# Logical and Relational Learning
## *A novel synthesis*

*Luc De Raedt*
luc.deraedt@cs.kuleuven.be

KATHOLIEKE UNIVERSITEIT LEUVEN

DTAI MACHINE LEARNING

ECML/PKDD 2008

# What is Logical and Relational Learning ?

Inductive Logic Programming

(Statistical) Relational Learning

UNION of

Mining and Learning in Graphs

Multi-Relational Data Mining

They all study the same problem

# The Problem

Learning from structured data, involving

- objects, and
- relationships amongst them

and possibly

- using background knowledge

# Purpose of this talk

- Relational learning is sometimes viewed as a new problem, but it has a long history
- Emphasize the role of symbolic representations (graphs & logic) and knowledge
- A modern view
  - logic as a toolbox for machine learning
- Overview of some of the available tools and techniques
- Illustration of their use in some of our recent work

# Overview

MOTIVATION

REPRESENTATIONS OF THE DATA

The LOGIC of LEARNING

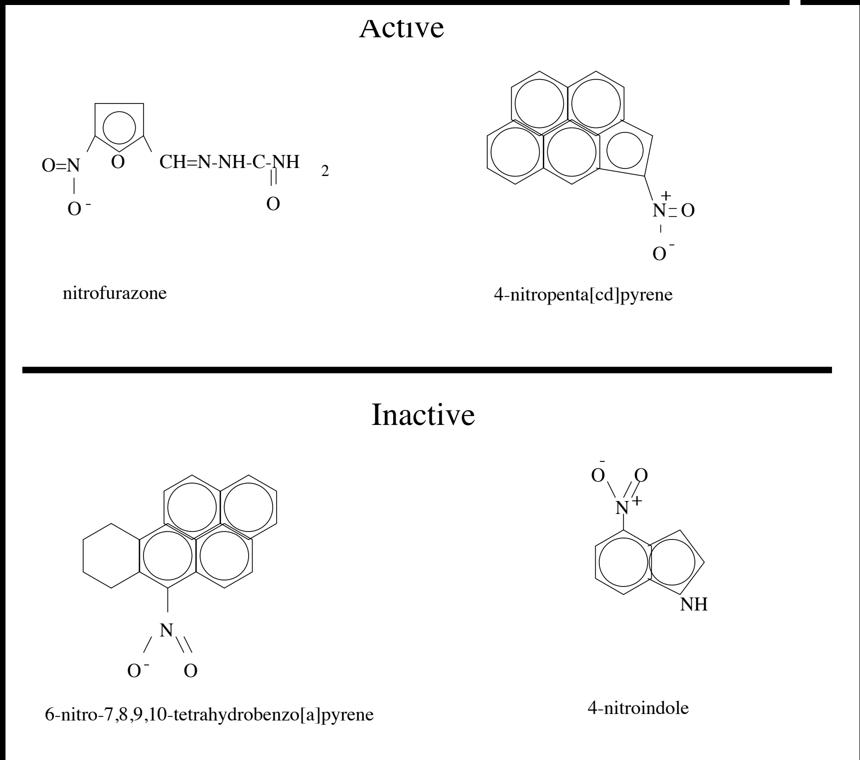
METHODOLOGY and SYSTEMS

LOGIC, RELATIONS and PROBABILITY

ILLUSTRATION in LINK MINING

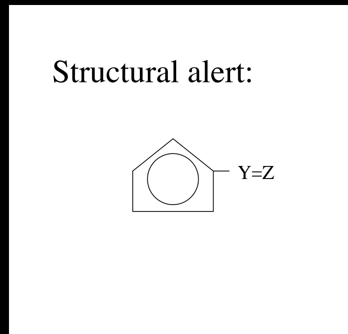
# The MOTIVATION

# Case 1: Structure Activity Relationship Prediction




Data = Set of Small Graphs

[Srinivasan et al. AIJ 96]




General Purpose
Logic Learning System

Uses and Produces
Knowledge

# Using and Producing Knowledge

LRL can use and produce knowledge

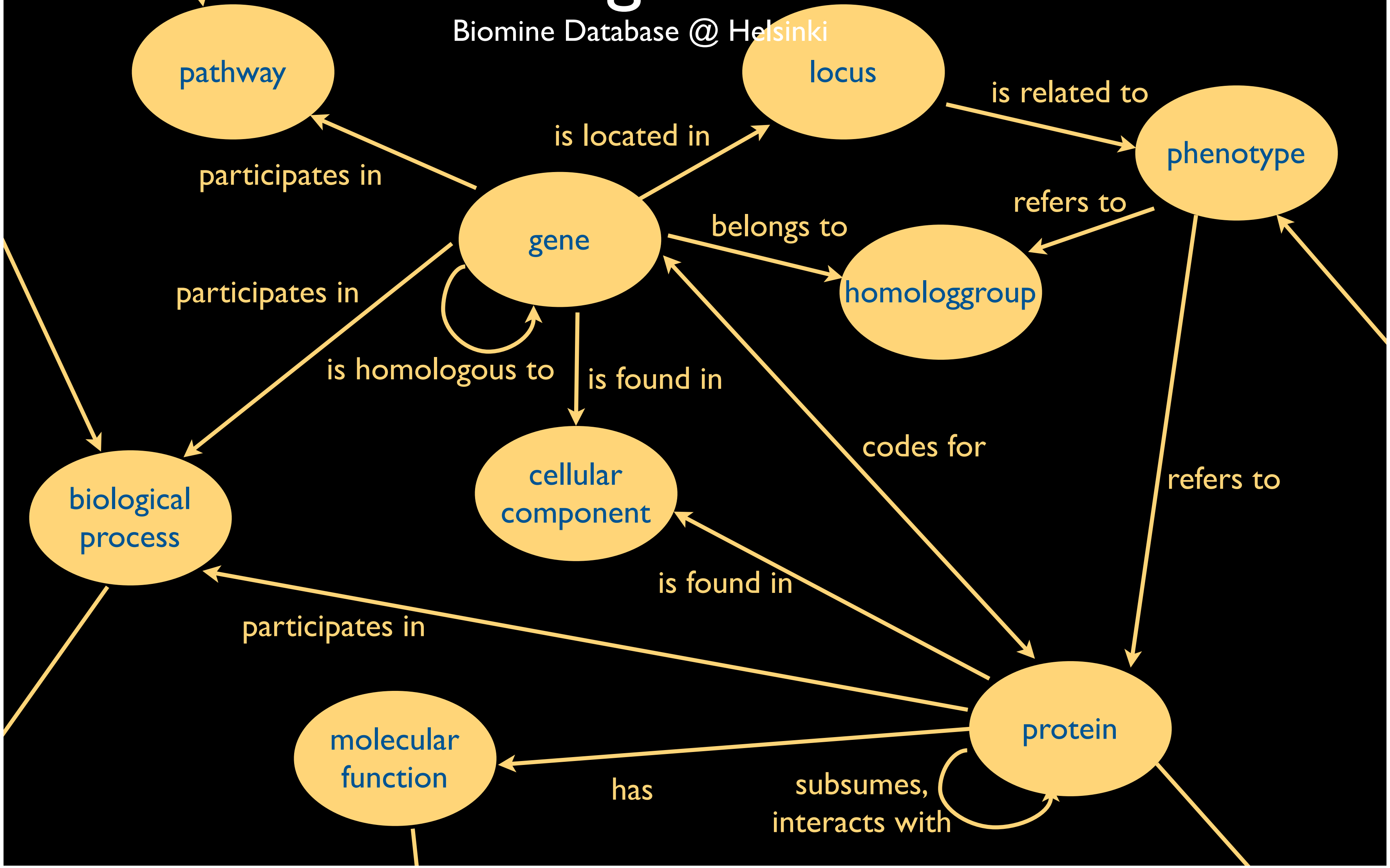
Result of learning task is understandable and interpretable

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

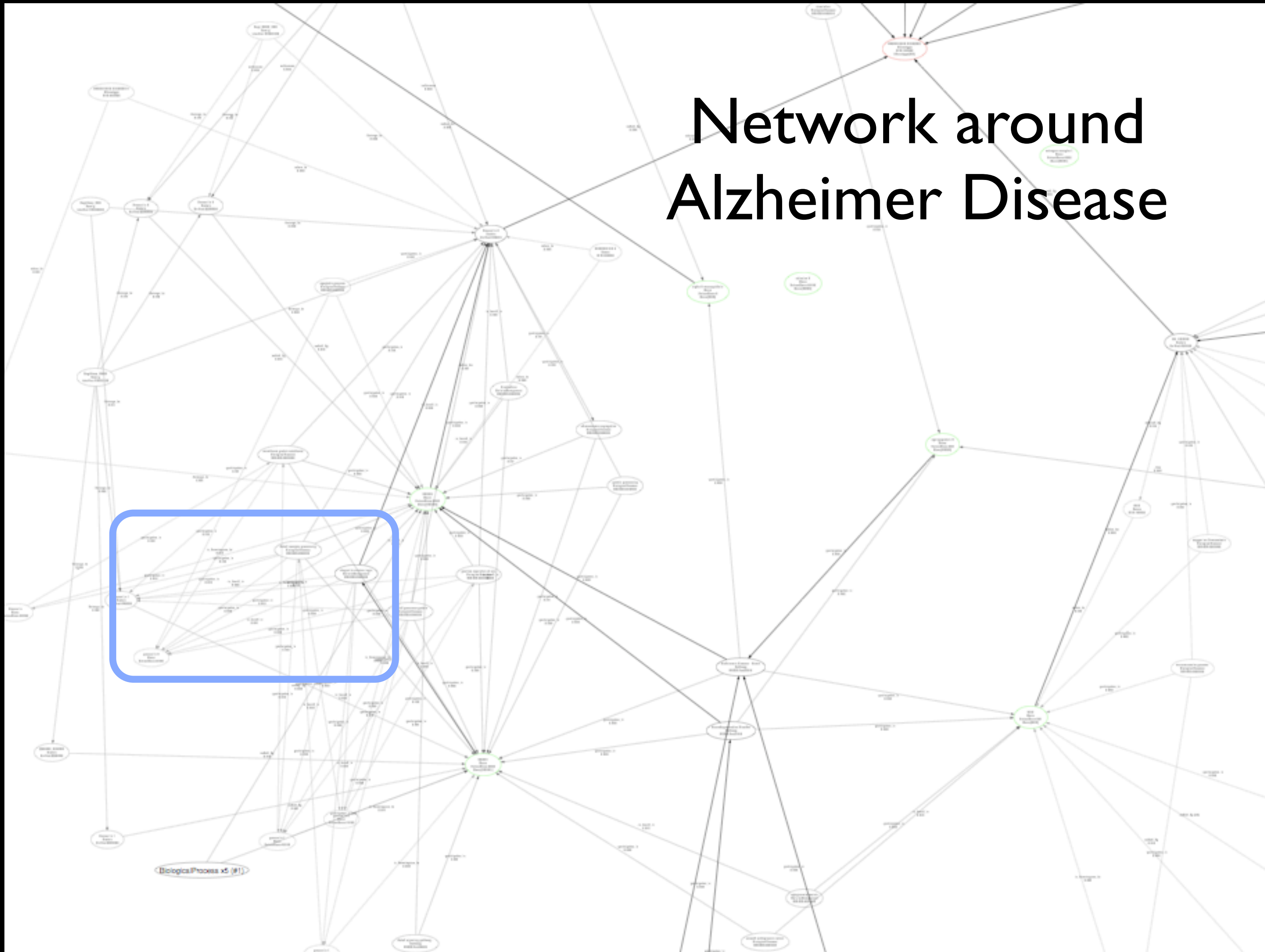
Data = Large (Probabilistic) Network

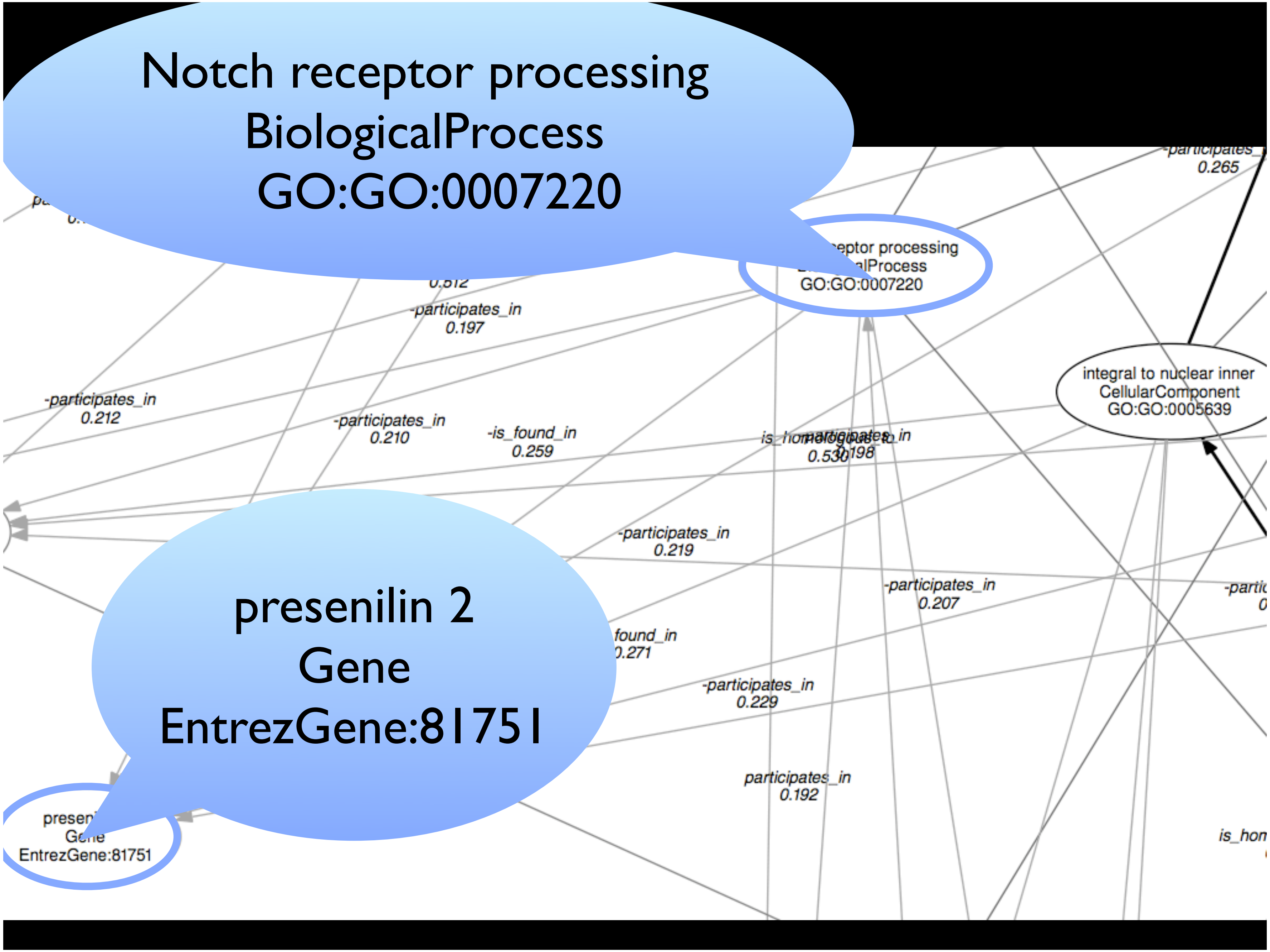
# Case 2: Biological Networks

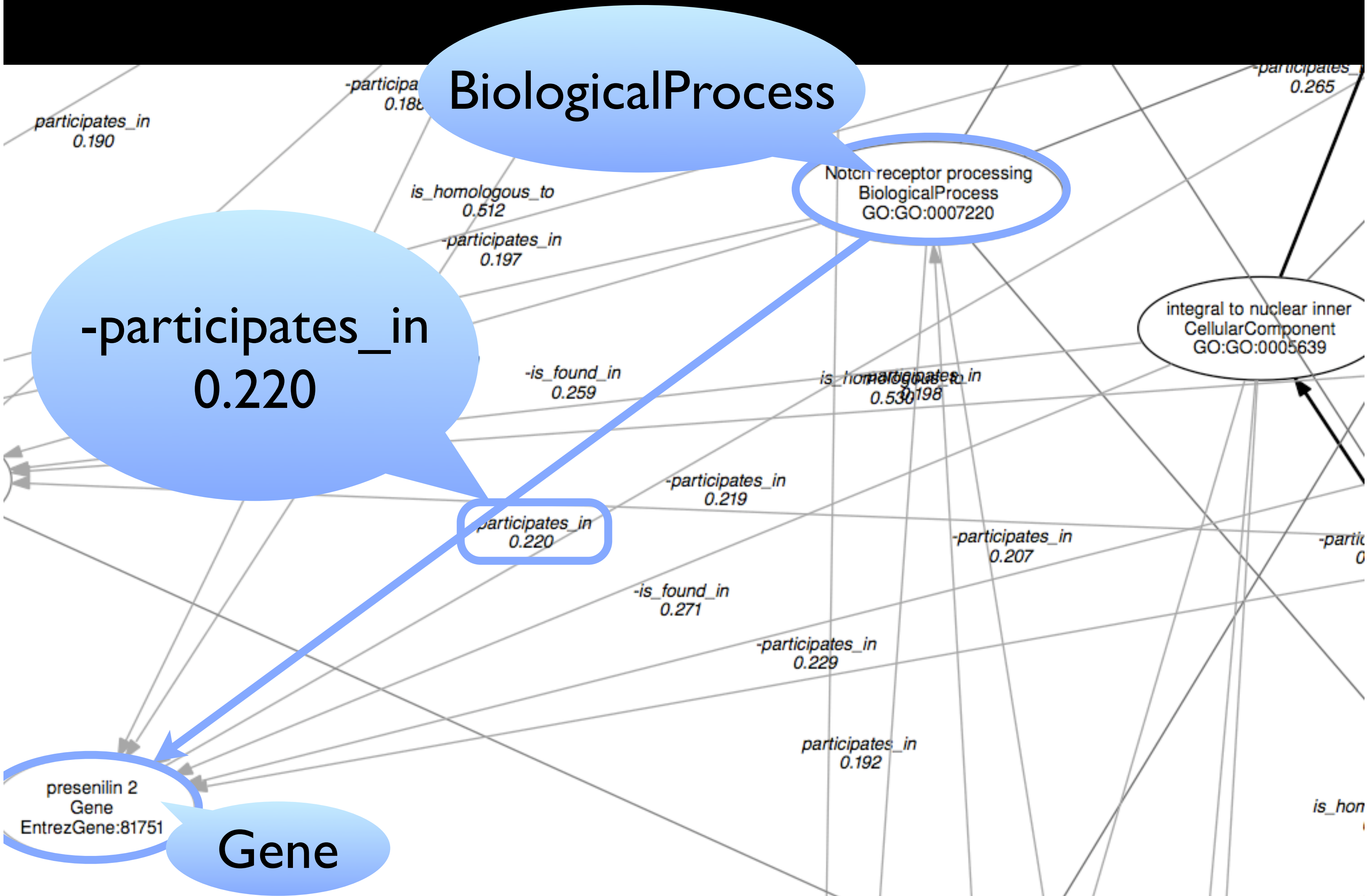
Biomine Database @ Helsinki



# Network around Alzheimer Disease




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

# 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

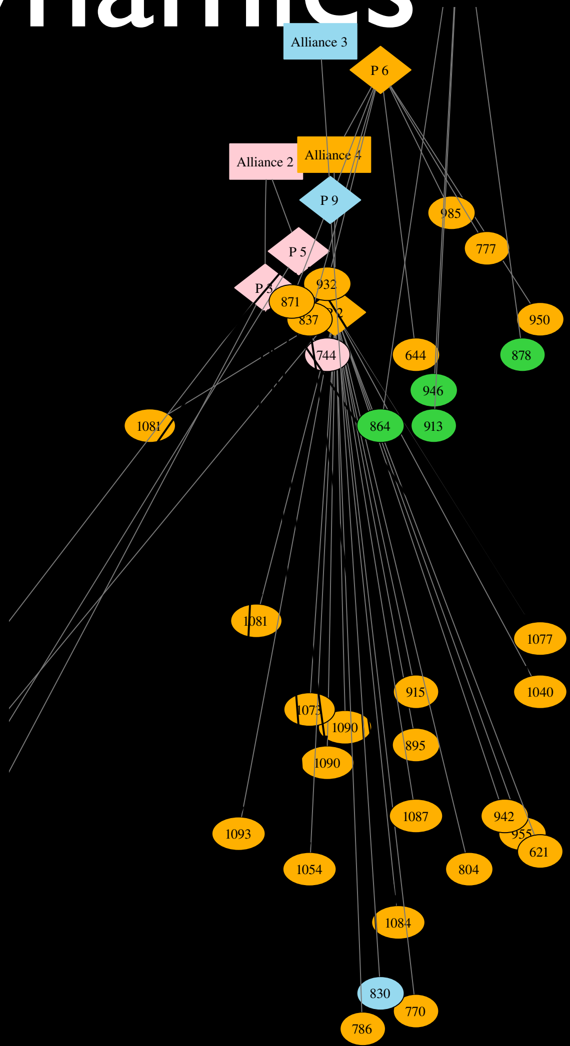
Fragment of world with

~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

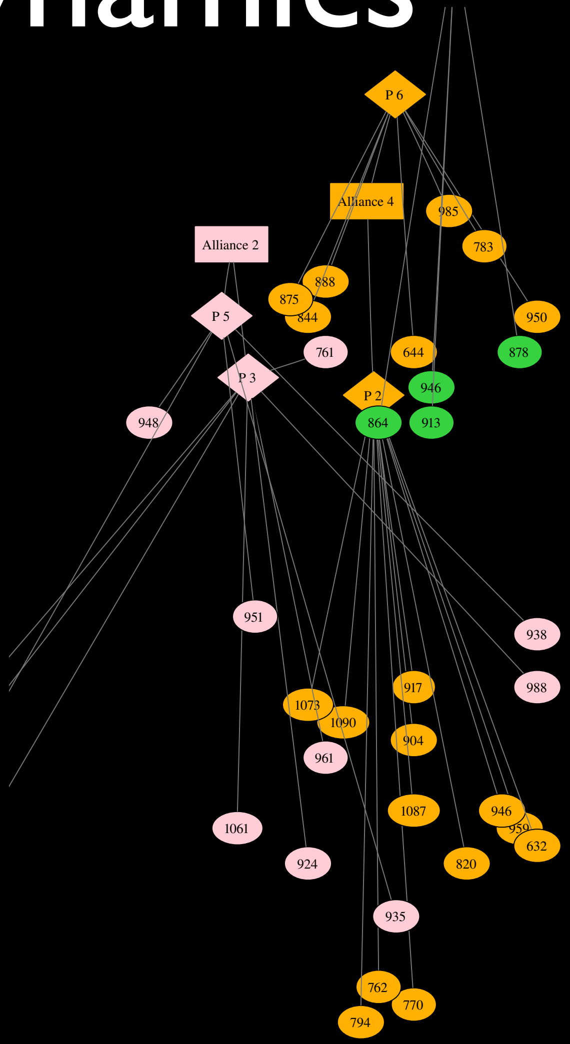
Fragment of world with

~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

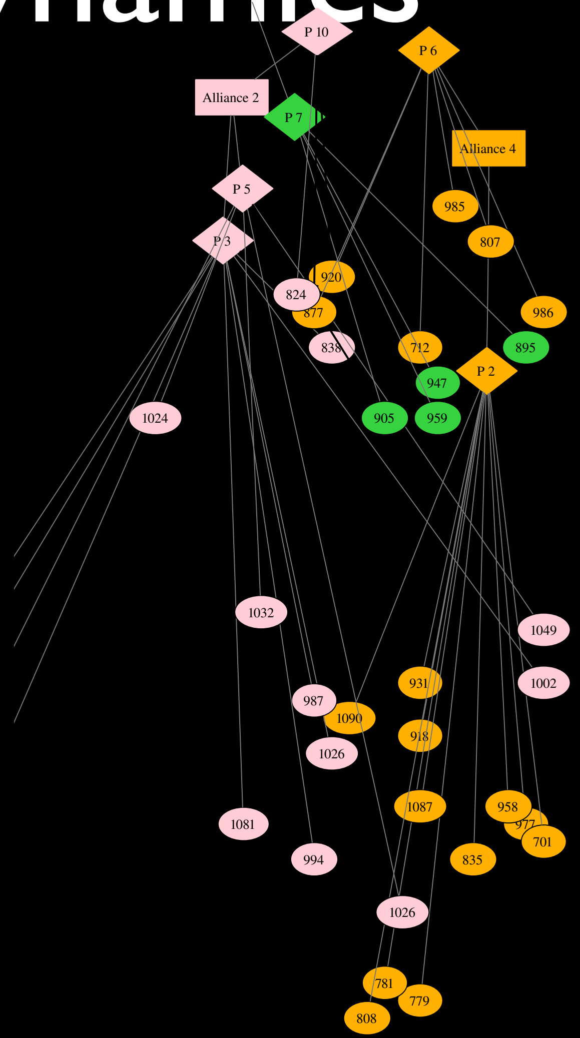
Fragment of world with

~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

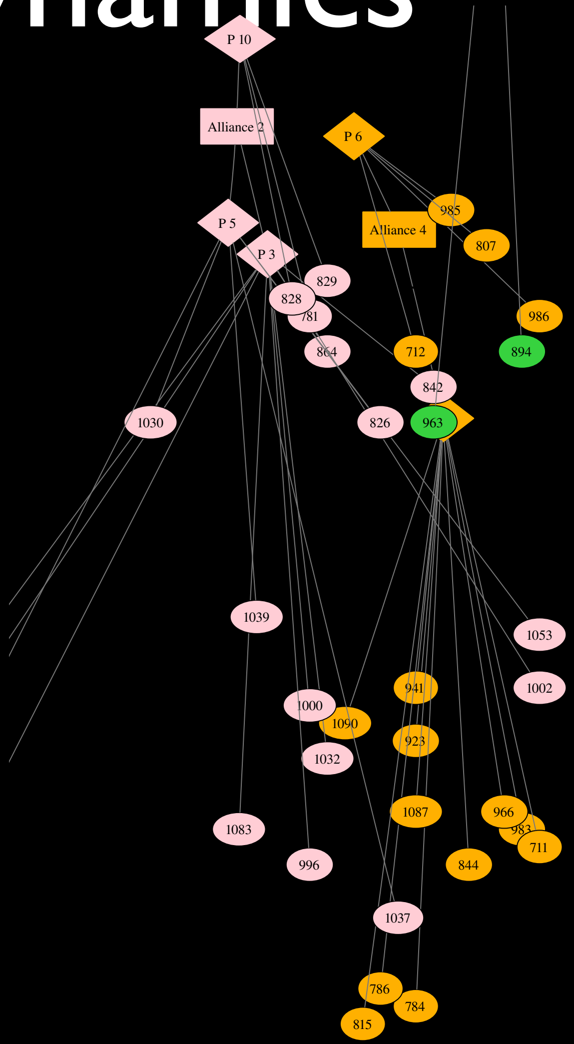
Fragment of world with

~10 alliances
~200 players
~600 cities

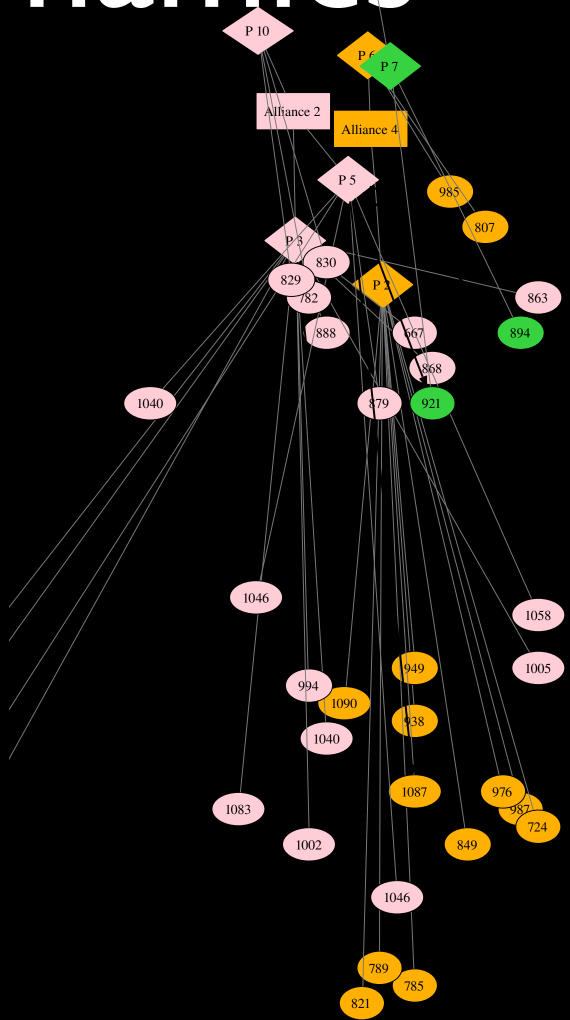
alliances color-coded

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

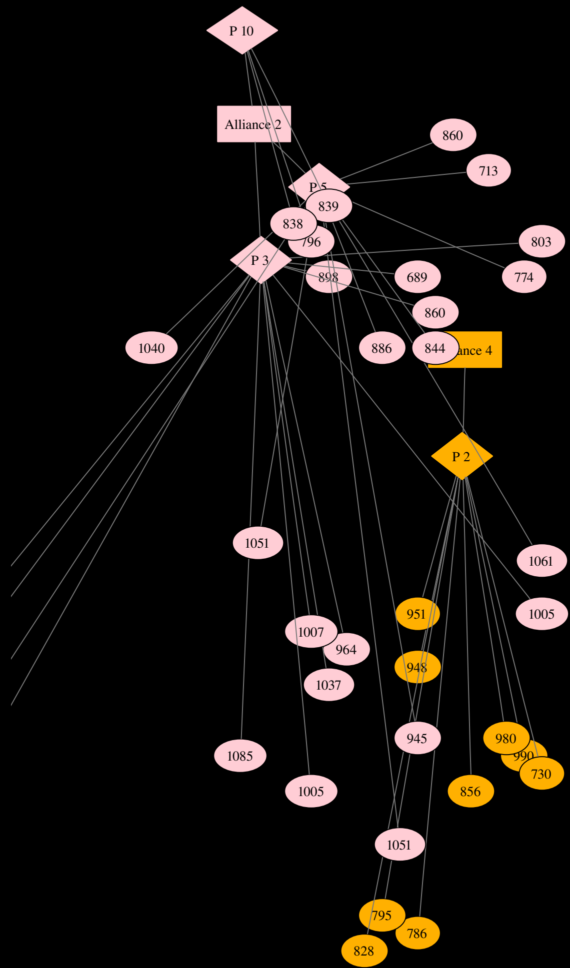
Fragment of world with

~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]



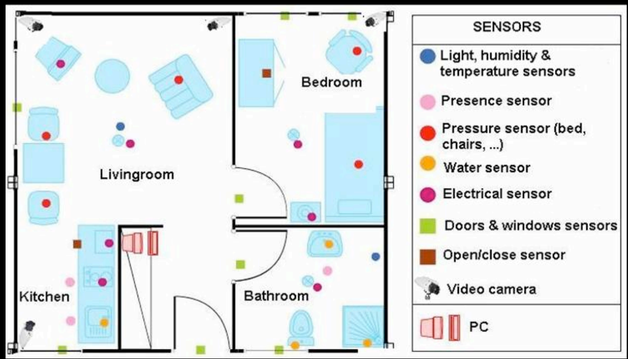
# Emerging Data Sets

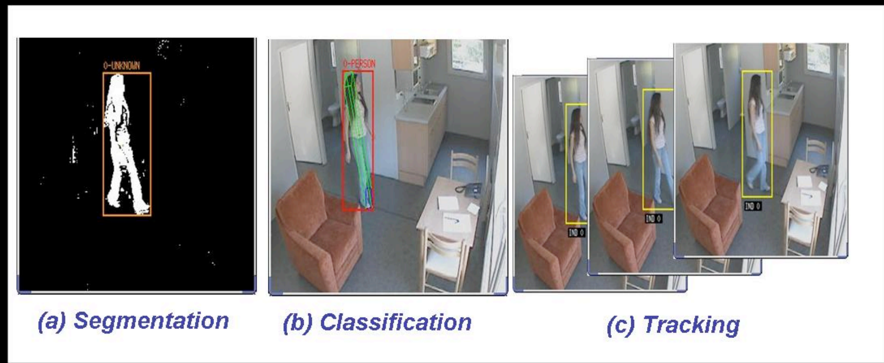
In many application areas :

- 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




(a) Segmentation (b) Classification (c) Tracking

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

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

# The LRL Problem

Learning from structured data, involving

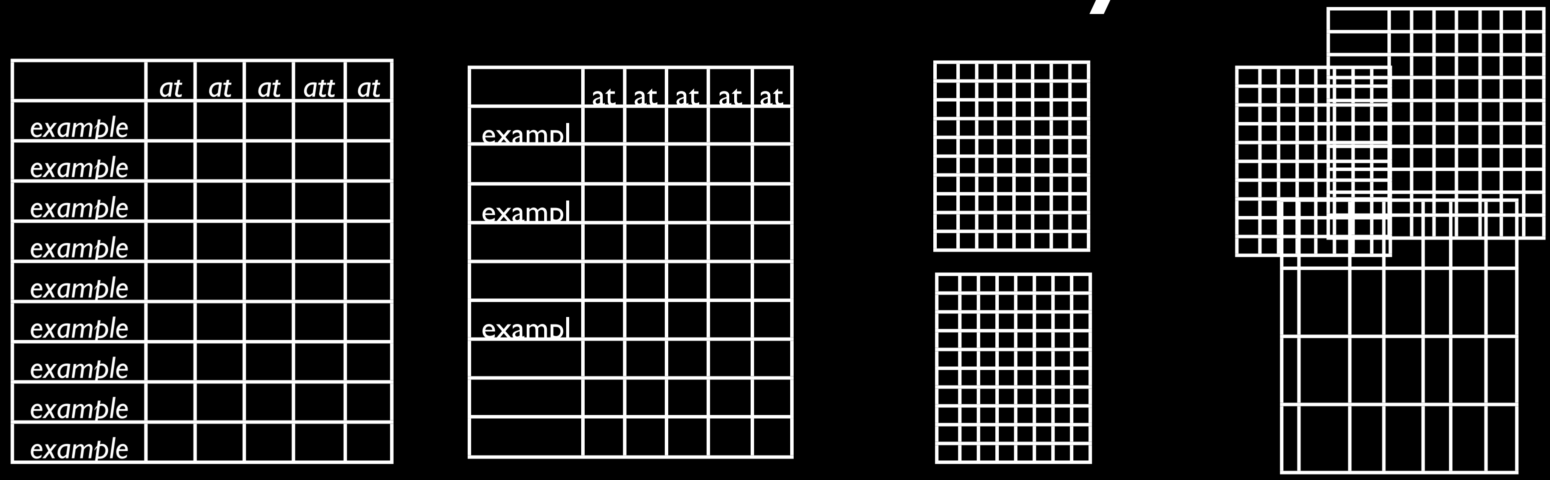
- objects, and relationships amongst them
- possibly using background knowledge

Very often :

- examples are small graphs or elements of a large network (possibly evolving over time)
- many different types of applications  and challenges

# REPRESENTING the DATA

# 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

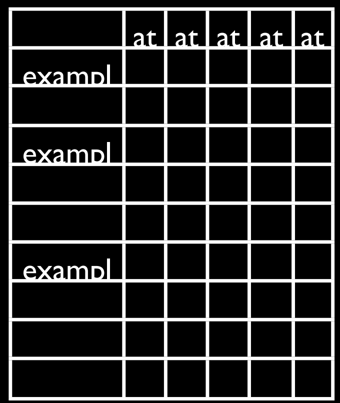
| | *at* | *at* | *at* | *att* | *at* |
|---|---|---|---|---|---|
| *example* | | | | | |
| *example* | | | | | |
| *example* | | | | | |
| *example* | | | | | |
| *example* | | | | | |
| *example* | | | | | |
| *example* | | | | | |
| *example* | | | | | |
| *example* | | | | | |

*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*

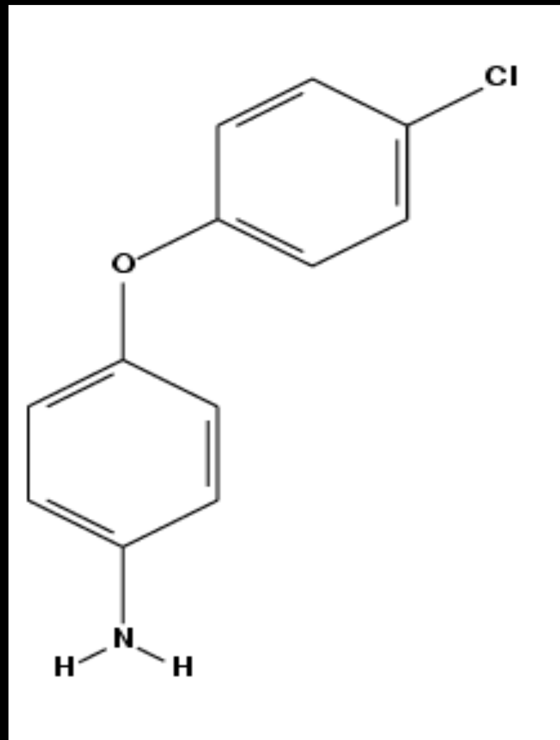
An example is positive if
there exists a tuple in the example
that satisfies particular properties

Boundary case between relational
and propositional learning.

A lot of interest in past 10 years

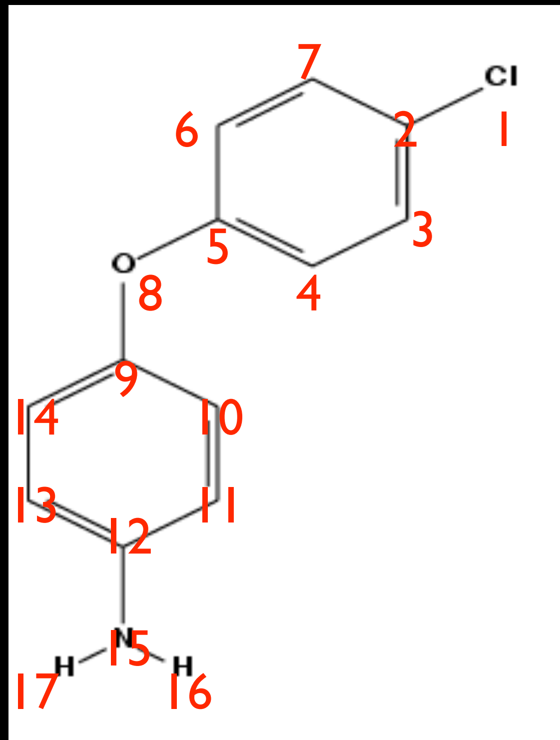
Applications: vision, chemo-informatics, ...

# Encoding Graphs

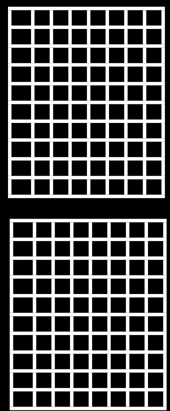


# Encoding Graphs

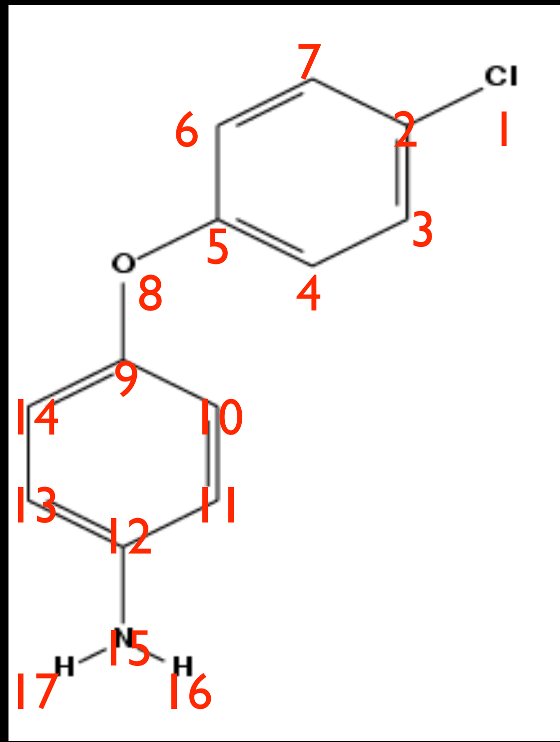
| | |
|---|---|
| atom(1,cl). | |
| atom(2,c). | |
| atom(3,c). | |
| atom(4,c). | |
| atom(5,c). | |
| atom(6,c). | bond(3,4,s). |
| atom(7,c). | bond(1,2,s). |
| atom(8,o). | bond(2,3,d). |
| ... | ... |



# Encoding Graphs



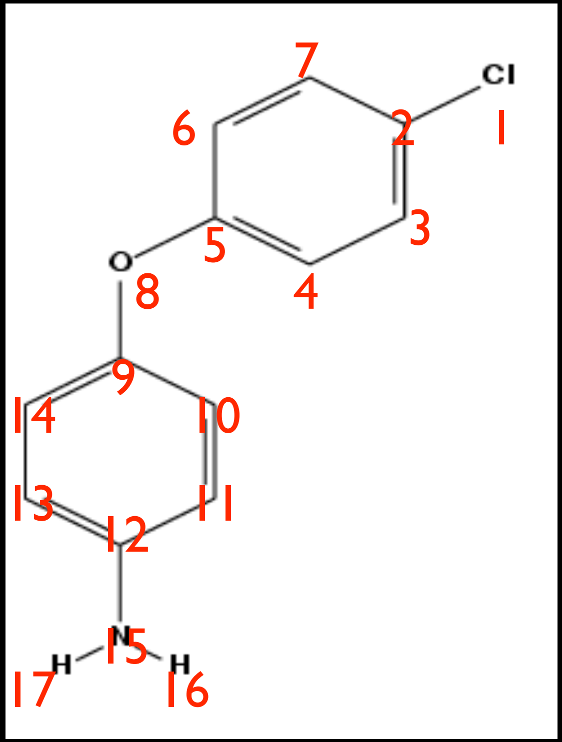
*2 relations*
*edge / vertex*
*graphs & networks*



# 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)

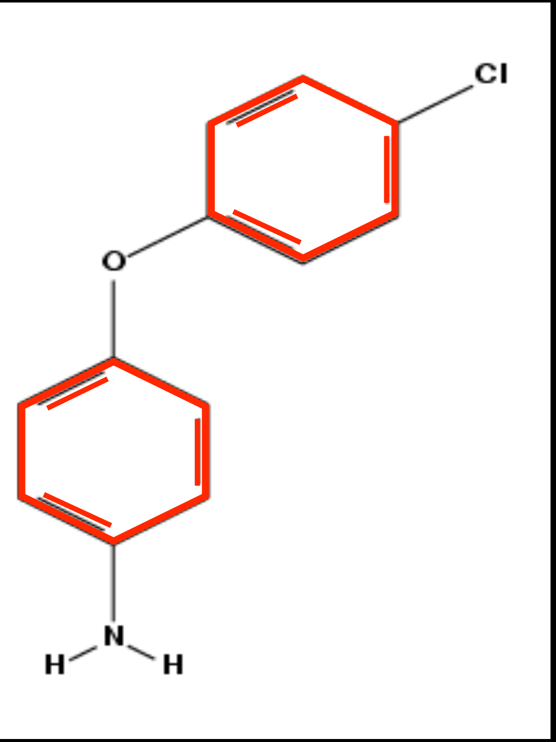
  halogen(A):- atom(X,br)

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

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

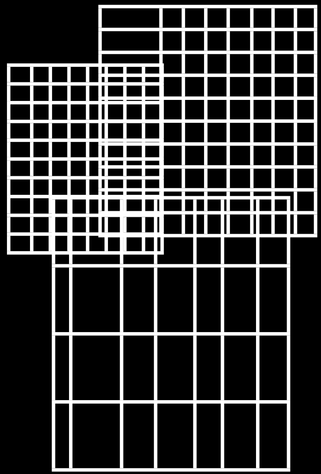
- encode functional group

  benzene-ring :- ...

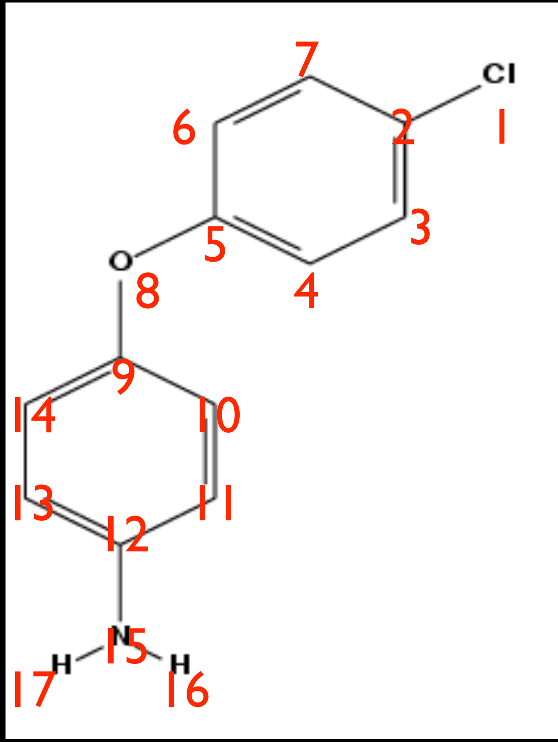



intentional versus extentional encodings

# Relational Representation



*multi-table*
*multiple-tuple*
*relational*



# Relational versus Graphs

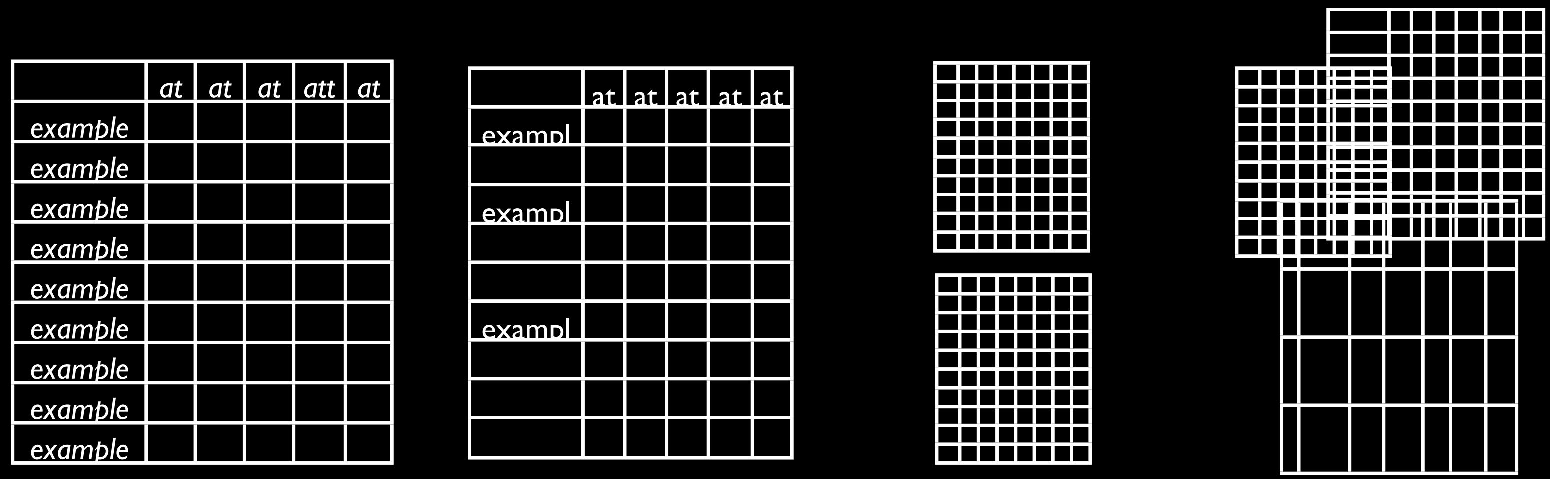
Advantages Relational

- 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-table* | *2 relations* | *multi-table* |
| *single-tuple* | *multiple-tuple* | *edge / vertex* | *multiple-tuple* |
| *attribute-value* | *multi-instance* | *graphs & networks* | *relational* |

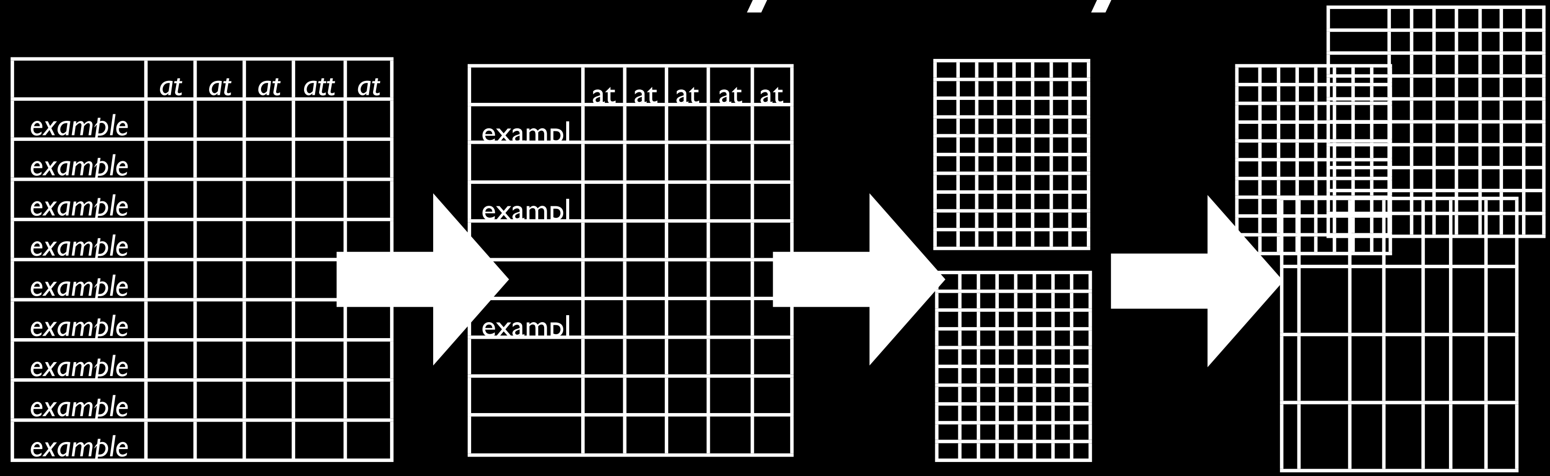
# Two questions

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)

Sometimes uses AGGREGATION

# Representational Hierarchy -- Systems




| *single-table* | *single-table* | *2 relations* | *multi-table* |
|---|---|---|---|
| *single-tuple* | *multiple-tuple* | *edge / vertex* | *multiple-tuple* |
| *attribute-value* | *multi-instance* | *graphs & networks* | *relational* |

# The Upgrading Methodology

Start from existing system for simpler representation

Extend it for use with richer representation (while trying to keep the original system as a special case)

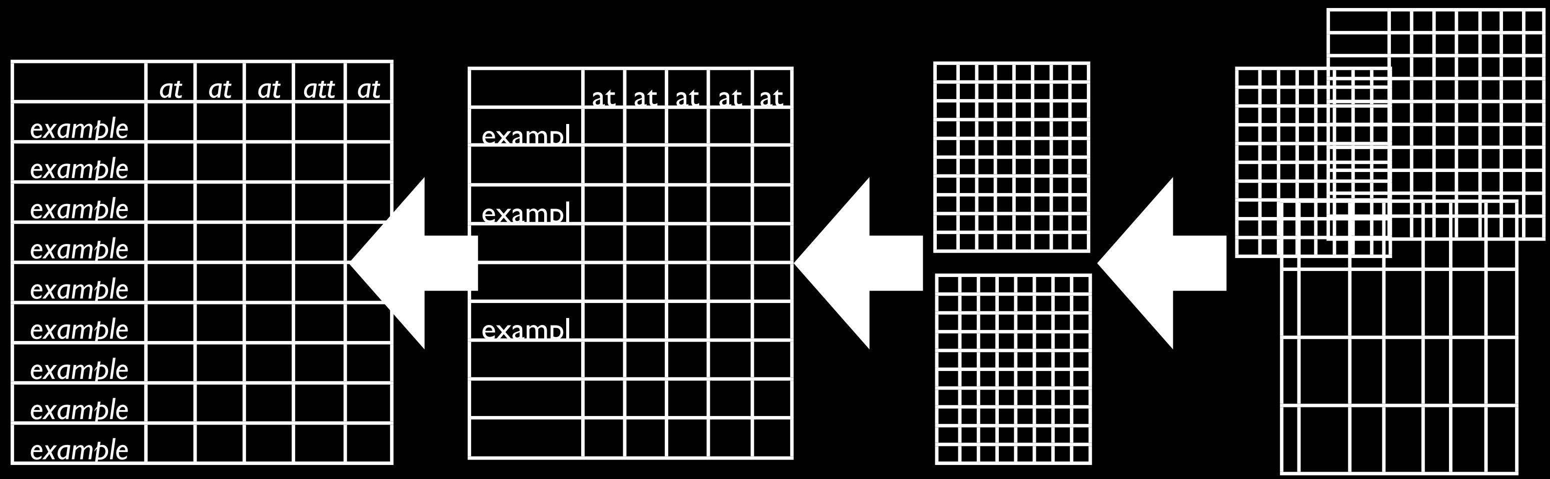
Illustrations follow.

# 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

# Propositionalization




| single-table | single-table | 2 relations | multi-table |
|---|---|---|---|
| single-tuple | multiple-tuple | edge / vertex | multiple-tuple |
| attribute-value | multi-instance | graphs & networks | relational |

Downgrading the data ?

# Propositionalization

**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 |

# Table-based Propositionalization

Define new relation

p(N,J,C,P,R,Co,L) :-

participant(N,J,C,P,R),

subscribes(N,Co),

length(Co,L).

**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 |

**COURSE Table**

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

**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 |

Multi-relational → multi-instance

under certain conditions → atttribute-value

# Query-based Propositionalization

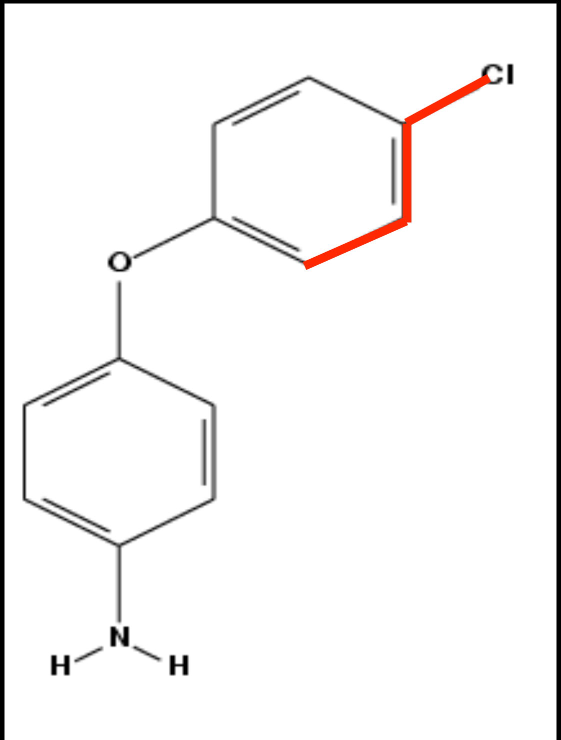
Compute a set of relevant features or queries.

Typically, (variant of) local pattern mining.

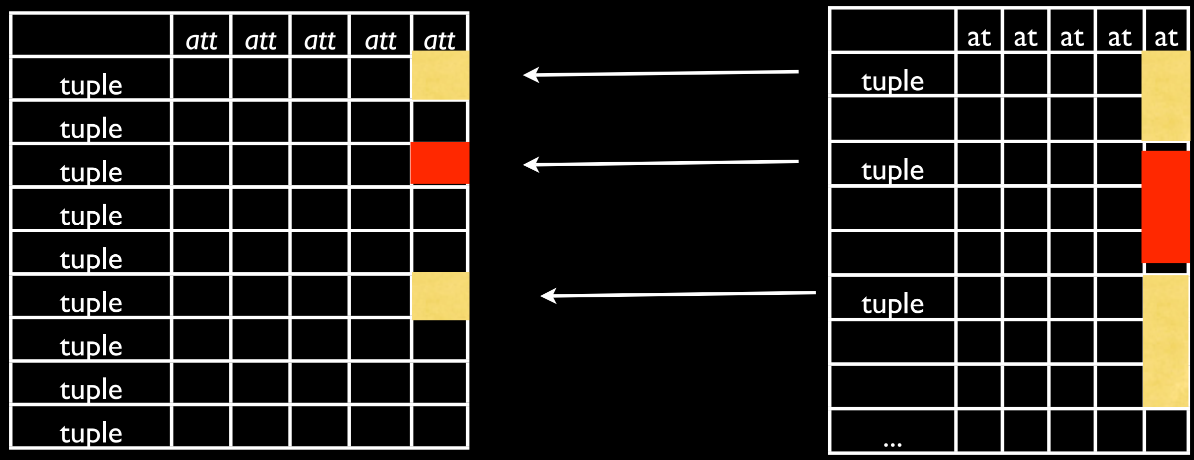
E.g. find all frequent or correlated subgraphs.

Use each feature as boolean attribute.

Good results in graph classification (using SVMs).



# Aggregation




*from multi-tuple relations to single-tuple*

# Aggregation

Introduce new attribute

For instance :

- number of courses followed

| 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 |

adams, 3

multi-instance/tuple → attribute-value

# Propositionalization and Aggregation

Often useful to reduce more expressive representation to simpler one but almost always results in information loss or combinatorial explosion

Shifts the problem

- how to find the right features / attributes

One example

- features = paths in a graph (for instance)
- which ones to select ?

still requires "relational" methods

# The LOGIC of LEARNING
# Coverage and Generality

# Typical Machine Learning Problem

**Given**

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

**Find**

- A hypothesis **h** in **Lh** that minimizes the loss function w.r.t. the examples **E** taking **B** into account

# The Hypothesis Language

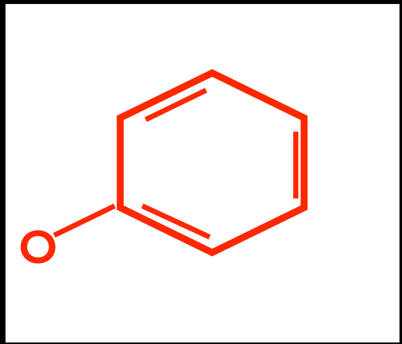
Prolog OWL First Order Logic

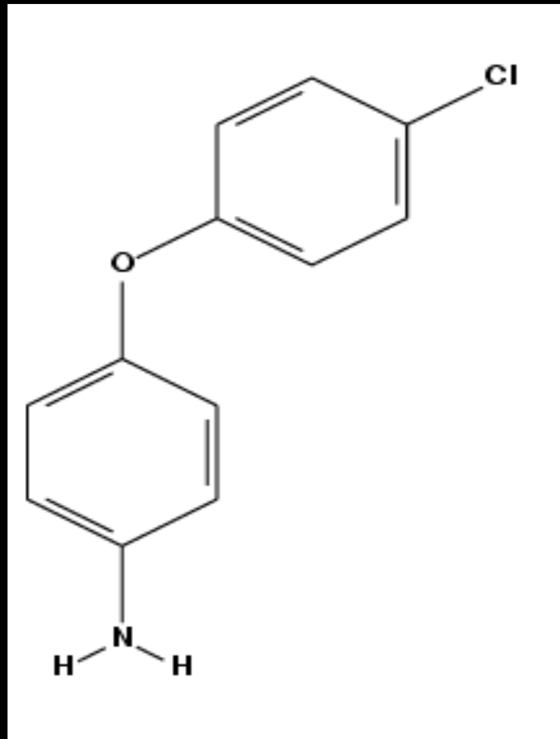
Graphs SQL Description Logic

Relational Calculi Entity-Relationship Model

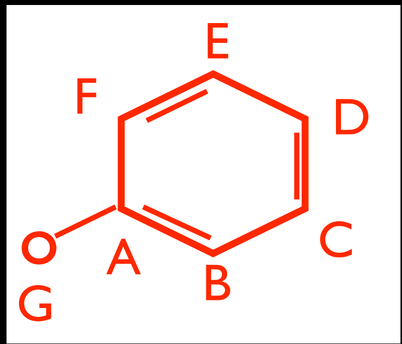
Choice probably not that important
though implementation & manipulation

# Covers Relation

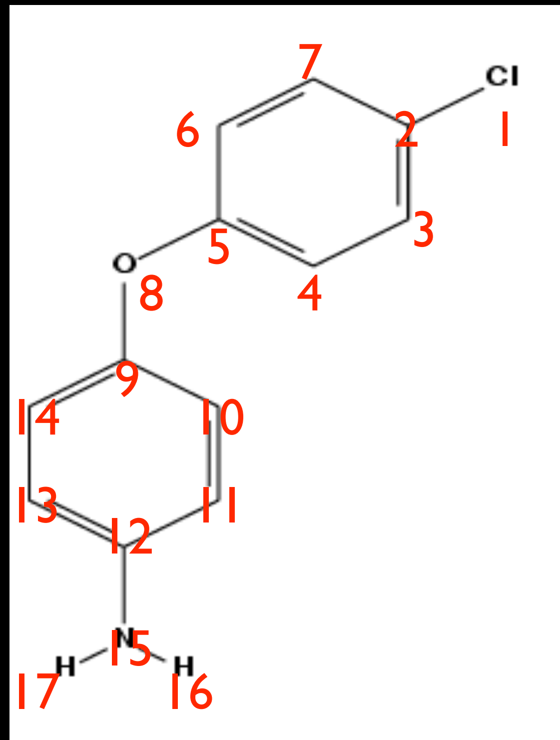


# Covers Relation




Subgraph Isomorphism
(bijection)
or
Homomorphism
(injection)



# Coverage

```
positive :-  atom(A,c),
             atom(B,c),
             bond(A,B,s),
             ....
```

OI-subsumption
(bijection)
or
theta-subsumption
(injection)

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

# Coverage

positive :- halogen(A),
halogen(B),
bond(A,B,s),
....

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)

Deduction

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).
...

# Generality Relation

An essential component of Symbolic Learning systems

G is more general than S if all examples covered by S are also covered by G

Using graphs

- subgraph isomorphism or homeomorphism

In logic

- theta or OI subsumption, in general $G \models S$

# Generality Relation

positive :- atom(X,c) $\models$ positive :- atom(X,c), atom(Y,o)

but also

positive :- halogen(X)

halogen(X) :- atom(X,c)

$\models$ positive :- atom(X,c)

# $G \models S$

S follows *deductively* from G

G follows *inductively* from S

therefore induction is the *inverse* of deduction

this is an operational point of view because there are many deductive operators $\vdash$ that implement $\models$

take any deductive operator and invert it and one obtains an inductive operator

# Various frameworks for generality

Depending on the form of G and S

- single clause
- clausal theory

Relative to a background theory $B \cup G \models S$

Depending on the choice of $\vdash$ to invert

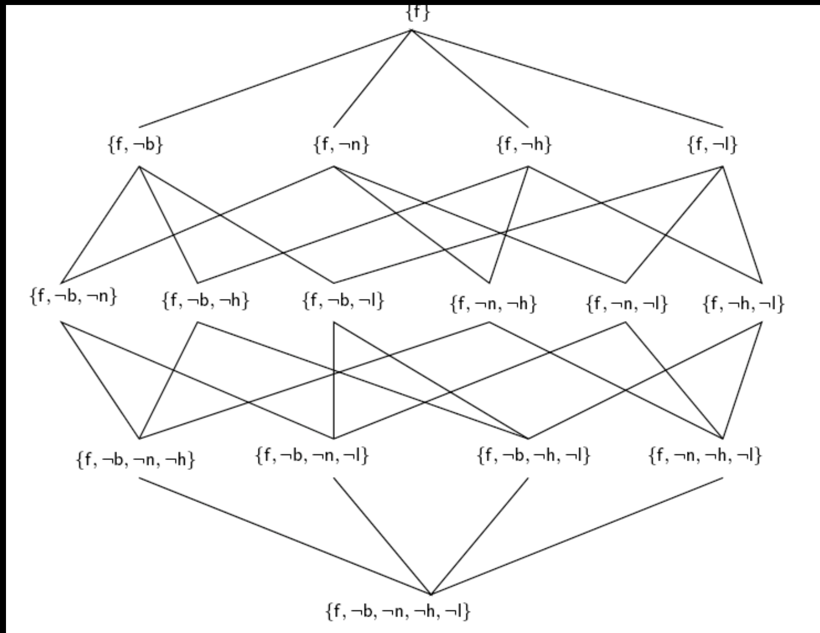
- subsumption (most popular)

# Subsumption in 3 Steps

Subsumption ~ generalization of graph morphisms

1. propositional
2. atoms
3. clauses (rules)

# Propositional Logic




{f,¬b,¬n} = f IF b and n = f :- b, n

$G \models S$ if and only if $G \subseteq S$

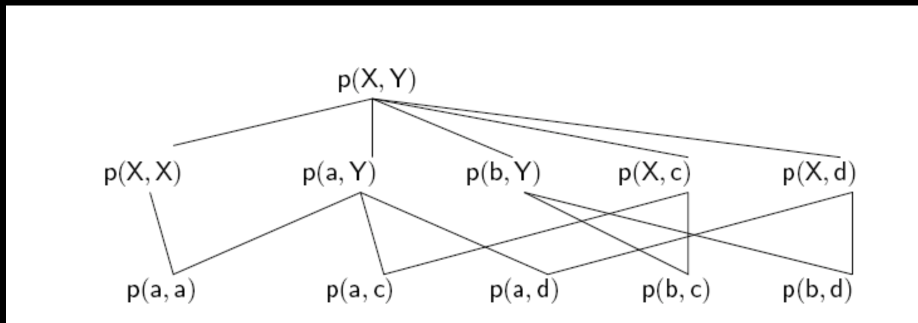
just like item-sets

# Logical Atoms

Does g=participant(adams, X, kul) match

s=participant(adams,researcher, kul) ?

Yes, because there is a substitution θ={X/researcher} such that gθ=s




more complicated, account for variable unification

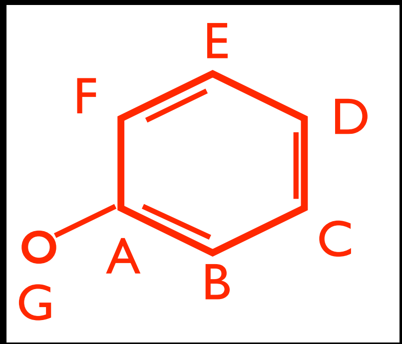
# Subsumption in Clauses

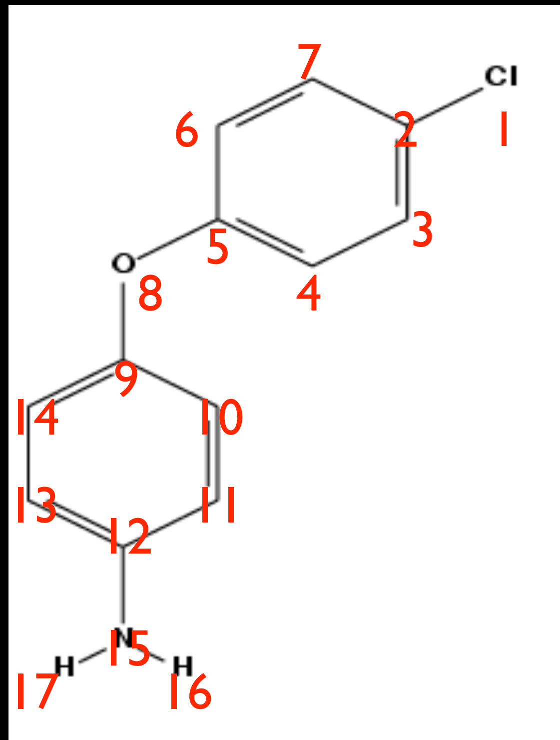
Combine propositional and atomic subsumption.

G subsumes S if and only if there is a substitution $\theta$ such that $G\theta \subseteq S$.

Graph - homeomorphism as special case

# Subsumption Relation




Subgraph Isomorphism
(bijection)
or
Homomorphism
(injection)

θ={G/8,A/5,B/4,C/3,D/2,E/7,F/6}

# Subsumption

positive :- atom(A,c),
atom(B,c),
bond(A,B,s),
....

OI-subsumption
(bijection)
or
theta-subsumption
(injection)

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).
...

$\theta$={G/8,A/5,B/4,C/3,D/2,E/7,F/6}

# Subsumption

Well-understood and studied, but complicated.

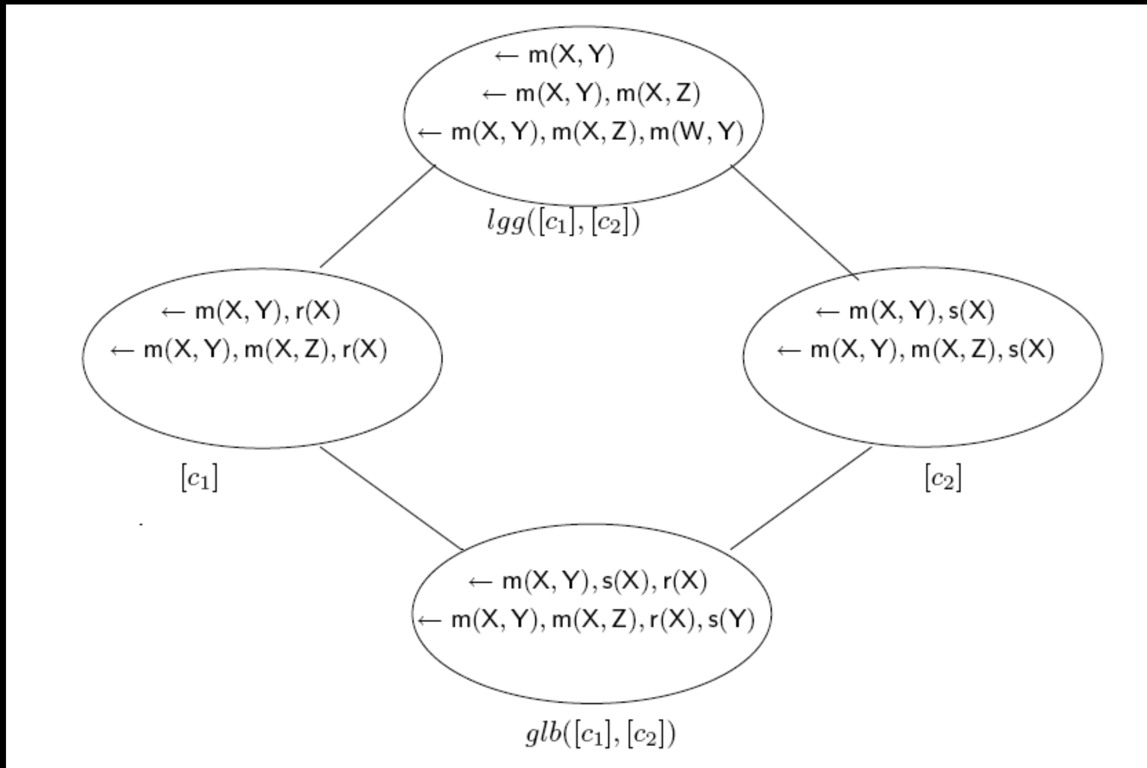
Testing subsumption (and subgraph-ismorphism) is NP-complete

Infinite chains (up and downwards exist)

Syntactic variants exist when working with homeomorphism (but not for isomorphism).

Computation of lub (lgg) and glb

# Theta-subsumption lattice

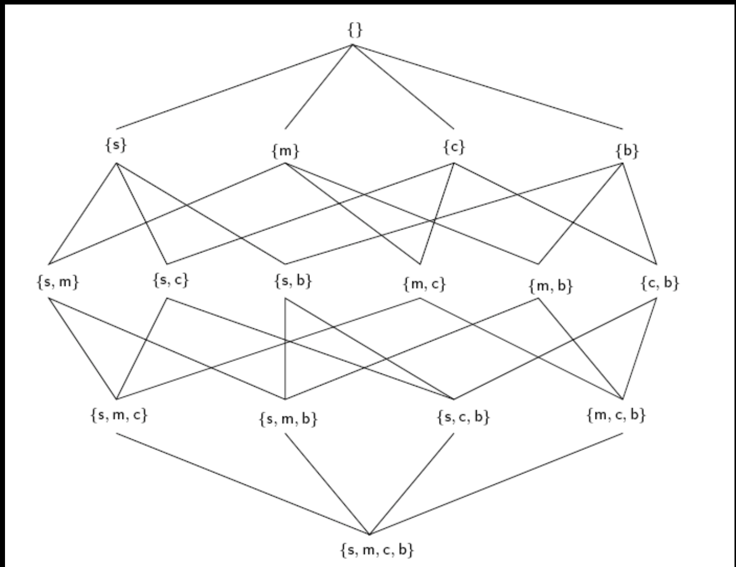



subgraph homeomorphism

# Using Generality

To define the search space that is traversed.

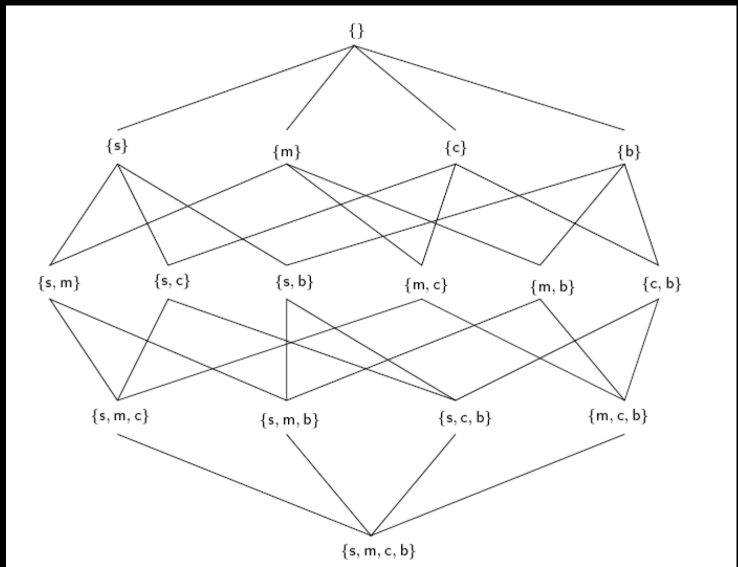
Cf. frequent item-set mining, concept-learning.

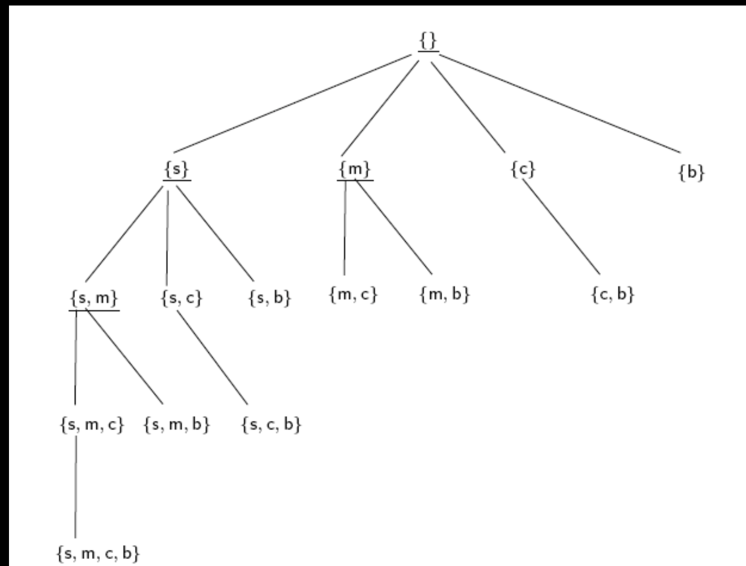


# Generality

Different types of search strategy:

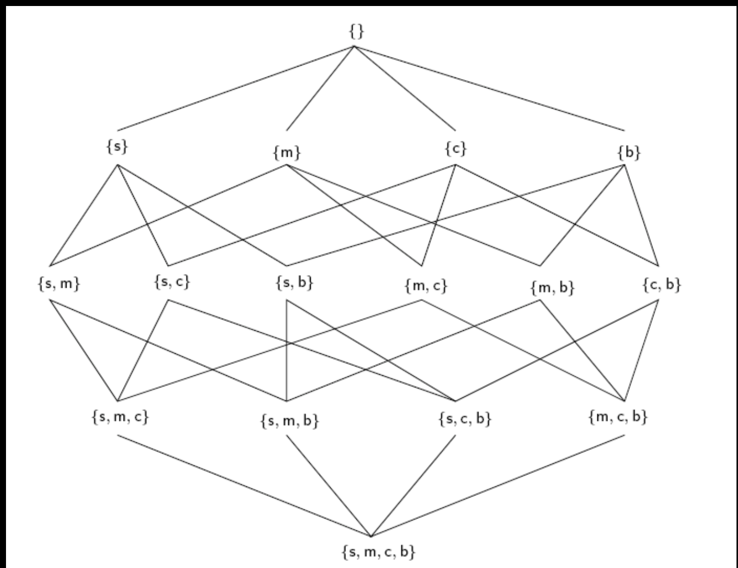
all solutions (freq. item-sets), top-k solutions (branch and bound algo.), heuristic (concept-learning)

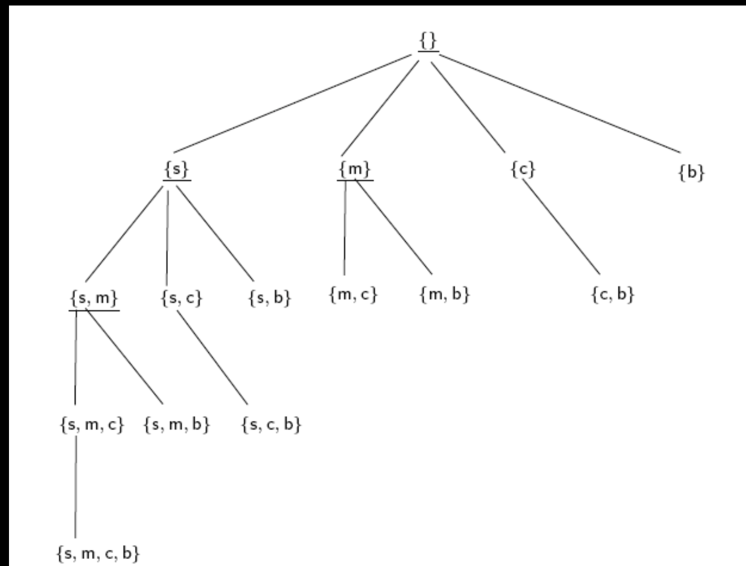


# $G \models S$

Generality relations and refinement operators are well-understood; they apply to simpler structures such as graphs (canonical form -- lexicographic orders)



# Refinement

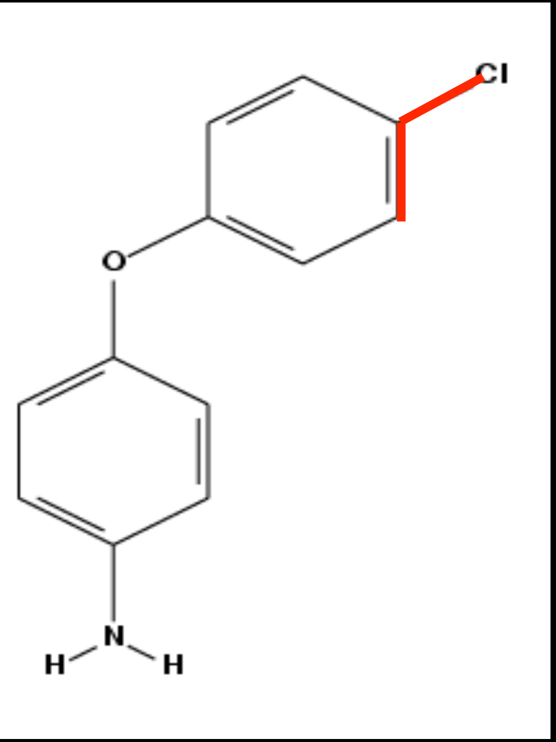
Graphs :

Adding edges

Relational learning

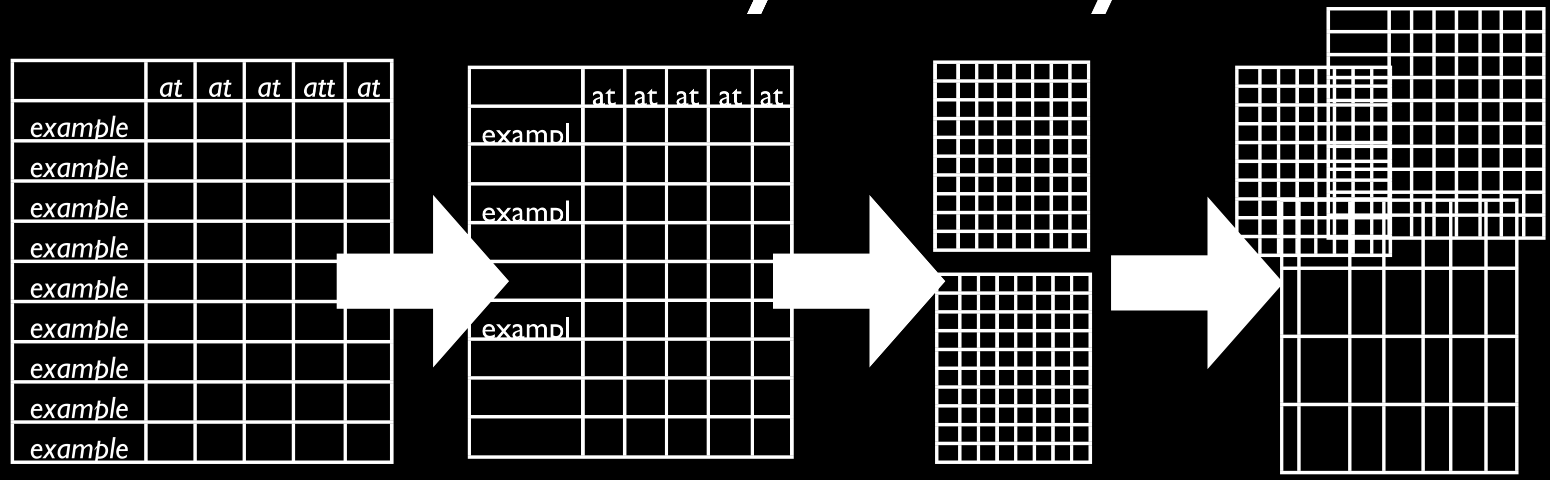
Adding literals

bond(A,B,s), bond(B,C,d), ...



# SYSTEMS & METHODOLOGY

# Representational Hierarchy -- Systems




| single-table | single-table | 2 relations | multi-table |
|---|---|---|---|
| single-tuple | multiple-tuple | edge / vertex | multiple-tuple |
| attribute-value | multi-instance | graphs & networks | relational |

## UPGRADING

# Two messages

LRL 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 $B \cup H \models e$

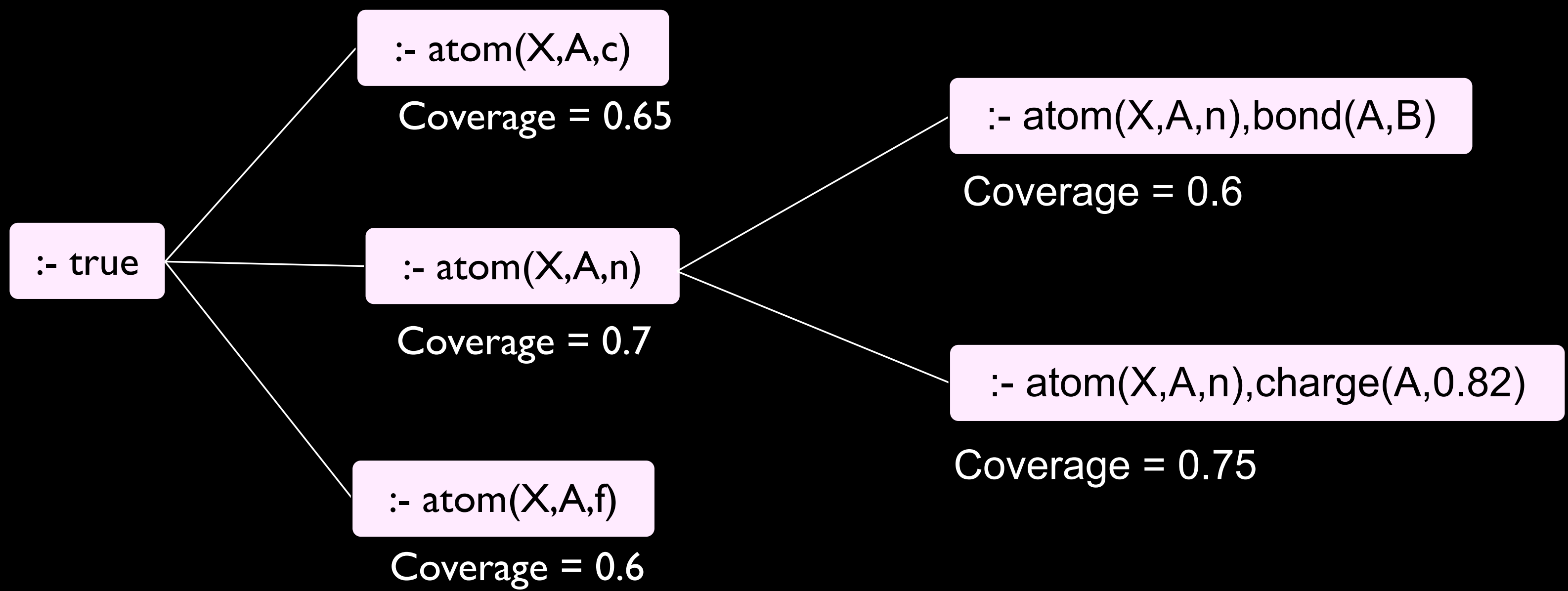
## Find

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

rules

# Searching for a rule

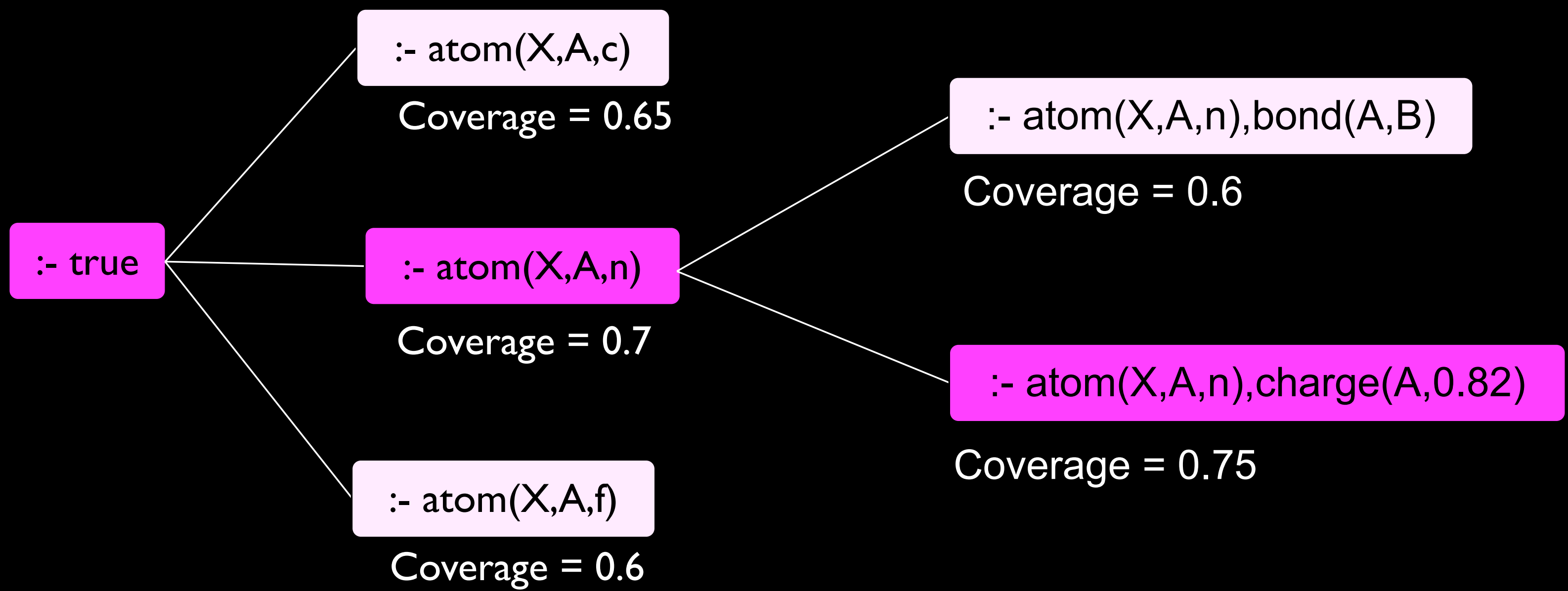
Greedy separate-and-conquer for rule set
Greedy general-to-specific search for single rule



# Searching for a rule

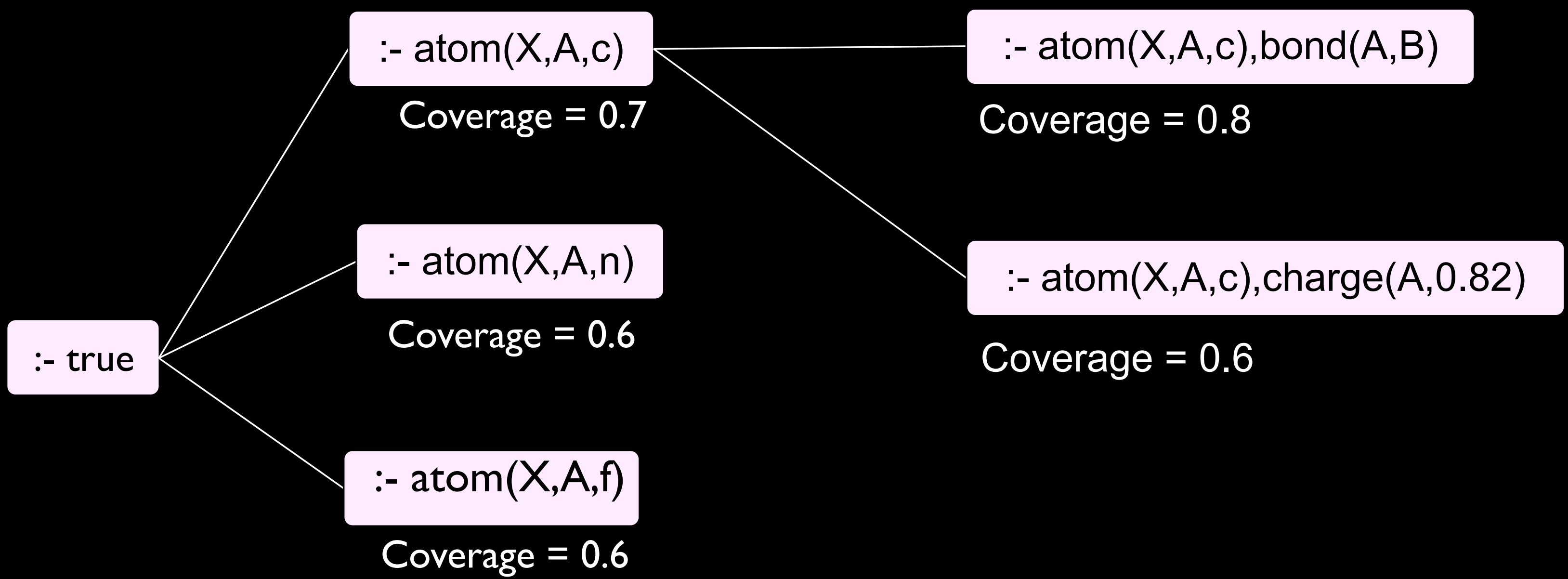
Greedy separate-and-conquer for rule set

Greedy general-to-specific search for single rule



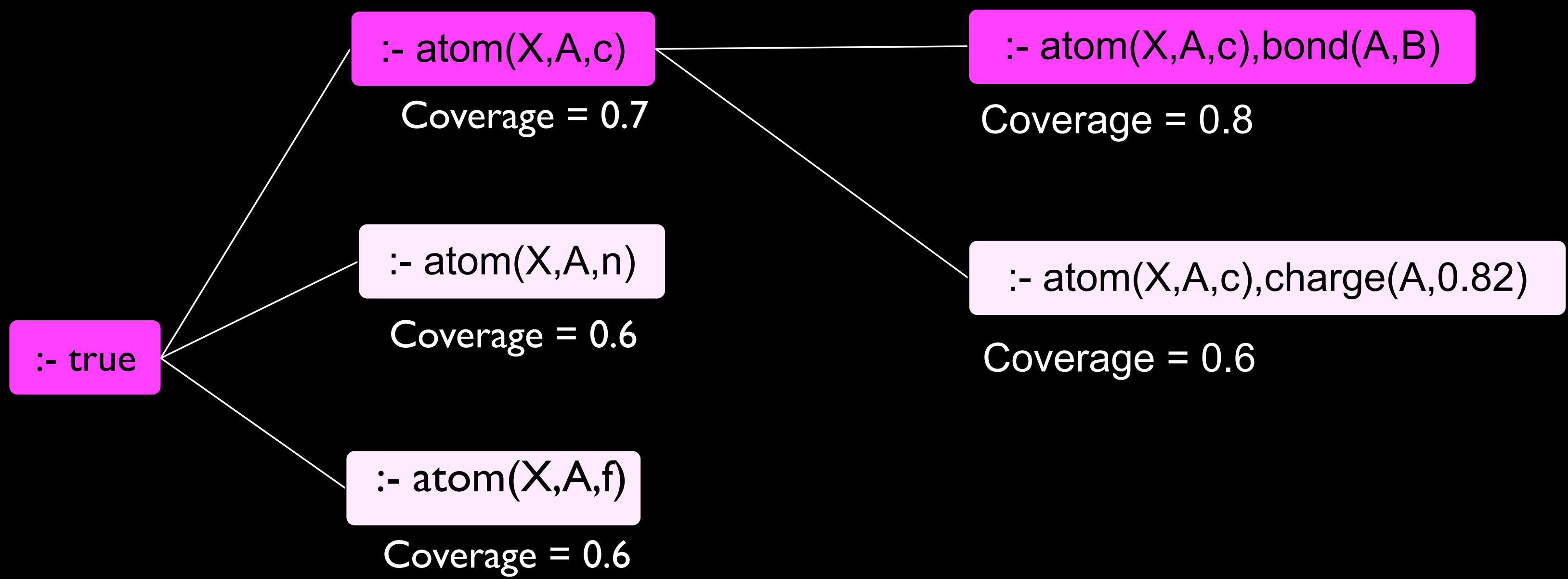

mutagenic(X) :- atom(X,A,n),charge(A,0.82)

# 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)

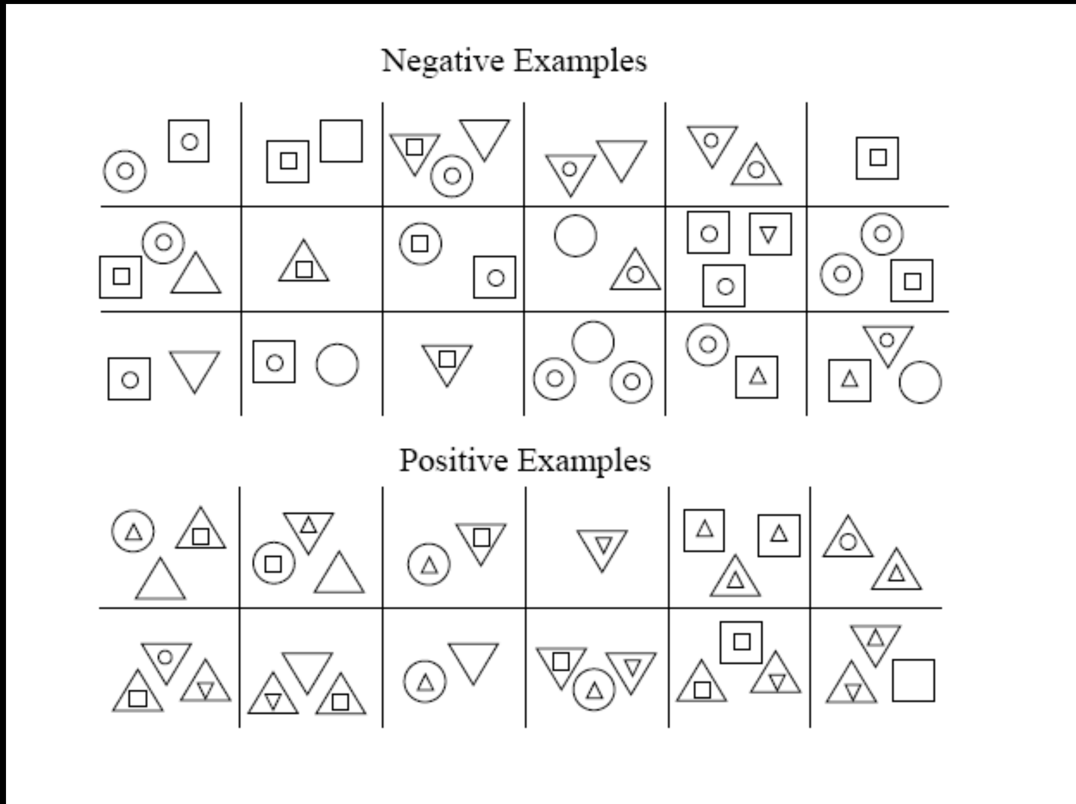
# FOIL

Key ideas / contributions

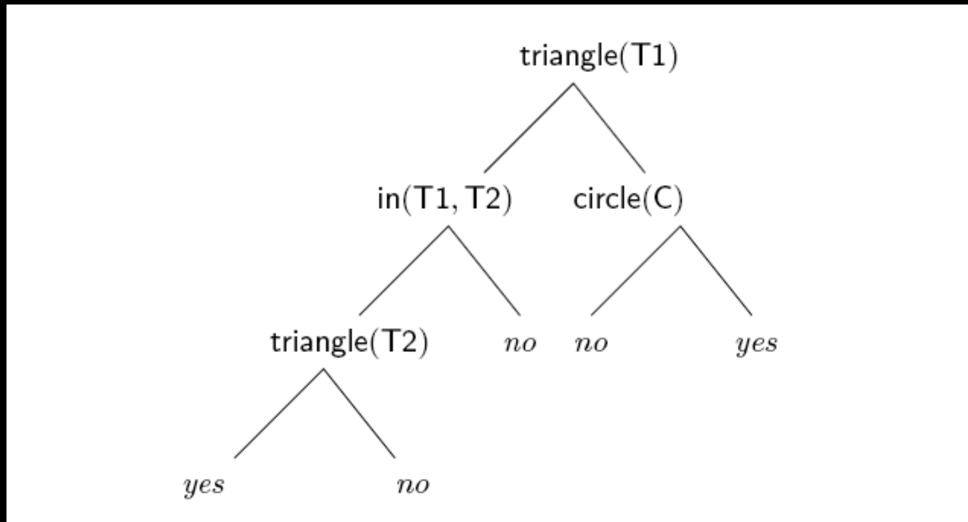
- determine the representation of examples and hypotheses
- select the right type of coverage and generality (subsumption)
- keep existing algorithm (CN2) but replace operators
- keep search strategy
- fast implementation.

# 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

# 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

# Typical Machine Learning Problem

**Given**

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

**Find**

- A hypothesis **h** in **Lh** that minimizes the loss function w.r.t. the examples **E** taking **B** into account

# Three possible SETTINGS

Learning from entailment  (FOIL)

- 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

The setting can matter a lot

A Knowledge Representation Issue

# Learning from interpretations

## Examples as "relational state descriptions"

- {triangle(t1), circle(c1), inside(c1,t1)}
- {triangle(t3), triangle(t4), inside(t3,t4), circle(c5)}

## Hypotheses consist of properties / constraints

- triangle(T) :- circle(C), inside(T,C)
- IF there is a circle C inside an object T THEN T is a triangle
- false :- circle(C1), circle(C2), inside(C1,C2)
- NO circle is inside another circle ...

# Learning from interpretations

## Examples

- Positive: { human(luc), human(lieve), male(luc), female(lieve)}

## Hypothesis (positives only)

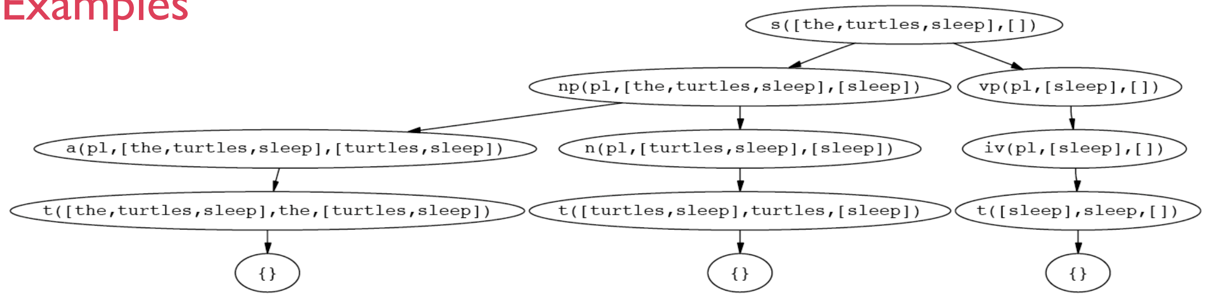
*(maximally specific that covers example)*

- human(X) :- female(X)
- human(X) :- male(X)
- false :- male(X), female(X)
- male(X); female(X) :- human(X)

OFTEN used for finding INTEGRITY CONSTRAINTS / FREQ. PATTERN MINING

# Learning from Proofs

Examples

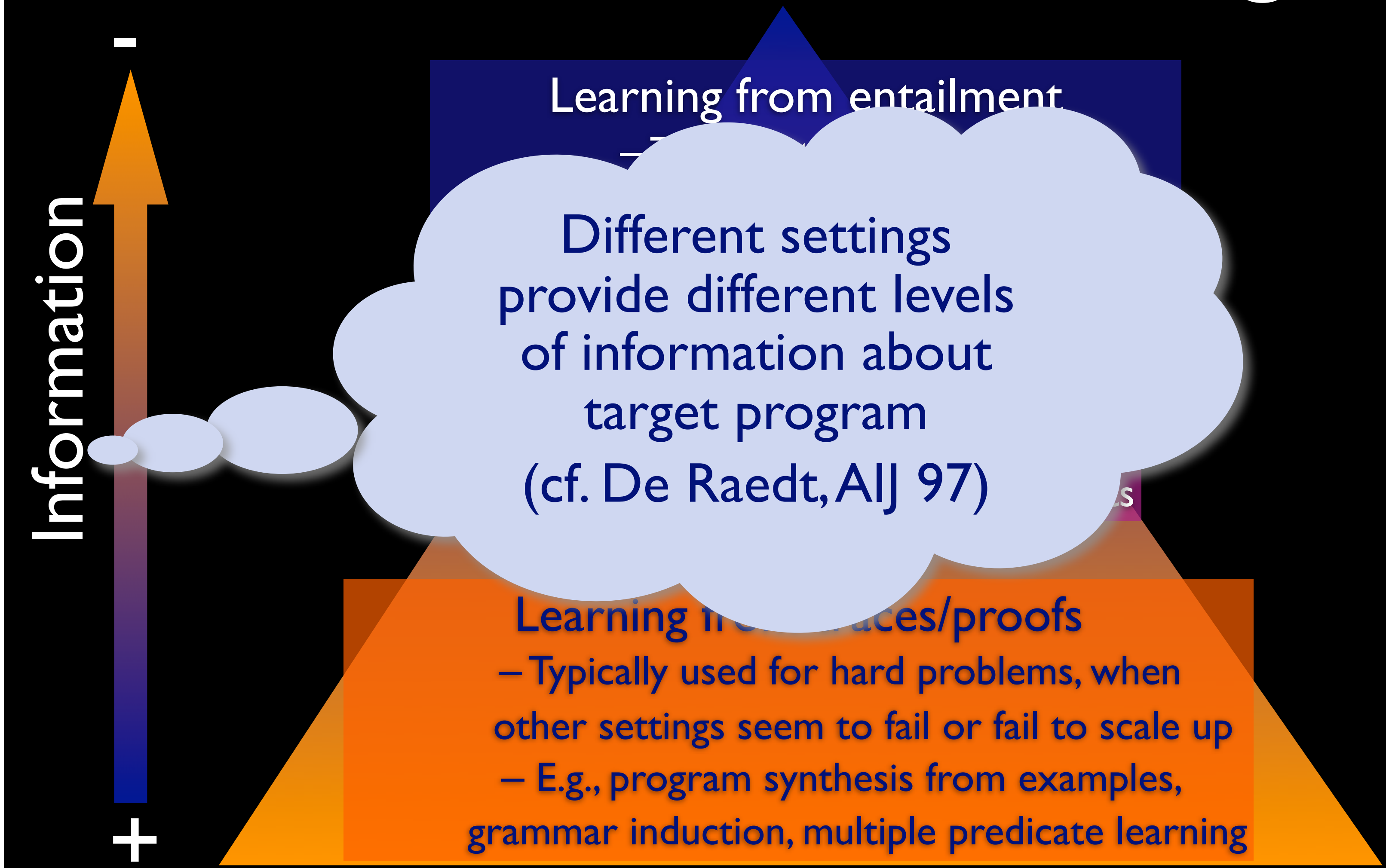



```
sentence(A, B) :- noun_phrase(C, A, D), verb_phrase(C, D, B).
noun_phrase(A, B, C) :- article(A, B, D), noun(A, D, C).
verb_phrase(A, B, C) :- intransitive_verb(A, B, C).
article(singular, A, B) :- terminal(A, a, B).
article(singular, A, B) :- terminal(A, the, B).
article(plural, A, B)   :- terminal(A, 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).
```

Hypothesis

Used in Treebank Grammar Learning & Program Synthesis

# Use of different Settings



# LOGIC, RELATIONS and PROBABILITY

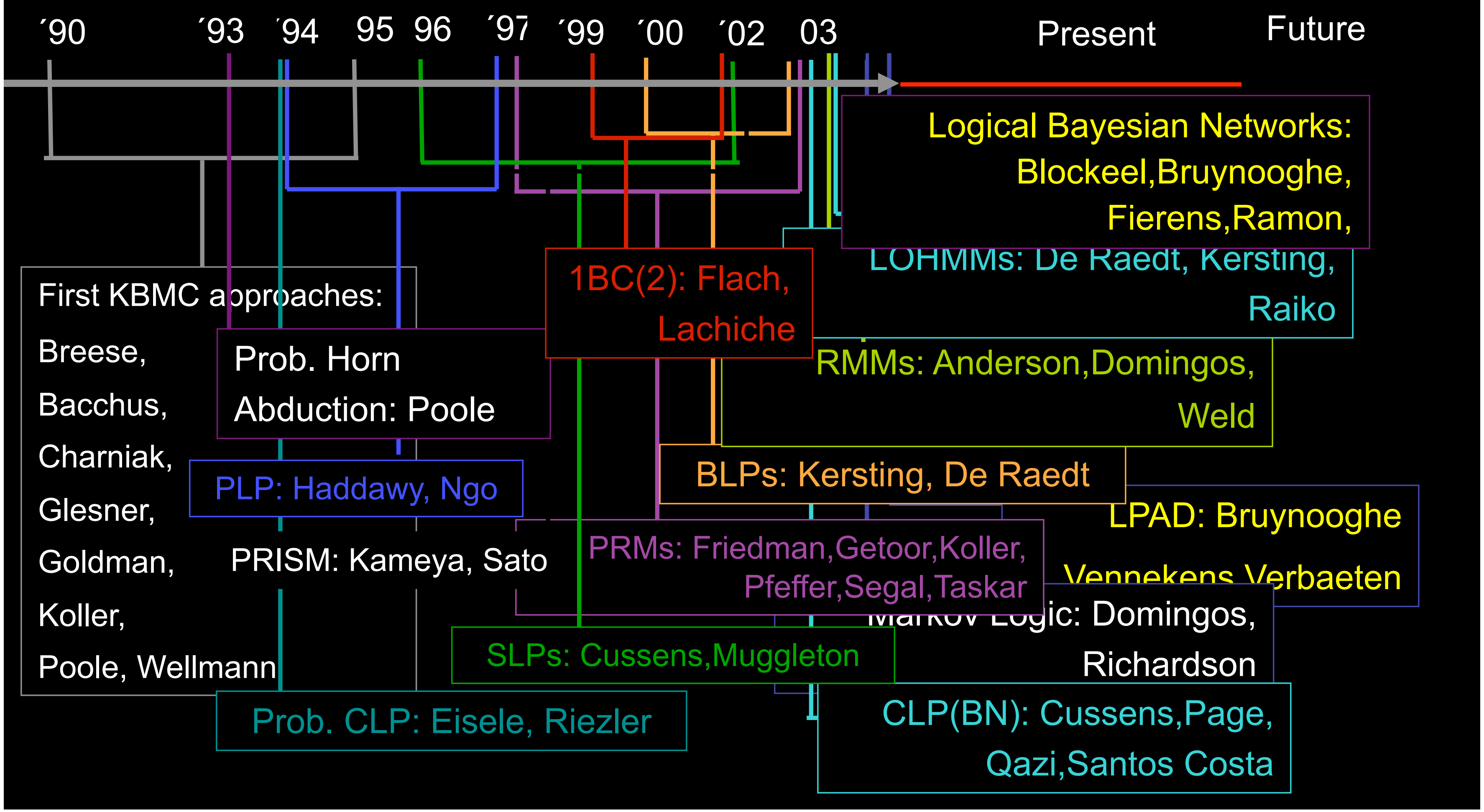
Joint work with **Kristian Kersting et al.**

# Statistical Relational Learning

Logic and relations alone are often insufficient

- but can be combined with probabilistic reasoning and models
- use logic as a toolbox

# Some SRL formalisms



# PLL: What Changes ?

Clauses annotated with probability labels

- E.g. in Sato's Prism, Eisele and Muggleton's SLPs, Kersting and De Raedt's BLPs, …

Prob. covers relation covers(e,H U B) = P(e | H,B)

- Probability distribution P over the different values e can take; so far only (true,false)

Knowledge representation issue

- Define probability distribution on examples / individuals
- What are these examples / individuals ? [cf. SETTINGS]

# Two key approaches

- Logical Probability Models [MLNs, PRMs, BLPs, ...]
    - Knowledge Based Model Construction, use (clausal) logic as a template
    - generate graphical model on which to perform probabilistic inference and learning
- Probabilistic Logical Models [ICL, PRISM, ProbLog, SLPs, ...]
    - Annotate logic with probabilities
    - perform inference and learning in logic
    - illustrate the idea of upgrading

# Probabilistic *generative* SRL 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 Issues

- Defining Lh and **P**
  - Clauses + Probability Labels
- Learning Methods
  - Parameter Estimation
    - Learning probability labels for fixed clauses
  - Structure learning
    - Learning both components

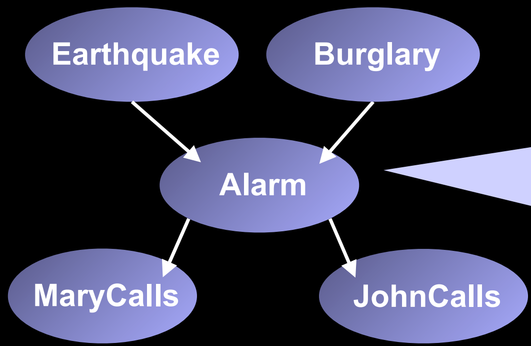
# PLL: Three Settings

- Probabilistic learning from interpretations
  - Bayesian logic programs, Koller's PRMs, Domingos' MLNs, Vennekens' LPADs
- Probabilistic learning from entailment
  - Eisele and 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.

# Learning from interpretations

- Possible Worlds -- Knowledge Based Model Construction
- Bayesian logic programs (Kersting & De Raedt)
- Markov Logic (Richardson & Domingos)
- Probabilistic Relational Models (Getoor, Koller, et al.)
- Relational Bayesian Nets (Jaeger), ...

# Bayesian Networks




| E | B | P(A \| B,E) | |
|---|---|---|---|
| $e$ | $b$ | 0.9 | 0.1 |
| $e$ | $\bar{b}$ | 0.2 | 0.8 |
| $\bar{e}$ | $b$ | 0.9 | 0.1 |
| $\bar{e}$ | $\bar{b}$ | 0.01 | 0.99 |

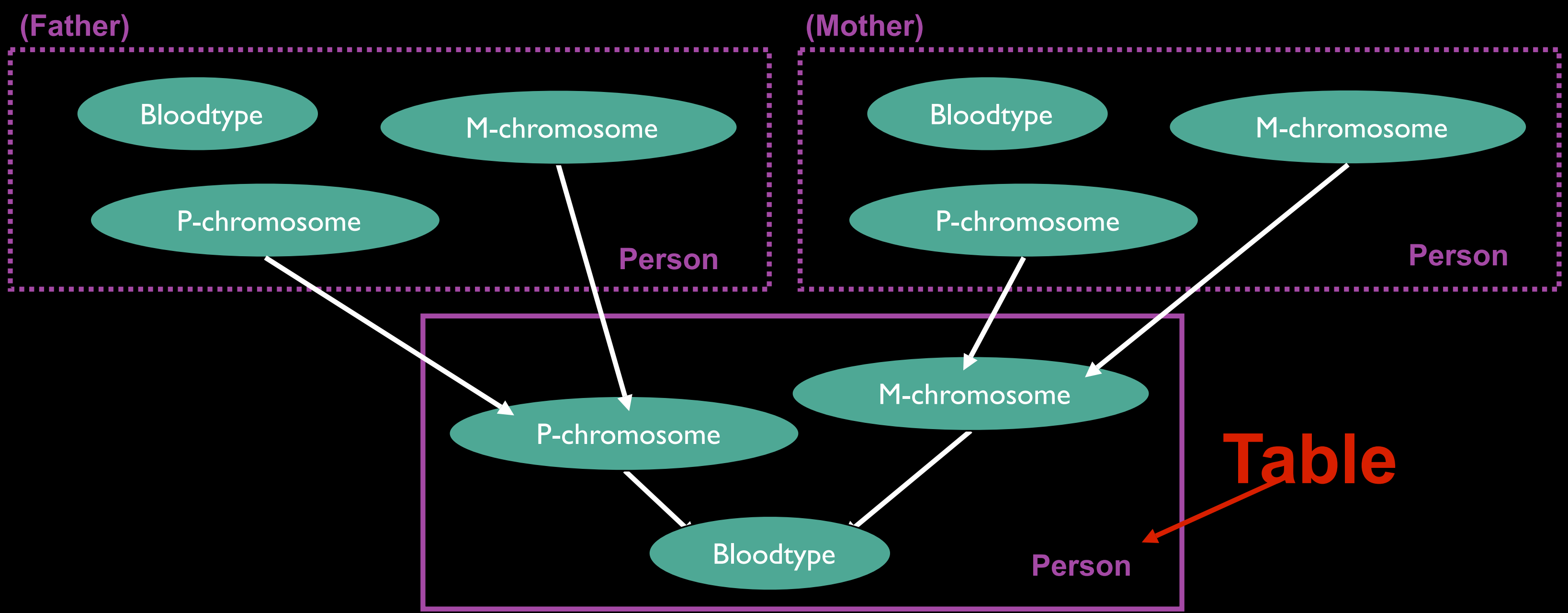
$$P(E,B,A,J,M) = P(E).P(B).P(A|E).\,P(A|B).P(J|A).P(M|A)$$

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

INTERPRETATION
STATE/DESCRIPTION
$\{A, \neg E, \neg B, J, M\}$

# Probabilistic Relational Models (PRMs)

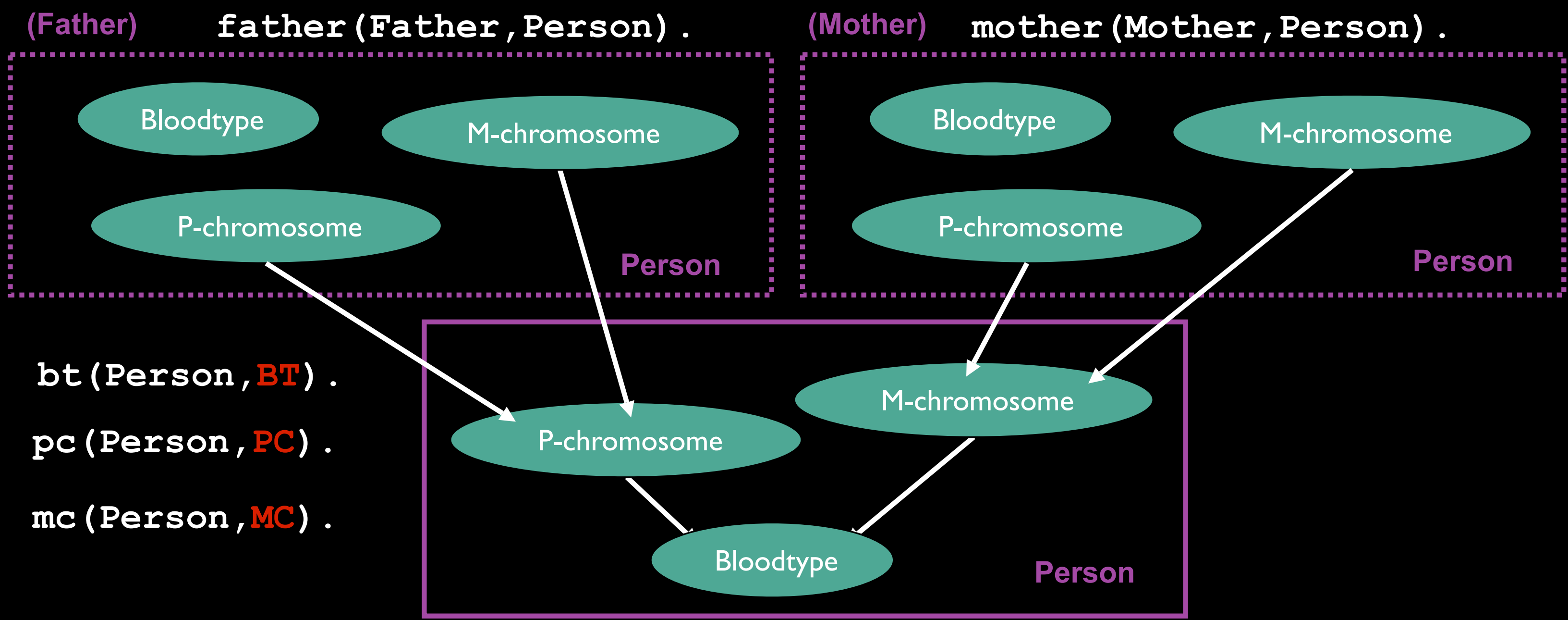
[Getoor,Koller, Pfeffer]




[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).
```

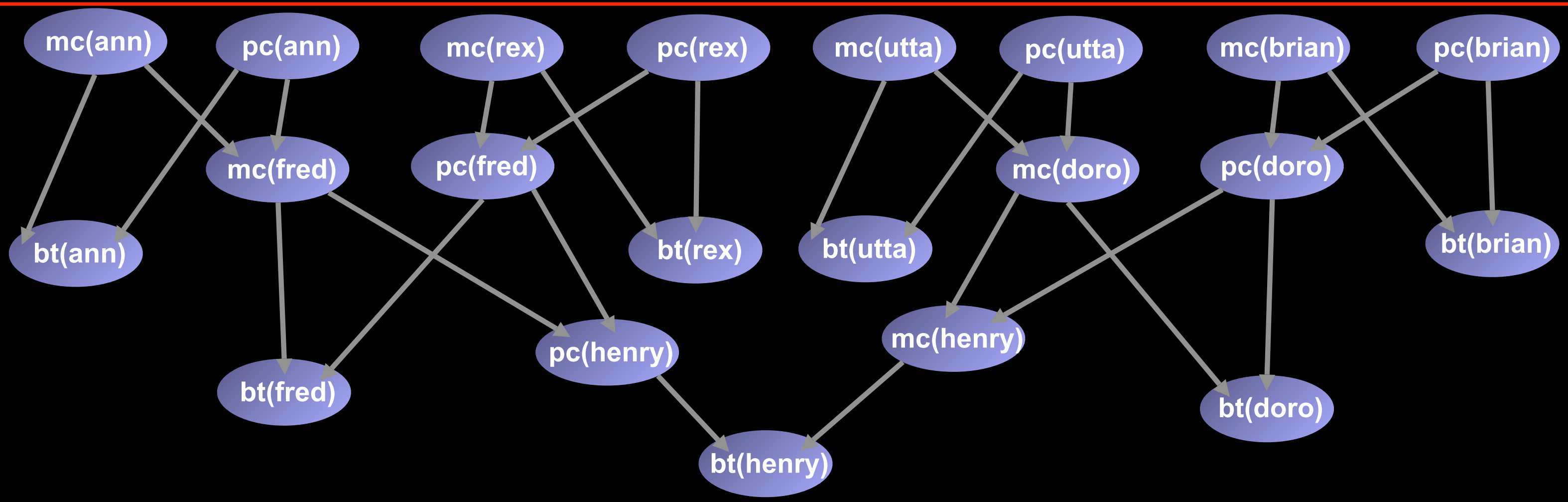
Extension

```
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

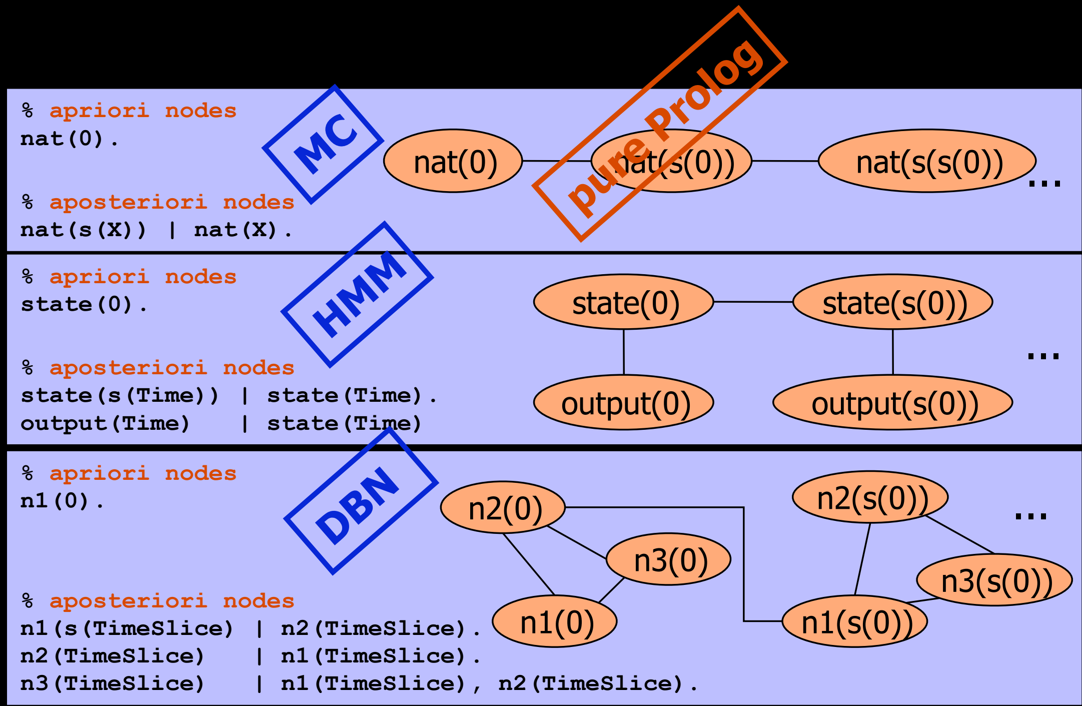
Advantages

- same intension used for multiple extensions
- parameters are being shared / tied together
- unification is essential
- learning becomes feasible

Typical use includes

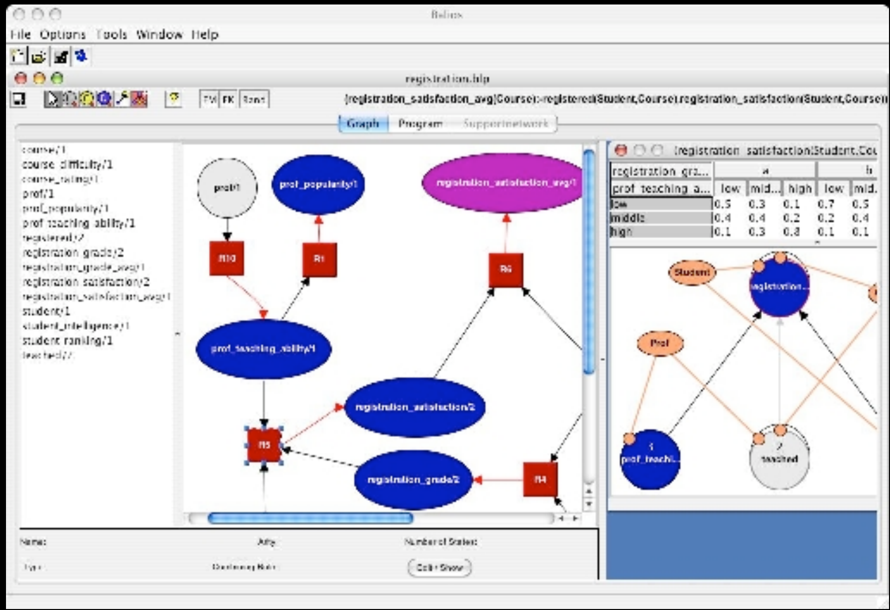
- prob. inference P(Q | E), P(bt(mary) | bt(john) =o-)
- max. likelihood parameter estimation & structure learning

# Bayesian Logic Programs




Prolog and Bayesian Nets as Special Case

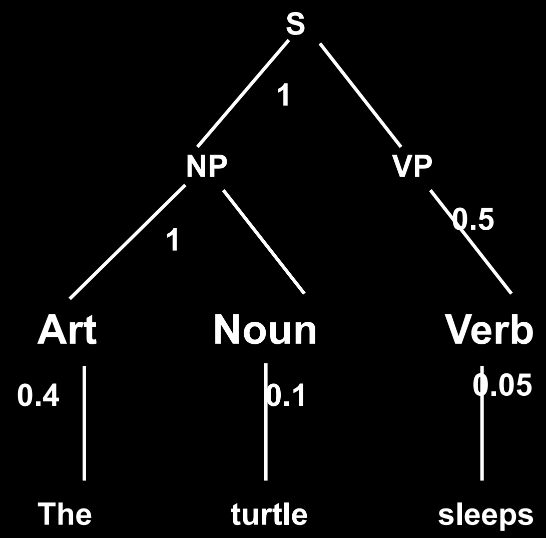
# Balios Tool



File Options Tools Window Help
Graph
Program
prof_popularity/1
registration_satisfaction_avg/1
prof_teaching_ability/1
registration_satisfaction/2
registration_grade/2
Student
Prof

# Learning from Proofs
# 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

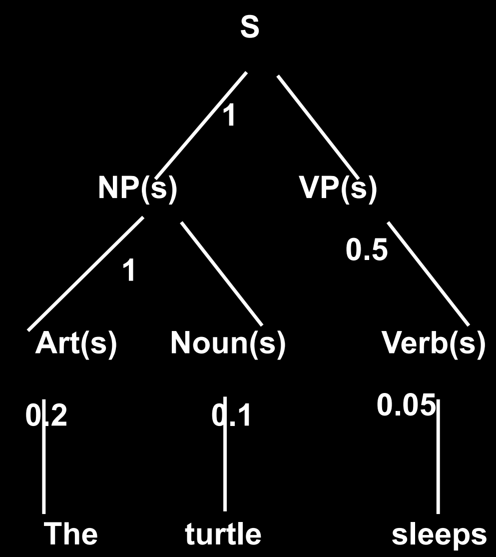
$P(parse\ tree) = \prod_i p_i^{c_i}$
where $p_i$ is the probability of rule $i$
and $c_i$ the number of times
it is used in the parse tree

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

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

# 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

# In SLP notation

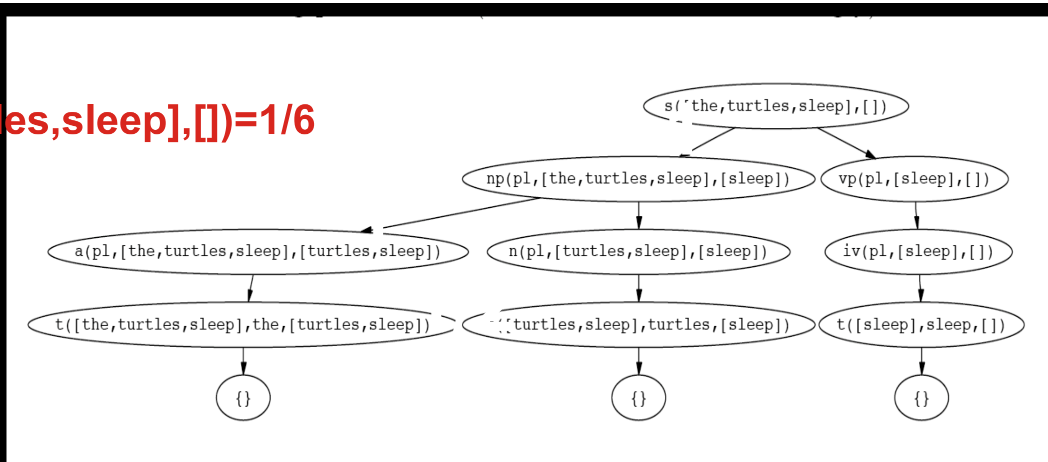
1

1/3

1/2

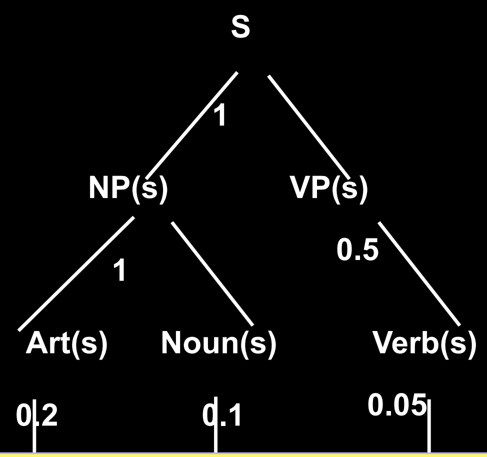
```
sentence(A, B) :- noun_phrase(C, A, D), verb_phrase(C, D, B).
noun_phrase(A, B, C) :- article(A, B, D), noun(A, D, C).
verb_phrase(A, B, C) :- intransitive_verb(A, B, C).
article(singular, A, B) :- terminal(A, a, B).
article(singular, A, B) :- terminal(A, the, B).
article(plural, A, B)   :- terminal(A, 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

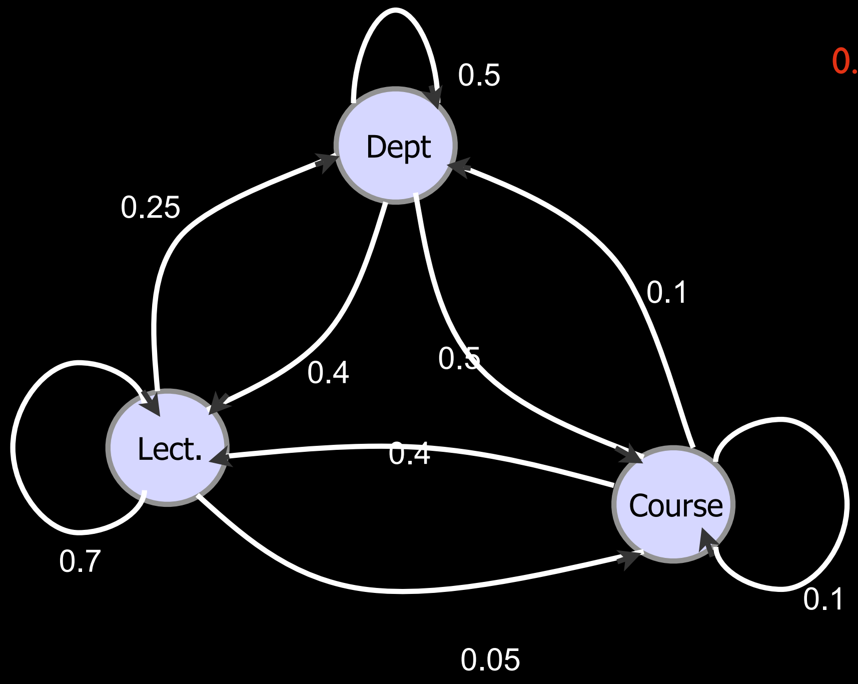
  department(cs,nebel) :-
      prof(nebel), in(cs), co(ai), lecturer(nebel,ai).

  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).

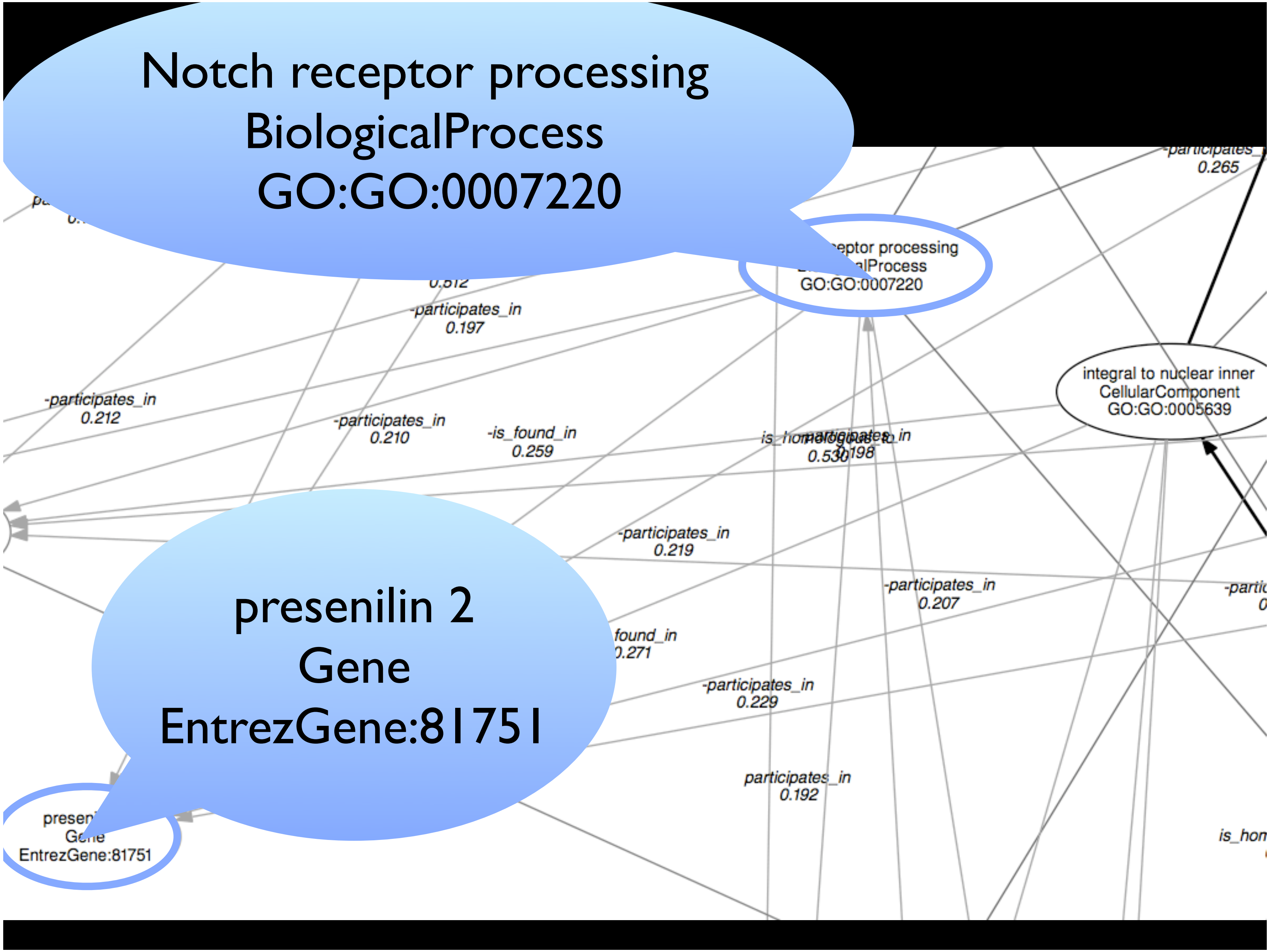
0.1 prof(nebel).
0.05 prof(burgard).
...

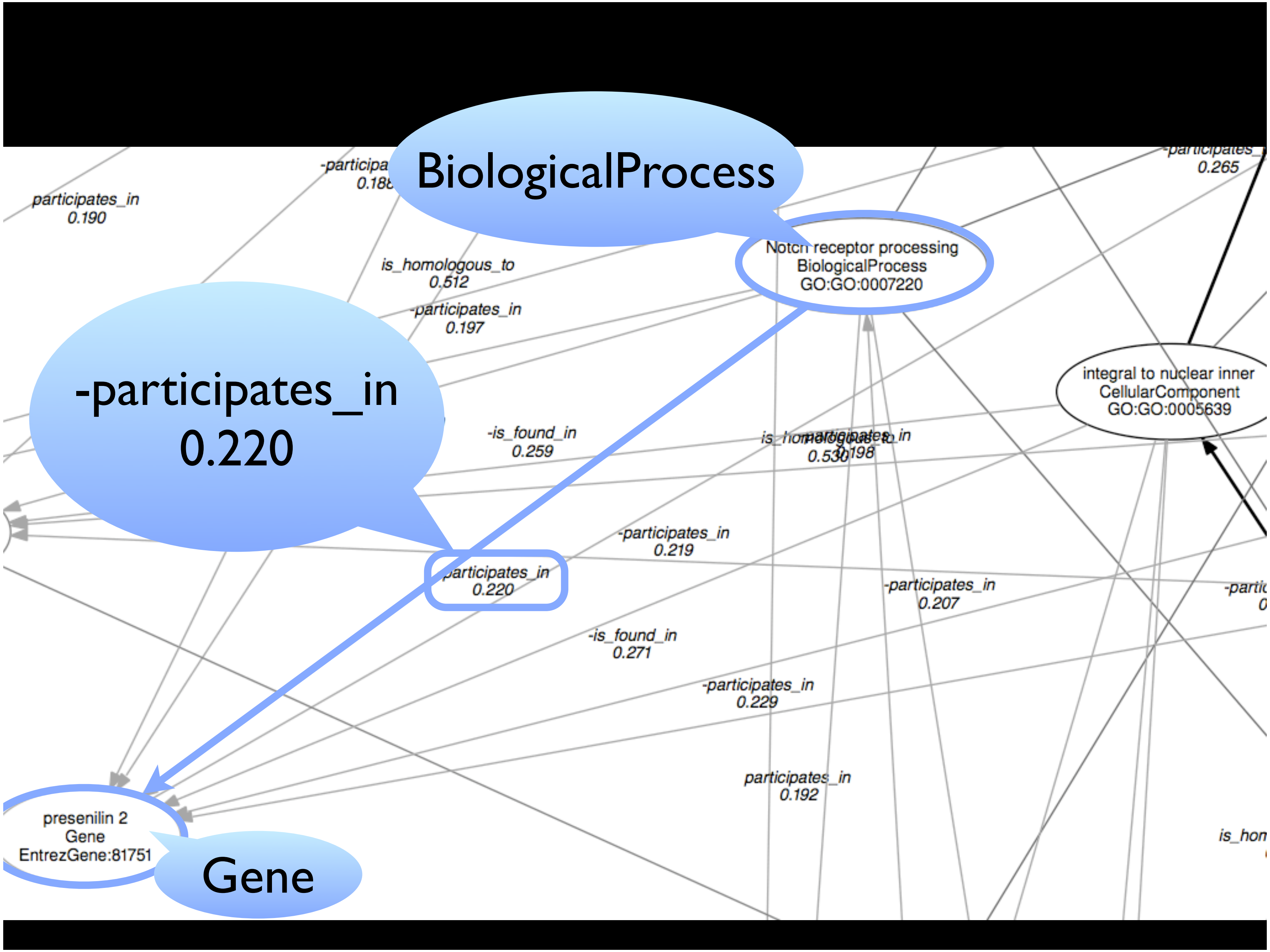
An interesting application exist using RMMs
[Anderson and Domingos, KDD 03]

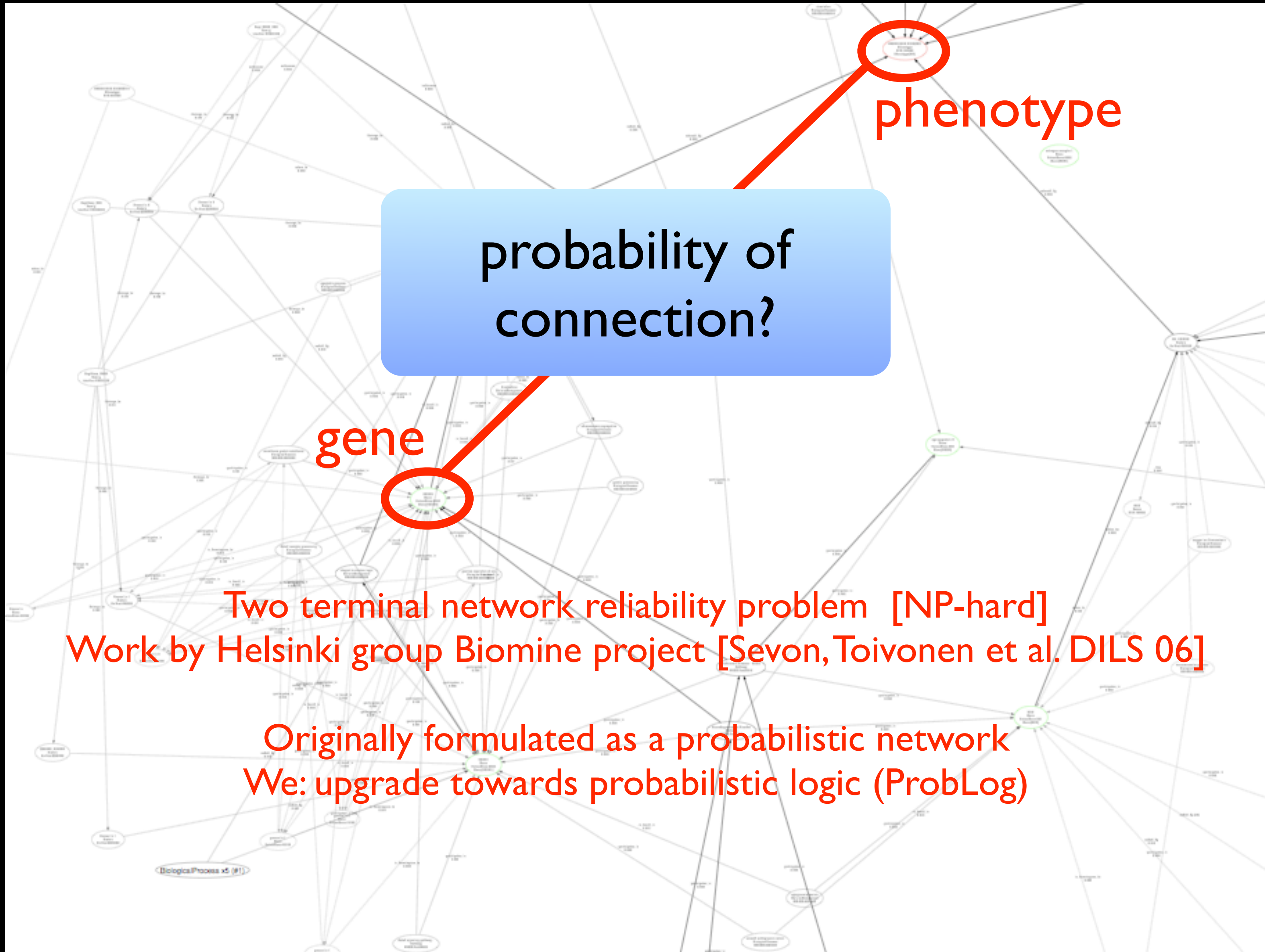
# Probabilities on Proofs

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.


Notch receptor processing
BiologicalProcess
GO:GO:0007220
presenilin 2
Gene
EntrezGene:81751
integral to nuclear inner
CellularComponent
GO:GO:0005639
-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
-participates_in
0.229
participates_in
0.192
0.265
EntrezGene:81751


BiologicalProcess
-participates_in
0.220
Gene
Notch receptor processing
BiologicalProcess
GO:GO:0007220
presenilin 2
Gene
EntrezGene:81751
integral to nuclear inner
CellularComponent
GO:GO:0005639
participates_in
0.190
is_homologous_to
0.512
-participates_in
0.197
-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
0.265




Two terminal network reliability problem [NP-hard]
Work by Helsinki group Biomine project [Sevon, Toivonen et al. DILS 06]

Originally formulated as a probabilistic network
We: upgrade towards probabilistic logic (ProbLog)

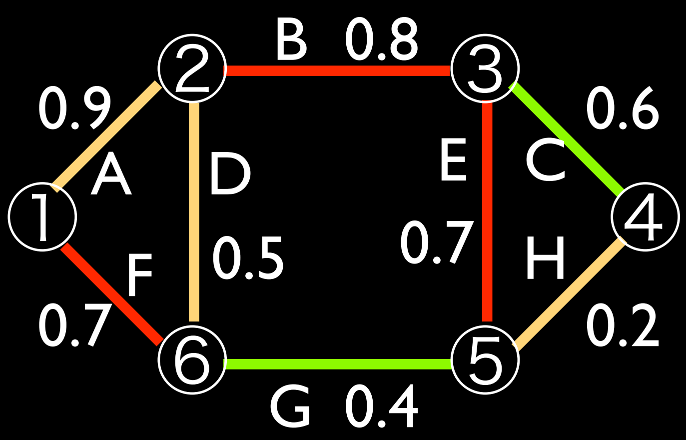
# Distribution Semantics

- Due to Taisuke Sato
  - provides a natural basis for many probabilistic logics
  - PRISM (Sato & Kameya), PHA & ICL (Poole), ProbLog (De Raedt et al.), CP-logic (Vennekens, ...)
  - Will represent a simplified and unifying view as in ProbLog [De Raedt et al.]

# Distribution Semantics

- probabilistic predicates F
  - define using p : $q(t_1,...,t_n)$
  - denotes that **ground** atoms $q(t_1,...,t_n)\theta$ are true with probability p
  - assume all ground probabilistic atoms to be **marginally independent**
- logical ones DB
  - define as usual using logic program -- WE : PATH predicate
- a similar semantics has been reinvented many times ----

# Example in ProbLog




facts mutually independent

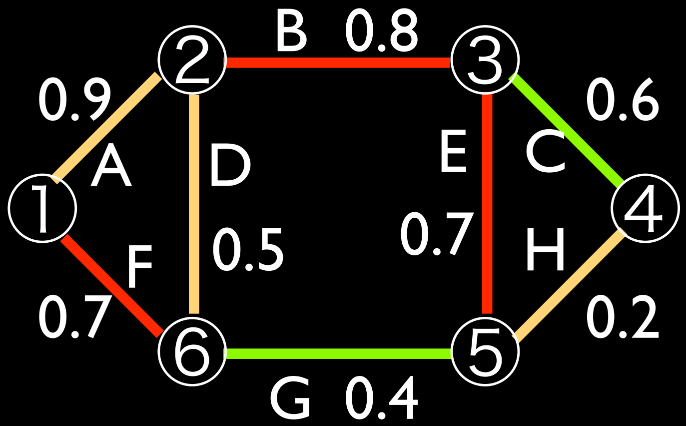
logical part L

ProbLog theory T

```
0.9 : y_edge(1,2).
0.8 : r_edge(2,3).
0.6 : g_edge(3,4).
...
```

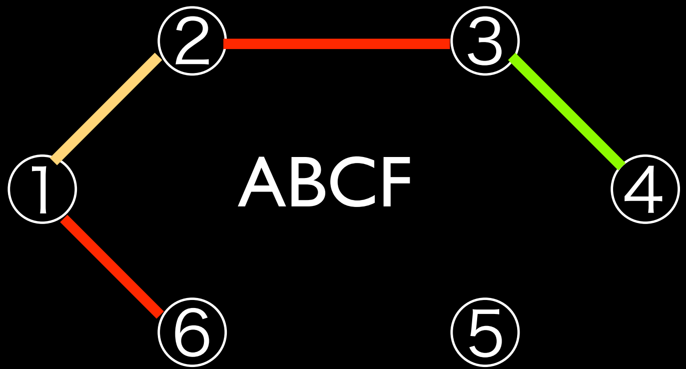
[De Raedt, Kimmig, Toivonen, IJCAI 07]

# Sampling Subprograms



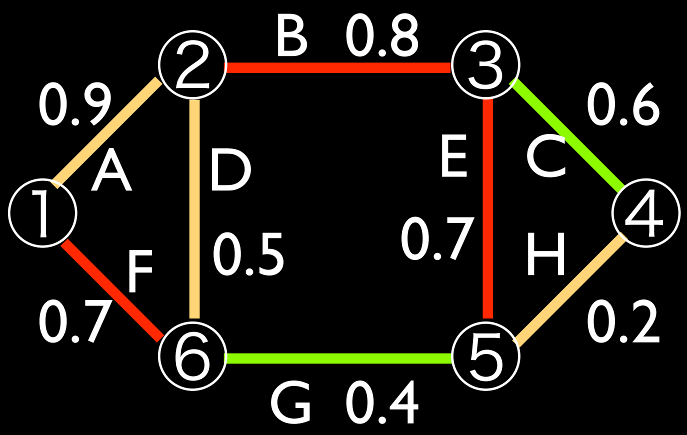

- Biased coins
- Independent

| A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|
| + | + | + | − | − | + | − | − |

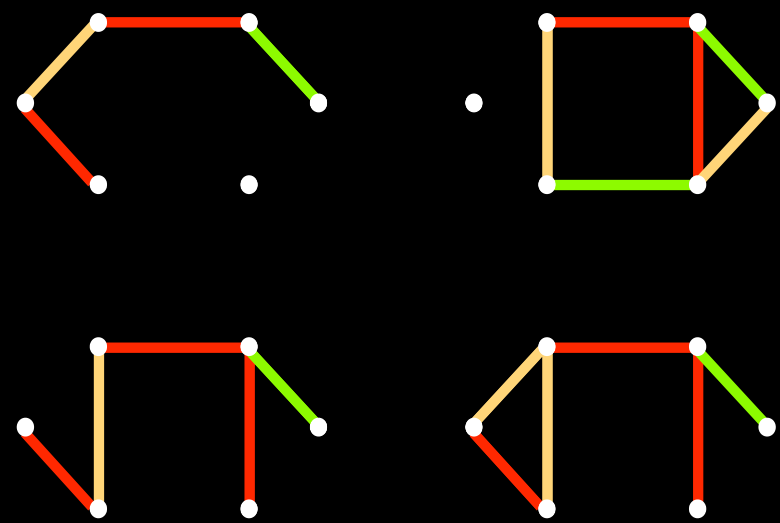



$$P = 0.9 \cdot 0.8 \cdot 0.6 \cdot (1 - 0.5) \cdot (1 - 0.7) \cdot 0.7 \cdot (1 - 0.4) \cdot (1 - 0.2)$$

# Queries




1 → 4

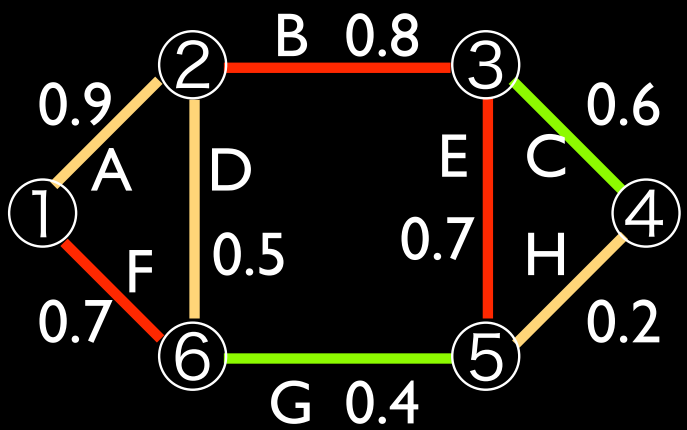


path(x,y) :- edge(x,y)
path(x,y) :- edge(x,z), path(y,z)

$$P(q|T) = \sum_{S \subseteq L, S \models q} P(S|T)$$

...

# Queries




①→④

path(x,y) :- edge(x,y)
path(x,y) :- edge(x,z), path(y,z)

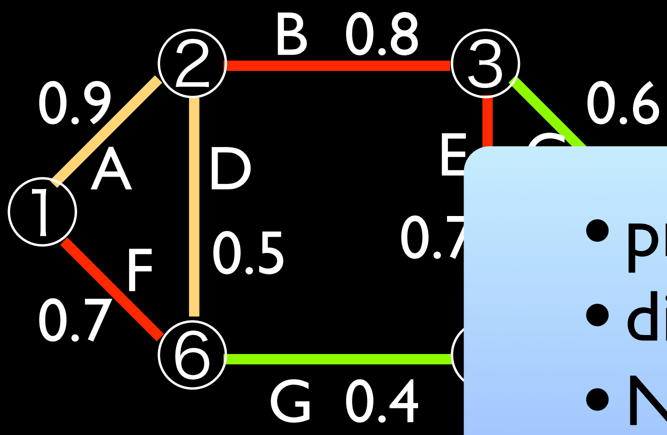
Key Point
of ProbLog and Logic

any relation can be defined

$$P(q|T) = \sum_{S \subseteq L, S \models q} P(S|T)$$

# Query Probability

## using proofs



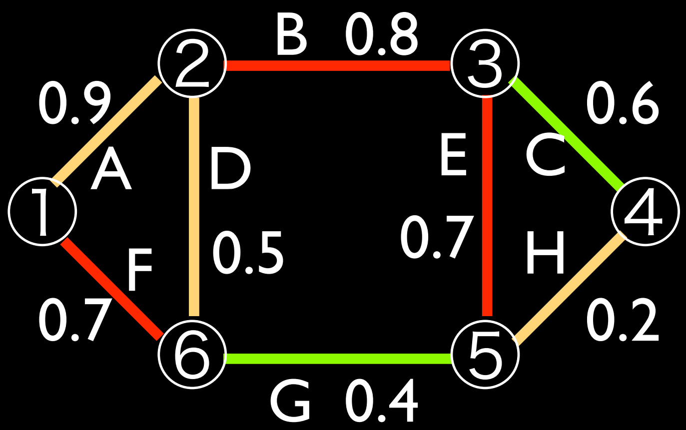

①→④

- proofs overlap
- disjoint sum
- NP-hard
- approximation algorithm [De Raedt et al, IJCAI 07]

$$P(\mathtt{path}(1,4)|T)$$
$$= P(ABC + ABEH + \ldots + FDBEH)$$

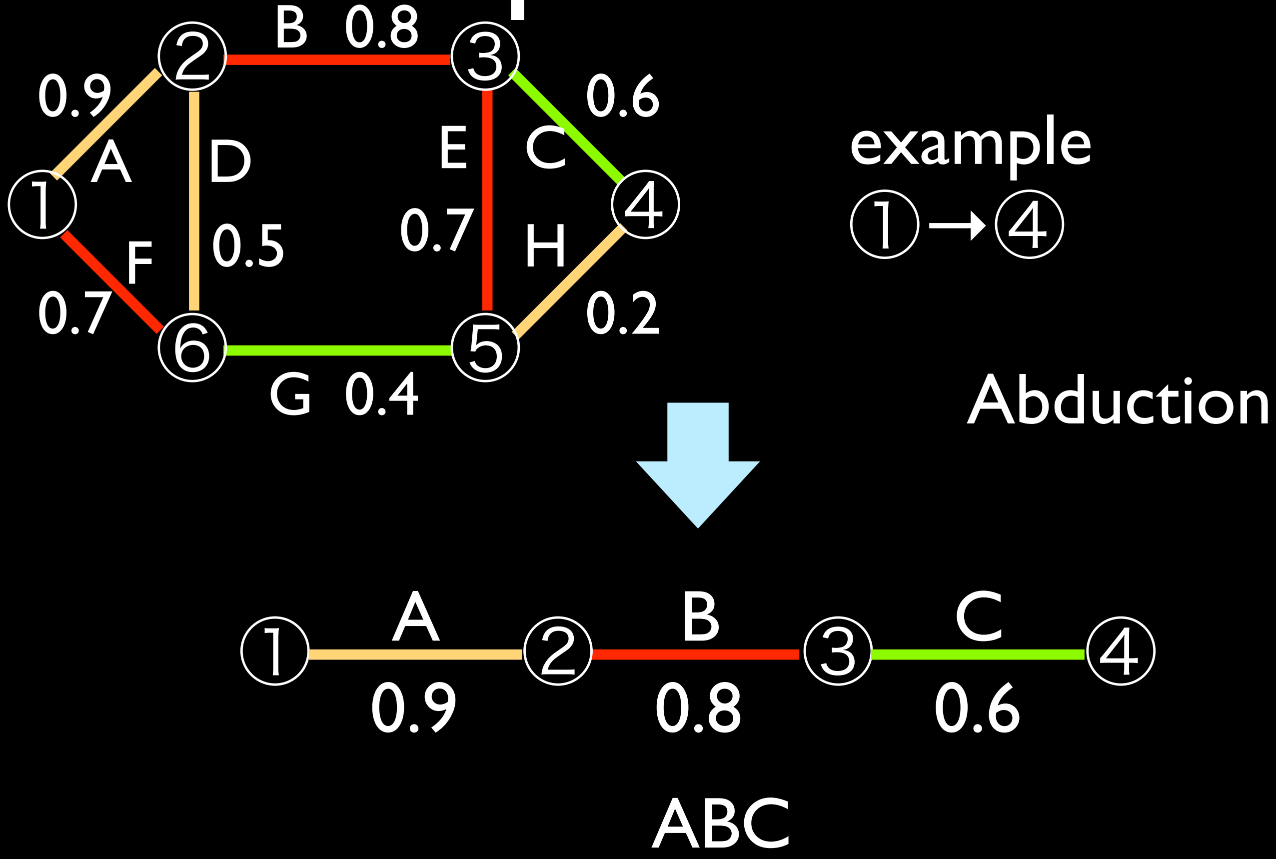
# Query Probability

## using Proofs




Prism (Sato) and ICL (Poole) avoid the disjoint problem by requiring that explanations do not overlap

# Most likely proof / explanation



# Semantics ProbLog

Not really new, rediscovered many times

Intuitively, a probabilistic database

Formally, a distribution semantics [Sato 95]

Other systems, such as Sato's Prism and Poole's ICL avoid the disjoint sum problem

- assume that explanations / proofs are mutually exclusive, that is,
- $P(A \vee B \vee C) = P(A) + P(B) + P(C)$

Long term vision: develop an optimized probabilistic Prolog implementation in which other SRL formalisms can be compiled. (work together with Vitor Santos Costa and Bart Demoen, integration in YAP Prolog planned)

# An ILLUSTRATION in LINK MINING

# Some learning tasks

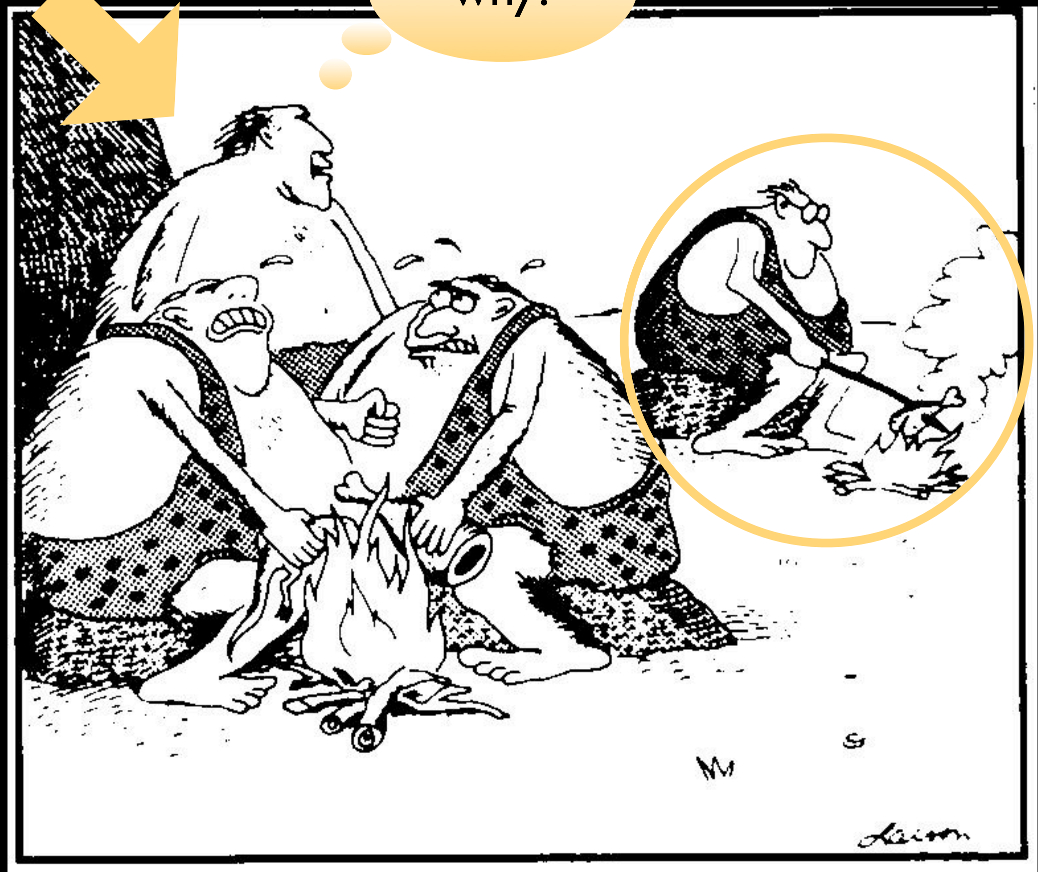
Following the upgrading idea

1. explanation based learning
2. local pattern mining
3. theory compression
4. parameter learning

# 1. Explanation Based Learning as presented by Gerald DeJong

why?

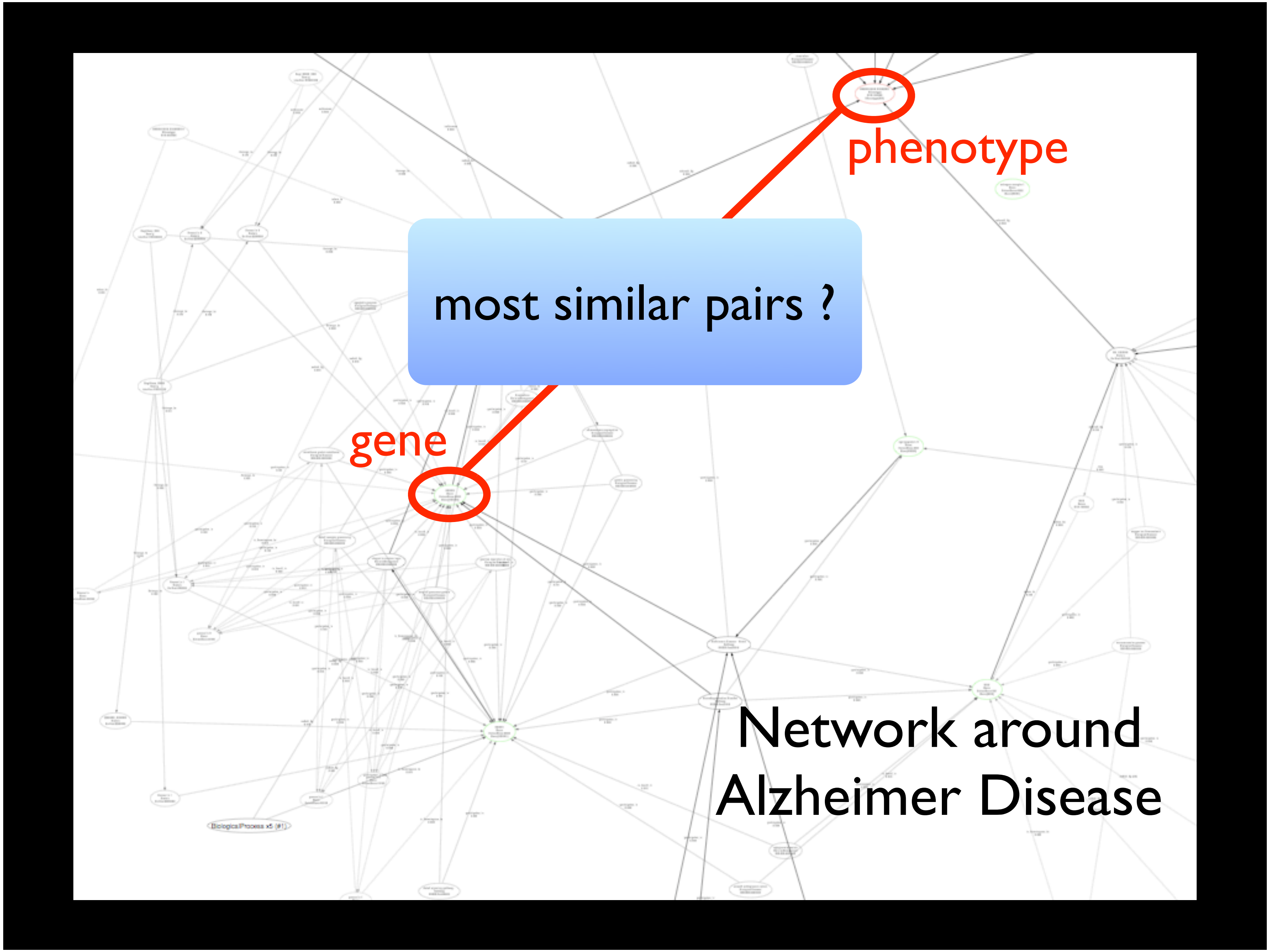
Theory: knowledge about world (fire, meat, ...)



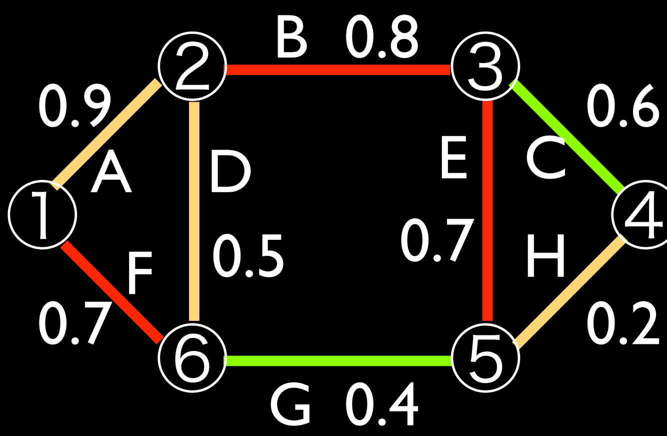
"Hey! Look what Zog do!"

Example: no burned hands

Explanation: use stick

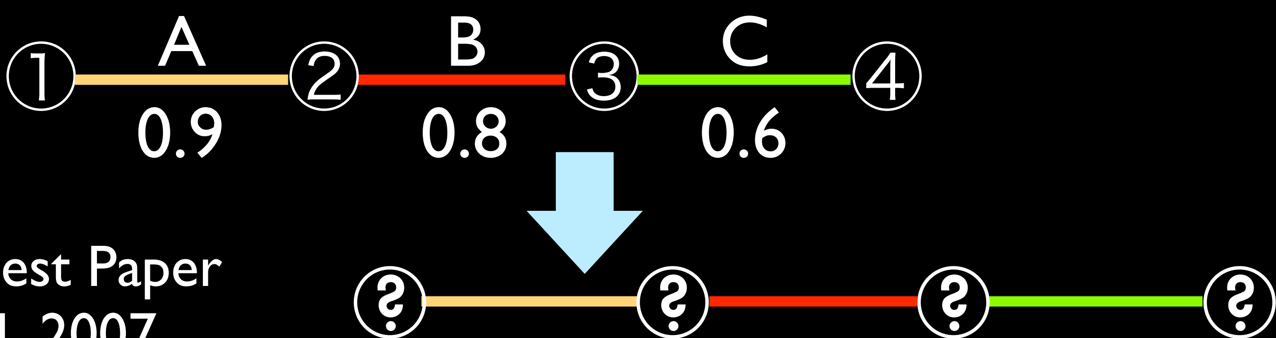

phenotype
most similar pairs ?
gene
Network around
Alzheimer Disease

# Most Likely Generalized Explanation



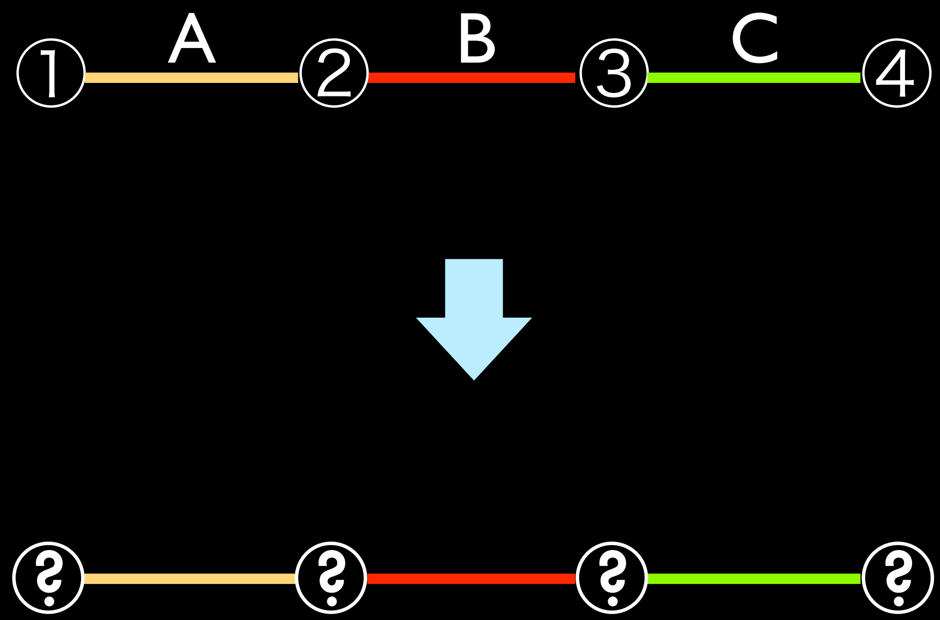

example
① → ④

path(x,y) :- edge(x,y)
path(x,y) :- edge(x,z), path(y,z)

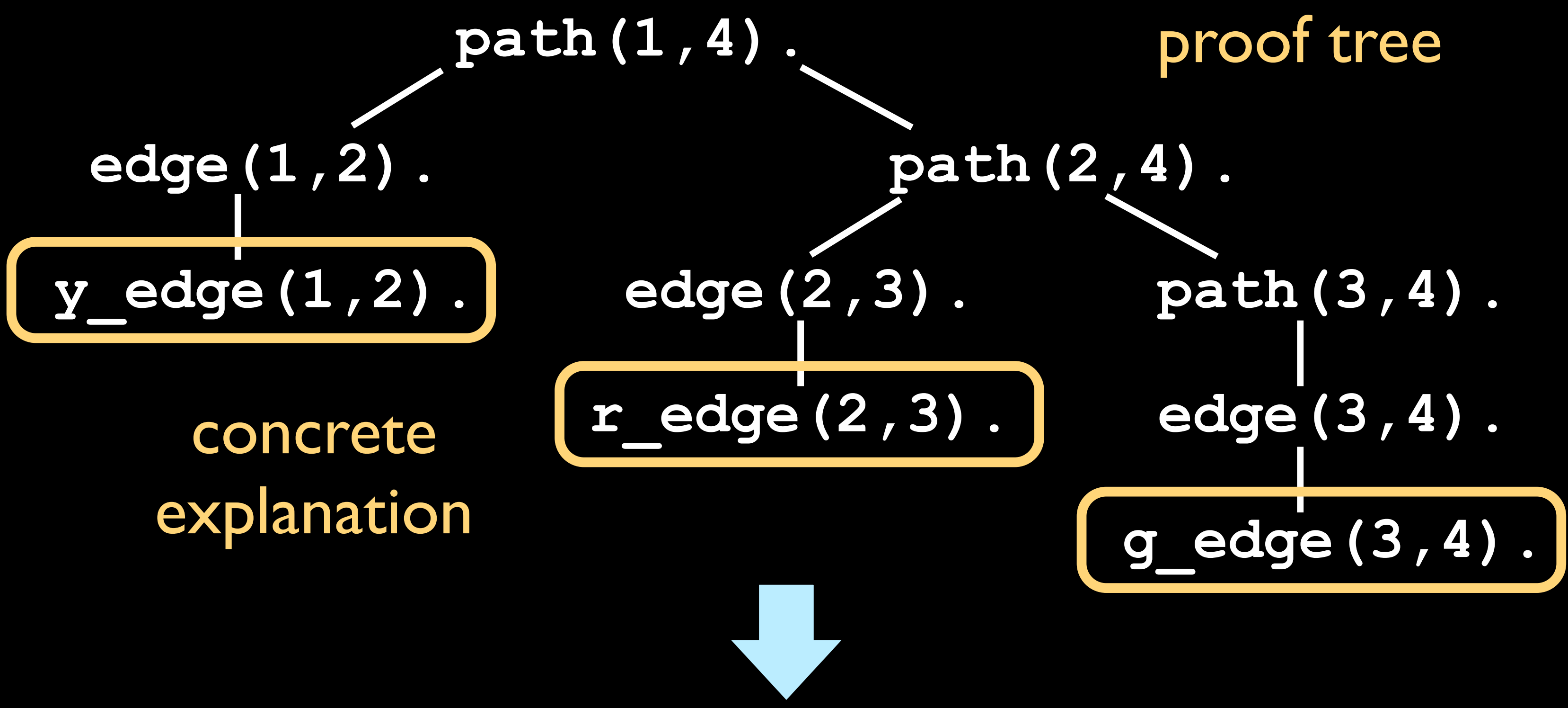



Kimmig et. al. Best Paper
Award ECML 2007

# Generalize Explanation

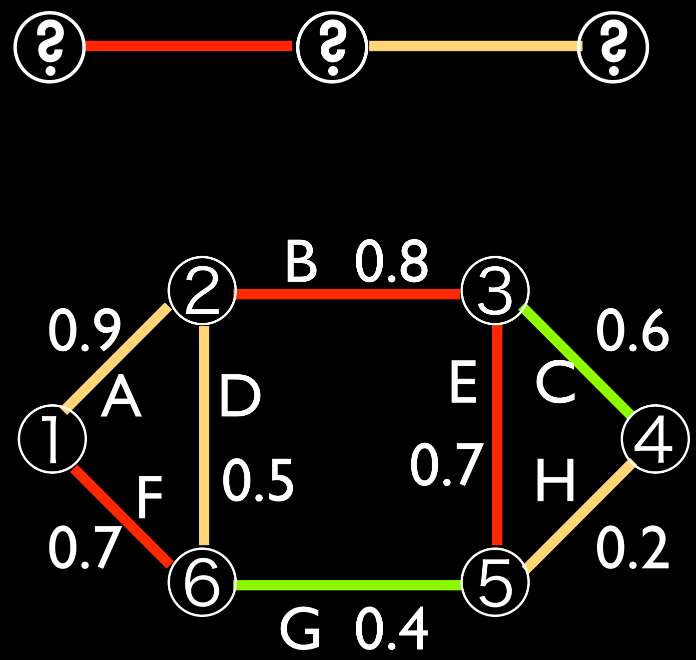


# Prolog Setting

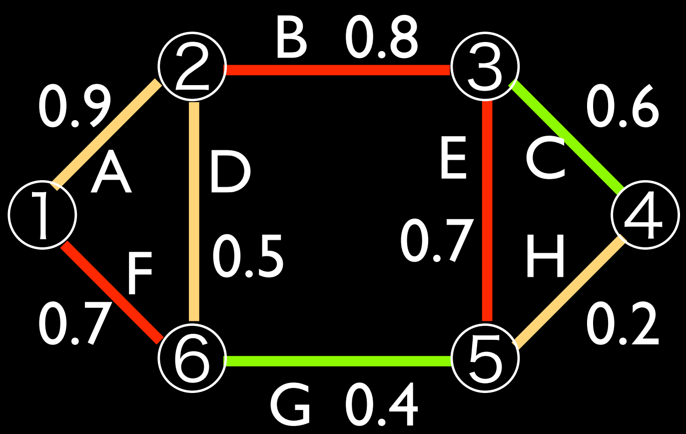



```
path(P,S) ←
      y_edge(P,Q),r_edge(Q,R),g_edge(R,S).
```

# Use of Generalized Explanation



# Use of Generalized Explanation




③→① 0.72

⑥→② 0.63

③→⑥ 0.40

①→② 0.35

③→④ 0.14

reasoning by similarity / analogy

# Experiments

| | depth | nodes | edges | ag | ng | pt | pos | neg |
|---|---|---|---|---|---|---|---|---|
| Alz1 | 4 | 122 | 259 | 14 | 15 | 3 | 182 | 2254 |
| Alz2 | 5 | 658 | 3544 | 17 | 20 | 4 | 272 | 5056 |
| Alz3 | 4 | 351 | 774 | 72 | 33 | 3 | 5112 | 27648 |
| Alz4 | 5 | 3364 | 17666 | 130 | 55 | 6 | 16770 | 187470 |
| Ast1 | 4 | 127 | 241 | 7 | 12 | 2 | 42 | 642 |
| Ast2 | 5 | 381 | 787 | 11 | 12 | 2 | 110 | 902 |

**Table 1.** Graph characteristics: search depth used during graph extraction, numbers of nodes and edges, number of genes annotated resp. not annotated with the corresponding disease and number of phenotypes, number of positive and negative examples for connecting two genes and a phenotype.

# Experiments

| | Alz1 | | | | | | Ast1 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | pos(1) | pos(3) | pos(5) | pos_n | pos_a | prec | pos(1) | pos(3) | pos(5) | pos_n | pos_a | prec |
| Alz1 | 0.95 | 2.53 | 3.95 | 6.91 | 16.82 | 0.46 | 1.00 | 3.00 | 4.86 | 6.86 | 10.57 | 0.23 |
| Alz2 | 0.84 | 2.24 | 3.60 | 7.37 | 18.65 | 0.42 | 0.86 | 2.86 | 4.71 | 6.86 | 14.56 | 0.22 |
| Alz3 | 0.99 | 2.64 | 4.09 | 23.20 | 126.09 | 0.48 | 1.00 | 2.71 | 4.14 | 6.86 | 28.00 | 0.24 |
| Alz4 | 0.84 | 2.23 | 3.58 | 7.37 | 18.80 | 0.42 | 0.86 | 2.29 | 3.43 | 5.14 | 28.00 | 0.15 |
| Ast1 | 0.09 | 0.26 | 0.44 | 2.07 | 2.07 | 0.02 | 1.00 | 3.00 | 4.86 | 17.14 | 17.14 | 0.34 |
| Ast2 | 0.08 | 0.23 | 0.38 | 2.00 | 2.00 | 0.01 | 0.86 | 2.57 | 4.29 | 16.57 | 16.57 | 0.20 |

**Table 2.** Averaged results over all examples learned on Alz1 resp. Ast1 and evaluated on 6 different graphs: number of positives among the first $k$ answers for $k = 1, 3, 5$, number of positves returned before the first negative, absolute number of positives returned, and precision.

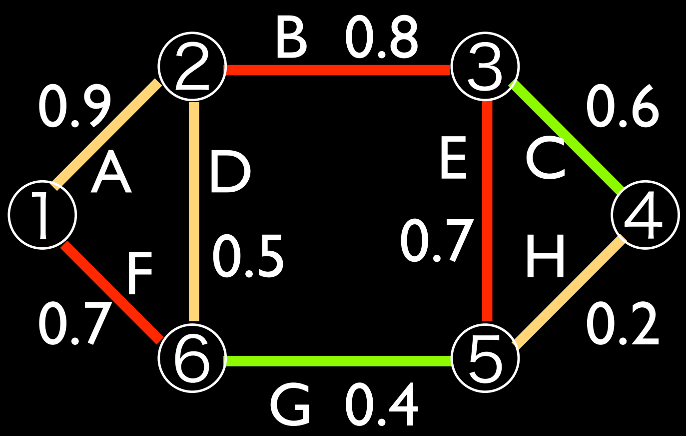
# PEBL Contributions

- EBL in probabilistic context
- Multiple explanations: most likely one
- Reasoning by analogy:
  background knowledge + likelihood

# 2. Probabilistic Pattern Mining

What are the most likely explanations the examples have in common ?

criterion: average probability is higher than threshold




③ ①

⑥ ②

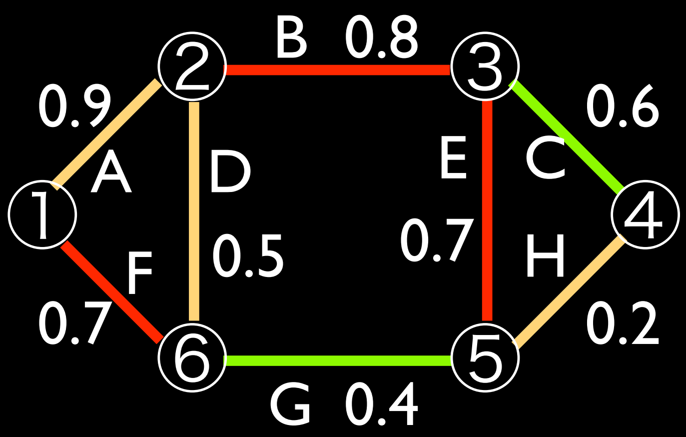
③ ⑥

① ②

③ ④

no definition of path

# Probabilistic Pattern Mining


③ ①

⑥ ②

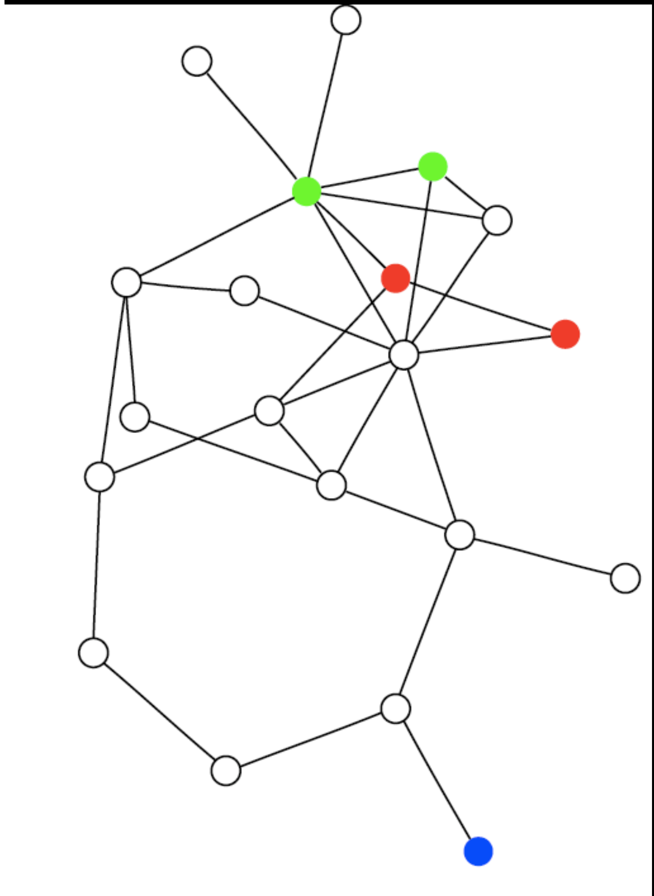
③ ⑥

① ②

③ ④

no definition of path

# 3. Probabilistic Theory Compression/ Revision

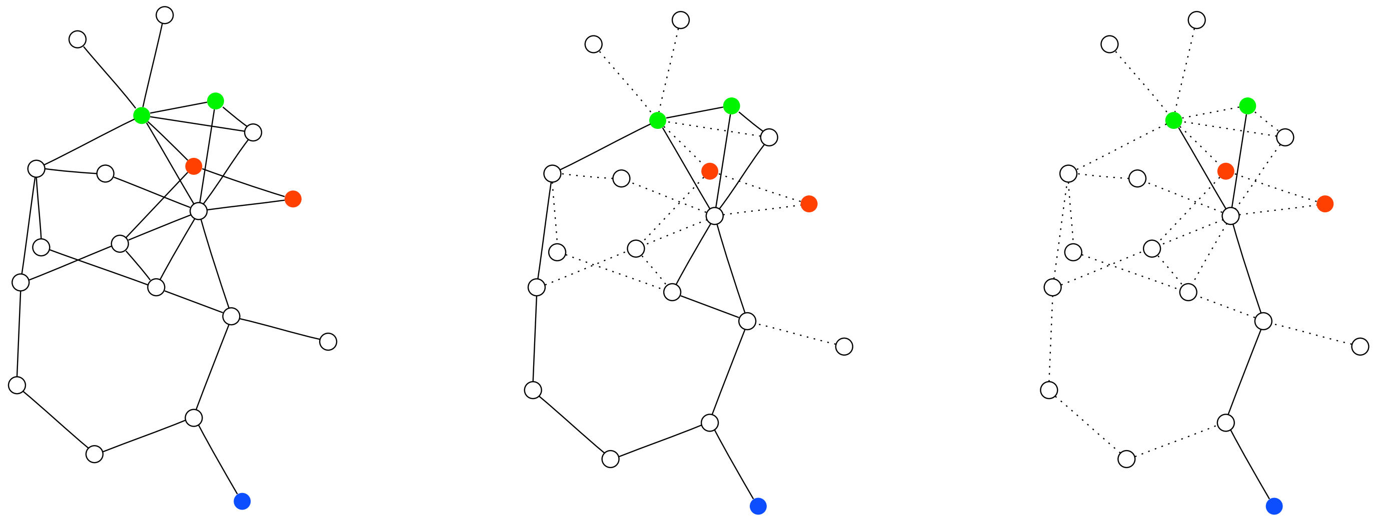


- Given
  - pos / neg interactions
  - Say (green, blue) / (red, blue)
- Find small network (k links) that maximizes prob positives and minimized prob negatives

De Raedt et al. MLJ 08

# Probabilistic Theory Compression

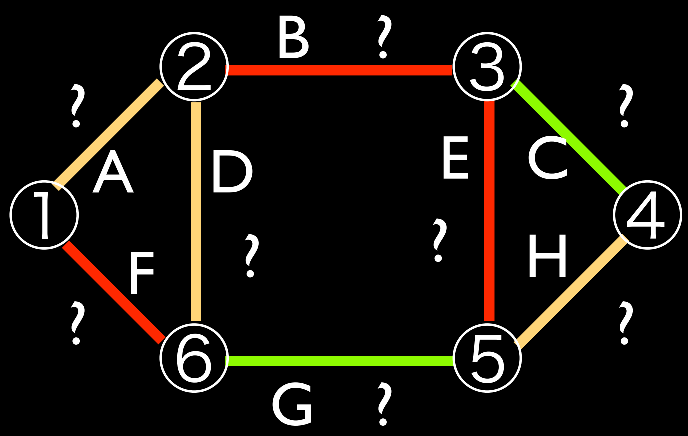
- Reduce to at most $k$ edges (greedy approach, reusing BDDs for scoring)
- Example: Green and blue should be connected, red and blue not (all edges have probability 0.5)



initially $\qquad$ $k = 15$ $\qquad$ $k = 5$

# 4. Parameter Estimation

using least squares and gradient




③→① 0.72

⑥→② 0.63

③→⑥ 0.40

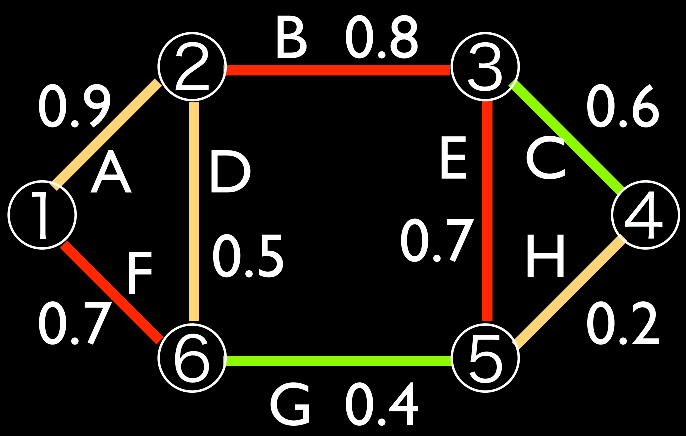
①→② 0.35

③→④ 0.14

Gutmann et al. ECML 08

# Parameter Estimation

using least squares and gradient




③→① 0.72

⑥→② 0.63

③→⑥ 0.40

①→② 0.35

③→④ 0.14

Gutmann et al. ECML 08

# Experiments

- For all of the settings specified, we did set up experiments that show that meaningful links can be (re)-discovered

# Conclusions

Logic and relational learning toolbox (take what you need)

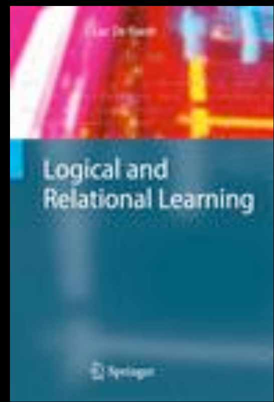
- rules & background knowledge
- generality & operators
- upgrading & downgrading
- graphs & relational database & logic
- learning settings
- propositionalization & aggregation
- probabilistic logics

# Further Reading

Luc De Raedt

Logical and Relational Learning

Springer, 2008, 401 pages, in print.




(should be on display at the Springer booth)

# Thanks to

Collaborators on previous tutorials and specific aspects of this work, esp.

- Kristian Kersting, Angelika Kimmig, Hannu Toivonen