Logical and Relational Learning
A novel synthesis

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

ECML/PKDD 2008
What is Logical and Relational Learning?

- Inductive Logic Programming
- (Statistical) Relational Learning
- Mining and Learning in Graphs
- Multi-Relational Data Mining

UNION of

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]
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
Case 2: Biological Networks

Data = Large (Probabilistic) Network

Biomine Database @ Helsinki
Network around Alzheimer Disease
presenilin 2
Gene
EntrezGene:81751

Notch receptor processing
BiologicalProcess
GO:GO:0007220
-participates_in 0.220
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

(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

<table>
<thead>
<tr>
<th>example</th>
<th>example</th>
<th>example</th>
<th>example</th>
<th>example</th>
</tr>
</thead>
</table>

*single-table*
*single-tuple*
*attribute-value*

<table>
<thead>
<tr>
<th>example</th>
<th>example</th>
<th>example</th>
<th>example</th>
<th>example</th>
</tr>
</thead>
</table>

*single-table*
*multiple-tuple*
*multi-instance*

<table>
<thead>
<tr>
<th>example</th>
<th>example</th>
<th>example</th>
<th>example</th>
<th>example</th>
</tr>
</thead>
</table>

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

<table>
<thead>
<tr>
<th>example</th>
<th>example</th>
<th>example</th>
<th>example</th>
<th>example</th>
</tr>
</thead>
</table>

*multi-table*
*multiple-tuple*
*relational*
## Attribute-Value

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

<table>
<thead>
<tr>
<th>at</th>
<th>at</th>
<th>at</th>
<th>att</th>
<th>at</th>
</tr>
</thead>
<tbody>
<tr>
<td>example</td>
<td>example</td>
<td>example</td>
<td>example</td>
<td>example</td>
</tr>
</tbody>
</table>
Multi-Instance

[Dietterich et al. AIJ 96]

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, 0.187)
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, 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)

Sometimes uses **AGGREGATION**
### Representational Hierarchy -- Systems

<table>
<thead>
<tr>
<th>single-table</th>
<th>single-tuple</th>
<th>2 relations</th>
<th>multi-table</th>
</tr>
</thead>
<tbody>
<tr>
<td>attribute-value</td>
<td>multi-instance</td>
<td>edge / vertex</td>
<td>relational</td>
</tr>
</tbody>
</table>

#### Diagram

1. Example (single-table, single-tuple, attribute-value)
2. Example (single-table, multiple-tuple, multi-instance)
3. Example (multiple-tuple, 2 relations, graphs & networks)
4. Example (multi-table, multiple-tuple, 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-tuple
- attribute-value

- single-table
- multiple-tuple
- multi-instance

- 2 relations
- edge / vertex
- graphs & networks

- multi-table
- multiple-tuple
- relational

Downgrading the data?
Propositionalization

<table>
<thead>
<tr>
<th>PARTICIPANT Table</th>
<th>COMPANY Table</th>
<th>COURSE Table</th>
<th>SUBSCRIPTION Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>JOB</td>
<td>COMPANY</td>
<td>PARTY</td>
</tr>
<tr>
<td>adams</td>
<td>researcher</td>
<td>scuf</td>
<td>no</td>
</tr>
<tr>
<td>blake</td>
<td>president</td>
<td>jvt</td>
<td>yes</td>
</tr>
<tr>
<td>king</td>
<td>manager</td>
<td>ucro</td>
<td>no</td>
</tr>
<tr>
<td>miller</td>
<td>manager</td>
<td>jvt</td>
<td>yes</td>
</tr>
<tr>
<td>scott</td>
<td>researcher</td>
<td>scuf</td>
<td>yes</td>
</tr>
<tr>
<td>turner</td>
<td>researcher</td>
<td>ucro</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table-based Propositionalization

Define new relation

\[ p(N,J,C,P,R,Co,L) :\]
\[ \text{participant}(N,J,C,P), \]
\[ \text{subscribes}(N,Co), \]
\[ \text{length}(Co,L). \]

Multi-relational → multi-instance
under certain conditions → attribute-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

<table>
<thead>
<tr>
<th>NAME</th>
<th>COURSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>adams</td>
<td>erm</td>
</tr>
<tr>
<td>adams</td>
<td>so2</td>
</tr>
<tr>
<td>adams</td>
<td>srw</td>
</tr>
<tr>
<td>blake</td>
<td>cso</td>
</tr>
<tr>
<td>blake</td>
<td>erm</td>
</tr>
<tr>
<td>king</td>
<td>cso</td>
</tr>
<tr>
<td>king</td>
<td>erm</td>
</tr>
<tr>
<td>king</td>
<td>so2</td>
</tr>
<tr>
<td>king</td>
<td>srw</td>
</tr>
<tr>
<td>miller</td>
<td>so2</td>
</tr>
<tr>
<td>scott</td>
<td>erm</td>
</tr>
<tr>
<td>scott</td>
<td>srw</td>
</tr>
<tr>
<td>turner</td>
<td>so2</td>
</tr>
<tr>
<td>turner</td>
<td>srw</td>
</tr>
</tbody>
</table>

multi-instance/tuple $\rightarrow$ attribute-value

adams, 3
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 \( \text{Le} \) to represent examples
• a logic language \( \text{Lh} \) to represent hypotheses
• a covers relation on \( \text{Le} \times \text{Lh} \)
• a loss function

Find

• A hypothesis \( h \) in \( \text{Lh} \) 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

![Chemical structures](image)
Covers Relation

Subgraph Isomorphism (bijection) or Homomorphism (injection)
Coverage

\[
\text{positive :- atom(A,c), atom(B,c), bond(A,B,s),} \\
\text{...} \\
\text{OI-subsumption (bijection)} \\
\text{or} \\
\text{theta-subsumption (injection)} \\
\]

\[
\begin{align*}
\text{atom(1,cl).} & \quad \text{atom(1,cl).} \\
\text{atom(2,c).} & \quad \text{atom(2,c).} \\
\text{atom(3,c).} & \quad \text{atom(3,c).} \\
\text{atom(4,c).} & \quad \text{atom(4,c).} \\
\text{atom(5,c).} & \quad \text{atom(5,c).} \\
\text{atom(6,c).} & \quad \text{atom(6,c).} \\
\text{atom(7,c).} & \quad \text{atom(7,c).} \\
\text{atom(8,o).} & \quad \text{atom(8,o).} \\
\text{...} & \quad \text{...} \\
\text{bond(3,4,s).} & \quad \text{bond(3,4,s).} \\
\text{bond(1,2,s).} & \quad \text{bond(1,2,s).} \\
\text{bond(2,3,d).} & \quad \text{bond(2,3,d).} \\
\end{align*}
\]
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).
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) ⊨ positive :- atom(X,c), atom(Y,o)

but also

positive :- halogen(X) ⊨ positive :- atom(X,c)
halogen(X) :- atom(X,c)
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 $B$ $U$ $G$ $\not=$ $S$

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, \neg b, \neg n\} = f \text{ IF } b \text{ and } n = f : \neg b, n

G \models S \text{ if and only if } G \subseteq S

just like item-sets
Logical Atoms

Does $g = \text{participant(adams, X, kul)}$ match $s = \text{participant(adams, researcher, kul)}$?

Yes, because there is a substitution $\theta = \{X/\text{researcher}\}$ such that $g\theta = s$

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 θ such that $G\theta \subseteq S$.

Graph - homeomorphism as special case
Subsumption Relation

Subgraph Isomorphism (bijection) or Homomorphism (injection)

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

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

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

\[ \theta = \{ G/8, A/5, B/4, C/3, D/2, E/7, F/6 \} \]
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)
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-tuple
attribute-value

single-table
multiple-tuple
multi-instance

2 relations
edge / vertex
graphs & networks

multi-table
multiple-tuple
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

Given

\[
B \cup H \models e
\]

\[
\begin{align*}
molecule(225). \\
logmutag(225, 0.64). \\
lumo(225, -1.785). \\
logp(225, 1.01). \\
nitro(225, [f1_4, f1_8, f1_10, f1_9]). \\
atom(225, f1_1, c, 21, 0.187). \\
atom(225, f1_2, c, 21, -0.143). \\
atom(225, f1_3, c, 21, -0.143). \\
atom(225, f1_4, c, 21, -0.013). \\
atom(225, f1_5, o, 52, -0.043). \\
\ldots \\
ring_size_5(225, [f1_5, f1_1, f1_2, f1_3, f1_4]). \\
hetero_aromatic_5_ring(225, [f1_5, f1_1, f1_2, f1_3, f1_4]).
\end{align*}
\]

Find

\[
\text{mutagenic}(M) : - \text{nitro}(M, R1), \logp(M, C), C > 1.
\]

\[
B \cup H \models e
\]
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

\[\text{mutagenic}(X) \leftarrow \text{atom}(X,A,n),\text{charge}(A,0.82)\]
mutagenic(X) :- atom(X,A,n),charge(A,0.82)
mutagenic(X) :- atom(X,A,c), bond(A,B)

Coverage = 0.8

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

Coverage = 0.6
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)
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
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 $L_e$ to represent examples
- a logic language $L_h$ to represent hypotheses
- a covers relation on $L_e \times L_h$
- a loss function

**Find**

- A hypothesis $h$ in $L_h$ 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: \{ \text{human(luc)}, \text{human(lieve)}, \text{male(luc)}, \text{female(lieve)} \}

Hypothesis (positives only)

(\text{maximally specific that covers example})

• \text{human}(X) :- \text{female}(X)
• \text{human}(X) :- \text{male}(X)
• false :- \text{male}(X), \text{female}(X)
• \text{male}(X); \text{female}(X) :- \text{human}(X)

OFTEN used for finding INTEGRITY CONSTRAINTS / FREQ. PATTERN MINING
Learning from Proofs

Examples

\[
\text{sentence}(A, B) :- \text{noun_phrase}(C, A, D), \text{verb_phrase}(C, D, B).
\]

\[
\text{noun_phrase}(A, B, C) :- \text{article}(A, B, D), \text{noun}(A, D, C).
\]

\[
\text{verb_phrase}(A, B, C) :- \text{intransitive_verb}(A, B, C).
\]

\[
\text{article}(\text{singular}, A, B) :- \text{terminal}(A, a, B).
\]

\[
\text{article}(\text{singular}, A, B) :- \text{terminal}(A, the, B).
\]

\[
\text{article}(\text{plural}, A, B) :- \text{terminal}(A, the, B).
\]

\[
\text{noun}(\text{singular}, A, B) :- \text{terminal}(A, \text{turtle}, B).
\]

\[
\text{noun}(\text{plural}, A, B) :- \text{terminal}(A, \text{turtles}, B).
\]

\[
\text{intransitive_verb}(\text{singular}, A, B) :- \text{terminal}(A, \text{sleeps}, B).
\]

\[
\text{intransitive_verb}(\text{plural}, A, B) :- \text{terminal}(A, \text{sleep}, B).
\]

\[
\text{terminal}([A|B], A, B).
\]

Used in Treebank Grammar Learning & Program Synthesis
Use of different Settings

Different settings provide different levels of information about target program (cf. De Raedt, AIJ 97)

- Learning from entailment
  - The most popular setting
  - Typically used for prediction
  - E.g., predicting activity of compounds

- Learning from traces/proofs
  - Typically used for hard problems, when other settings seem to fail or fail to scale up
  - E.g., program synthesis from examples, grammar induction, multiple predicate learning

Information

-
LOGIC, RELATIONS and PROBABILITY

Joint work with Kristian Kersting et al.
Logic and relations alone are often insufficient

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

First KBMC approaches:
- Breese
- Bacchus
- Charniak
- Glesner
- Goldman
- Koller
- Poole, Wellman

Prob. Horn
- Abduction: Poole

Prob. CLP: Eisele, Riezler

LPAD: Bruynooghe

Vennekens, Verbaeten

Markov Logic: Domingos, Richardson

Prob. CLP: Eisele, Riezler

Prob. CLP: Eisele, Riezler

Prob. Horn
- Abduction: Poole

PRIM: Haddawy, Ngo

1BC(2): Flach, Lachiche

RMMs: Anderson, Domingos, Weld

LOHMMs: De Raedt, Kersting, Raiko

Logical Bayesian Networks: Blockeel, Bruynooghe, Fierens, Ramon

First KBMC approaches:
- Breese
- Bacchus
- Charniak
- Glesner
- Goldman
- Koller
- Poole, Wellman

Prob. Horn
- Abduction: Poole

Prob. CLP: Eisele, Riezler

LPAD: Bruynooghe

Vennekens, Verbaeten

Markov Logic: Domingos, Richardson

Prob. CLP: Eisele, Riezler

Prob. Horn
- Abduction: Poole

PRIM: Haddawy, Ngo

1BC(2): Flach, Lachiche

RMMs: Anderson, Domingos, Weld

LOHMMs: De Raedt, Kersting, Raiko

Logical Bayesian Networks: Blockeel, Bruynooghe, Fierens, Ramon
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 $\text{covers}(e, H \cup B) = P(e \mid H, B)$

- 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 $L_e$ to represent examples
- a language $L_h$ to represent hypotheses
- a probabilistic covers $P$ relation on $L_e \times L_h$

**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

\[
P(E, B, A, J, M) = P(E) \cdot P(B) \cdot P(A|E) \cdot P(A|B) \cdot P(J|A) \cdot P(M|A)
\]

**INTERPRETATION**

**STATE/DESCRIPTION**

\[ \{A, \neg E, \neg B, J, M\} \]
Probabilistic Relational Models (PRMs)

[Getoor, Koller, Pfeffer]

Table

[Getoor, Koller, Pfeffer]
Probabilistic Relational Models (PRMs)

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).

\[ bt(Person, BT) \mid pc(Person, PC), mc(Person, MC). \]

\[ pc_father(Person, PC_f) \mid pc(Person, PC), father(Person, PC). \]
\[ mc(Person, MC) \mid pc_mother(Person, PC_m), mc(Person, MC_m). \]
\[ pc(Person, PC) \mid pc_father(Person, PC_f), mc_father(Person, MC_f). \]
\[ bt(Person, BT) \mid pc(Person, PC), mc(Person, MC). \]

\[ RV \rightarrow State \]
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

- **MC**:
  - % apriori nodes
  - nat(0).
  - % aposteriori nodes
  - nat(s(X)) | nat(X).

- **HMM**:
  - % apriori nodes
  - state(0).
  - % aposteriori nodes
  - state(s(Time)) | state(Time).
  - output(Time)   | state(Time).

- **DBN**:
  - % apriori nodes
  - n1(0).
  - % aposteriori nodes
  - n1(s(TimeSlice)) | n2(TimeSlice).
  - n2(TimeSlice)   | n1(TimeSlice).
  - n3(TimeSlice)   | n1(TimeSlice), n2(TimeSlice).

Prolog and Bayesian Nets as Special Case
Balios Tool
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

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

\[ P(\text{sentence}) = \sum_{p: \text{parsetree}} P(p) \]

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
In SLP notation

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).

\[ P(s([\text{the,turtles,sleep}],[])=1/6 \]
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(sing) -> sleep
0.5 VP(plur) -> sleeps
...

What about “A turtles sleeps”?  
P(derivation tree) = 1x1x.5x.1x .2 x.05
SLPs

\[ P_d(\text{derivation}) = \prod_i p_i^{c_i} \]
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
\[ np(\text{Num}) \rightarrow \text{art}(\text{Num}), \text{noun}(\text{Num}) \] and \( \text{art}(\text{sing}) \rightarrow \text{a} \)
\[ \text{noun}(\text{plural}) \rightarrow \text{turtles} \]

Normalization necessary
\[ P_s(\text{proof}) = \frac{P_d(\text{proof})}{\sum_i P_d(\text{proof}_i)} \]
Example Application

- Consider traversing a university website
- Pages are characterized by predicate
  \[ \text{department}(\text{cs}, \text{nebel}) \] denotes the page of cs following the link to nebel
- Rules applied would be of the form
  \[ \text{department}(\text{cs}, \text{nebel}) :\]
  \[ \text{prof}(\text{nebel}), \text{in}(\text{cs}), \text{co}(\text{ai}), \text{lecturer}(\text{nebel}, \text{ai}). \]
  \[ \text{pagetype1}(t_1, t_2) :\]
  \[ \text{type1}(t_1), \text{type2}(t_2), \text{type3}(t_3), \text{pagetype2}(t_2, t_3) \]
- SLP models probabilities over traces / proofs / web logs

\[ \text{department}(\text{cs}, \text{nebel}), \text{lecturer}(\text{nebel}, \text{ai007}), \text{course}(\text{ai007}, \text{burgard}), \ldots \]

This is actually a Logical Markov Model
Logical Markov Model

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
-participates_in 0.220

BiologicalProcess
Network around Alzheimer Disease gene phenotype probability of connection?

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 \( p : q(t_1,\ldots,t_n) \)
  • denotes that ground atoms \( q(t_1,\ldots,t_n) \) 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

 ProbLog theory T

 logical part L

 [De Raedt, Kimmig, Toivonen, IJCAI 07]
Sampling Subprograms

- Biased coins
- Independent

\[ P = 0.9 \cdot 0.8 \cdot 0.6 \cdot (1 - 0.5) \cdot (1 - 0.7) \cdot 0.7 \cdot (1 - 0.4) \cdot (1 - 0.2) \]
Queries

\[
p(x,y) := \text{edge}(x,y) \\
p(x,y) := \text{edge}(x,z), \text{path}(y,z)
\]

\[
P(q|T) = \sum_{S \subseteq L, S \models q} P(S|T)
\]
path(x,y) :- edge(x,y)
path(x,y) :- edge(x,z), path(y,z)

\[ P(q|T) = \sum_{S \subseteq L, S \models q} P(S|T) \]
Query Probability
using proofs

\[ P(\text{path}(1, 4)|T) \]
\[ = P(ABC + ABEH + ... + FDBEH) \]

- proofs overlap
- disjoint sum
- NP-hard
- approximation algorithm
  [De Raedt et al, IJCAI 07]
Prism (Sato) and ICL (Poole) avoid the disjoint problem by requiring that explanations do not overlap.
Most likely proof / explanation

Abduction

example ①→④

ABC
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,
- \[ P(A \lor B \lor C) = P(A) + P(B) + P(C) \]

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

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

Example: no burned hands

“Hey! Look what Zog do!”

Explanation: use stick
Network around Alzheimer Disease

most similar pairs?

phenotype
gene
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

1 A 2 B 3 C 4

? 3 2 1

137
Prolog Setting

proof tree

path(1,4).

edge(1,2).
y_edge(1,2).

edge(2,3).
r_edge(2,3).

edge(3,4).
g_edge(3,4).

path(2,4).

path(3,4).

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

concrete explanation
Use of Generalized Explanation
Use of Generalized Explanation

reasoning by similarity / analogy
Experiments

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

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

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

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 positives 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

no definition of path
Probabilistic Pattern Mining

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

$k = 15$

$k = 5$
4. Parameter Estimation

using least squares and gradient

\[
\begin{align*}
\theta & \rightarrow \theta_0 = 0.72 \\
\theta & \rightarrow \theta_0 = 0.63 \\
\theta & \rightarrow \theta_0 = 0.40 \\
\theta & \rightarrow \theta_0 = 0.35 \\
\theta & \rightarrow \theta_0 = 0.14
\end{align*}
\]

Gutmann et al. ECML 08
Parameter Estimation

using least squares and gradient

Gutmann et al. ECML 08

\[ \begin{align*}
\phi & \rightarrow \theta \rightarrow \omega \rightarrow \phi & \\
\phi & \rightarrow \Theta & \omega & \rightarrow \omega & \\
\phi & \rightarrow \Theta & \omega & \rightarrow \omega & \\
\phi & \rightarrow \Theta & \omega & \rightarrow \omega & \\
\end{align*} \]

\[ \begin{align*}
\phi & \rightarrow \Theta & 0.72 & \\
\phi & \rightarrow \Theta & 0.63 & \\
\phi & \rightarrow \Theta & 0.40 & \\
\phi & \rightarrow \Theta & 0.35 & \\
\phi & \rightarrow \Theta & 0.14 & \\
\end{align*} \]
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

(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