Hybrid Abductive Inductive Learning

- Combining bottom-up and top-down search
  - » *Observation predicate learning (OPL)*
  - » *Non-observation predicate learning (NOPL)*
  - » *Progol5 and incompleteness*
  - » *Abduction and Induction Cycle*

- Hybrid Abductive Inductive Learning (HAIL)
  - » *Generalised Bottom Set Semantics*
  - » *Algorithm*
  - » *Examples*

Inverse Entailment

$$B, H \models E \quad \text{iff} \quad B, \neg E \models \neg H$$

Assumes H and E to be single Horn clauses:

(h) $l_1 \vee \neg l_2 \vee \ldots \vee \neg l_n$ $\quad$ ($\neg$h) $\neg l_1\theta \wedge l_2\theta \wedge \ldots \wedge l_n\theta$

(e^+) $a_1 \vee \neg a_2 \vee \ldots \vee \neg a_m$ $\quad$ ($\neg e^+$) $\neg a_1\partial \wedge a_2\partial \wedge \ldots \wedge a_m\partial$

Sets of Skolemised ground literals

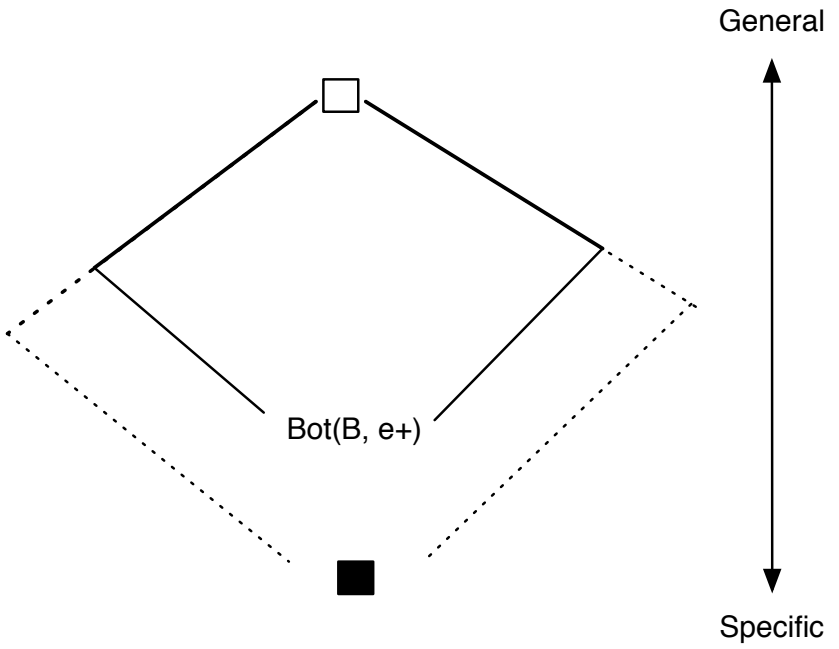
1. $B, \neg e^+ \models \underbrace{\neg l_1\partial \wedge l_2\partial \wedge \ldots \wedge l_n\partial}_{\neg Bot(B, e^+)}$

Let's denote $Bot(B, e^+)$ as $ground(h_\perp)$

$B, ground(h_\perp) \models e^+$

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

h θ-subsumes $Bot(B, e^+)$



An Example

B

animal(X) ← pet(X)
pet(X) ← dog(X)

e^+

nice (X) ← dog(X)

h

????

$\neg e^+ = dog(sk) \wedge \neg nice(sk)$

$B, \neg e^+ \models pet(sk) \wedge dog(sk) \wedge \neg nice(sk) \wedge animal(sk)$

ground($\neg h_\perp$)

$\neg Bot(B, e^+) =$ pet(sk) ∧ dog(sk) ∧ ¬ nice(sk) ∧ animal(sk)

$Bot(B, e^+) =$ ¬pet(sk) ∨ ¬dog(sk) ∨ nice(sk) ∨ ¬animal(sk)

ground($h_\perp$)

$h_\perp = \{nice(X) \leftarrow pet(X), dog(X), animal(X)\}$

$h_\perp \models Bot(B, e^+)$

h θ-subsumes $Bot(B, e^+)$

Language Bias

It defines the language of the hypothesis. It is composed of a set of mode declarations.

Mode declarations: modeh(r, s)
modeb(r, s)

scheme is a ground atom with predicate name and placemarkers +t, -t, #t for unary type t

r is a recall, an integer that indicates how many times the predicate may appear in a rule

modeh indicates the predicate may appear as head predicate of the rule to learn.
modeb indicates the predicate may apperar as body predicate

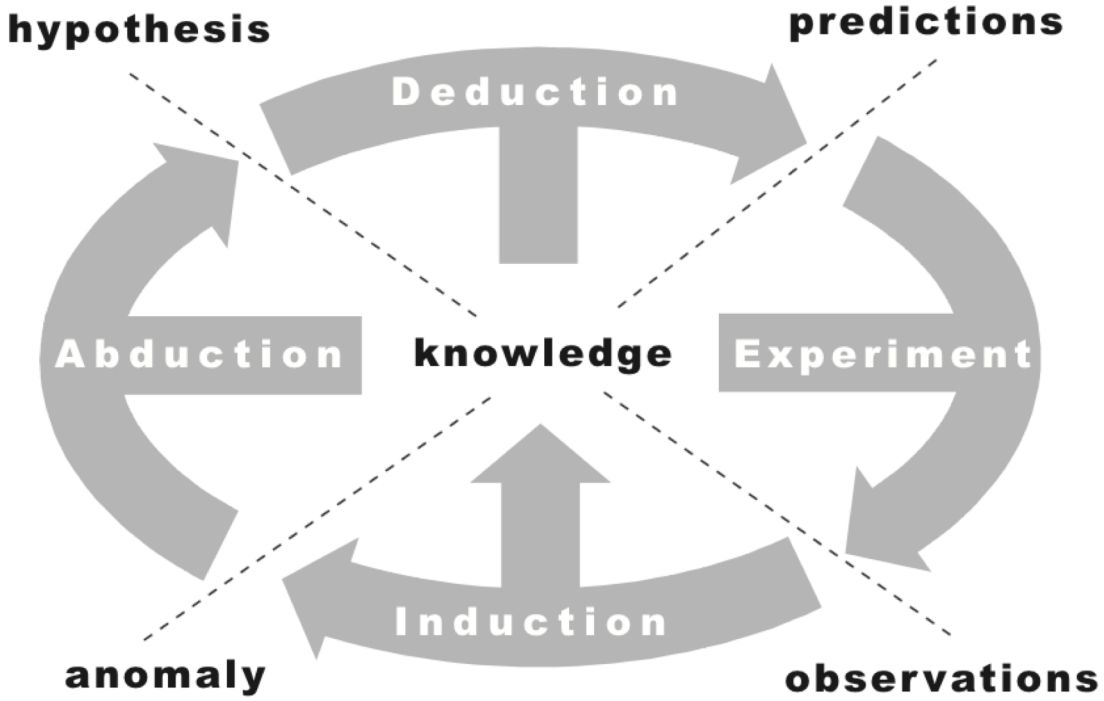
grandfather(X, Y) ← father(X, Z),
parent(Z,Y)

modeh(1, grandfather(+p,+p))
modeb(1, father(+p, -p))
modeb(1, parent(+p, +p))

where p means person

Abduction – Induction Cycle

Peirce Inferential Theory



Abduction extends the theory with explanations, induction extends the given theory to a new theory, which can provide new observable consequences.

NON-Observation Predicate Learning

Observation Predicate Learning (OPL)

- Hypothesis and Examples define the same predicate

Training Examples	Background	Hypothesis

vp(1, 3) + "She ran quickly"
vp(0, 1) -
……
	np(0, 1);
	vp(1, 2);
	mod(2, 3)
		vp(Start, End) ← vp(Start, Middle), mod(Middle, End).

Non-Observation Predicate Learning (OPL)

- Hypothesis and Examples define different predicates

Training Examples	Background	Hypothesis
"The nasty man hit the dog" s(0, 6) +	s(Start, End) ← np(Start, Middle), vp(Middle, End) np(Start, End) ← det(Start, Middle), noun(Middle, End) det(0, 1); noun(1, 2); ……	np(Start, End) ← det(Start, Middle1), adj(Middle1,Middle 2), noun(Middle2, End).

Inverse Entailment and non-OPL

What about this example?

B

hasbeak(X) ← bird (X)
bird(X) ← vulture (X)

e^+

hasbeak(tweety)

h

????

$\neg e^+ = \neg$ hasbeak(tweety)

B , $\neg e^+ \models \neg$hasbeak(tweety) $\wedge \neg$bird(tweety) $\wedge \neg$vulture(tweety)

$Bot(B, e^+)$ = hasbeak(tweety) $\vee$ bird(tweety) $\vee$ vulture(tweety)

$ground(h_\perp)$ = { hasbeak(tweety) }

$h_\perp$ = { hasbeak(X) }

Computable using resolution for FOL but not SLD for definite clauses

Inverse Entailment and non-OPL

B

hasbeak(X) ← bird (X)
bird(X) ← vulture (X)

e^+

hasbeak(tweety)

h

h_1= bird(X)
h_2 = vulture(X)

$\neg e^+ = \neg$ hasbeak(tweety)

B , $\neg e^+ \models \neg$hasbeak(tweety) $\wedge \neg$bird(tweety) $\wedge \neg$vulture(tweety)

$Bot(B, e^+)$ = bird(tweety) $\vee$ vulture(tweety) $\vee$ hasbeak(tweety)

B

hasbeak(X) ← bird (X)
bird(X) ← vulture (X)
non_bird(X) ← non_hasbeak(X)
non_vulture(X) ← non_bird(X)

e^+

hasbeak(tweety)

B, $\neg e^+ \vdash_{SLD}$ non_hasbeak(tweety) $\wedge$ non_bird(tweety) $\wedge$ non_vulture(tweety)

$Bot(B, e^+)$ = hasbeak(twetty) $\vee$ bird(tweety) $\vee$ vulture(tweety)

Progol5

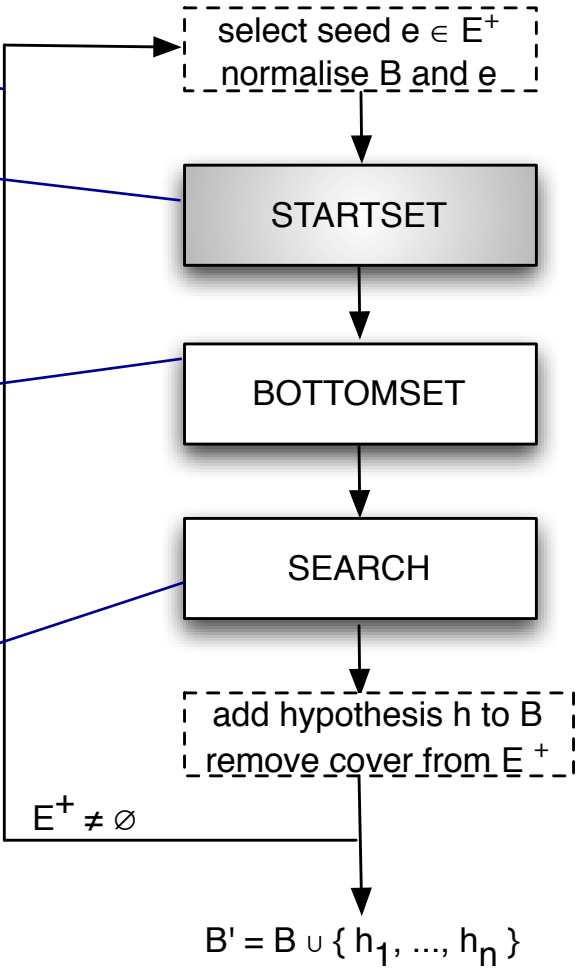
Proof procedure for a given learning task <B, E^+, E^-, M>:

Covering loop

Compute a ground head atom a in Bot(B, e) that is subsumed by some atom in modeh in L_M

Compute a set of ground body atoms $b_1, \ldots, b_n$ in Bot(B, e) such that $a \leftarrow b_1, \ldots, b_n$ is subsumed by some clause in L_M

Compute compressed hypothesis h in L_M that subsumes $a \leftarrow b_1, \ldots, b_n$



Progol5 - Example

B

meal(X) ← fries(X), burger(X)
burger(X) ← fries(X), offer(X)
offer(mD)
offer(bK)
burger(mD)
burger(tR)

E^+

meal(mD)
meal(bK)

M

modeh(*, fries(+))
modeb(*, offer(+))

E^-

← meal(tR)
← burger(X), vegan(X)

meal(X) ← fries(X), burger(X)
non_fries(X) ← non_meal(X), burger(X)
non_burger(X) ← non_meal(X), fries(X)
burger(X) ← fries(X), offer(X)
non_fries(X) ← non_burger(X), offer(X)
non_offer(X) ← non_burger(X), fries(X)
offer(mD)
offer(bK)
burger(mD)
burger(tR)
non_meal(mD)

← non_fries(X)

Can you generate the SLD tree derivation?

Can you generate a lattice of θ-subsumption?

H is fries(X) ← offer(X)

Incompleteness of StartSet

The StartSet procedure is incomplete with respect to the computation of positive literals in the Bottom Set.

B

a ← b, c
b ← c

E^+

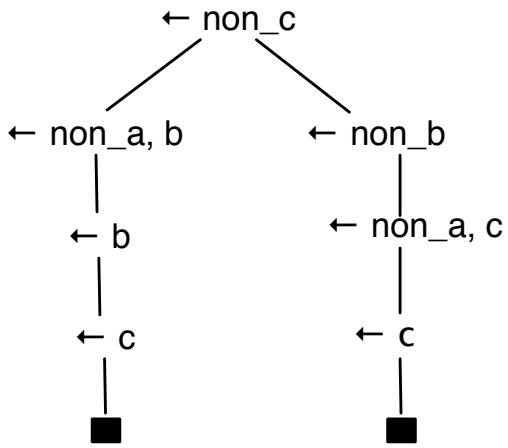
a

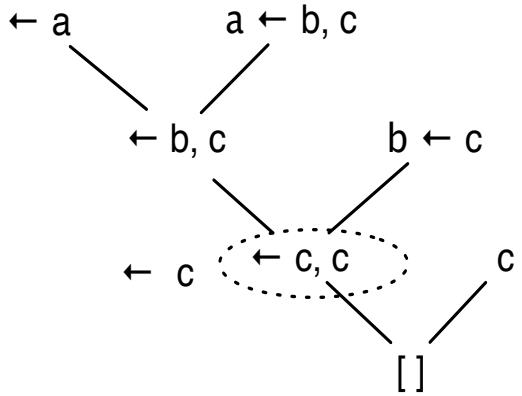
M

modeh(*, c)

B'

a ← b, c
b ← c
non_b ← non_a, c
non_c ← non_a, b
non_c ← non_b
non_a





c in Bot(B, e) BUT B' ∪ {non_e} ⊬ non_c

Generalised Bottom Set

$$\text{Bot}(B, e) = \{lg \mid B \wedge \neg e^{+} \models \neg lg\}$$
$$= \{a \mid B \wedge \neg e^{+} \models \neg a\} \cup \{\neg b \mid B \wedge \neg e^{+} \models b\}$$
$$= \wedge\{b \mid B \wedge \neg e^{+} \models b\} \rightarrow \vee\{a \mid B \wedge a \models e^{+}\}$$

StartSet

Abduction

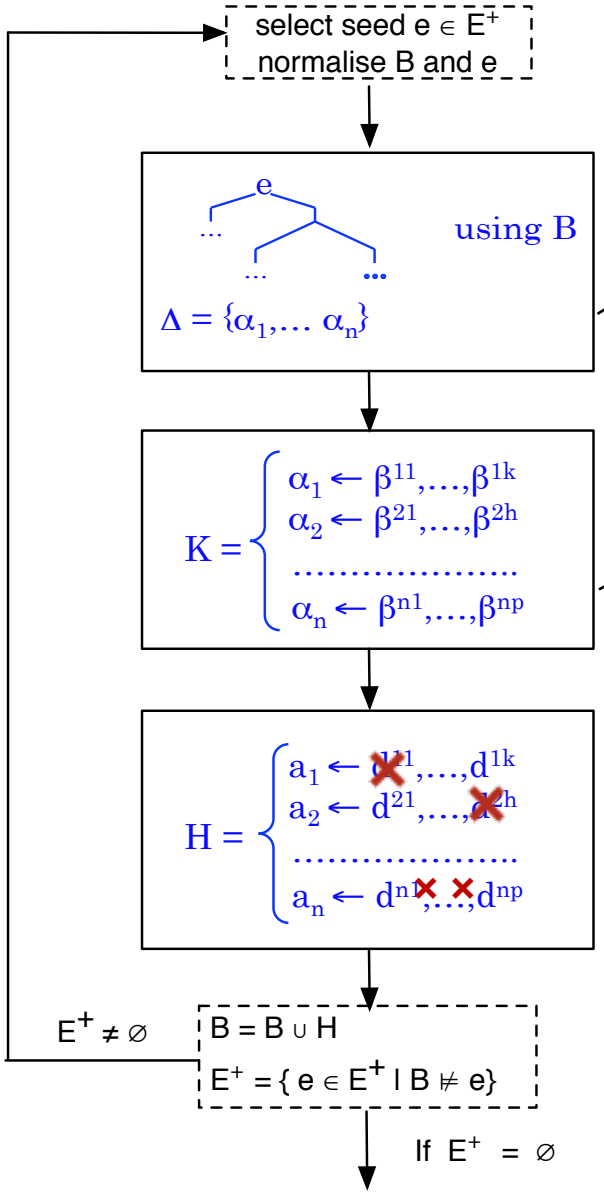
$$\text{Kernel}(B, e) = \{\Delta \mid B \cup \Delta \models e^{+}\} \cup \{\neg b \mid B \cup \neg e^{+} \models b\}$$
$$= \wedge\{b \mid B \cup \neg e^{+} \models b\} \rightarrow \vee\{\Delta \mid B \cup \Delta \models e\}$$

$$K = \begin{cases} a_1 \leftarrow b_{11}, b_{21}, \ldots, b_{n1} \\ a_2 \leftarrow b_{12}, b_{22}, \ldots, b_{m2} \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_k \leftarrow b_{1k}, b_{2k}, \ldots, b_{hk} \end{cases}$$

Hypothesis is a set of clauses that subsumes K.

$H \succcurlyeq K$

HAIL Algorithm



Abduction

<B, Ab, IC>
where Ab = { hθ| modeh(_, h( ))}
IC possibly empty.
$\Delta = \{\alpha_1, \ldots, \alpha_n\} \subseteq Ab$

Deduction

$B \cup \neg e \vdash \beta^{ij}$ for any modeb(_, β( ))

Induction

Definition

A theory H can be learned by *Kernel Set Subsumption* iff there exists a Kernel Set K of B and e such that $H \succcurlyeq K$

Soundness and Completeness

H is learned by KSS iff $B \cup H \models e$

An Example

B

meal(X) ← burger(X), fries(X)
burger(X) ← offer(X), fries(X)
offer(mD) bistro(mD)
offer(bK) bistro(tR)
burger(tR) bistro(bK)

E+

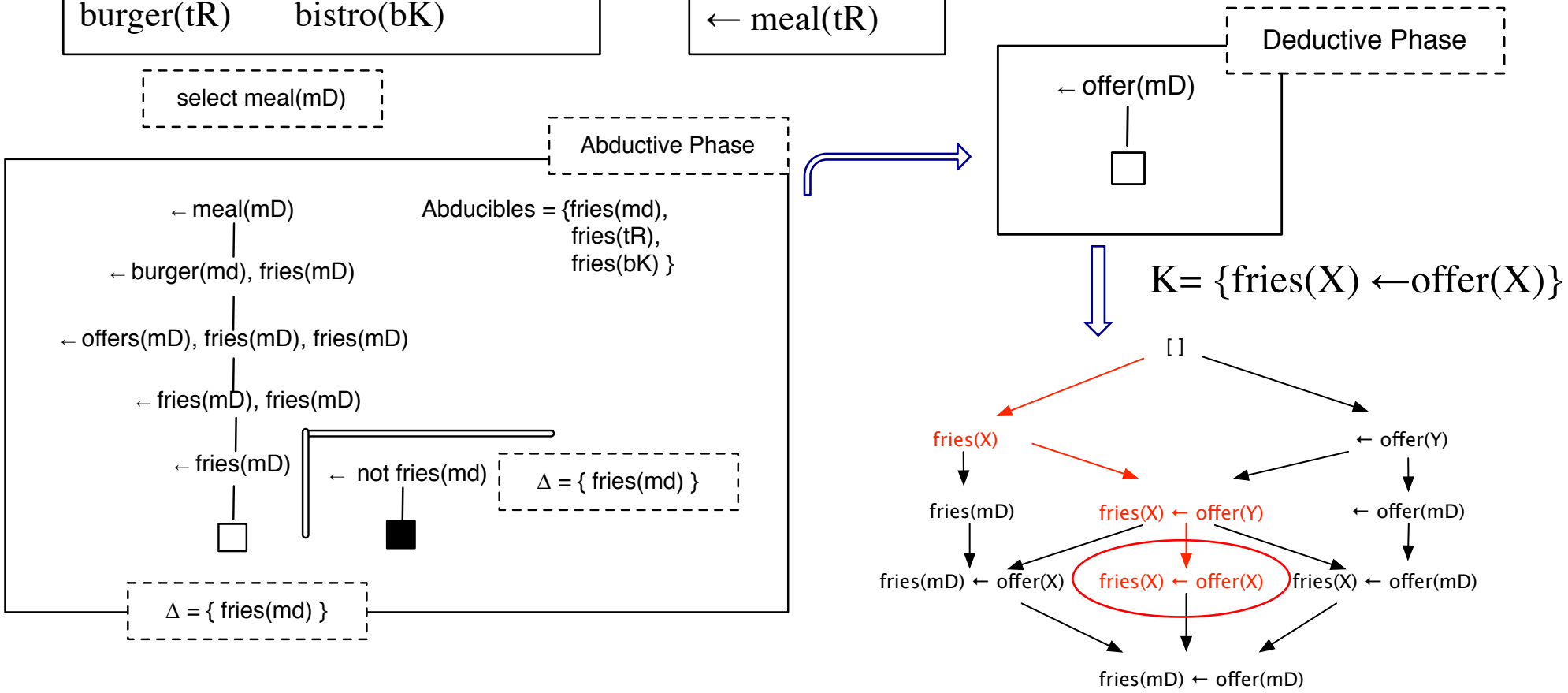
meal(mD)
meal(bK)

E-

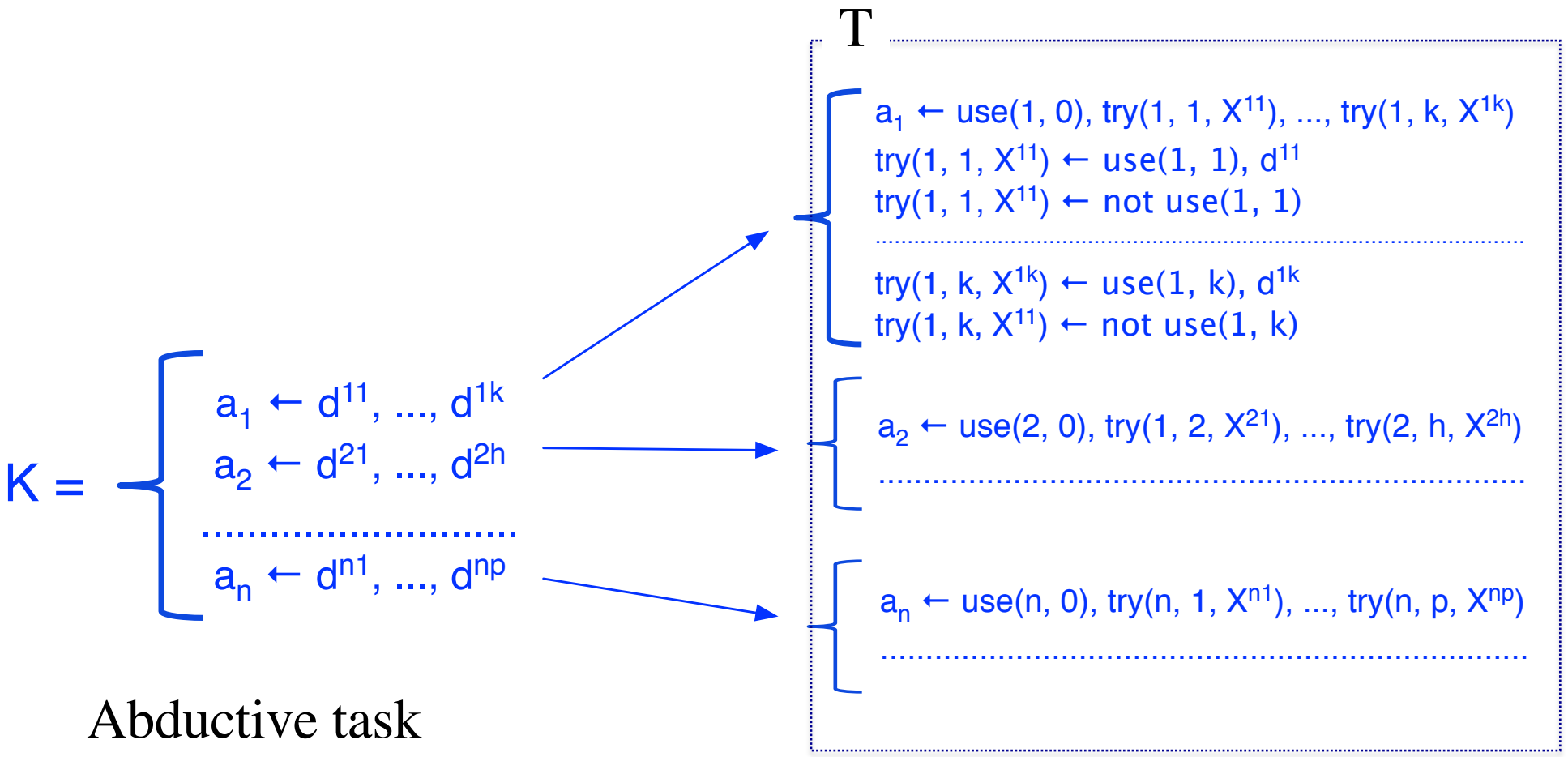
← meal(tR)

M

modeh(*, fries(+bistro))
modeb(*, offer(+bistro))



Using Abduction in the Induction Step



Abductive task

$U = < B \cup T, \{use\}, \emptyset >$

Abductive solution = {use(…), …} $\Longrightarrow$ $H \geqslant K$

$B \cup H \models E$

Using Abduction in the Induction Step

$K = \{ \text{fries}(X) \leftarrow \text{offer}(X)$

$T = \begin{array}{l} \text{fries}(X) \leftarrow \text{use}(1,0), \text{try}(1,1,X) \\ \text{try}(1,1,X) \leftarrow \text{not use}(1,1) \\ \text{try}(1,1,X) \leftarrow \text{use}(1,1), \text{offer}(X) \end{array}$

Abductive task: $< B \cup T, \{\text{use}(1,0), \text{use}(1,1)\}, \emptyset >$

Finds abductive solution $\Delta \subseteq \{\text{use}(1,0), \text{use}(1,1)\}$ such that $B \cup T \cup \Delta \models E$.

$\Delta = \{\text{use}(1,0), \text{use}(1,1)\}$

$\Downarrow$

$H = \{\text{fries}(X) \leftarrow \text{offer}(X)\}$

$\Delta = \{\text{use}(1,0), \text{not use}(1,1)\}$
would give $H = \text{fries}(X)$
and $B \cup T \cup \Delta \models \text{meal}(tR)$

Abduction – Induction Cycle Revisited

Hybrid Abductive Inductive Learning is an approach that truly combines abduction and induction to support knowledge extraction.

- Abduction makes assumptions on relevant missing information.
- Deduction helps constructing the Kernel Set of clauses that bridge the background knowledge with the abductive explanations.
- Induction is the process of generalising: looking for alternative hypothesis that would still cover the examples, what we expect to be true or to be false.

  Abduction could also be used to help searching for most compressed hypothesis in a subsumption lattice

Comparing approaches

HAIL ⩾ Alecto ⩾ Progol5 ⩾ Aleph ⩾ Progol

	Rules using Non-OP	Rules for Non-OP	Multiple clauses	Fully automated
Progol	✗	✗	✗	✓
Aleph	✓	✗	✗	✓
Progol5	✓	✓	✗	✓
Alecto	✓	✓	✓	✗
Hail	✓	✓	✓	✓

Summary

- Defined
  - OPL and NOPL
  - Inverse Entailment as a mechanism for combining bottom-up/top-down search of hypothesis
  - Language Bias (mode declaration)
- Introduced Bottom Set Generalisation
- Presented Progol5 and its contrapositive mechanism for computing NOPL
- Shown incompleteness of Progol5 with respect to Bottom Set Generalisation.
- Described Hybrid Abductive Inductive Learning as a way for realising Pierce's abduction-induction cycle.