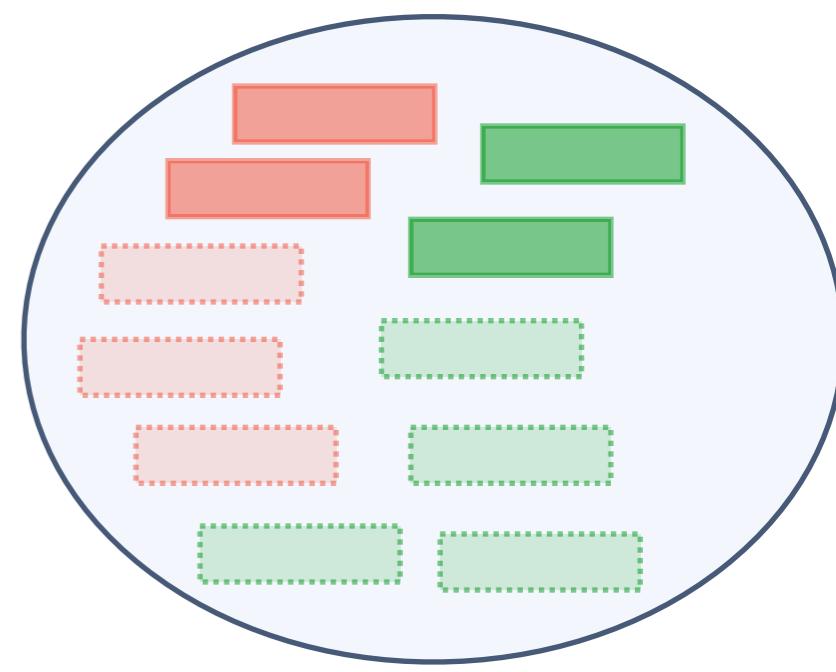
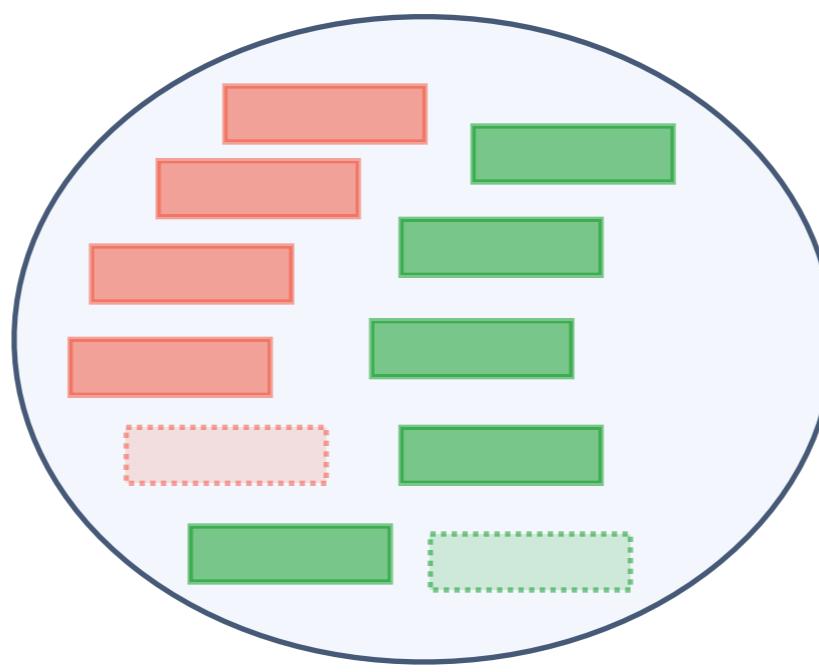
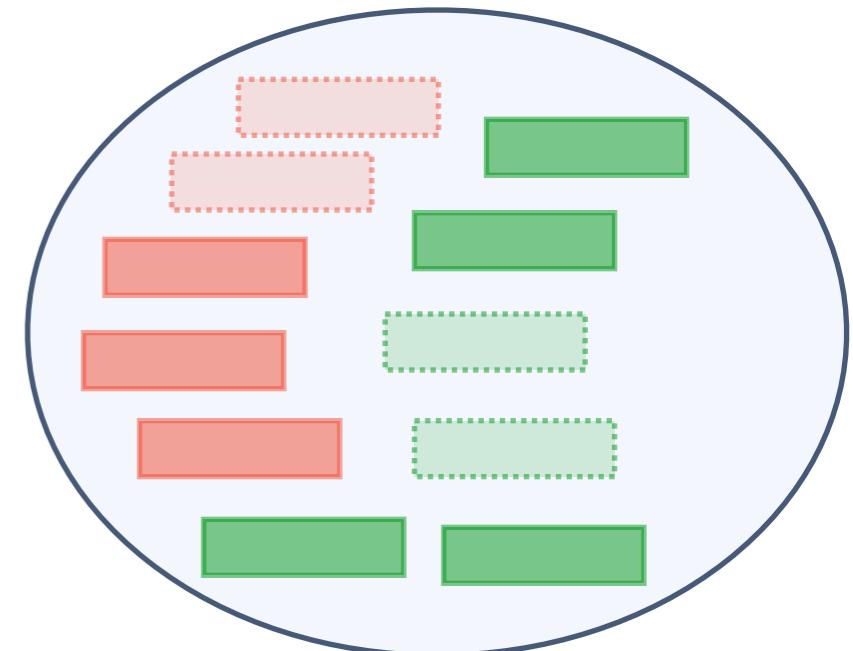
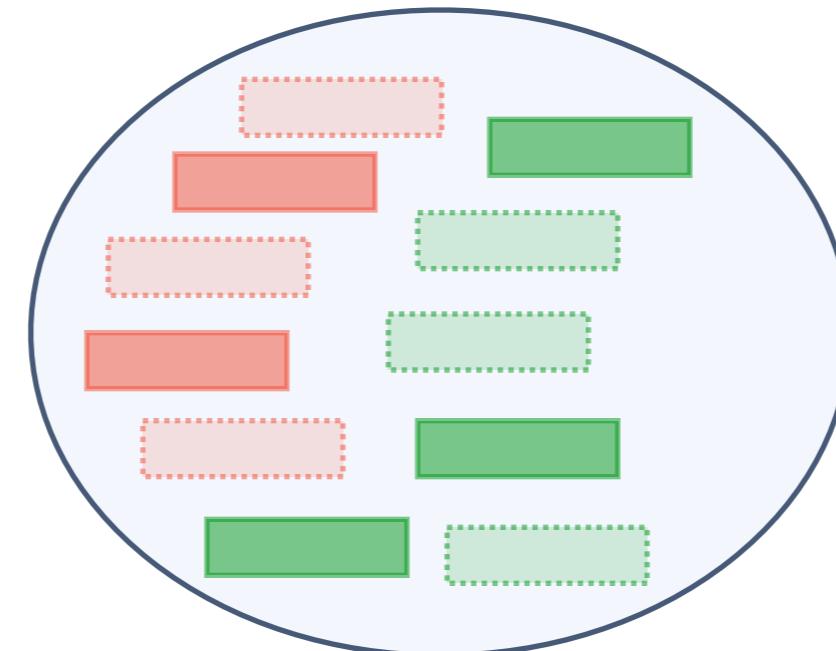
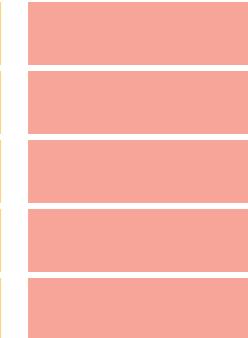
```
class(Page,C)      :- links_to(OtherPage,Page),  
class(OtherPage,OtherClass),  
link_class(OtherPage,Page,OtherClass,C).
```

Sampling Interpretations

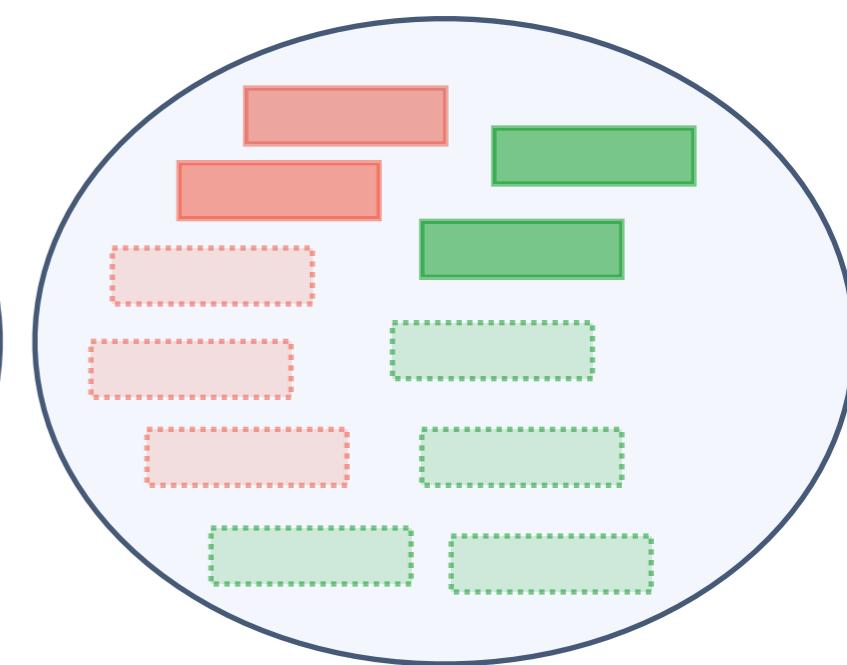
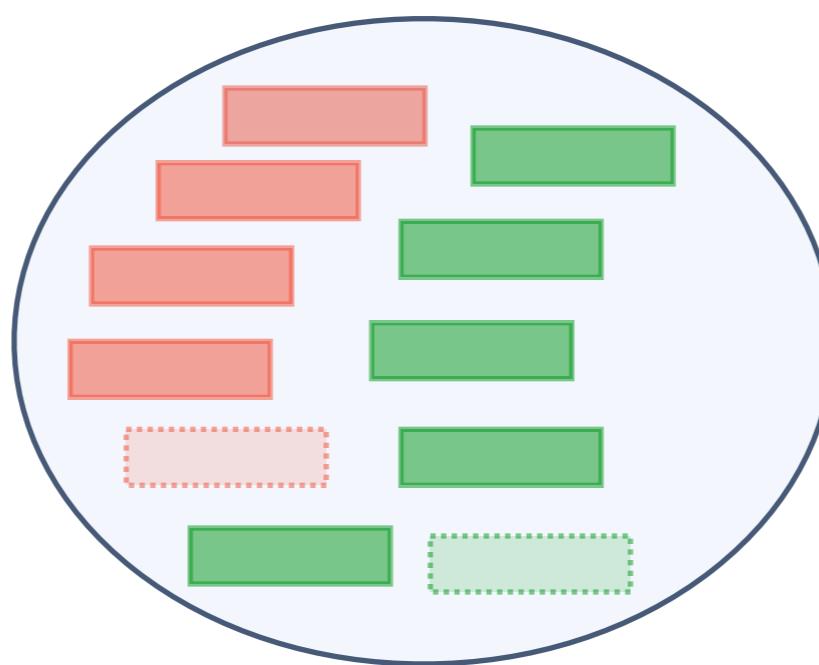
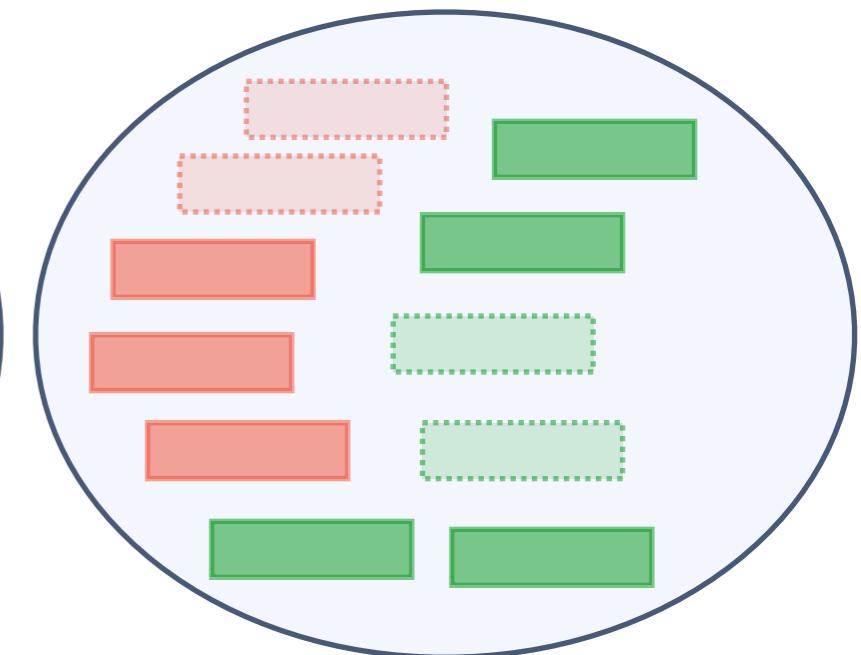
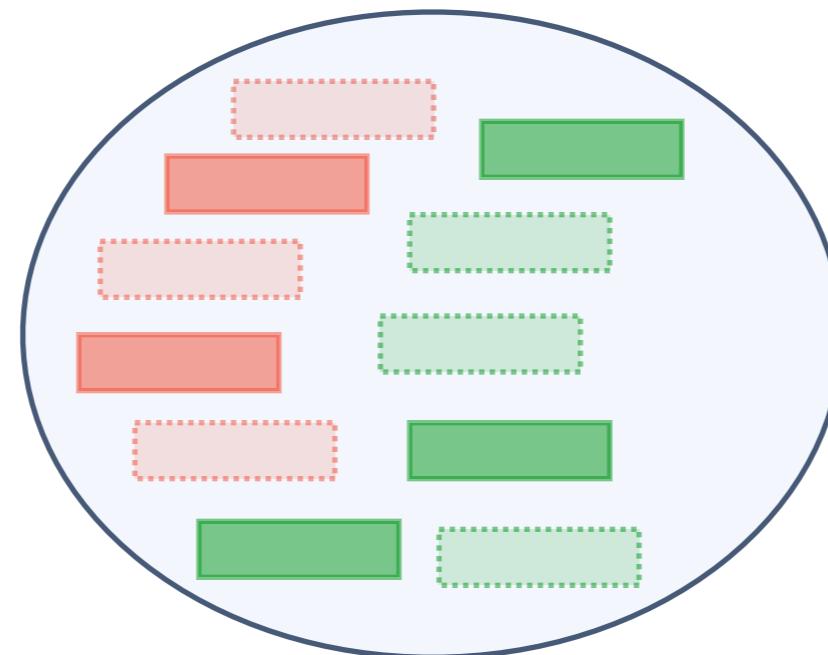
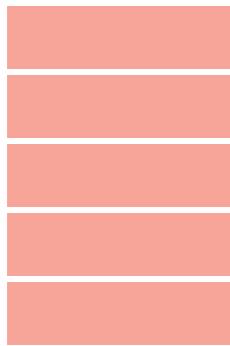


Sampling Interpretations

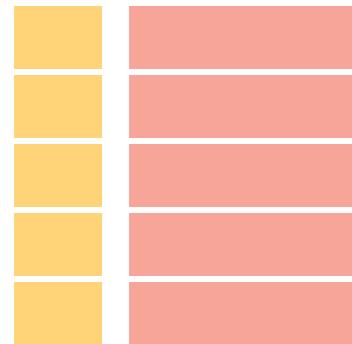
Yellow



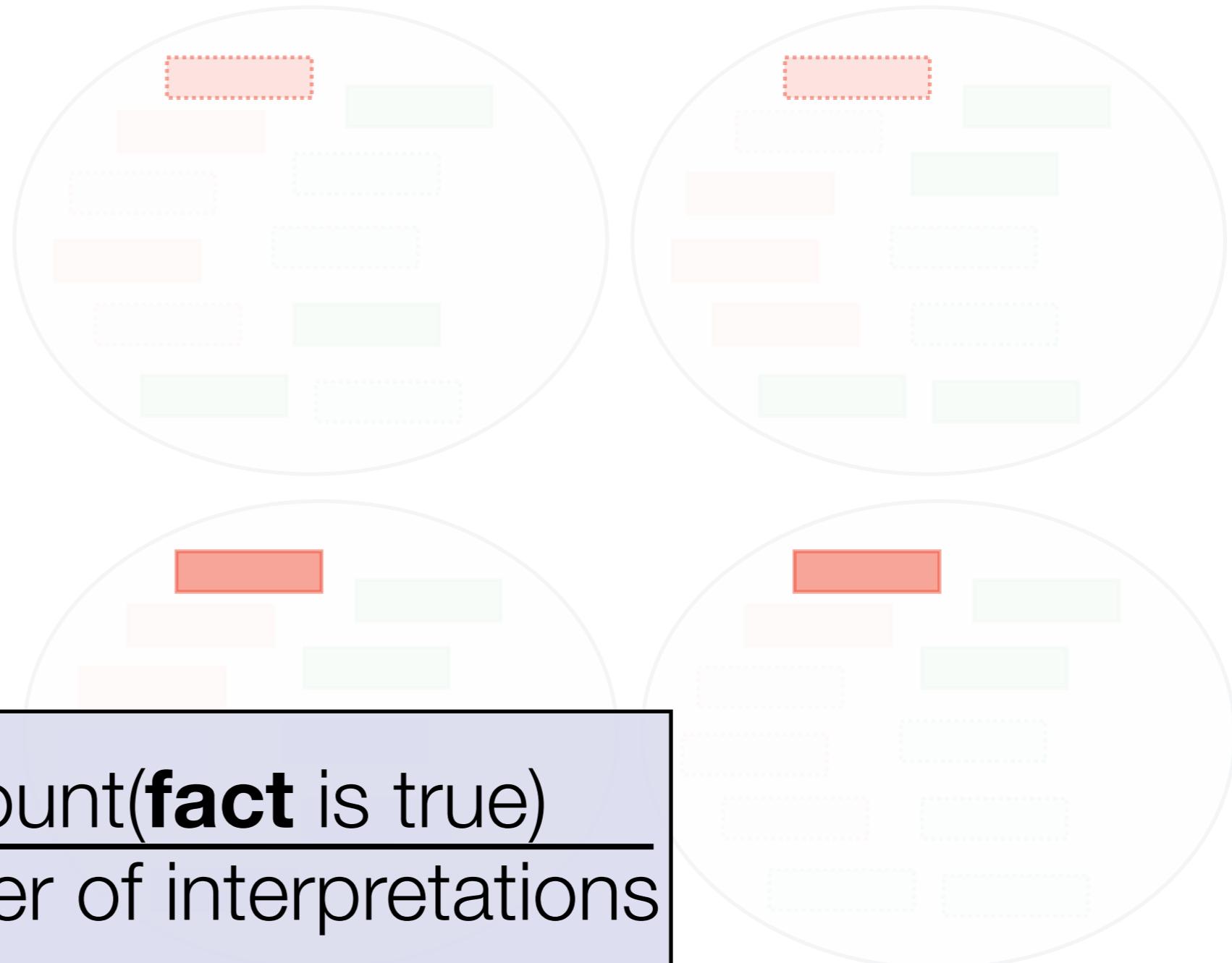
Parameter Estimation



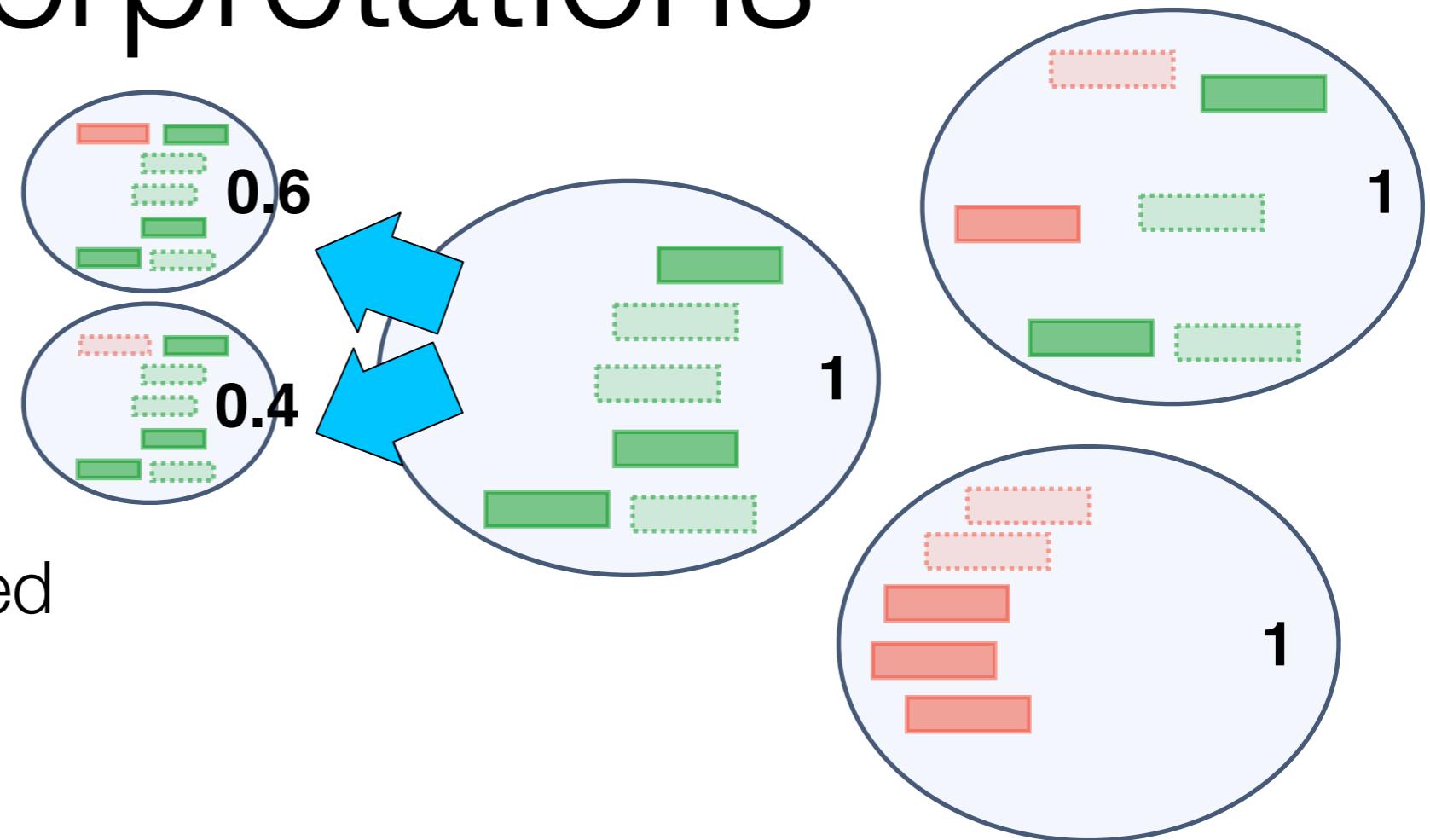
Parameter Estimation



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



Learning from partial interpretations



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

Upgrading relational learning

	Prolog	ProbLog
	infl(a,b)	0.4 :: infl(a,b)
Reasoning	query true? yes/no	query true? with probability P

Upgrading relational learning

	Prolog	ProbLog
Reasoning	query true? yes/no	query true? with probability P
Machine Learning	example covered? yes/no	example covered? with probability P

Structure Learning: ProbFOIL

Upgrading FOIL to learn a set of rules from **probabilistic** facts

```
0.7::father(piет, wim).  
0.9::father(bart, pieter).  
0.6::father(tom, greet).  
mother(emma,piет).  
mother(emma,greet).  
mother(greet,pieter).  
grandmother(emma,ilse).  
0.7::grandmother(emma,wim).  
....
```

Structure Learning: ProbFOIL

Upgrading FOIL to learn a set of rules from **probabilistic** facts

```
0.7::father(piет, wim).  
0.9::father(bart, pieter).  
0.6::father(tom, greet).  
mother(emma,piet).  
mother(emma,greet).  
mother(greet,pieter).  
grandmother(emma,ilse).  
0.7::grandmother(emma,wim).  
....
```

```
grandmother(X,Y) :-  
    mother(X,Z),  
    father(Z,Y).
```

Structure Learning: ProbFOIL

Upgrading FOIL to learn a set of rules from
probabilistic facts

```
0.4987::rain(1).  
0.3591::windok(1).  
0.4534::sunshine(1).  
0.3257::surfing(1).  
0.7391::rain(2).  
0.6022::windok(2).  
0.9837::sunshine(2).  
0.2592::surfing(2).  
0.2898::rain(3).  
0.7423::windok(3).  
0.2275::sunshine(3).  
0.5688::surfing(3)....
```

Structure Learning: ProbFOIL

Upgrading FOIL to learn a set of rules from
probabilistic facts

```
0.4987::rain(1).  
0.3591::windok(1).  
0.4534::sunshine(1).  
0.3257::surfing(1).  
0.7391::rain(2).  
0.6022::windok(2).  
0.9837::sunshine(2).  
0.2592::surfing(2).  
0.2898::rain(3).  
0.7423::windok(3).  
0.2275::sunshine(3).  
0.5688::surfing(3)....
```

```
surfing(X) :- \+ rain(X), windok(X).  
surfing(X) :- \+ rain(X), sunshine(X)
```

Prob2FOIL

- Learning **probabilistic rules** from probabilistic facts

```
0.4987::rain(1).  
0.3591::windok(1).  
0.4534::sunshine(1).  
0.3257::surfing(1).  
0.7391::rain(2).  
0.6022::windok(2).  
0.9837::sunshine(2).  
0.2592::surfing(2).  
0.2898::rain(3).  
0.7423::windok(3).  
0.2275::sunshine(3).  
0.5688::surfing(3).  
...
```

```
$ pip install probfoil
```

Prob2FOIL

- Learning **probabilistic rules** from probabilistic facts

```
0.4987::rain(1).  
0.3591::windok(1).  
0.4534::sunshine(1).  
0.3257::surfing(1).  
0.7391::rain(2).  
0.6022::windok(2).  
0.9837::sunshine(2).  
0.2592::surfing(2).  
0.2898::rain(3).  
0.7423::windok(3).  
0.2275::sunshine(3).  
0.5688::surfing(3).  
...  
  
0.7023::surfing(A) :- \+rain(A).  
0.01243::surfing(A) :- true.
```

```
$ pip install probfoil
```

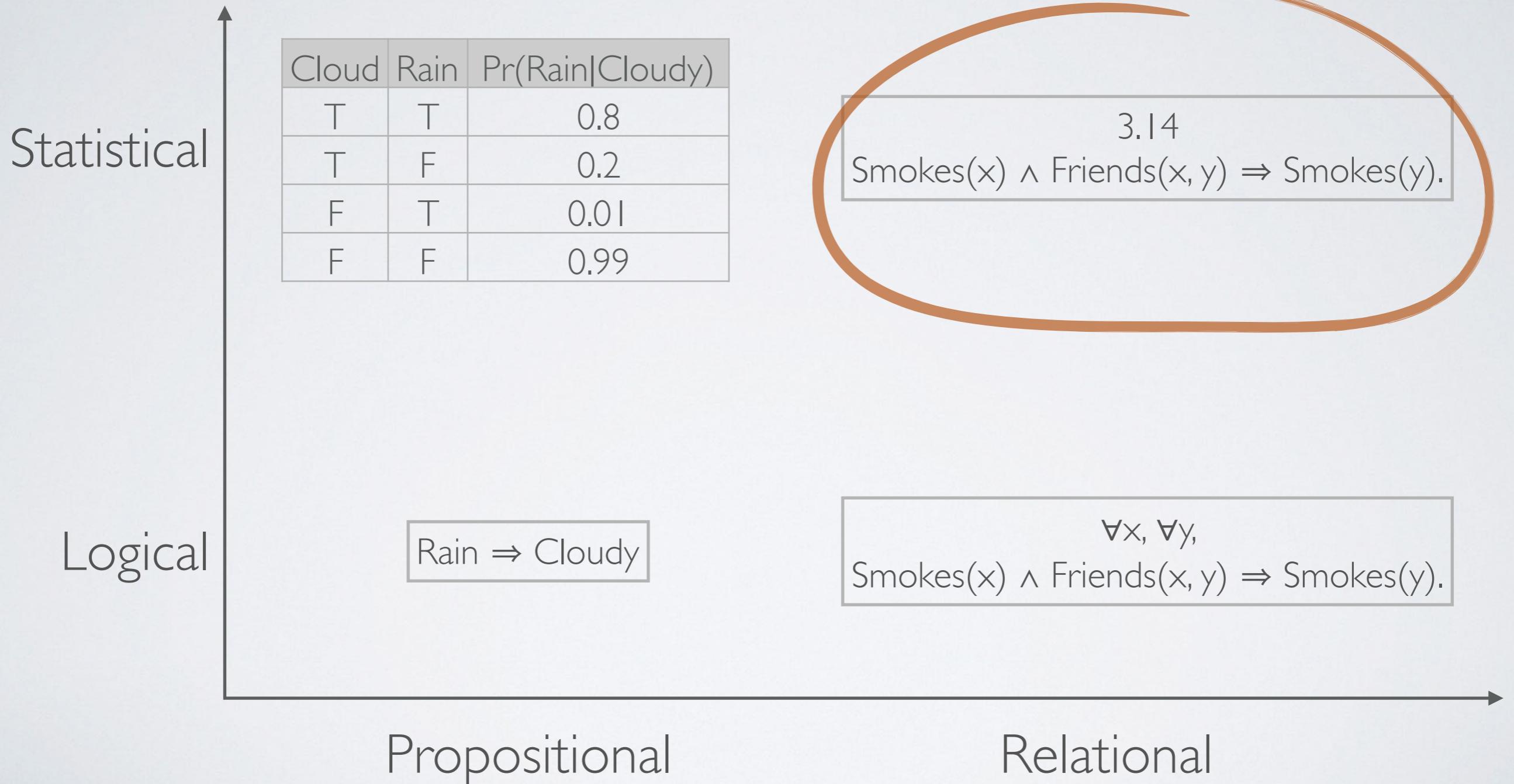
Lifted Inference

Just a taste

Lifted inference

- Exploiting symmetries & repeated structure
- Reasoning on first order level as much as possible
- Aiming at independence from number of objects
- Approximation: grouping similar computations
- Close ties with Probabilistic Databases
- Very active research area

STATISTICAL RELATIONAL MODELS



WHY WEIGHTED MODEL COUNTING?

Model Counting MC	Count the satisfying assignments of a propositional sentence
Weighted Model Counting WMC	Associate a weight with each assignment and compute sum of weights
Weighted First-Order Model Counting WFOMC	Exploit symmetries in a first-order sentence

Applications in:

- Probabilistic graphical models
- Statistical relational models
- Probabilistic logic models

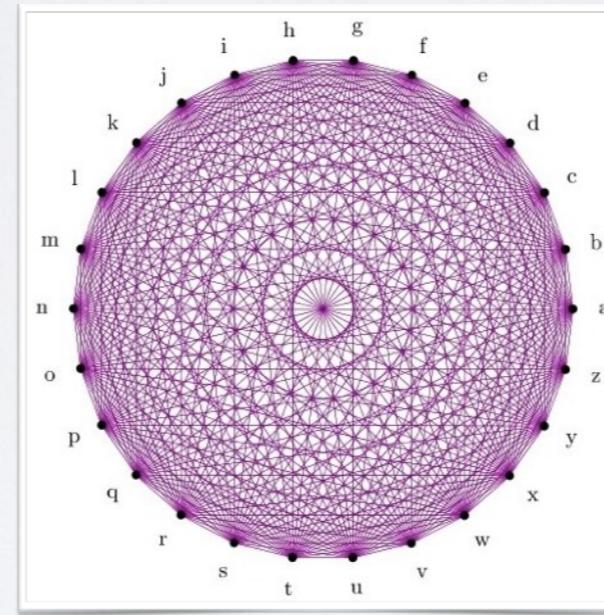
For example:

- Computing probabilities using lifted inference
- Learning the weights from data

WFOMC FOR MLNS

3.14 FacultyPage(x) \wedge Linked(x, y) \Rightarrow CoursePage(y).

As a probabilistic graphical model:
26 pages, 728 random variables, 676 factors



Real data set:
1000 pages, 1002000 random variables, 1000000 factors
→ Highly intractable probability computation?
Lifted inference (WFOMC) in milliseconds!

WEIGHTED MODEL COUNTING

A logical theory and a weight function

Stress \Rightarrow Smokes.

literal \rightarrow weight

Stress \rightarrow 4

\neg Stress \rightarrow 1

Smokes \rightarrow 1

\neg Smokes \rightarrow 2

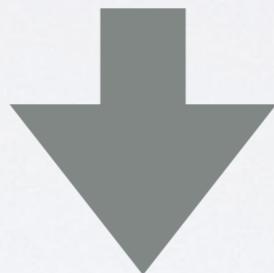
Stress	Smokes	Δ	WM
4	1		4
4	2		8
1	1		1
1	2		2
		\sum 3 models	7

MC = 3

WMC = 7

WEIGHTED FIRST-ORDER MODEL COUNTING

$$WMC(\Delta, w) = \sum_{\omega \models \Delta} \prod_{l \in \omega} w(l)$$



$$\text{WFOMC}(\Delta, \mathbf{D}, w, \bar{w}) = \sum_{\omega \models_{\mathbf{D}} \Delta} \frac{\prod_{l \in \omega_0} \bar{w}(\text{pred}(l))}{\prod_{l \in \omega_1} w(\text{pred}(l))}$$

False literals True literals

WEIGHTED FIRST-ORDER MODEL COUNTING

A first-order logical theory and two weight functions

Stress(Alice) \Rightarrow Smokes(Alice).

Stress	Smokes	Δ	WM
4	1		4
4	2		8
1	1		1
1	2		2
		Σ	3 models
			7

$\text{pred} \rightarrow \text{weight}$

w

Stress \rightarrow 4

Smokes \rightarrow 1

\bar{w}

Stress \rightarrow 1

Smokes \rightarrow 2

$MC = 3$

$WMC = 7$

WEIGHTED FIRST-ORDER MODEL COUNTING

Stress(A) \Rightarrow Smokes(A).

Stress(B) \Rightarrow Smokes(B).

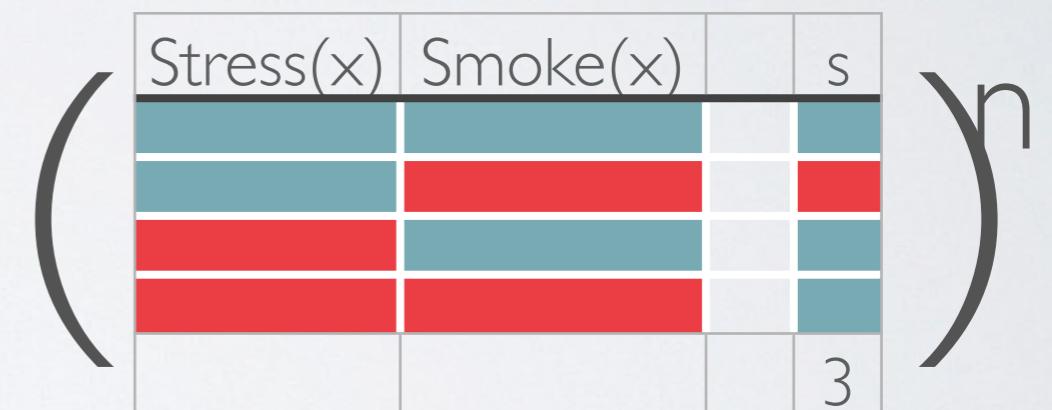
The figure consists of a 10x10 grid of colored bars. The columns are labeled at the top as Stress(A), Smokes(A), Stress(B), Smokes(B), and S. The rows are numbered 1 through 10 on the left. Each cell contains a bar whose color depends on its value. A yellow double-line border highlights specific cells in the Stress(B) and Smokes(B) columns across all rows. The colors used are teal, red, light blue, and yellow.

	Stress(A)	Smokes(A)	Stress(B)	Smokes(B)	S
1	Teal	Teal	Yellow	Red	Teal
2	Teal	Teal	Red	Red	Teal
3	Teal	Teal	Red	Red	Teal
4	Red	Red	Yellow	Red	Red
5	Red	Red	Red	Red	Red
6	Red	Red	Yellow	Red	Red
7	Red	Red	Red	Red	Teal
8	Red	Red	Yellow	Red	Red
9	Red	Red	Red	Red	Teal
10	Red	Red	Yellow	Red	Teal

$$w(\cdot) = |$$

$\forall x, \text{Stress}(x) \Rightarrow \text{Smokes}(x).$

$$\mathbf{D} = \{\mathbf{A}, \mathbf{B}\}$$
$$|\mathbf{D}| = n = 2$$



Expression independent of domain size

$$WMC = 9$$

$$WFOMC = 3^n = 9$$

WFOMC

Domain: n people

$$\forall y, \text{ParentOf}(y) \wedge \text{Female} \Rightarrow \text{MotherOf}(y).$$

if Female: $\forall y, \text{ParentOf}(y) \Rightarrow \text{MotherOf}(y)$. WFOMC = 3^n

if \neg Female: $\forall y, \text{true}$. WFOMC = 4^n

$$\text{WFOMC} = 3^n + 4^n$$

WFOMC

Domain: n people

$$\forall x, \forall y, \text{ParentOf}(x, y) \wedge \text{Female}(x) \Rightarrow \text{MotherOf}(x, y).$$

if x known: $\forall y, \text{ParentOf}(A, y) \wedge \text{Female}(A) \Rightarrow \text{MotherOf}(A, y).$

$$\text{WFOMC} = 3^n + 4^n$$

$$\text{WFOMC} = (3^n + 4^n)^n$$

WFOMC

Domain: n people

$$\forall x, \forall y, \text{Smokes}(x) \wedge \text{Friends}(x, y) \Rightarrow \text{Smokes}(y).$$

We know the k smokers:
4 disjunct groups

If $\text{Smokes}(A) = \text{true} \wedge \text{Smokes}(B) = \text{true}$
 $\text{true} \wedge \text{Friends}(A, B) \Rightarrow \text{true}.$

If $\text{Smokes}(A) = \text{true} \wedge \text{Smokes}(C) = \text{false}$
 $\text{true} \wedge \text{Friends}(A, C) \Rightarrow \text{false.}$ ↗ k(n-k)

If $\text{Smokes}(D) = \text{false} \wedge \text{Smokes}(B) = \text{true}$
 $\text{false} \wedge \text{Friends}(D, B) \Rightarrow \text{true.}$

If $\text{Smokes}(D) = \text{false} \wedge \text{Smokes}(C) = \text{false}$
 $\text{false} \wedge \text{Friends}(D, C) \Rightarrow \text{false.}$

$2^{n^2 - k(n-k)}$ models



We know there are k smokers:

$$\binom{n}{k} 2^{n^2 - k(n-k)} \text{ models}$$

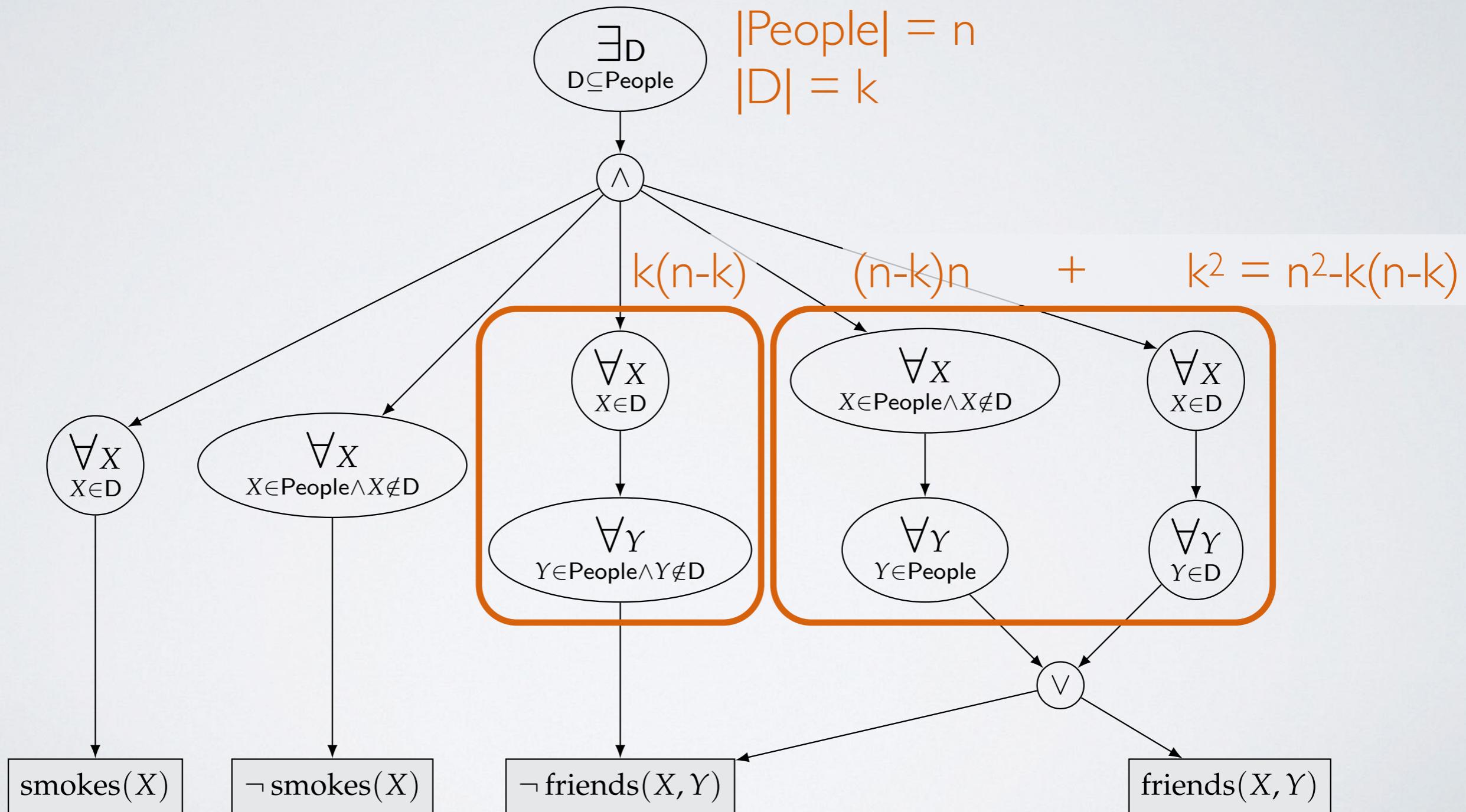


WFOMC:

$$\sum_{k=0}^n \binom{n}{k} 2^{n^2 - k(n-k)}$$

WFOMC WITH FO-D-DNNF

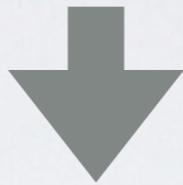
$$\forall x, \forall y, \text{Smokes}(x) \wedge \text{Friends}(x, y) \Rightarrow \text{Smokes}(y).$$



PROBABILISTIC LOGIC PROGRAM

0.1::Attends(x).
0.3::Series :- Attends(x).

~Noisy-OR



Series $\Leftrightarrow \exists x, \text{Attends}(x) \wedge \text{ToSeries}(x)$.

$$\begin{array}{ll} w(\text{Attends}(\cdot)) = 0.1 & w(\text{ToSeries}(\cdot)) = 0.3 \\ w(\neg \text{Attends}(\cdot)) = 0.9 & w(\neg \text{ToSeries}(\cdot)) = 0.7 \\ w(\cdot) = 1 & \end{array}$$

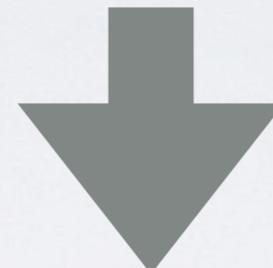


Weighted (First-Order) Model Counting

$\Pr(\text{Series} \mid \text{Attends}(A))$

PROBABILISTIC LOGIC PROGRAM

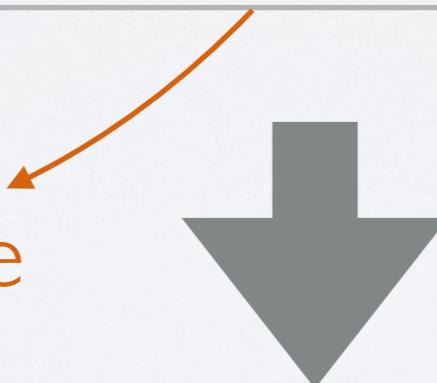
Series $\Leftrightarrow \exists x, \text{Attends}(x) \wedge \text{ToSeries}(x).$



Domain = {A,B}

Series $\Leftrightarrow [(\text{Attends}(A) \wedge \text{ToSeries}(A)) \vee (\text{Attends}(B) \wedge \text{ToSeries}(B))].$

Lost first-order structure

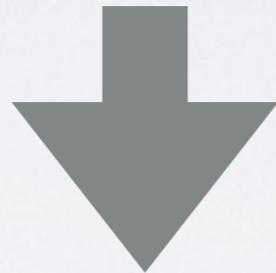


CNF, d-DNNF, AC, ...
Weighted Model Counting

$\Pr(\text{Series} \mid \text{Attends}(A))$

PROBABILISTIC LOGIC PROGRAM

```
0.1::Attends(x).  
0.3::Series :- Attends(x).
```



```
Series  $\Leftrightarrow \exists x, \text{Attends}(x) \wedge \text{ToSeries}(x).$ 
```

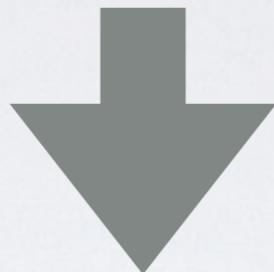
$$w(\text{Attends}) = 0.1 \quad w(\text{ToSeries}) = 0.3$$

$$\bar{w}(\text{Attends}) = 0.9 \quad \bar{w}(\text{ToSeries}) = 0.7$$

$$w(\cdot) = | \quad \bar{w}(\cdot) = |$$

PROBABILISTIC LOGIC PROGRAM

Series $\Leftrightarrow \exists x, \text{Attends}(x) \wedge \text{ToSeries}(x)$.



Series $\Leftrightarrow Z$

$\forall x, Z \vee \neg[\text{Attends}(x) \wedge \text{ToSeries}(x)]$

$Z \vee S$

$\forall x, S \vee \neg[\text{Attends}(x) \wedge \text{ToSeries}(x)]$

$w(\text{Attends}) = 0.1 \quad w(\text{ToSeries}) = 0.3$

$\bar{w}(\text{Attends}) = 0.9 \quad \bar{w}(\text{ToSeries}) = 0.7$

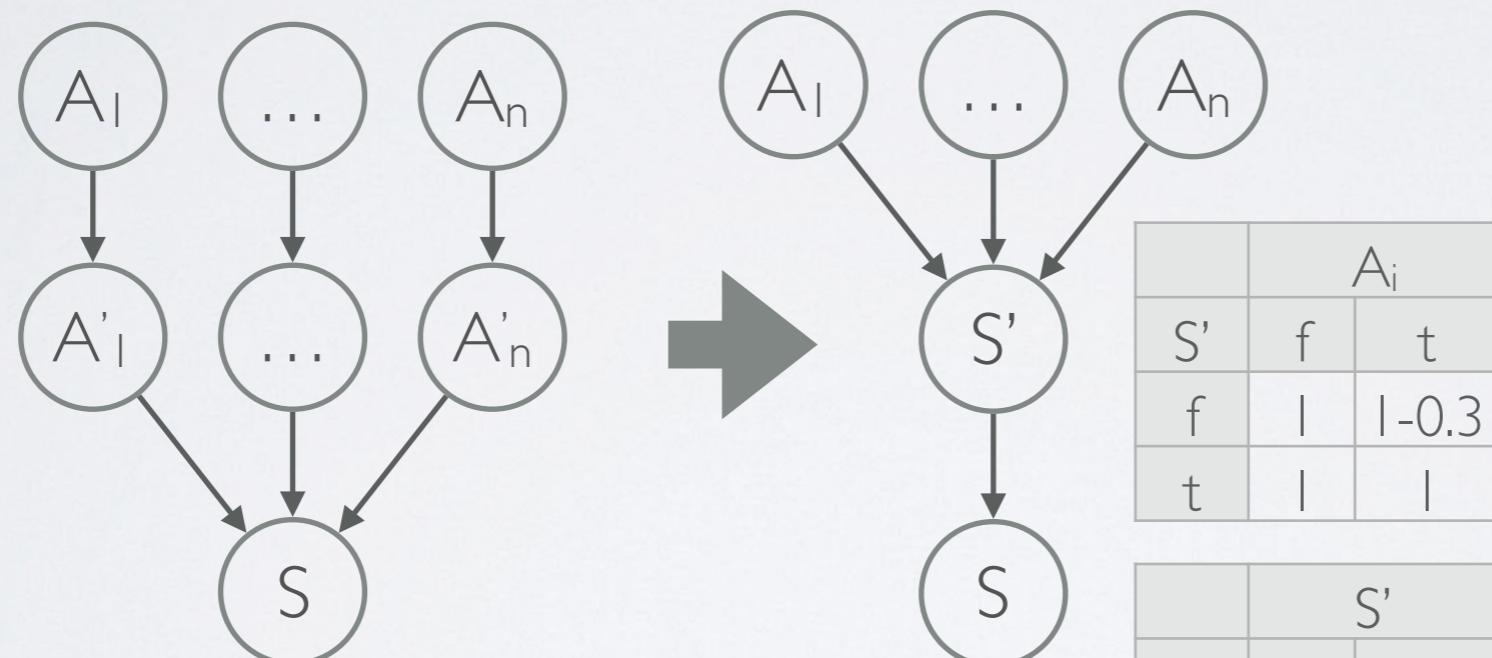
$\bar{w}(S) = -1$

$w(\cdot) = | \quad \bar{w}(\cdot) = |$

NOISY-OR

Multiplicative factorisation (or tensor rank-one decomposition) for noisy-OR is a special case of the Skolemization procedure

0.1::Attends(x).
0.3::Series :- Attends(x).



$$S = T \Leftrightarrow \exists A'_i, A'_i = T$$

S is false if all A_i are inhibited

→ only I model

Relax for this model and then compensate

→ C-FOVE operator
[Kisynski et al, IJCAI 2011]

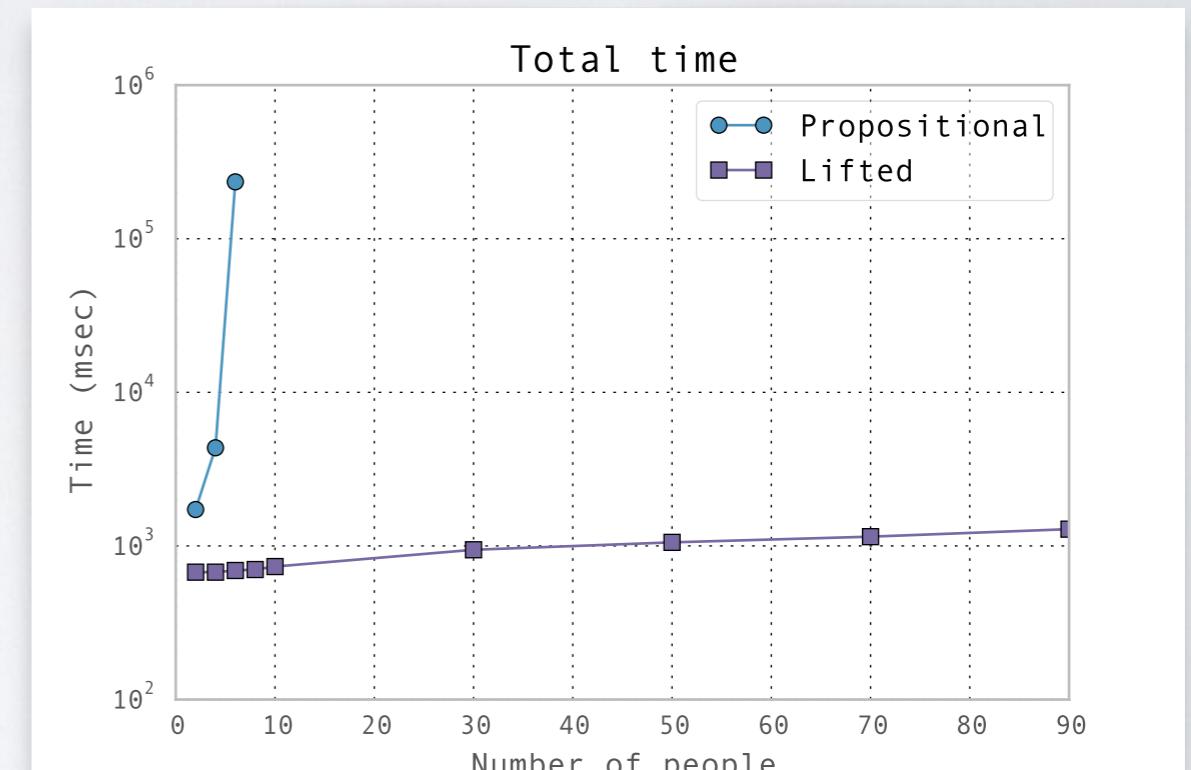
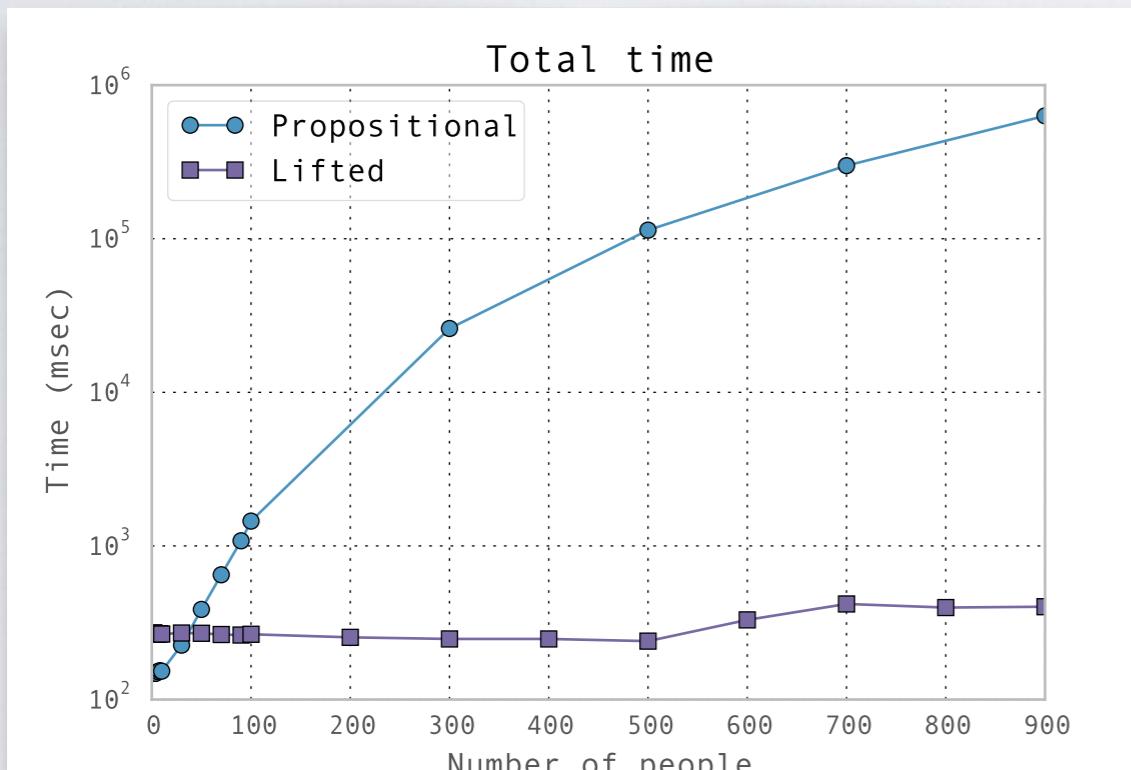
→ GC-FOVE operator
[Bellodi et al, ICLP 2014]

[Diez et al, 2003]
[Takikawa et al, 1999]

EXPERIMENTS

0.1 :: Attends(x).

0.3 :: Series :- Attends(x).



0.1 :: Attends(x).

0.3 :: Series :- Attends(x), Coauthor(x, y), Attends(y).



<http://dtai.cs.kuleuven.be/wfomc>

<https://github.com/UCLA-StarAI/Forclift>

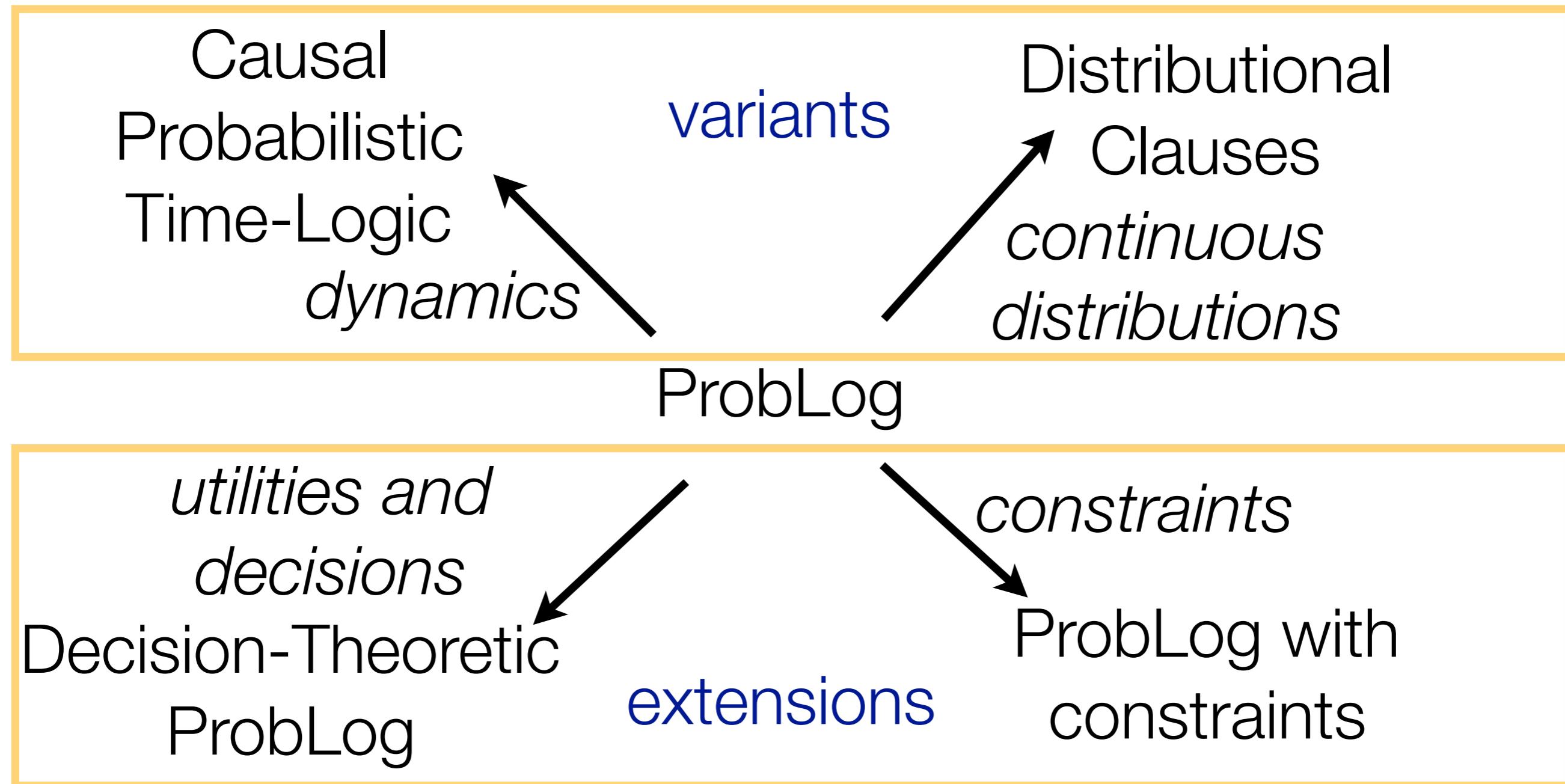


Language Extensions and Variants

- Dynamics under uncertainty
- Continuous-valued random variables
- Decision making
- Constraint

Language Extensions and Variants

not included
in ProbLog
system



Dynamic ProbLog

Efficient Probabilistic Inference for Dynamic Relational Models

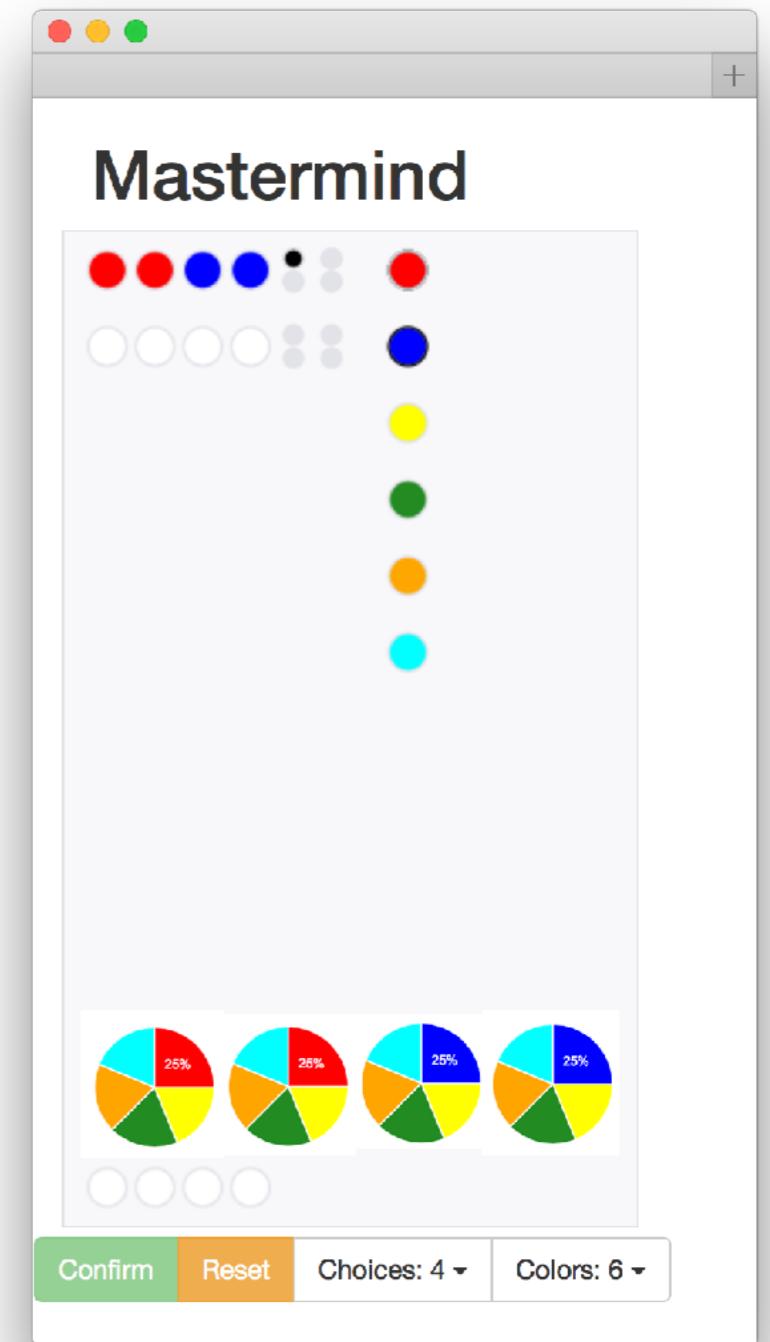
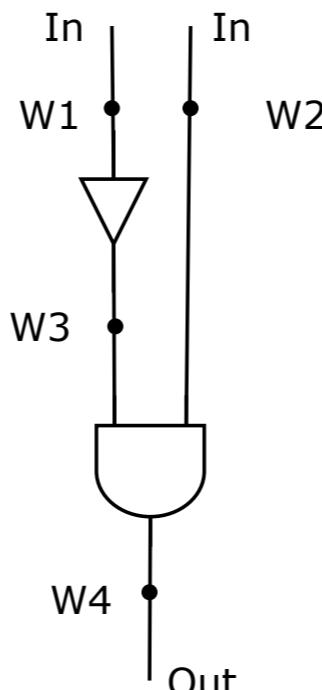
Jonas Vlasselaer, Wannes Meert, Guy Van den Broeck and Luc De Raedt

```
wire(1). wire(2).
wire(3). wire(4).
in(1). in(2). out(4).
gate(a,not,[1],3).
gate(b, and, [3,2],4).

0.990::healthy(G,0) :- gate(G,_,_,_).
0.990::healthy(G,T) :- T>0, healthy(G,T-1).
0.001::healthy(G,T) :- T>0, \+healthy(G,T-1).

0.5::high(W,T) :- in(W).
0.5::high(W,T) :- gate(G,_,_,W), \+healthy(G,T).
high(W,T) :- gate(G,not,[I],W), healthy(G,T),
             \+high(I,T).
high(W,T) :- gate(G, and, [I,J],W), healthy(G,T),
             high(I,T), high(J,T).

input(W,T) :- in(W), high(W,T).
output(W,T) :- out(W), high(W,T).
```



Distributional Clauses (DC)

- Discrete- and continuous-valued random variables
- Inference: particle filter

Distributional Clauses (DC)

- Discrete- and continuous-valued random variables
- Inference: particle filter

random variable with Gaussian distribution

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



Distributional Clauses (DC)

- Discrete- and continuous-valued random variables
- Inference: particle filter

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

**comparing values of
random variables**



Distributional Clauses (DC)

- Discrete- and continuous-valued random variables
- Inference: particle filter

```
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**



Distributional Clauses (DC)

- Discrete- and continuous-valued random variables
- Inference: particle filter

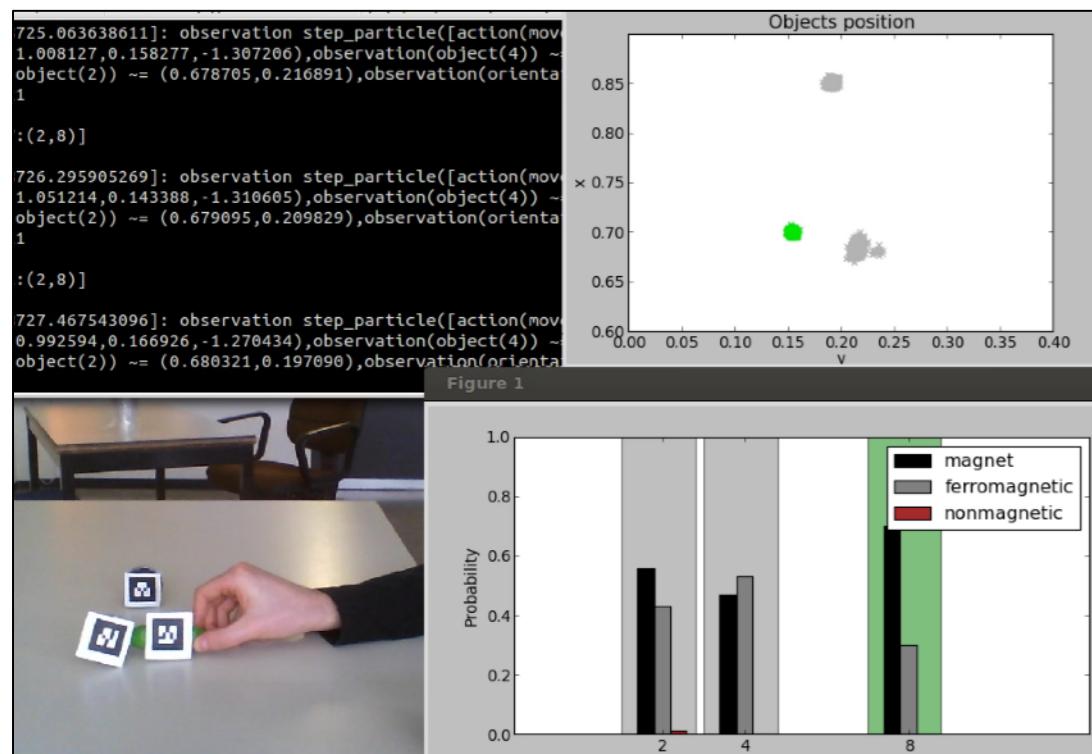
```
length(Obj) ~ gaussian(6.0,0.45) :- type(Obj,glass) .  
stackable(OBot,OTop) :-  
    slength(OBot) ≥ slength(OTop) ,  
    swidth(OBot) ≥ swidth(OTop) .  
ontype(Obj,plate) ~ finite([0 : glass, 0.0024 : cup,  
                            0 : pitcher, 0.8676 : plate,  
                          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



Relational State Estimation over Time

Magnetism scenario

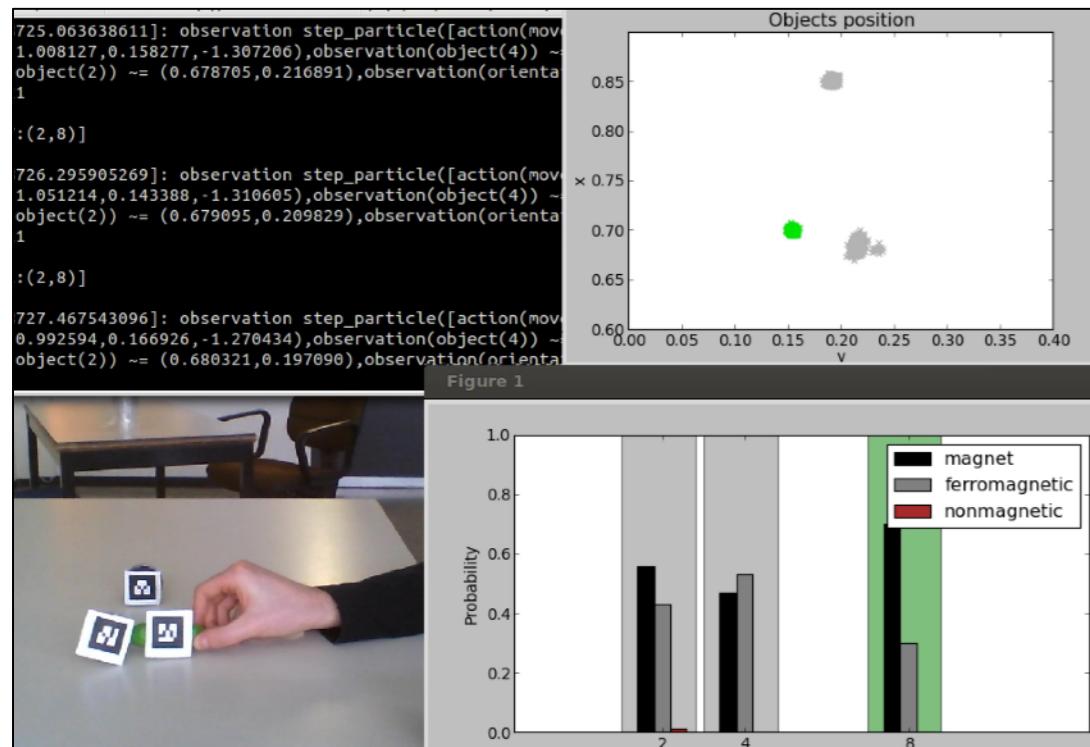
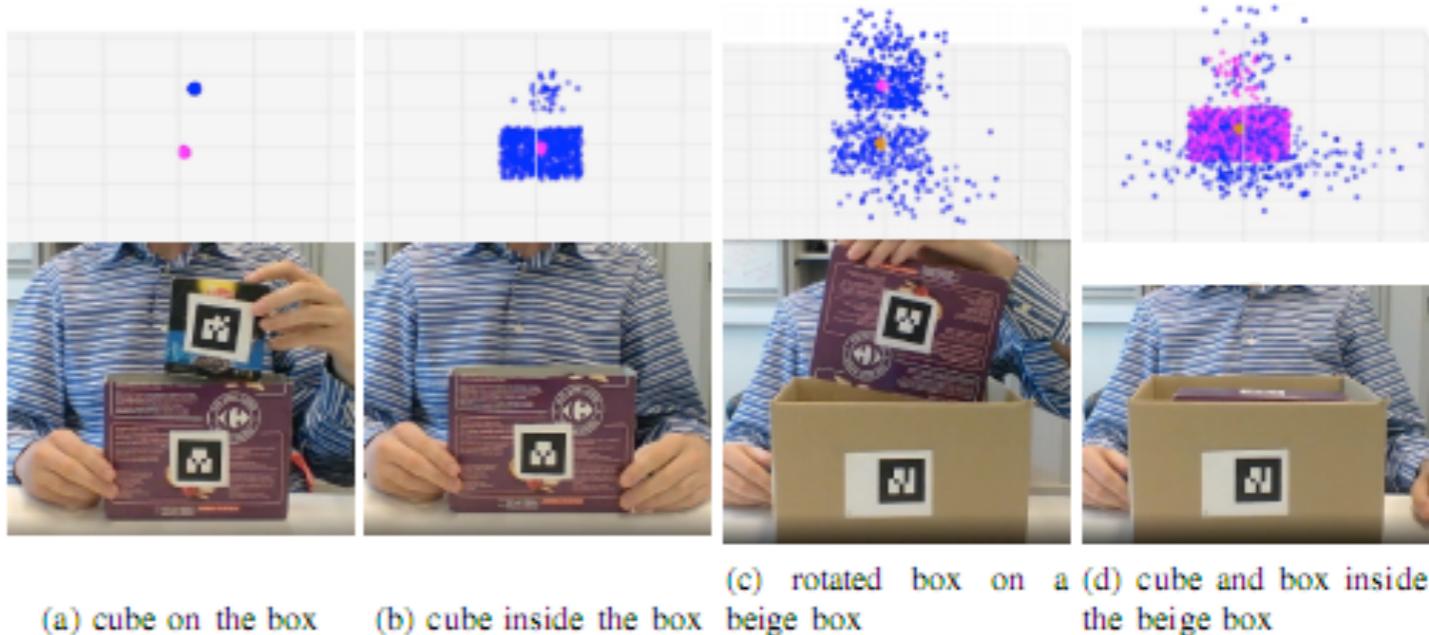
- object tracking
- category estimation from interactions



Relational State Estimation over Time

Magnetism scenario

- object tracking
- category estimation from interactions



Box scenario

- object tracking even when invisible
- estimate spatial relations

Speed 0x Queries (updated every 5 steps)

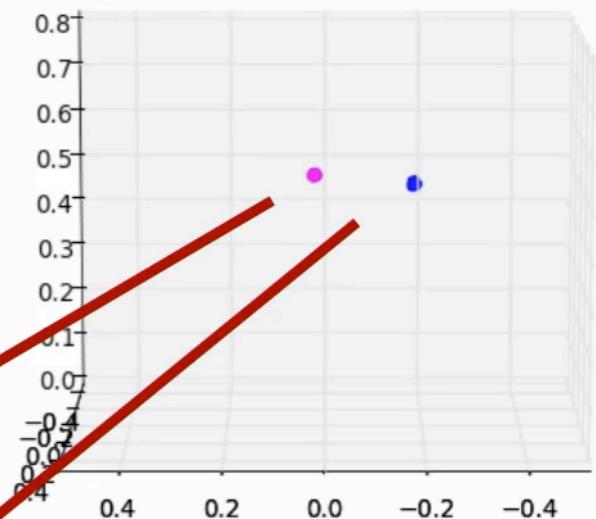
```
[]  
  
on(X,Y):  
[1.0:(3,(table)),1.0:(4,(table))]  
  
inside(X,Y):  
[]  
  
tr_inside(X,Y):  
[]
```



Box ID=4

Cube ID=3

Particles



IROS 13

Speed 0x Queries (updated every 5 steps)

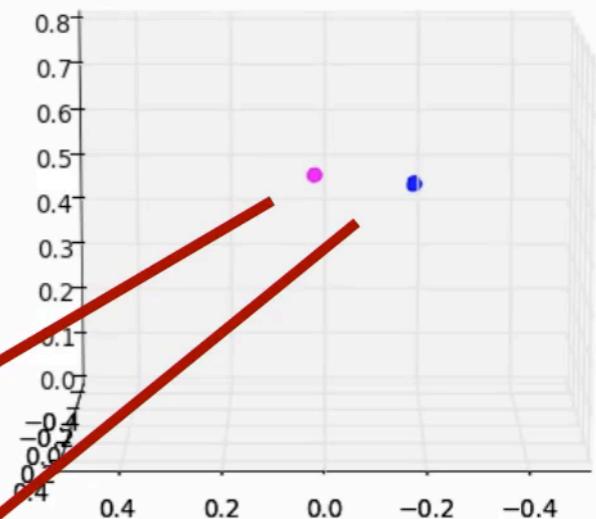
```
[]  
  
on(X,Y):  
[1.0:(3,(table)),1.0:(4,(table))]  
  
inside(X,Y):  
[]  
  
tr_inside(X,Y):  
[]
```



Box ID=4

Cube ID=3

Particles



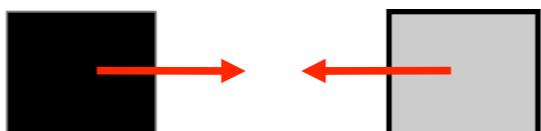
IROS 13

Magnetic scenario

- 3 object types: magnetic, ferromagnetic, nonmagnetic

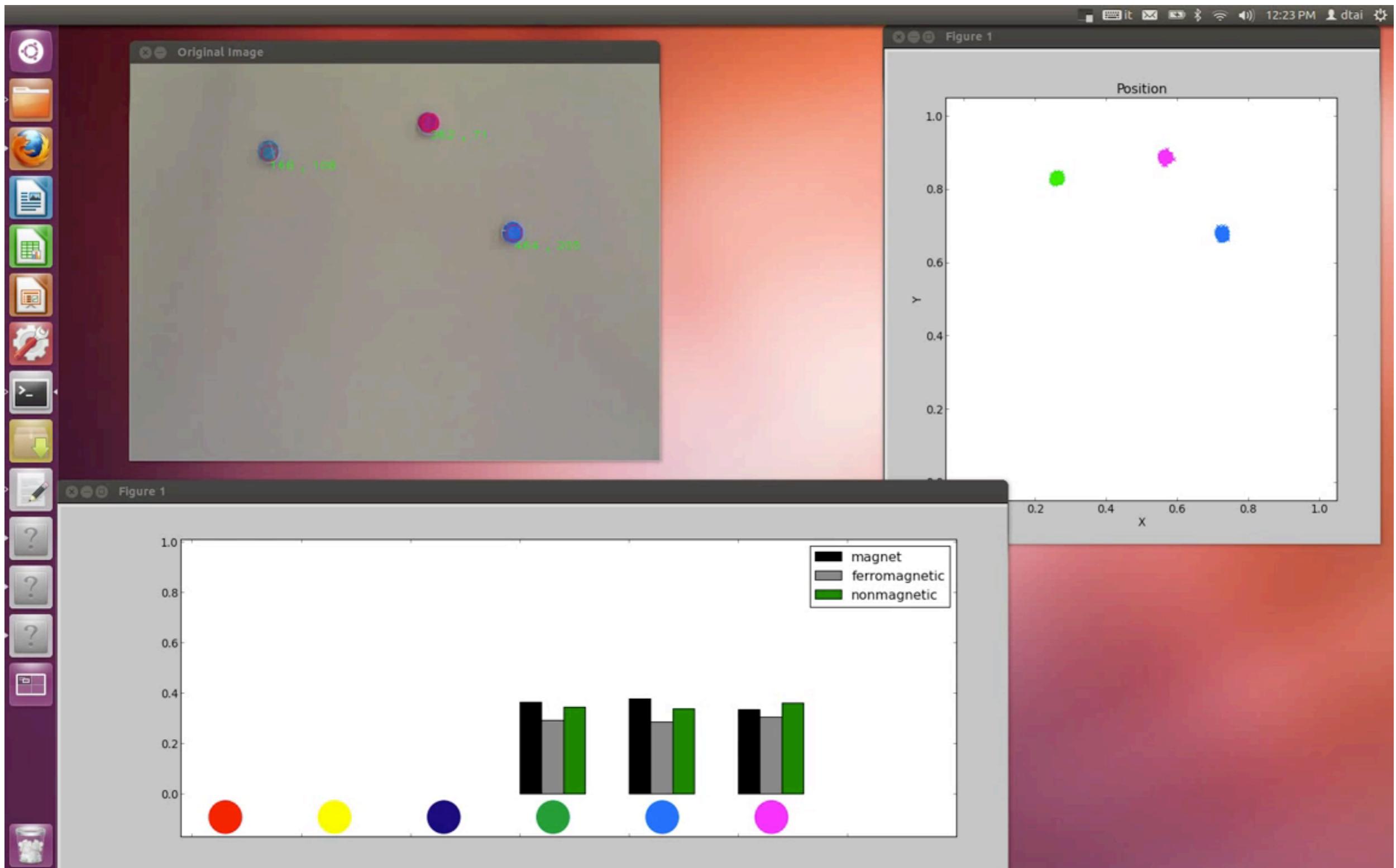


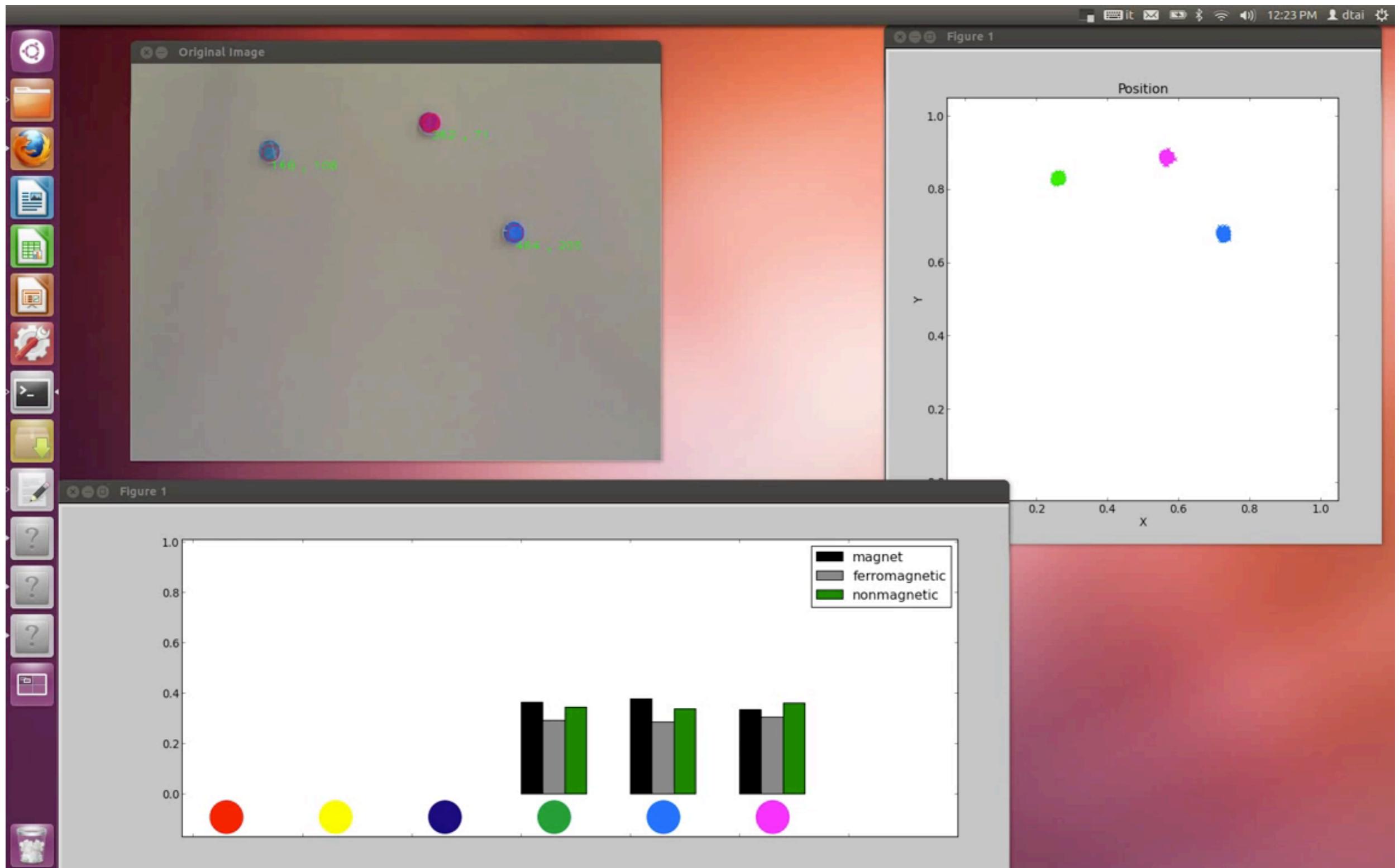
- Nonmagnetic objects do not interact
- A magnet and a ferromagnetic object attract each other



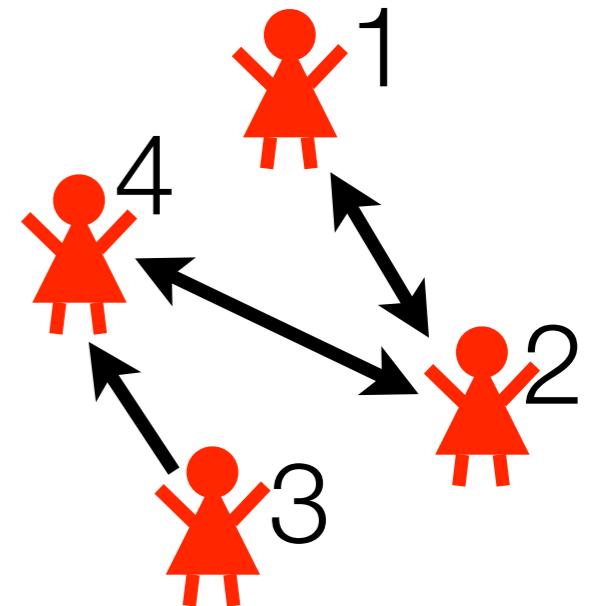
- Magnetic force that depends on the distance
- If an object is held magnetic force is compensated.







DTProbLog



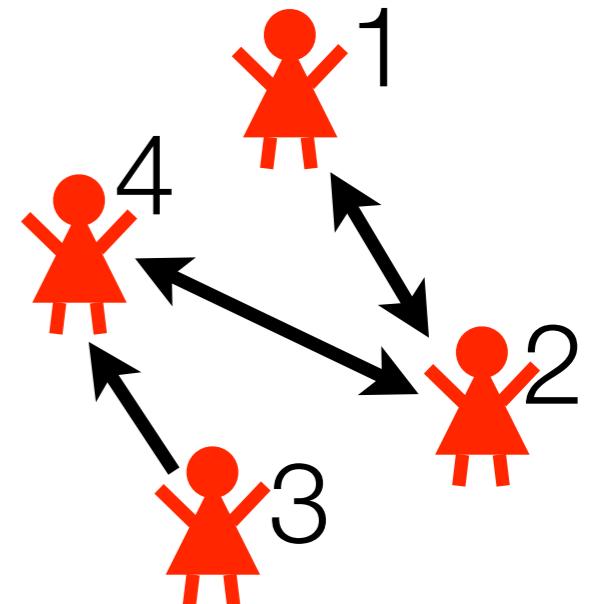
```
person(1).  
person(2).  
person(3).  
person(4).
```

```
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

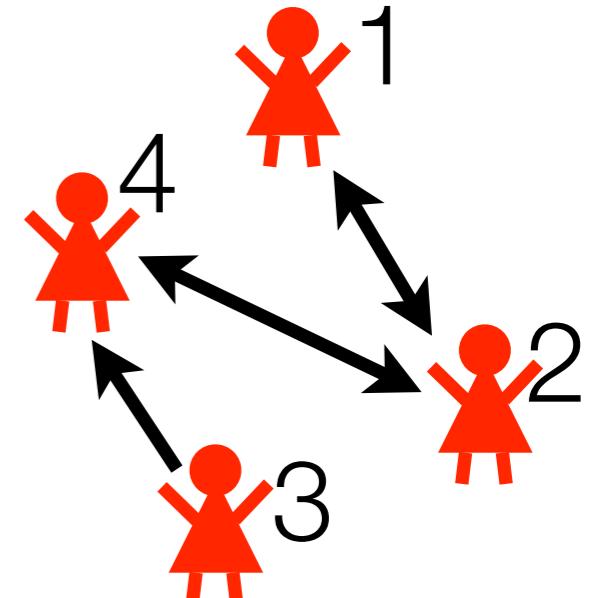
```
? :: marketed(P) :- person(P).
```

```
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).
```

**probabilistic facts
+ logical rules**

```
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).
```

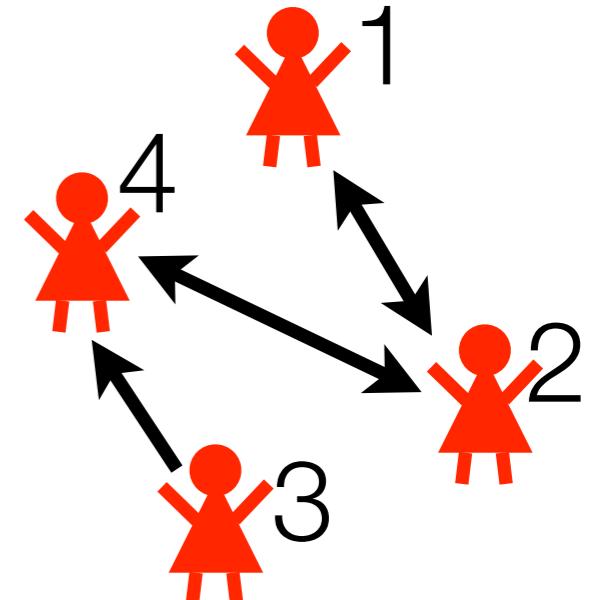
```
0.2 :: buy_marketing(P) :- person(P).
```

```
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).
```

utility facts: cost/reward if true

DTProbLog

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

```
0.3 :: buy_trust(X,Y) :- friend(X,Y).
```

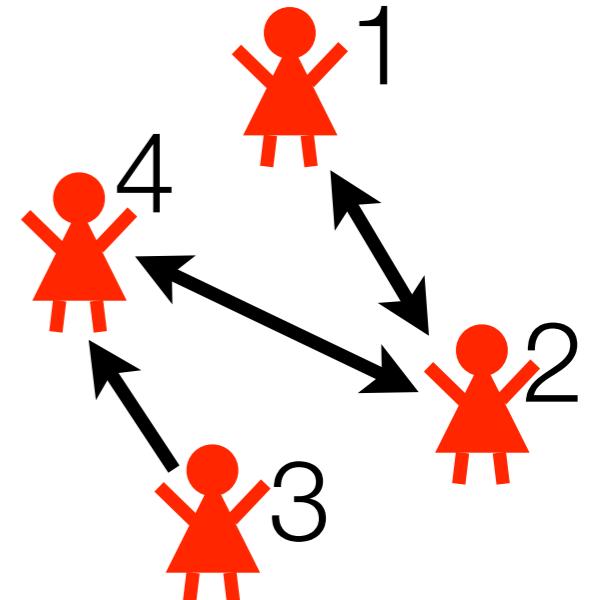
```
0.2 :: buy_marketing(P) :- person(P).
```

```
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).
```

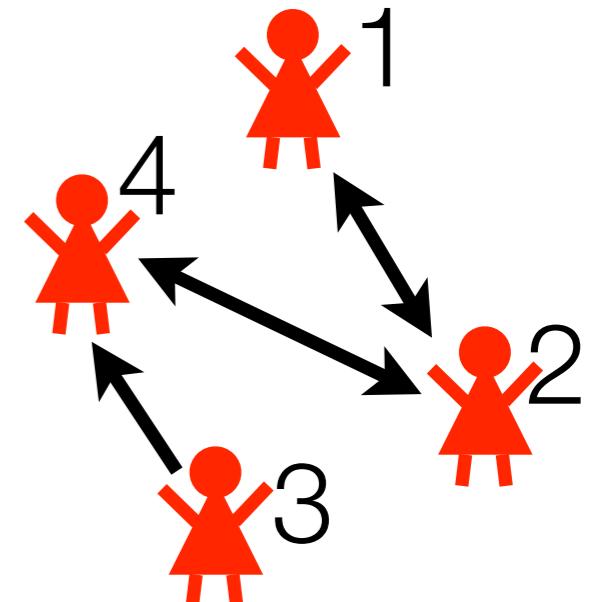
```
0.2 :: buy_marketing(P) :- person(P).
```

```
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).
```

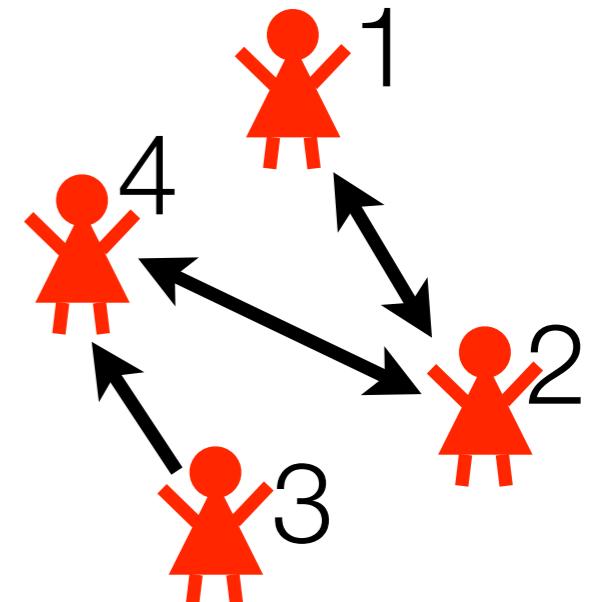
```
0.2 :: buy_marketing(P) :- person(P).
```

```
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).
```

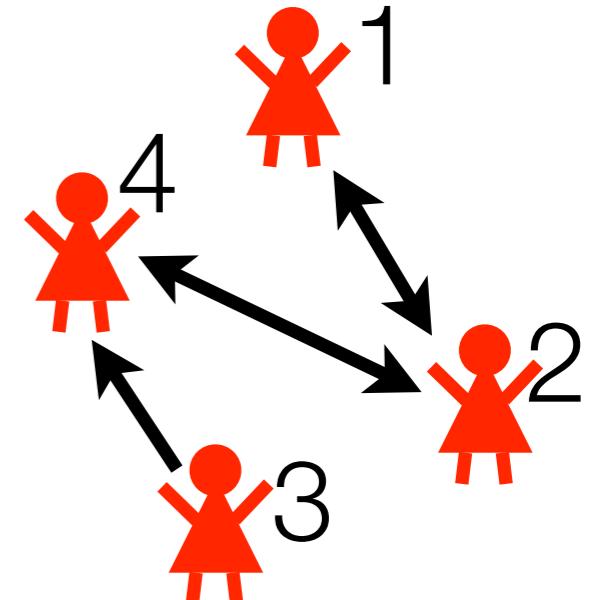
```
0.2 :: buy_marketing(P) :- person(P).
```

```
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).
```

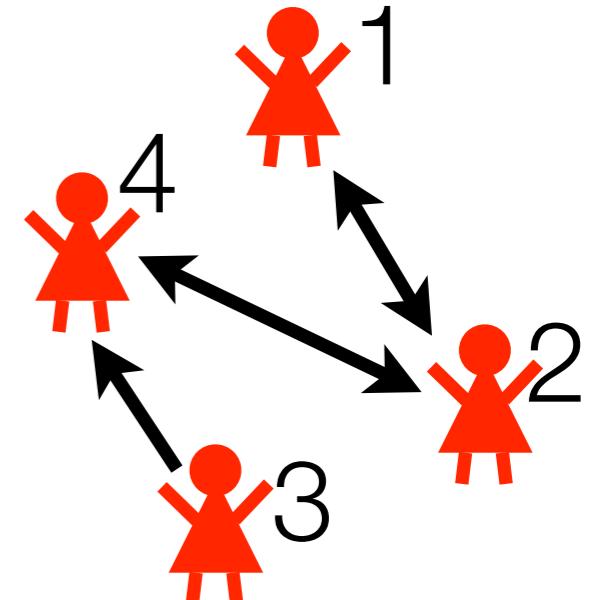
```
0.2 :: buy_marketing(P) :- person(P).
```

```
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)
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) .

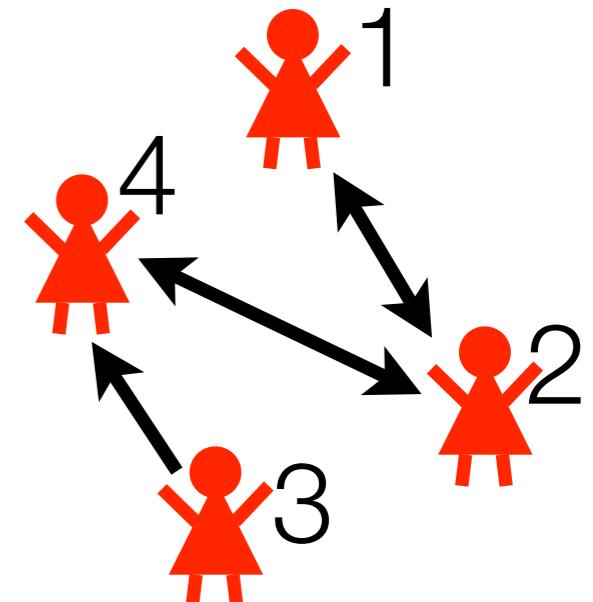
buys(P) => 5 :- person(P) .

marketed(P) => -3 :- person(P) .

$$\text{utility} = -3 + -3 + 5 + 5 = 4$$

$$\text{probability} = 0.0032$$

marketed(1)		marketed(3)
	bt(2,1) bt(2,4)	bm(1)
buys(1)	buys(2)	



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).

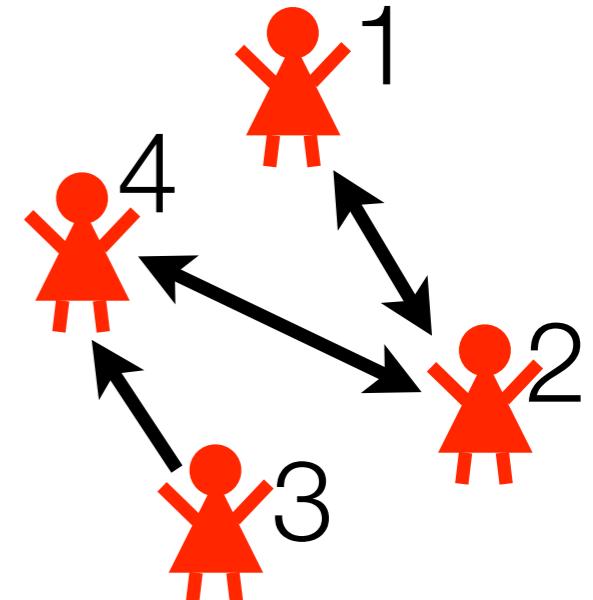
buys(P) => 5 :- person(P).

marketed(P) => -3 :- person(P).

$$\text{utility} = -3 + -3 + 5 + 5 = 4$$

$$\text{probability} = 0.0032$$

marketed(1)		marketed(3)
	bt(2,1) bt(2,4)	bm(1)
buys(1)	buys(2)	



person(1).

person(2).

person(3).

person(4).

friend(1,2).

friend(2,1).

friend(2,4).

friend(3,4).

friend(4,2).

world contributes
0.0032×4 to
expected utility of
strategy

DTProbLog

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

```
0.3 :: buy_trust(X,Y) :- friend(X,Y).
```

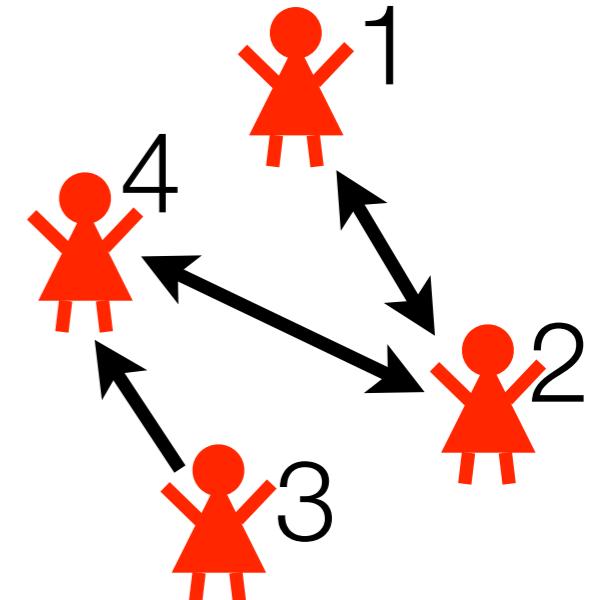
```
0.2 :: buy_marketing(P) :- person(P).
```

```
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

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).
```

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

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	

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).
```

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	

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 first-order logic formulas

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	

cProbLog: constraints on possible worlds

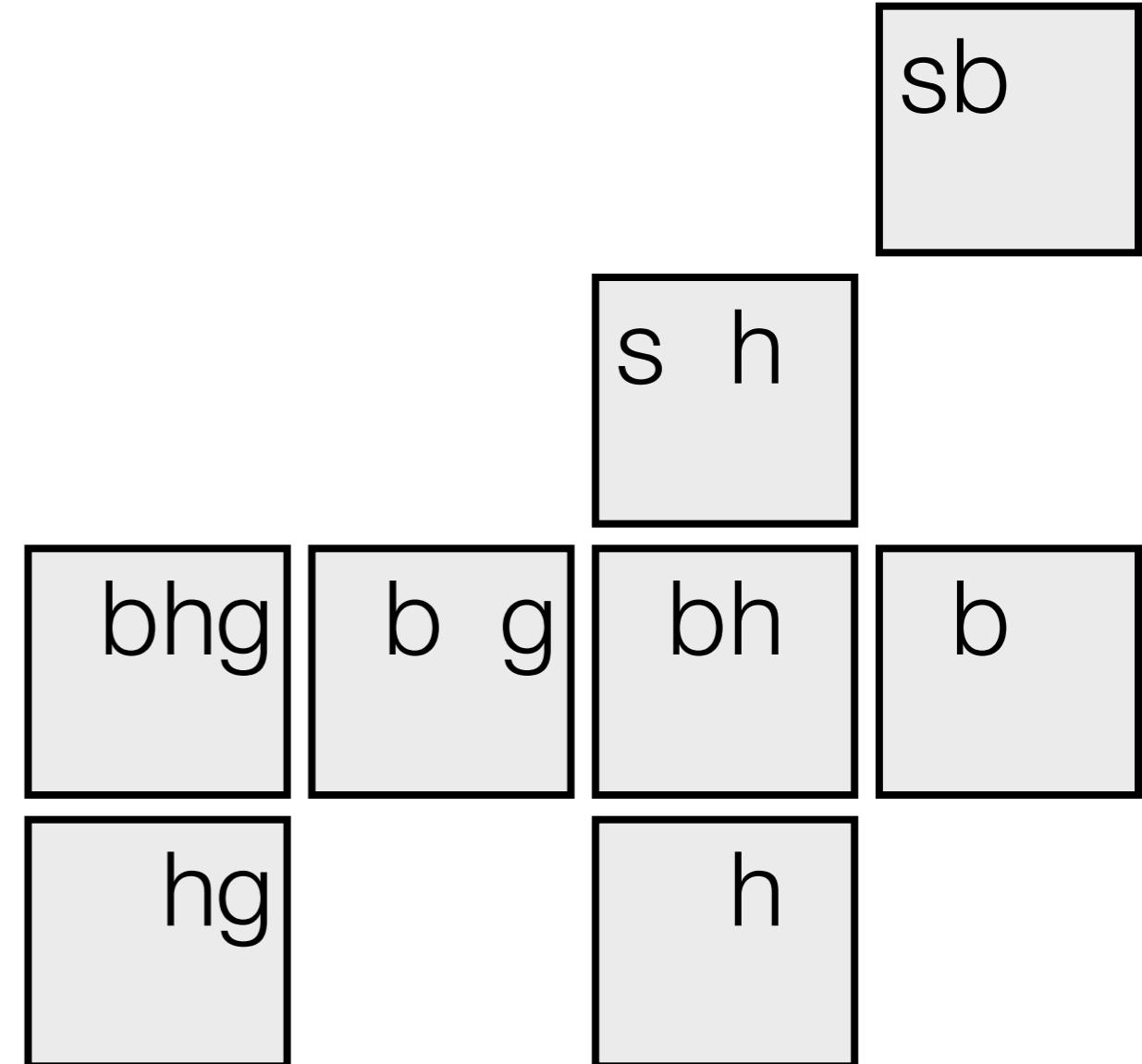
```
weight(skis, 6).  
weight(boots, 4).  
weight(helmet, 3).  
weight(gloves, 2).
```

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

```
excess(Limit) :- ...
```

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

constraints as first-order logic formulas



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

sb

s h

bhg

b g

bh

b

hg

h

constraints as first-order logic formulas

Summary

Getting started

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

- Interactive tutorial
- Online interface for inference and parameter estimation
- Offline version for download
(or `pip install problog`)

The screenshot shows a web-based ProbLog editor. At the top, there's a navigation bar with links for ProbLog, Home, Download, Publications, Help, People, Tutorial, and Online. The main content area has a heading about higher-order functions and meta-predicates. Below that, a code editor displays the following Prolog-like code:

```
1 :- use_module(library(apply)).  
2 :- use_module(library(lists)).  
3 :-  
4 PH::make_coin(C,PH).  
5 coin(C) :- make_coin(C,0.8).  
6 cumsum(C, In, Out) :- coin(C), Out is In+1; !, coin(C), Out=In.  
7 totals(S) :-  
8 ... findall(X, between(1,10,X), L),  
9 ... foldl(cumsum, L, 0, S).  
10 :-  
11 query(total(_)).
```

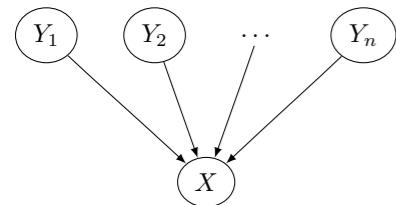
Below the code editor is a button labeled "Evaluate". To the right, a table shows the results of the query `total(_)`:

Query	Location	Probability
total(0)	11:7	1.024e-
total(1)	11:7	4.096e-
total(2)	11:7	7.3728e-
total(3)	11:7	0.000786432
total(4)	11:7	0.005505024
total(5)	11:7	0.026424115

Assignment

Write your own state-of-the-art inference and learning engine based on recent advances in probabilistic graphical models

Bayesian net



logic
encoding

```
0.3::x :- y(1).
0.4::x :- y(2).
...
0.6::x :- y(n).
evidence(y(1)).
query(x).
```

ProbLog program

$$x \Leftrightarrow y(1) \vee y(2) \vee \dots \vee y(n)$$

y(1)

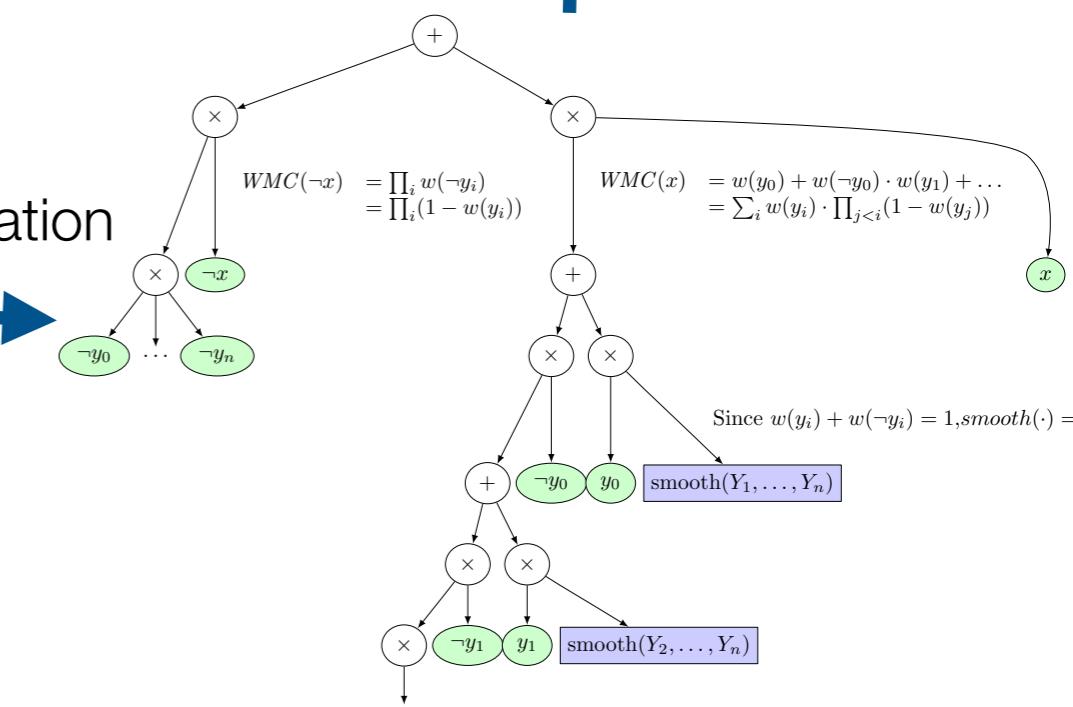
CNF

$$\begin{aligned} &x \vee \neg y(1) \\ &x \vee \neg y(2) \\ &\dots \\ &x \vee \neg y(n) \\ &\neg x \vee y(1) \vee y(2) \vee \dots \vee y(n) \\ &y(1) \end{aligned}$$

Knowledge Compilation

$$\Pr(x \mid y(1))$$

Model Counting



Acknowledgements

Maurice Bruynooghe
Bart Demoen
Luc De Raedt
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
Jonas Vlasselaer