An Introduction to Statistical Relational Learning

Luc De Raedt
Katholieke Universiteit Leuven

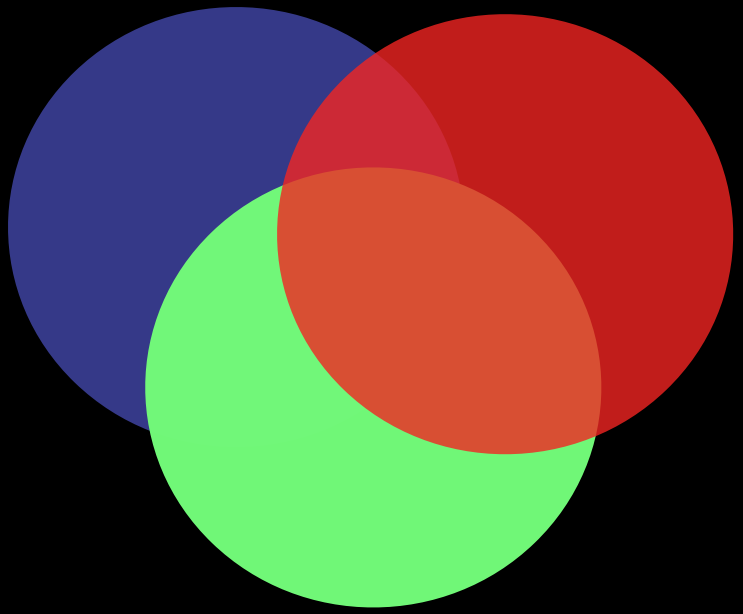
joint work with Kristian Kersting

Probabilistic Logic Learning*

One of the key open questions of artificial intelligence concerns

"probabilistic logic learning",

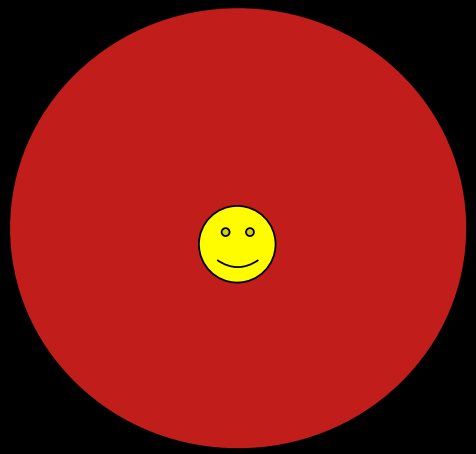
i.e. the integration of probabilistic reasoning with logical or relational representations and machine learning.



*In the US, sometimes called Statistical Relational Learning

Probabilistic Logic Learning*

The knowledge representation framework par excellence



Logic: Syntax

Variable
constant

class(Book,Topic) :- Clause(s)

author(Book,Author), favorite-topic(Author,Topic).

the class of Book is Topic

the Book is written by Author

Topic is the favorite of the Author

favorite-topic(rowlings,fantasy) Fact(s)

the favorite topic of rowlings is fantasy

predicate/relation

Logic: three views

entailment/logical consequence

H |= class(harrypotter,fantasy)

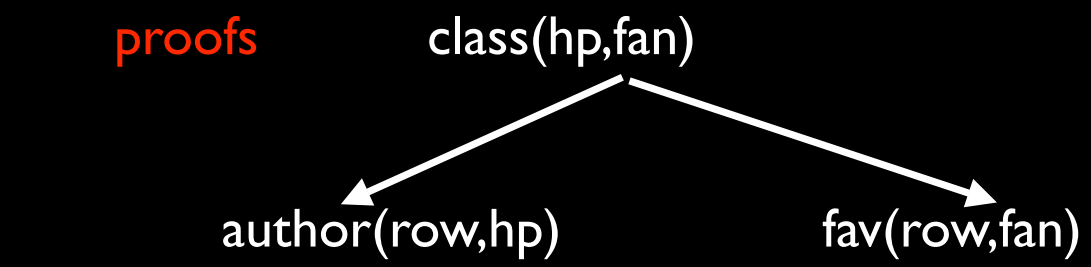
class(harrypotter,fantasy) follows from H

interpretation {class(hp,fan), author(row,hp), fav(row,fan)}

is a model for H

is a possible world given H

proofs

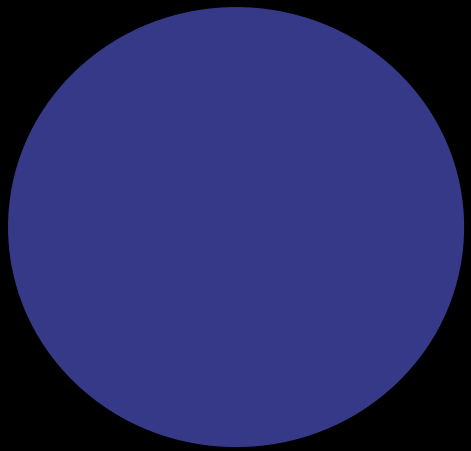


Probabilistic Logic Learning*

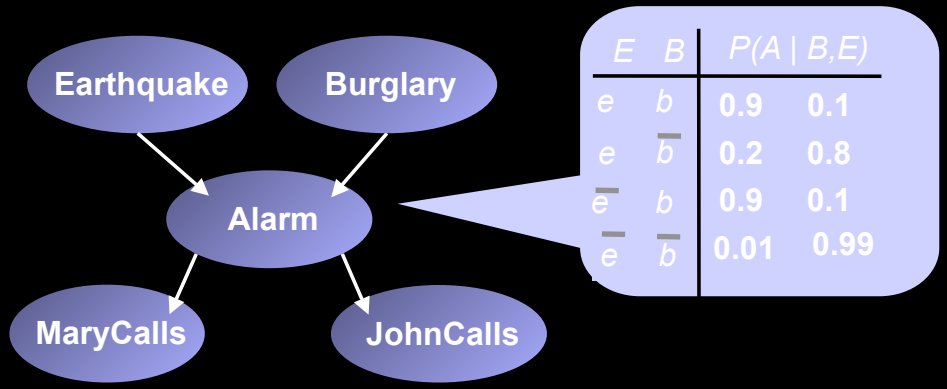
Reasoning about uncertainty

Graphical models, such as Bayesian and Markov Networks

Probabilistic grammars and databases.



Bayesian Networks



$$P(E,B,A,J,M) = P(E).P(B).P(A|E).\ P(A|B).P(J|A).P(M|A)$$

Structure &
Parameters

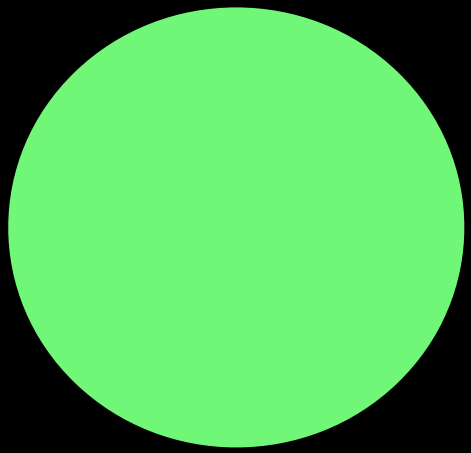
INTERPRETATION
STATE/DESCRIPTION
{A, ¬E,¬B, J, M}

Probabilistic Logic Learning*

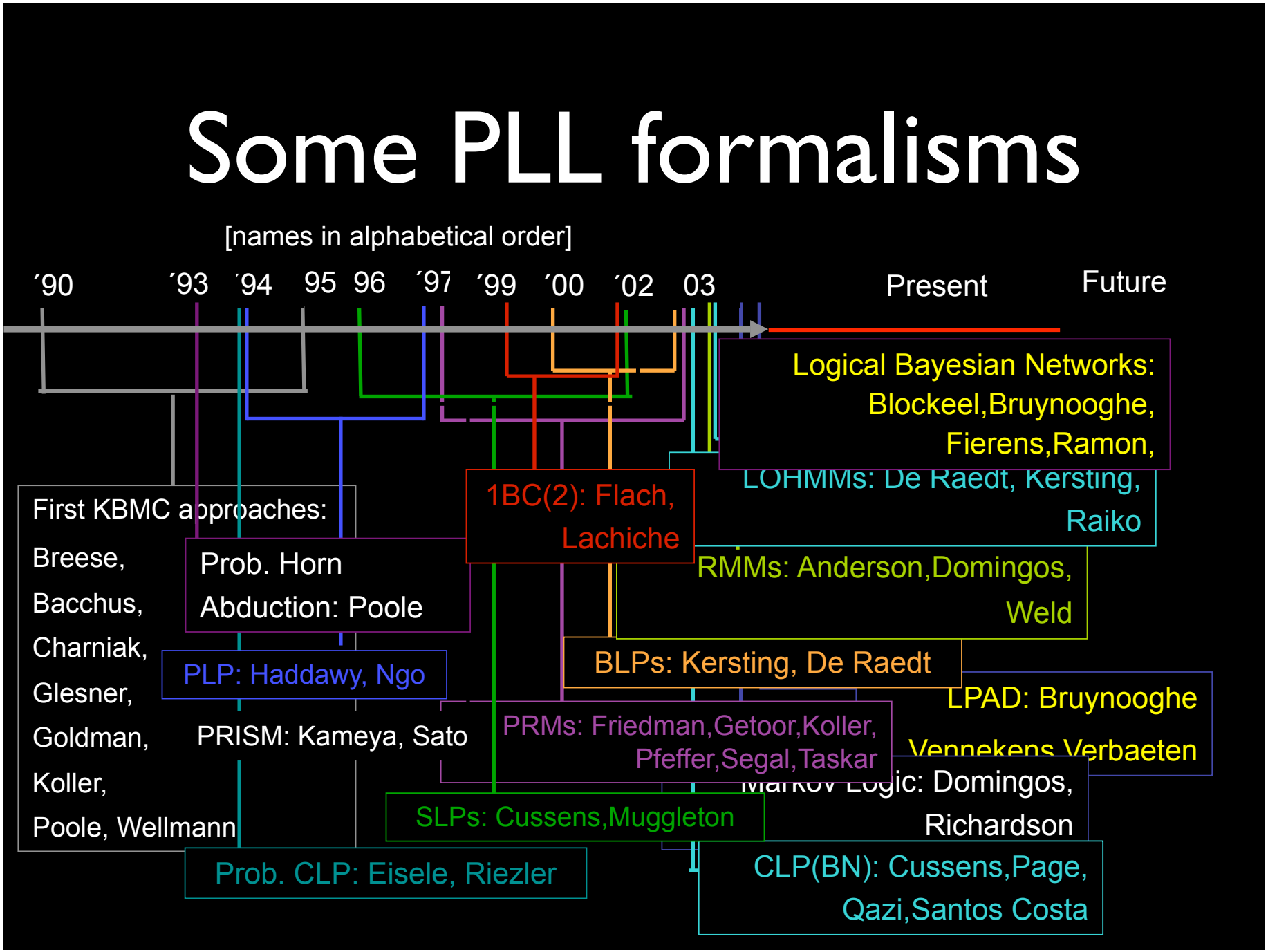
Learning the model from data

Structure & Parameters

Logical & Statistical



Some PLL formalisms



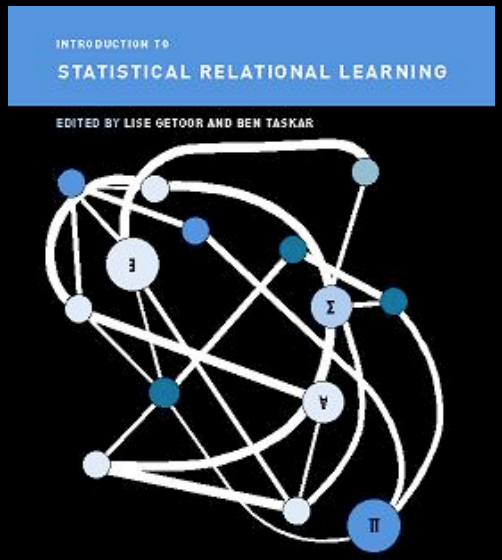
What I am not going to do

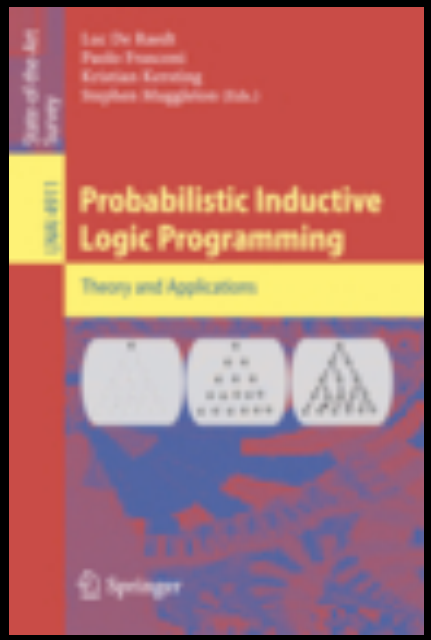
- Introduce yet another "lingua franca" or probabilistic logic
  - analogy with programming languages & knowledge representation
- Talk a lot about logic

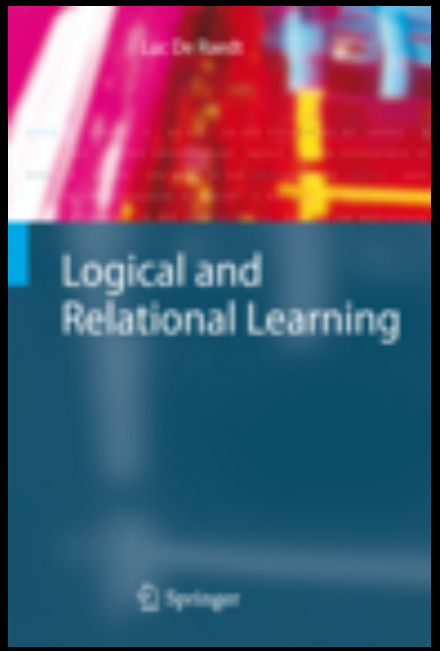
Rather

- I will start from logic + learning and extend it with probabilistic reasoning
- I will argue that traditional logic + learning settings apply to probabilistic logic learning as well
- I will argue that this is probably useful for applications and that learning is essential

Books







Also, survey paper De Raedt, Kersting, SIGKDD Explorations 03

Overview

A. Motivation for Probabilistic Logic Learning

B. Logic + Learning

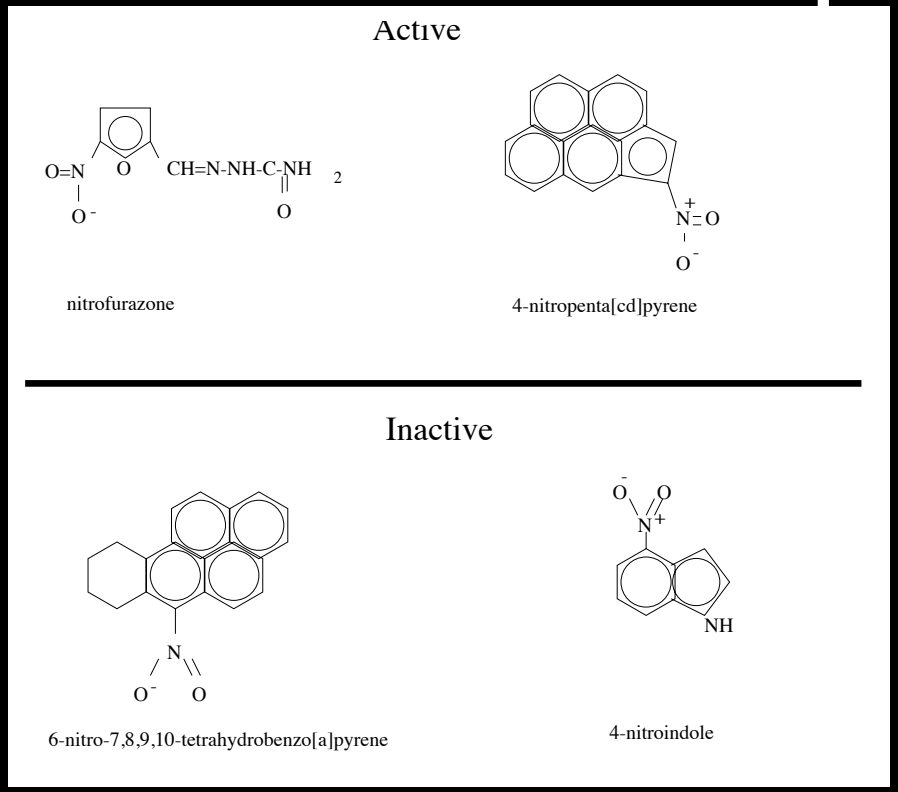
C. Adding probability

A. Motivation

A. Motivation

A.1 Some cases

Case 1: Structure Activity Relationship Prediction



[Srinivasan et al. AIJ 96]

Structural alert:

General Purpose
Logic Learning System

Data = Set of Small Graphs

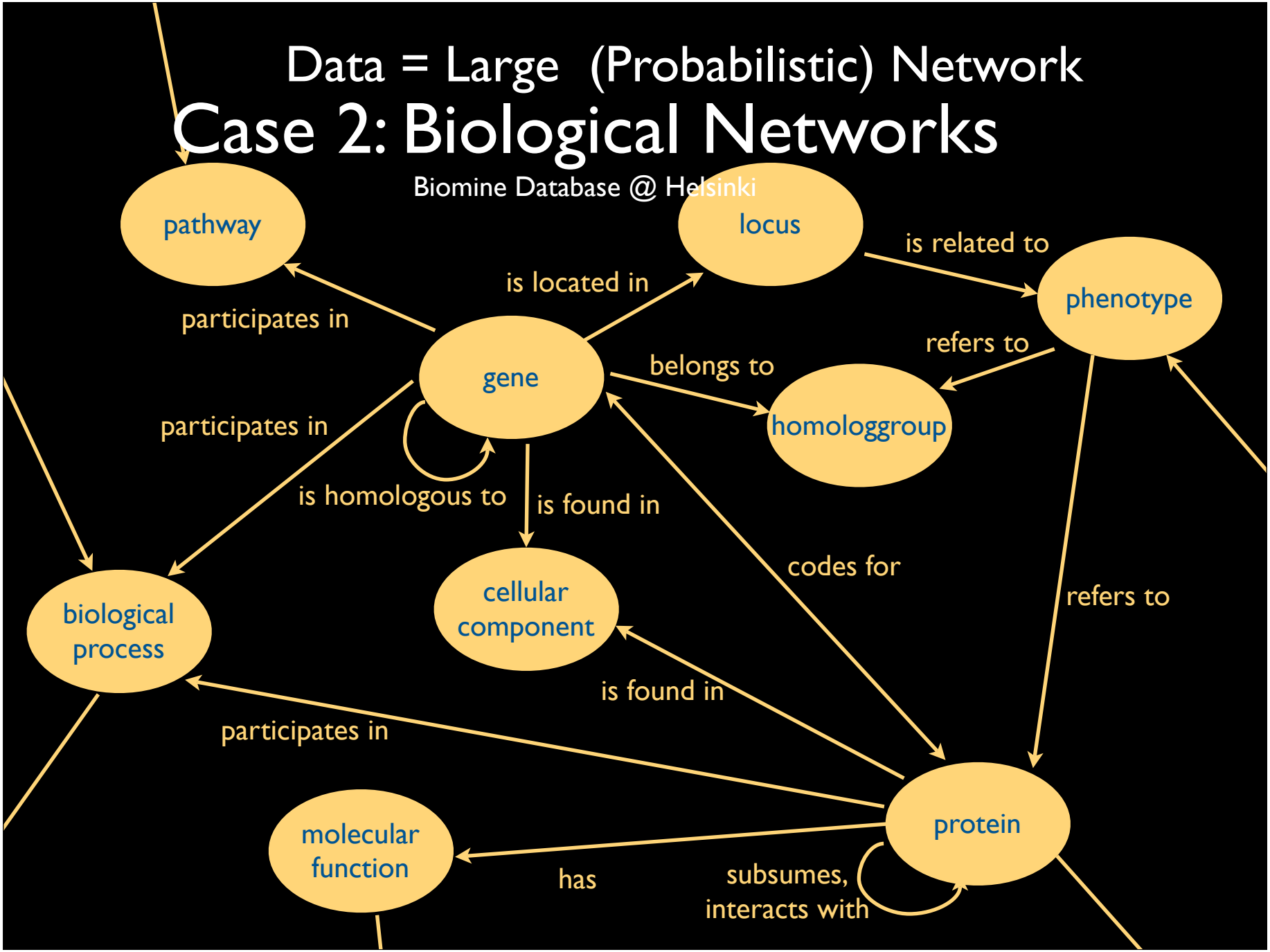
Uses and Produces
Knowledge

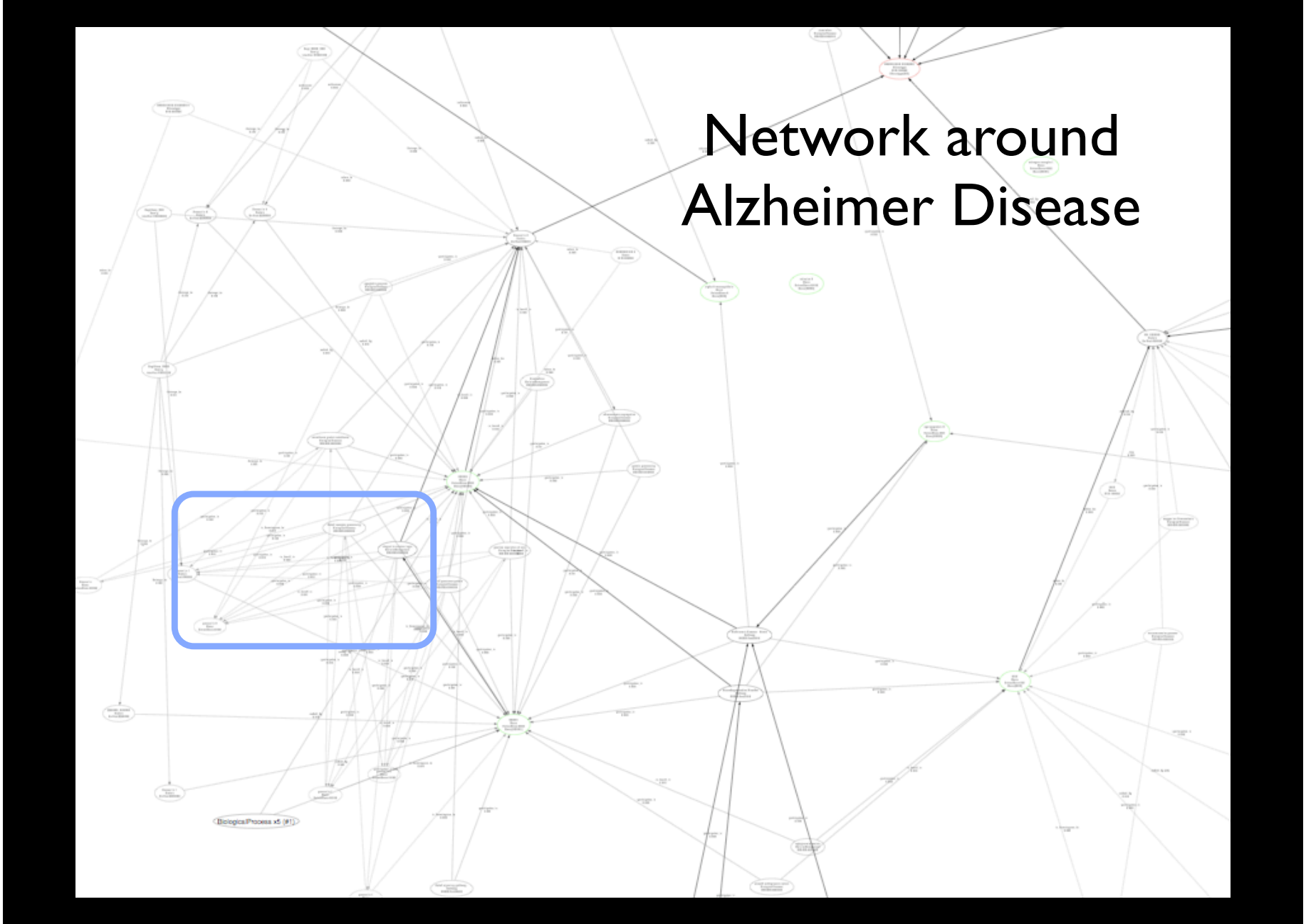
Using and Producing Knowledge

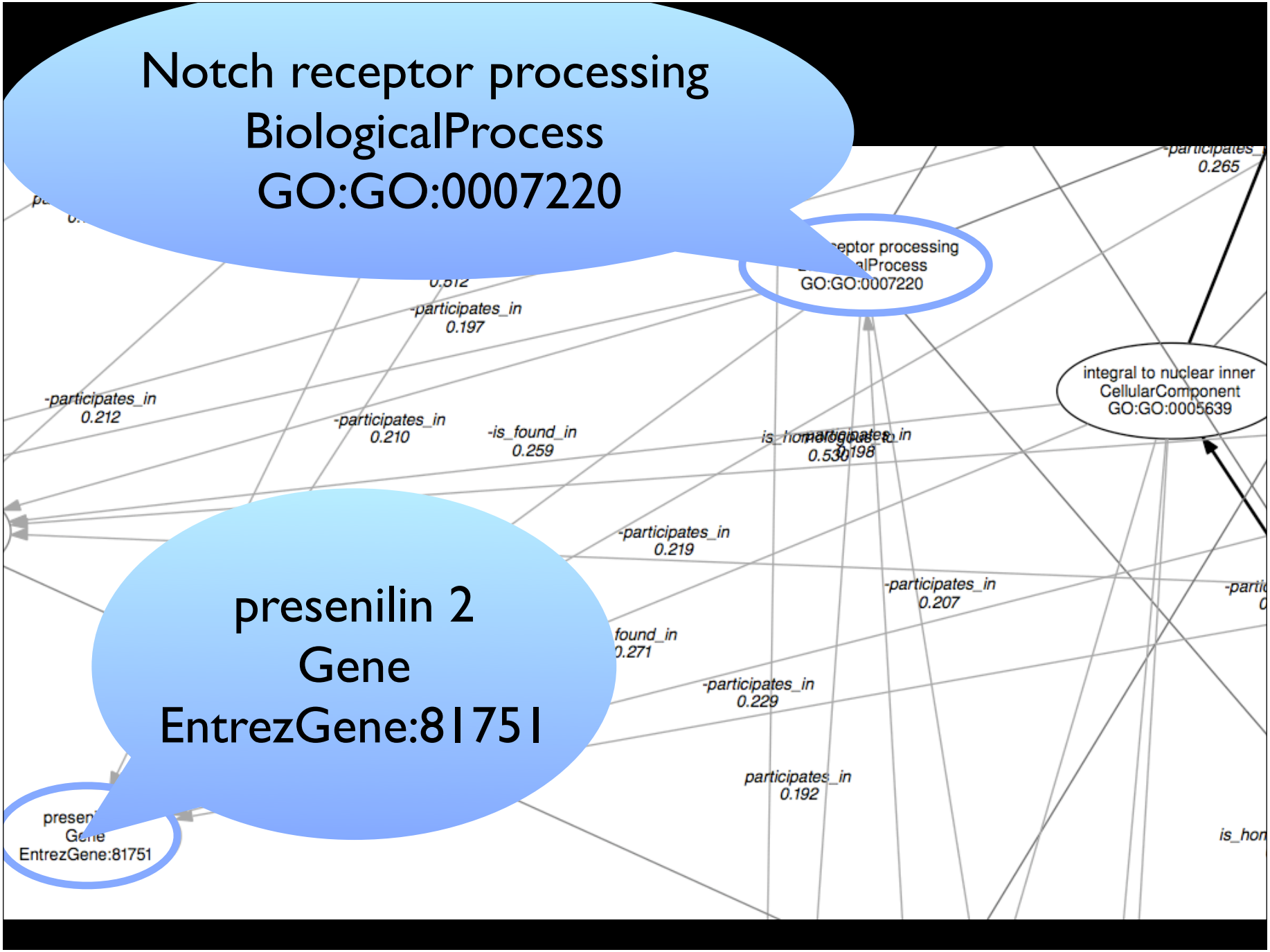
Logical and relational learning systems can use and produce knowledge (ILP)

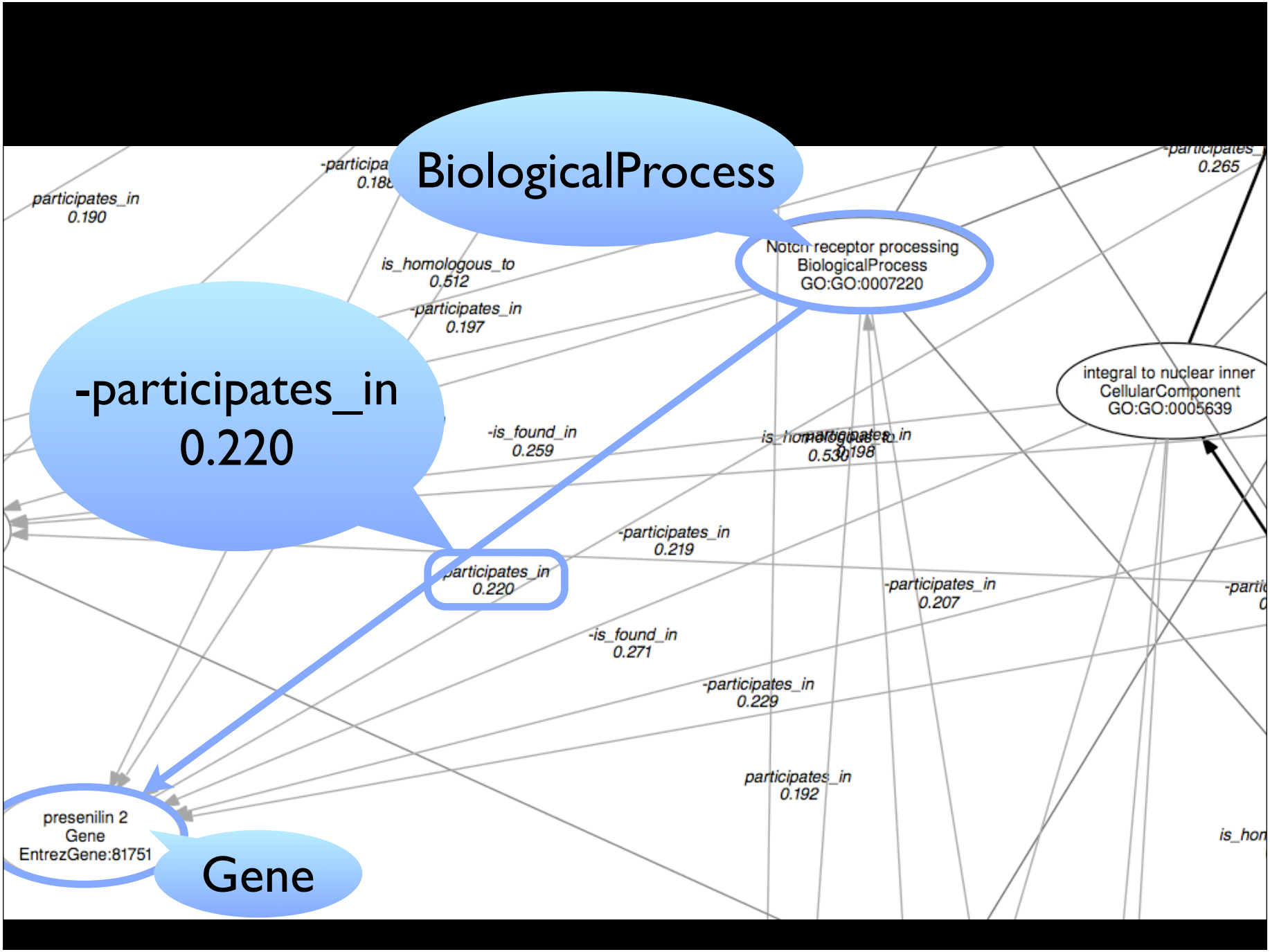
Result of learning task is understandable and interpretable

Logical and relational learning algorithms can use background knowledge, e.g. ring structures


Data = Large (Probabilistic) Network
Case 2: Biological Networks
Biomine Database @ Helsinki
pathway
locus
is related to
phenotype
is located in
participates in
refers to
gene
belongs to
homologgroup
participates in
is homologous to
is found in
codes for
refers to
biological process
cellular component
is found in
participates in
protein
molecular function
has
subsumes, interacts with


Network around
Alzheimer Disease


Notch receptor processing
BiologicalProcess
GO:GO:0007220
presenilin 2
Gene
EntrezGene:81751
-participates_in
0.197
-participates_in
0.212
-participates_in
0.210
-is_found_in
0.259
-participates_in
0.219
-participates_in
0.207
found_in
0.271
-participates_in
0.229
participates_in
0.192
integral to nuclear inner
CellularComponent
GO:GO:0005639
0.265
0.198
0.530
EntrezGene:81751


BiologicalProcess
-participates_in
0.220
Gene
participates_in
0.190
is_homologous_to
0.512
-participates_in
0.197
Notch receptor processing
BiologicalProcess
GO:GO:0007220
integral to nuclear inner
CellularComponent
GO:GO:0005639
-is_found_in
0.259
-participates_in
0.219
participates_in
0.220
-participates_in
0.207
-is_found_in
0.271
-participates_in
0.229
participates_in
0.192
presenilin 2
Gene
EntrezGene:81751

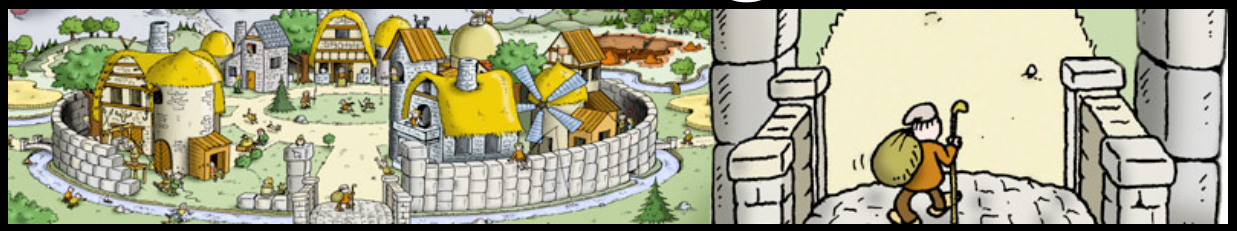
Questions to ask

How to support the life scientist in using and discovering new knowledge in the network ?

- Is gene X involved in disease Y ?
- Should there be a link between gene X and disease Y ? If so, what type of link ?
- What is the probability that gene X is connected to disease Y ?
- Which genes are similar to X w.r.t. disease Y?
- Which part of the network provides the most information (network extraction) ?
- ...

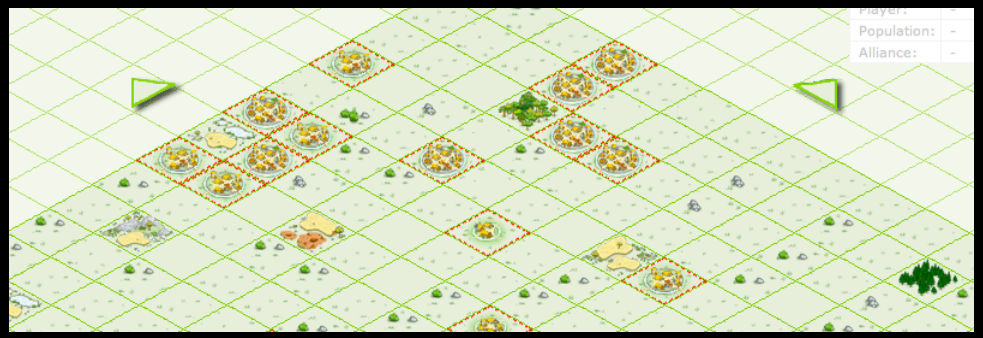
See *Invited Talk on Tuesday*

Case 3: Evolving Networks



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

[Thon et al. ECML 08]



World Dynamics

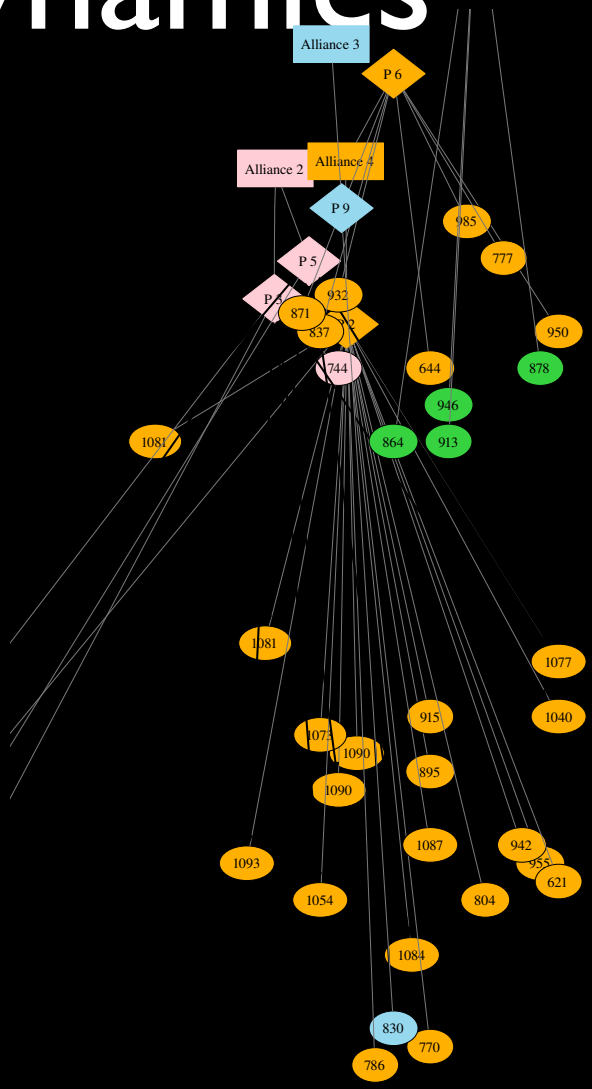
Fragment of world with

~10 alliances
~200 players
~600 cities

alliances color-coded

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

[Thon, Landwehr, De Raedt, ECML08]



World Dynamics

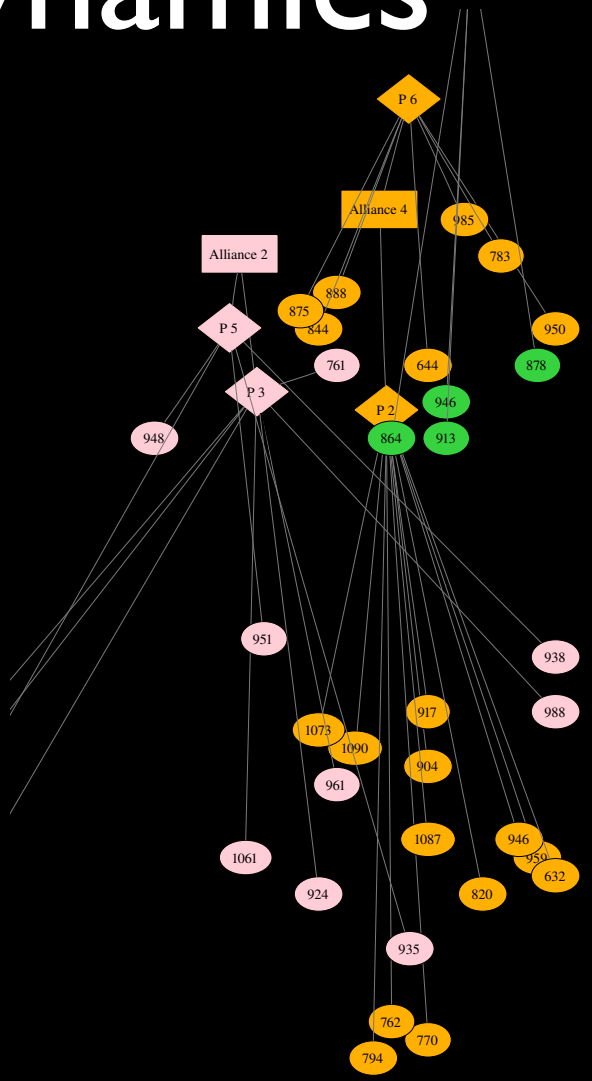
Fragment of world with

~10 alliances
~200 players
~600 cities

alliances color-coded

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

[Thon, Landwehr, De Raedt, ECML08]



World Dynamics

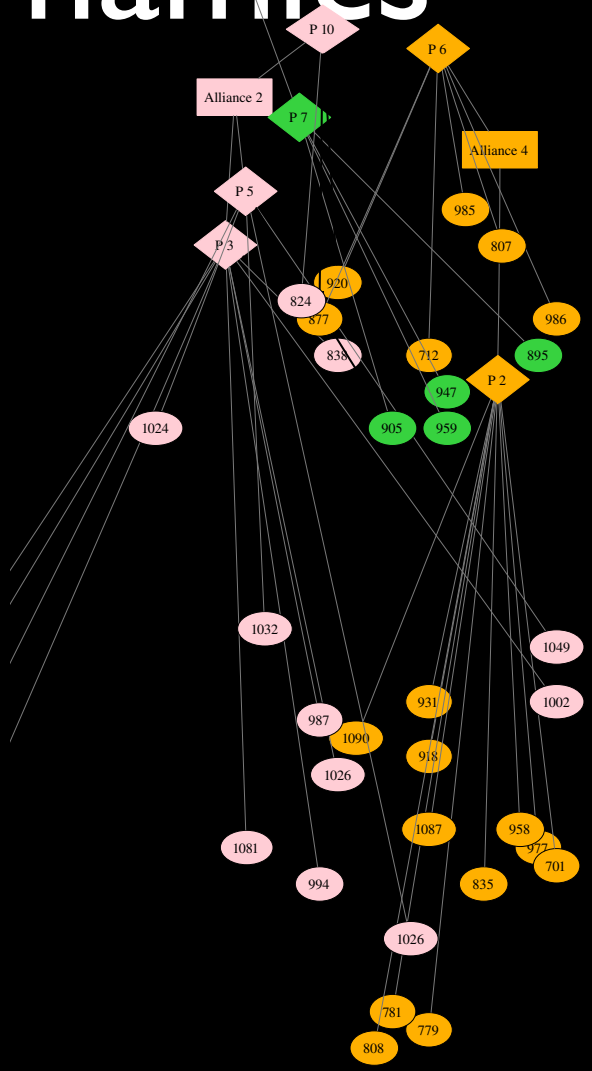
Fragment of world with

~10 alliances
~200 players
~600 cities

alliances color-coded

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

[Thon, Landwehr, De Raedt, ECML08]



World Dynamics

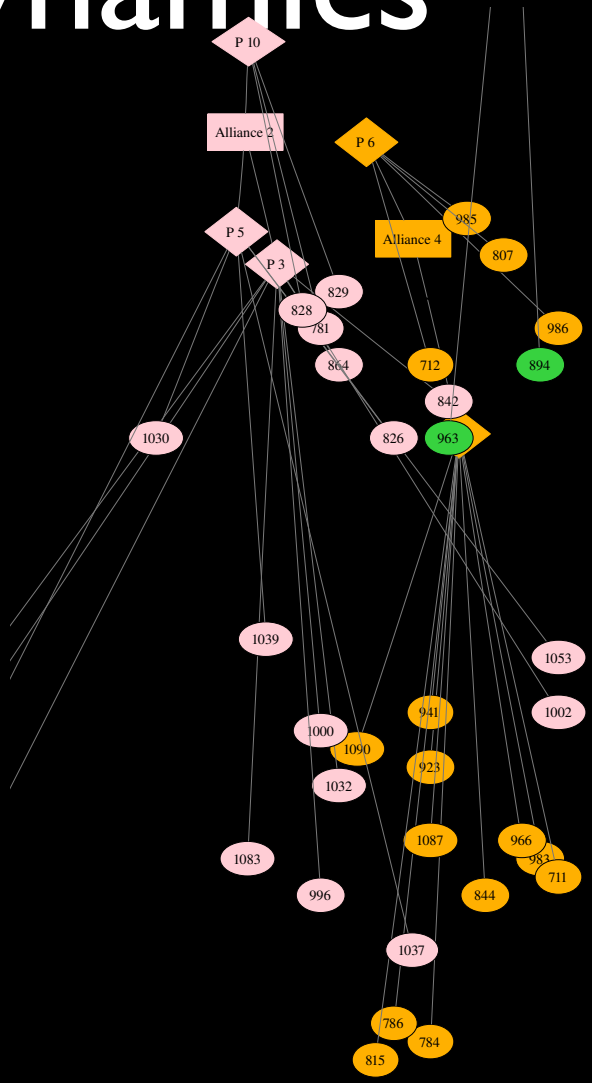
Fragment of world with

~10 alliances
~200 players
~600 cities

alliances color-coded

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

[Thon, Landwehr, De Raedt, ECML08]



World Dynamics

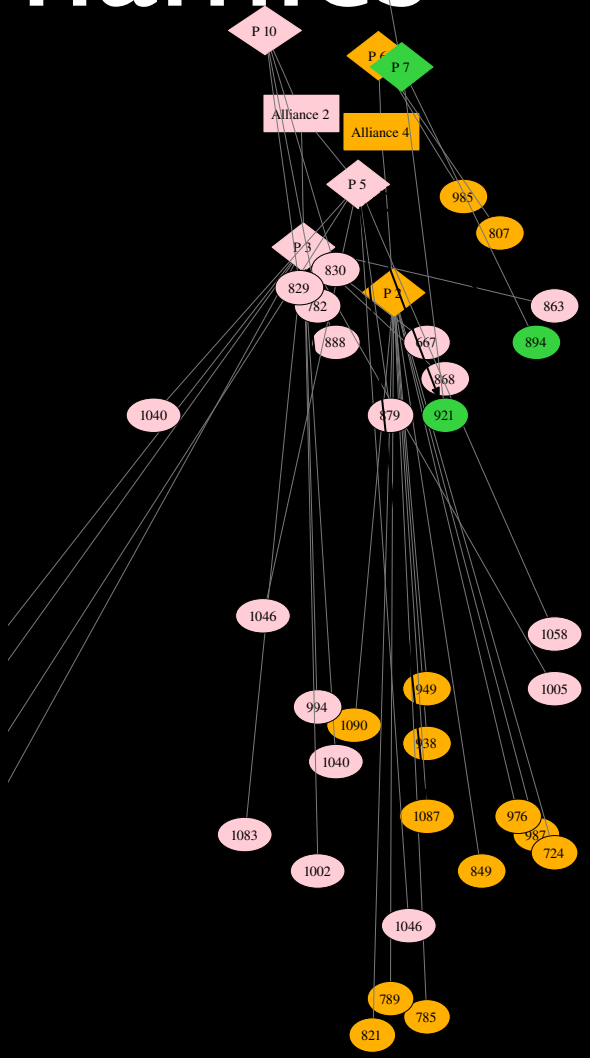
Fragment of world with

~10 alliances
~200 players
~600 cities

alliances color-coded

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

[Thon, Landwehr, De Raedt, ECML08]



World Dynamics

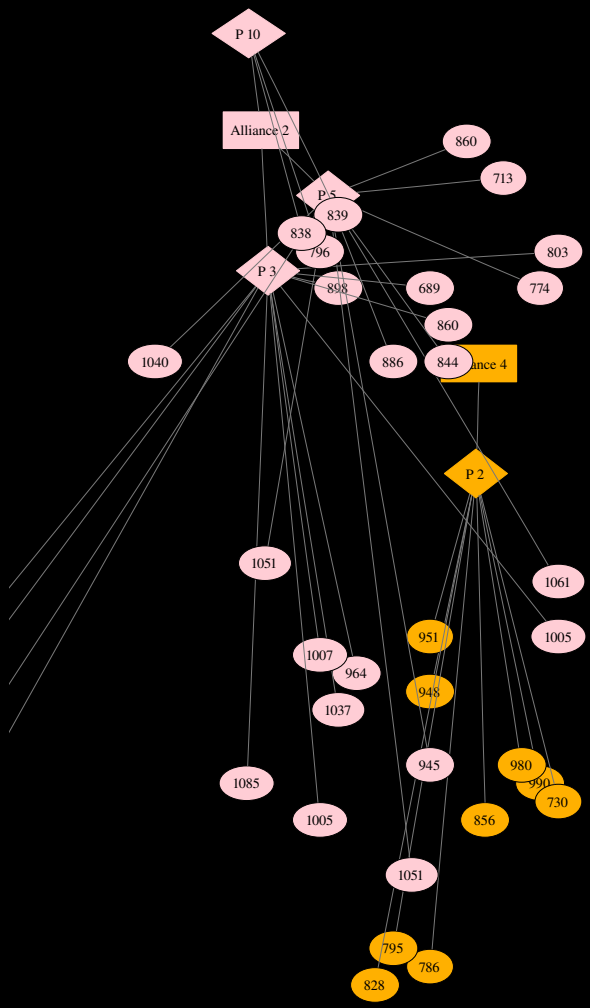
Fragment of world with

~10 alliances
~200 players
~600 cities

alliances color-coded

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

[Thon, Landwehr, De Raedt, ECML08]



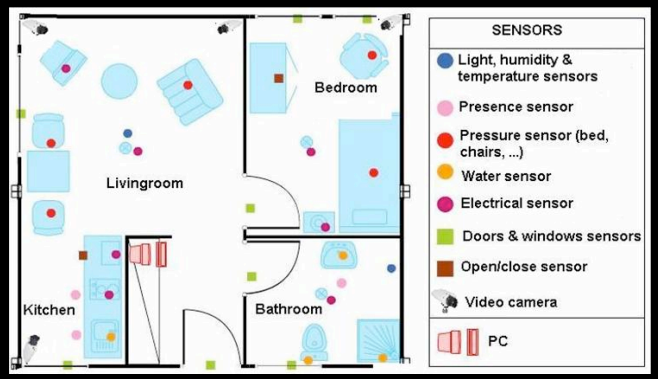
Emerging Data Sets

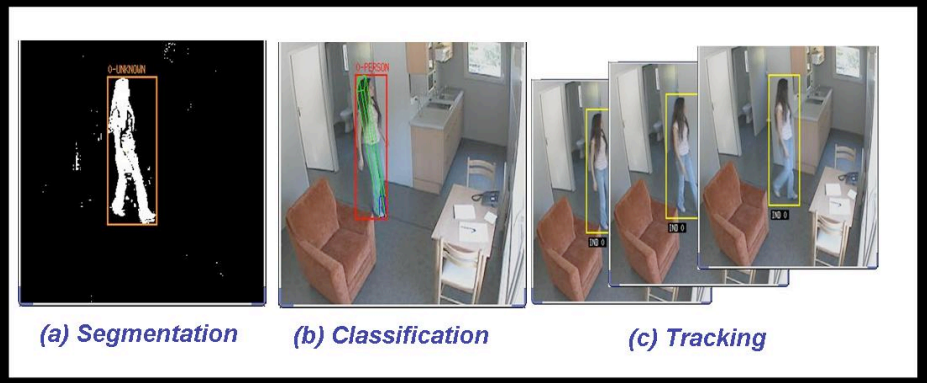
In many application areas :

- vision, surveillance, activity recognition, robotics, ...
- data in relational format are becoming available
- use of knowledge and reasoning is essential
- in Travian -- ako STRIPS representation

GerHome Example

Action and Activity Learning





(courtesy of Francois Bremond, INRIA-Sophia-Antipolis)

http://www-sop.inria.fr/orion/personnel/Francois.Bremond/topicsText/gerhomeProject.html

“Relational” Robotics

Robotics

- several lower level problems have been (more or less) solved
- time to bridge the gap between lower and higher level representations ?

[Kersting et al. Adv. Robotics 07]

Motivation for PLL

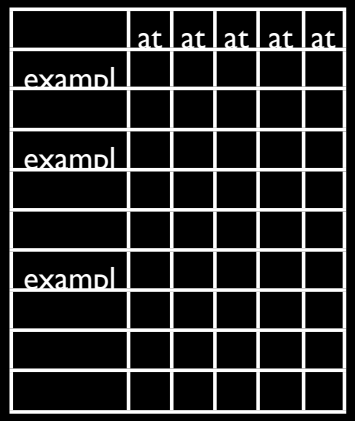
- Representational limitations of (standard)
  - probabilistic representations & logic
  - We: computational logic
- An example
  - graphs & networks
  - Essential : objects & relationships !! In many applications.

A. Motivation

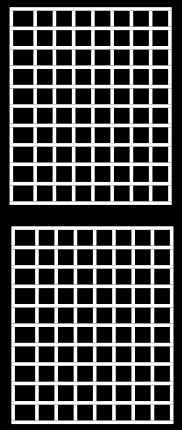
A.2 Representational Issues

Represent the data Hierarchy

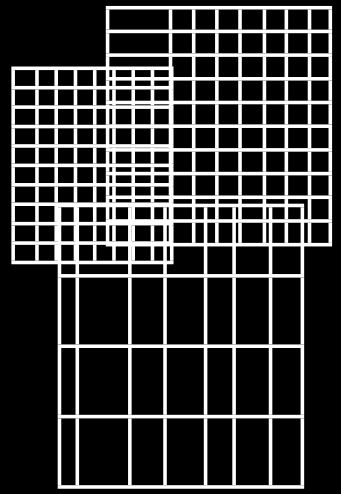
single-table
single-tuple
attribute-value



single-table
multiple-tuple
multi-instance



2 relations
edge / vertex
graphs & networks



multi-table
multiple-tuple
relational

Attribute-Value

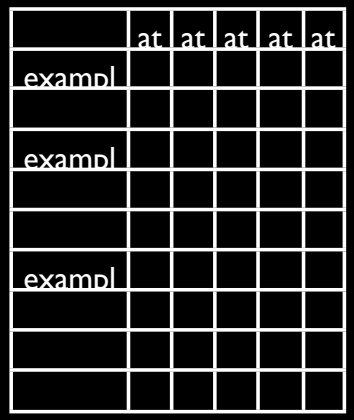
	at	*at*	*at*	*att*	*at*
example					
example					
example					
example					
example					
example					
example					
example					
example					

single-table
single-tuple
attribute-value

Traditional Setting in Machine Learning
(cf. standard tools like Weka)

Multi-Instance

[Dietterich et al. AIJ 96]



single-table
multiple-tuple
multi-instance

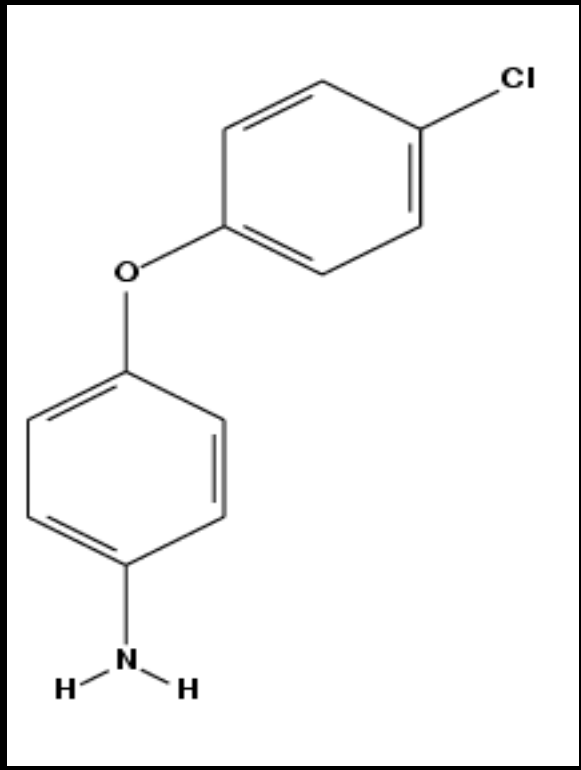
An example is positive if
there exists a tuple in the example
that satisfies particular properties

Boundary case between relational
and propositional learning.

A lot of interest in past 10 years

Applications: vision, chemo-informatics, ...

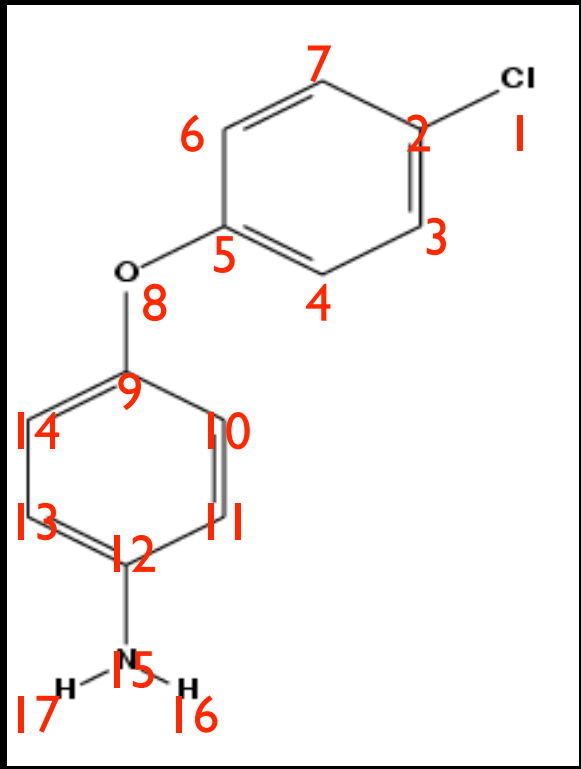
Encoding Graphs



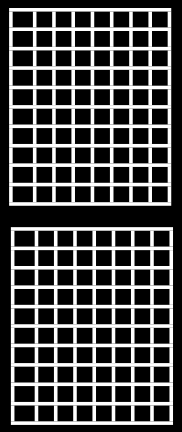
Encoding Graphs

atom(1,cl).
atom(2,c).
atom(3,c).
atom(4,c).
atom(5,c).
atom(6,c).
atom(7,c).
atom(8,o).
...

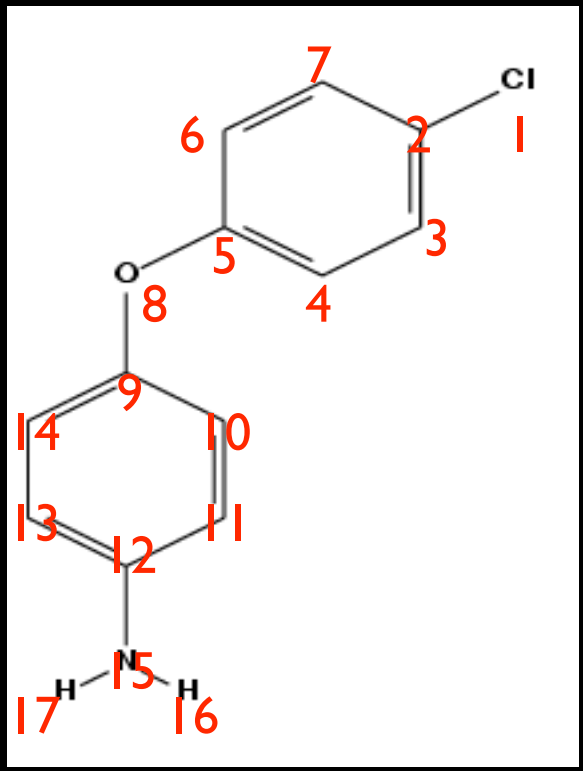
bond(3,4,s).
bond(1,2,s).
bond(2,3,d).
...



Encoding Graphs



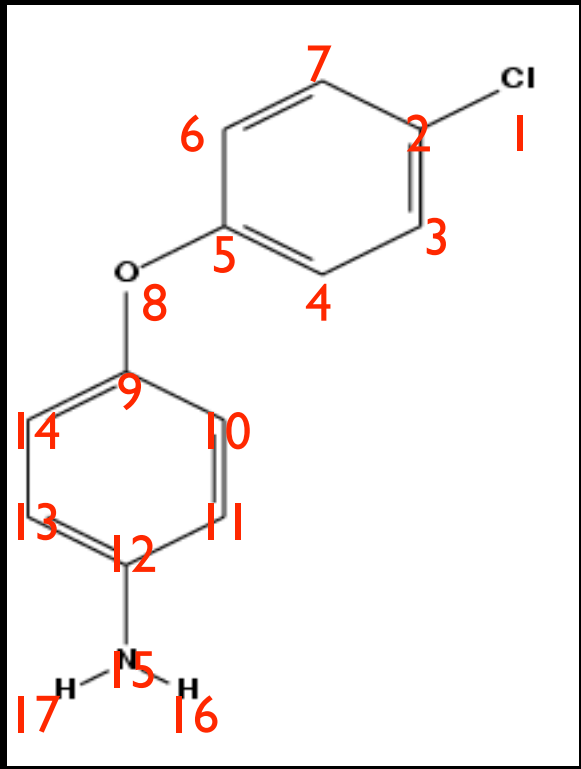
2 relations
edge / vertex
graphs & networks



Encoding Graphs

atom(1,cl,21,0.297)
atom(2,c,21, 0187)
atom(3,c,21,-0.143)
atom(4,c,21,-0.143)
atom(5,c,21,-0.143)
atom(6,c,21,-0.143)
atom(7,c,21,-0.143)
atom(8,o,52,0.98)
...

bond(3,4,s).
bond(1,2,s).
bond(2,3,d).
...



Note: add identifier for molecule

Encoding Knowledge

Use background knowledge in form of rules

- encode hierarchies

  halogen(A):- atom(X,f)

  halogen(A):- atom(X,cl)

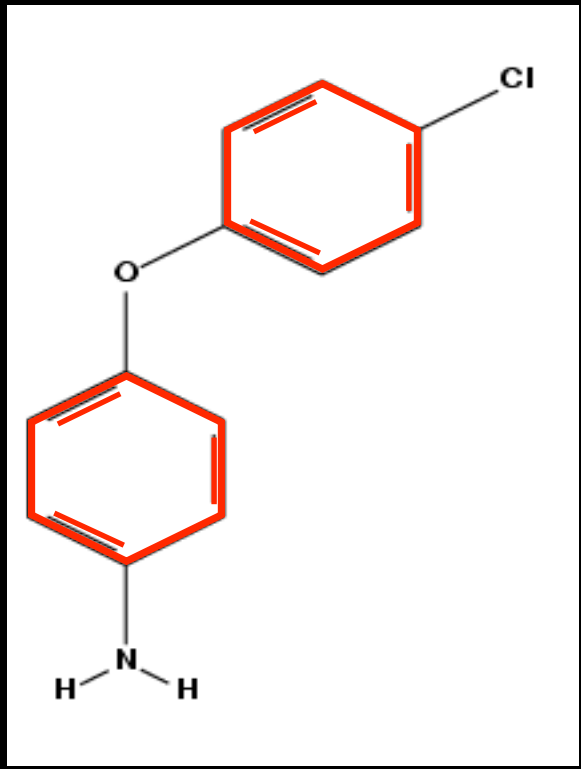
  halogen(A):- atom(X,br)

  halogen(A):- atom(X,i)

  halogen(A):- atom(X,as)

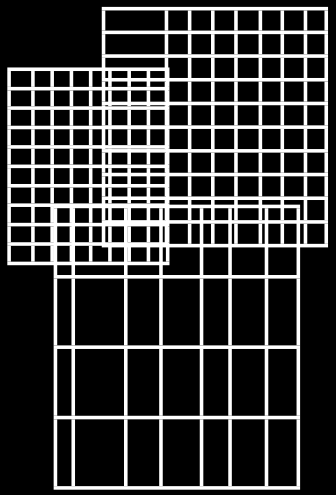
- encode functional group

  benzene-ring :- ...

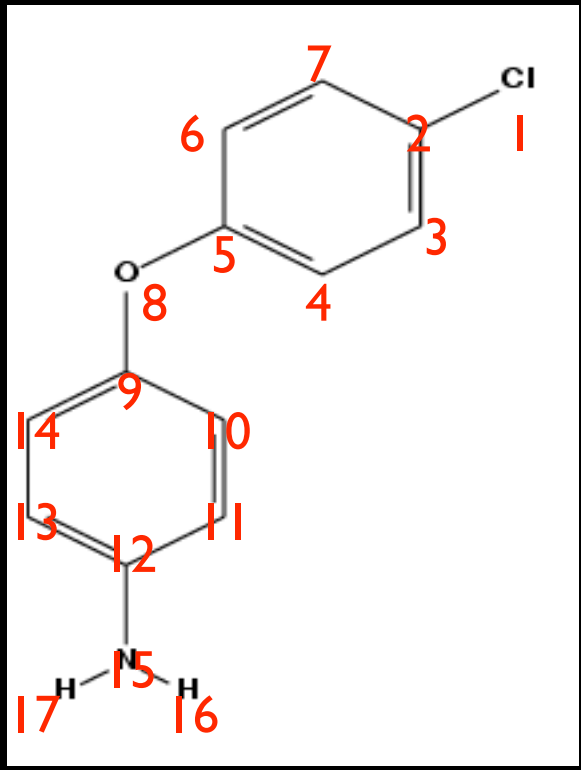


intentional versus extentional encodings

Relational Representation



multi-table
multiple-tuple
relational



Relational versus Graphs

Advantages Relational

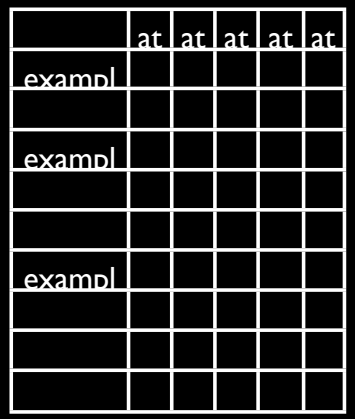
- background knowledge in the form of rules, ontologies, features, ...
- relations of arity > 2 (but hypergraphs)
- graphs capture structure but annotations with many features/labels is non-trivial

Advantages Graphs

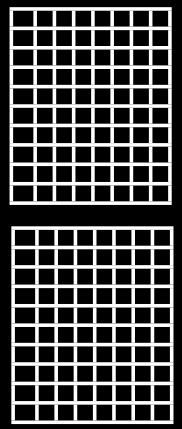
- efficiency and scalability
- full relational is more complex
- matrix operations

The Hierarchy

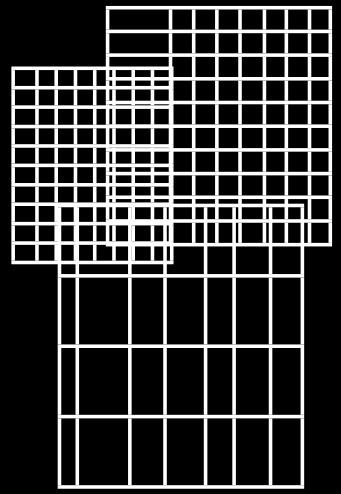
single-table
single-tuple
attribute-value



single-table
multiple-tuple
multi-instance



2 relations
edge / vertex
graphs & networks



multi-table
multiple-tuple
relational

Two questions

UPGRADING : Can we develop systems that work with richer representations (starting from systems for simpler representations)?

PROPOSITIONALISATION: Can we change the representation from richer representations to simpler ones ? (So we can use systems working with simpler representations)

B. Logic + Learning

B. Logic + Learning

B.1. Problem Settings

Traditional *predictive* ILP Problem

Given

- a set of positive and negative examples **(Pos, Neg)**
- a
- a
- a
- a covers relation on Le x Lh

Concept-learning in a

logical/relational representation

Find

- A hypothesis **h** over **Lh** that covers all positive **Pos** and no negative **Neg** examples taking **B** into account

Three possible settings

- Learning from entailment
  - covers(H,e) iff H |= e
- Learning from interpretations
  - covers(H,e) iff e is a model for H
- Learning from proofs or traces.
  - covers(H,e) iff e is proof given H

Three possible settings

- Learning from entailment
  - covers(H,e) iff H |= e
  - examples are facts / clauses

The Mutagenicity dataset

Background theory

```
molecule(225).
logmutag(225,0.64).
lumo(225,-1.785).
logp(225,1.01).
nitro(225,[f1_4,f1_8,f1_10,f1_9]).
atom(225,f1_1,c,21,0.187).
atom(
atom
atom
atom
...
ring
hete
```

```
bond(225,f1_1
bond(225,f
bond(2
b
_5,7).
5,f1_1,7).
f1_8,f1_9,2).
).
).
).
).
```

hypo

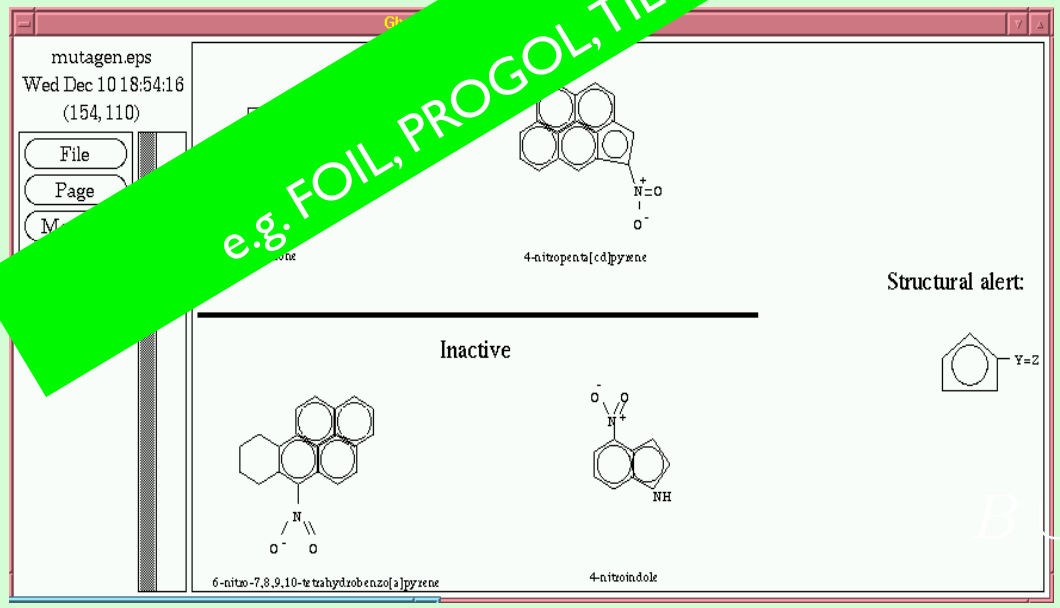
```
mut
```

Exa

mut

Applications

Systems:
e.g. FOIL, PROGOL, TILDE, ALEPH, GOLEM, ...



$B \cup H \models e$

Three possible settings

- Learning from interpretations
  - covers(H,e) iff e is a model for H
  - examples are Herbrand interpretations / state descriptions / possible worlds

Learning from interpretations

Examples

- Positive: { human(luc), human(lieve), male(luc), female(lieve)}
- Negative: { bat(dracula), male(dracula), vampire(dracula)}

Hypothesis

- human(X) :- male(X)

Learning from interpretations

Examples

- Positive: { human(luc), human(lieve), mal… …e(lieve)}

Hypothesis (positives or…

…ications

…integrity constraints /

freq…t patterns in relational databases

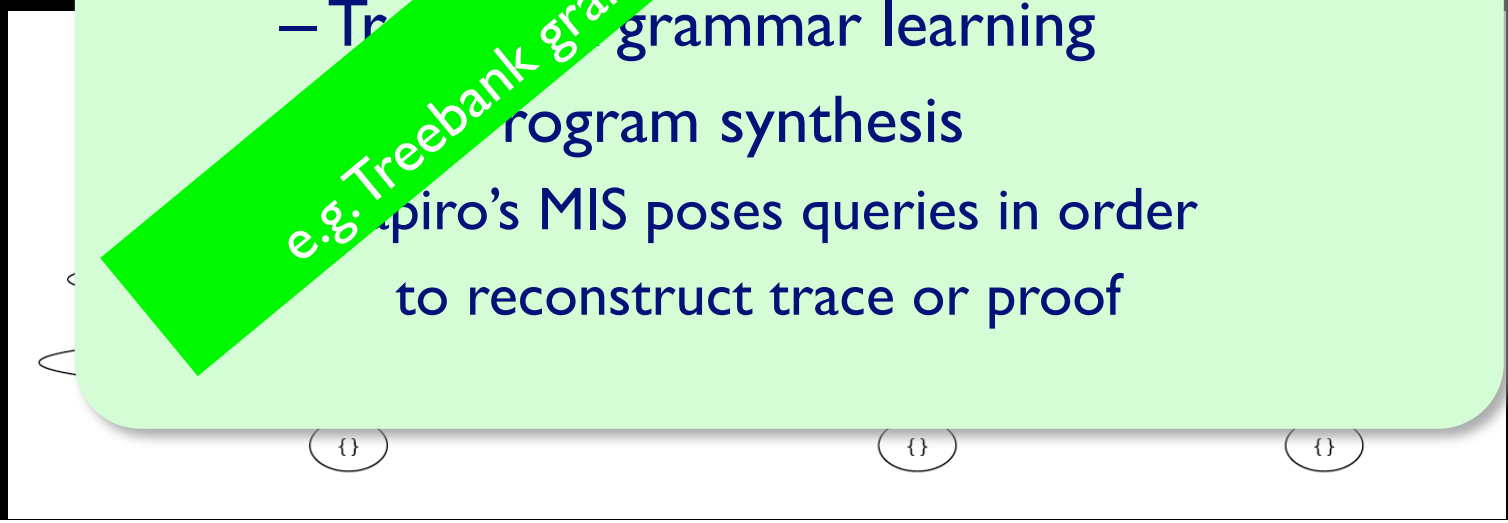
Systems:
e.g. LOGAN-H, Claudien, ICL, Warmr, …

Three possible settings

- Learning from proofs or traces.
  - covers(H,e) iff e is proof given H

An example

```
sentence(A, B) :- noun_phrase(C, A, D), verb_phrase(C, D, B).
noun_phrase(A, B, C) :- article(A, B, D), no        D, C).
verb_phrase(A, B, C) :- intransitive_verb(
article(singular, A, B) :- terminal(A,
article(singular, A, B) :- terminal(
article(plural, A, B)   :- termina        B).
noun(singular, A, B) :- termina        , B).
noun(plural, A, B)   :- termi        cles, B).
intr
int
ter
```



A tions

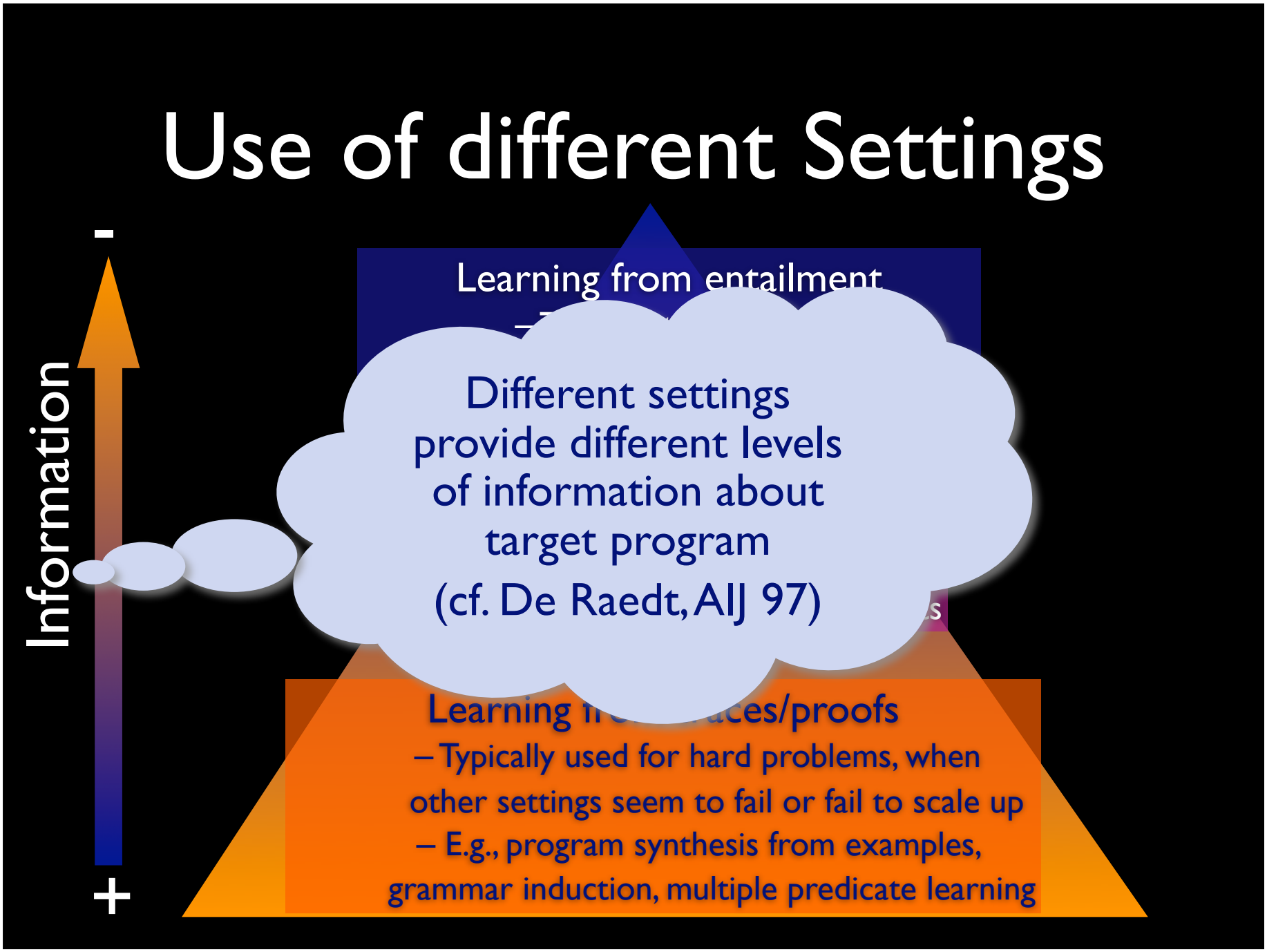
– Tr grammar learning

rogram synthesis

piro's MIS poses queries in order to reconstruct trace or proof

Systems:
e.g. Treebank grammars, Tracy, MIS, ...

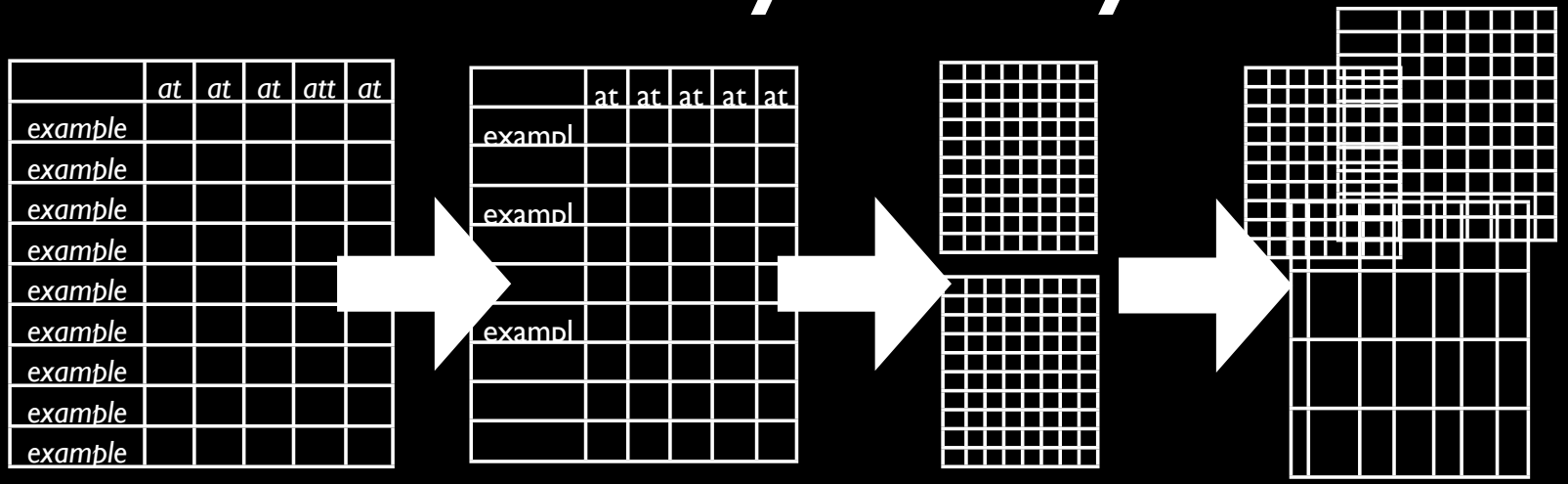
Use of different Settings



B. Logic + Learning

B.2. Systems & Methodology

Representational Hierarchy -- Systems



single-table
single-tuple
attribute-value

single-table
multiple-tuple
multi-instance

2 relations
edge / vertex
graphs & networks

multi-table
multiple-tuple
relational

UPGRADING

Two messages

Logical and Relational Learning applies essentially to any machine learning and data mining task, not just concept-learning

- distance based learning, clustering, descriptive learning, reinforcement learning, bayesian approaches

there is a recipe that is being used to derive new LRL algorithms on the basis of propositional ones

- not the only way to LRL

Learning Tasks

- rule-learning & decision trees [Quinlan 90], [Blockeel 96]
- frequent and local pattern mining [Dehaspe 98]
- distance-based learning (clustering & instance-based learning) [Horvath, 01], [Ramon 00]
- probabilistic modeling (cf. statistical relational learning)
- reinforcement learning [Dzeroski et al. 01]
- kernel and support vector methods

Logical and relational representations can (and have been) used for all learning tasks and techniques

The RECIPE

Start from well-known propositional learning system

Modify representation and operators

- e.g. generalization/specialization operator, similarity measure, …
- often use theta-subsumption as framework for generality

Build new system, retain as much as possible from propositional one

LRL Systems and techniques

FOIL ~ CN2 – Rule Learning (Quinlan MLJ 90)

Tilde ~ C4.5 – Decision Tree Learning (Blockeel & DR AIJ 98)

Warmr ~ Apriori – Association rule learning (Dehaspe 98)

Progol ~~ AQ – Rule learning (Muggleton NGC 95)

Graph miners ...

A case : FOIL

Learning from entailment -- the setting $B \cup H \models e$

Given

```
molecule(225).                                  bond(225,f1_1,f1_2,7).
logmutag(225,0.64).                             bond(225,f1_2,f1_3,7).
lumo(225,-1.785).                               bond(225,f1_3,f1_4,7).
logp(225,1.01).                                 bond(225,f1_4,f1_5,7).
nitro(225,[f1_4,f1_8,f1_10,f1_9]).              bond(225,f1_5,f1_1,7).
atom(225,f1_1,c,21,0.187).                      bond(225,f1_8,f1_9,2).
atom(225,f1_2,c,21,-0.143).                     bond(225,f1_8,f1_10,2).
atom(225,f1_3,c,21,-0.143).                     bond(225,f1_1,f1_11,1).
atom(225,f1_4,c,21,-0.013).                     bond(225,f1_11,f1_12,2).
atom(225,f1_5,o,52,-0.043).                     bond(225,f1_11,f1_13,1).
...
ring_size_5(225,[f1_5,f1_1,f1_2,f1_3,f1_4]).
hetero_aromatic_5_ring(225,[f1_5,f1_1,f1_2,f1_3,f1_4]).
```

background

mutagenic(225), ... examples $B \cup H \models e$

Find

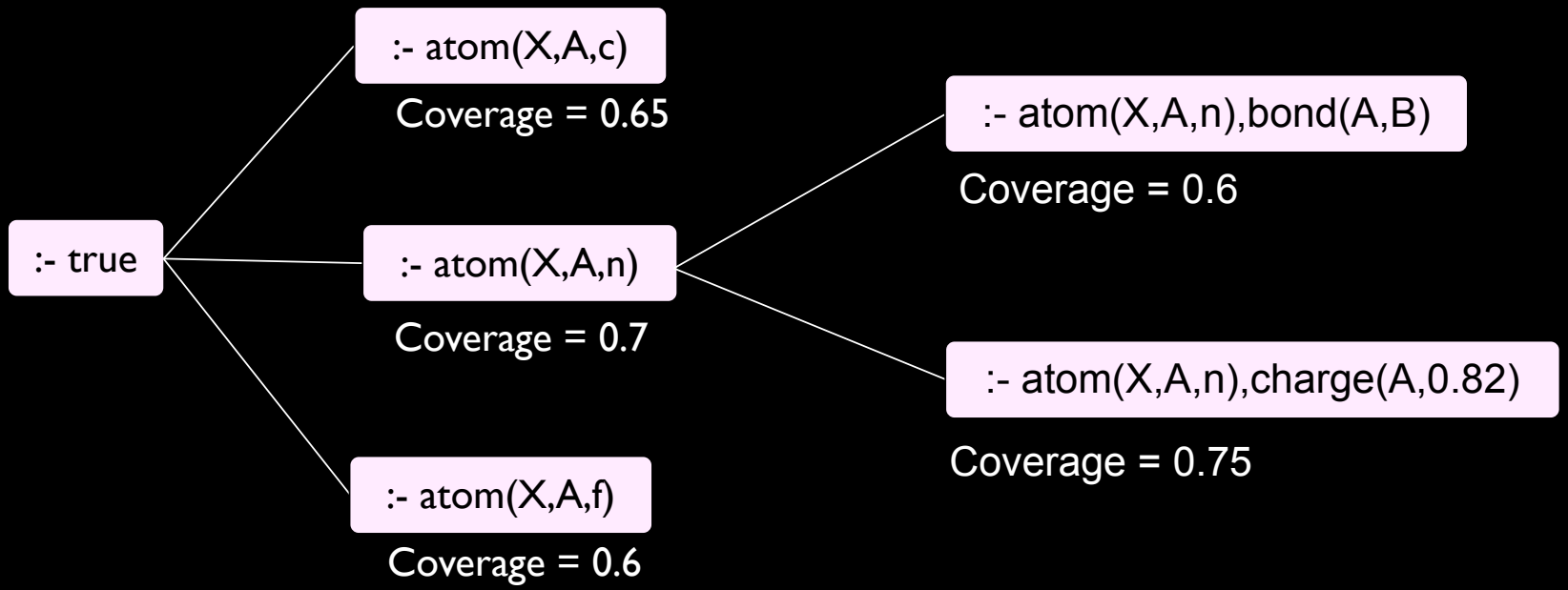
```
mutagenic(M) :- nitro(M,R1), logp(M,C), C > 1 .
```

rules

Searching for a rule

Greedy separate-and-conquer for rule set

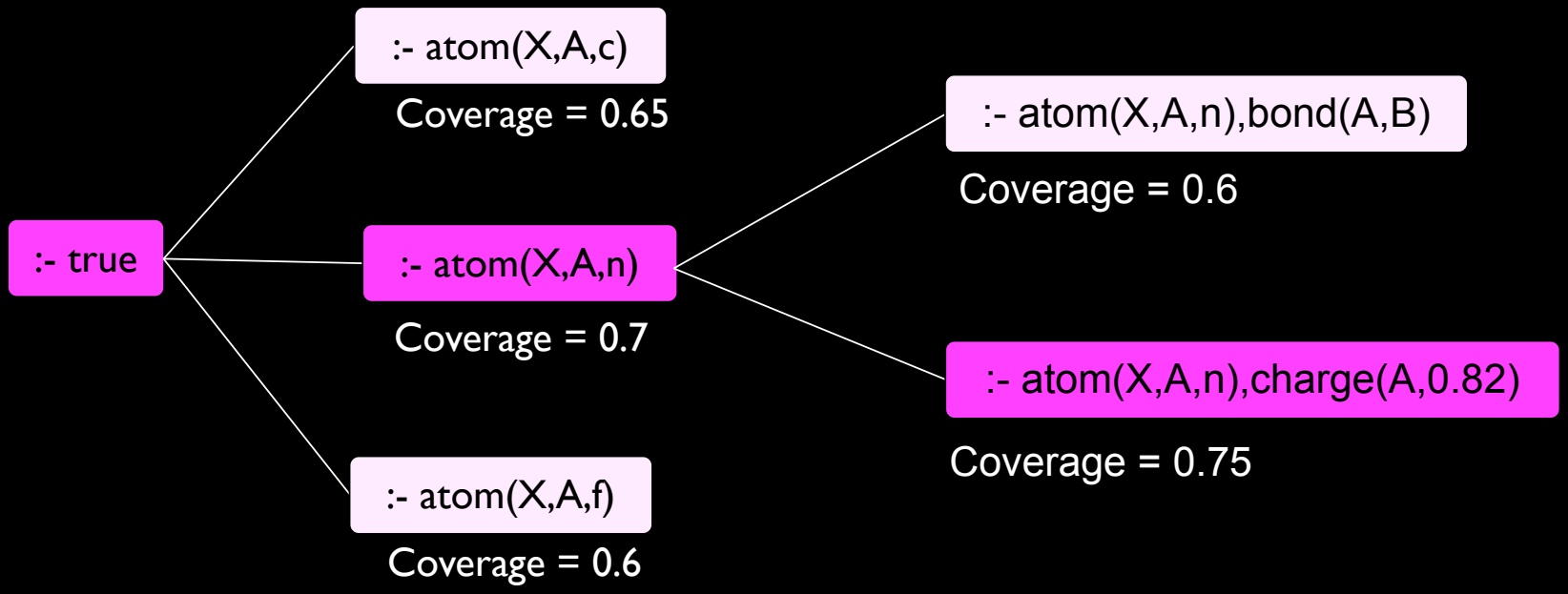
Greedy general-to-specific search for single rule



Searching for a rule

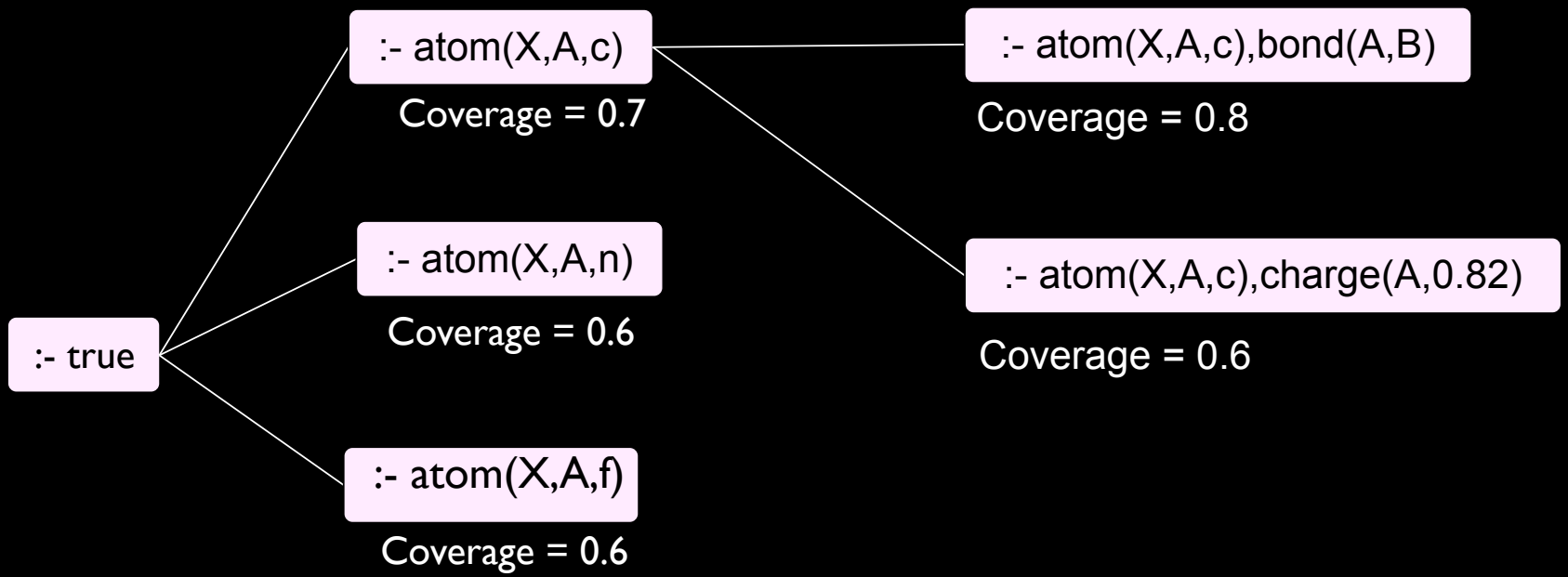
Greedy separate-and-conquer for rule set

Greedy general-to-specific search for single rule



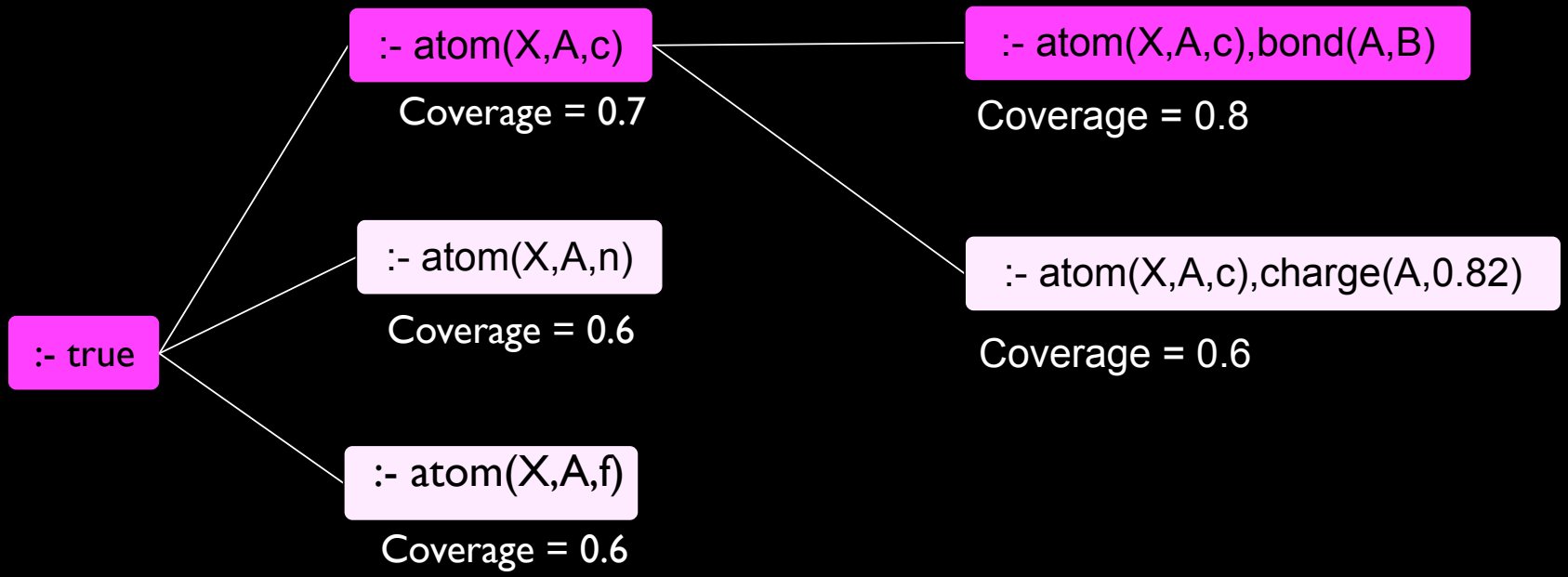
mutagenic(X) :- atom(X,A,n),charge(A,0.82)

FOIL



mutagenic(X) :- atom(X,A,n),charge(A,0.82)

FOIL



mutagenic(X) :- atom(X,A,c),bond(A,B)

mutagenic(X) :- atom(X,A,n),charge(A,0.82)

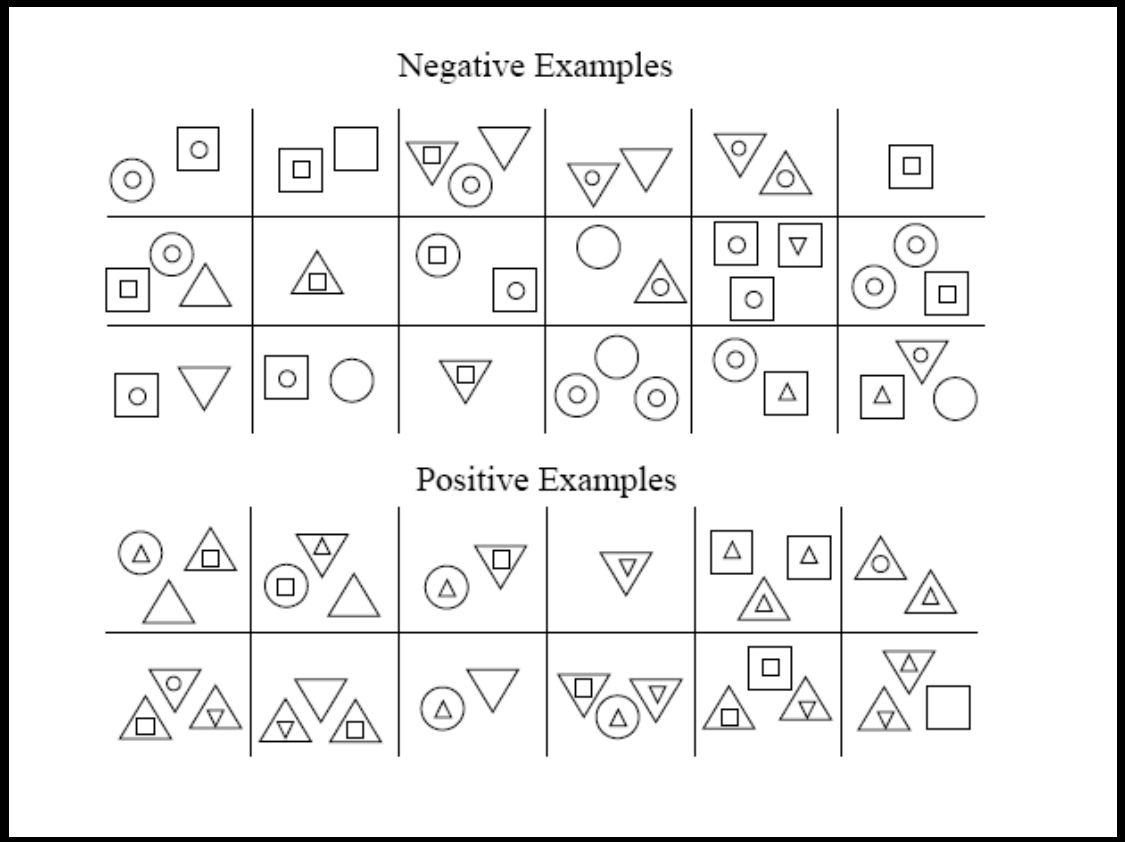
mutagenic(X) :- atom(X,A,c),charge(A,0.45)

mutagenic(X) :- atom(X,A,c),bond(A,B)

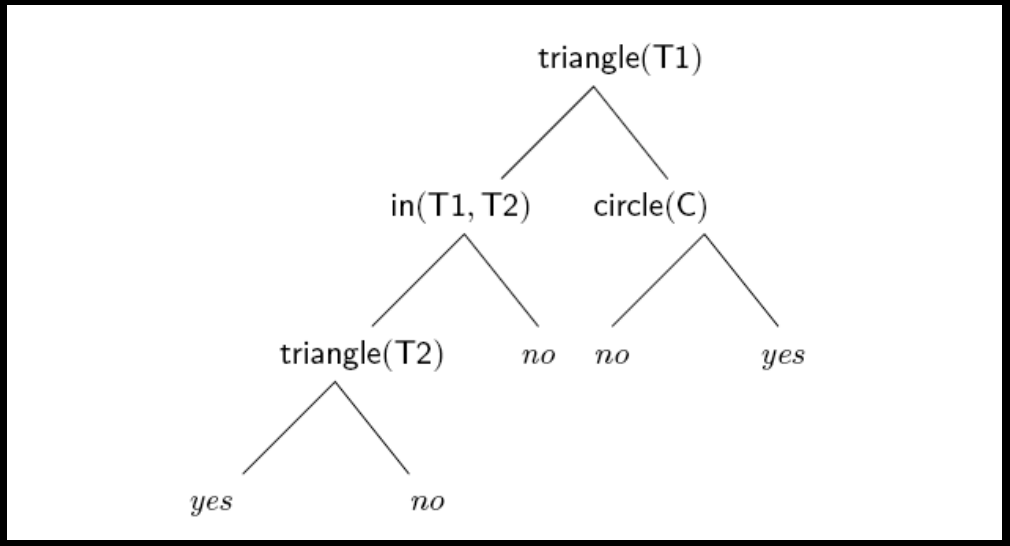
mutagenic(X) :- atom(X,A,n),charge(A,0.82)

Tilde

Logical Decision Trees (Blockeel & De Raedt AIJ 98)



A logical decision tree

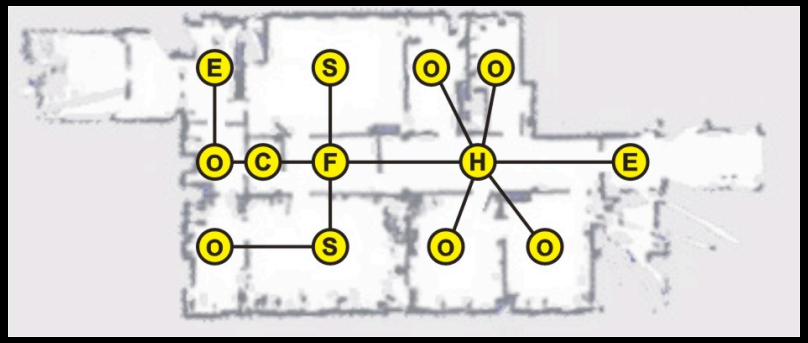


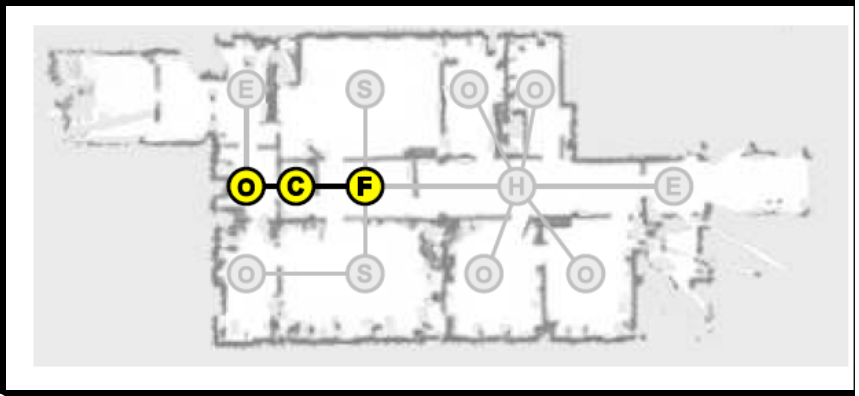
IF triangle(T1), in(T1, T2), triangle(T2) THEN Class = yes
ELSIF triangle(T1), in(T1, T2) THEN Class = no
ELSIF triangle(T1) THEN Class = no
ELSIF circle(C) THEN Class = no
ELSE Class = yes

Search in unknown environment

Popular Algorithms

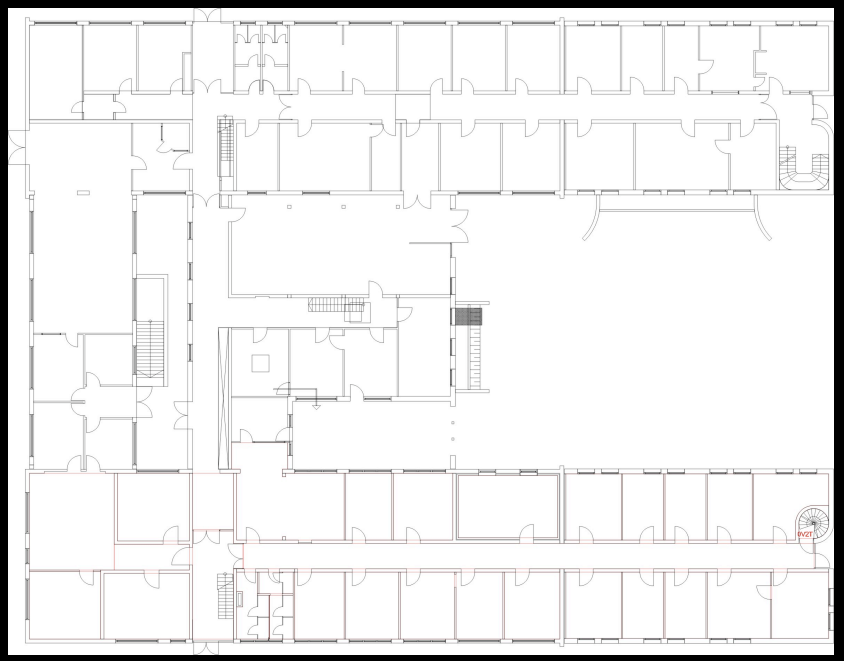
- Uninformed LRTA* [Koenig]
- Depth-first

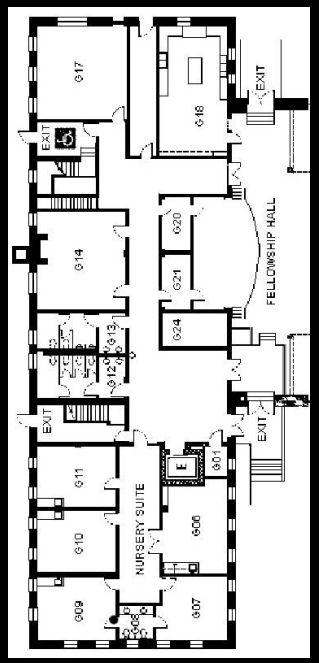




Kersting et al. Advanced Rob. 03

Buildings




G17
G18
EXIT
EXIT
G20
G21
G14
FELLOWSHIP HALL
G24
G13
G12
EXIT
EXIT
G11
G01
NURSERY SUITE
G10
G06
G09
G08
G07

Relational MDPs

Abstract states:

$$\text{in}(X), \text{conn}(X,Y), \text{room}(X), \text{hPassage}(Y)$$

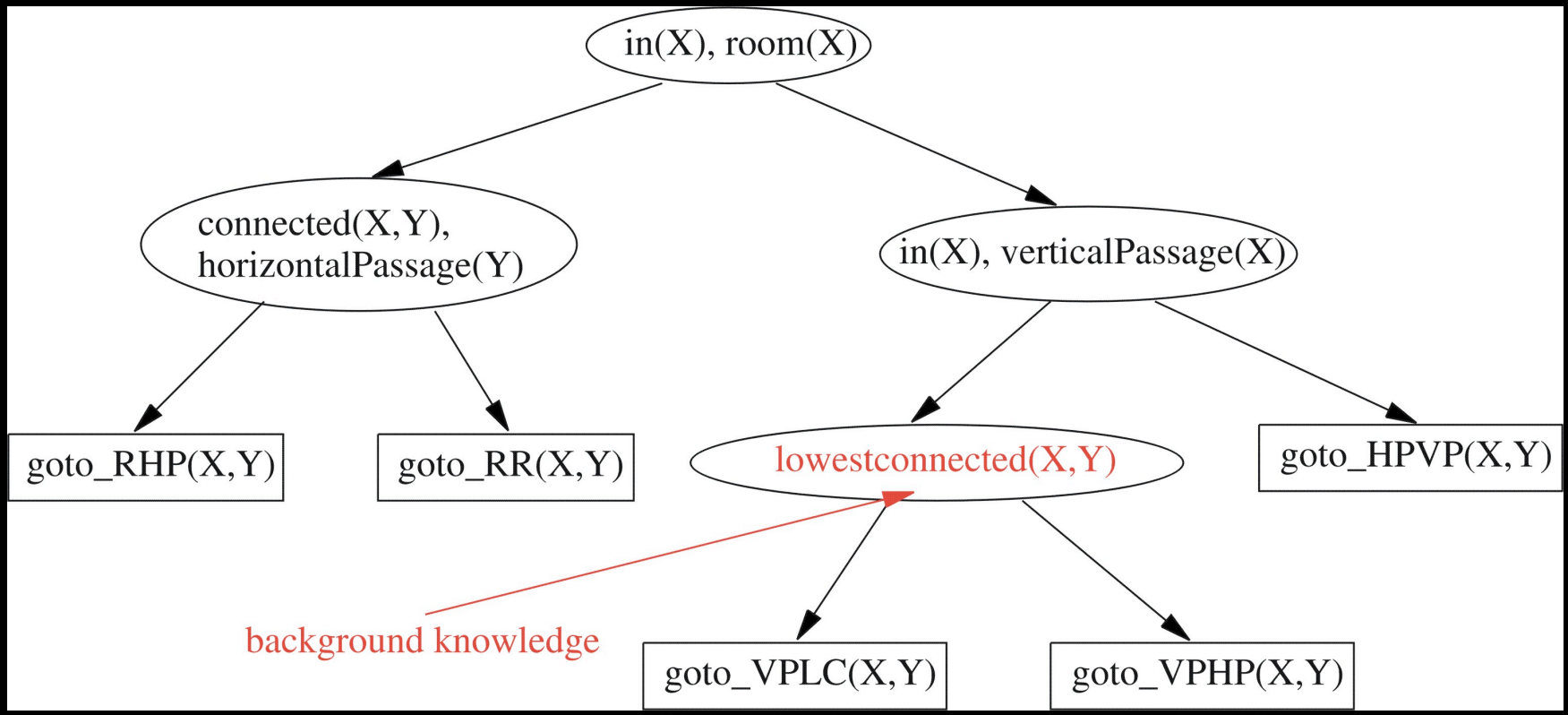
Abstract actions:

$$\text{in}(Y), \text{conn}(X,Y), \text{hPassage}(Y), \text{room}(X), X \neq Y \xleftarrow{p : \text{goto\_RHP}(X,Y)} \text{in}(X), \text{conn}(X,Y), \text{hPassage}(Y), \text{room}(X), X \neq Y$$

Abstract reward model:

$$10 \neg \text{in}(l_1)$$
$$0 \neg \text{true}$$

Policy



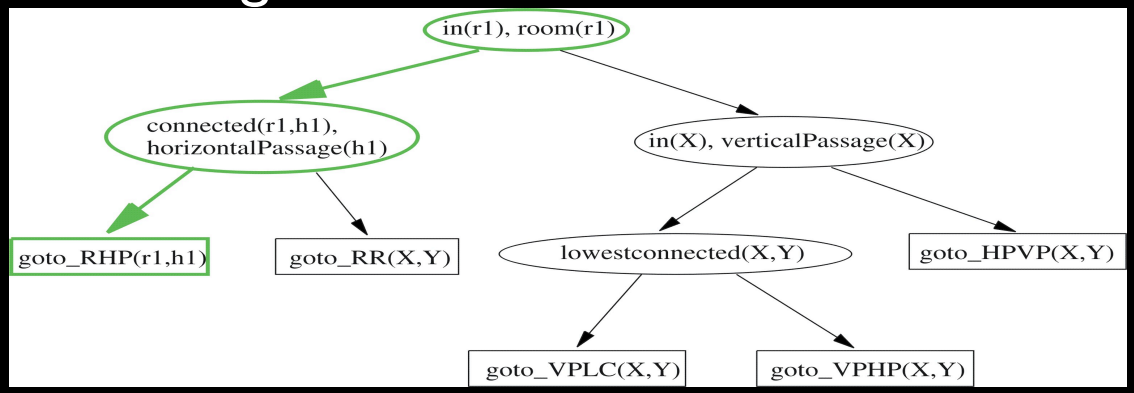
logical decision trees [Blockeel & De Raedt AIJ 98]

Using a Relational Polichy

1. Start in State

$$\text{in}(r_1), \underbrace{\text{conn}(r_1, h_1), \text{conn}(h_1, s_1)}_{\text{building description}}$$

2. Evaluate logical decision tree



3. Execute Action
4. Observe New State
5. Repeat if goal not yet reached.

WARMR

PARTICIPANT Table

NAME	JOB	COMPANY	PARTY	R_NUMBER
adams	researcher	scuf	no	23
blake	president	jvt	yes	5
king	manager	ucro	no	78
miller	manager	jvt	yes	14
scott	researcher	scuf	yes	94
turner	researcher	ucro	no	81

SUBSCRIPTION Table

NAME	COURSE
adams	erm
adams	so2
adams	srw
blake	cso
blake	erm
king	cso
king	erm
king	so2
king	srw
miller	so2
scott	erm
scott	srw
turner	so2
turner	srw

COMPANY Table

COMPANY	TYPE
jvt	commercial
scuf	university
ucro	university

COURSE Table

COURSE	LENGTH	TYPE
cso	2	introductory
erm	3	introductory
so2	4	introductory
srw	3	advanced

What to count ? Keys.

PARTICIPANT Table

NAME	JOB	COMPANY	PARTY	R_NUMBER
adams	researcher	scuf	no	23
blake	president	jvt	yes	5
king	manager	ucro	no	78
miller	manager	jvt	yes	14
scott	researcher	scuf	yes	94
turner	researcher	ucro	no	81

SUBSCRIPTION Table

NAME	COURSE
adams	erm
adams	so2
adams	srw
blake	cso
blake	erm
king	cso
king	erm
king	so2
king	srw
miller	so2
scott	erm
scott	srw
turner	so2
turner	srw

COMPANY Table

COMPANY	TYPE
jvt	commercial
scuf	university
ucro	university

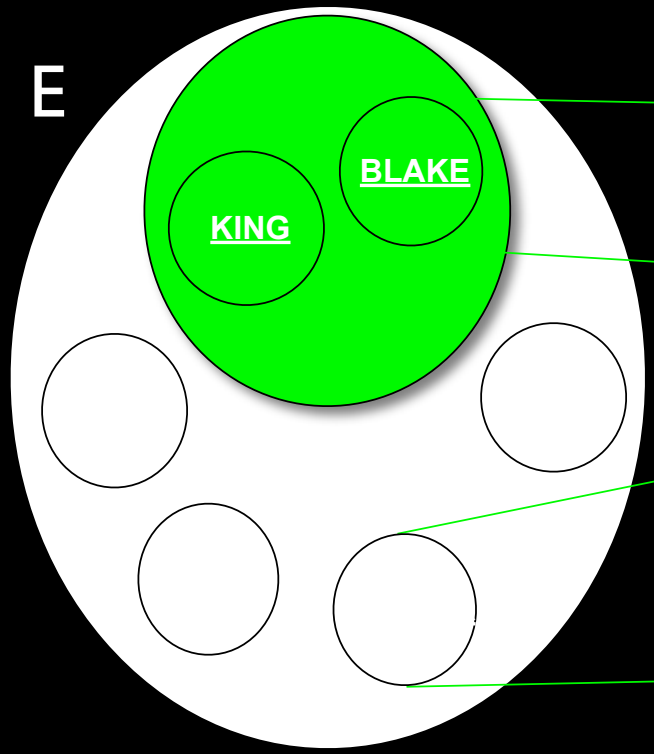
COURSE Table

COURSE	LENGTH	TYPE
cso	2	introductory
erm	3	introductory
so2	4	introductory
srw	3	advanced

Coverage

H: *“there is a subscription for a course of length less than 3”*

?- subscription(K,C), course(K,C,L,_), (L < 3).



participant(blake,president,jvt,yes,5).
subscription(blake,cso).
subscription(blake,erm).

course(cso,2,introductory).
course(erm,3,introductory).
course(so2,4,introductory).
course(srw,3,advanced).

K=blake
C = cso
L = 2
Yes

participant(turner,researcher,scuf,no,23).
subscription(turner,erm).
subscription(turner,so2).
subscription(turner,srw).

K=turner
No

Descriptive Data Mining

multi-relational database

association rules over multiple relations

PARTICIPANT Table

NAME	JOB	COMPANY	PARTY	R_NUMBER
adams	researcher	scuf	no	23
blake	president	jvt	yes	5
king	manager	ucro	no	78
miller	manager	jvt	yes	14
scott	researcher	scuf	yes	94
turner	researcher	ucro	no	81

SUBSCRIPTION Table

NAME	COURSE
adams	erm
adams	so2
adams	srw
blake	cso
blake	erm
king	cso
king	erm
king	so2
king	srw
miller	so2
scott	erm
scott	srw
turner	so2
turner	srw

COMPANY Table

COMPANY	TYPE
jvt	commercial
scuf	university
ucro	university

COURSE Table

COURSE	LENGTH	TYPE
cso	2	introductory
erm	3	introductory
so2	4	introductory
srw	3	advanced

Selected, Preprocessed, and Transformed Data

"**IF** ?- participant(P,C,PA,X), course(P,Y,advanced) **THEN** ?-PA = no.

support: 20 %
confidence: 75 %

"**IF** participant follows an advanced course **THEN** she skips the welcome party"

support: 20 %
confidence: 75 %

Warmr

First order association rule :

- IF Query1 THEN Query2
- Shorthand for
- IF Query1 THEN Query1 and Query2
- to obtain variable bindings

IF ?- participant(P,C,PA,X), course(P,Y,advanced) THEN PA=no

IF ?- participant(P,C,PA,X), course(P,Y,advanced) succeeds for P
THEN ?- participant(P,C,PA,X), course(P,Y,advanced), PA=no

Warmr ~ Apriori

Works as Apriori :

- keeping track of frequent and infrequent queries
- order queries by theta-subsumption
- using special mechanism (bias) to declare type of queries searched for

Generalizes many of the specific variants of Apriori : item-sets, episodes, hierarchies, intervals, ...

Upgrading

Key ideas / contributions FOIL + Tilde + Warmr

- determine the representation of examples and hypotheses
- select the right type of coverage and generality (subsumption)
- keep existing algorithm (CN2 or C4.5) but replace operators (often theta-subsumption)
- keep search strategy
- fast implementation

Distance Based Learning

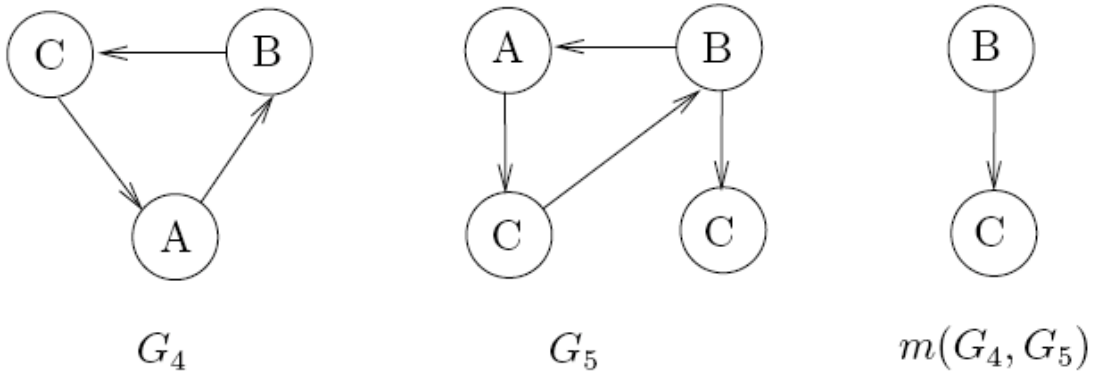


Fig. 9.8. The maximal common subgraph $m(G_4, G_5)$ of G_4 and G_5

$$d_{graph}(G_1, G_2) = |G_1| + |G_2| - 2|mcs(G_1, G_2)|$$

Distances

| h | = the size of a hypothesis h

| g | ≤ | s | if g is more general than s

mgg(t,u) = set of minimally general generalizations

| mgg(t,u) | = max size of element in mgg(t,u)

under mild conditions the following is a metric

$$d_{\preceq}(x, y) = |x| + |y| - 2|mgg(x, y)|$$

The RECIPE

Relevant for ALL levels of the hierarchy

Still being applied across data mining,

- mining from graphs, trees, and sequences

Works in both directions

- upgrading and downgrading !!!

Mining from graphs or trees as downgraded Relational Learning

Many of the same problems / solutions apply to graphs as to relational representations

From Upgrading to Downgrading

Work at the right level of representation

- trade-off between expressivity & efficiency

The **old** challenge: **upgrade** learning techniques for simpler representations to richer ones.

The **new** challenge: **downgrade** more expressive ones to simpler ones for efficiency and scalability; e.g. graph miners.

Note: systems using rich representations form a **baseline**, and can be used to test out ideas.

Relevant also for ALL machine learning and data mining tasks

Learning Tasks

Logical and relational representations can (and have been) used for all learning tasks and techniques

- rule-learning & decision trees
- frequent and local pattern mining
- distance-based learning (clustering & instance-based learning)
- probabilistic modeling (cf. statistical relational learning)
- reinforcement learning
- kernel and support vector methods

C. Probabilistic Logic Learning

PLL: What Changes ?

- Clauses annotated with probability labels
  - E.g. in Sato's Prism, Muggleton's SLPs, Kersting and De Raedt's BLPs, …
- Prob. covers relation covers(e,H U B) = P(e | H,B)
  - Likelihood of example given background and hypothesis
  - Probability distribution P over the different values e can take; so far only (true,false)
  - impossible examples have probability 0
- Knowledge representation issue
  - Define probability distribution on examples / individuals
  - What are these examples / individuals ?

Probabilistic *generative* ILP Problem

Given

- a set of examples **E**
- a background theory **B**
- a language **Le** to represent examples
- a language **Lh** to represent hypotheses
- a probabilistic covers **P** relation on **Le x Lh**

Find

- hypothesis h* maximizing some score based on the probabilistic covers relation; often some kind of maximum likelihood

PLL: Three Settings

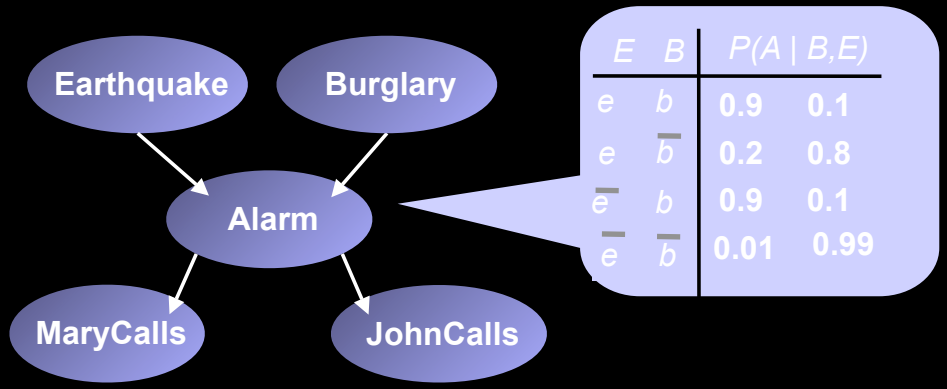
- Probabilistic learning from interpretations
  - Bayesian logic programs, Koller's PRMs, Domingos' MLNs, Vennekens' LPADs
- Probabilistic learning from entailment
  - Muggleton's Stochastic Logic Programs, Sato's Prism, Poole's ICL, De Raedt et al.'s ProbLog
- Probabilistic learning from proofs
  - Learning the structure of SLPs; a tree-bank grammar based approach, Anderson et al.'s RMMs, Kersting et al.

C. Probabilistic Logic Learning
C.1. Learning from Interpretations

Graphical Model

- Defines a joint probability distribution in a compact way; attractive from a computational perspective
- The structure encodes some conditional independency assumptions.
- Two types :
  - Directed -- Bayesian Network
  - Undirected -- Markov Network

Bayesian Networks



$$P(E,B,A,J,M) = P(E).P(B).P(A|E,B).P(J|A).P(M|A)$$

Assumption 1 *(cf. Russell and Norvig [2004]) Each node X_i in the graph is conditionally independent of any subset* $\mathbf{A}$ *of nodes that are* non-descendants *of X_i given a joint state of $\mathbf{Pa}(X_i)$, that is, $\mathbf{P}(X_i \mid \mathbf{A}, \mathbf{Pa}(X_i)) = \mathbf{P}(X_i \mid \mathbf{Pa}(X_i))$.*

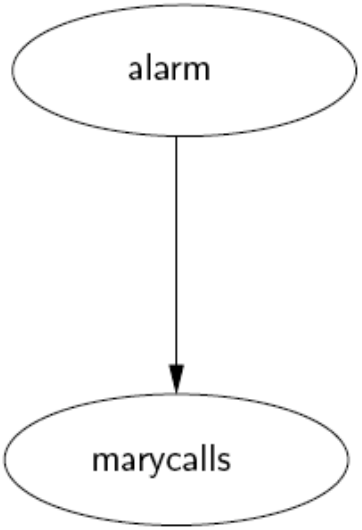
Using the Joint Prob. Distribution,
any query P(Q |evidence) can be answered

Learning

Two aspects:

- parameters, e.g. max likelihood
- structure learning

Toy Example



P(alarm)
$(\lambda_0, 1-\lambda_0)$

alarm	P(marycalls)
true	$(\lambda_1, 1-\lambda_1)$
false	$(\lambda_2, 1-\lambda_2)$

Fig. 8.6. A simple Bayesian network. $P(\mathsf{a}) = \lambda_0; P(\mathsf{m}|\mathsf{a}) = \lambda_1; P(\mathsf{m}|\neg\mathsf{a}) = \lambda_2$

Parameter Estimation

complete data set
simply counting

A1	A2	A3	A4	A5	A6	
true	true	false	true	false	false	X1
false	true	true	true	false	false	X2
...	...	...	...	...	...	⋮
true	false	false	false	true	true	XM

Parameter Estimation

A1	A2	A3 (hidden/ latent)	A4	A5	A6
true	true	?	true	false	false
?	true	?	?	false	false
...	...	...	...	...	...
true	false	?	false	true	?

incomplete data set

Real-world data: states of some random variables are missing

E.g. medical diagnose: not all patient are subjects to all test

Parameter reduction, e.g. clustering, ...

? missing value

On the Alarm Net

	alarm	marycalls
e_1	true	true
e_2	true	true
e_3	false	false
e_4	false	true
...		

Table 8.3. A completely observed data set.

	alarm	marycalls
e_1	true	true
e_2	true	?
e_3	false	false
e_4	?	true

Table 8.4. A data set with missing values.

maximize likelihood:
max *P(E | M)*

Derivation (fully obs)

$$P(\mathsf{a}, \neg\mathsf{m}) = \lambda_0(1-\lambda_2)$$

Generalize :

$$P(E|M) = \lambda_0^{|a|}.(1-\lambda_0)^{|\neg a|}.\lambda_1^{|a\wedge m|}(1-\lambda_1)^{|a\wedge\neg m|}.\lambda_2^{|\neg a\wedge m|}.(1-\lambda_2)^{|\neg a\wedge\neg m|}$$

Use log likelihood:

$$\begin{aligned} L = \log P(E|M) \quad = \quad & |a|.\log\lambda_0 + |\neg a|.\log(1-\lambda_0) + \\ & |a\wedge m|.\log\lambda_1 + |a\wedge\neg m|.\log(1-\lambda_1) + \\ & |\neg a\wedge m|.\log\lambda_2 + |\neg a\wedge\neg m|.\log(1-\lambda_2) \end{aligned}$$

Derivation (fully obs) (2)

$$\begin{aligned}
\frac{\partial L}{\partial \lambda_0} &= \frac{|a|}{\lambda_0} - \frac{|\neg a|}{1-\lambda_0} \\
\frac{\partial L}{\partial \lambda_1} &= \frac{|a \wedge m|}{\lambda_1} - \frac{|a \wedge \neg m|}{1-\lambda_1} \\
\frac{\partial L}{\partial \lambda_2} &= \frac{|\neg a \wedge m|}{\lambda_2} - \frac{|\neg a \wedge \neg m|}{1-\lambda_2}
\end{aligned}$$

Yielding as solutions

$$\lambda_0 = \frac{|a|}{|a| + |\neg a|} \tag{1}$$

$$\lambda_1 = \frac{|a \wedge m|}{|a \wedge m| + |a \wedge \neg m|} \tag{2}$$

$$\lambda_2 = \frac{|\neg a \wedge m|}{|\neg a \wedge m| + |\neg a \wedge \neg m|} \tag{3}$$

So, simply counting !

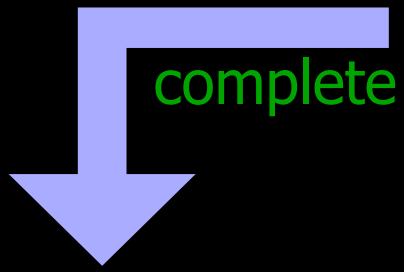
Parameter Estimation (part. obs)

Idea Expectation Maximization

Incomplete data ?

- Complete data (Imputation)
  - most probable?, average?, ... value
- Count
- Iterate

EM Idea: Complete the data



A	M
true	true
true	?
false	true
true	false
false	?

incomplete data

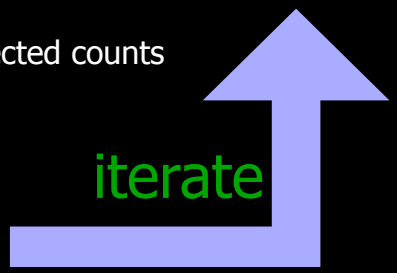
$$P(\mathsf{m}|\mathsf{a}) = 0.6$$

$$P(\mathsf{m}|\neg\mathsf{a}) = 0.2$$

complete data

expected counts

A	M	N
true	true	1.6
true	false	1.4
false	true	1.2
false	false	0.8



$$\lambda_1 = P(\mathsf{m}|\mathsf{a}) = \frac{1.6}{1.6 + 0.4} = 0.54$$

$$\lambda_2 = P(\mathsf{m}|\neg\mathsf{a}) = \frac{1.2}{1.2 + 0.8} = 0.6$$

Parameter Tying

$$P(E|M) = \lambda_0^{|a|}.(1-\lambda_0)^{|\neg a|}.\lambda_1^{|a\wedge m|}(1-\lambda_1)^{|a\wedge\neg m|}.\lambda_2^{|\neg a\wedge m|}.(1-\lambda_2)^{|\neg a\wedge\neg m|}$$
$$\lambda_3^{|a\wedge j|}.(1-\lambda_3)^{|a\wedge\neg j|}.\lambda_4^{|\neg a\wedge j|}(1-\lambda_4)^{|\neg a\wedge\neg j|}$$

Assume $\lambda_1 = \lambda_3$ and $\lambda_2 = \lambda_4$. The likelihood simplifies to

$$P(E|M) = \lambda_0^{|a|}.(1-\lambda_0)^{|\neg a|}.\lambda_1^{|a\wedge m|+|a\wedge j|}(1-\lambda_1)^{|a\wedge\neg m|+|a\wedge\neg j|}.$$
$$\lambda_2^{|\neg a\wedge m|+|\neg a\wedge j|}.(1-\lambda_2)^{|\neg a\wedge\neg m|+|\neg a\wedge\neg m|}$$

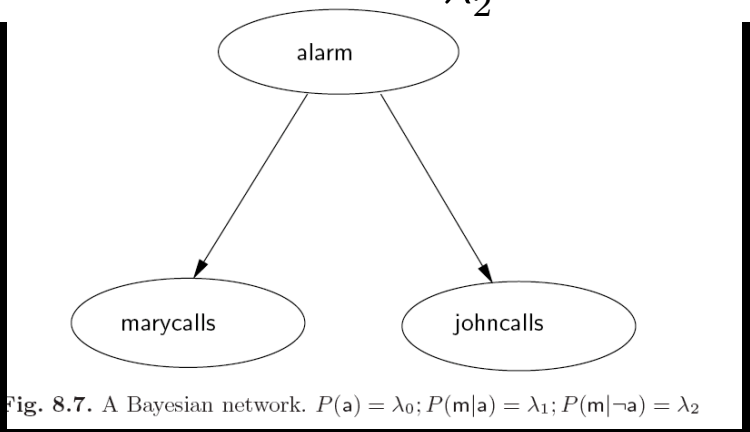


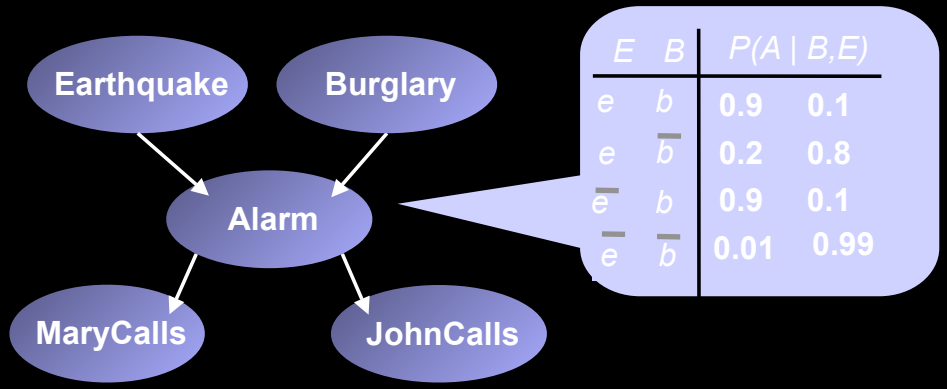
Fig. 8.7. A Bayesian network. $P(\mathsf{a}) = \lambda_0; P(\mathsf{m}|\mathsf{a}) = \lambda_1; P(\mathsf{m}|\neg\mathsf{a}) = \lambda_2$

Structure Learning

Of Bayesian networks

- one approach greedily searches through the space of possible networks:
- iterate
  - perform operation on graph (add, delete, reverse arcs)
  - estimate parameters
  - evaluate quality

Bayesian Networks

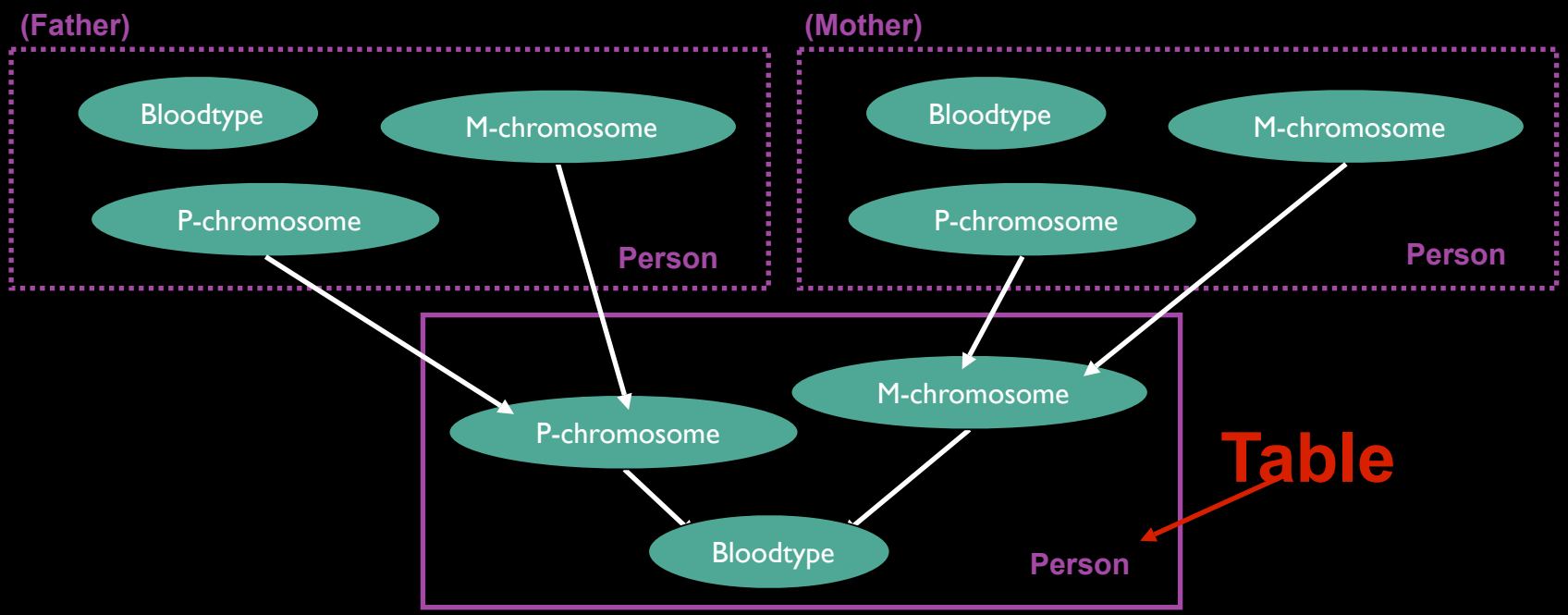


$$P(E,B,A,J,M) = P(E).P(B).P(A|E).\ P(A|B).P(J|A).P(M|A)$$

```
earthquake.
burglary.
alarm :- earthquake, burglary.
marycalls :- alarm.
johncalls:- alarm.
```

Probabilistic Relational Models (PRMs)

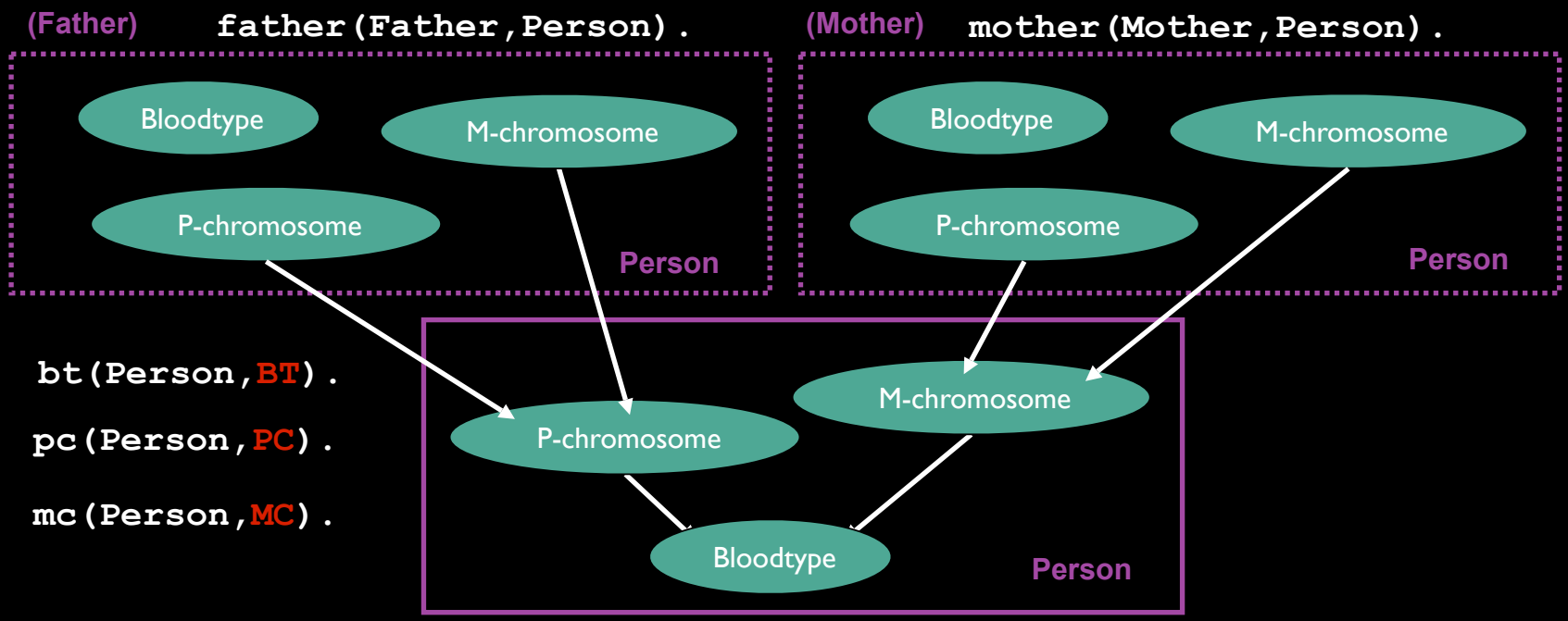
[Getoor,Koller, Pfeffer]



[Getoor,Koller, Pfeffer]

Probabilistic Relational Models (PRMs)

[Getoor,Koller, Pfeffer]



Dependencies (CPDs associated with):

```
bt(Person,BT) :- pc(Person,PC), mc(Person,MC).

pc(Person,PC) :- pc_father(Father,PCf), mc_father(Father,MCf).
```

View :

```
pc_father(Person,PCf) |  father(Father,Person),pc(Father,PC).
                                    ...
```

Probabilistic Relational Models (PRMs)
Bayesian Logic Programs (BLPs)

```
father(rex,fred).      mother(ann,fred).
father(brian,doro).    mother(utta, doro).
father(fred,henry).    mother(doro,henry).
```

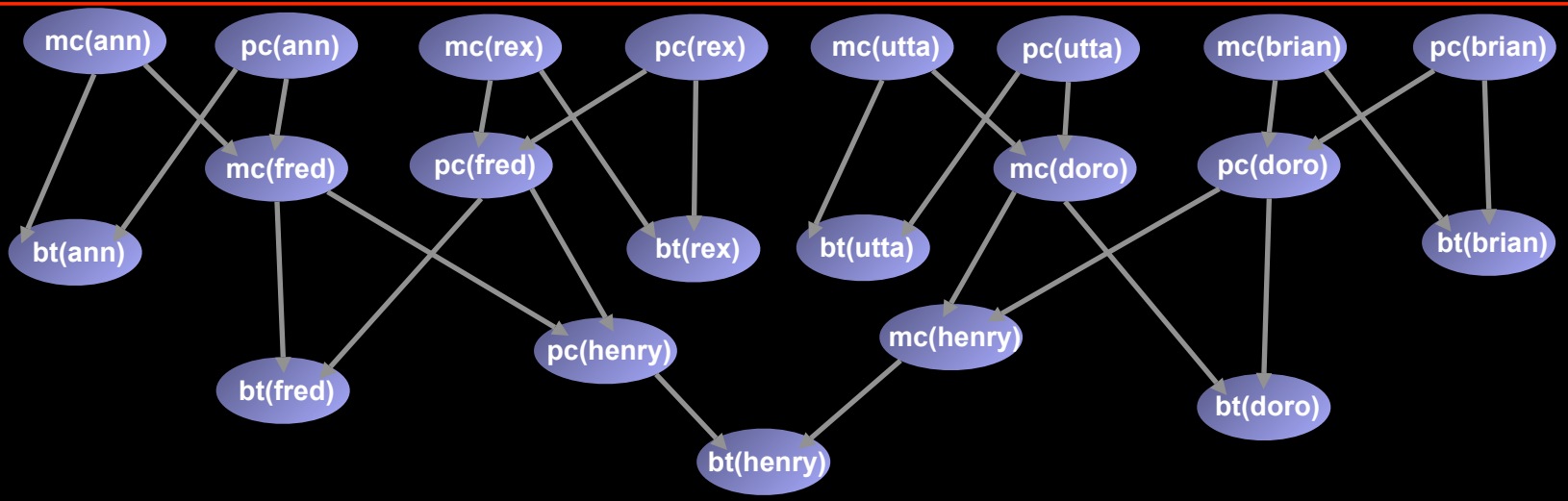
Extension

```
pc_father(Person,PCf) |  father(Father,Person),pc(Father,PC).
                                    ...
mc(Person,MC)  | pc_mother(Person,PCm), pc_mother(Person,MCm).
pc(Person,PC)  | pc_father(Person,PCf), mc_father(Person,MCf).
bt(Person,BT)  | pc(Person,PC), mc(Person,MC).
```

Intension

RV

State



Knowledge Based Model Construction

Extension + Intension =>Probabilistic Model

Advantages

same intension used for multiple extensions

parameters are being shared / tied together

unification is essential

learning becomes feasible

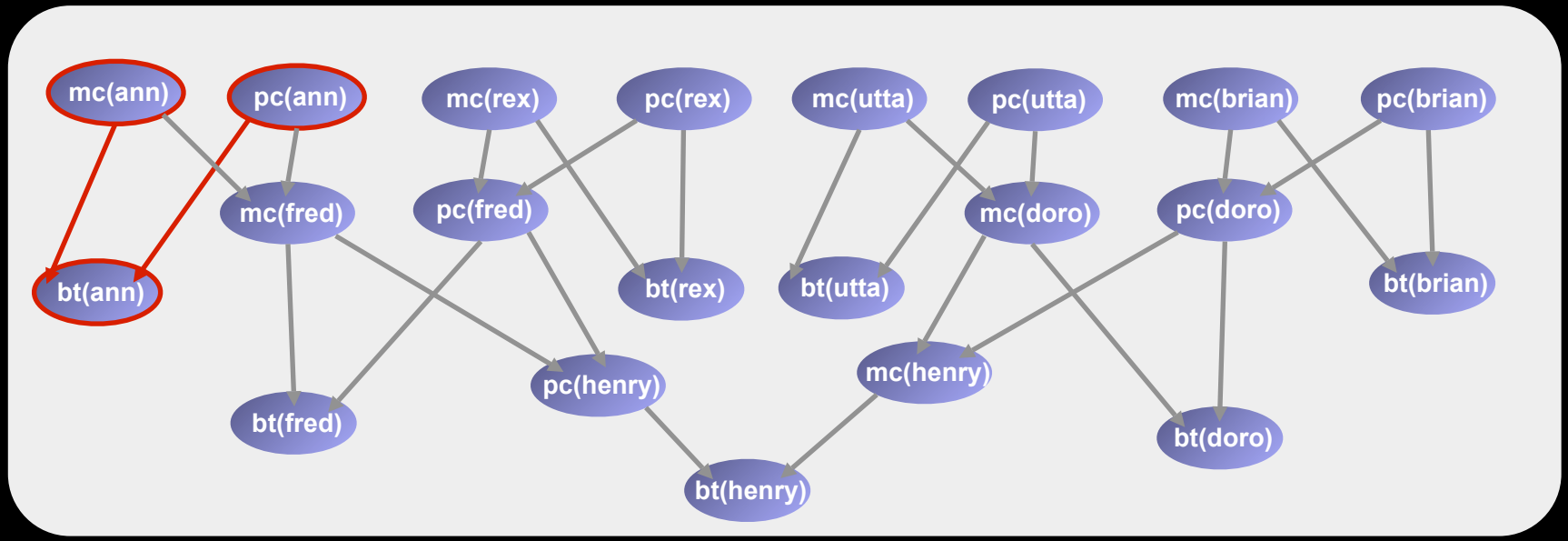
Typical use includes

prob. inference P(Q | E), P(bt(mary) | bt(john) =o-)

max. likelihood parameter estimation & structure learning

Answering Queries

P(`bt(ann)`**)?**

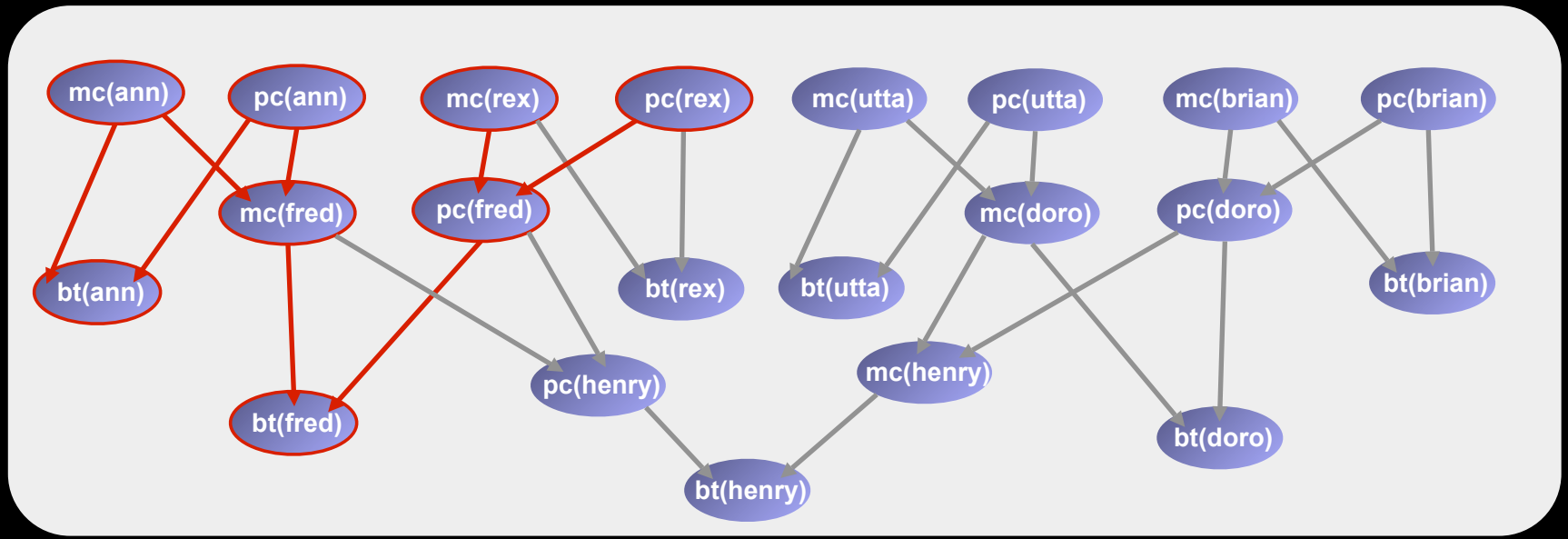


Answering Queries

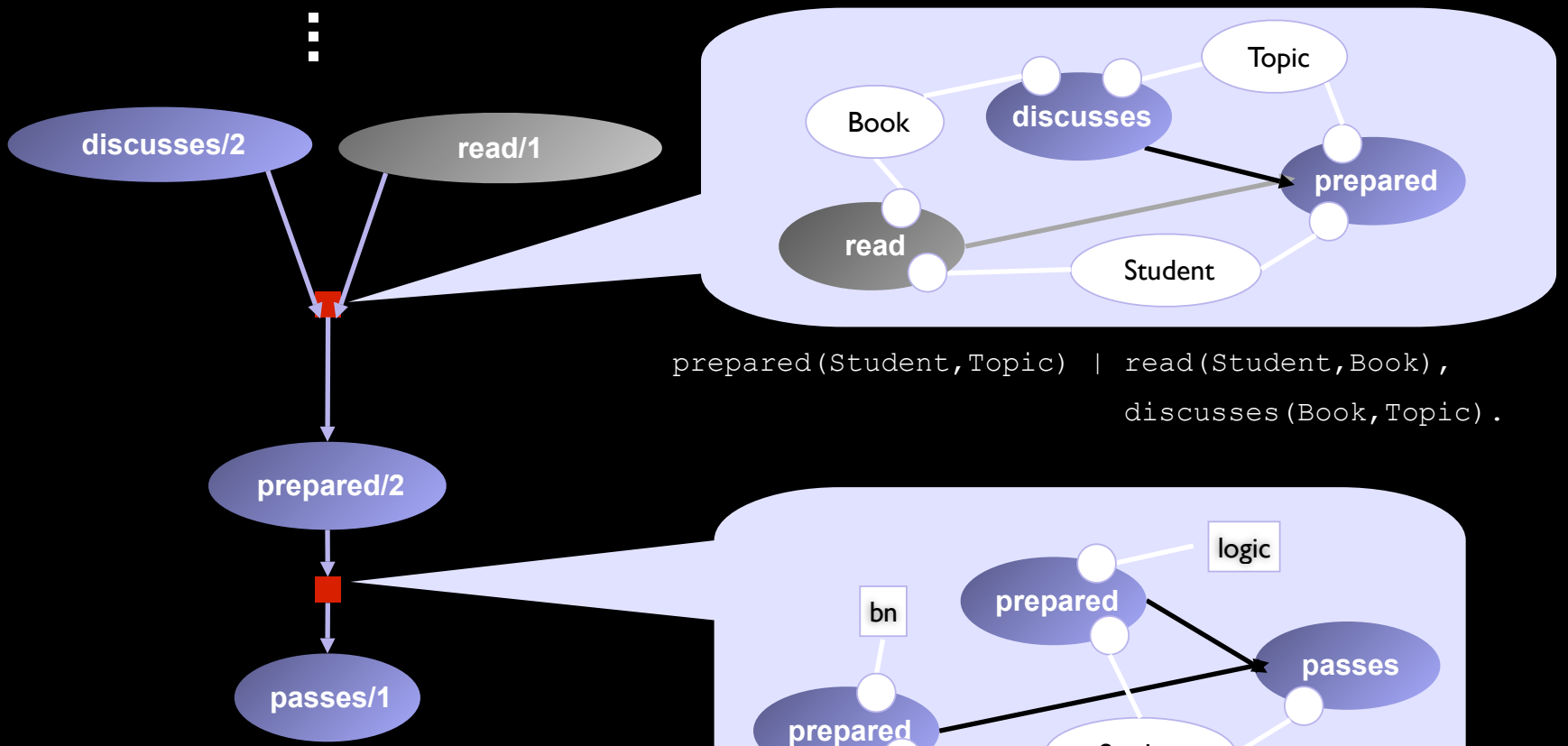
Bayes' rule

$$P(\texttt{bt(ann)} \mid \texttt{bt(fred)}) = \frac{P(\texttt{bt(ann)}, \texttt{bt(fred)})}{P(\texttt{bt(fred)})}$$

$P(\texttt{bt(ann)}, \texttt{bt(fred)})$?



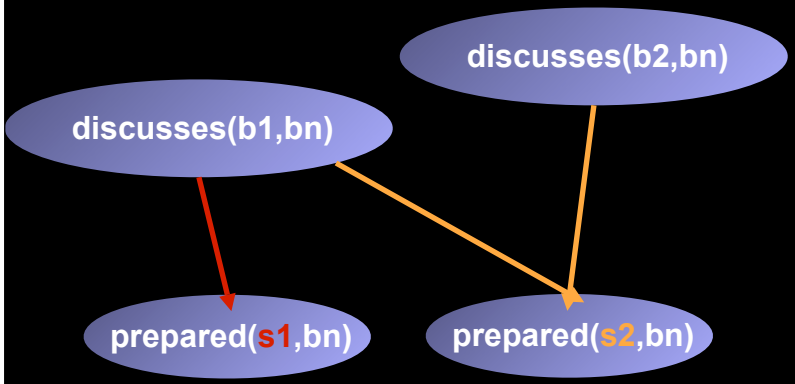
Combining Partial Knowledge

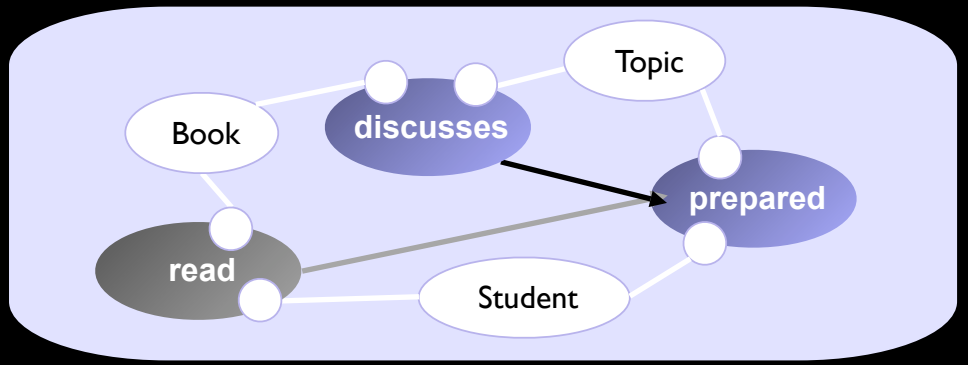



```
prepared(Student,Topic) | read(Student,Book),
                          discusses(Book,Topic).
```

```
passes(Student) | prepared(Student,bn),
                  prepared(Student,logic).
```

Combining Partial Knowledge

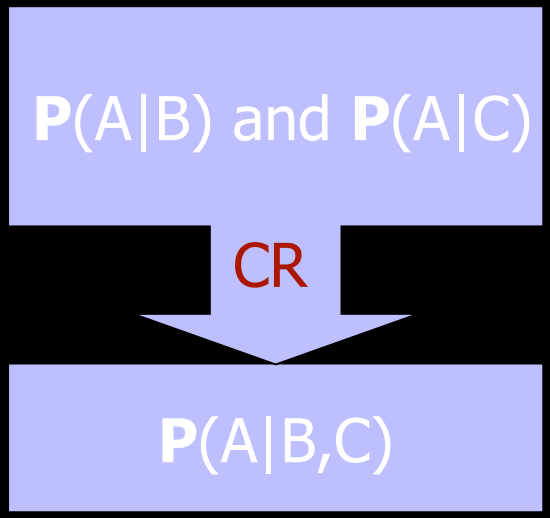


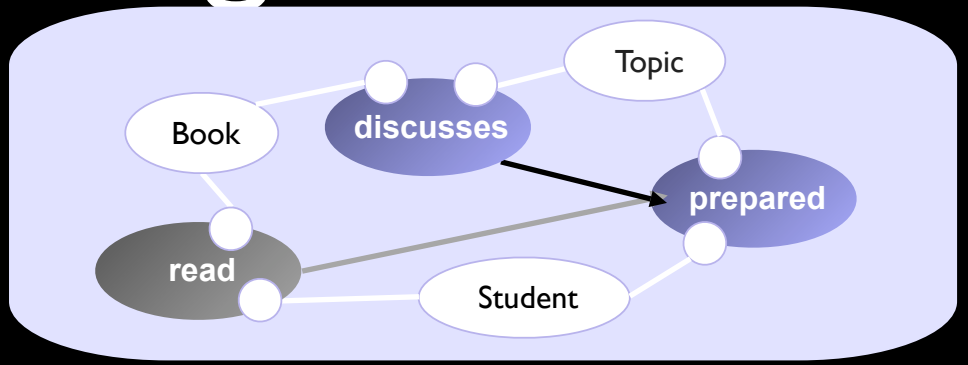



```
prepared(Student,Topic) | read(Student,Book),
                          discusses(Book,Topic).
```

- variable # of parents for prepared/2 due to read/2
  - whether a student prepared a topic depends on the books she read
- CPD only for one book-topic pair

Combining Rules



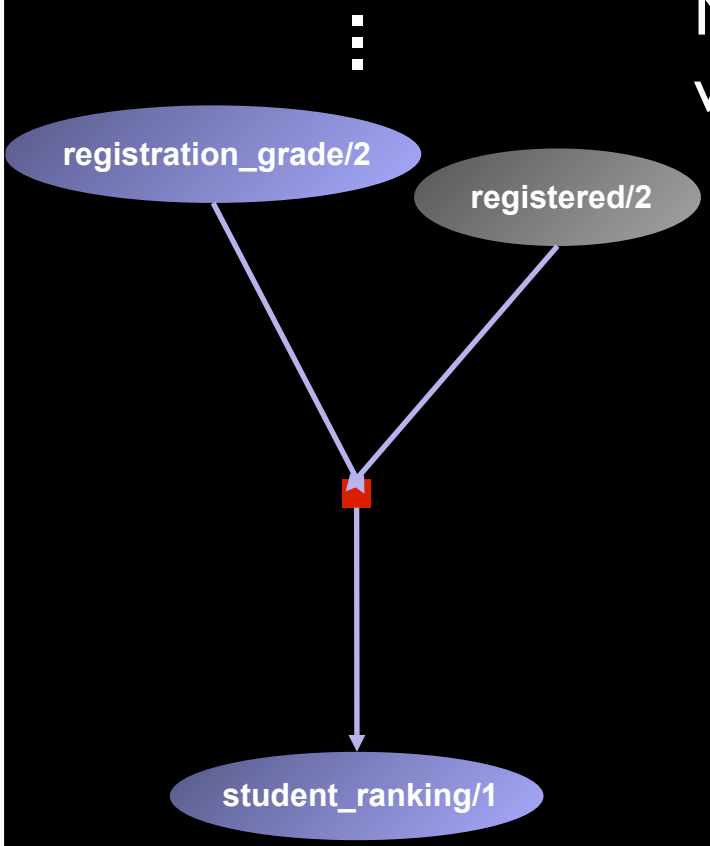



```
prepared(Student,Topic) | read(Student,Book),
                          discusses(Book,Topic).
```

Any algorithm which
has an empty output if and only if the input is empty
combines a set of CPDs into a single (combined) CPD
E.g. noisy-or, regression, ...

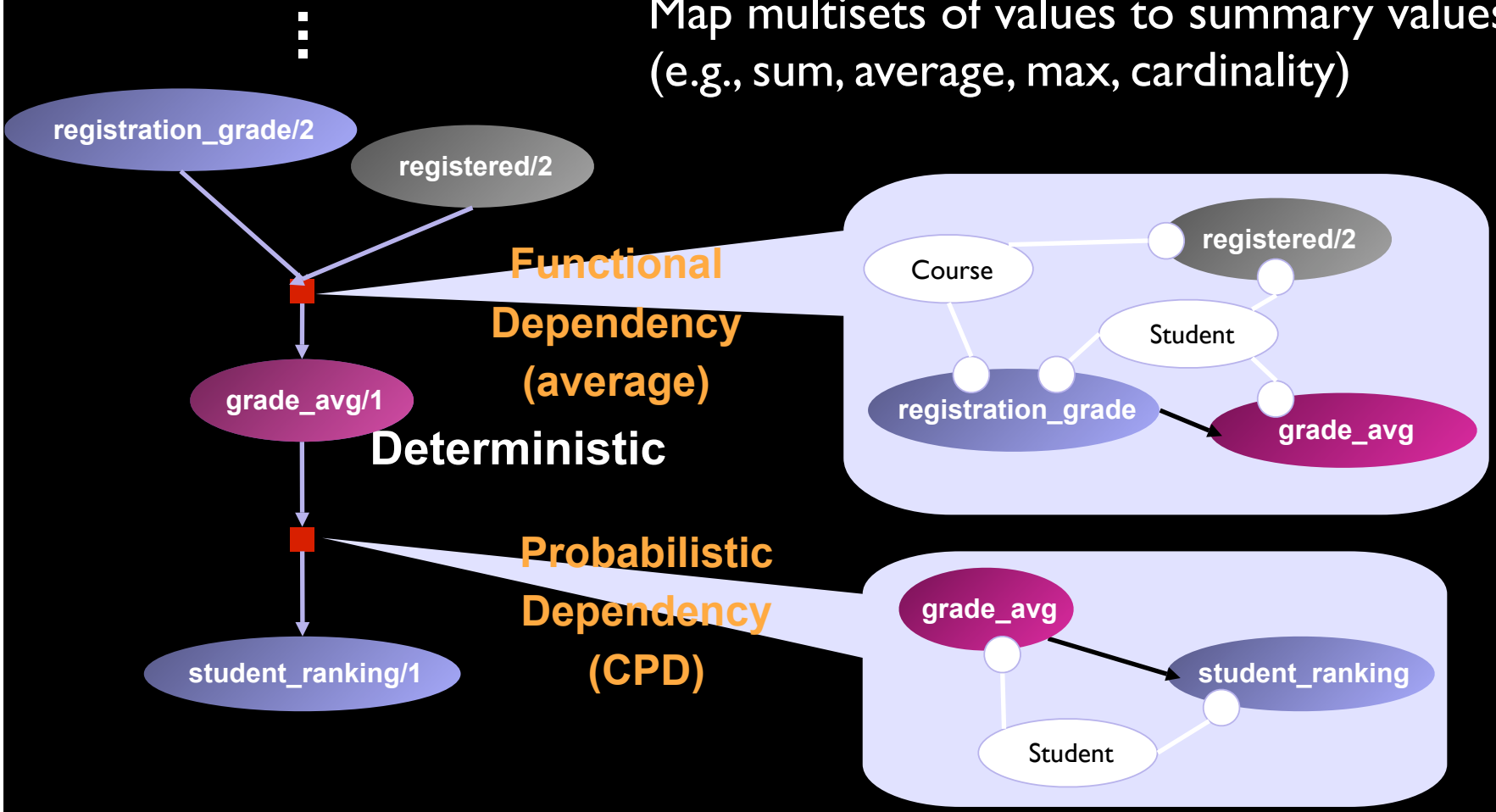
Aggregates

Map multisets of values to summary values (e.g., sum, average, max, cardinality)

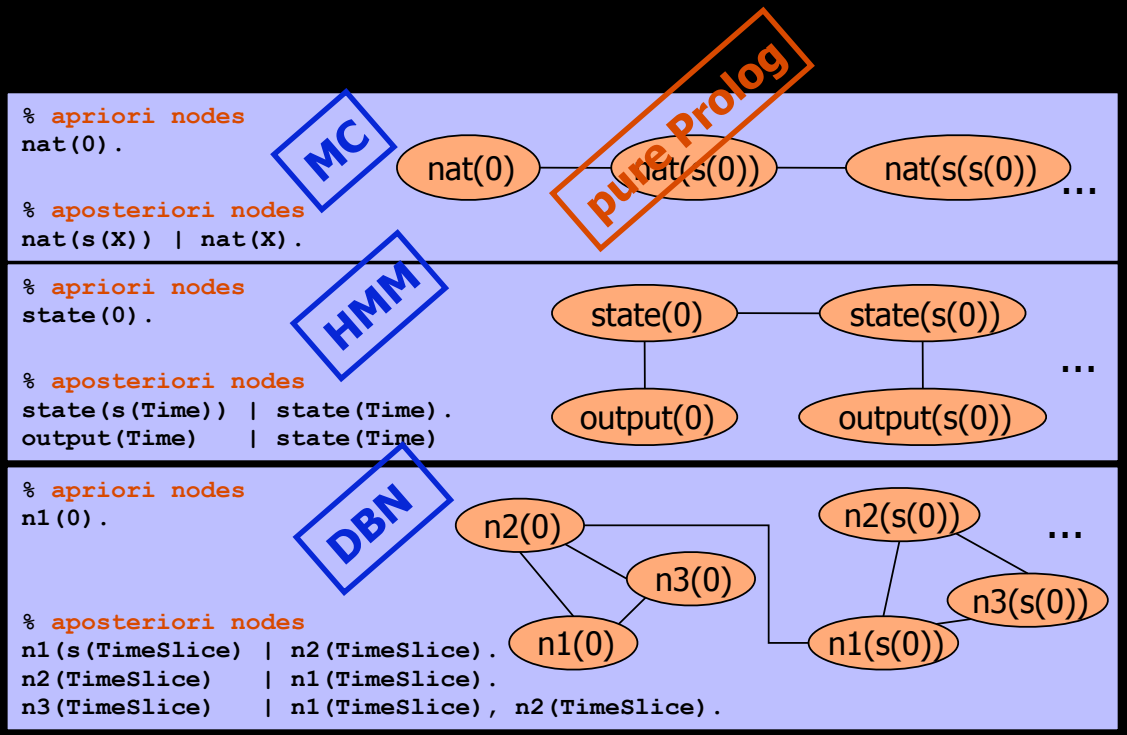


Aggregates

Map multisets of values to summary values
(e.g., sum, average, max, cardinality)



Bayesian Logic Programs



Prolog and Bayesian Nets as Special Case

Learning: the data

RVs + States = (partial) Herbrand interpretation
Probabilistic learning from interpretations

Family(1)

pc(brian)=b,
bt(ann)=a,
bt(brian)=?,
bt(dorothy)=a

Background

m(ann,dorothy),
f(brian,dorothy),
m(cecily,fred),
f(henry,fred),
f(fred,bob),
m(kim,bob),
...

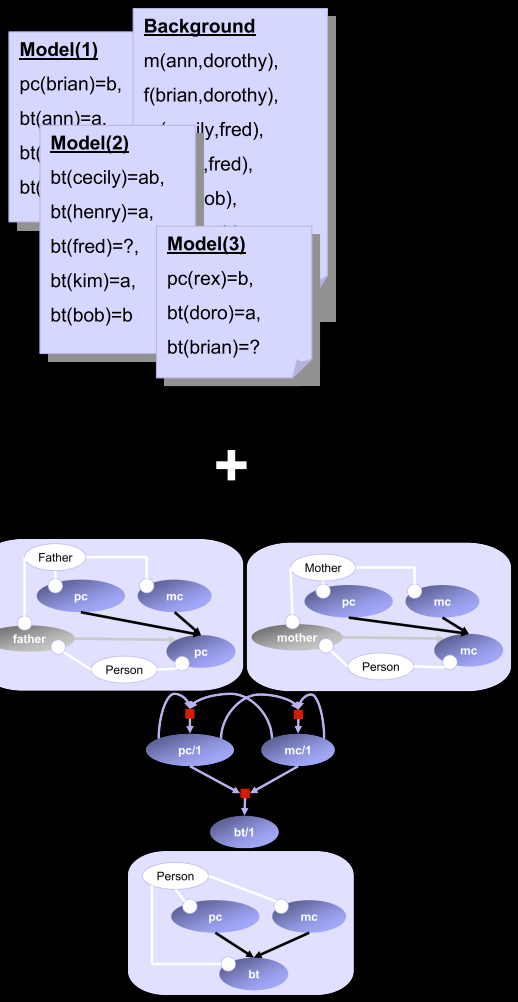
Family(3)

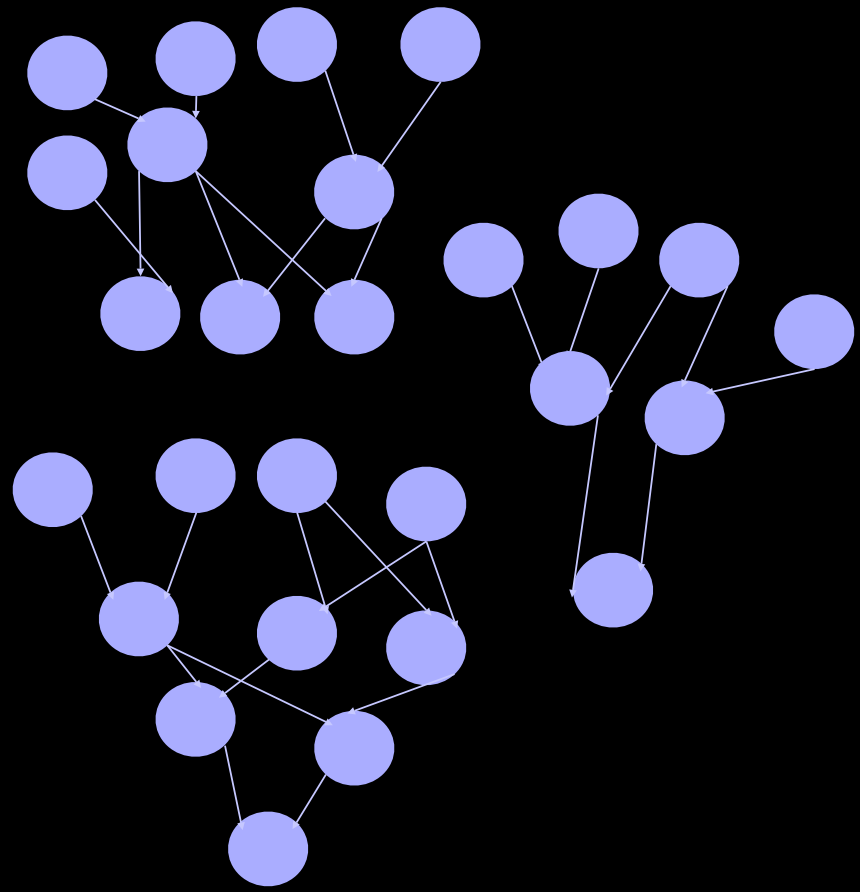
pc(rex)=b,
bt(doro)=a,
bt(brian)=?

Family(2)

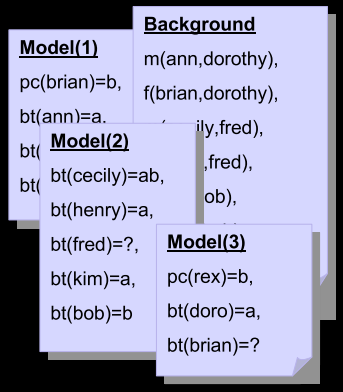
bt(cecily)=ab,
pc(henry)=a,
mc(fred)=?,
bt(kim)=a,
pc(bob)=b

Parameter Estimation

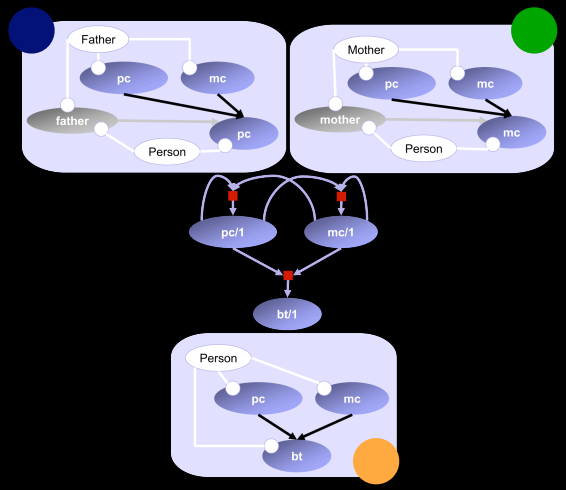


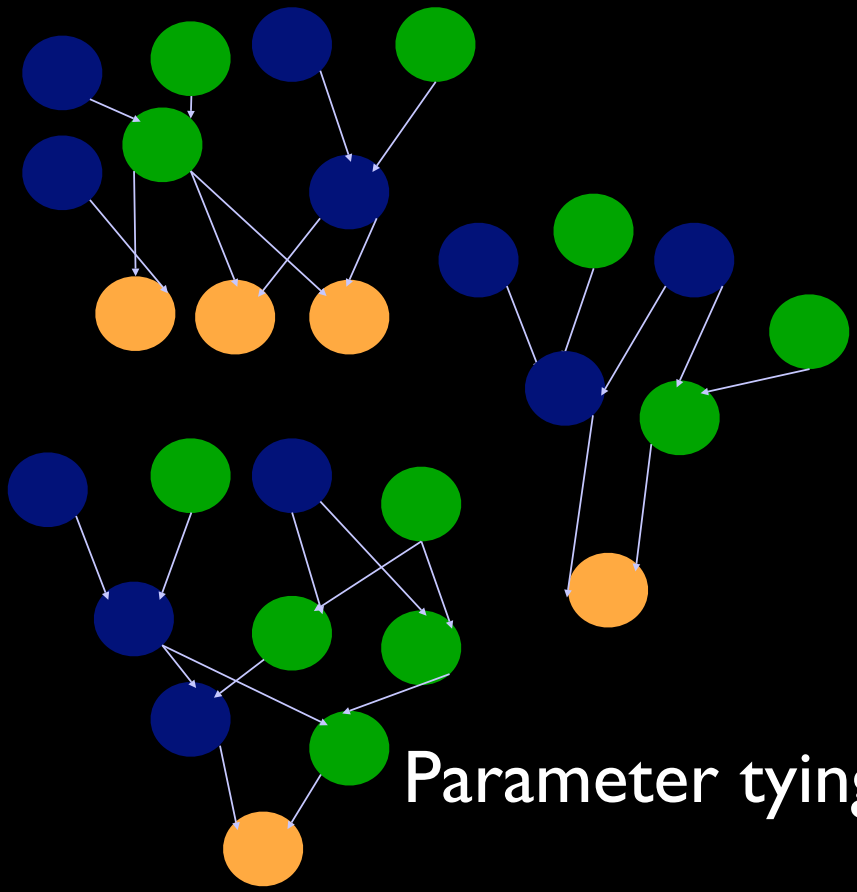


Parameter Estimation



+



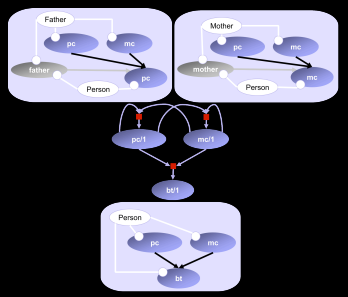


Parameter tying

Expectation Maximization

EM-algorithm:
iterate until convergence

Logic Program L



Background
m(ann,dorothy),
f(brian,dorothy),
...ily,fred),
...,fred),
...ob),

Model(1)
pc(brian)=b,
bt(ann)=a.
bt(
bt(

Model(2)
bt(cecily)=ab,
bt(henry)=a,
bt(fred)=?,
bt(kim)=a,
bt(bob)=b

Model(3)
pc(rex)=b,
bt(doro)=a,
bt(brian)=?

Initial Parameters q0

Current Model (M,qk)

Expected counts of a clause

$\sum_{\text{Ground Instance GI}} \sum_{\text{DataCase DC}}$ **P(head(GI), body(GI) | DC)**

$\sum_{\text{Ground Instance GI}} \sum_{\text{DataCase DC}}$ **P(head(GI), body(GI) | DC)**

$\sum_{\text{Ground Instance GI}} \sum_{\text{DataCase DC}}$ **P(body(GI) | DC)**

Maximization

Update parameters
(ML, MAP)

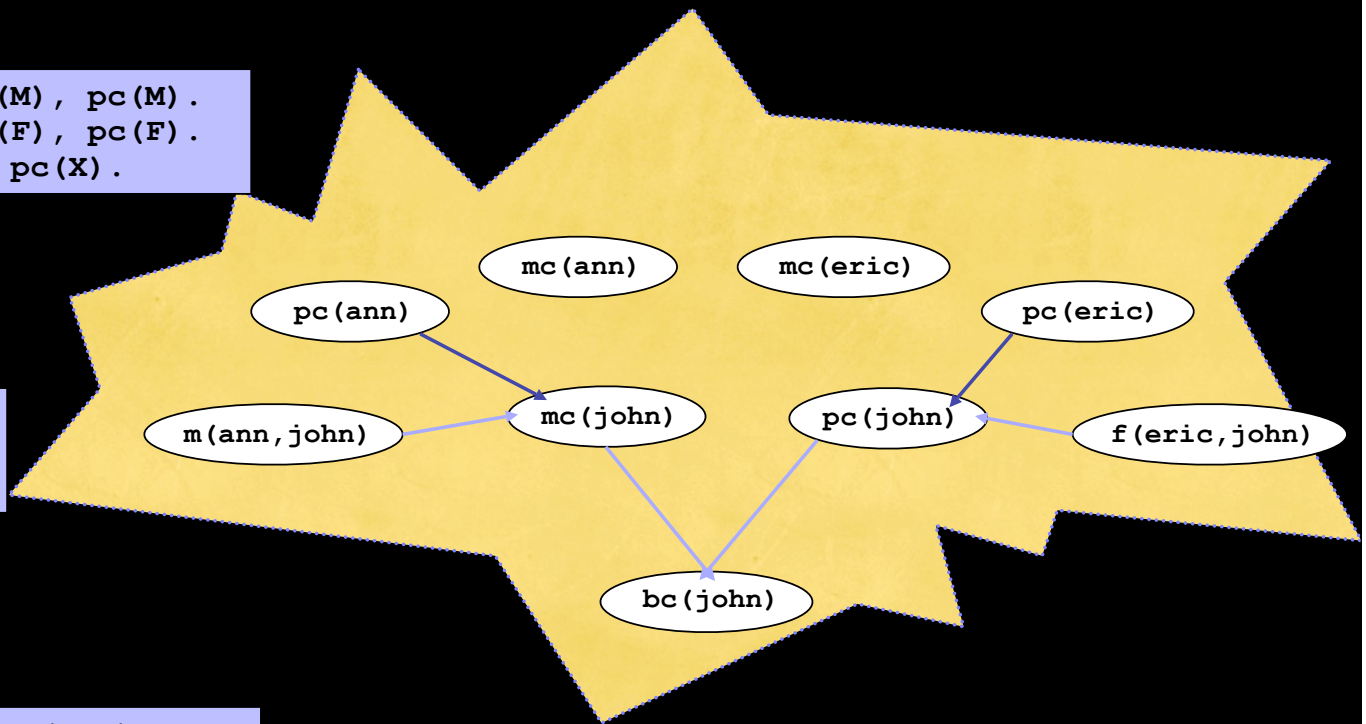
Structure Learning

Original program

```
mc(X) | m(M,X), mc(M), pc(M).
pc(X) | f(F,X), mc(F), pc(F).
bt(X) | mc(X), pc(X).
```

Initial hypothesis

```
mc(X) | m(M,X).
pc(X) | f(F,X).
bt(X) | mc(X).
```



Refinement

```
mc(X) | m(M,X).
pc(X) | f(F,X).
bt(X) | mc(X), pc(X).
```

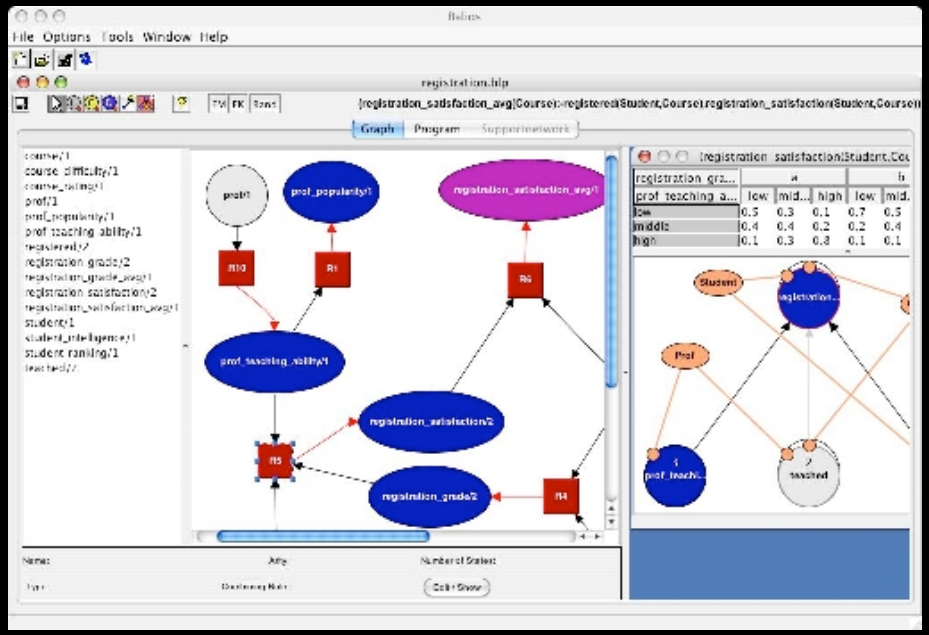
Refinement

```
mc(X) | m(M,X),pc(X).
pc(X) | f(F,X).
bt(X) | mc(X), pc(X).
```

Bayesian Logic Programs (BLPs)

- Unique probability distribution over Herbrand interpretations
  - Finite branching factor, finite proofs, no self-dependency
- Highlight
  - Separation of qualitative and quantitative parts
  - Functors
- Graphical Representation
- Discrete and continuous RV
- BNs, DBNs, HMMs, SCFGs, Prolog ...
- Turing-complete programming language
- Learning

Balios Tool


File Options Tools Window Help
registration.blp
Graph
Program
prof1
prof_popularity1
registration_satisfaction_avg1
registration_satisfaction2
registration_grade2
prof_teaching_ability1
student1
student ranking1
registration_satisfaction_avg1
registration_satisfactionStudent,Cou
registration_gra...
prof_teaching_a...
low
middle
high
Student
Prof

Undirected Probabilistic Relational

So far, directed graphical models only

Impose acyclicity constraint

Undirected graphical models do not impose the acyclicity constraint

Undirected Probabilistic Relational

Two approaches

- Relational Markov Networks (RMNs) (Taskar et al.)
- Markov Logic Networks (MLNs) (Anderson et al.)

Idea

- Semantics defined in terms of Markov Networks (undirected graphical models)
- More natural for certain applications

RMNs ~ undirected PRM

MLNs ~ undirected BLP

A Markov Network/Random Field

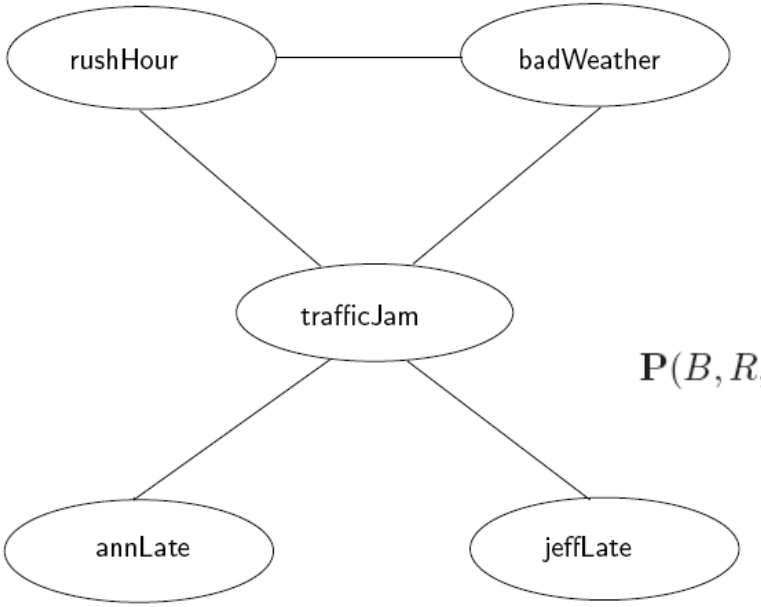


Fig. 8.3. A Markov network

$$\mathbf{P}(X_1, \cdots, X_n) = \frac{1}{Z} \prod_{c:clique} \mathbf{f}_c(X_{c1}, \cdots, X_{ck_c})$$

$$Z = \sum_{x_1, \cdots, x_n} \prod_{c:clique} f_c(X_{c1} = x_{c1}, \cdots, X_{ck_c} = x_{ck_c})$$

$$\mathbf{P}(B, R, T, J, A) = \frac{1}{Z} \mathbf{f}_{T,R,B}(T, R, B)\ \mathbf{f}_{J,T}(J, T)\ \mathbf{f}_{A,T}(A, T)$$

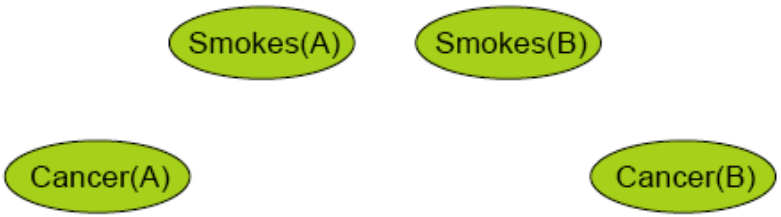
$$f_c(X_{c1}, \cdots, X_{ck_c}) = e^{w_i F_i(X_{c1}, \cdots, X_{ck_c})}$$

Assumption 2 *Each node X_i in the graph is conditionally independent of any subset $\mathbf{A}$ of nodes that are not neighbours of X_i given a joint state of $\mathbf{Ne}(X_i)$, that is, $\mathbf{P}(X_i \mid \mathbf{A}, \mathbf{Ne}(X_i)) = \mathbf{P}(X_i \mid \mathbf{Ne}(X_i))$. The expression $\mathbf{Ne}(X_i)$ refers to the set of all neighbours of X_i.*

Markov Logic

1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x, y) \Rightarrow \big(Smokes(x) \Leftrightarrow Smokes(y)\big)$

Suppose we have two constants: **Anna** (A) and **Bob** (B)

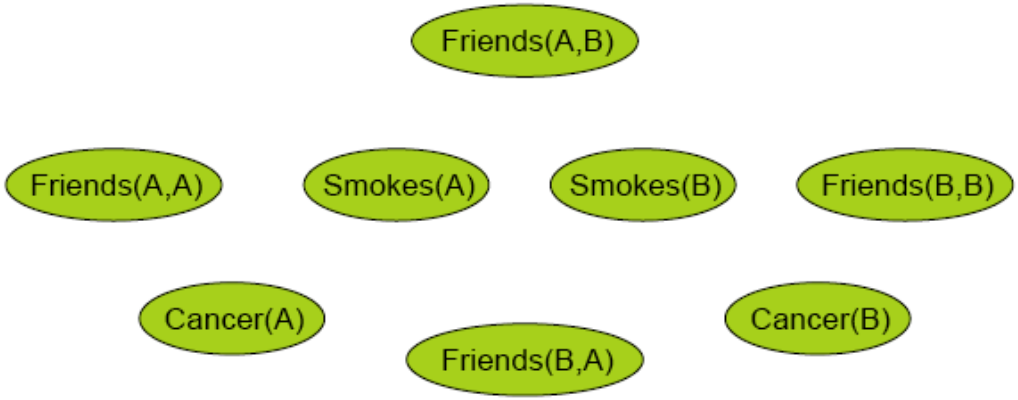


(slide and work by Domingos et al.)

Markov Logic

1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x, y) \Rightarrow \left(Smokes(x) \Leftrightarrow Smokes(y)\right)$

Suppose we have two constants: **Anna** (A) and **Bob** (B)

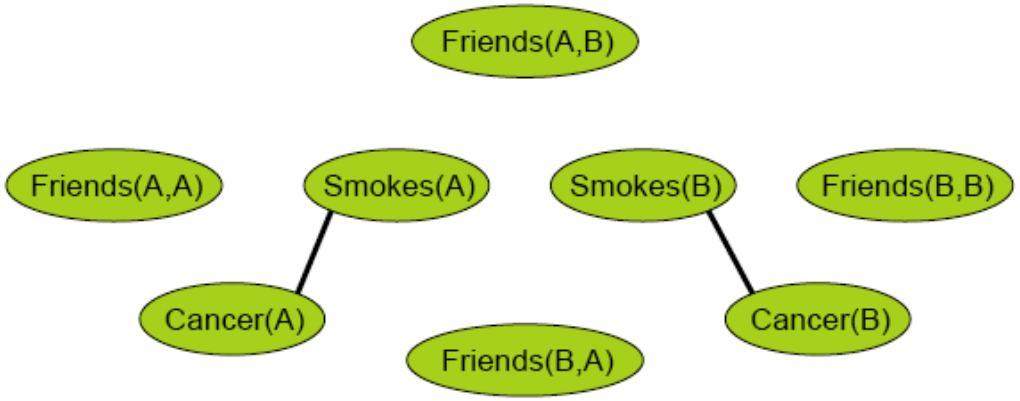


(slide and work by Domingos et al.)

Markov Logic

1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x, y) \Rightarrow \big(Smokes(x) \Leftrightarrow Smokes(y)\big)$

Suppose we have two constants: **Anna** (A) and **Bob** (B)

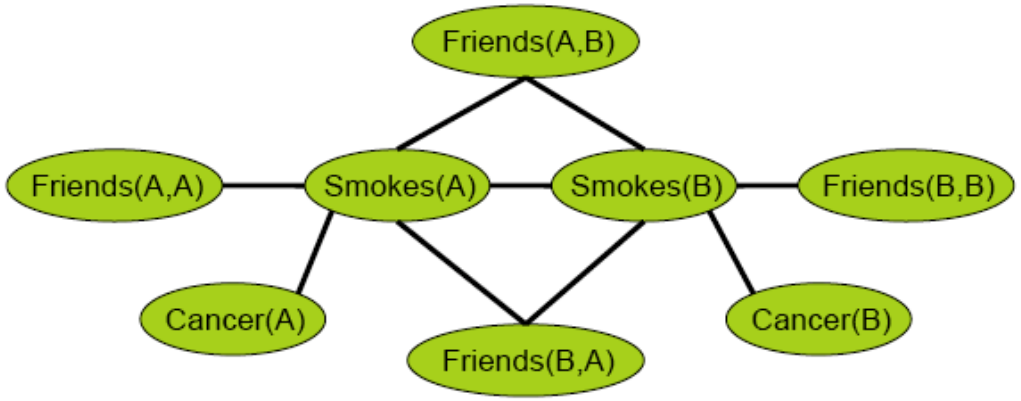


(slide and work by Domingos et al.)

Markov Logic

1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x, y) \Rightarrow \big(Smokes(x) \Leftrightarrow Smokes(y)\big)$

Suppose we have two constants: **Anna** (A) and **Bob** (B)



(slide and work by Domingos et al.)

Some interesting problems

- Inference is generally intractable
- Most Probable State Estimation
- Lazy Max Walk Sat algorithm

MLNs

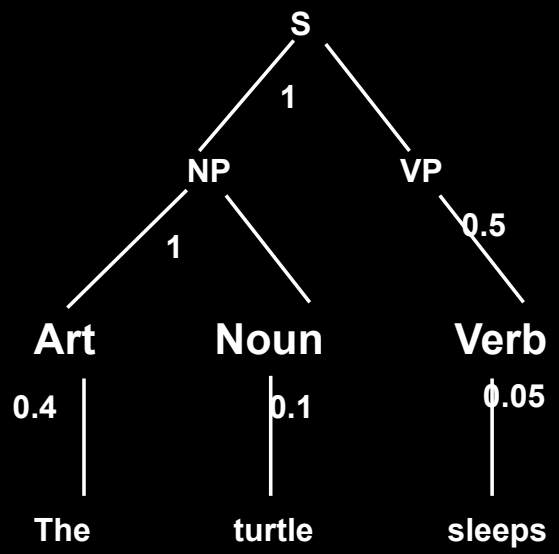
- Probably the best known and most elaborated approach to SRL
- Many different inference & learning algorithms known; discriminative, generative, structure learning, ...
- See website Pedro Domingos and Alchemy system

C. Probabilistic Logic Learning

C.2. Learning from proofs & entailment

Probabilistic Context Free Grammars

1.0 : S -> NP, VP
1.0 : NP -> Art, Noun
0.6 : Art -> a
0.4 : Art -> the
0.1 : Noun -> turtle
0.1 : Noun -> turtles
…
0.5 : VP -> Verb
0.5 : VP -> Verb, NP
0.05 : Verb -> sleep
0.05 : Verb -> sleeps
….



P(parse tree) = 1x1x.5x.1x.4x.05

PCFGs

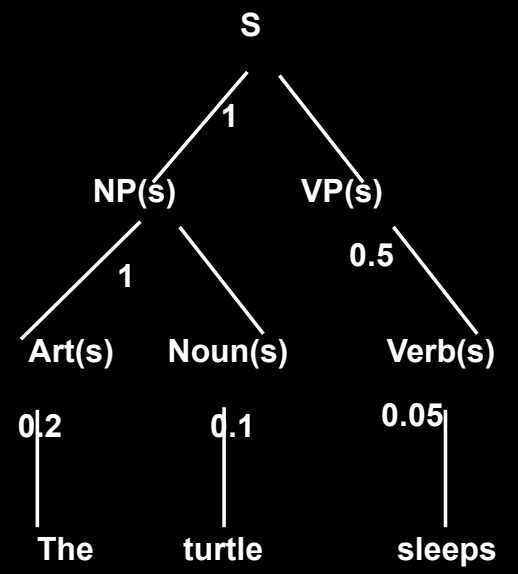
$P(parse\ tree) = \prod_i p_i c_i$
where p_i is the probability of rule i
and c_i the number of times
it is used in the parse tree

$P(sentence) = \sum_{p:parsetree} P(p)$

Observe that derivations always succeed, that is
$S \to T, Q$ and $T \to R, U$
always yields
$S \to R, U, Q$

Probabilistic DCG

1.0 S -> NP(Num), VP(Num)
1.0 NP(Num) -> Art(Num), Noun(Num)
0.6 Art(sing) -> a
0.2 Art(sing) -> the
0.2 Art(plur) -> the
0.1 Noun(sing) -> turtle
0.1 Noun(plur) -> turtles
…
0.5 VP(Num) -> Verb(Num)
0.5 VP(Num) -> Verb(Num), NP(Num)
0.05 Verb(sing) -> sleep
0.05 Verb(plur) -> sleeps
….



P(derivation tree) = 1x1x.5x.1x .2 x.05

As a Stochastic Logic Program

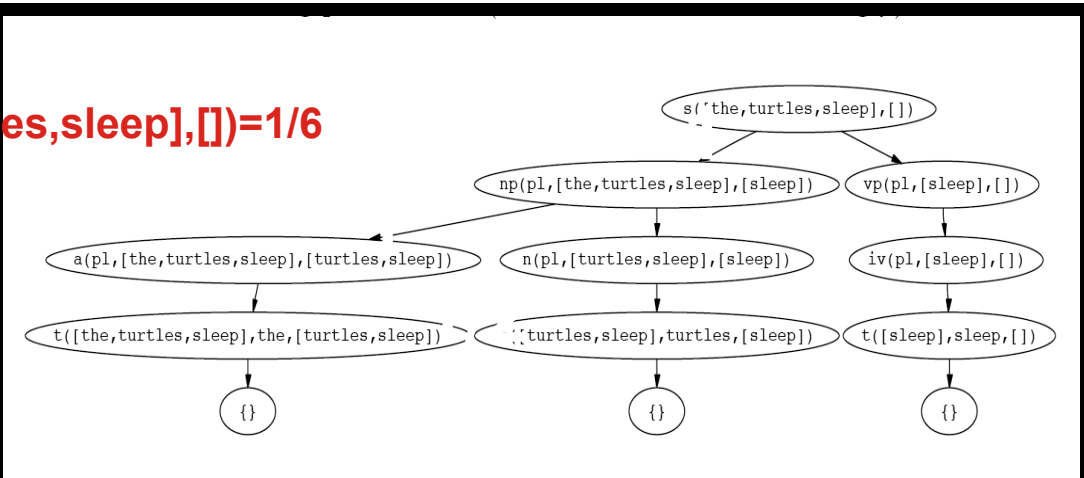
1

1/3

1/2

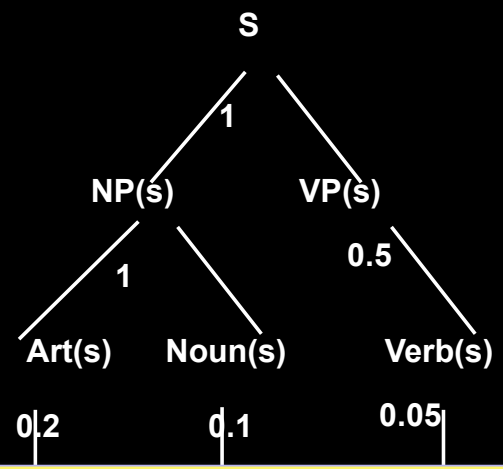
```
sentence(A, B) :- noun_phrase(C, A, D), verb_phrase(C, D, B).
noun_phrase(A, B, C) :- article(A, B, D), noun(A, D, C).
verb_phrase(A, B, C) :- intransitive_verb(A, B, C).
article(singular, A, B) :- terminal(A, a, B).
article(singular, A, B) :- terminal(A, the, B).
article(plural, A, B)   :- terminal(A, the, B).
noun(singular, A, B) :- terminal(A, turtle, B).
noun(plural, A, B)   :- terminal(A, turtles, B).
intransitive_verb(singular, A, B) :- terminal(A, sleeps, B).
intransitive_verb(plural, A, B) :- terminal(A, sleep, B).
terminal([A|B],A,B).
```

P(s([the,turtles,sleep],[])=1/6



Probabilistic DCG

1.0 S -> NP(Num), VP(Num)
1.0 NP(Num) -> Art(Num), Noun(Num)
0.6 Art(sing) -> a
0.2 Art(sing) -> the
0.2 Art(plur) -> the
0.1 Noun(sing) -> turtle
0.1 Noun(plur) -> turtles
…
0.5 VP
0.5 VP
0.05 Verb(sing) -> sleep
0.05 Verb(plur) -> sleeps
….



What about "A turtles sleeps" ?

P(derivation tree) = 1x1x.5x.1x .2 x.05

SLPs

$P_d(derivation) = \prod_i p_i^{c_i}$
where p_i is the probability of rule i
and c_i the number of times
it is used in the parse tree

Observe that some derivations now fail due to unification, that
$np(Num) \rightarrow art(Num), noun(Num)$ and $art(sing) \rightarrow a$
$noun(plural) \rightarrow turtles$

Normalization necessary
$P_s(proof) = \frac{P_d(proof)}{\sum_i P_d(proof_i)}$

Example Application

- Consider traversing a university website
- Pages are characterized by predicate
  department(cs,nebel) denotes the page of cs following the link to nebel
- Rules applied would be of the form

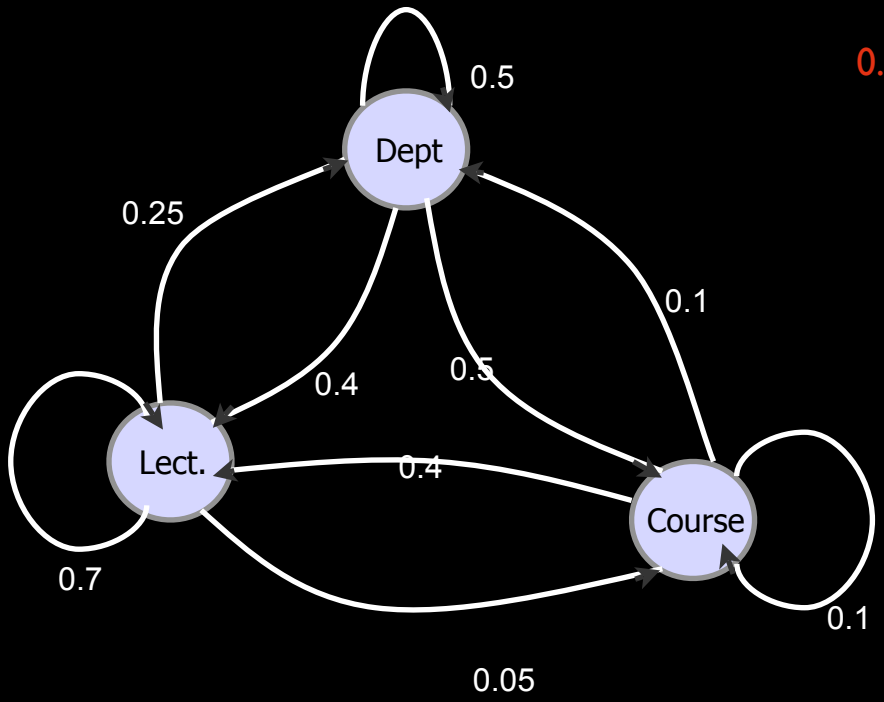
```
department(cs,nebel) :-
        prof(nebel), in(cs), co(ai), lecturer(nebel,ai).
pagetype1(t1,t2) :-
        type1(t1), type2(t2), type3(t3), pagetype2(t2,t3)
```

- SLP models probabilities over traces / proofs / web logs

```
department(cs,nebel), lecturer(nebel,ai007),
  course(ai007,burgard), …
```

This is actually a Logical Markov Model

Logical Markov Model




```
0.4 department(D,L) :-
      prof(L),
      in(D),
      co(C),
      lecturer(L,C).
```

```
0.1 prof(nebel).
0.05 prof(burgard).
…
```

Learning SLPs from entailment

Entailment

Given

- a set of ground facts — **sentences**
  - s([the, boy, rides])
- structure of the logic program — **CFG**

Find

- the parameters of the logic program that maximize the log likelihood — **PCFG**

Parameter estimation

- Failure adjusted maximization algorithm (Cussens)
  - Only facts observed
  - Proofs and Failures not observed ! — **no failures**
- Inside-outside algorithm for Prism (Sato)
- EM based techniques

Learning SLPs from entailment

Entailment

Given

- a set of ground facts **sentences**
- ~~the structure of the logic program~~

Find **CFG**

- the logic program and its parameters that maximizes the log likelihood

PCFG

Structure learning

- requires one to solve the underlying relational learning problem
- as well as the parameter estimation problem
- Very hard because the information contained in the examples is very limited. **no failures**
- What about using richer examples ? proofs ?

Learning SLPs from entailment

Entailment

Given

- a set of proof trees **sentences**
- ~~the structure of the logic program~~

Find **CFG**

- the logic program and its parameters that maximizes the log likelihood

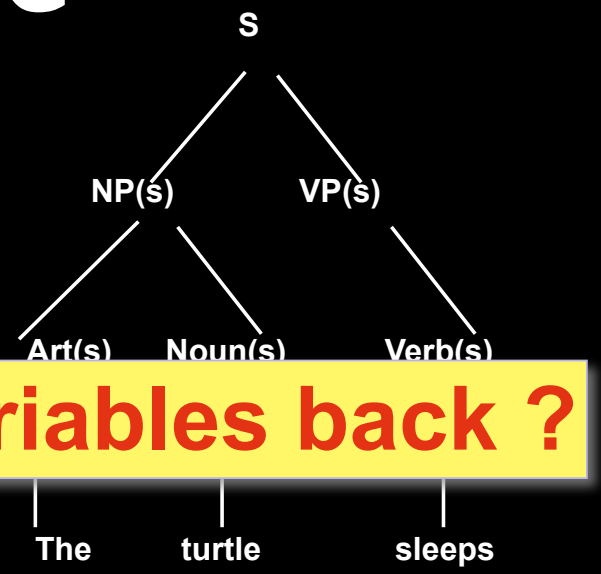
PCFG

Structure learning

- analogy with learning from tree banks, cf. De Raedt et al. AAAI 05.

no failures

Learning from parse tree



S -> NP(s), VP(s)
NP(s) -> Art(s),
N
VP(s) -> Verb(s)
Art(s) -> the
Noun(s) -> turtle
Verb(s) -> sleeps

How to get the variables back ?

Generalizing Rules in SLPs

Generalization in ILP

- Take two clauses for same predicate and replace them by the lgg under theta-subsumption (Plotkin), e.g

```
department(cs,nebel) :-
        prof(nebel), in(cs), course(ai), lect(nebel,ai).
department(cs,burgard) :-
        prof(burgard), in(cs),course(ai), lect(burgard,ai)
```

- Induce

```
department(cs,P) :-
   prof(P), in(cs),course(ai), lect(P,ai)
```

Strong logical constraints

Replacing the rules r1 and r2 by the lgg should preserve the proofs !

So, two rules r1 and r2 should only be generalized when

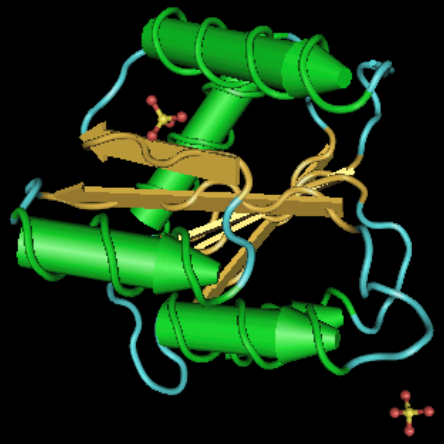
- There is a one to one mapping (with corresponding substitutions) between literals in r1, r2 and lgg(r1,r2)

Exclude

- father(j,a) :- m(j),f(a),parent(j,a)
- father(j,t) :- m(j),m(t), parent(j,t)

[Kersting et al.; Kersting, Gaertner]

Protein Fold Recognition



```
[helix(h(right,3to10),5),
 helix(h(right,alpha),13),
      strand(null,7),
      strand(minus,7),
      strand(minus,5),
 helix(h(right,3to10),5),…]
```

If one of the two similar proteins has a known structure, can build a rough model of the protein of unknown structure.

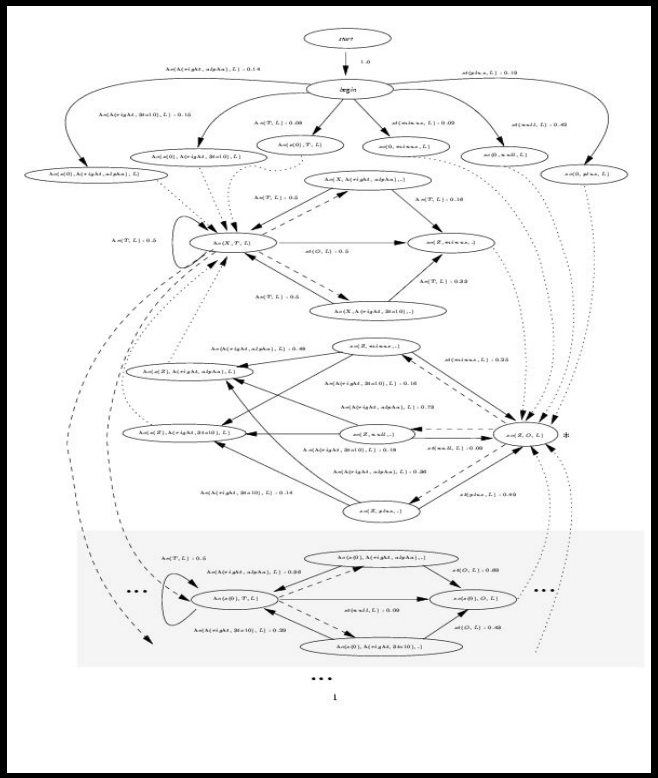
LoHMM

[Kersting et al., JAIR 2006]

~120 parameters

vs.

over 62000 parameters



Secondary structure of domains of proteins

(from PDB and SCOP)
fold1: TIM beta/alpha barrel fold, fold2: NAD(P)-binding Rossman-fold fold23: Ribosomal protein L4, fold37: glucosamine 6-phosphate deaminase/isomerase old fold55: leucine aminopeptidas fold. 3187 logical sequences (> 30000 ground atoms)

Results

[Kersting et al.; Kersting, Gaertner; Landwehr et al.]

Accuracy: 74% vs. 82.7% (1622 vs. 1809 / 2187)

Majority vote: 43%

	fold1	fold2	fold23	fold37	fold55
precision	0.86 / 0.89	0.69 / 0.86	0.56 / 0.82	0.72 / 0.70	0.66 / 0.74
recall	0.78 / 0.87	0.67 / 0.81	0.71 / 0.85	0.66 / 0.72	0.96 / 0.86

Web Log Data
[Anderson et al., KDD 03]

Relational Markov Models

Log data of web sides

KDDCup 200

RMM

- Home()
- Boutique()
- Departments()
- Legcare_vendor()
- Lifestyles()
- Vendor()
- AssortmentDefault()
- Assortment(Assortment)
- ProductDetailLegcareDefault()
- ProductDetailLegcare(Product)
- ProductDetailLegwearDefault()
- ProductDetailLegwearProduct(Product)
- ProductDetailLegwearAssortment(Assortment)
- ProductDetailLegwearProdCollect(Product, Collection)
- ProductDetailLegwearProdAssort(Product, Assortment)
- ProductDetailLegwear(Product, Collection, Assortment)

Probabilities on Proofs

Two views

- stochastic logic programs define a prob. distribution over atoms for a given predicate.
  - The sum of the probabilities = 1.
  - Sampling. Like in probabilistic grammars.
- distribution semantics define a prob. distribution over possible worlds/interpretations. Degree of belief.

Probabilities on Facts

- Sato's PRISM and Poole's ICL and PHA use disjoint statements (or multi-valued switches)

  disjoint(p_1:a_1, ... , p_k:a_k)

  disjoint(0.5:head(C); 0.5:tail(C))

  head(C) and tail(C) are mutually exclusive

  false :- head(C), tail(C)

- for such statements: one of the alternatives is true, instead of true or false for a single fact.

PRISM (Sato) / ICL (Poole)

A logic program in which probability labels are attached to facts;

Clauses carry no probability label (or equiv. P = 1)

Probability distributions can be defined in a related/similar fashion on proofs / on explanations / on atoms - though some differences

Abductive reasoning with probabilities !

PRISM and ICL more expressive than SLPs

- SLPs can be easily transformed into ICL or PRISM

ICL / PRISM example

```
btype('A') : - (gtype(a,a); gtype(a,o); gtype(o,a)).

btype('B') : - (gtype(b,b); gtype(b,o); gtype(o,b)).

btype('O') : - gtype(o,o).

btype('AB') : - (gtype(a,b); gtype(b, a)).

gtype(X,Y) :- gene(father,X),   gene(mother,Y).

gene(P,G) :- msw(gene,P,G).          probabilistic switch

disjoint( p1: msw(gene,P,o),
          p2: msw(gene,P,a),
          p3: msw(gene,P,b))          outcomes + probabilities switch
```

Infer e.g. P(btype('A'))

Abduction

btype('A') iff gtype(a,a) or gtype(a,o) or gtype(o,a)

gtype(a,a) iff msw(gene,father,a) and msw(gene,mother,a)

gtype(a,o) iff msw(gene,father,a) and msw(gene,mother,o)

gtype(o,a) iff msw(gene,father,o) and msw(gene,mother,a)

So, btype('A') = p1xp1 + p1Xp2+p2Xp1

Can be used for sampling as well.

ICL/PRISM Assumption

- The Exclusive Explanation Assumption :
  - avoid the NP-complete problem by assuming that all explanations (that is proofs, or sets $S \subseteq F$, are mutually exclusive)
  - under this condition (which can be enforced in different ways)
    - $P(A \vee B \vee C) = P(A) + P(B) + P(C)$

Learning from entailment

Given

- examples of the form btype(x)
- structure of program

Estimate parameters of switches

Nice results -- HMMs, PCFGs, some type of BNs as special case

Sato's Prism one of the most advanced PLL systems

Has been used by Cussens to learn SLPs

Learning from Entailment and Proofs

SLPs extend PCFGs as a representation

Proof-trees for SLPs correspond to parse-trees in PCFGs

Upgrading the learning from tree-banks setting for use in SLPs

Allows one also to elegantly model and study RMMs and LOHMMs

- Sequential relational / logical traces.

Defining probabilities on facts provides interesting perspective

- PRISM and ICL
- Probabilistic Abduction
- Also learning ...

C. Probabilistic Logic Learning

C.3. Discriminative models & propositionalization

Probabilistic *discriminative* ILP Problem

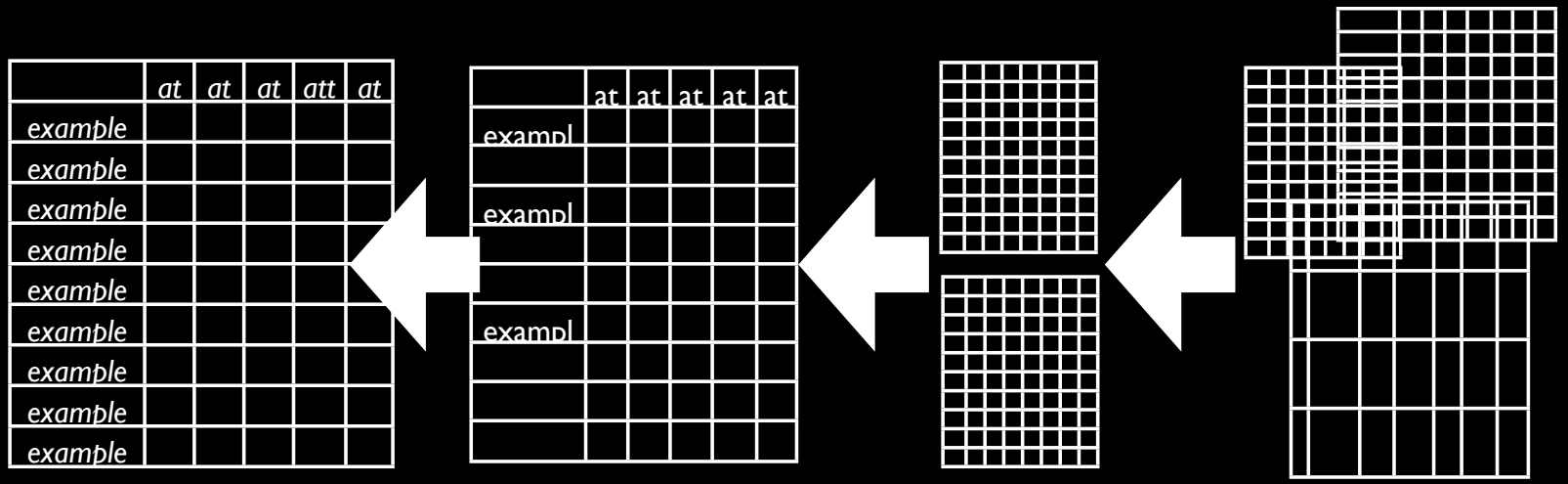
Given

- a set of examples **E, positive and negatives.**
- a background theory **B**
- a language **Le** to represent examples
- a language **Lh** to represent hypotheses
- a probabilistic covers **P** relation on **Le x Lh**

Find

- hypothesis h* maximizing some score based on the probabilistic covers relation; often some kind of conditional likelihood

Propositionalization



single-table
single-tuple
attribute-value

single-table
multiple-tuple
multi-instance

2 relations
edge / vertex
graphs & networks

multi-table
multiple-tuple
relational

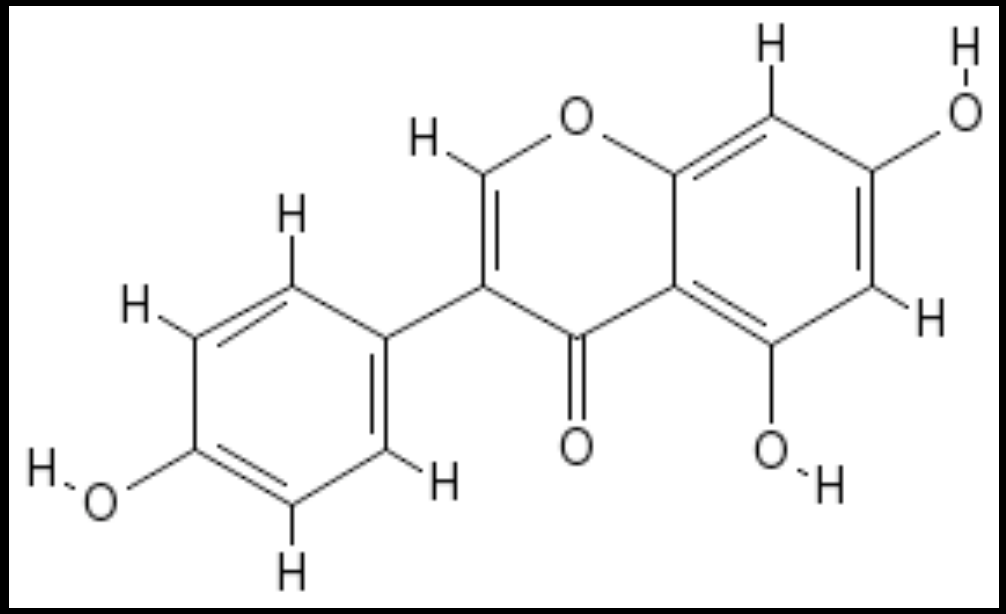
Downgrading the data ?

Two questions

UPGRADING : Can we develop systems that work with richer representations (starting from systems for simpler representations)?

PROPOSITIONALISATION: Can we change the representation from richer representations to simpler ones ? (So we can use systems working with simpler representations)

Relational Data and Clausal Theories



Examples E

$pos(m_1)$
$neg(m_2)$
$pos(m_3)$
$pos(m_4)$
...

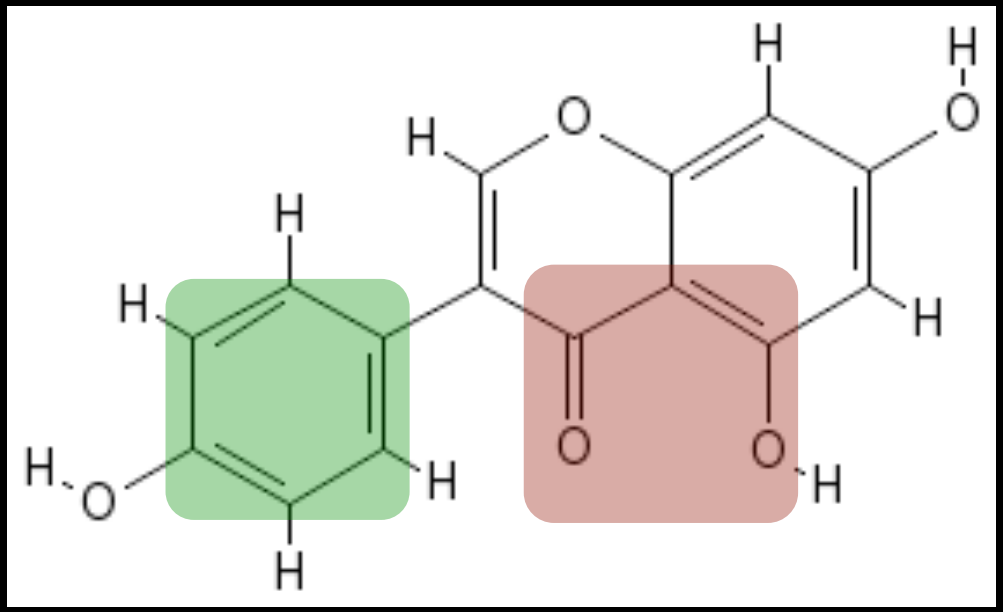
Background Knowledge B

$molecule(m_1)$
$atom(m_1, a_{11}, c)$
$atom(m_1, a_{12}, c)$
$bond(m_1, a_{12}, a_{11})$
$bond(m_1, a_{13}, a_{12})$
...

$molecule(m_2)$
$atom(m_2, a_{21}, o)$
$atom(m_2, a_{22}, n)$
$bond(m_2, a_{21}, a_{22})$
$charge(m_2, a_{21}, 0.82)$
...

$aromaticRing(M) :-$
$bond(M, A, B, -), bond(M, B, C, =),$
$bond(M, C, D, -), bond(M, D, E, =),$
$bond(M, E, F, -), bond(M, F, A, =).$

Relational Data and Clausal Theories



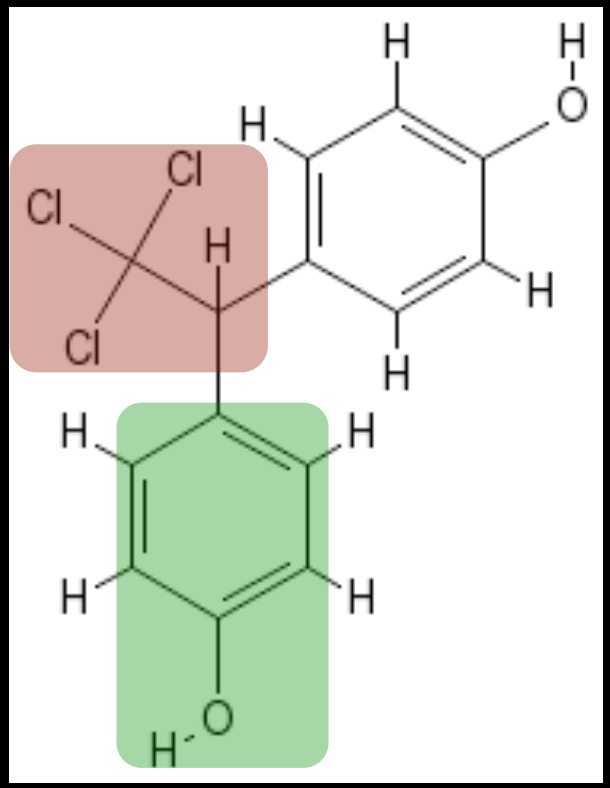
Clauses match on structural features of e.g. molecules
Set of clauses is called theory or hypothesis

$$pos(X) \leftarrow bond(A,B,=), bond(B,C,-), bond(C,D,=) \\ bond(D,E,-), bond(E,F,=), bond(F,A,-)$$

$$pos(X) \leftarrow atom(A,o), bond(A,B,=), bond(B,C,-) \\ bond(C,D,=), bond(D,E,-), atom(E,o)$$

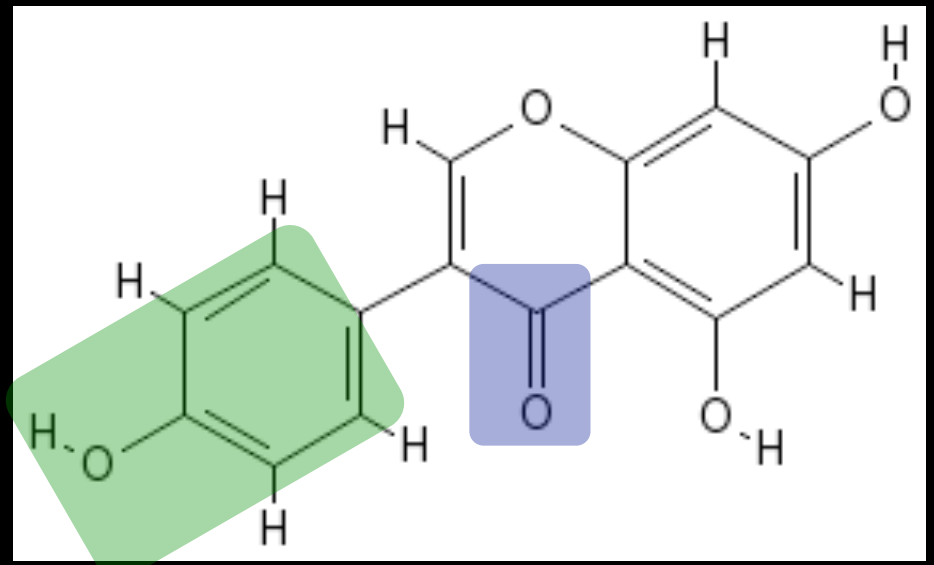
Propositionalization

Clauses c_1, c_2 c_3



$$\varphi_H(e_1) = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$$

$$\varphi_H(e_2) = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$$



Logical Theories

Theory $H = \{c_1, ..., c_k\}$ can be seen as a set of features

$$\varphi_H(e) = \begin{pmatrix} \pi_1 \\ \pi_2 \\ ... \\ \pi_k \end{pmatrix}$$

$$\pi_i = \begin{cases} 1 & : \quad c_i \text{ matches } e \\ 0 & : \quad \text{otherwise} \end{cases}$$

Disjunction: example positive if one of the clauses matches (hypothesis c1 v ... v ck)

	$\varphi_H(e_1)$	$\varphi_H(e_2)$	$\varphi_H(e_3)$	...
c_1	0	0	1	...
c_2	1	0	0	...
c_3	1	0	0	...
$class$	pos	neg	pos	...

How to learn H?

Traditional static propositionalization approaches

a. ILP + SVM
   1. Run ILP system to generate small set of clauses
   2. Train SVM based on these features

c. Pattern Mining + SVM
   1. Generate large set of clauses, e.g., all clauses that are frequent in the training data
   2. Train SVM based on these features

Problem: Two steps are independent

- we will not find the hypothesis maximizing our score
- Theory that yields optimal similarity measure different from best ILP theory!
- Information loss

nFOIL: Clauses + Probabilistic Model

nFOIL / tFOIL: combine clauses using a (tree augmented) naive Bayes model

$$
\begin{aligned}
P(p\theta \mid H, B) &= P(p\theta \mid c_1\theta, ..., c_k\theta) \\
&= \frac{P(c_1\theta, ..., c_k\theta \mid p\theta)P(p\theta)}{P(c_1\theta, ..., c_k\theta)} \\
&= \frac{\prod_{i=1}^{k} P(c_i\theta \mid p\theta)P(p\theta)}{P(c_1\theta, ..., c_k\theta)}
\end{aligned}
$$

Instantiated class predicate

Instantiated claus

Defines distribution over class variable for a given example
(substitution, instantiates to example)

kFOIL: Clauses + Kernel

kFOIL: combine clauses using a kernel machine

Idea: Theory H measures similarity

- e.g., standard dot product kernel $K_P(e_1, e_2, H) = \langle \varphi_H(e_1), \varphi_H(e_2) \rangle$

„number of features examples have in common“

- polynomial kernel $K_P(e_1, e_2, H) = (\langle \varphi_H(e_1), \varphi_H(e_2) \rangle + 1)^p$

„conjunctions of up to p features“

- Gaussian kernel $K_G(e_1, e_2, H) = exp(-\frac{\|\varphi_H(e_1) - \varphi_H(e_2)\|^2}{2\sigma})$

„symmetric difference: number of features which hold on only one example“

Classification can be obtained from similarity by kernel machine

How to learn H?

Looking for a hypothesis which is

- the best feature set in a naive Bayes model (nFOIL)
- defines the best similarity measure (kFOIL)

The optimization criterion is different from standard ILP

Use a score for clause search that takes into account the way the clauses are combined

- For nFOIL / tFOIL: Conditional likelihood (probabilistic coverage)

$$\prod_{e \in E} P(e \mid H, B) = \prod_{e \in E} \frac{\prod_{i=1}^{k} P(c_i\theta \mid e) P(e)}{P(c_1\theta, ..., c_k\theta)}$$

- For kFOIL: Performance of the kernel machine on the training data

The Score Function

Score of a theory H

- Theory defines kernel $K(e_i, e_j, H)$
- Parameters $\alpha_1, ..., \alpha_n, b$ are obtained by standard SVM training
- Given parameters, examples can be labeled according to

$$class(e) = sign(\sum_{i=1}^{n} \alpha_i y(e_i) K(e, e_i, H) + b)$$ (classification)

$$f(e) = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) K(e, e_i, H) + b$$ (regression)

- Most simple choice for $score(E, H, B)$ Accuracy/RMSE

How to learn H?

Dynamic propositionalization [Landwehr et. al. JMLR 06]

- Integrate the two steps
- Modify ILP algorithm: **search for clauses guided by performance of the combination method (NB/SVM)**

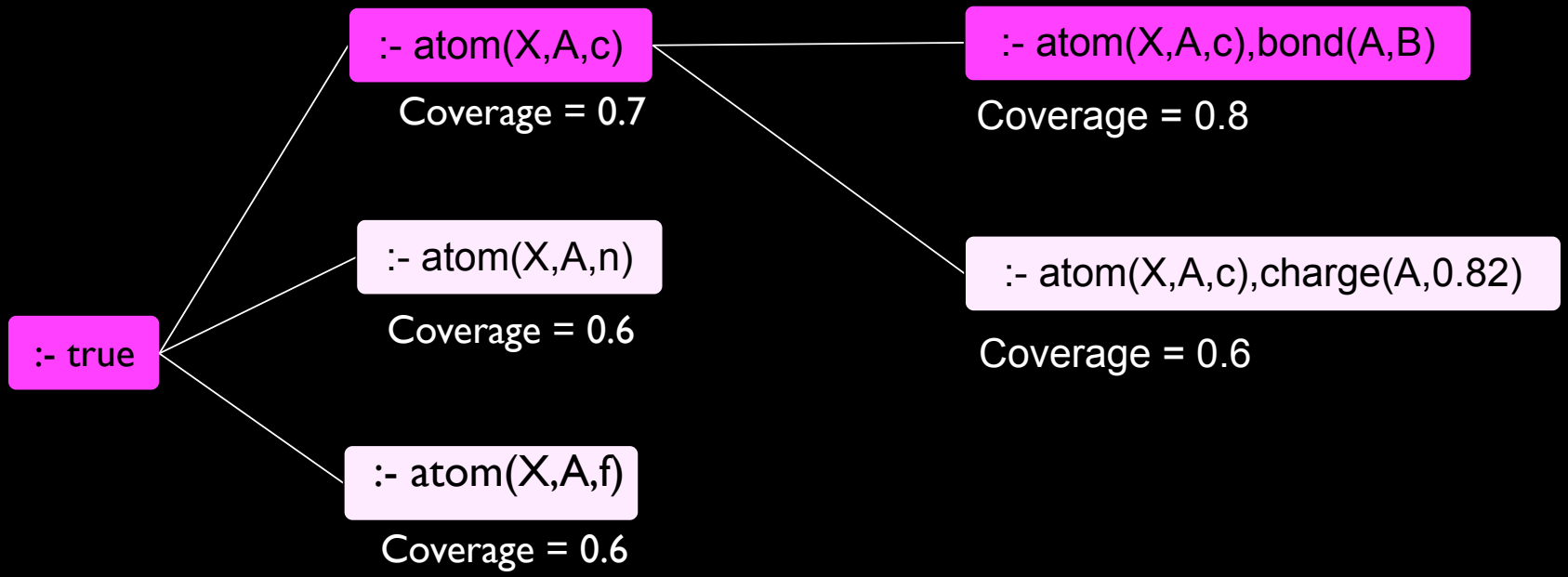
FOIL-style search

- Greedy general-to-specific search for clauses employing refinement operator

$$\rho(c) = \{c' \mid c' \text{ is minimal specialization of c in } \mathcal{L}\}$$

- Greedy incremental search for clause set

FOIL



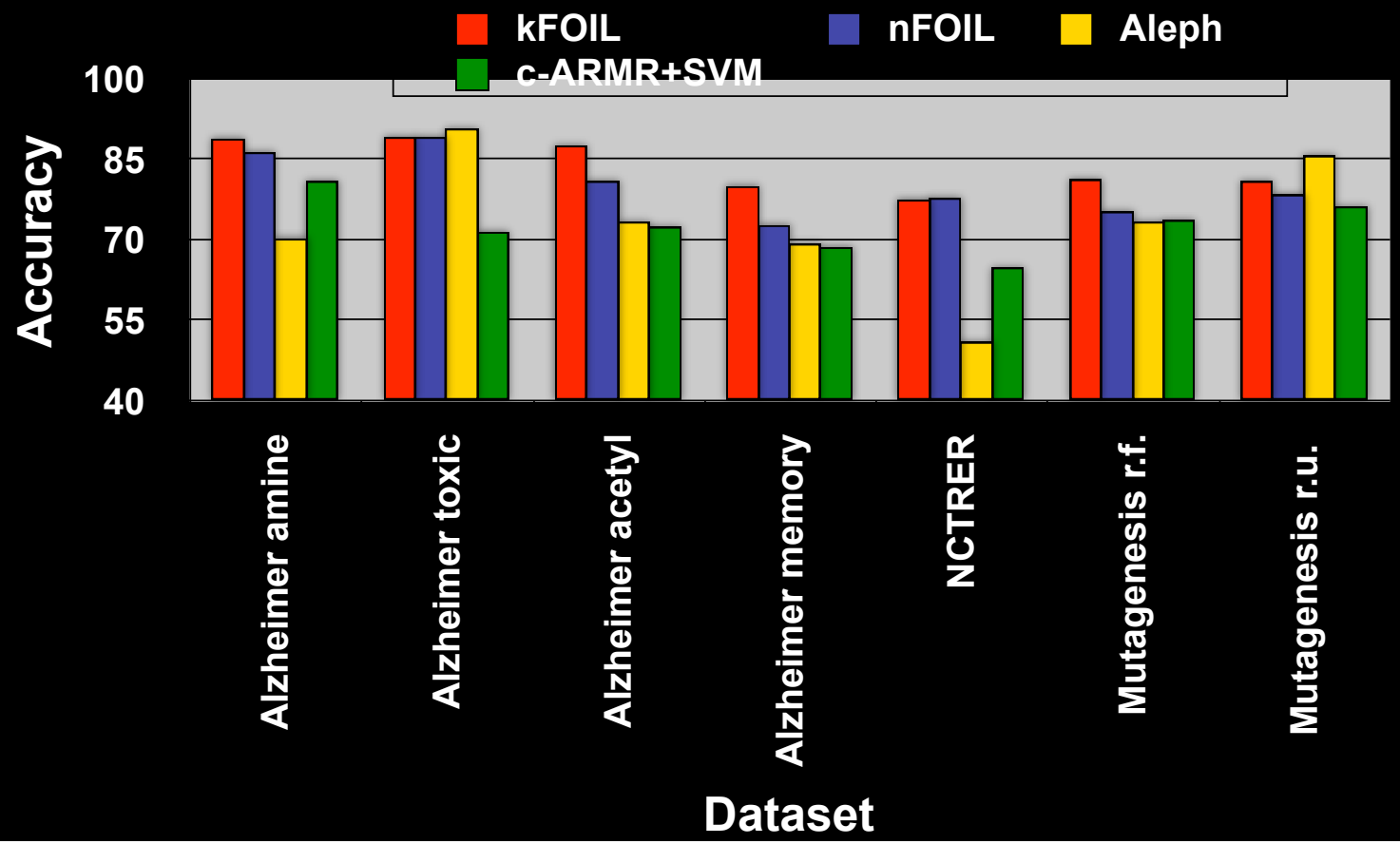
mutagenic(X) :- atom(X,A,c),bond(A,B)

mutagenic(X) :- atom(X,A,n),charge(A,0.82)

Experimental Results: Classification

Comparison against Aleph (ILP) and c-ARMR+SVM (static propositionalization)

on several ILP benchmark datasets



Propositionalization

Dynamic propositionalization can yield more accurate models than ILP or static propositionalization

- key idea: guide clause search by score related to combination method
- typically yields clause sets which are better suited to the particular combination method

Added computational complexity if a costly clause combination method (such as SVM) is used

- However, complexity can be reduced by incrementally refining the combination model when the clause set changes
- This brings dynamic propositionalization computationally close to ILP (on the datasets we looked at)

Conclusions

A flavor of Probabilistic ILP or Statistical Relational Learning from a logical perspective

A definition of Probabilistic ILP

- based on a probabilistic coverage notion and annotated logic programs

Different Probabilistic ILP setting as for traditional ILP

- Learning from entailment (Parameter Est. SLPs/Prism)
- Learning from interpretations (BLPs, PRMs & Co)
- Learning from traces or proofs (SLPs, RMMs, LOHMMs)
- Discriminative ILP (FOIL + Naïve Bayes or Kernels)

Probabilistic Logic Learning

Excellent Area for Research & Ph.D. topics !

THANK YOU !