Logic-based Learning

Alessandra Russo and Mark Law

Joint Tutorial
Overview

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

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
... Logic-based Learning in AI

Advantages

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

![Logic-based Learning](image)

Automate scientific process using AI techniques to carry out cycles of scientific experiments. Automatically originate hypotheses that explain observations, devise experiments to test the hypothesis and physically run the experiments.

([Letters to Nature 2003])

**Active molecules**

**Inactive molecules**

**Learning Grammars**

<table>
<thead>
<tr>
<th>Training Examples</th>
<th>Learned knowledge</th>
<th>Prior knowledge</th>
</tr>
</thead>
<tbody>
<tr>
<td>s(0, 3) +</td>
<td>np(X, Y) ← word(“She”, X, Y)</td>
<td>word(“She”, 0, 1)</td>
</tr>
<tr>
<td></td>
<td>mod(X, Y) ← word(quickly, X, Y)</td>
<td>word(quickly, 2, 3)</td>
</tr>
<tr>
<td></td>
<td>s(X, Y) ← np(X, Z), vp(Z, Y)</td>
<td>word(ran, 1, 2)</td>
</tr>
<tr>
<td></td>
<td>vp(X, Y) ← v(X, Y)</td>
<td>← v(1, 3)</td>
</tr>
</tbody>
</table>
... Logic-based Learning in AI

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
An intuitive example

Learn the concept of “verb phrase”

“She ran quickly”

\[
\begin{align*}
\text{e}^+ & \quad vp(1,3) \\
\text{e}^- & \quad vp(0,1) \\
\text{e}^- & \quad vp(0,3)
\end{align*}
\]

Background knowledge \( BK \) \[
\begin{align*}
\text{np}(0, 1). \\
\text{v}(1, 2). \\
\text{mod}(2, 3). \\
s(X,Y) \leftarrow np(X,Z), vp(Z,Y) \\
vp(X,Y) \leftarrow v(X,Y)
\end{align*}
\]

\( BK \cup H \models e^+ \)

\( BK \cup H \not\models e^- \)

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

“She ran quickly”

\[
\begin{align*}
\text{e}^+ & \quad s(0,3) \\
\text{e}^- & \quad s(0,1)
\end{align*}
\]

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).

Empty hypothesis

Hypothesis space

Given specific examples

Search space for computing possible solutions

How do we define a learning task?

How do we search for solutions in a given search space?
Learning Task: informal definition

Given

- Set of *positive examples* (E⁺) and set of *negative examples* (E⁻) in \( L_e \)
- Background knowledge (B) in \( L_B \)
- Set of possible solutions (S_M) in a *language bias* \( L_H \)
- *Covers* relation over \( L_B, L_H \) and \( L_e \)

Find

- Solution \( H \in S_M \) such that:
  - *Covers*(B, H, e) for every \( e \in E^+ \) (H is complete)
  - \( \neg\text{Covers}(B, H, e) \) 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]
Learning Task: informal definition

Given
- Set of positive examples ($E^+$) and set of negative examples ($E^-$) in $L_e$
- Background knowledge (B) in $L_B$
- Set of possible solutions ($S_M$) in a language bias $L_H$
- Covers relation over $L_B$, $L_H$ and $L_e$
- Quality criterion over $L_B$, $L_M$ and $L_e$, 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]
\( \mathcal{L}_B \) 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

\[
\text{Covers}(B, H, e) \iff B \cup H \models e, \text{ where } H \subseteq S_M
\]

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

- \( \text{Covers}(B, H, e) \quad \forall e \in E^+ \)
- \( \neg \text{Covers}(B, H, e) \quad \forall e \in E^- \)
Consider $T_{LFE} = (B, S_M, E^+, E^-)$ task given by:

\[ B = \{ \begin{array}{l}
  \text{parent(ann, mary)} \\
  \text{parent(ann, tom)} \\
  \text{parent(tom, eve)} \\
  \text{parent(tom, ian)} \\
  \text{female(ann)} \\
  \text{female(mary)} \\
  \text{female(eve)} \\
\end{array} \]

\[ S_M = \{ \begin{array}{l}
  h_1 = \text{daughter(X,Y)} \leftarrow \text{female(X)} \\
  h_2 = \text{daughter(X,Y)} \leftarrow \text{parent(Y,X)} \\
  h_3 = \text{daughter(X,Y)} \leftarrow \text{parent(Y,X), female(X)} \\
\end{array} \]

\[ E^+ = \{ \begin{array}{l}
  \text{daughter(mary, ann)} \\
  \text{daughter(eve, tom)} \\
\end{array} \]

\[ E^- = \{ \begin{array}{l}
  \text{daughter(tom, ann)} \\
  \text{daughter(eve, ann)} \\
\end{array} \]

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

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

$B \cup \{h_3\} \models \text{daughter(mary, ann)}$

$B \cup \{h_3\} \models \text{daughter(eve, tom)}$

$B \cup \{h_3\} \models \text{daughter(eve, ann)}$

$B \cup \{h_3\} \not\models \text{daughter(tom, ann)}$
Expressivity of interpretable knowledge

- Progol
- Aleph
- Progol5
- Alecto
- Inthelex

Year:
- 1995
- 1998
- 1999
- 2000
- 2001
- 2003
- 2004
- 2006
- 2008
- 2009
- 2010
- 2011
- 2014

Horn theories
Normal Logic Programs
Datalog
Answer Set Programs + preferences
Answer Set Programs with choice
Our recent advancement...

learning choice rules, hard constraints and preferences

Non-monotonic learning

Expressivity of interpretable knowledge

Year


Horn theories

Datalog

Normal Logic Programs

Answer Set Programs

Answer Set Programs with choice

Meta-abductive method

Abductive method

Abductive method

Meta-abductive method

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

1. $B \cup H \models e^+ \quad \forall e^+ \in E^+$
2. $B \cup H \not\models e^- \quad \forall e^- \in E^-$

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

Empty hypothesis

Hypothesis space $S_M$

Given specific examples

Generality Relation

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

$\neg\text{covers}(B, H_i, e^+) \Rightarrow \neg\text{covers}(B, H_j, e^+)$

$\text{covers}(B, H_j, e^-) \Rightarrow \text{covers}(B, H_i, e^-)$

$H_i$ generalises $H_j$ iff $H_i \not\theta\text{-subsumes} H_j$
Defining the hypothesis space

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

head declaration: \texttt{modeh(s)}

body declaration: \texttt{modeb(s)} \quad s \text{ is a ground atom with one or more}

placemarkers: \texttt{+t}, \texttt{-t}, \texttt{#t}, \texttt{where t denotes a type}

$$M = \begin{cases} 
\text{modeh(grandfather(+p,+p))} \\
\text{modeb(father(+p,-p))} \\
\text{modeb(parent(+p,+p))} \\
\text{where p is type person}
\end{cases}$$

$\text{grandfather(X,Y) ← father(X,Z), parent(Z,Y)}$

Compatible with $M$

$\text{grandfather(X,Y) ← parent(X,Z), father(Z,Y)}$

Not compatible with $M$

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

Use of efficient generalisation operators.
- Plotkin’s least general generalisation
- Muggleton’s inverse resolution (GOLEM, CIGOL,…)

Use of efficient specialisation operators.
- Shapiro’s refinement operators
- Quinlan’s FOIL system

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 $e^+ \in E^+$

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

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

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

1. $B \cup \{\neg e^+\} \models \neg l_1 \land \neg l_2 \land \ldots \land \neg l_n \land \neg \text{Bot}(B,e^+)$
2. $\neg \text{Bot}(B,e^+) \models \neg h \implies h \models \text{Bot}(B,e^+)$

$h$ is derivable by Bottom Generalisation iff $h \theta$-subsumes $\text{Bot}(B,e^+)$
Probol

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

```
select seed $e \in E^+$
normalise $B$ and $e$

\[ \text{BOTTOMSET} \]

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

\[ \text{SEARCH} \]

add hypothesis $h$ to $B$
remove cover from $E^+$

$E^+ \neq \emptyset$

$B' = B \cup \{ h_1, ..., h_n \}$

Not able to support non-observation predicate learning
```
Use *Covering loop*: compute an hypothesis for each seed example $e^+$

*Mode Declarations* $M$ to constrain the computation of the $\text{Bot}(B,e^+)$

- **STARTSET**
  - select seed $e \in E^+$
  - normalise $B$ and $e$

- **BOTTOMSET**
  - Compute $B^*$, by extending $B$ with contrapositives, computes negated (head) atom in $\neg \text{Bot}(B^*, e^+)$ compatible with mode $h$

- **SEARCH**
  - Computes (body) positive atoms in $\neg \text{Bot}(B, e^+)$ compatible with mode $b$
  - Top-down specialisation to compute $h$ that $\theta$-subsumes $\text{Bot}(B, e^+)$

- $E^+ \neq \emptyset$
  - add hypothesis $h$ to $B$
  - remove cover from $E^+$

$B' = B \cup \{ h_1, ..., h_n \}$
Use Covering loop: compute an hypothesis for each seed example $e^+$

Mode Declarations $M$ to constrain the computation of the $\text{Bot}(B, e^+)$

---

**Progol5**

```
B = \begin{cases} 
p(X) \leftarrow q(X) \\
r(a) \end{cases} 
E^+ = \{p(a)\} \quad E^- = \{p(b)\}
M = \{ \text{mode}(q(+\text{any})), \text{mode}(r(+\text{any})) \}
```

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

```
B \cup \neg e^+ \models \{\neg q(a)\} 
B \cup \neg e^+ \models \{r(a)\} 
\neg \text{Bot}(B, e^+) = \{\neg q(a), r(a)\} 
\text{Bot}(B, e^+) = q(a) \leftarrow r(a)
```

---

Imperial College
London
Incompleteness of Progol5

\[ B = \begin{cases} a \leftarrow b, c \\ b \leftarrow c \end{cases} \]

\[ E^+ = \{ a \} \quad h = \{ c \} \]

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

\[ B \cup \{ \neg e^+ \} = B \cup \{ \neg a \} \models \neg a \land \neg c \]

\[ c \in \text{Bot}(B,e) \]

\[ h \text{ } \theta\text{-subsumes } \text{Bot}(B,e^+) \]

\[ E^+ = \{ a \} \quad h = \{ c \} \]

\[ B^* = \begin{cases} a \leftarrow b, c \\ c^* \leftarrow a^*, b \\ b^* \leftarrow a^*, c \\ b \leftarrow c \\ c^* \leftarrow b^* \end{cases} \]

Failed SLD derivation

\[ c \not\in \text{STARTSET}(B, e^+) \]
Generalising Bottom Set

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

Bottom Set
\[
\{ \neg \alpha \mid B \cup \{ \neg e \} \models \neg \alpha \} \cup \{ \neg \beta \mid B \cup \{ \neg e \} \models \beta \}
\]

Abductive explanation
\[
B \cup \{ \alpha_1, \ldots, \alpha_1 \} \models e^+.
\]

Kernel Set
\[
\alpha_1 \leftarrow \beta_1^{11}, \ldots, \beta_1^{1m}
\]
\[
\alpha_i \leftarrow \beta_i^{11}, \ldots, \beta_i^{1h}
\]
\[
\alpha_n \leftarrow \beta_n^{11}, \ldots, \beta_n^{1k}
\]

Deductive consequences
\[
B \models \beta_i^{1j}
\]

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.

\[ \Delta = \{ \alpha_1, \ldots, \alpha_n \} \]

using \( B \) to select seed \( e \in E^+ \)

\[ \begin{align*}
K &= \{ \alpha_1 \leftarrow \beta_1^{11}, \ldots, \beta_1^{1k} \\
&\quad \alpha_2 \leftarrow \beta_2^{21}, \ldots, \beta_2^{2h} \\
&\quad \ldots \\
&\quad \alpha_n \leftarrow \beta_n^{n1}, \ldots, \beta_n^{np} \}
\end{align*} \]

Abduction

\[ H = \{ \begin{align*}
a_1 &\leftarrow d_1^{11}, \ldots, d_1^{1k} \\
a_2 &\leftarrow d_2^{21}, \ldots, d_2^{2h} \\
&\quad \ldots \\
a_n &\leftarrow d_n^{n1}, \ldots, d_n^{np} \}
\end{align*} \]

Deduction

\[ B \vdash \beta_i^j \text{ for any mode}(\beta(.)) \]

Induction

\[ \text{A clausal theory } H \text{ is derivable by Kernel Set Subsumption from } B \text{ and } e \text{ iff } \]

\[ H \theta \text{-subsumes } K(B,e^+) \]

HAIL example

B =

sad(X) ← tired(X), poor(X)
academic(oli)
academic(ale)
academic(kb)
student(oli)
lecturer(ale)
lecturer(kb)

1. Abduction:
   \( \Delta = \{ \text{tired(ale), poor(ale)} \} \)

2. Deduction:
   B \models \{ \text{academic(ale), academic(kb), lecturer(ale), lecturer(kb)} \}
   \[ K_g = \begin{cases} 
   \text{tired(ale)} \leftarrow \text{academic(ale), lecturer(ale)} \\
   \text{poor(ale)} \leftarrow \text{academic(ale), lecturer(ale)} 
   \end{cases} \]

3. H =
   \[ M = \begin{cases} 
   \text{modeh(tired(+academic))} \\
   \text{modeh(poor(+academic))} \\
   \text{modeb(lecturer(+academic))} \\
   \text{modeb(academic\'(+academic))} 
   \end{cases} \]

E⁺ =

\[ \begin{cases} 
   \text{sad(ale)} \\
   \text{sad(kb)} 
   \end{cases} \]

E⁻ =

\[ \begin{cases} 
   \text{sad(oli)} \\
   \text{poor(oli)} 
   \end{cases} \]
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

\[ B = \begin{cases} 
    \text{even}(s(X)) \leftarrow \text{odd}(X) \\
    \text{even}(0) 
\end{cases} \]

\[ M = \begin{cases} 
    \text{modeh}(\text{odd}(s(+\text{any}))) \\
    \text{modeb}(\text{even}(+\text{any})) 
\end{cases} \]

\[ E^+ = \text{odd}(s(s(s(0)))) \]

Can we learn the following concept

\[ H = \begin{cases} 
    \text{odd}(s(X)) \leftarrow \text{even}(X) 
\end{cases} \]

\[ \text{Bot}(B,e^+) = \{\text{odd}(s(s(s(0)))) \leftarrow \text{even}(0)\} \]

\[ \text{H not } \theta\text{-subsumes } \text{Bot}(B,e^+) \]

\[ \text{Kg } (B,e^+) = \{\text{odd}(s(s(s(0)))) \leftarrow \text{even}(0)\} \]

\[ \text{H not } \theta\text{-subsumes } \text{K}(B,e^+) \]

So where is the problem?
Induction on Failure

Yamamoto’s example

\[
B = \begin{cases} 
\text{even}(s(X)) \leftarrow \text{odd}(X) \\
\text{even}(0) 
\end{cases}
\]

\[
M = \begin{cases} 
\text{modeh}(\text{odd}(s(+\text{any}))) \\
\text{modeb}(\text{even}(+\text{any})) 
\end{cases}
\]

\[
E^+ = \begin{cases} 
\text{odd}(s(s(0)))) 
\end{cases}
\]

? \ H = \begin{cases} 
\text{odd}(s(X)) \leftarrow \text{even}(X) 
\end{cases}
Yamamoto’s example

\[ B = \begin{cases} \text{even}(s(X)) & \text{← odd}(X) \\ \text{even}(0) \end{cases} \]

\[ M = \begin{cases} \text{mode}(\text{odd}(s(+\text{any}))) \\ \text{mode}(\text{even}(+\text{any})) \end{cases} \]

\[ E^+ = \{\text{odd}(s(s(0))))\} \]

\[ \text{? } H = \begin{cases} \text{odd}(s(X)) & \text{← even}(X) \end{cases} \]

\[
T_0 = \{\text{odd}(s(s(0)))) & \text{← even}(s(s(0)), \text{even}(0))\}
\]

body atoms proved abductively

\[
T_1 = \{\text{odd}(s(0)))) & \text{← even}(0) \}
\]

\[
T = T_0 \cup T_1
\]
Extending Kernel Sets to Connected Theories

\[ T = T_1 \cup T_2 \cup \ldots \cup T_n \]

A clausal theory \( H \) is derivable by Connected Theory Generalisation from \( B \) and \( e \) iff \( H \theta \)-subsumes \( T \)

1) \( B \cup T_1^+ \models E \)
2) \( B \cup T_j^+ \models T_{j-1}^- \quad 1 < j \leq n-1 \)
3) \( B \models T_n^- \)
IE: semantics generalisations

Bottom Set
Progol5

Kernel Set
Hail

Connected Theory
Imparo

Bot(B, E)

B, \neg E

Deduce

\Delta

Deduce

B, E

Abduce

Bot(B, E)

Kernel(B, E)

Bot(B, E)

Kernel(B, e)

Bot(B, E)

Kernel(B, e)

Bot(B, E)
But, …what about non-monotonic learning?
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

\[
\begin{align*}
B &= \begin{cases}
\text{obeys}(X,Y) \leftarrow \text{not officer}(X), \text{officer}(Y) \\
\text{wears}._\text{hat}(\text{price}) \\
\text{wears}._\text{hat}(\text{osbourne}) \\
\text{has}._\text{stripe}(\text{osbourne})
\end{cases} \\
E^+ &= \begin{cases}
\text{obeys}((\text{prince,osbourne})
\end{cases} \\
M &= \begin{cases}
\text{mode}_{\text{h}}(\text{officer}(+\text{any})) \\
\text{mode}_{\text{b}}(\text{has}._\text{stripe}(+\text{any}))
\end{cases} \\
H_{\text{ground}} &= \begin{cases}
\text{officer}(\text{osbourne}) \leftarrow \text{wears}._\text{hat}(\text{osbourne})
\end{cases} \\
H &= \begin{cases}
\text{officer}(X) \leftarrow \text{wears}._\text{hat}(X)
\end{cases}
\end{align*}
\]
Top-Directed Abductive Learning (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

\[
T_M = \text{Pre-processing}(B, E, S_M)
\]

\[
\Delta = \text{Abduce}(B \cup T_M, \{\text{rule(.)}\}, \emptyset) \text{ with goal E}
\]

\[
H = \text{Post-processing}(\Delta, M)
\]
**TAL: Example**

\[ B = \text{even}(0) \]

\[ M = \begin{align*}
\text{eh:} & \quad \text{modeh(even(+nat))} \\
\text{oh:} & \quad \text{modeh(odd(+nat))} \\
\text{bno:} & \quad \text{modeb(not odd(+nat))} \\
\text{be:} & \quad \text{modeb(even(+nat))} \\
\text{bs:} & \quad \text{modeb(+nat = s(+nat))}
\end{align*} \]

\[ E = \begin{align*}
\text{odd(s(s(s(0))))} \\
\text{not odd(s(s(0)))}
\end{align*} \]

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

\[ H = \{ \text{odd(X) } \leftarrow \text{s(X) = Y, even(Y)} \\
\text{even(X) } \leftarrow \text{not odd(X)} \} \]
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
Collaborators...

Krysia Broda
Oliver Ray
Tim Kimber
Domenico Corapi

Dalal Alrajeh
Katsumi Inoue
Mark Law

Piotr Chabierski

Sebastian Uchitel
Naranker Dulay
Jeff Kramer
Questions?