707.009
Foundations of Knowledge Management
„Inductive Concept Learning and ILP“

How can we uncover conceptual relations from examples / background knowledge?

Markus Strohmaier

Univ. Ass. / Assistant Professor
Knowledge Management Institute
Graz University of Technology, Austria

e-mail: markus.strohmaier@tugraz.at
web: http://www.kmi.tugraz.at/staff/markus

Acknowledgements: Course slides in part based on

…the following resources:

- "Inductive Logic Programming: Theory and Methods"
  - Stephen Muggleton and Luc de Raedt
- "Inductive Logic Programming: Techniques and Applications"
  - Nada Lavrac and Saso Dzeroski
- Slides "Introduction to Inductive Logic Programming"
  - Nada Lavrac
- Slides "Inductive Logic Programming: Basic Approaches"
  - Chongbing Liu

https://wiki.ldv.ei.tum.de/tiki-index.php?page=Induktive+Logische+Programmierung

Overview

Today‘s Agenda:

Inductive Concept Learning & ILP

- Problem Definition
- Success Criteria
- Inductive Logic Programming
- Selected ILP Techniques

Wissensorganisation – Zwei Herangehensweise

Formale vs. inhaltliche Struktur

Viele Informationen liegen in unstrukturierten Freitexten (ohne Struktur) vor. Aussagekräftig aber schlecht auswertbar

Zwei Herangehensweisen:

- Verwendung einer standardisierten Sprache a **priori** (stark formalisiert)
- Interpretation der heterogenen Sprache a **posteriori** (NLP, …)

Taxonomien, Ontologien, Semantische Netze

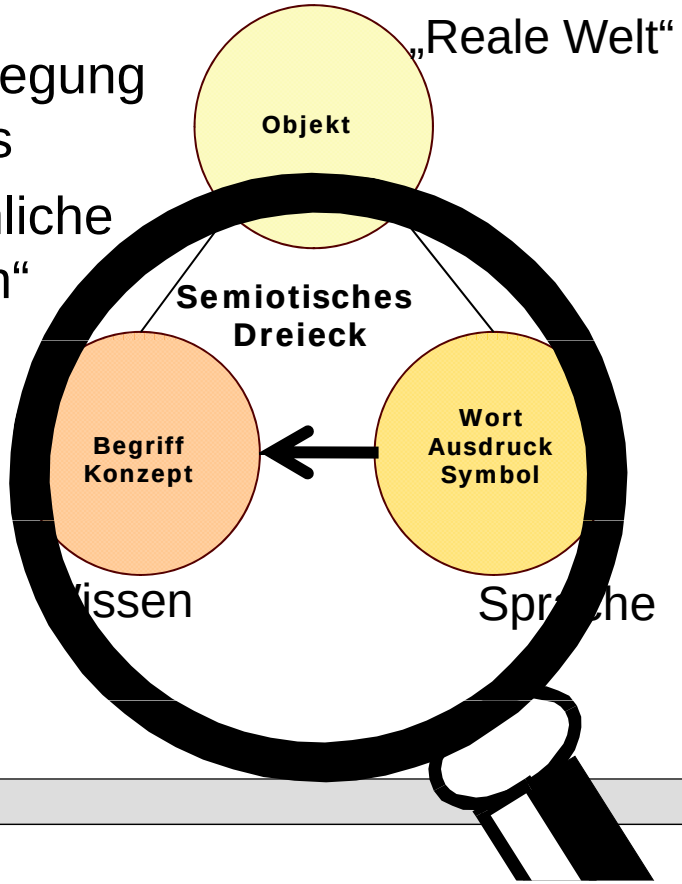
Schlüsselwort-extraktion, Folksonomies

Was sind Konzeptsysteme?

Konzeptsysteme sind Systeme von unterscheidbaren *Konzepten*, die mittels *Relationen* in Beziehung zueinander gesetzt werden und in einer natürlicheren *Sprache* formuliert werden können

Zielsetzung: Entwicklung und Festlegung eines gemeinsamen Verständnisses

Repräsentationssysteme: menschliche Sprache, Logik, „Computersprachen“



Inductive Concept Learning

$daughter(X, Y) \leftarrow$ **?**

How would you define the daughter(X,Y) relationship based on 1) Background K and 2) Training examples?

Training examples	
$daughter(mary, ann).$	$\oplus$
$daughter(eve, tom).$	$\oplus$
$daughter(tom, ann).$	$\ominus$
$daughter(eve, ann).$	$\ominus$

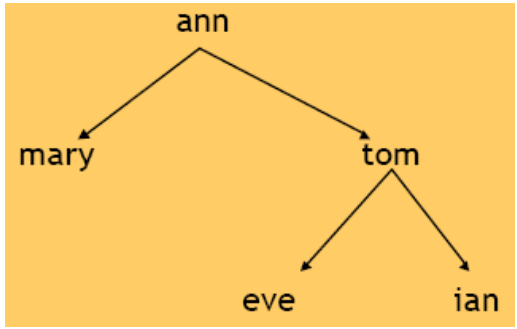
Background knowledge	
$parent(ann, mary).$	$female(ann).$
$parent(ann, tom).$	$female(mary).$
$parent(tom, eve).$	$female(eve).$
$parent(tom, ian).$	

$$c = daughter(X, Y) \leftarrow female(X), parent(Y, X)$$

Inductive Concept Learning

- E+ = {daughter(mary,ann),daughter(eve,tom)}
  E- = {daughter(tom,ann),daughter(eve,ann)}

- B = {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)}



The Problem of Inductive Concept Learning

U is a universal set of objects

C is a concept: a subset of objects in U $\mathcal{C} \subseteq \mathcal{U}$

To learn a concept C means to learn to recognize objects in C, to be able to tell whether

$x \in \mathcal{C}$ for each $x \in \mathcal{U}$

Example:

U may be the set of all patients in a register, and $\mathcal{C} \subseteq \mathcal{U}$

The set of all patients having a particular disease

Describing objects and concepts

Objects are described in *object description languages*

Concepts are described in *concept description languages*

Describing Objects:

In attribute-value object description languages, objects are described as attribute-value pairs.

e.g. card(diamonds, 7)

card is a predicate, values are the constants diamonds and 7

Describing objects and concepts

What are the implications of describing concepts extensionally / intensionally?

Describing Concepts:

- Extensionally:
  - Listing the descriptions of all of its instances
  - Example: *Pair* in poker can be extensionally described by the set of all pairs of cards which have the same rank

- Intensionally:
  - In a separate concept description language which allows for more compact and concise concept descriptions, e.g. in the form of rules

$$pair \quad \textbf{if} \quad [Rank_1 = 7] \wedge [Rank_2 = 7] \vee [Rank_1 = 8] \wedge [Rank_2 = 8] \vee \ldots [Rank_1 = a] \wedge [Rank_2 = a]$$

$$pair \quad \textbf{if} \quad Rank_1 = Rank_2$$

  - Horn clause: $pair(Suit_1, Rank_1, Suit_2, Rank_2) \leftarrow Rank_1 = Rank_2$

Inductive Concept Learning

Is a procedure for determining whether a given object belongs to a given concept i.e. whether the description of the object satisfies the description of the concept.

Definitions:

- A **Fact** denotes an object description
- A **Hypothesis** denotes an intensional concept description
- An **example** *e* for learning a concept *C* is a labeled *fact*
- A label could be either ⊕ (pos. Examples) or ⊖ (neg. examples)
- ε^+ denotes the set of positive examples
- ε^- denotes the set of negative examples

Inductive Concept Learning

Inductive concept learning Given a set $\mathcal{E}$ of positive and negative examples of a concept $\mathcal{C}$, find a hypothesis $\mathcal{H}$, expressed in a given concept description language $\mathcal{L}$, such that:

- every positive example $e \in \mathcal{E}^+$ is covered by $\mathcal{H}$,
- no negative example $e \in \mathcal{E}^-$ is covered by $\mathcal{H}$.

To test the coverage, the function

$$covers(\mathcal{H}, e)$$

Returns the value true if e is covered by H, and false otherwise.

Example

Is

$$pair(card(diamonds, 7), card(hearts, 7))$$

covered by

$$pair(card(Suit_1, Rank_1), card(Suit_2, Rank_2)) \leftarrow Rank_1 = Rank_2$$

?

Completeness and consistency of a hypothesis

A hypothesis H is **complete** with respect to the examples E if H covers all the positive examples.

$$\text{if } covers(\mathcal{H}, \mathcal{E}^+) = \mathcal{E}^+$$

A hypothesis H is **consistent** with respect to examples E if it covers none of the negative examples.

$$\text{if } covers(\mathcal{H}, \mathcal{E}^-) = \emptyset$$

Completeness and consistency of a hypothesis

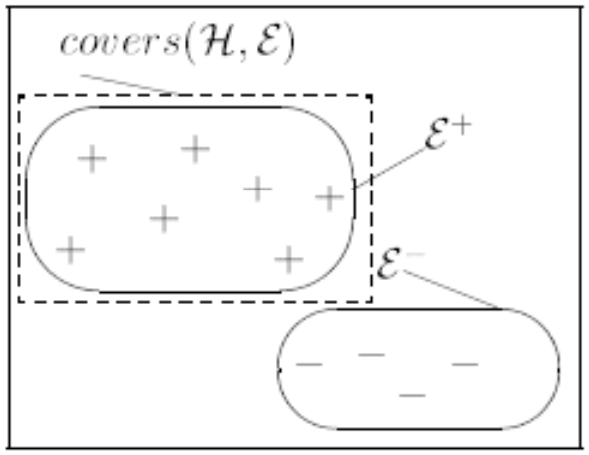
Complete:

if $covers(\mathcal{H}, \mathcal{E}^+) = \mathcal{E}^+$

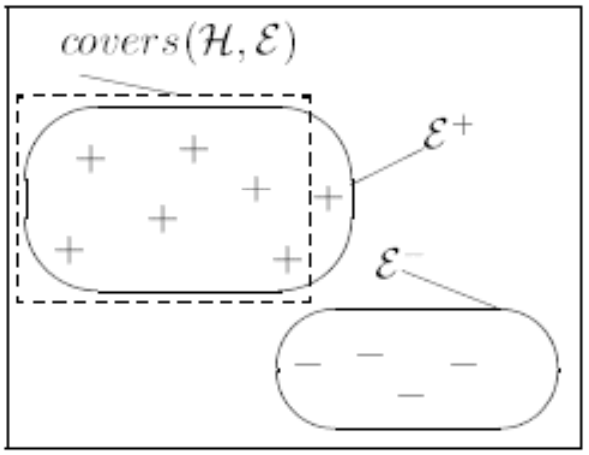
Consistent:

if $covers(\mathcal{H}, \mathcal{E}^-) = \emptyset$

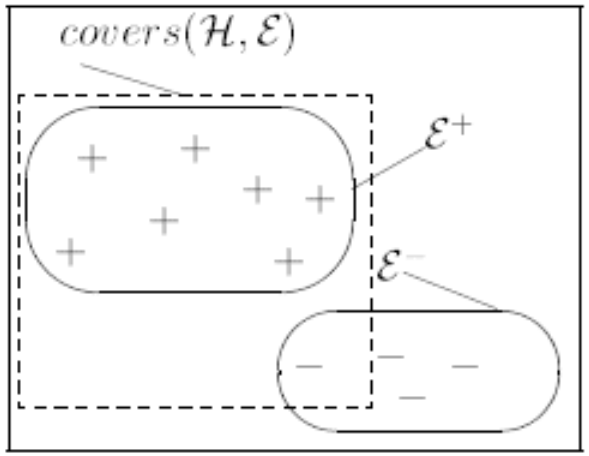
$\mathcal{H}$: complete, consistent



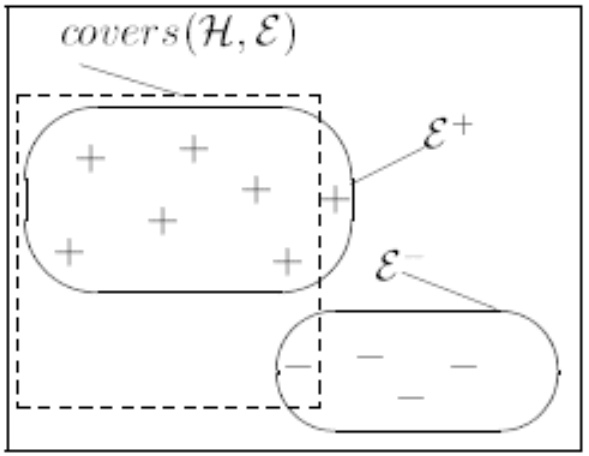
$\mathcal{H}$: incomplete, consistent



$\mathcal{H}$: complete, inconsistent



$\mathcal{H}$: incomplete, inconsistent



Other criteria for success

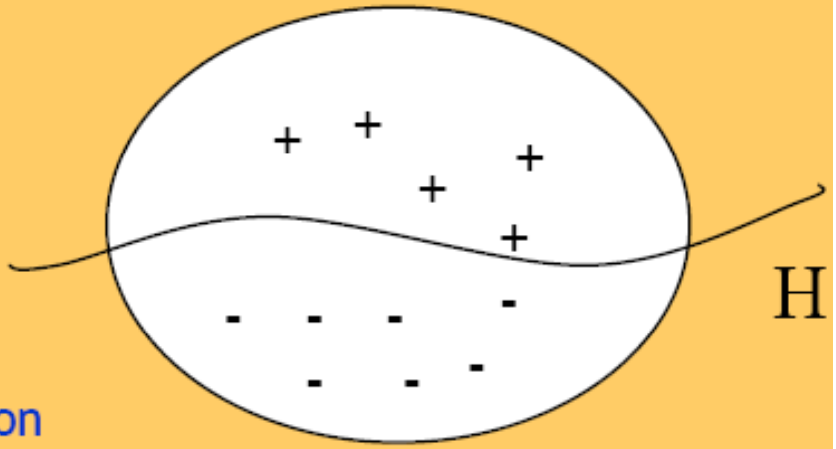
- Classification accuracy of H:
  The percentage of objects correctly classified by the hypothesis

- Transparency of H:
  the extent to which a hypothesis is understandable to humans

Predictive vs. Descriptive Induction

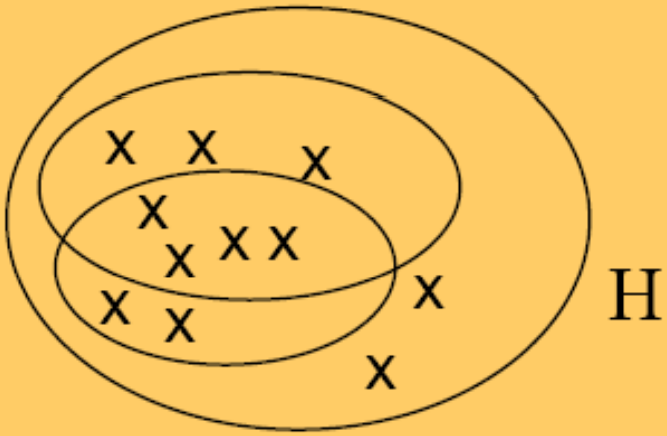
Predictive induction: Inducing classifiers, aimed at solving classification/prediction tasks

- Classification rule learning, Decision tree learning, ...
- Bayesian classifier, ANN, SVM, ..
- Data analysis through hypothesis generation and testing



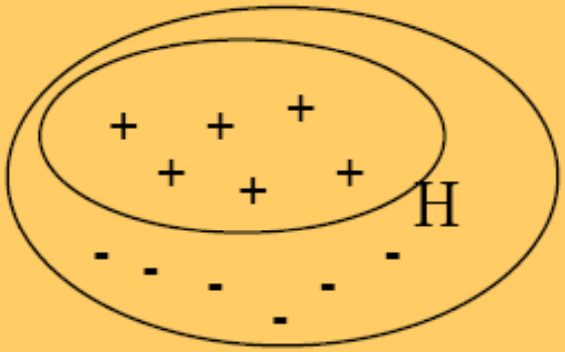
Descriptive induction: Discovering regularities, uncovering patterns, aimed at solving KDD tasks

- Symbolic clustering, Association rule learning, Subgroup discovery, ...
- Exploratory data analysis



Predictive ILP

- **Given:**
  - A set of observations
    - positive examples E^+
    - negative examples E^-
  - background knowledge B
  - hypothesis language L_H
  - covers relation



- **Find:**
  A hypothesis $H \in L_H$, such that (given B) H covers all positive and no negative examples

- In logic, **find** H such that
  - $\forall e \in E^+ : B \wedge H \models e$ (H is complete)
  - $\forall e \in E^- : B \wedge H \not\models e$ (H is consistent)

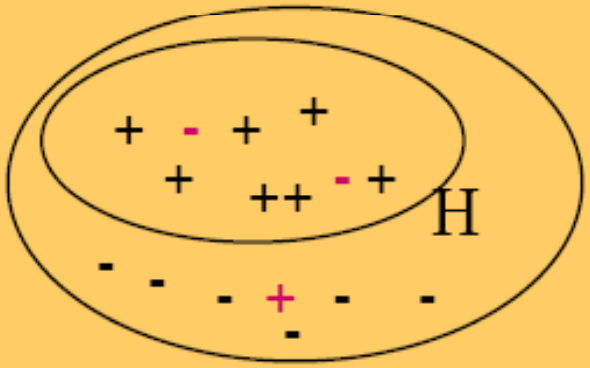
- In ILP, E are ground facts, B and H are (sets of) definite clauses

Predictive ILP

- **Given:**
  - A set of observations
    - positive examples E^+
    - negative examples E^-
  - background knowledge B
  - hypothesis language L_H
  - covers relation
  - quality criterion

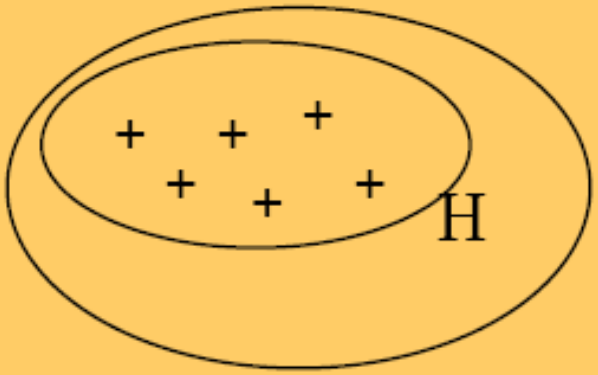
- **Find:**
  A hypothesis $H \in L_H$, such that (given B) H is optimal w.r.t. some quality criterion, e.g., max. predictive accuracy A(H)

(instead of finding a hypothesis $H \in L_H$, such that (given B) H covers all positive and no negative examples)



Descriptive ILP

- **Given:**
  - A set of observations (positive examples E^+)
  - background knowledge B
  - hypothesis language L_H
  - covers relation
- **Find:**
  Maximally specific hypothesis $H \in L_H$, such that (given B) H covers all positive examples
- In logic, **find** H such that $\forall c \in H$, c is true in some preferred model of $B \cup E$ (e.g., least Herbrand model $M(B \cup E)$)
- **In ILP, E are ground facts, B are (sets of) general clauses**



Empirical ILP

Given:

- a set of training examples $\mathcal{E}$, consisting of true $\mathcal{E}^+$ and false $\mathcal{E}^-$ ground facts of an unknown predicate p,
- a description language $\mathcal{L}$, specifying syntactic restrictions on the definition of predicate p, and
- background knowledge $\mathcal{B}$, defining predicates q_i (other than p) which may be used in the definition of p and which provide additional information about the arguments of the examples of predicate p.

Find:

- a definition $\mathcal{H}$ for p, expressed in $\mathcal{L}$, such that $\mathcal{H}$ is complete and consistent with respect to the examples $\mathcal{E}$ and background knowledge $\mathcal{B}$.

An Example ILP Problem

- The task is to define the target relation daugther (X,Y), which states that person X is a daughter of person Y, in terms of the **background knowledge** relations **female and parent**.

Answering: When is X the daugher of Y?

The Target Relation:

$$daughter(X, Y) \leftarrow female(X), parent(Y, X)$$

Training Examples:

Training examples		*Background knowledge*	
$daughter(mary, ann).$	$\oplus$	$parent(ann, mary).$	$female(ann).$
$daughter(eve, tom).$	$\oplus$	$parent(ann, tom).$	$female(mary).$
$daughter(tom, ann).$	$\ominus$	$parent(tom, eve).$	$female(eve).$
$daughter(eve, ann).$	$\ominus$	$parent(tom, ian).$	

Potential Target Predicate Relations:

$$daughter(X, Y) \leftarrow female(X), mother(Y, X)$$
$$daughter(X, Y) \leftarrow female(X), father(Y, X)$$

Different ILP Learners

- Single concept learners and multiple concepts learners
- Batch learners and incremental learners
- Non-Interactive learners and interactive learners

Empirical ILP Learners:

non-interactive single concept batch learners

An Simple Example

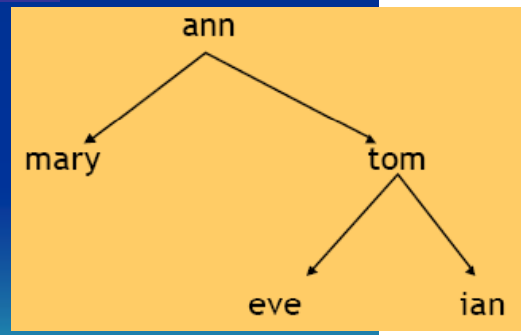
- Training Examples

$\mathcal{E}^+$ = { daughter(mary, ann).
daughter(eve, tom). }

$\mathcal{E}^-$ = { daughter(tom, ann).
daughter(eve, ann). }

- Background Knowledge

parent(ann, mary).
parent(ann, tom).
parent(tom, eve).
parent(tom, ian)
female(ann).
female(mary).
female(eve).



- Concept learned:

$daughter(X,Y) \leftarrow female(X), parent(Y,X)$

Sample problem
Knowledge discovery

- *E+ = {daughter(mary,ann),daughter(eve,tom)}*
  E- = {daughter(tom,ann),daughter(eve,ann)}
- *B = {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)}

• **Predictive ILP - Induce a definite clause**

daughter(X,Y) ← female(X), parent(Y,X).

or a set of definite clauses

daughter(X,Y) ← female(X), mother(Y,X).

daughter(X,Y) ← female(X), father(Y,X).

• **Descriptive ILP - Induce a set of (general) clauses**

← daughter(X,Y), mother(X,Y).

female(X) ← daughter(X,Y).

mother(X,Y); father(X,Y) ← parent(X,Y).

Generality

Generality: Let C and D be two clauses.
C is more general than D iff

$C \models D$, i.e., iff $cov(D) \subseteq cov(C)$

Note: Logical entailment (even between Horn clauses) is undecidable.

Subsumption

Substitution: $\theta = \{X_1/t_1,\ldots,X_n/t_n\}$ is an assignment of terms t_i to variables X_i

Subsumption: Let C and D be two clauses.
C subsumes D iff there is a θ, such that $C\theta \subseteq D$

Example: $p(a,b) \leftarrow r(b,a)$
is subsumed by ($\theta = \{X/a, Y/b\}$)

$p(X,Y) \leftarrow r(Y,X)$

subsumes

$p(X,Y) \leftarrow r(Y,X), q(X)$

Note: Subsumption is decidable (but NP complete)

Note: if C subsumes D, then C |= D. The reverse is not always true !

Substitution, Subsumption

Example 2.1 (Substitution, subsumption)
Let c be the clause:

$$c = daughter(X, Y) \leftarrow parent(Y, X)$$

A substitution $\theta = \{X/mary, Y/ann\}$ applied to clause c is obtained by applying θ to each of its literals:

$$c\theta = daughter(mary, ann) \leftarrow parent(ann, mary)$$

Clause c θ-subsumes the clause

$$c' = daughter(X, Y) \leftarrow female(X), parent(Y, X)$$

under the empty substitution $\theta = \emptyset$, since the set

$$\{daughter(X, Y), \overline{parent(Y, X)}\}$$

is a proper subset of $\{daughter(X, Y), \overline{female(X)}, \overline{parent(Y, X)}\}$.

X is EITHER the daughter of Y OR Y is not the parent of X

$daughter(X, Y) \leftarrow parent(Y, X)$

equiv. to

$\{daughter(X, Y), \overline{parent(Y, X)}\}$

Substitution, Subsumption

Let c be the clause:

$$c = daughter(X, Y) \leftarrow parent(Y, X)$$

Clause c θ-subsumes the clause

$$c' = daughter(X, X) \leftarrow female(X), parent(X, X$$

under the substitution $\theta = \{Y/X\}$.

Clause c θ-subsumes the clause

$$daughter(mary, ann) \leftarrow female(mary), parent(ann, mary), parent(ann, tom)$$

under the substitution $\theta = \{X/mary, Y/ann\}$.

Substitution, Subsumption

Two important properties:

If c Θ-subsumes c´ then c logically entails c´, c|=c´

The relation <= introduces a lattice on the set of reduced clauses

ILP as search of program clauses

- **Semantic generality**
  Hypothesis H_1 is semantically more general than H_2 w.r.t. background theory B if and only if $B \cup H_1 \models H_2$
- **Syntactic generality or θ-subsumption**
  (most popular in ILP)
  - Clause c_1 θ-subsumes c_2 ($c_1 \geq_\theta c_2$)
    if and only if $\exists\theta : c_1\theta \subseteq c_2$
  - Hypothesis $H_1 \geq \theta\, H_2$
    if and only if $\forall c_2 \in H_2$ exists $c_1 \in H_1$ such that $c_1 \geq \theta\, c_2$
- **Example**

  c1 = daughter(X,Y) ← parent(Y,X)
  c2 = daughter(mary,ann) ← female(mary),
  parent(ann,mary),
  parent(ann,tom).

  c1 θ-subsumes c_2 under θ = {X/mary,Y/ann}

ILP as search of program clauses

- **Properties**
  - soundness: if $c_1 \geq_\theta c_2$ then $c_1 \models c_2$
  - incompleteness: the opposite does not hold: (e.g., self-recursive clauses
    c_1 = p(f(X)) ← p(X) and c_2 = p(f(f(Y))) ← p(Y))
  - decidable (but NP-complete)
  - transitive and reflexive but not anti-symmetric
- **Lattice on reduced clauses**
  - equivalence classes [*c*]:
    c_1 = parent(X,Y) ← mother(X,Y)
    c_2 = parent(X,Y) ← mother(X,Y), mother(X,Z)
  - c_1 *reduced* clause of c_2 iff c_1 minimal subset of literals such that $c_1 \equiv_\theta c_2$
  - equivalence classes induce a lattice (Plotkin)
    - any two clauses have unique *lub* (the *lgg*)
    - any two clauses have *glb*

ILP as a search of program clauses

An ILP learner can be described by

- the **structure of the space of clauses**
- its **search strategy**

• uninformed search (depth-first, breadth-first, iterative deepening)

• heuristic search (best-first, hill-climbing, beam search)

- its **heuristics**
  - for directing search
  - for stopping search (quality criterion)

Learning

Two strategies for learning

- Top-down search of refinement graphs
- Bottom-up search
  - building least general generalizations
  - inverting resolution (CIGOL)
  - inverting entailment (PROGOL)

Refinement operator

$C \rightarrow ref(C)$

Minimal (most general) specialization:

D is subsumed by C, but there is no more general D' which is also subsumed by C.

- Apply a substitution θ to the clause
- Add a literal to the clause

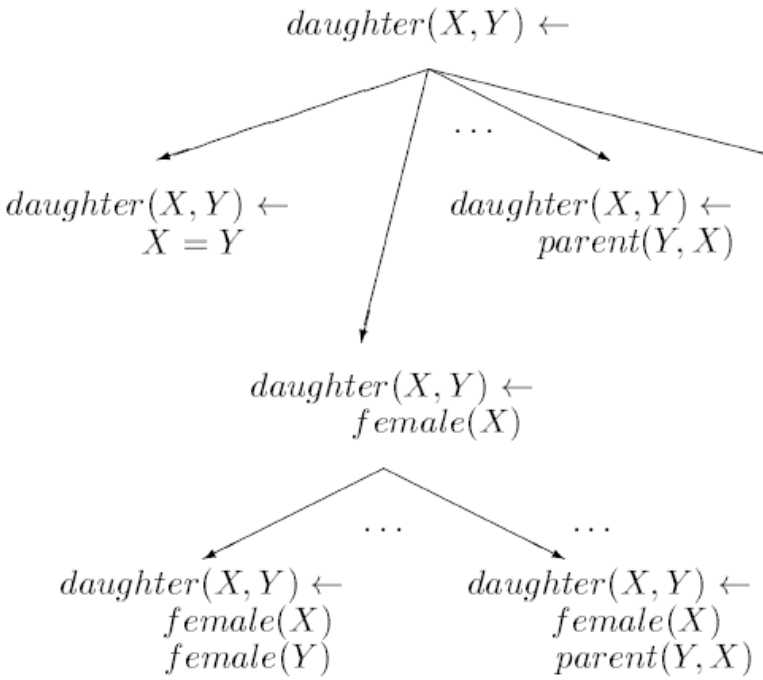
Refinement operator defines the specialization graph.

The hypothesis language has to be specified: predicates, functors, constants, (types of arguments).

Size of the search space increases with the number of existential variables in the body.

Top-Down Search of Refinement Graphs

Top down learners start from the most general clauses and repeatedly refine (specialize) them until they no longer cover negative examples.



The learner generates the set of refinements / clauses:

$$\rho(c) = \{daughter(X,Y) \leftarrow L\}$$

where L is one of following literals:[1]

- literals having as arguments the variables from the head of the clause: $X = Y$, $female(X)$, $female(Y)$, $parent(X,X)$, $parent(X,Y)$, $parent(Y,X)$, and $parent(Y,Y)$, and
- literals that introduce a new distinct variable Z ($Z \neq X$ and $Z \neq Y$) in the clause body: $parent(X,Z)$, $parent(Z,X)$, $parent(Y,Z)$, and $parent(Z,Y)$,

Structuring the Hypothesis Space

$daughter(X, Y) \leftarrow$ **?**

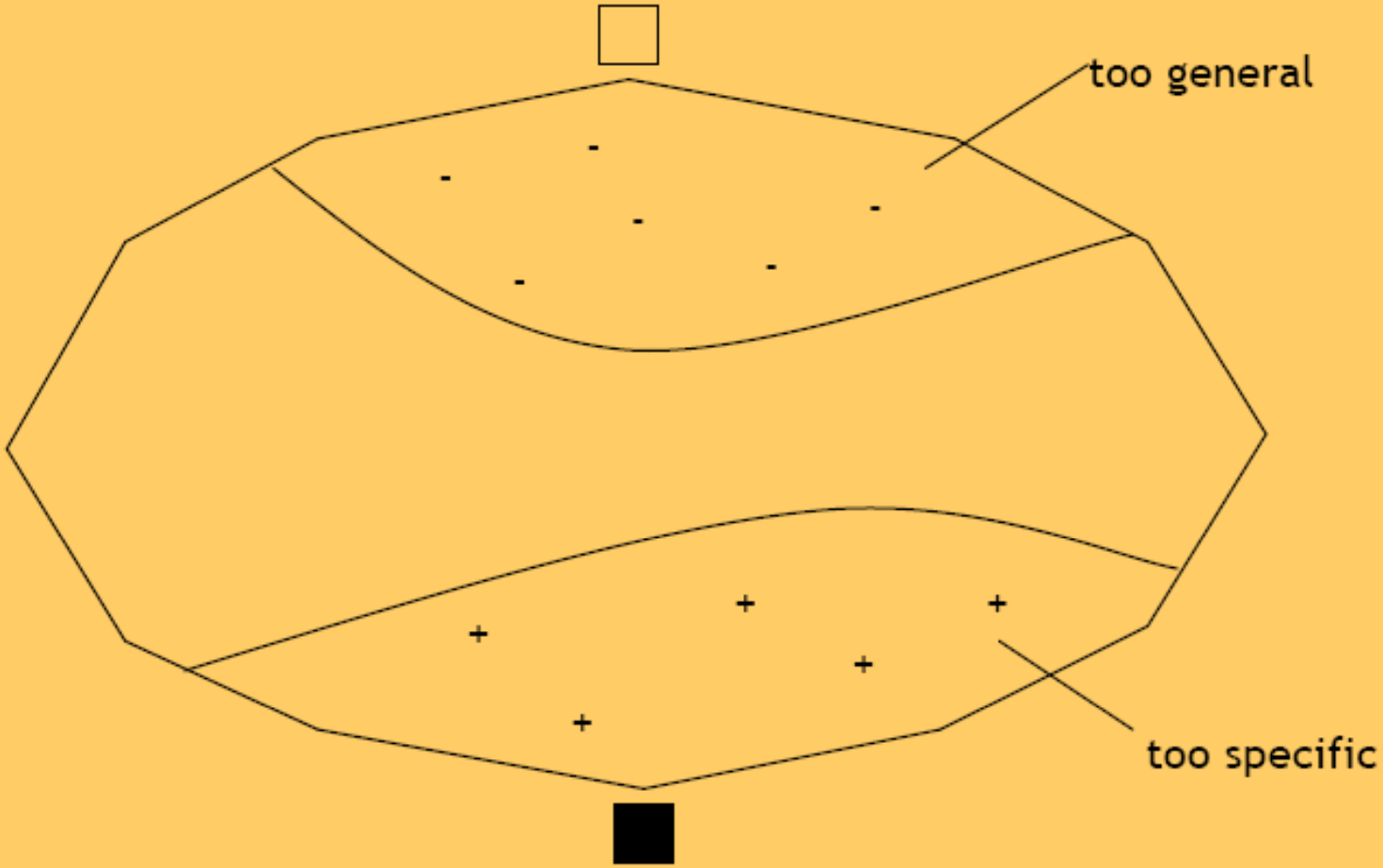
In ILP, the search space is determined by the language of logic programs $\mathcal{L}$ consisting of program clauses of the form $T \leftarrow Q$, where T is an atom $p(X_1, \ldots, X_n)$ and Q is a conjunction of literals $L_1, \ldots, L_m$. The vocabulary of predicate symbols q_i in the literals L_i of the body of a clause is determined by the predicates from the background knowledge $\mathcal{B}$. If $\mathcal{L}$ allows for recursive predicate definitions, the predicate symbol in L_i can also be the predicate symbol p itself.

ILP as search of program clauses

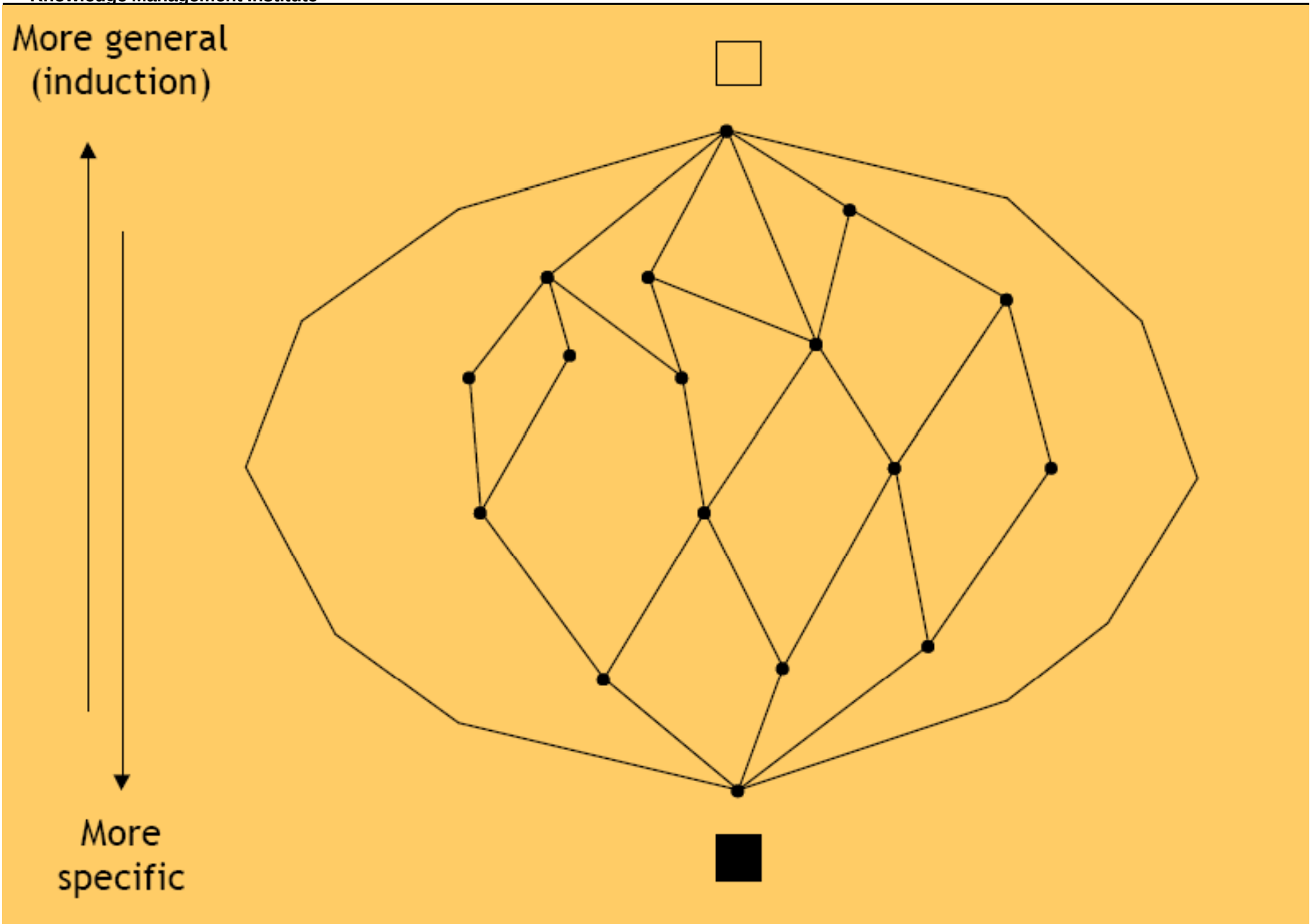
- **Structure of the hypothesis space**
  based on the generality relation
  - G is more general than S if and only if
    covers(S) $\subset$ covers(G)
- **Pruning the search space**
  - If H does not cover an example *e* then no specialization of *e* will cover *e*
    $\rightarrow$ used with positive examples for pruning
  - If H covers and example *e* then all generalizations of *e* will also cover *e*
    $\rightarrow$ used with negative examples for pruning
- **Version space**
  The above properties determine the space of possible solutions

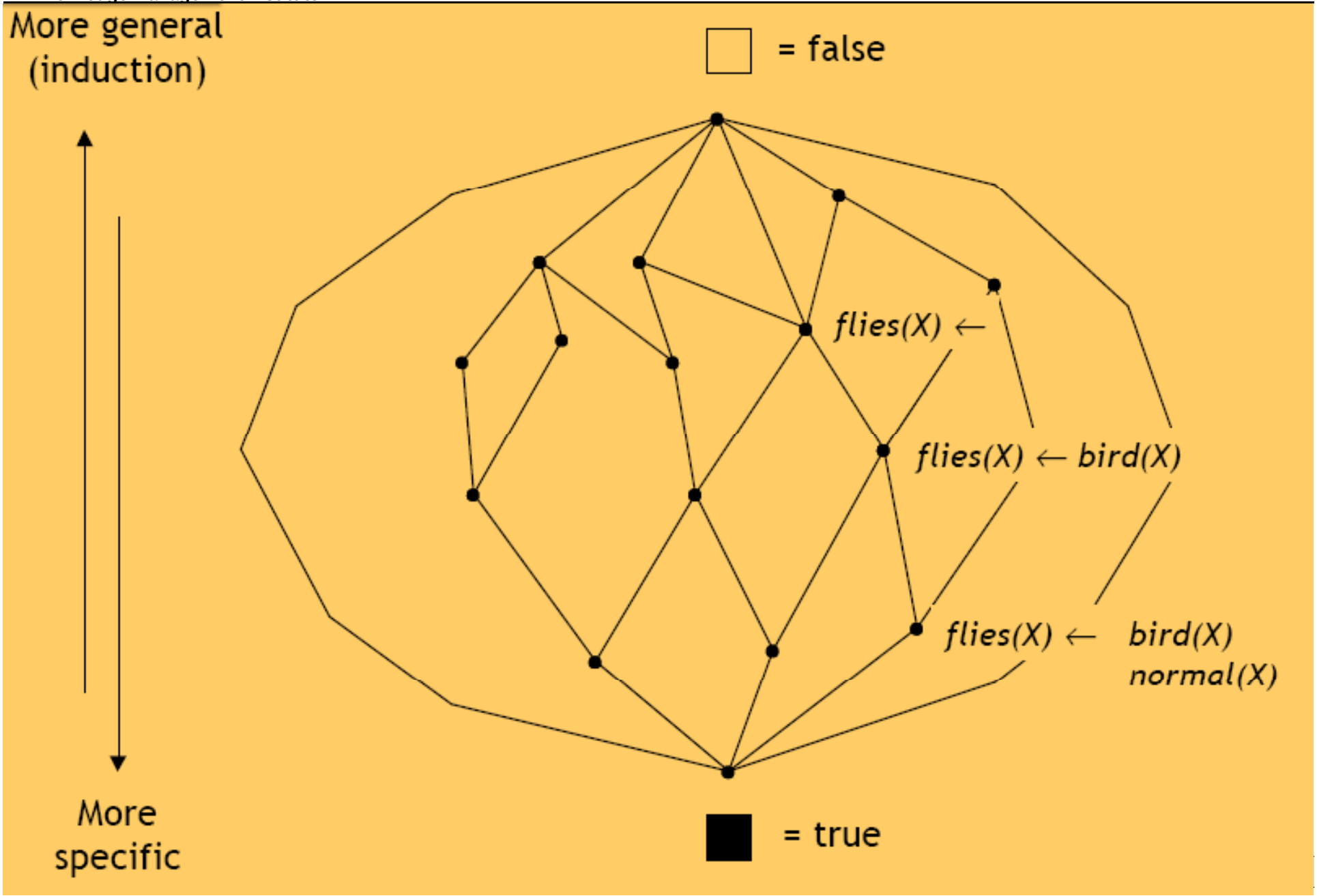
Hypothesis space

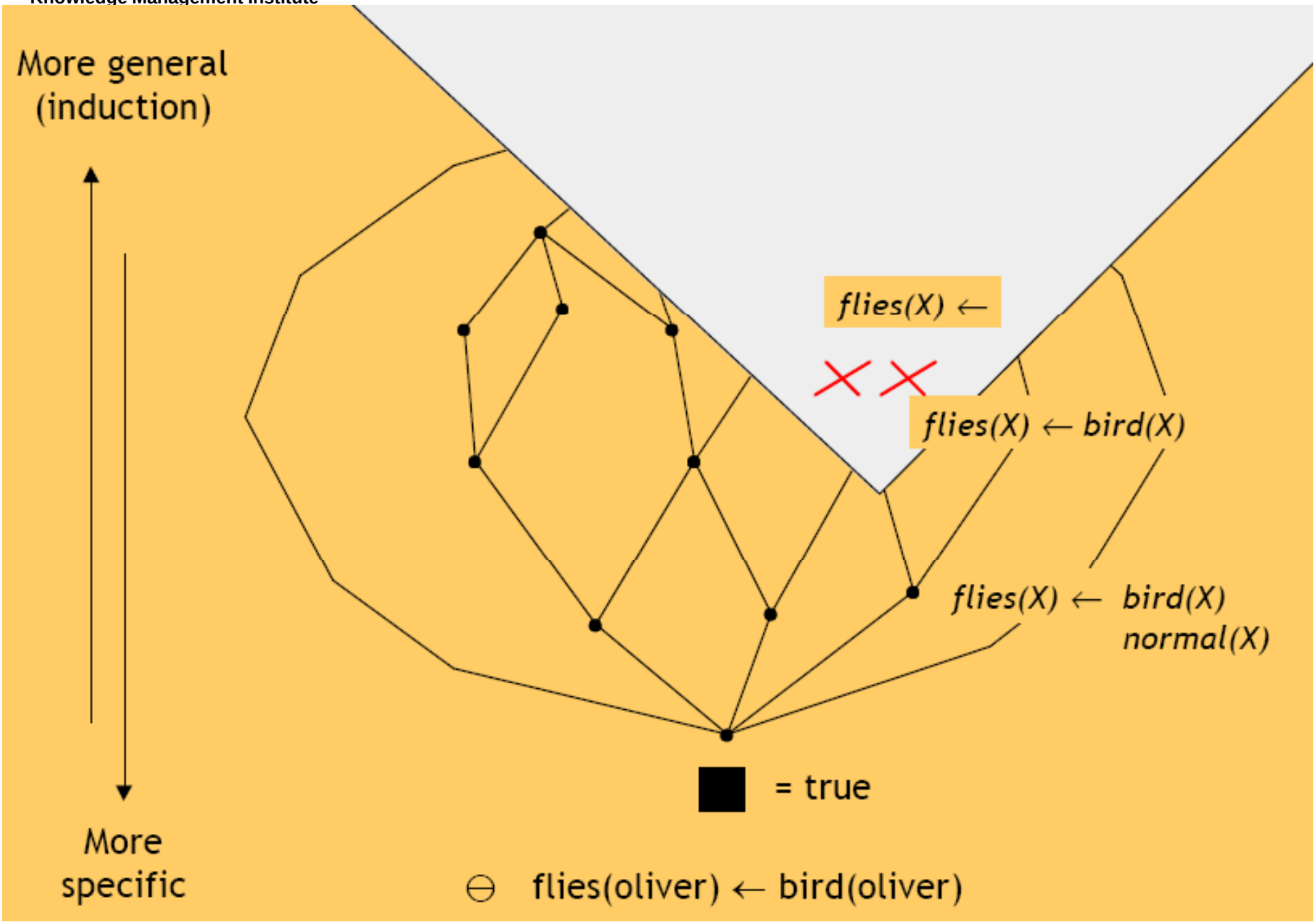
The Version Space model



How to structure the hypothesis space?
How to move from one hypothesis to another?


More general
(induction)
More
specific


More general
(induction)
= false
flies(X) ←
flies(X) ← bird(X)
flies(X) ← bird(X)
normal(X)
More
specific
= true

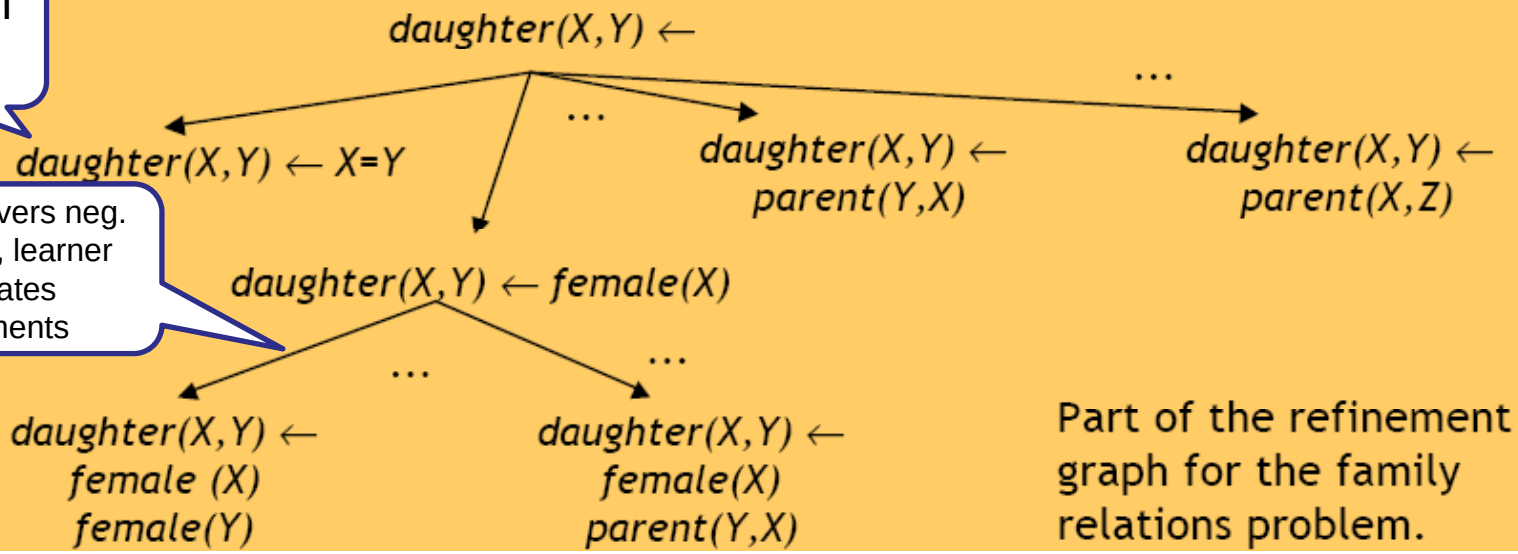

More general
(induction)
flies(X) ←
flies(X) ← bird(X)
flies(X) ← bird(X)
normal(X)
= true
More
specific
⊖ flies(oliver) ← bird(oliver)

Generality ordering of clauses

Training examples		Background knowledge	
daughter(mary,ann).	⊕	parent(ann,mary).	female(ann.).
daughter(eve,tom).	⊕	parent(ann,tom).	female(mary).
daughter(tom,ann).	⊖	parent(tom,eve).	female(eve).
daughter(eve,ann).	⊖	parent(tom,ian).	

Breadth first

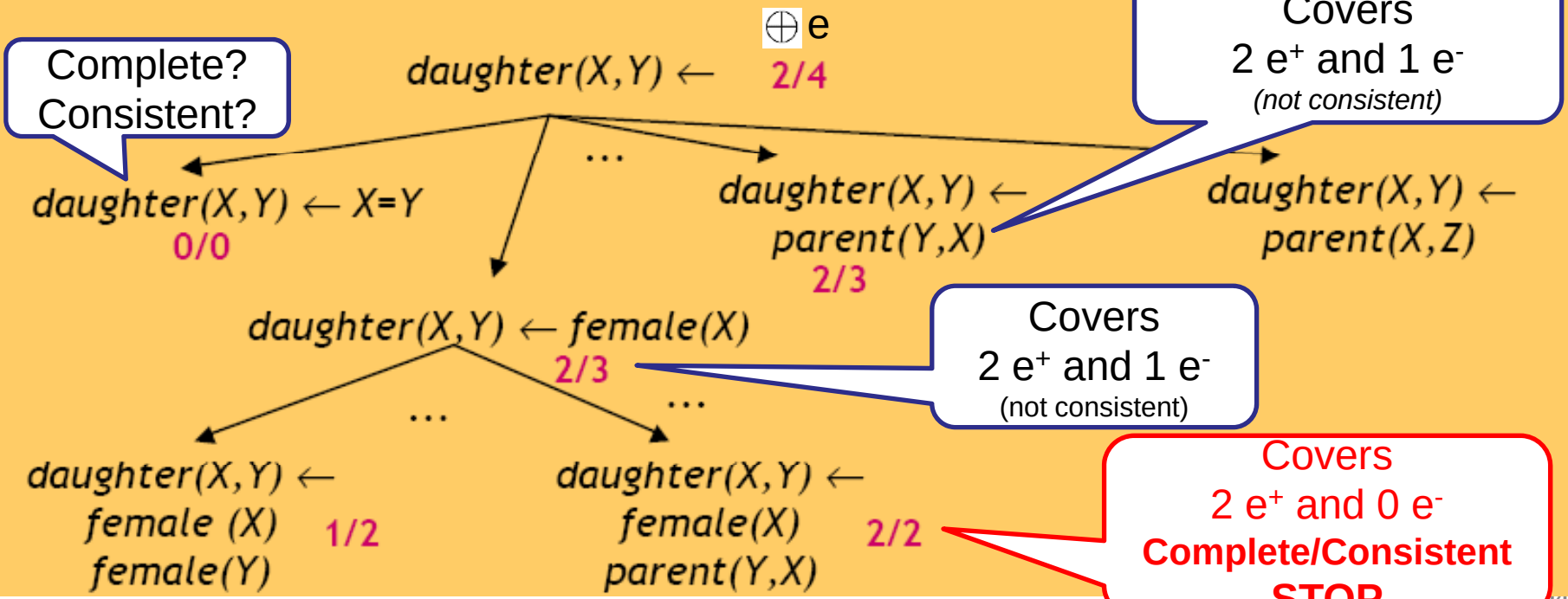
When c covers neg. examples, learner generates refinements



Part of the refinement graph for the family relations problem.

Greedy search of the best clause

Training examples		Background knowledge	
daughter(mary,ann).	⊕	parent(ann,mary).	female(ann.).
daughter(eve,tom).	⊕	parent(ann,tom).	female(mary).
daughter(tom,ann).	⊖	parent(tom,eve).	female(eve).
daughter(eve,ann).	⊖	parent(tom,ian).	



Demo

HANNE

http://139.18.2.57:8080/hanne/org.nlp2rdf.navigator.Application/Application.html

Conclusions

Inductive Concept Learning:

An inductive logic approach to concept learning

Comparison to Formal Concept Analysis

Requires more structured knowledge (cf. RDF, Semantic Web!)

Any questions?

See you next week!