Inductive Logic Programming 1.1
Inverting Entailment and Progol

Stephen Muggleton
Department of Computing
Imperial College, London

5th July, 2018

Overview of Inductive Logic Programming

Lecture 1.1 Inverse entailment and Progol [1+2]

Lecture 1.2 Meta-Interpretive Learning of grammars [3]

Lecture 1.3 MIL for Dyadic Datalog [4]

Lecture 2.1 MIL and bias reformulation [5]

Lecture 2.2 Meta-Interpretive Learning from noisy images [6]

Lecture 2.3 Stochastic Logic Programs and Bayesian Meta-Interpretive Learning [7,8]

Lecture material

Lecture material available from:

```
http://www.doc.ic.ac.uk/~shm/FLOC_ILP/Lecture1.1.pdf
http://www.doc.ic.ac.uk/~shm/FLOC_ILP/Lecture1.2.pdf
http://www.doc.ic.ac.uk/~shm/FLOC_ILP/Lecture1.3.pdf
http://www.doc.ic.ac.uk/~shm/FLOC_ILP/Lecture2.1.pdf
http://www.doc.ic.ac.uk/~shm/FLOC_ILP/Lecture2.2.pdf
http://www.doc.ic.ac.uk/~shm/FLOC_ILP/Lecture2.3.pdf
...
```

Papers for Lecture 1.1

Paper01: S.H. Muggleton. Inverse entailment and Progol. New Generation Computing, 13:245-286, 1995.

Paper02: S.H. Muggleton and C.H. Bryant. Theory completion using inverse entailment. In Proc. of the 10th International Workshop on Inductive Logic Programming (ILP-00), pages 130-146, Berlin, 2000. Springer-Verlag.

What is generalisation?

Statement A	Daffy Duck can fly
Statement B	All ducks can fly
Statement C	Marek lives in London
Statement D	Marek lives in England

Simple generalisation
Atom and Clause Subsumption

Given a substitution $\theta = \{v_1/t_1, \ldots, v_n/t_n\}$ and formula F. $F\theta$ is formed by replacing every variable v_i in F by t_i.

Atom A subsumes atom B, $A \succeq B$, iff there exists a substitution θ such that $A\theta = B$.

Clause C subsumes clause D, $C \succeq D$, iff there exists a substitution θ such that $C\theta \subseteq D$.

Generalisation example revisited

Daffy Duck can fly	$can\_fly(\text{daffy})$
All ducks can fly	$can\_fly(x)$

$$can\_fly(x) \succeq can\_fly(\text{daffy})$$

$$\theta = \{x/\text{daffy}\}$$

Generalisation as entailment

Entailment

C more general than D iff $C \models D$

Relative Entailment

C more general than D wrt B iff $B, C \models D$

Generalisation - harder example

C	Marek lives in London	lives(marek,london)
D	Marek lives in England	lives(marek,england)

Background knowledge

lives(x,england) $\leftarrow$ lives(x,london)

ILP general logical setting

B	Background Knowledge - Logic Program
E	Examples - Set of ground unit clauses
H	Hypothesis - Logic Program

Given B, E find H such that

$$B, H \models E$$

Search and refinement

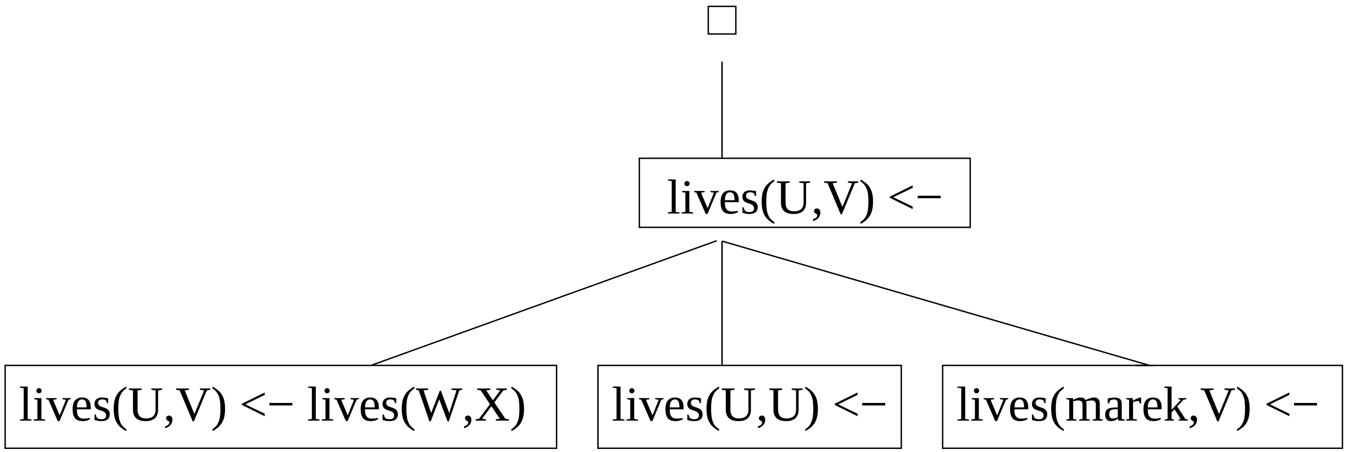
Given B, E find H such that

$$B, H \models E$$

Q : Algorithmically how do we find H given B, E?

A : Search space of clauses from simple to complex (general to specific) or complex to simple (specific to general). This process is called Clause Refinement .

Shapiro refinement graph ρ



Infinite descent search space
(Shapiro's MIS, Quinlan's FOIL)

$\square$

H

$$\square \succeq H$$

Inverting entailment

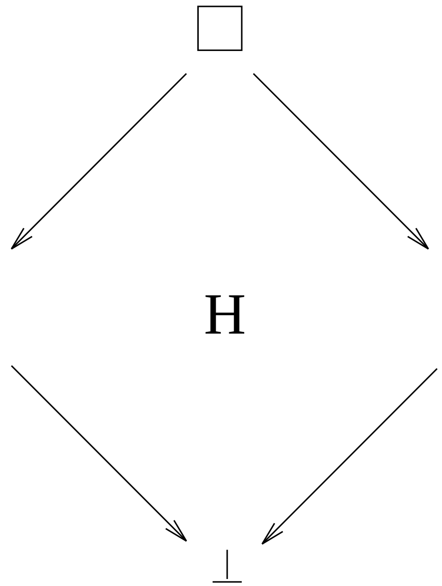
$$B \wedge H \models E$$

$$B \wedge \overline{E} \models \overline{H}$$

$$B \wedge \overline{E} \models \overline{\bot} \models \overline{H}$$

$$H \models \bot$$

Finite interval search space (Progol)



$$\Box \succeq H \succeq \bot$$

Effect of $\perp$

Search reduction using $\perp$ in mutagenesis domain

- average $\perp$ clause 26 atoms
- consider clauses with at most 3 atom/bond literals in body
- hypothesis space without $\perp$ is 3.5×10^7 clauses
- hypothesis space with bottom 2.5×10^3 clauses
- average pruned admissible search 2500 clauses

Summary

- Logical entailment provides a general framework for the notion of generalisaton.
- Refinement provides a mechanism for search through the space of generalisatons.
- Inverse entailment is a model-theoretic approach to ILP based on algebraic transformation of logical constraints.
- Progol uses admissible search and is efficient because it supports *finite interval* search.
- Mutagenesis example shows that *finite interval* is much more efficient than *infinite descent* search.