A methodology of ILP

Luc De Raedt
Albert-Ludwigs-University
Freiburg
Three messages

1. ILP applies essentially to any machine learning and data mining task, not just concept-learning
   - distance based learning, clustering, descriptive learning, reinforcement learning, bayesian approaches

2. there is a recipe that is being used to derive new ILP algorithms on the basis of propositional ones
   - not the only way to ILP
Following recipe can be used to obtain ILP systems

- 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
Three messages (2)

3. ILP as an Expressive framework has many special cases, which have often been studied separately:
   - Warmr: episodes, hierarchies, ... special case of Warmr formalism
   - Multi-instance problem
   • ILP should be used as a benchmark
   • ILP as a workbench for developing new systems
   • Desirable to have propositional learning system as a special case
Attribute value learning

• Example is single fact or tuple in table
  - e.g. playtennis(1,sunny,high,weak,humid)
• Hypothesis is set of rules with single tuple variable/literal
  pos(E) :-
    playtennis(E,Outlouk,high,Wind,humid)
Multi-instance learning

• Dietterich et al. AIJ 96
• Example is set of facts or tuples for single table
  - playtennis(1,sunny,high,weak,humid)
  - playtennis(1,sunny,medium,strong,humid)
  - ...
• Hypothesis is set of rules with single tuple variable/literal
  - pos(E) :- playtennis(E,Outlouk,high,wind,humid)
• clearly special case of ILP!
Multi-Relational Data Mining

• Essentially corresponds to using the “datalog” or “relational” subset of Prolog, or a subset thereof
  - Even though the representation may be based on graphs, E-R model, SQL etc.
  - May have good reasons (e.g. graphs are visual)
• There is nothing “sacred” about the use of Logic
• However, it is important to have
  - A well-understood representation with a sound semantics and a close correspondence between syntax and semantics.
  - This facilitates concepts like: coverage, refinement operators, ... etc.
  - Compare graphs, clausal logic and SQL
• Everything I say about ILP also holds for MRDM
Quinlan’s FOIL

• Learns Datalog (no functors) programs from examples and extensional background knowledge
• Inspired on CN2
• One of the first and best known ILP systems (in the public domain)
• Quinlan (MLJ 1990)
• Many variants!
Quinlan's FOIL ~ CN2

- Examples: true and false facts (P,N)
- Background theory:
  - extensional database (set of facts) B
  - fast!
- Finds: H such that
  - B and H |= p for all p in P
  - B and H |=\= n for all n in N
Quinlan's FOIL: an example

- \( P = \{ \text{father}(\text{luc}, \text{soetkin}), \text{father}(\text{luc}, \text{maarten}) \} \)
- \( N = \{ \text{father}(\text{lieve}, \text{soetkin}), \text{father}(\text{luc}, \text{luc}) \} \)
- \( B = \{ \text{parent}(\text{lieve}, \text{soetkin}), \text{parent}(\text{luc}, \text{soetkin}), \text{parent}(\text{lieve}, \text{maarten}), \text{parent}(\text{luc}, \text{maarten}), \text{female}(\text{soetkin}), \text{female}(\text{maarten}), \text{male}(\text{maarten}), \text{male}(\text{luc}) \} \)
- \( H : \text{father}(X,Y) :\text{male}(X), \text{parent}(X,Y) \)
Quinlan’s FOIL: algorithm

- Essentially as CN2
- covering algorithm with hill climbing
  - (beam size = 1)
- rules -> clauses
- generality -> theta-subsumption
  - specialization operator
- basic version works with Datalog only
- some problems / solutions for recursion
A FOIL example

father(X,Y)

father(X,X)

father(X,Y) :-
  female(X)

father(X,Y) :-
  male(Y)

father(X,Y) :-
  male(X)

father(X,Y) :-
  male(X), parent(X,Y)

father(X,Y) :-
  male(X), female(Y)
Subsumption in propositional logic

pos

pos :- p
pos :- q
pos :- r

pos :- p,q
pos :- p,r
pos :- q,r

pos :- p,q,r
Subsumption in simple logical atoms

\[ P(X,Y,Z) \]
\[ P(a,Y,Z) \quad \ldots \quad P(X,b,Z) \quad \ldots \quad P(X,Y,c) \]
\[ P(a,b,Z) \quad \ldots \quad P(a,Y,c) \quad \ldots \quad P(X,b,c) \]
\[ P(a,b,c) \]
Subsumption in simple logical atoms

\[ P(X, Y) \]

\[ P(X, X) \quad \ldots \quad P(a, Y) \quad P(b, Y) \quad \ldots \quad P(X, a) \quad \ldots \quad P(X, b) \]

\[ P(a, a) \quad \ldots \quad P(a, b) \quad \ldots \quad P(b, b) \quad \ldots \]
Subsumption in logical atoms

\[ P(X) \]

\[ P(f(Y)) \ldots P(g(Y)) \ldots P(h(Y,Z)) \ldots \]

\[ P(f(f(W))) \quad P(f(g(W))) \]

\[ P(f(f(f(U)))) \ldots \]

\[ P(f(f(f(f(V)))) \ldots \]
Structure

\[
p(X,Y) :- m(X,Y)
\]
\[
p(X,Y) :- m(X,Y), m(X,Z)
\]
\[
p(X,Y) :- m(X,Y), m(X,Z), m(X,U)
\]
\[
p(X,Y) :- m(X,Y), s(X)
\]
\[
p(X,Y) :- m(X,Y), m(X,Z), s(X)
\]
\[
p(X,Y) :- m(X,Y), s(X), r(X)
\]
\[
p(X,Y) :- m(X,Y), m(X,Z), s(X), r(X)
\]

 Reduced
Muggleton’s Progol NGC 95

- Based on AQ and versionspaces
- Richest ILP system from logical perspective
  - illustration of logical method/approach in ILP
  - Probably not a direct upgrade
- Examples are true and false horn clauses
- Background: Prolog program
- Example e is covered if $B \cup H \models e$
- Illustration: see FOIL example
Progol Method

• Covering algorithm

• Rule finding algorithm:
  - AQ based: take (positive) seed to guide search
    • find maximally specific clause covering example within given bias (inverse entailment)
    • then perform general to specific search under theta-subsumption bounded by this clause
  - compression based heuristic
Inverse entailment

\[ B \cup h \models e \iff B \cup \neg e \models \neg h \]

find all atoms entailed by \( B \cup \neg e \)
call this \( \neg h \) and negate again to
give maximally specific \( h \)
called bottom clause
Inverse entailment example

\[ B = \{ \text{mammal}(X) :\text{-} \text{dog}(X); \]
\[ \quad \text{mammal}(X) :\text{-} \text{cat}(X); \]
\[ \quad \text{cat}(\text{saar}) \} \]
\[ e = \text{nice}(\text{saar}) \]

let bias assume arguments in examples must be mentioned in entailed facts!

\[ B \cup \neg e \models \text{mammal}(\text{saar}), \text{cat}(\text{saar}), \neg \text{nice}(\text{saar}) \]
gives \[ h = \text{nice}(\text{saar}) :\text{-} \text{mammal}(\text{saar}), \text{cat}(\text{saar}) \]
Bounded search in Progol

nice(saar) :- mammal(saar), cat(saar)

dog does not appear in search!
Distance Based Learning

• Bisson (AAAI 92):
  - ako first order distance used in a clustering system
  - working system but not really well-founded?
    • Distances measure
    • Clustering algorithm not along the lines of propositional algorithms
RIBL

• Emde and Wettschereck (ICML 96)
  - use Bisson’s distance for Relational Instance Based Learning (RIBL)
  - follows well-known algorithm (k-nearest neighbour) with semi-distance
  - excellent performance! (e.g. on NMR)
Distances

• Used distances measures were not proper distances
• distance $d$ should satisfy
  - $d(x,y)=0 \iff x=y$ (reflexive)
  - $d(x,y)=d(y,x)$ (symmetric)
  - $d(x,y)+d(y,z) \geq d(x,z)$ (triangle inequality)
• Work by Nienhuys-Cheng, Hutchinson, Ramon et al., Sebag
We need

- Some distance between atoms
- Some method to upgrade a distance on a space of points to a distance on the space of sets of points.
  - As a special case we have then a distance between sets of atoms
- Some method to upgrade a distance between sets of atoms to a distance between clauses (taking into account common variables in different atoms)
• Definition for Atoms:
  - \( d(e,e)=0 \)
  - \( d(p(t_1,\ldots,t_m),q(s_1,\ldots,s_n))=1 \)
    for \( p \neq q \) or \( n \neq m \)
  - \( d(p(t_1,\ldots,t_m),p(s_1,\ldots,s_m))=\frac{(d(t_1,s_1)+\ldots+d(t_m,s_m))}{2m} \)

• e.g. \( d(p(f(a),c,e),p(f(b),c,d))= \frac{(d(f(a),f(b))+d(c,c)+d(e,d))}{2.3} = \frac{(\frac{d(a,b)}{2.1}+0+1)}{6} = \frac{1}{2}+1=\frac{3}{12}=\frac{1}{4} \)

• Can be extended using Haussdorf distance methods for sets ...
• Given a real distance measure:
  - develop ILP systems starting from propositional ones
  - change representation
  - in some cases (mRNA - Horvath et al.):
    • incorporate distance measure from application domain as a piece of background knowledge, e.g. homology
    • work with lists and terms (edit distances)
Systems and techniques

- RIBL 2 (Horvath et al. MLJ)
  - applied to mRNA signal structure detection
- RDBC (Kirsten Wrobel ILP98),(Kirsten Ph.D. 02)
  - agglomerative clustering method
  - K-means and k-medoid
- AP (Muggleton and Bain ILP 98)
  - analogical prediction
- IBFL (Ramon and De Raedt ILP 98)
  - predict functions (from structured terms to structured terms)
- TIC (Blockeel et al. ICML 98):
  - top-down clustering/regression
Distances

- Bisson AAAI 92
- Hutchinson ECML 97
- Ramon and Bruynooghe ILP 98, Acta Informatica
- Horvath et al MLJ
- Nienhuys-Cheng ILP
- Theses
  - Ramon 02 / Kirsten 02
First order association rules
Dehaspe’s Warmr ~ Apriori

• Finds first order association rules from database
• Based on Agrawal et al.’s Apriori
• (Dehaspe et al., DMKD Journal 99, KDD 98, ILP 97)
What to count? Keys.

**COMPANY Table**

<table>
<thead>
<tr>
<th>COMPANY</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>jvt</td>
<td>commercial</td>
</tr>
<tr>
<td>scuf</td>
<td>university</td>
</tr>
<tr>
<td>ucro</td>
<td>university</td>
</tr>
</tbody>
</table>

**COURSE Table**

<table>
<thead>
<tr>
<th>COURSE</th>
<th>LENGTH</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>cso</td>
<td>2</td>
<td>introductory</td>
</tr>
<tr>
<td>erm</td>
<td>3</td>
<td>introductory</td>
</tr>
<tr>
<td>so2</td>
<td>4</td>
<td>introductory</td>
</tr>
<tr>
<td>srw</td>
<td>3</td>
<td>advanced</td>
</tr>
</tbody>
</table>

**SUBSCRIPTION Table**

<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>

**PARTICIPANT Table**

<table>
<thead>
<tr>
<th>NAME</th>
<th>JOB</th>
<th>COMPANY</th>
<th>PARTY</th>
<th>R_NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>adams</td>
<td>researcher</td>
<td>scuf</td>
<td>no</td>
<td>23</td>
</tr>
<tr>
<td>blake</td>
<td>president</td>
<td>jvt</td>
<td>yes</td>
<td>5</td>
</tr>
<tr>
<td>king</td>
<td>manager</td>
<td>ucro</td>
<td>no</td>
<td>78</td>
</tr>
<tr>
<td>miller</td>
<td>manager</td>
<td>jvt</td>
<td>yes</td>
<td>14</td>
</tr>
<tr>
<td>scott</td>
<td>researcher</td>
<td>scuf</td>
<td>yes</td>
<td>94</td>
</tr>
<tr>
<td>turner</td>
<td>researcher</td>
<td>ucro</td>
<td>no</td>
<td>81</td>
</tr>
</tbody>
</table>
Coverage

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

\[ \exists (K,C,L,T): \text{subscription}(K,C) \land \text{course}(K,C,L,T) \land (L < 3) \]

?- \text{subscription}(K,C), \text{course}(K,C,L,_) , (L < 3).

K=blake
C = cso
L = 2
Yes

K=turner
No

E: BLAKE
KING
MILLER
BLAKE
ADAMS
SCOTT
TURNER

participant(blake,president,jvt,yes,5).
subscription(blake,cso).
subscription(blake,erm).
course(cso,2,introductory).
course(erm,3,introductory).
course(so2,4,introductory).
course(srw,3,advanced).

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

multi-relational database

association rules over multiple relations

Registration database

Selected, Preprocessed, and Transformed Data

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

support: 20 %
confidence: 75 %

“IF participant follows an advanced course
THEN she skips the welcome party”

support: 20 %
confidence: 75 %
Warmr

- First order association rule:
  - IF Query1 THEN Query2
  - Shorthand for
    - IF Query1 THEN Query1 and Query2
    - to obtain variable bindings
- IF ?- participant(P,C,PA,X), course(P,Y,advanced) THEN ?- PA=no
- IF ?- participant(P,C,PA,X), course(P,Y,advanced) succeeds for P THEN ?- participant(P,C,PA,X), course(P,Y,advanced), PA=no succeeds for P

- Counting:
  - number of 'keys' for which queries succeed
Warmr ~ Apriori

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

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

EMPTY(10)

A(5)                    B(6)                     C(3)

AB(4)           AC(2)           BC(2)

ABC(2)
Adaptation of Apriori

Init Level  d := 0
Init Q0 := { T }
Init Frequent := ∅
Init Infrequent := ∅
While Qd not empty do
    find frequency(query) for all queries in Qd
    delete the queries q from Qd with frequency(q) < minfreq
    and add to Infrequent
    Frequent := Frequent U Qd
    compute Q(d+1)
    d:= d + 1
Return F
Levelwise computing $Q(d+1)$

$init \ Q(d+1) := \emptyset$
for each $q$ in $Q_d$
    compute $\rho(q)$
    add all $q'$ in $\rho(q)$ to $Q(d+1)$ for which $\rho^{-1}(q') \cap infrequent$ is not empty
Finding association rules

From frequent queries
  \( q \)
  \( q \) and \( lit \)
derive
  if \( q \) then \( lit \)
Ranking Association Rules

- Given frequent patterns:
  - find all association rules
  - often too many

- Also, statistically insignificant ones:
  \[ P(\text{cheese}) = 0.66 \]
  \[ P(\text{cheese} \mid \text{bread and butter}) = 0.66 \]
  if bread and butter then cheese
  \[ p(\text{conclusion} \mid \text{condition}) \sim p(\text{conclusion}) \]
  statistical significance tests to rank association rules (e.g. binomial)
Significance ranking

• Suppose:
  - if condition then conclusion
• compute $p = p(\text{conclusion})$ on data set
• assume the probability that a tuple satisfies the conclusion is binomially distributed with probability $p$
• compute $q = p(\text{conclusion} \mid \text{condition})$ on data set
• compute the tightest confidence interval that $q$ falls into assuming $q$ is a sample from the probability distribution.
• If interval larger than say 99% rule is considered interesting or significant
Variants

Nijssen IJCAI 01
efficient
De Raedt
unpublished
Stahl PKDD 99
Wrobel et al
KDD Subgroup Discovery
Dehaspe and De Raedt MLJ 97
Claudien - pre-decessor of Warmr
PAC-learning and Claudien

First result in PAC-learning
(Valliant CACM 84):

$k$-CNF is learnable from propositional interpretations
($k = \text{bound on nof literals}$)

$$h_1 \lor \ldots \lor h_n \leftarrow b_1 \land \ldots \land b_m$$
PAC-learning and Claudien

First result in learning from interpretations (De Raedt - Dzeroski AIJ 94):
jk-Clausal Theories are learnable from finite Herbrand interpretations
(k = bound on nof literals,
j = bound on size of literal)
\( h_1 \lor \ldots \lor h_n \leftarrow b_1 \land \ldots \land b_m \)
Characteristic induction

Given:
- one set of herbrand interpretations $P$
  (possibly computed using common background)
- definition of hypothesis space (language bias) $L$

Find:
- all clauses in $L$ that satisfy all $p$ in $P$
  (all regularities expressible in $L$ that hold in $P$)
Claudien: descriptive/characteristic induction

Example: one interpretation
{ human(luc), human(lieve), male(luc), female(lieve) }
L: no constants/one variable
Find:
  human(X) :- male(X).
  human(X) :- female(X).
  female(X); male(X) :- human(X)
false :- male(X), female(X).
Claudien : search space

Optimal refinement operator !
First order decision trees
Tilde ~ C4.5

class(e1, fix)
worn(e1, gear)
worn(e1, chain)
First order decision tree

- `worn(E,X)`
  - `not_replaceable(E,X)`
  - `class(E,sendback)`
  - `class(E,fix)`
  - `class(E,keep)`
Equivalent Prolog Program

class(E, sendback) :- worn(E, X), not_replaceable(E, X), !.
class(E, fix) :- worn(E, X), !.
class(E, keep).
cuts can be eliminated with predicate invention.

To classify example e:
ask ?- class(e, C) to prolog database B and H
e.g. ?- class(e3, C) results in C = sendback
Tilde Method

• Employ C4.5 (Quinlan) method and heuristics
• C4.5 is a special case of Tilde when
  - binary tests
  - single propositional literals in leaves
• Tilde inherits well-known properties of C4.5
  - heuristics, noise-handling, speed!
  - Blockeel and De Raedt AIJ 98
• Specialization operator uses theta-subsumption (on partly built clauses)
• See also Kramer AAAI 96 SRT
  - Logical regression trees!
Large Data Sets

• Classical trade-off between in computer science between expressivity and efficiency
• Also applies to ILP versus AVL
• If AVL applicable then use it
• However, ILP can handle large datasets!
ILP and efficiency

• Four key ideas:
  - use explicit partitioning of database
    - learn from interpretations
  - invert
    - for all hypotheses in queue do
      - for all examples in database do
        » test coverage
  - Bias
  - Optimized prolog/ilp engines ILProlog and Query Packs (Blockeel et al. JAIR 02)
ILP and efficiency

• Partitioning of database:
  - problem to load whole database in CPU
    • beyond scope of Prolog implementations
  - solution: partition database into examples
  - load examples one by one
  - minimize number of passes through database
  - inspired on propositional data mining
  - e.g. (Mehta et al. EDBT 96), (Agrawal)
Warmr / Tilde for Large Data Sets

• Cf. (Blockeel et al. DMKD 99)

• Mutagenesis dataset:
  - 188 examples, 10512 facts, size 250 KB
  - TildeLDS 123 sec. (compilation 3 sec)
  - multiply dataset by 2, 4, 8, ..., 512
  - concepts remains constant
  - largest dataset 130 Mbyte, 96000 exs., 76000 secs ~ 21 cpu hours

• Contrast with initial results on mutagenesis
  - same order of magnitude for single run!
Lineair in examples
Propositionalization

• Debate on AVL versus ILP
• ILP can theoretically be transformed into AVL
  - features = e.g. queries (as in Warmr)
  - BUT combinatorial explosion of features and datasets
  - intractable
• Retain only relevant features
  - this is propositionalization
  - relevance heuristic necessary
  - declarative restrictions
Propositionalisation

• Use e.g. Warmr’s features in AVL learners such as C4.5
  - excellent results in PTE-2
• Hybrid approaches exist
  - propositionalise while learning (Alphonse and Rouveiro)
  - derive features and use distance (Sebag)
  - Kramer et al. ILP 98
  - Flach and Lachiche  ILP 1BC
Other upgrades

- Bayesian Logic Programs and PRMS
  - Upgrade Bayesian Nets
- Stochastic Logic Programs (Muggleton) upgrade stochastic context free grammars
- Merlin (Bostrom) upgrades induction of finite state automata to class of logic programs
Relational reinforcement learning (Dzeroski et al MLJ 01)

- combining reinforcement learning and inductive logic programming
- proof of the principle
- block’s world
- learning to plan optimally
  - effects of actions unknown to learner
  - preconditions known
  - reward = 1 if and only if goal reached
- Q-learning and Top-down induction of logical decision trees
RRL: learning to plan and act

• **Given**:  
  - \textbf{unknown} \( \text{delta} : S \times A \rightarrow S \)  
  - \( \text{pre} : S \times A \rightarrow \{\text{true, false}\} \)  
  - \( \text{goal} : S \rightarrow \{\text{true, false}\} \)  
  - a starting state \( s \) in \( S \)  
  - \( r : S \times A \rightarrow \{0, 1\} \) (\( r=1 \) iff \( \text{goal}(\text{delta}(s, a)) \))  

• **Find**: a minimal sequence of actions \( a_1, \ldots, a_n \) in \( A \) such that  
  - \( \text{goal}(\text{delta}(\ldots(\text{delta}(s, a_1) \ldots a_n)\ldots)) = \text{true} \)  
  - \( \text{pre}(\text{delta}(\ldots(\text{delta}(s, a_1) \ldots a_i)\ldots)) = \text{true} \) for all \( i \)  
  - \( \max \sum_i \gamma^i r_i \)
RRL for planning and acting

• Planning but
  - effects of actions UNKNOWN delta

• Policy learning :
  - $\Pi : S \rightarrow A$ gives action $a_t$ in $s_t$
  - $r_t : S \times A \rightarrow \{0,1\}$ gives 1 iff
    $\text{goal}(\text{delta}(s,a)) = \text{true}$
  - goal states are absorbing
  - $\text{argmax}_{\Pi} V^\Pi (s_t) = \sum \gamma^i r_{t+i} (s_{t+i},a_{t+i})$
Example

• Planning in block’s world
• Classical actions:
  - move(X,Y) where X is block and Y is block or floor,
  - move(X,Y) only possible when X and Y are free or X is free and Y is floor
• Classical goals: on(a,b), on(b,c)
Q-learning

• Encode policy

  - \( \Pi : S \rightarrow A \) gives action \( a_t \) in \( s_t \)
  - \( \Pi(s) = \text{argmax}_a Q(s,a) \)
  - \( Q(s,a) \) gives immediate reward by executing \( a \) in \( s \) + discounted sum of rewards by following optimal policy after this
  - Q-learning learns \( Q \)
An example: on(a,b)

- Move(c,floor) $Q=0.81$
- Move(b,c) $Q=0.9$
- Move(a,b) $Q=1.0$
RRL: TILDE regression

Induced regression tree: e.g.
- action(move(A,B)), goal(on(C,D))
- on(C,D)?
  • Yes: 0
  • no: action(move(C,D))?
    • Yes: 1
    • No: action(move(D,B))?
      • Yes: 0.9
      • No: 0.81
action(move(A,B)) , goal(on(C,D))

on(C,D) ?
  +--yes: [0]
  +--no: action(move(C,D)) ?
    +--yes: [1]
    +--no: on(B,C) ?
      +--yes: [0.729]
      +--no: on(B,D) ?
        +--yes: [0.729]
        +--no: action(move(A,C)) ?
          +--yes: [0.81]
          +--no: action(move(A,D)) ?
            +--yes: [0.81]
            +--no: clear(D) ?
              +--yes: on(C,B) ?
                +--yes: on(A,C) ?
                  +--yes: [0.9]
                  +--no: clear(C) ?
                    +--yes: [0.9]
                    +--no: [0.81]
                +--no: [0.9]
              +--no: clear(C) ?
                +--yes: on(C,B) ?
                  +--yes: [0.9]
                  +--no: [0.81]
                +--no: [0.81]
Conclusions

• There is a recipe to derive new ILP algorithms based on existing propositional ones
  - not the only approach in ILP
• ILP not only applies to concept-learning but to essentially any machine learning or data mining problem
• ILP is an expressive framework that has many others as a special case
The future of ILP

• Lies in the upgrading of
  - Probabilistic Models
  - Support Vector Machines
  - Reinforcement Learning
  - Neural Networks
  - ...

The future of ILP

• Lies in downgrading ILP
  - To make it more efficient for specific applications
  - E.g. sequences
    • Consider a query
      - Theta-subsumption is NP-complete
    • Consider the query as an ordered sequence
      - Substring matching is polynomial!
      - latex(ecml, tex) xdvi(ecml, dvi) dvips(ecml, dvi)
        lpr(ecml, ps)
Thanks to
Hendrik Blockeel, Luc Dehaspe,
Saso Dzeroski, Nada Lavrac,
Jan Ramon and Wim Van Laer
Generality

$G$ is more general than $S$ iff and only if $G \models S$

because

$KB \cup S \models e$ (S covers e)
we know $G \models S$ so $KB \cup U \cup G \models KB \cup U \cup S$
therefore $KB \cup U \cup G \models e$ ($G$ covers e)
Generality

\[ \text{pos}(K) : \neg \text{triangle}(K, T) \]

\[ \text{pos}(K) : \neg \text{triangle}(K, T), \text{circle}(K, C) \]

\[ \text{pos}(K) : \neg \text{triangle}(K, T), \text{circle}(K, C), \text{in}(K, T, C) \]

Is more general
A more complex example

grandparent(X,Y) :- parent(X,Z), parent(Z,Y)
parent(X,Z) :- father(X,Z)

|- 

grandparent(X,Y) :- father(X,Z), parent(Z,Y)
\[ G \models S \]

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 |- 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
  - full first order theory

• Depending on the choice of $\vdash$ to invert
  - theta subsumption (most popular !)
  - resolution
  - implication
Subsumption in Propositional logic

Clause $g$ subsumes clause $s$ if and only $g \models s$
or, equivalently
$g \subseteq s$

$pos : - p,q,r \models pos : - p,q,r,s,t$
because
$\{pos, \neg p, \neg q, \neg r\} \subseteq \{pos, \neg p, \neg q, \neg r, \neg s, \neg t\}$
Subsumption in propositional logic

pos

pos :- p
pos :- q
pos :- r

pos :- p, q
pos :- p, r
pos :- q, r

pos :- p, q, r
Subsumption in propositional logic

- Perfect structure
- Complete lattice
  - any two clauses have unique
    - least upper bound (least general generalization)
    - greatest lower bound
- No syntactic variants
- Easy specialization, generalization
Subsumption in logical atoms

• $g$ subsumes $s$ if and only if there is a substitution $\theta$ such that $g\theta = s$
• e.g. $p(X,Y,X)$ subsumes $p(a,Y,a)$
• e.g. $p(f(X),Y)$ subsumes $p(f(a),Y)$
Subsumption in simple logical atoms

\[ P(X, Y, Z) \]

\[ P(a, Y, Z) \quad \ldots \quad P(X, b, Z) \quad \ldots \quad P(X, Y, c) \]

\[ P(a, b, Z) \quad \ldots \quad P(a, Y, c) \quad \ldots \quad P(X, b, c) \]

\[ P(a, b, c) \]
Subsumption in simple logical atoms

\[ P(X,Y) \]

\[ P(X,X) \ldots \quad P(a,Y) \quad P(b,Y) \ldots \quad P(X,a) \ldots P(X,b) \]

\[ P(a,a) \ldots \quad P(a,b) \ldots \quad P(b,b) \ldots \]
Subsumption in logical atoms

\[ P(X) \]

\[ \begin{align*}
P(f(Y)) & \quad \ldots \\
P(g(Y)) & \quad \ldots \\
P(h(Y,Z)) & \quad \ldots
\end{align*} \]

\[ \begin{align*}
P(f(f(W))) & \quad P(f(g(W))) \\
P(f(f(f(U)))) \quad \ldots
\end{align*} \]

\[ \begin{align*}
P(f(f(f(U)))) \quad \ldots
\end{align*} \]

\[ \begin{align*}
P(f(f(f(f(V))))) \quad \ldots
\end{align*} \]
Subsumption in logical atoms

• $g$ subsumes $s$ if and only if there is a substitution $\theta$ such that $g\theta = s$
• Still nice properties and complete lattice up to variable renaming
  - $p(X,a)$ and $p(U,a)$
  - greatest lower bound = unification
  - unification $p(X,a)$ and $p(b,U)$ gives $p(a,b)$
  - least upper bound = anti-unification = lgg
  - lgg $p(X,a,b)$ and $p(c,a,d) = p(X,a,Y)$
  - lgg $p(X,f(X,c))$ and $p(a,f(a,Y))$ gives $p(U,f(U,T))$
Lgg of atoms

• lgg of terms:
  \[ lgg(t,t) = t \]
  \[ lgg(f(s_1, ..., s_n), f(t_1, ..., t_n)) = f(lgg(s_1, t_1), ..., lgg(s_n, t_n)) \]
  \[ lgg(f(s_1, ..., s_n), g(t_1, ..., t_m)) = V \text{ (throughout)} \]

• lgg of atoms:
  \[ lgg(p(s_1, ..., s_n), p(t_1, ..., t_n)) = p(lgg(s_1, t_1), ..., lgg(s_n, t_n)) \]
  \[ lgg(p(s_1, ..., s_n), q(t_1, ..., t_m)) = \text{undefined} \]
Operators

• **Specialization**:
  - apply a substitution \( \{ X / Y \} \) where \( X,Y \) already appear in atom
  - apply a substitution \( \{ X / f(Y_1, \ldots, Y_n) \} \) where \( Y_i \) new variables
  - apply a substitution \( \{ X / c \} \) where \( c \) is a constant

• **Generalization**:
  - apply an inverse substitution
  - turn ‘term’ into variable
    - \( p(a,f(b)) \) becomes \( p(X,f(b)) \) or \( p(a,f(X)) \)
    - \( p(a,a) \) becomes \( p(X,X) \) or \( p(a,X) \) or \( p(X,a) \)
  - replace two occurrences of variable \( X \) into \( X_1 \) and \( X_2 \)
    - \( p(X,X) \) becomes \( p(X_1,X_2) \)
Theta-subsumption (Plotkin 70)

- Most important framework for inductive logic programming. Used by all major ILP systems.
- S and G are single clauses
- Combines propositional subsumption and subsumption on logical atoms
- \( c_1 \) theta-subsumes \( c_2 \) if and only if there is a substitution \( \theta \) such that \( c_1 \theta \subseteq c_2 \)

\[
\begin{align*}
\text{c1 : father(X,Y) :- parent(X,Y),male(X)} \\
\text{c2 : father(jef,paul) :- parent(jef,paul), parent(jef,an), male(jef), female(an)} \\
\theta = \{ X / jef, Y / paul \}
\end{align*}
\]
\[ d_1 : p(X,Y) :- q(X,Y), q(Y,X) \]
\[ d_2 : p(Z,Z) :- q(Z,Z) \]
\[ d_3 : p(a,a) :- q(a,a) \]
\[ \text{theta}(1,2) : \{X / Z, Y / Z\} \]
\[ \text{theta}(2,3) : \{Z/a\} \]
\[ d_1 \text{ is a generalization of } d_3 \]
\[ \text{contrast with Michalski's 'turning constants into variables'}! \]
\[ \text{Mapping several literals onto one leads (sometimes) to combinatorial problems} \]
Properties

- Soundness: if \( c_1 \) theta-subsumes \( c_2 \) then \( c_1 \models c_2 \)
- Incompleteness (but only for self-recursive clauses) wrt logical entailment
  - \( c_1 : p(f(X)) :- p(X) \)
  - \( c_2 : p(f(f(Y))) :- p(Y) \)
- Decidable (but NP-complete)
- Transitive and reflexive but not anti-symmetric
Structure

\[ p(X,Y) :- m(X,Y) \]
\[ p(X,Y) :- m(X,Y), m(X,Z) \]
\[ p(X,Y) :- m(X,Y), m(X,Z), m(X,U) \]

\[ p(X,Y) :- m(X,Y), r(X) \]
\[ p(X,Y) :- m(X,Y), m(X,Z), r(X) \]

\[ p(X,Y) :- m(X,Y), s(X) \]
\[ p(X,Y) :- m(X,Y), m(X,Z), s(X) \]

\[ p(X,Y) :- m(X,Y), r(X) \]
\[ p(X,Y) :- m(X,Y), m(X,Z), r(X) \]

 reduced
Properties (2)

• Equivalence classes \([c]\):
  - \(\text{parent}(X,Y) : \neg \text{mother}(X,Y), \neg \text{mother}(X,Z)\)
  - \(\text{parent}(X,Y) : \neg \text{mother}(X,Y)\)

• \(c_1\) reduced clause of \(c_2\) iff \(c_1\) minimal subset of literals of \(c_2\) that is equivalent with \(c_2\)
  - \(\text{parent}(X,Y) : \neg \text{mother}(X,Y), \neg \text{mother}(X,Z)\)
  - \(\text{parent}(X,Y) : \neg \text{mother}(X,Y) : \text{reduced form}\)
  - this gives an algorithm for reduction
  - reduced class = representative of equivalence class, unique up to variable renaming
Properties (3)

• Equivalence classes induce a lattice \( L \)
  - any two equivalence classes have least upper bound (least general generalization - lgg)
  - any two equivalence classes have greatest lower bound

• infinite descending and ascending chains exist, e.g.
  - \( h(X_1,X_2) \)
  - \( h(X_1,X_2) :- p(X_1,X_2) \)
  - \( h(X_1,X_2) :- p(X_1,X_2), p(X_2,X_3) \)
  - ....
  - \( h(X,X) :- p(X,X) \)
Lgg of clauses

- lgg of literals (= atoms or negated atoms):
  - lgg(atom1, atom2) = see above
  - lgg(not atom1, not atom2) = not lgg(atom1, atom2)
  - lgg(not atom1, atom2) = undefined

- lgg of clauses:
  - lgg({l1, ..., lm}, {k1, ..., kn}) = {lgg(li, kj) | lgg(li, kj) defined}

- f(t, a) :- p(t, a), m(t), f(a)
- f(j, p) :- p(j, p), m(j), m(p)
- lgg = f(X, Y) :- p(X, Y), m(X), m(Z)
Specialization Operators

- **Refinement operator (Shapiro):**
  - $\rho(c) \subseteq \{c' \mid c' \text{ is a maximally general specialization of } c\}$ (theory)
  - $\rho(c) \subseteq \{c \cup \{l\} \mid l \text{ is literal}\} \cup \{c\theta \mid \theta \text{ is a substitution}\}$ (practice)
  - $\rho(\text{parent}(X,Y))$ includes:
    - $\text{parent}(X,X)$
    - $\text{parent}(X,Y) :- \text{male}(X)$
    - $\text{parent}(X,Y) :- \text{parent}(Y,Z)$,
    - ....
**d**: daughter, **p**: parent, **f**: female, **m**: male

\[
\begin{align*}
  d(X,Y) & \quad d(X,X) \\
  d(X,Y) & \quad p(Y,X) \\
  d(X,Y) & \quad d(X,Y) \\
  d(X,Y) & \quad d(X,Y) \\
  f(X) & \quad d(X,Y) \\
  p(X,Y) & \quad d(X,Y) \\
  d(X,Y) & \quad d(X,Y) \\
  p(X,Z) & \quad d(X,Y) \\
  f(X) & \quad d(X,Y) \\
  f(X) & \quad d(X,Y) \\
  p(X,Y) & \quad d(X,Y) \\
  f(X),f(Y) & \quad d(X,Y) \\
  f(X),p(X,Y) & \quad d(X,Y)
\end{align*}
\]
Generalization operator

- On single clause:
  - should return all proper minimal generalizations of given clause
  - problematic for some clauses
  - e.g. h(X,X) :- p(X,X)
  - infinite clauses!
  - Bound the size of clauses
  - better: to start from two clauses and apply lgg
Conclusions

• There is a recipe to derive new ILP algorithms based on existing propositional ones
  - not the only approach in ILP
• ILP not only applies to concept-learning but to essentially any machine learning or data mining problem
• ILP is an expressive framework that has many others as a special case
Successful instances

- Learning rules
  - Foil, Progol, ICL
- Distance based learning
  - RIBL, RDBC, TIC, AP, ...
- First order association rules and decision trees
  - Warmr/Tilde, ...
- Computational learning theory
  - Learning from interpretations
- ...

- Large Data Sets
- Bayesian approaches
  - 1BC, SLP, BLP, PRMs, LOHMMs, ...
- Relational Reinforcement Learning
- Open
  - Support Vector Machines
  - Neural Nets
The future of ILP

- Lies in the upgrading of
  - Probabilistic Models ?
  - Support Vector Machines ??
  - Reinforcement Learning ??
  - Neural Networks ???
  - ...

The future of ILP

• Lies in downgrading ILP
  - To make it more efficient for specific applications
  - E.g. sequences
    • Consider a query
      - Theta-subsumption is NP-complete
    • Consider the query as an ordered sequence
      - Substring matching is polynomial!
      - latex(ecml,tex) xdvi(ecml,dvi) dvips(ecml,dvi)
      - lpr(ecml,ps)
Others

• Revisit Theory revision?
• Declarative ILP and RDM
  – Inductive databases !!!