Inductive Logic Programming

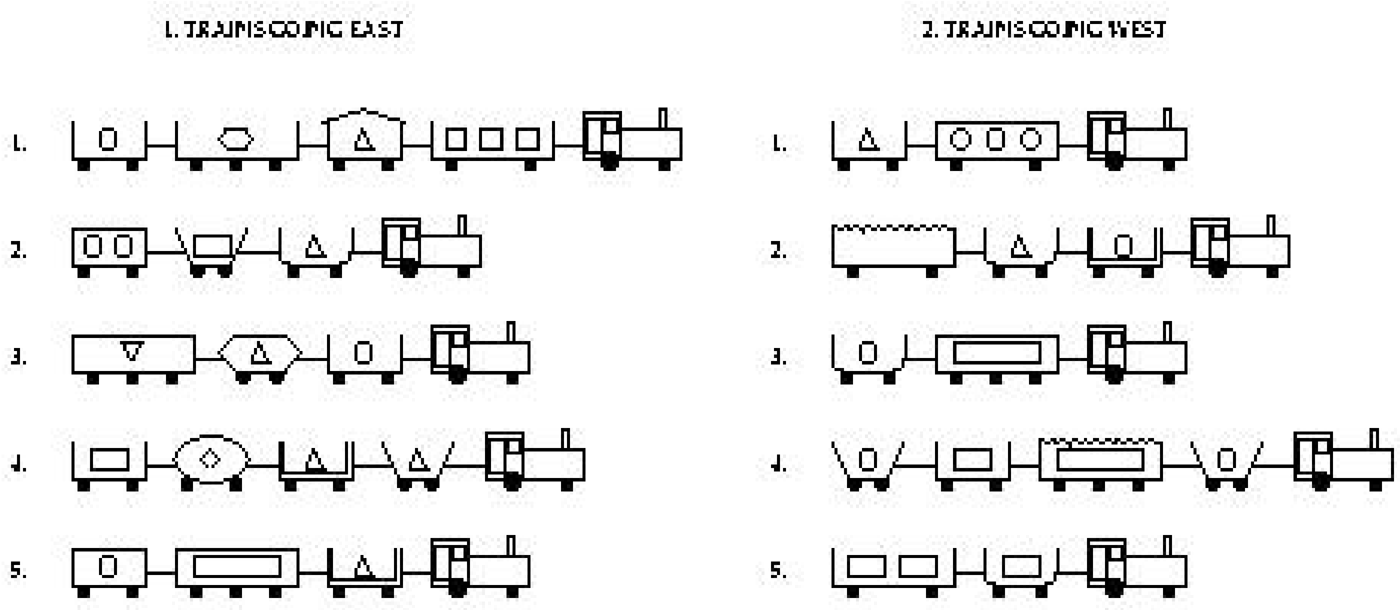
Tomáš Horváth

The presentation

- Inductive Logic Programming (ILP)
  - (Multi) Relational Data Mining method
  - Machine Learning + Logic Programming
  - complex data structures
    - medicine, genetics, chemistry, economic ...
- Goals
  - give a basic overview on ILP

East-West trains

- what makes a train to go eastward?



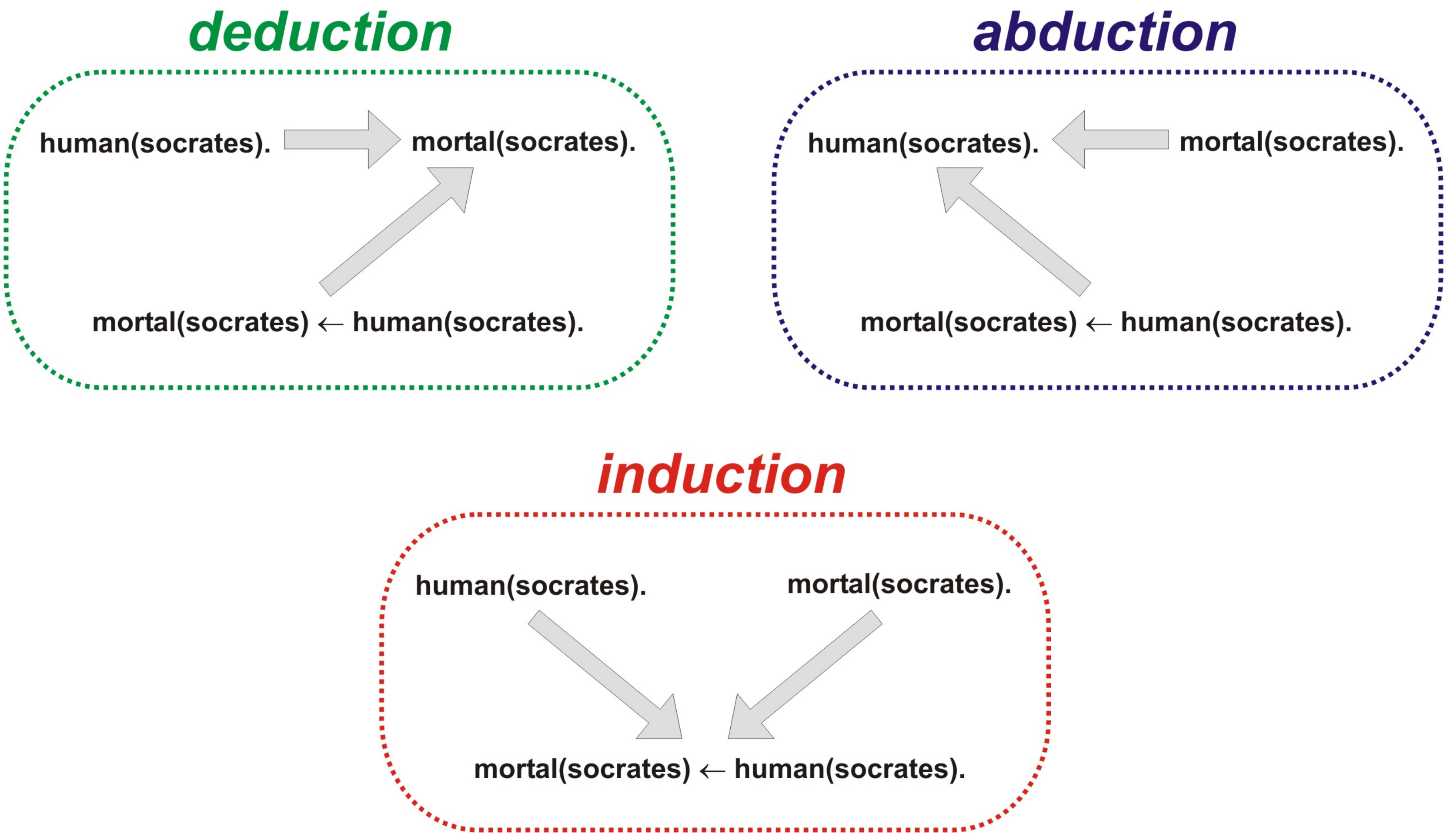
Outlines

- Basic concepts
- ILP techniques
  - refinement graphs (FOIL)
  - inverse resolution (CIGOL)
  - relative least generalization (GOLEM)
  - inverse entailment (ALEPH)

Several forms of reasoning

- (Background) Knowledge
  - Socrates is a human
- Observations (Examples)
  - Socrates is mortal
- Theory (Hypothesis)
  - IF X is a human THEN X is mortal

Several forms of reasoning



Several forms of reasoning

- Deduction (Abduction)
  - if the theory and background knowledge (examples) are true then the examples (background knowledge) are also true.
- Induction
  - an induced theory from given examples and background knowledge need not be true in case of other examples or background knowledge not used in the induction process

General ILP task

- Given
  - Background Knowledge ***B***
  - Examples ***E***
    - Positive e^+
    - Negative e^- (sometimes not used in the learning process)
- Find
  - Hypothesis ***H***, such that
    - covers all positive examples (*completeness*)
    - covers non of the negative examples (*consistency*)
    - a complete and consistent hypothesis is *correct*

Normal setting (predictive)

- Representations
  - example *e* – definite clause (fact)
  - background knowledge *B* – definite program
  - hypothesis *H* – definite program

- H *covers* e w.r.t. B if
  - $(H \cup B) \models e$

Normal setting (predictive)

- Positive examples
  - { daughter(mary,ann), daughter(eve,tom) }
- Negative examples
  - { daughter(tom,ann), daughter(eve,ann) }
- Background Knowledge
  - { mother(ann,mary), mother(ann,tom), father(tom,eve), father(tom,ian), female(ann), female(mary), female(eve), male(ian), male(tom), parent(X,Y)←mother(X,Y), parent(X,Y) ← father(X,Y) }
- Hypotheses
  - { daughter(X,Y) ← female(X), parent(Y,X) }
  - { daughter(X,Y)←female(X), mother(Y,X); daughter(X,Y)←female(X), father(Y,X) }

Non-monotonic setting (descriptive)

- Representations
  - example *e* – <u>Herbrand interpretation</u>
    - often just positive examples
  - background knowledge *B* – definite program
  - hypothesis *H* – definite program

- H *covers* e w.r.t. B if
  - *H is true in the least Herbrand model M(B∪E)*

Non-monotonic setting (descriptive)

- Examples
  - { mother(lieve,soetkin), father(luc,soetkin), parent(lieve,soetkin), parent(luc,soetkin), male(luc), female(lieve), female(soetkin), human(lieve), human(luc), human(soetkin) }
  - { mother(blaguna,sonja), father(veljo,saso), father(veljo,sonja), parent(blaguna,saso), parent(blaguna,sonja), parent(veljo,saso), parent(veljo,sonja), male(veljo), male(saso), female(blaguna), female(sonja), human(veljo), human(saso), human(blaguna), human(sonja) }
- Empty background knowledge
- Hypothesis
  - { parent(X,Y)←mother(X,Y); parent(X,Y)←father(X,Y); mother(X,Y)∨father(X,Y)←parent(X,Y); ←mother(X,Y),father(X,Y); human(X)←female(X); human(X)←male(X); female(X)∨male(X)←human(X); ←female(X),male(X); female(X)←mother(X,Y); male(X)←father(X,Y); human(X)←parent(X,Y); human(Y)←parent(X,Y); ←parent(X,X) }

Non-monotonic setting (descriptive)

- Examples
  - { class(fix), worn(gear), worn(chain) }
  - { class(sendback), worn(engine), worn(chain) }
  - { class(sendback) ,worn(wheel) }
  - { class(ok) }
- Background knowledge
  - { replaceable(gear), replaceable(chain), not_replaceable(engine), not_replaceable(wheel) }
- Hypothesis
  - { class(sendback)$\leftarrow$worn(X), not_replaceable(X) }

Non-monotonic setting (descriptive)

- Positive examples
  - { daughter(mary,ann), daughter(eve,tom) }
- Negative examples
  - { daughter(tom,ann), daughter(eve,ann) }
- Background Knowledge
  - { mother(ann,mary), mother(ann,tom), father(tom,eve), father(tom,ian), female(ann), female(mary), female(eve), male(ian), male(tom), parent(X,Y)←mother(X,Y), parent(X,Y) ← father(X,Y) }
- Hypotheses
  - { daughter(X,Y) ← female(X), parent(Y,X) }
  - { ←daughter(X,Y), mother(X,Y); female(X)←daughter(X,Y); mother(X,Y)∨father(X,Y)←parent(X,Y) }

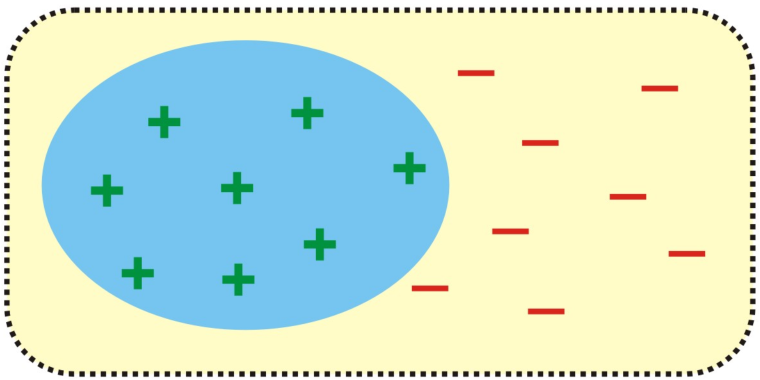
Predictive vs. Descriptive ILP

- Predictive
  - Learn a reason why positives are positives and negatives are negatives
  - You know what You are looking for, but you don't know what it looks like.
  - Separate examples and background knowledge
  - often used

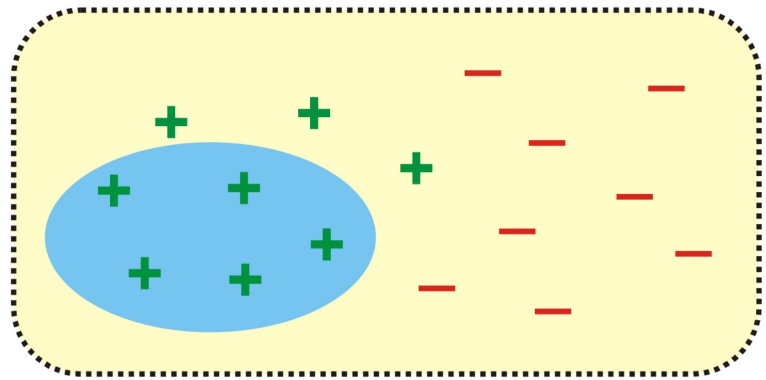
- Descriptive
  - Find something interesting about the data
  - You don't know what You are looking for
  - all background knowledge about an example is incorporated in this example

Completeness and Consistency

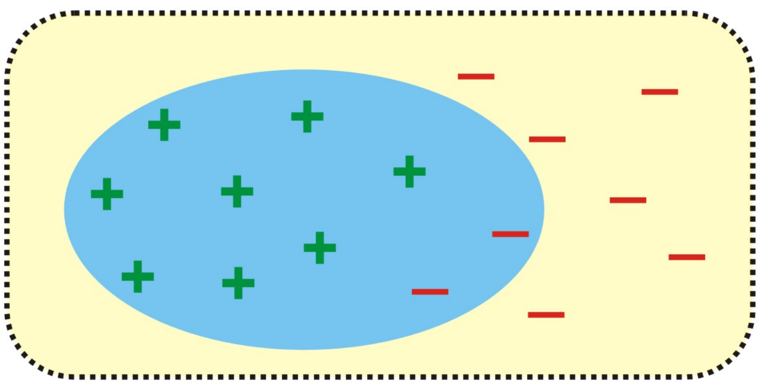
complete, consistent (CORRECT)



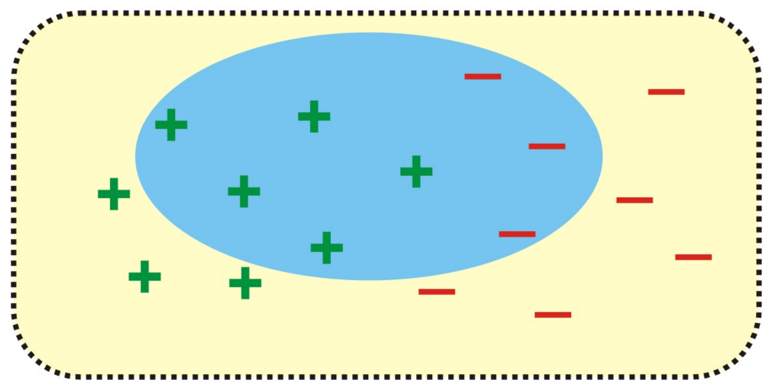
not complete, consistent (OFTEN)



complete, not consistent (WRONG)



not complete, not consistent (WRONG)



Specialisation vs. Generalization

- $C \models D$
  - D is a specialisation of C
  - C is a generalization of D

- if $C \not\models e$ then $D \not\models e$
- if $D \models e$ then $C \models e$

The general ILP algorithm

- Input: E^+, E^-, B
- Output: H

- begin
  - initialize H
  - repeat
    - if H is not consistent specialize it
    - if H is not complete generalize it
  - until H is not correct
  - output H
- end

Subsumption Theorem

- cover relation "*|=*"
  - hard to implement
  - not decidable
  - need a framework to solve this problem

- subsumption
  - a clause C subsumes a clause D (*C≥D*) if *(∃θ) Cθ ⊆ D*
    - C=p(X)←q(a),r(Y) = {p(X),¬q(a),¬r(Y)} ≥ {p(b),¬q(a),¬r(c),¬s(Z)} = p(b)←q(a),r(c),s(Z)=D for θ={ X/b, Y/c}
  - if C ≥ D then C |= D (the converse does not hold)
    - C=P(f(X))←P(X), D=P(f^2(X))←P(X)

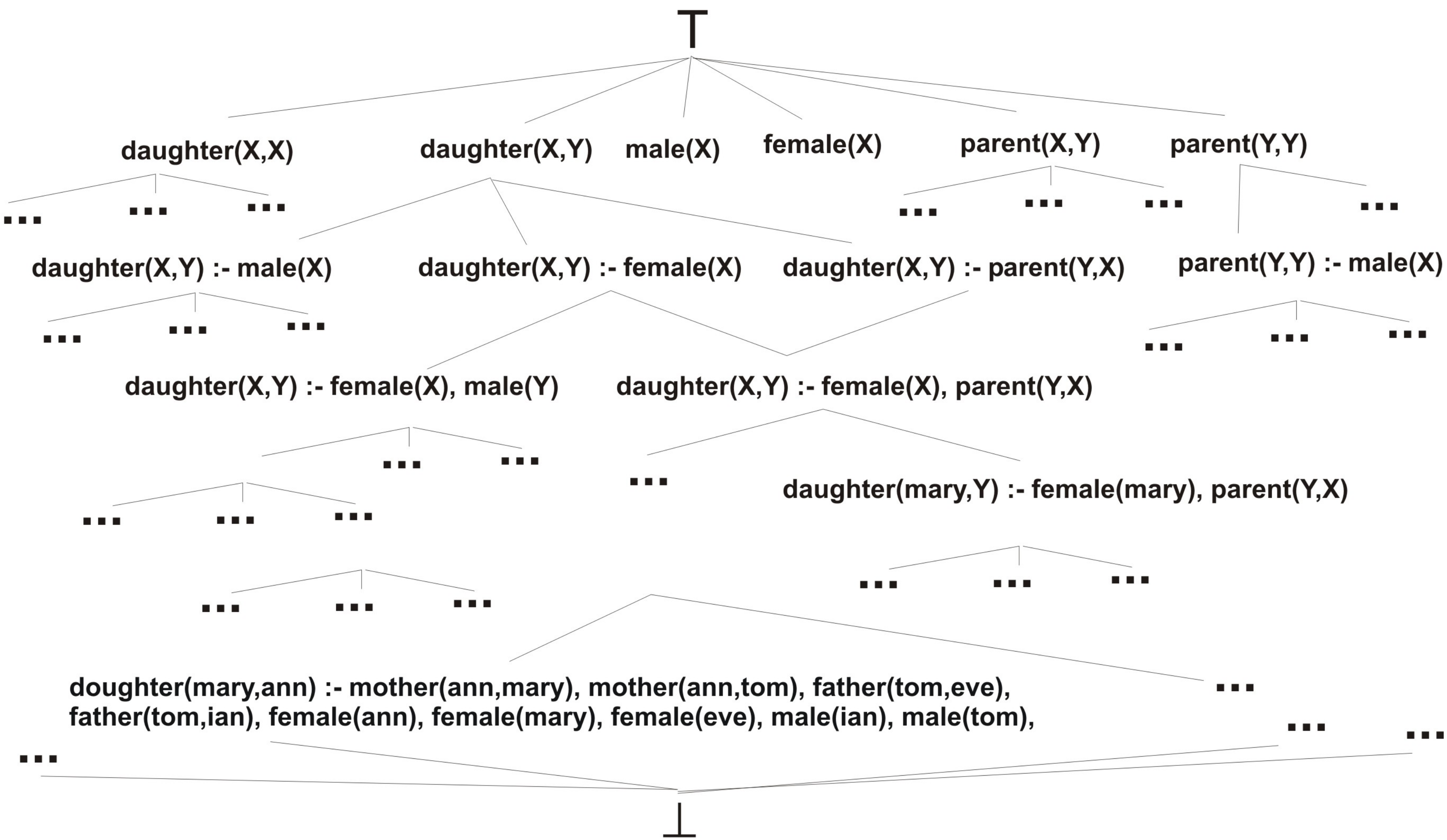
Subsumption Theorem

- SLD-refutation theorem
  - Let Σ is a set of Horn clauses. Then Σ is unsatisfiable iff $\Sigma \mid\!-_{sr} \Box$.
- SLD-Subsumption theorem
  - Let Σ is a set of Horn clauses and C a Horn clause. Then $\Sigma \models C$ iff $\Sigma \mid\!-_{sd} C$.
- SLD-refutation theorem and SLD-Subsumption theorem are equivalent.
- $\Sigma \mid\!-_{sr} C$ if there exists an SLD-resolution of C from Σ.
- $\Sigma \mid\!-_{sd} C$ if there exists an SLD-resolution of a clause D from Σ such that $D \geq C$ (D subsumes C)

Hypothesis space

- Space of (all) Horn clauses H
  - ordered by subsumption
- for every finite set S⊆H there exists a greatest specialisation of S in H
- for every finite set S⊆H there exists a least generalisation of S in H
- H ordered by ≥ is a lattice
  - ⊥ - bottom element
  - T – top element

Hypothesis space



Hypothesis space

- Large space of all hypotheses
  - need for a space of acceptable hypotheses
    - language bias
- Refinement operator ρ:H$\rightarrow$H
  - determine the hypothesis space (refinement graph)
  - specialisation operator
    - $\rho(C)=D,\ C \models D$
      - applies a substitution θ to C
      - adds literal to the body of C
  - generalisation operator
    - $\rho(C)=D,\ D \models C$
      - applies an inverse substitution θ^{-1} to C
      - removes literal from the body of C

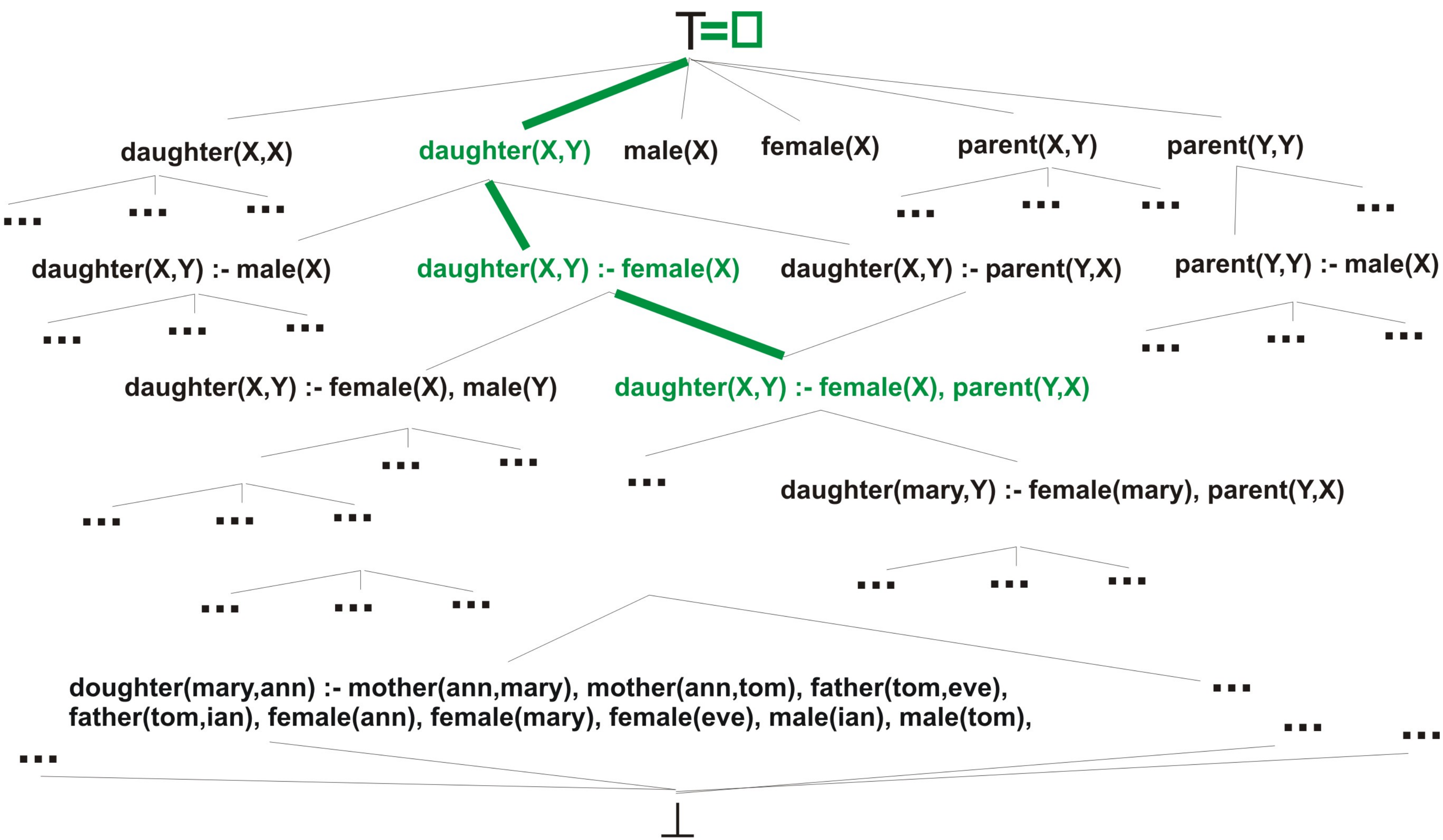
Outlines

- Basic concepts
- ILP techniques
  - refinement graphs (FOIL)
  - inverse resolution (CIGOL)
  - relative least generalization (GOLEM)
  - inverse entailment (ALEPH)
- Applications
- Future directions of ILP

Searching refinement graphs

- top-down searching of refinement graph
- starting with T= □
- depth-first search
- implemented in system FOIL

Searching refinement graphs



Inverse resolution

- bottom-up approach
- applying inverse resolution to clauses
  - V-operators
    - absorption
    - identification
  - W-operators
    - intra-construction
    - inter-construction
- predicate invention
- not deterministic
- implemented in system CIGOL

Inverse resolution

absorption

$q \leftarrow A$ $\quad$ $p \leftarrow q,B$

$p \leftarrow A,B$

intra-construction

$q \leftarrow B$ $\quad$ $p \leftarrow A,q$ $\quad$ $q \leftarrow C$

$p \leftarrow A,B$ $\quad$ $p \leftarrow A,C$

identification

$p \leftarrow A,q$ $\quad$ $q \leftarrow B$

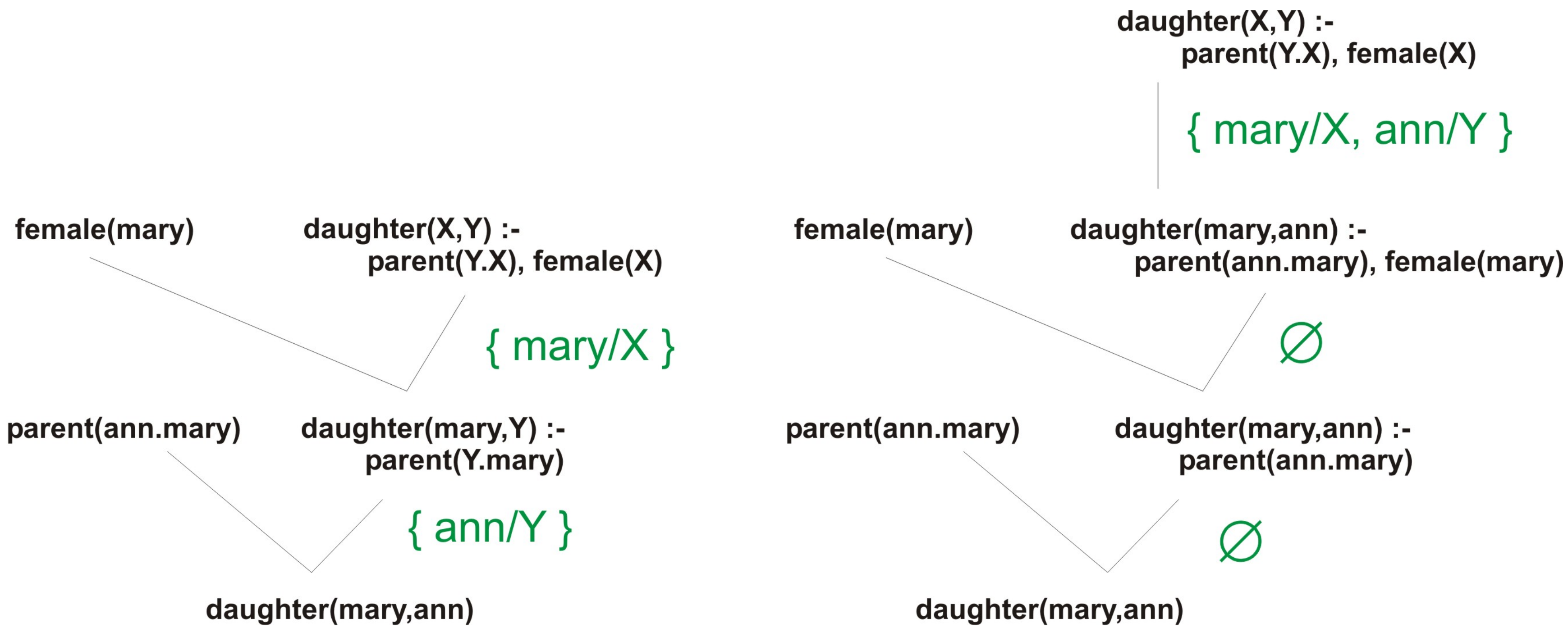
$p \leftarrow A,B$

inter-construction

$p \leftarrow r,B$ $\quad$ $r \leftarrow A$ $\quad$ $q \leftarrow r,C$

$p \leftarrow A,B$ $\quad$ $q \leftarrow A,C$

Inverse resolution



Relative least generalization

- $H \cup B \models e$
  - Let H consist of single clause C
  - $C \cup B \models e \Rightarrow C \models B \rightarrow e$
  - if e – atom, B – atoms then $e \leftarrow B$ is a Horn clause

- $C \geq_B D$ if $C \geq (D \cup \{\neg L_1, \ldots, \neg L_n\})$
- $LGS((D_1 \cup \{\neg L_1, \ldots, \neg L_n\}), \ldots, (D_m \cup \{\neg L_1, \ldots, \neg L_n\})$ is an $RLGS_B$ of $\{D_1, \ldots, D_m\}$ relative to $B=\{L_1, \ldots, L_n\}$ in H

- bottom-up approach
  - searches correct $LGRS_B$ of positive examples
- implemented in system GOLEM

Relative least generalization

- $RLGS_B$(daughter(mary,ann),daughter(eve,tom)) for B={female(mary), parent(ann,mary), female(eve), parent(tom,eve), female(ann)} is
- daughter($V_{m,e}$,$V_{a,t}$) $\leftarrow$ parent(ann,mary), parent(tom,eve), female(mary), female(eve), female(ann),parent($V_{a,t}$,$V_{m,e}$), female($V_{m,e}$), female($V_{m,a}$), female($V_{a,e}$).
  - if C\{L} covers at least as many positive examples and at most as many negative examples as C then the literal L is irrelevant
- after removing irrelevant literals we get daughter($V_{m,e}$,$V_{a,t}$) $\leftarrow$ parent($V_{a,t}$,$V_{m,e}$), female($V_{m,e}$), so daughter(X,Y) $\leftarrow$ parent(Y,X), female(X)

Inverse entailment

- $H \cup B \models e$
  - Let H consist of single clause C
  - $C \cup B \models e \Rightarrow B \cup \neg e \models \neg C$
  - $\neg\bot$ is a (possibly infinite) conjunction of ground literals which are true in every model of $B \cup \neg e$
  - $B \cup \neg e \models \neg\bot$
  - $\neg C$ is true in all models of $B \cup \neg e \Rightarrow \neg C$ contains a subset of $\neg\bot$
  - $B \cup \neg e \models \neg\bot \models \neg C \Rightarrow C \models \bot$

- top-down approach
  - searches for clauses which subsumes $\bot$
  - ability to have rules in background knowledge
- implemented in system ALEPH
  - language declarations

Inverse entailment

- $\bot_{daughter(mary,ann)}$ = daughter(A, B) :- mother(B, A), female(B), female(A), parent(B, A).
- $\bot_{daughter(eve,tom)}$ = daughter(A, B) :- father(B, A), female(A), male(B), parent(B, A).

- H = { daughter(A, B) :- female(A), parent(B, A). }

References

- http://www.cs.bris.ac.uk/~ILPnet2/
- Shan-Hwei Nienhuys-Cheng, Ronald de Wolf: *Foundations of Inductive Logic Programming.* Springer-Verlag, 1997, ISBN 3540629270.
- Nada Lavrač, and Sašo Džeroski. *Inductive Logic Programming: Techniques and Applications.* Ellis Horwood, New York, 1994.
- Proceedings of the Conference on Inductive Logic Programming (ILP), since 1990.