Logic-based Learning

Alessandra Russo and Mark Law

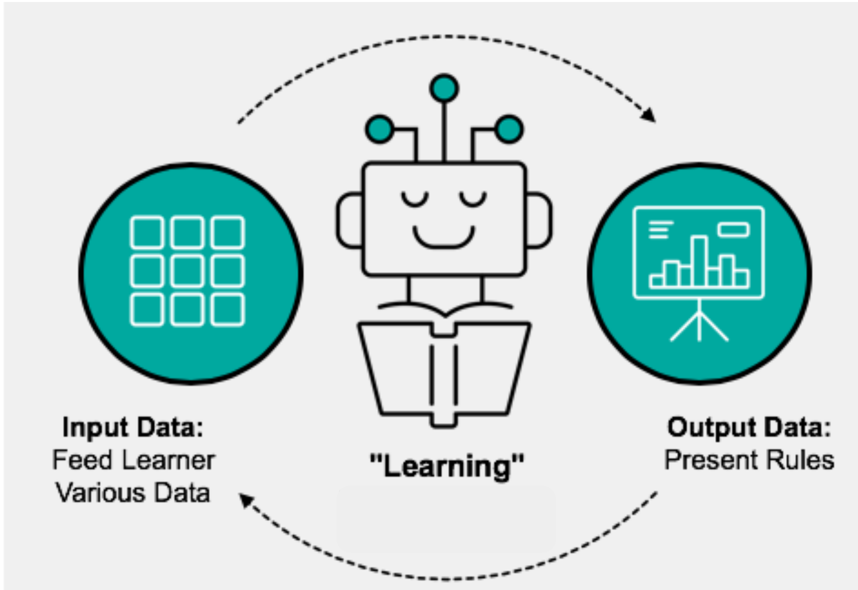
Joint Tutorial

RW 2019
The 15th Reasoning Web Summer School
20-24 September 2019 - Bolzano, Italy

Imperial College London

Overview

- Introducing Logic-based Learning
- Learning from entailment
  - Definition of learning task
  - Semantics
  - Algorithms
- Non-Monotonic Learning
  - Meta-level Learning
- Some applications

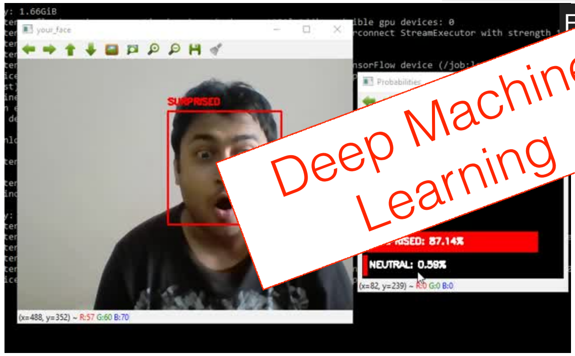


Imperial College London

Machine Learning in AI…

Deep neural networks learns from human expert games and games of self-play. [*Nature 2016]*

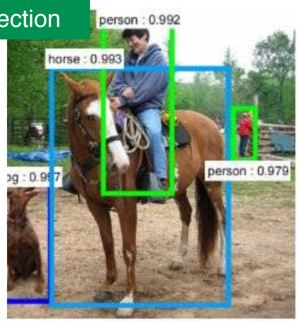

Single system teaches itself to master chess, shoji and go, using rules of the game.



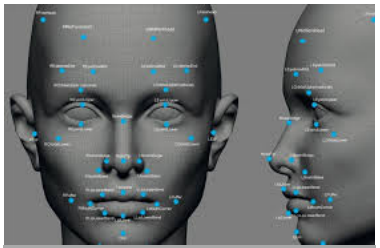
Fa… emotion
…tion

Deep Machine Learning


Object segmentation



Bounding box for object detection


Facial recognition

Advantages

- Learns from large datasets
- Very effective for single specific tasks
- Sometimes better than humans

Drawbacks

- Not able to use prior knowledge
- Not able to generalise
- Learned models are not interpretable

Imperial College London

… Logic-based Learning in AI

Artificially-intelligent Robot Scientist 'Eve' could boost search for new drugs

Published

04 Feb 2015

Image

Eve, the Robot Scientist

Automate scientific process using AI techniques to carry out cycles of scientific experiments. Automatically originate hy… s that explain observations, device experiments to … pothesis and physically run the experiments
[Letters to Nature 2003]

Logic-based Learning

Active molecules

Inactive molecules

Carcinogenicity, structural description of organic compounds

Advantages

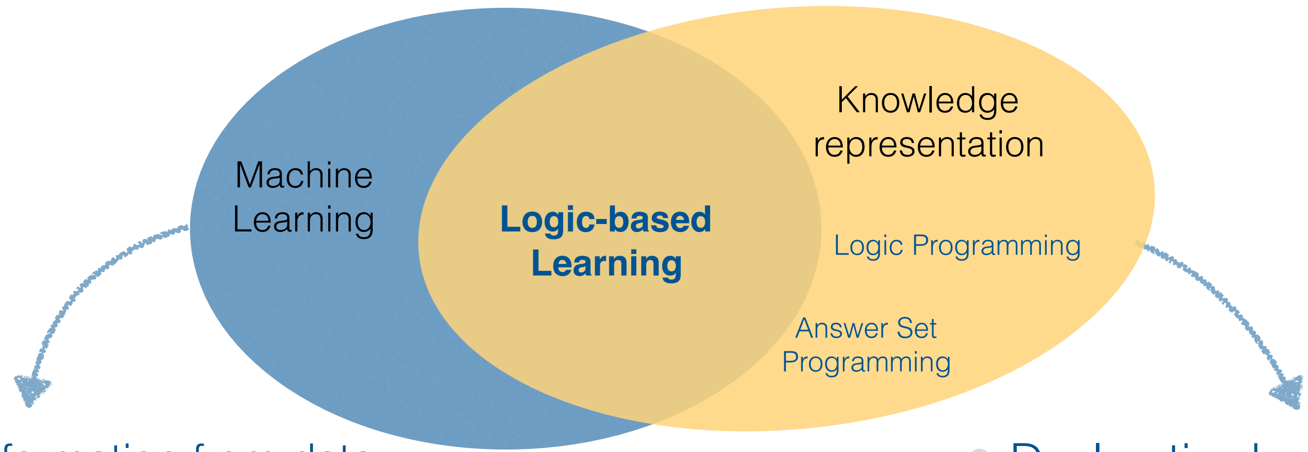
- Uses prior knowledge
- Able to generalise
- Can support continuous learning
- Learns from few examples
- Learned models are interpretable

Learning Grammars

Training Examples	Learned knowledge	Prior knowledge
s(0, 3) +	np(X, Y) ← word("She", X, Y)	word("She", 0, 1)
	mod(X, Y) ← word(quickly, X, Y)	word(quickly, 2, 3)
	s(X, Y) ← np(X, Z), vp(Z, Y)	word(ran, 1, 2)
	vp(X, Y) ← v(X, Y)	← v(1, 3)

Imperial College London

… Logic-based Learning in AI



- Extract information from data
- Make predictions on unseen data
- Learning from past observations

- Declarative knowledge
- Clear semantics
- Sound (and complete) inference

General-purpose machine learning algorithms

- learn from small (noisy) labelled structured data using declarative prior knowledge
- learn declarative knowledge expressed in some predicate logic formalism
  - can support transfer and continuous learning
- learned models are interpretable, and guaranteed to meet semantic properties

Imperial College London

An intuitive example

Learn the concept of "verb phrase"

$vp(Start, End) \leftarrow v(Start, Middle), mod(Middle, End)$

"$_0$She$_1$ ran$_2$ quickly$_3$"

e^+ vp(1,3)
e^- vp(0,1)
e^- vp(0,3)

$$BK \cup \mathbf{H} \models e^+$$
$$BK \cup \mathbf{H} \not\models e^-$$

Observation Predicate Learning

Background knowledge BK $\begin{cases} np(0, 1). \\ v(1, 2). \\ mod(2, 3). \end{cases}$

s(X,Y)←np(X,Z),vp(Z,Y)
vp(X,Y)←v(X,Y)

What about learning concepts that are different from the given example?

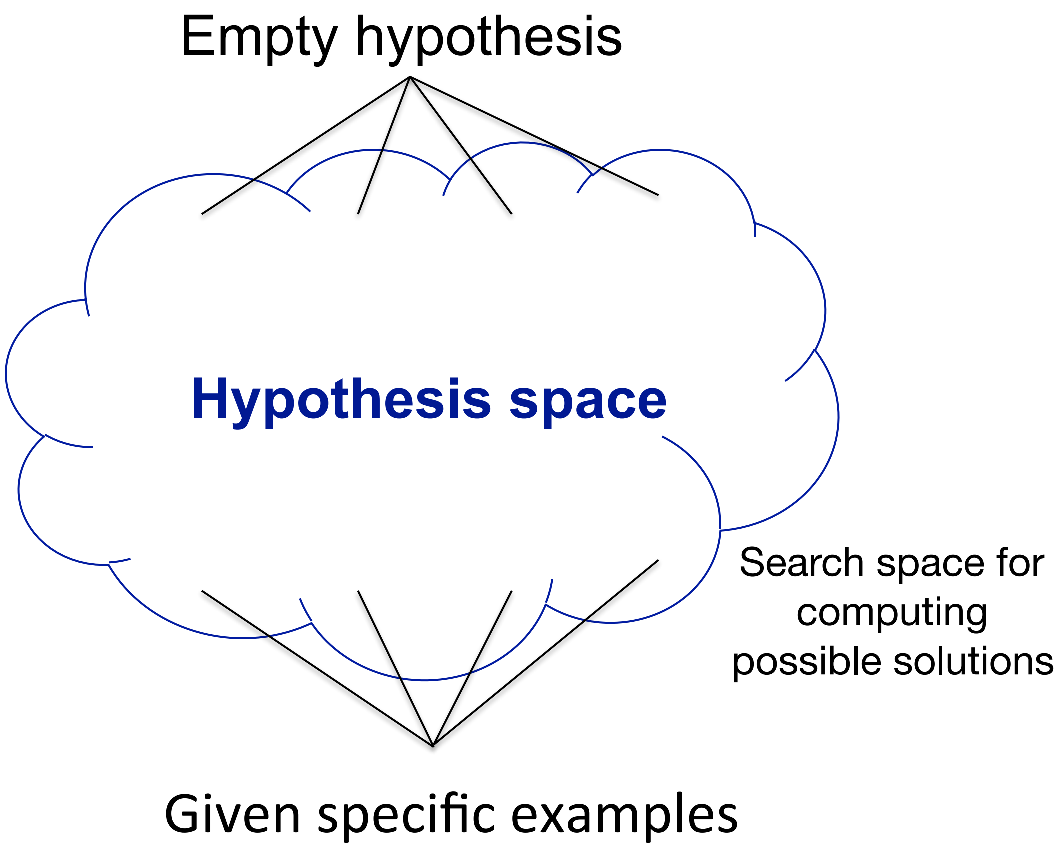
"$_0$She$_1$ ran$_2$quickly$_3$"

e^+ s(0,3)
e^- s(0,1)

Non-Observation Predicate Learning

Learning as a search problem

Logic-based learning: a computational mechanism for inducing declarative programs from examples of what is known to be true or false (in the models of the learned programs).



How do we define a learning task?

How do we search for solutions in a given search space?

Imperial College London

Learning Task: informal definition

Given

- Set of *positive examples* (E^+) and set of *negative examples* (E^-) in $\mathcal{L}_e$
- Background knowledge (B) in $\mathcal{L}_B$
- Set of possible solutions (S_M) in a *language bias* $\mathcal{L}_H$
- *Covers* relation over $\mathcal{L}_B$, $\mathcal{L}_H$ and $\mathcal{L}_e$

Find

- Solution $H \in S_M$ such that:
  - *Covers*(B, H, e) for every $e \in E^+$ (H is complete)
  - $\neg$*Covers*(B, H, e) for every $e \in E^-$ (H is consistent)

Different notions of Covers relation capture different learning frameworks.

[Logical Setting for concept-learning. Luc De Raedt, AIJ 95, 187-201]

Imperial College London

Learning Task: informal definition

Given

- Set of *positive examples* (E^+) and set of *negative examples* (E^-) in $\mathcal{L}_e$
- Background knowledge (B) in $\mathcal{L}_B$
- Set of possible solutions (S_M) in a *language bias* $\mathcal{L}_H$
- *Covers* relation over $\mathcal{L}_B$, $\mathcal{L}_H$ and $\mathcal{L}_e$
- *Quality* criterion over $\mathcal{L}_B$, $\mathcal{L}_M$ and $\mathcal{L}_e$, scoring possible solutions

Find

- Solution $H \in S_M$ such that:
  - *Covers*(B, H, e) for every $e \in E^+$ (H is complete)
  - $\neg$*Covers*(B, H, e) for every $e \in E^-$ (H is consistent)
  - H has the highest quality.

Different notions of Covers relation capture different learning frameworks.

[Logical Setting for concept-learning. Luc De Raedt, AIJ 95, 187-201]

Imperial College London

Learning from entailment

$\mathcal{L}_B$ and $\mathcal{L}_M$ are languages for definite clausal theories

A *learning from entailment* task T_{LFE} is a tuple (B, S_M, E^+, E^-) where

B is a definite clausal theory,

S_M is a set of clauses,

E^+ and E^- are sets of facts

Covers(B, H, e) iff $B \cup H \models e$, where $H \subseteq S_M$

A clausal theory $H \subseteq S_M$ is an inductive solution of T_{LFE} if and only if

- *Covers*(B, H, e) $\forall e \in E^+$
- $\neg$*Covers*(B, H, e) $\forall e \in E^-$

LFE: example of learning task

Consider $T_{LFE} = (B, S_M, E^+, E^-)$ task given by:

$$B = \left\{ \begin{array}{l} parent(ann, mary) \\ parent(ann, tom) \\ parent(tom, eve) \\ parent(tom, ian) \\ female(ann) \\ female(mary) \\ female(eve) \end{array} \right.$$

$$E^+ = \left\{ \begin{array}{l} daughter(mary, ann) \\ daughter(eve, tom) \end{array} \right.$$

$$E^- = \left\{ \begin{array}{l} daughter(tom, ann) \\ daughter(eve, ann) \end{array} \right.$$

$$S_M = \left\{ \begin{array}{l} h_1 = daughter(X,Y) \leftarrow female(X) \\ h_2 = daughter(X,Y) \leftarrow parent(Y,X) \\ h_3 = daughter(X,Y) \leftarrow parent(Y,X), female(X) \end{array} \right.$$

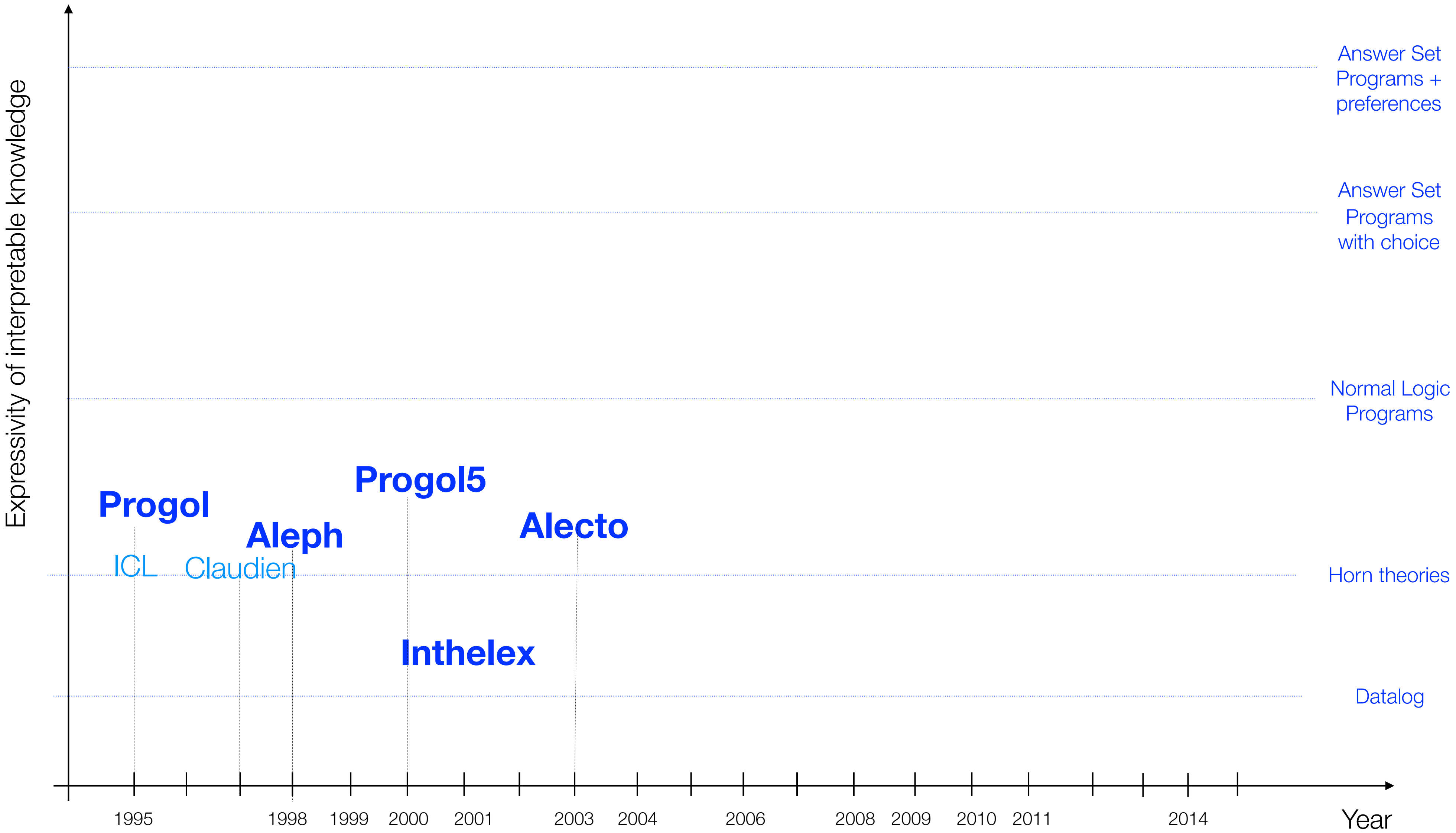
$B \cup \{h_1\} \models daughter(eve, ann)$ ➡ h_1 is an not an inductive solution of T_{LFE}

$B \cup \{h_2\} \models daughter(tom, ann)$ ➡ h_2 is an not an inductive solution of T_{LFE}

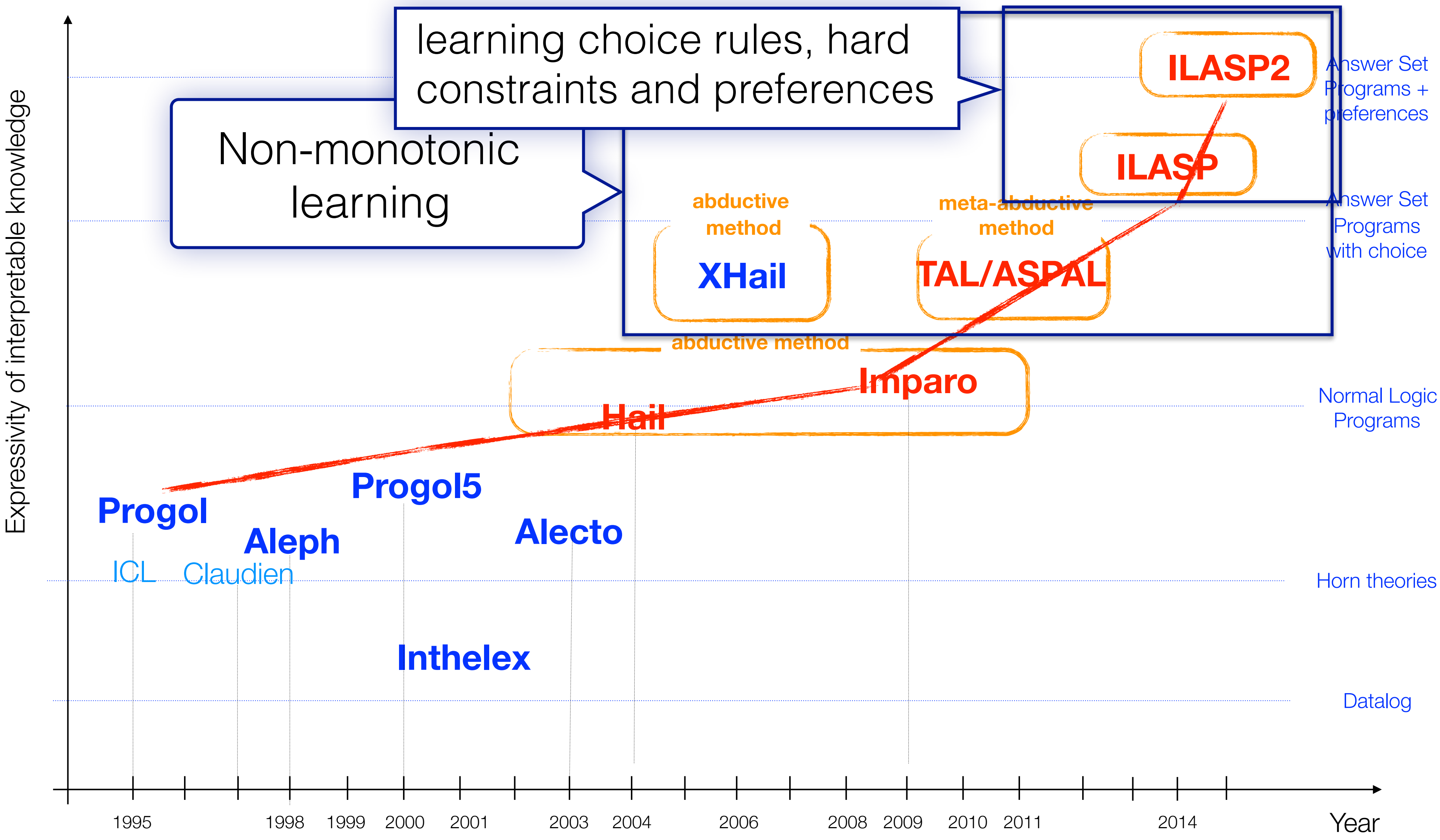
$$\left. \begin{array}{l} B \cup \{h_3\} \models daughter(mary, ann) \\ B \cup \{h_3\} \models daughter(eve, tom) \\ B \cup \{h_3\} \not\models daughter(eve, ann) \\ B \cup \{h_3\} \not\models daughter(tom, ann) \end{array} \right\}$$

➡ **h_3 is an inductive solution of T_{LFE}**

Early Algorithms and Systems…



Our recent advancement…



The complexity and generality of Learning Answer Set Programs, Mark Law, Alessandra Russo, Krysia Broda, AIJ (2018).

Learning from entailment

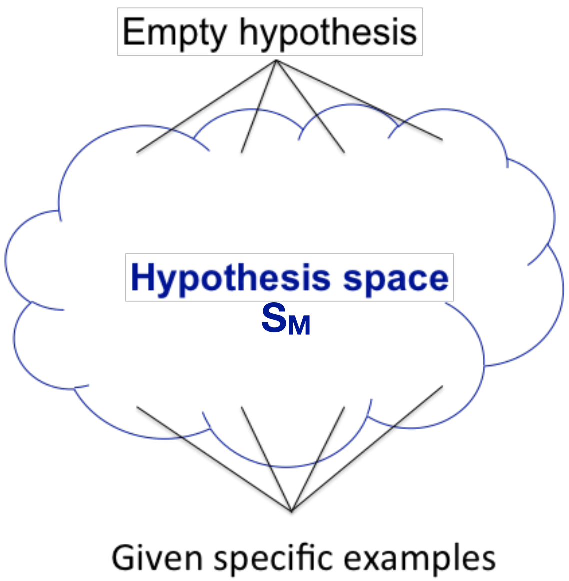
Definition

A LFE task T_{LFE} is a tuple (B, S_M, E^+, E^-) where B is a definite clausal theory, called *background knowledge*, S_M is a set of clauses, called *hypothesis space*, E^+ is a set of facts, called *positive examples,* and E^- is a set of facts, called *negative examples*.

An hypothesis $H \subseteq S_M$ is an inductive solution of T_{LFE} if and only if

(i) $B \cup H \models e^+ \quad \forall e^+ \in E^+$ (ii) $B \cup H \not\models e^- \quad \forall e^- \in E^-$

How do we search for solutions in a given hypothesis space?



Generality Relation

H_i more general then H_j iff $H_i \models H_j$

- $\neg covers(B, H_i, e^+) \Rightarrow \neg covers(B, H_j, e^+)$
- $covers(B, H_j, e^-) \Rightarrow covers(B, H_i, e^-)$
- H_i generalises H_j iff H_i θ-subsumes H_j

Imperial College
London

Defining the hypothesis space

Language bias $\mathcal{L}_H$ is defined declaratively by mode declarations.

head declaration: modeh(s)

body declaration: modeb(s)

s is a ground atom with one or more placemarkers: +t, -t, #t, where t denotes a type

M = modeh(grandfather(+p,+p))
modeb(father(+p,-p))
modeb(parent(+p,+p)

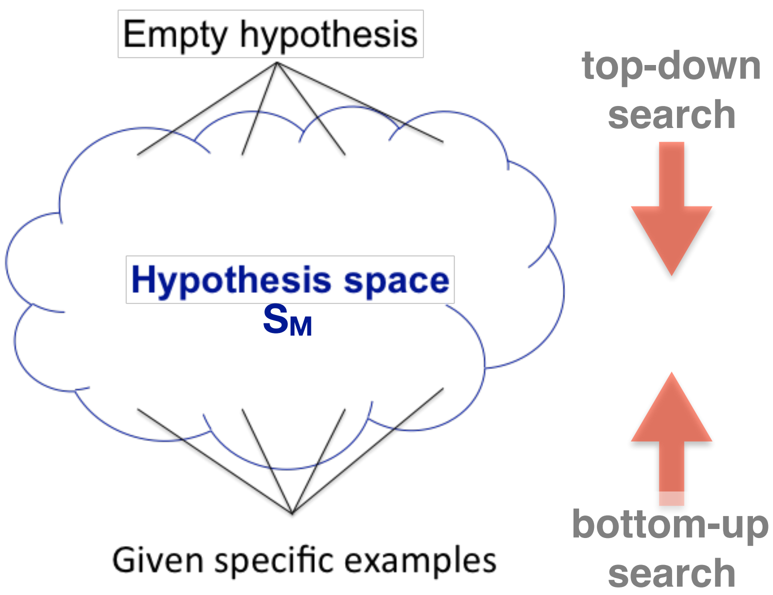
where p is type person

grandfather(X,Y) ← father(X,Z), parent(Z,Y) Compatible with M

grandfather(X,Y) ← parent(X,Z), father(Z,Y) Not compatible with M

S_M is the set of all clauses that are compatible with the set of mode declarations M.

Searching for solutions



Use of efficient specialisation operators.

- Shapiro's refinement operators
- Quinlan's FOIL system

Use of efficient generalisation operators.

- Plotkin's least general generalisation
- Muggleton's inverse resolution (GOLEM, CIGOL,…)

Mixed approach:

Covering loop over the set of positive examples

1. Compute most specific solution for a given example
2. Generalise the most specific clause.

Progol5
HAIL
Imparo

Inverse entailment (IE) Approach

Mechanism for computing the most specific solution for a given example

Given a learning task $T_{LFE} = (B, S_M, E^+, E^-)$, and an example $e^+ \in E^+$

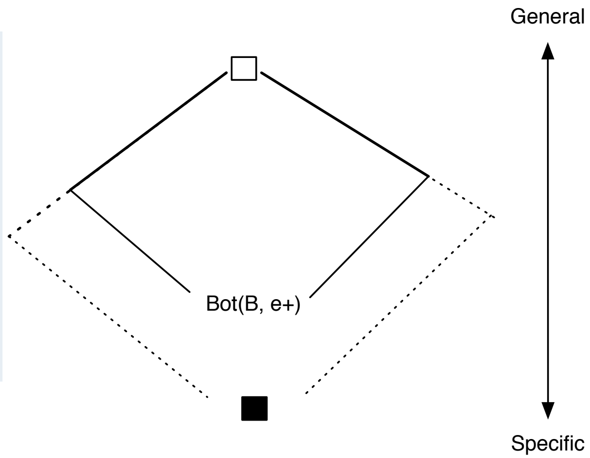
$$B \cup \{h\} \models e^+ \quad \text{iff} \quad B \cup \{\neg e^+\} \models \neg h$$

The negation of an hypothesis can be generated deductively from $B \cup \{\neg e^+\}$.

Let $\neg Bot(B,e^+)$ be the negation of the most specific clause that covers a given examples, called Bottom Clause, denoted $Bot(B,e^+)$.

1. $B \cup \{\neg e^+\} \models \underbrace{\neg l_1\partial \wedge l_2\partial \wedge \ldots \wedge l_n\partial}_{\neg Bot(B,e^+)}$

2. $\neg Bot(B,e^+) \models \neg h \quad \Rightarrow \quad h \models Bot(B,e^+)$

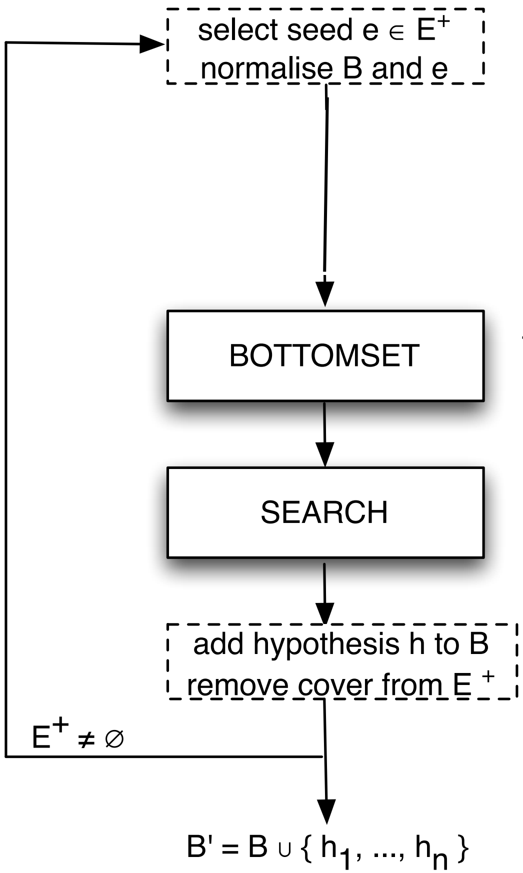


h is derivable by Bottom Generalisation iff h θ-subsumes $Bot(B,e^+)$

Imperial College London

Progol

- Use *Covering loop*: compute an hypothesis for each seed example e^+
- *Mode Declarations* M to constrain the computation of the $Bot(B,e^+)$



$\neg Bot(B,e^+)$ computed using SLD.

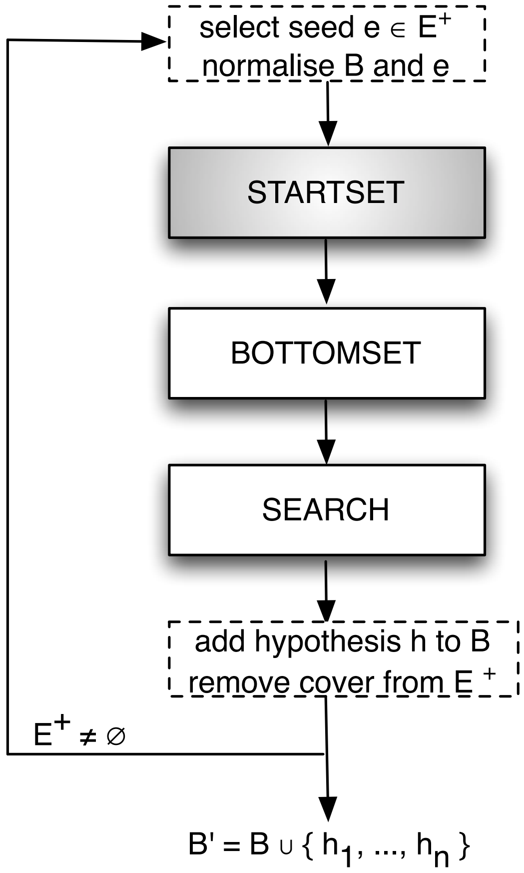
top-down specialisation to compute h

Not able to support non-observation predicate learning

Imperial College London

Progol5

- Use *Covering loop*: compute an hypothesis for each seed example e^+
- *Mode Declarations* M to constrain the computation of the Bot(B,e^+)



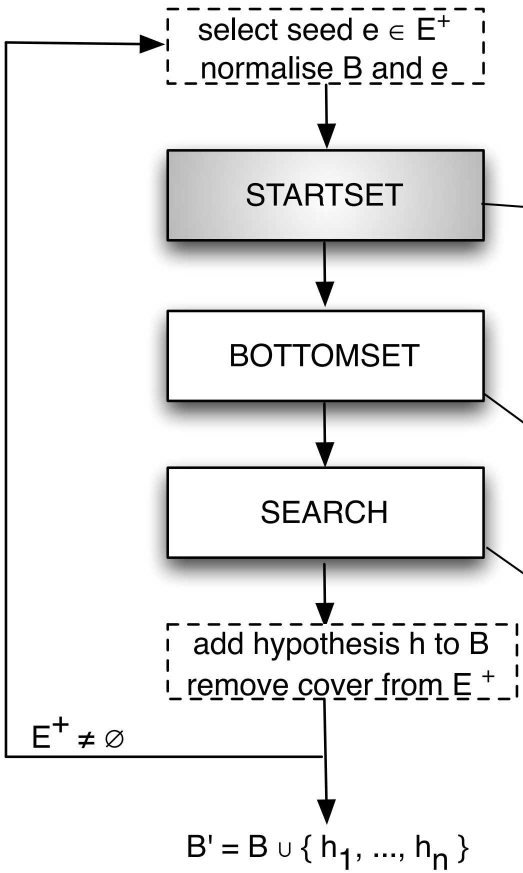
Compute B*, by extending B with contrapositives, computes negated (head) atom in ¬Bot(B*,e^+) compatible with modeh

Computes (body) positive atoms in ¬Bot(B,e^+) compatible with modeb

Top-down specialisation to compute h that θ-subsumes Bot(B,e^+)

Progol5

- Use *Covering loop*: compute an hypothesis for each seed example e^+
- *Mode Declarations* M to constrain the computation of the Bot(B,e^+)



$$B = \begin{cases} p(X) \leftarrow q(X) \\ r(a) \end{cases} \qquad E^+ = \{p(a)\} \qquad E^- = \{p(b)\}$$

$$M = \{\ \text{modeh}(q(+\text{any})),\ \text{modeb}(r(+\text{any}))\ \}$$

$$B^* = \begin{cases} p(X) \leftarrow q(X) \\ q^*(X) \leftarrow p^*(X) \\ r(a) \end{cases} \qquad \neg e^+ = \{p^*(a)\}$$

$$\begin{array}{c} q^*(X) \\ | \\ p^*(X) \\ | \quad \theta = \{X/a\} \\ \square \end{array}$$

$$B \cup \{\neg e^+\} \models \{\neg q(a)\}$$

$$B \cup \{\neg e^+\} \models \{r(a)\}$$

$$\neg Bot(B, e^+) = \{\neg q(a), r(a)\}$$

$$Bot(B, e^+) = q(a) \leftarrow r(a)$$

$$h = q(X) \leftarrow r(X)$$

Imperial College London

Incompleteness of Progol5

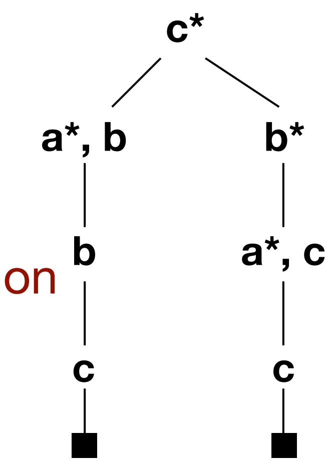
$B = \left\{ \begin{array}{l} a \leftarrow b,\ c \\ b \leftarrow c \end{array} \right.$ $\quad E^+ = \{a\}$ $\quad h = \{ c \}$

h is derivable by Bottom Generalisation but cannot be computed by Progol5

$B \cup \{\neg e^+\} = B \cup \{\neg a\} \models \neg a \wedge \neg c$ $\quad \Rightarrow \quad$ $c \in Bot(B,e)$

h θ-subsumes $Bot(B,e^+)$

$B^* = \left\{ \begin{array}{l} a \leftarrow b,\ c \\ c^* \leftarrow a^*, b \\ b^* \leftarrow a^*, c \\ b \leftarrow c \\ c^* \leftarrow b^* \end{array} \right.$ $\quad E^+ = \{a\}$ $\quad h = \{ c \}$



Failed SLD derivation

$c \notin STARTSET(B, e^+)$

Generalising Bottom Set

Theory completion by contrapositive is not sufficient to compute the full semantics of Bottom Generalisation

Bottom Set $\{\, lg \mid B \cup \{\neg e^+\} \models \neg lg \} = \{\alpha \mid B \cup \{\neg e^+\} \models \neg\alpha\} \cup \{\neg\beta \mid B \cup \{\neg e\} \models \beta\}$

Abduction　　　　Deduction

$= \{\alpha \mid B \cup \alpha \models e^+\} \cup \{\neg\beta \mid B \models \beta\}$

KernelSet

$$\left\{\begin{array}{l} \alpha_1 \leftarrow \beta^{11}, \ldots, \beta^{1m} \\ \vdots \\ \alpha_i \leftarrow \beta^{i1}, \ldots, \beta^{ih} \\ \vdots \\ \alpha_n \leftarrow \beta^{11}, \ldots, \beta^{1k} \end{array}\right.$$

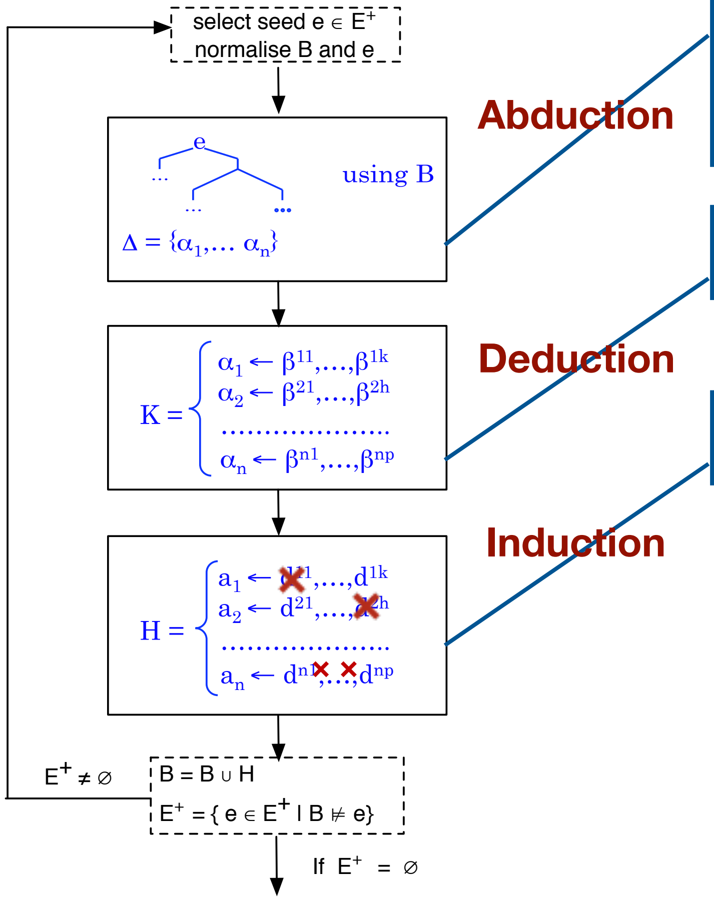
Deductive consequences
$B \models \beta^{ij}$

Abductive explanation
$B \cup \{\alpha_1, \ldots, \alpha_1\} \models e^+$

Oliver Ray, Krysia Broda, Alessandra Russo. Generalised Kernel Sets for Inverse Entailment. ICLP 2004: 165-17.

Hybrid abductive inductive learning (HAIL)

Consider a LFE task $T_{LFE} = (B, S_M, E^+, E^-)$ where B is a definite clausal theory, S_M is a set of clauses, E^+ is a set of positive examples and E^- is a set of negative examples.



(B, Ab, IC), abductive task:
Ab = $\{h\theta \mid \text{modeh}(h(.))\}$, IC possibly empty
$\Delta = \{\alpha_1,\ldots,\alpha_n\} \subseteq \text{Ab}$

$B \vdash \beta^{ij}$ for any modeb(β(.))

H θ-subsumes K

A clausal theory H is derivable by Kernel Set Subsumption from B and e iff H θ-subsumes K(B,e^+).

Oliver Ray, Krysia Broda, Alessandra Russo, A Hybrid Abductive Inductive Proof Procedure. Logic Journal of the IGPL 12(5): 371-397 (2004).

Imperial College London

HAIL example

$$B = \begin{cases} sad(X) \leftarrow tired(X), poor(X) \\ academic(oli) \\ academic(ale) \\ academic(kb) \\ student(oli) \\ lecturer(ale) \\ lecturer(kb) \end{cases}$$

$$E^+ = \begin{cases} sad(ale) \\ sad(kb) \end{cases} \qquad E^- = \begin{cases} sad(oli) \\ poor(oli) \end{cases}$$

$$M = \begin{cases} modeh(tired(+academic)) \\ modeh(poor(+academic) \\ modeb(lecturer(+academic) \\ modeb(academic`(+academic)) \end{cases}$$

1. Abduction:

   Δ = {tired(ale), poor(ale)}

2. Deduction:

   $B \models$ {academic(ale), academic(kb), lecturer(ale), lecturer(kb)}

$$K_g = \begin{cases} tired(ale) \leftarrow academic(ale), lecturer(ale) \\ poor(ale) \leftarrow academic(ale), lecturer(ale) \end{cases}$$

$$K = \begin{cases} tired(X) \leftarrow academic(X), lecturer(X) \\ poor(X) \leftarrow academic(X), lecturer(X) \end{cases}$$

3. $H = \begin{cases} tired(X) \\ poor(X) \leftarrow lecturer(X) \end{cases}$

Imperial College London

Further case of incompleteness

Progol5 incomplete for non-Observation Predicate Learning (non-OPL)

Bottom Set incomplete with respect to multiple clause learning

Yamamoto's example

$B = \left\{ \begin{array}{l} even(s(X)) \leftarrow odd(X) \\ even(0) \end{array} \right.$ $M = \left\{ \begin{array}{l} modeh(odd(s(+any))) \\ modeb(even(+any)) \end{array} \right.$ $E^+ = \left\{ odd(s(s(s(0)))) \right.$

Can we learn the following concept $H = \left\{ odd(s(X)) \leftarrow even(X) \right.$

$Bot(B,e^+) = \{odd(s(s(s(0)))) \leftarrow even(0)\}$ ➡ H not θ-subsumes $Bot(B,e^+)$

$K_g\ (B,e^+) = \{odd(s(s(s(0)))) \leftarrow even(0)\}$ ➡ H not θ-subsumes $K(B,e^+)$

So where is the problem?

Induction on Failure

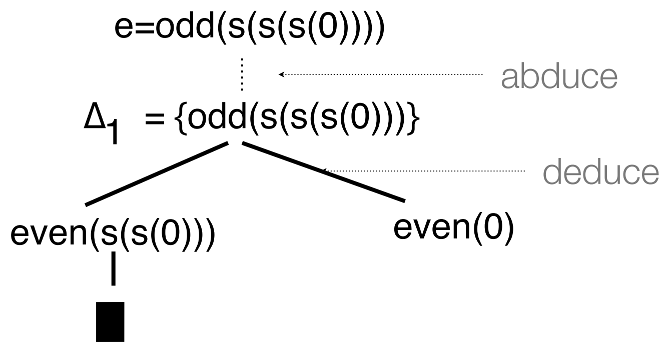
Yamamoto's example

$$B = \left\{ \begin{array}{l} even(s(X)) \leftarrow odd(X) \\ even(0) \end{array} \right.$$

$$M = \left\{ \begin{array}{l} modeh(odd(s(+any))) \\ modeb(even(+any)) \end{array} \right.$$

$$E^{+} = \left\{ odd(s(s(s(0)))) \right.$$

$$?\, H = \left\{ odd(s(X)) \leftarrow even(X) \right.$$



Induction on Failure

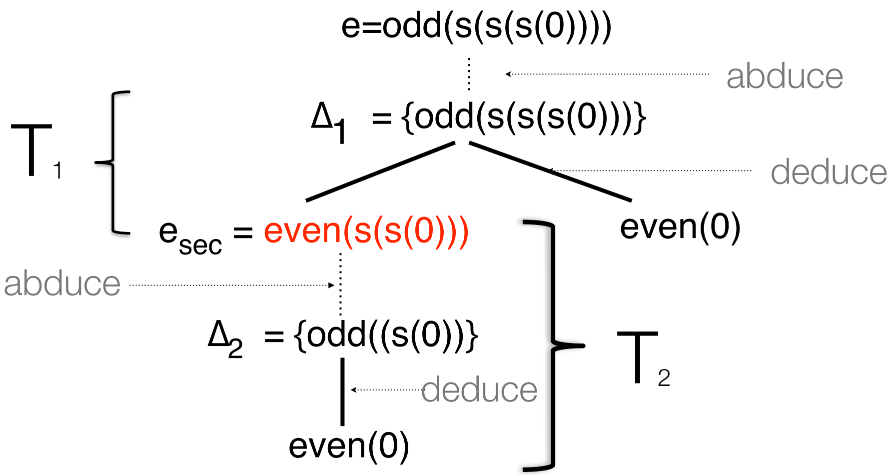
Yamamoto's example

$$B = \left\{ \begin{array}{l} even(s(X)) \leftarrow odd(X) \\ even(0) \end{array} \right.$$

$$M = \left\{ \begin{array}{l} modeh(odd(s(+any))) \\ modeb(even(+any)) \end{array} \right.$$

$$E^+ = \left\{ odd(s(s(s(0)))) \right.$$

$$?\, H = \left\{ odd(s(X)) \leftarrow even(X) \right.$$



$T_0 = \{\, odd(s(s(s(0))) \leftarrow even(s(s(0)),\ even(0)\}$

body atoms proved abductively

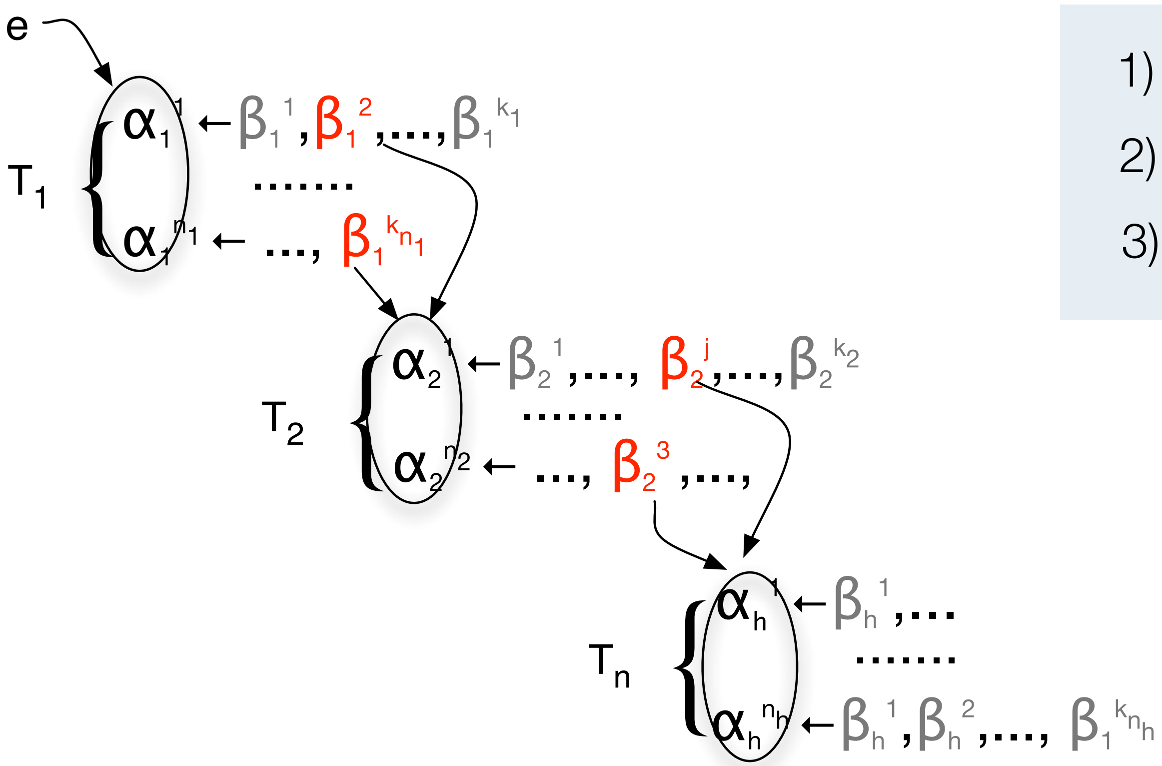
$T_1 = \{\, odd(s(0))) \leftarrow even(0)\ \}$

$$\overbrace{odd(s(X)) \leftarrow even(X)\}}^{H} \models \overbrace{\begin{array}{l} T_0 = \{\, odd(s(s(s(0))) \leftarrow even(s(s(0)),\ even(0)\} \\ T_1 = \{\, odd(s(0))) \leftarrow even(0)\ \} \end{array}}^{T = T_0 \cup T_1}$$

Imperial College London

Extending Kernel Sets to Connected Theories

$$T = T_1 \cup T_2 \cup \ldots \cup T_n$$



1) $B \cup T_1^+ \models E$
2) $B \cup T_j^+ \models T_{j-1}^-$ $\quad 1 < j \leq n-1$
3) $B \models T_n^-$

A clausal theory H is derivable by Connected Theory Generalisation from B and e iff H θ-subsumes T

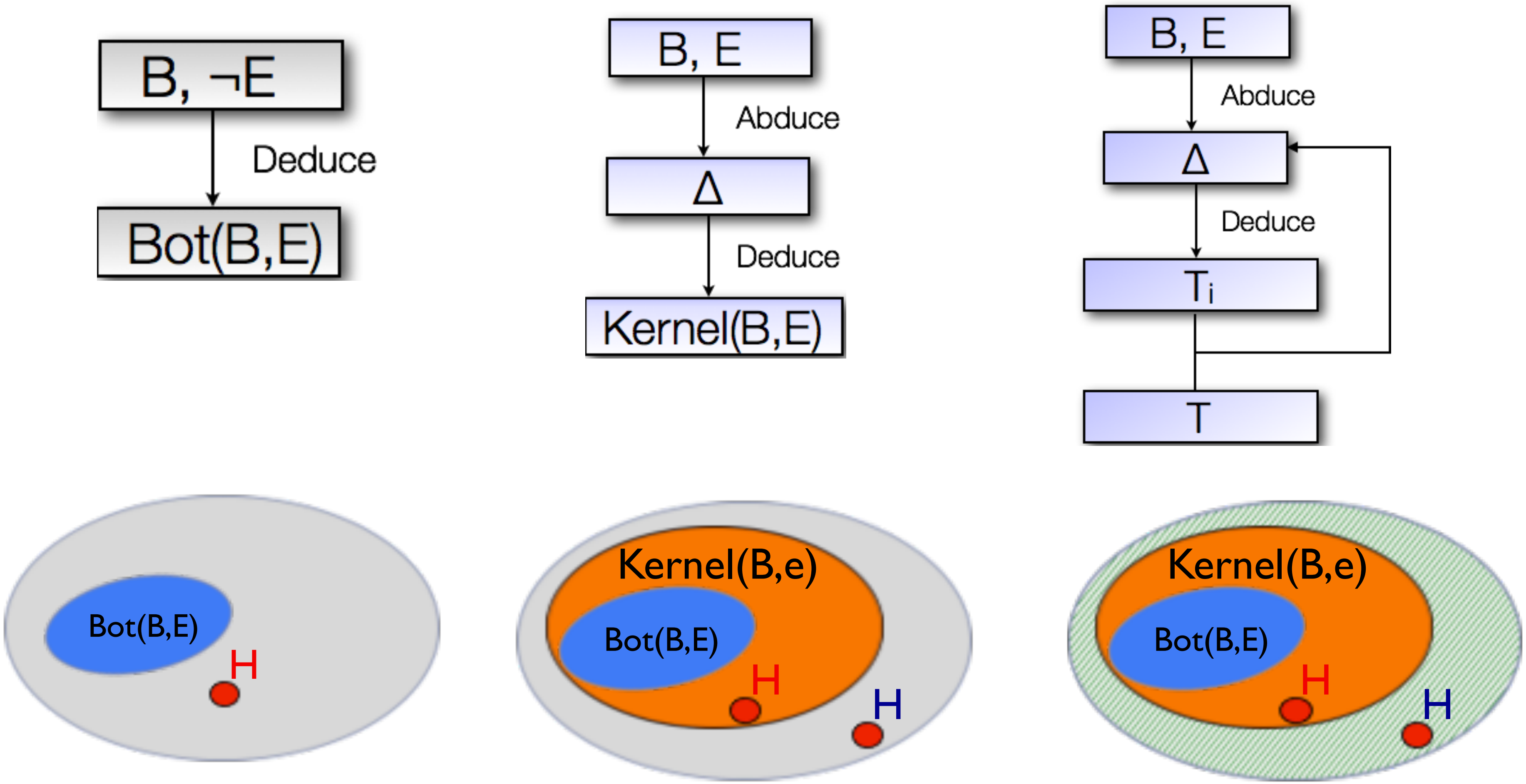
Imperial College London

IE: semantics generalisations

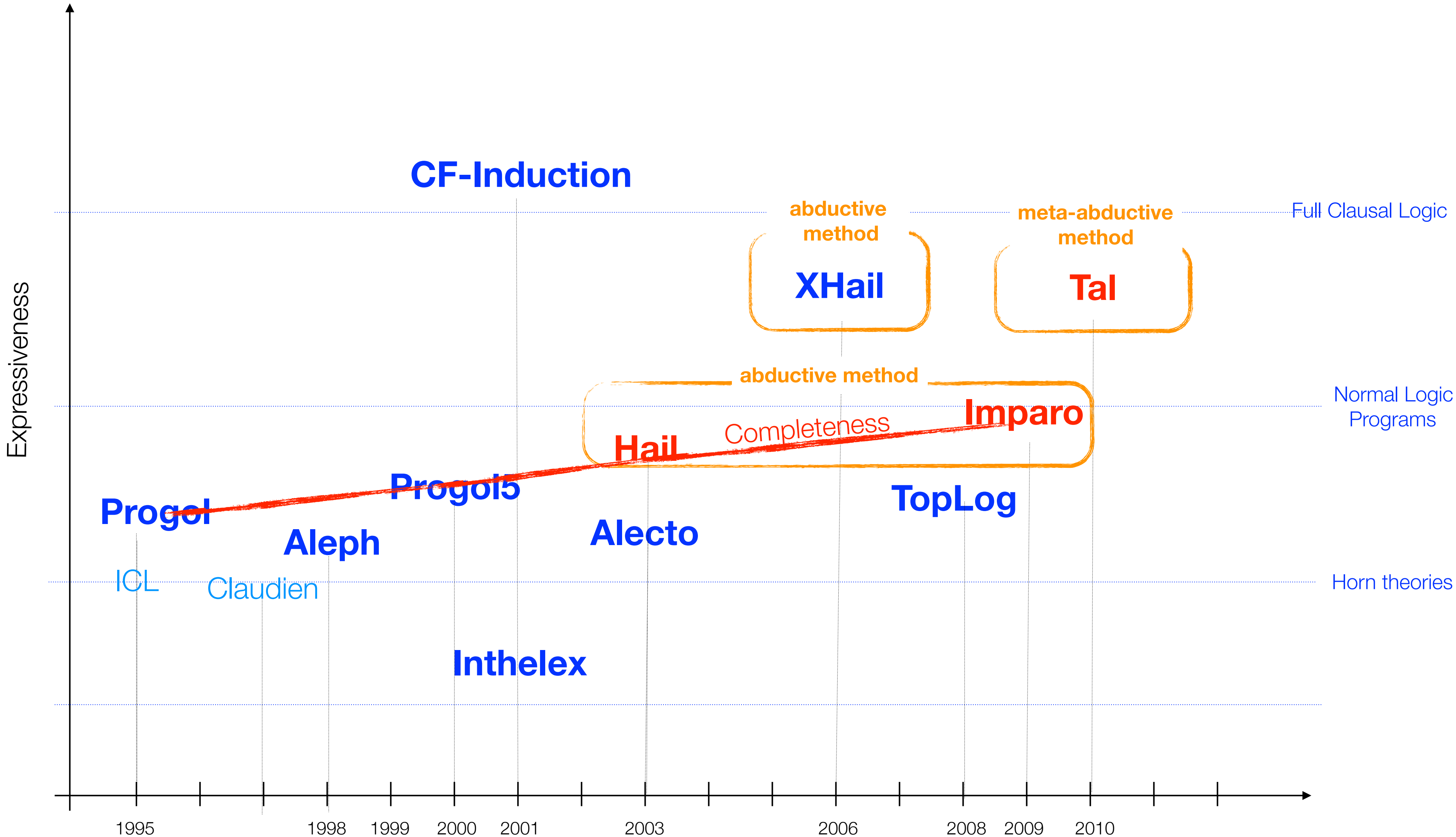
Bottom Set
Progol5

Kernel Set
Hail

Connected Theory
Imparo



But,…what about non-monotonic learning?



Imperial College London

But,…what about non-monotonic learning?

Background knowledge (B) and hypothesis (H) are normal logic programs

- Covering loop search strategy is no longer applicable
- Incremental learning and generalisation techniques for definite programs are unsound

B = obeys(X,Y) ← *not* officer(X), officer(Y)
wears_hat(price)
wears_hat(osbourne)
has_stripe(osbourne)

H_{ground} = officer(osbourne) ← wears_hat(osbourne)

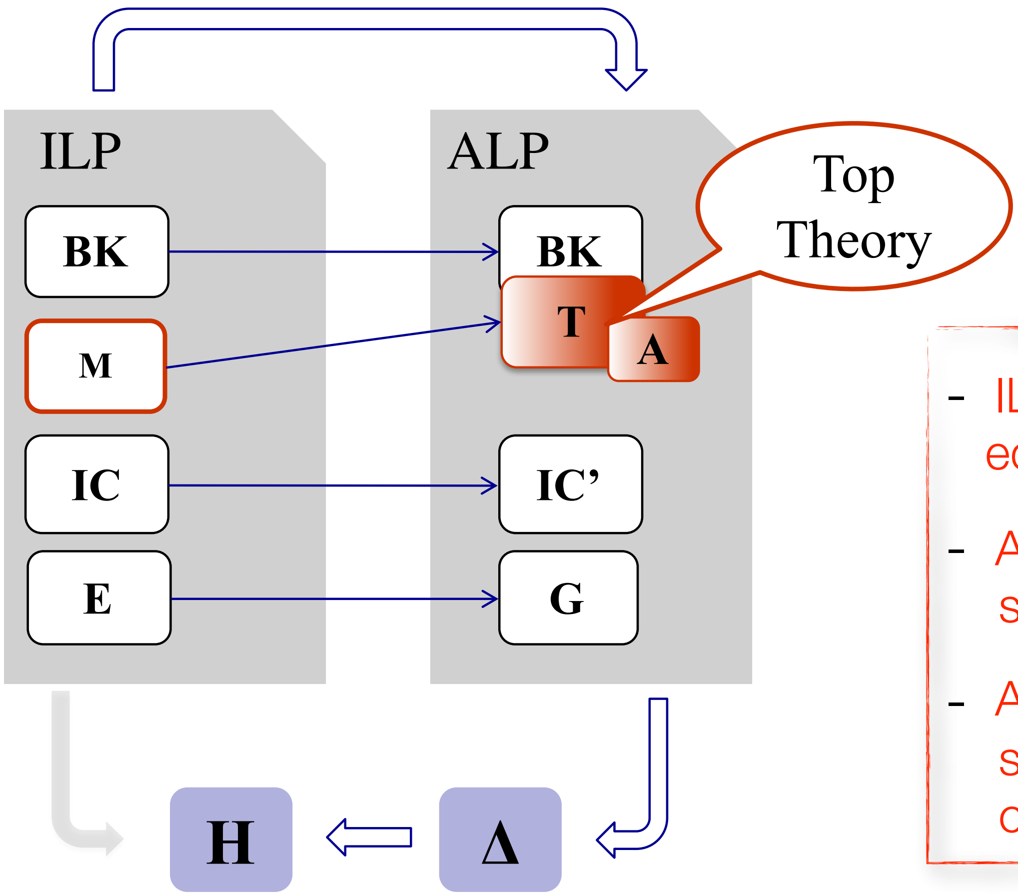
E^+ = obeys(prince,osbourne)

H = ~~officer(X) ← wears_hat(X)~~

M = modeh(officer(+any))
modeb(has_stripe(+any))

Top-Directed Abductive Leaning (TAL)

- Learning hypotheses using a top-down approach.
- Computation and generalization of hypotheses combined into a single abductive-based proof procedure.



- ILP task translated into an equivalent ALP task
- Abductive reasoning over the structure of the rules
- ALP is used to compute the full solutions, instead of being a component of a learning system

Top-Directed Abductive Leaning (TAL)

Algorithm: TAL

Input: Learning task <B, S_M, E>

B background knowledge, E examples, S_M hypothesis space

Output: H hypothesis

T_M = Pre-processing(B, E, S_M)

Δ = Abduce(B ∪ T_M, {rule(.)}, Ø) with goal E

H = Post-processing(Δ, M)

TAL: Example

B = [even(0)

M =
eh: modeh(even(+nat))
oh: modeh(odd(+nat))
bno: modeb(not odd(+nat))
be: modeb(even(+nat))
bs: modeb(+nat = s(+nat))

E = | odd(s(s(s(0))))
not odd(s(s(0)))

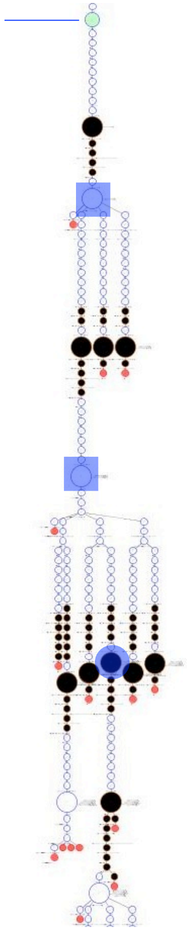
```
even(X) ← body( [X], [(eh, [ ], [ ])] )
odd(X) ← body([X], [(oh, [ ], [ ])] )

body(InputSoFar, Rule) ← rule(Rule)
body(InputSoFar, Rule) ←
        not odd(X),
        link_variables([X], InputSoFar, Links),
        append(Rule, [(bno, Links,[ ])], NRule),
        append(InputSoFar, [ ], NewInputs),
        body(NewInputs, NRule).
body(InputSoFar, Rule) ←
        even(X),
        link_variables([X], InputSoFar, Links),
        append(Rule, [(be, [ ], Links)], NRule),
        append(InputSoFar, [ ], NewInputs),
        body(NewInputs, NRule).
body(Inputs, Rule) ←
        s(X) = Y,
        link_variables([X], InputSoFar, Links),
        append(Rule, [(bs, [ ], Links)], NRule),
        link_variables([X], Inputs, Links),
        append(InputSoFar, [ ], NewInputs),
        body(NewInputs, NRule).
```

Δ = { rule([(oh,[ ], [ ]), (bs, [ ], [1]), (be, [ ], [2])]),
rule([(eh,[ ], [ ]), (bno, [ ], [1])] }

H = { odd(X) ←s(X) = Y, even(Y)
even(X) ← not odd(X) }

odd(s(s(s(0)))
not odd(s(s(0)))



Imperial College London

Top-directed abductive learning: Summary

- Reuse of existing abductive proof procedures,
- Can support definition of meta-integrity constraints on the language bias

✓Able to learn:

- normal programs (with NAF)
- non-observed concepts
- recursive and connected theories

✓ Sound and Completeness with respect to 3-valued completion semantics

Imperial College London

Collaborators…

Krysia Broda

Oliver Ray

Tim Kimber

Domenico Corapi

Dalal Alrajeh

Katsumi Inoue

Mark Law

Piotr Chabierski

Sebastian Uchitel

Naranker Dulay

Jeff Kramer

Imperial College London

Questions?

Imperial College
London