Logic-based Learning of Answer Set Programs

Mark Law and Alessandra Russo

mark.law09@imperial.ac.uk
a.russo@imperial.ac.uk

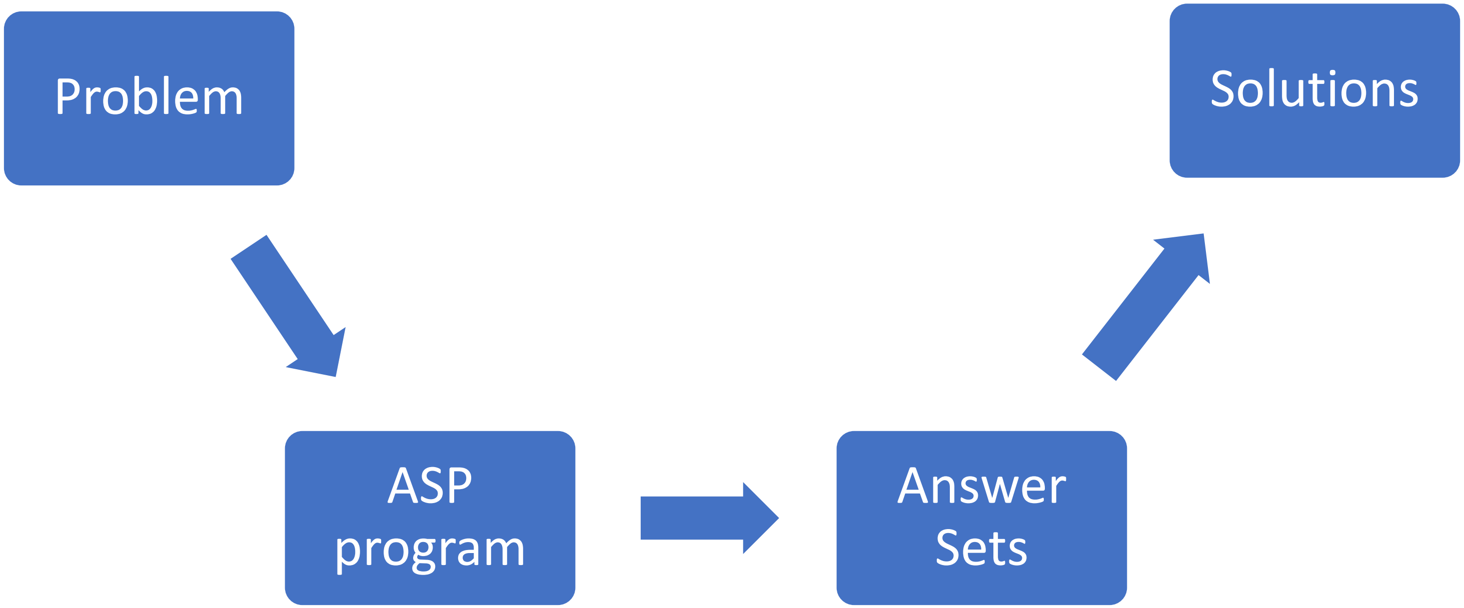
Structure

- Answer Set Programming (ASP)
  - Very brief overview of the Answer Set Programming
  - Brave and cautious entailment

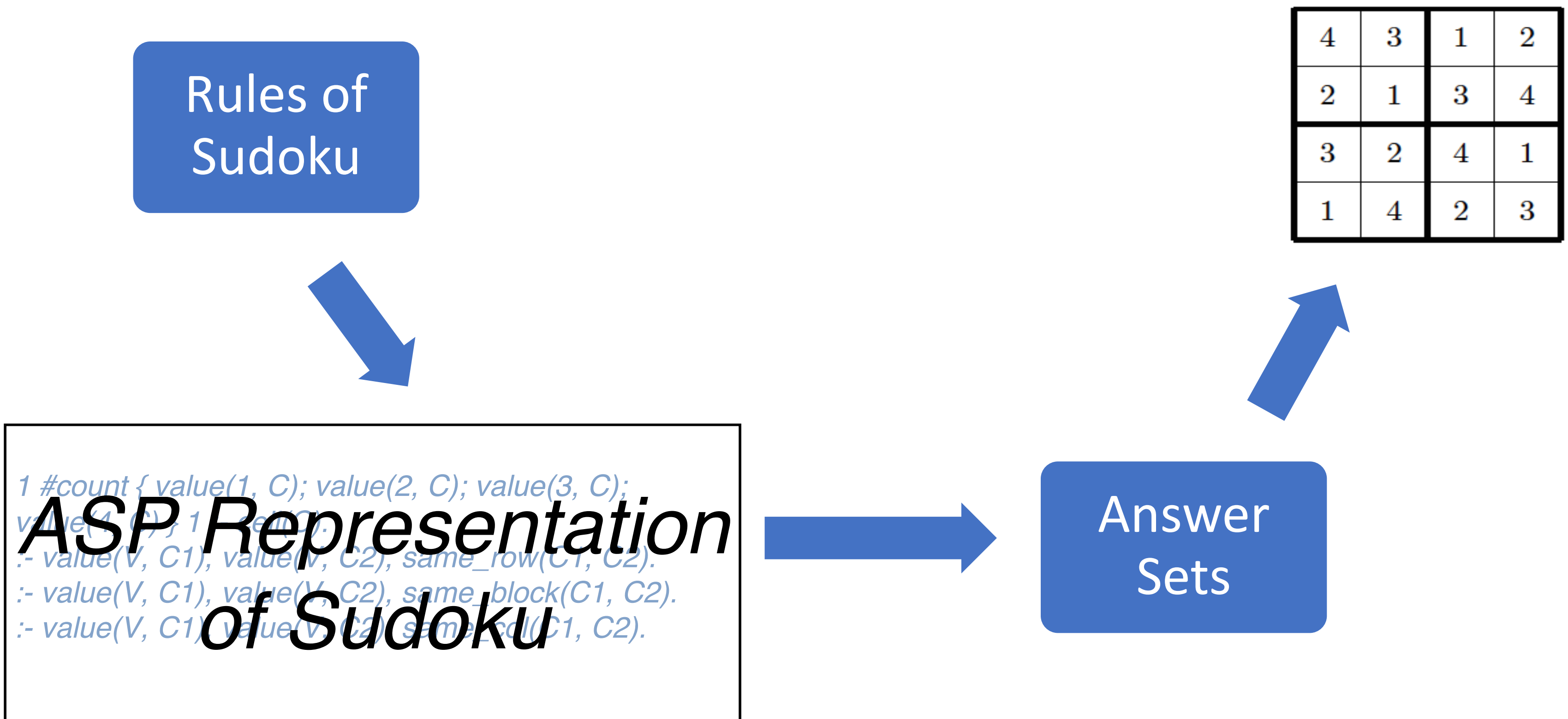
- Initial approaches to learning in ASP
  - Brave and cautious induction in ASP
  - The algorithm of ASPAL
  - Limitations of brave and cautious induction

- Learning from Answer Sets (LAS) and ILASP
  - Relationship to other learning approaches
  - Context-dependent examples
  - Preference learning in ASP
  - Learning from noisy examples
  - The ILASP Algorithm for computing the optimal solutions of any LAS task

Answer Set Programming



Answer Set Programming



Syntax: Normal rules

Let $\texttt{h, b}_1\texttt{, ..., b}_m\texttt{, c}_1\texttt{, ..., c}_n$ be atoms. A *normal rule* is of the form:

$$\underbrace{\texttt{h}}_{\text{Head}}\ \texttt{:-}\ \underbrace{\texttt{b}_1\texttt{, ..., b}_m\texttt{, not c}_1\texttt{, ..., not c}_n}_{\text{Body}}\texttt{.}$$

If the body of the rule is satisfied then its head must also be satisfied.

Syntax: Choice rules

Let $h_1, \ldots, h_k, b_1, \ldots, b_m, c_1, \ldots, c_n$ be atoms. A *choice rule* is of the form:

$$\underbrace{\texttt{lb}\ \{h_1;\ \ldots;\ h_k\}\ \texttt{ub}}_{\text{Head}}\ \texttt{:-}\ \underbrace{b_1,\ \ldots,\ b_m,\ \texttt{not}\ c_1,\ \ldots,\ \texttt{not}\ c_n.}_{\text{Body}}$$

If the body of the choice rule is satisfied then between `lb` and `ub` of $\{h_1, \ldots, h_k\}$ must be satisfied.

Syntax: Constraints

Let $b_1, \ldots, b_m, c_1, \ldots, c_n$ be atoms. A *constraint* is of the form:

$$:- \underbrace{b_1, \ldots, b_m, \text{not } c_1, \ldots, \text{not } c_n}_{\text{Body}}.$$

The body of the constraint must not be satisfied. Constraints are used to filter out unwanted answer sets.

Answer Sets and Entailment

The *answer sets* of a program are a special subset of its Herbrand models.

An atom *A* is *bravely entailed* by a program *P* if it is true in *at least one* answer set of *P (written* $P \models_b A$*)*.

An atom A is *cautiously entailed* if it is true in *every* answer set of *P (written* $P \models_c A$*)*.

```
1 {p ; q} 1.
r.
```

$\models_b$ p

$\models_b$ q

$\models_b$ r

$\models_c$ r

Abduction in ASP

Consider the abductive task:

B

```
wobblyWheel :- brokenSpokes.
wobblyWheel :- flatTyre.
flatTyre :- leakyValve.
flatTyre :- puncturedTube.
```

IC

```
:- not puncturedTube, leakyValve.
```

O

```
wobblyWheel
```

Ab

```
brokenSpokes
puncturedTube
leakyValve
```

How could we represent this in ASP?

```
0 { brokenSpokes; puncturedTube; leakyValve } 3.
wobblyWheel :- brokenSpokes
wobblyWheel :- flatTyre
flatTyre :- leakyValve
flatTyre :- puncturedTube.
:- not puncturedTube, leakyValve.
:- not wobblyWheel.
```

Semantics of Weak Constraints

Weak constraints represent preferences in ASP.

$$\texttt{:}\!\sim\ \underbrace{\texttt{bought(Item), cost(Item, M).}}_{\text{Body}}\ \overbrace{\texttt{[}\underbrace{\texttt{M}}_{\text{Weight}}\texttt{@}\underbrace{\texttt{1}}_{\text{Priority Level}}\texttt{, }\underbrace{\texttt{Item, M}}_{\text{Terms}}\texttt{]}}^{\text{Tail}}$$

For any program *P* and answer set *A*, *Weak(P, A)* is the set of all (unique) *tails* of weak constraints in *ground(P)* whose body is satisfied by *A*.

The aim is to minimise the sum $\sum_{[wt@lev,t_1,...t_n]\in Weak(P,A)} wt$.

High priority levels are more important than low priority levels.

Journey Preference Example

Avoid walking through areas with a high crime rating

Minimise the number of buses

Minimise the distance walked

```
:~ mode(Leg, walk), crime_rating(Leg, C), C > 4 . [1@3, Leg]
:~ mode(Leg, bus) . [1@2, Leg]
:~ mode(Leg, walk), distance(Leg, Distance) . [Distance@1, Leg]
```

Journey A	Journey B	Journey C	Journey D
• Walk 400m through an area with crime rating of 2. • Take the bus 3km through an area with crime rating 4.	• Take the bus 4km through an area with crime rating of 2 • Walk 1km through an area with crime rating 5.	• Take the bus 400m through an area with crime rating of 2. • Take a second bus 3km through an area with crime rating 4	• Take a bus 2km through an area with crime rating 5. • Walk 2km through an area with crime rating 1.

What is the ordering of the 4 journeys?

Journey A > Journey D > Journey C > Journey B

Logic-based Learning under the Answer Set Semantics

Induction for definite programs

Standard setting for ILP:

- Background knowledge *B* a definite program
- Positive examples E^+ atoms
- Negative examples E^- atoms
- Find a hypothesis *H* such that:

$$\forall e^+ \in E^+ : B \cup H \models e^+$$
$$\forall e^- \in E^- : B \cup H \not\models e^-$$

Cautious Induction

Cautious setting for ILP under the Answer Set semantics:

- Background knowledge *B* an ASP program
- Positive and negative examples E^+ and E^- (atoms)
- Find a hypothesis *H* such that:
  - *B* U *H* is satisfiable (has at least one Answer Set)
  - for all Answer Sets *A* of *B* U *H* :

$$\forall e^+ \in E^+ : e^+ \in A$$

$$\forall e^- \in E^- : e^- \notin A$$

Cautious Induction : Example

B

```
p :- not q.
q :- not p, not r.
s :- r.
```

E^+

```
s
```

E^-

```
q
```

Which of the following hypotheses are cautious inductive solutions?

`s.`	`:- not s.` `:- q.`	`p.` `s.`	`r.`

Cautious Induction : Example

B

```
p :- not q.
q :- not p, not r.
s :- r.
```

E^+

```
s
```

E^-

```
q
```

Which of the following hypotheses are cautious inductive solutions?

s.	:- not s. :- q.	p. s.	r.
{ p, s }, { q, s }	NONE!	{ p, s }	{p, r, s }

Cautious Induction : Example

B

```
p :- not q.
q :- not p, not r.
s :- r.
```

E^+

```
s
```

E^-

```
q
```

Which of the following hypotheses are cautious inductive solutions?

s.	:- not s. :- q.	p. s.	r.
{ p, s }, { q, s }	NONE!	{ p, s }	{p, r, s }
✗	✗	✓	✓

Cautious Induction : Limitations

What examples could we give to learn the program:

```
1 { value(C, heads); value(C, tails) } 1 :- coin(C).
coin(c1).
```

Brave Induction

Brave setting for ILP under the Answer Set semantics:

- Background knowledge *B* an ASP program
- Positive and negative examples E^+ and E^- (atoms)
- Find a hypothesis *H* such that:
  - there is *at least one* Answer Set *A* of *B* U *H* :

$$\forall e^+ \in E^+ : e^+ \in A$$

$$\forall e^- \in E^- : e^- \notin A$$

Brave Induction : Example

B

```
p :- not q.
q :- not p, not r.
s :- r.
```

E^+

```
s
```

E^-

```
q
```

Which of the following hypotheses are brave inductive solutions?

s.	:- not s. :- q.	p. s.	r.
{ p, s }, { q, s }	NONE!	{ p, s }	{p, r, s }

Brave Induction : Example

B

```
p :- not q.
q :- not p, not r.
s :- r.
```

E^+

```
s
```

E^-

```
q
```

Which of the following hypotheses are brave inductive solutions?

`s.`	`:- not s.` `:- q.`	`p.` `s.`	`r.`
`{ p, s },` `{ q, s }`	NONE!	`{ p, s }`	`{p, r, s }`
✓	✗	✓	✓

Implementations

Two of the main non-monotonic ILP algorithms compute the solutions to brave induction tasks:

- XHAIL (Ray 09)
  - An extension of the HAIL algorithm.
  - ILED (Katzouris, Artikis, Paliouras, 2015) and Inspire (Kamzi, Schüller, Saygın, 2017) are extensions of XHAIL.
- ASPAL (Corapi, Russo, Lupu 2011)
  - Encodes of a brave ILP task into an ASP program.
  - RASPAL (Athakravi, Corapi, Broda, Russo) is an extension of ASPAL.
- We will only have time to cover ASPAL in this tutorial.

ASPAL: Skeleton rules

A skeleton rule for mode declarations <M_h, M_b> is a compatible rule where all the constants placemarkers are replaced with different variables instead of constants.

M_h, M_b

```
modeh(penguin(+bird))
modeb(not can(+bird, #ability))
```

B

```
bird(a).          bird(b).
ability(fly).     ability(swim).
can(a, fly).      can(b, swim).
```

The rule `penguin(V1) :- bird(V1), not can(V1, C1).` represents:

```
penguin(V1) :- bird(V1), not can(V1, fly).
penguin(V1) :- bird(V1), not can(V1, swim).
```

ASPAL: Skeleton Rules

M_h, M_b

```
modeh(penguin(+bird))
modeb(not can(+bird, #ability))
```

B

```
bird(a).          bird(b).
ability(fly).     ability(swim).
can(a, fly).      can(b, swim).
```

$L_{max} = 2$, $V_{max} = 1$

L_{max} is the maximum number of literals allowed to appear in the body (not including atoms used to enforce types).

V_{max} is the maximum number of variables.

Sk_M

```
penguin(V1) :- bird(V1).
penguin(V1) :- bird(V1), not can(V1, C1).
penguin(V1) :- bird(V1), not can(V1, C1), not can(V1, C2).
```

ASPAL: ASP encoding

Given S_M, B, E^+, and E^-, we can encode the search for inductive solutions as an ASP program.

We assign each rule `R` in $\mathtt{Sk_M}$ a unique identifier $\mathtt{R_{ID}}$.

Each `R` in $\mathtt{Sk_M}$ is associated with a meta level atom $\mathtt{rule(R_{ID},\ C1,\ \ldots,\ Cn)}$, called $\mathtt{R_{meta}}$. The ground instances of these atoms represent rules in $\mathtt{S_M}$.

e.g. Given the skeleton rule `p(V1, V2) :- q(V1, C1), r(V2, C2)` with ID `2`, the atom `rule(2, a, b)` represents:

```
p(V1, V2) :- q(V1, a), r(V2, b).
```

The goal is to find these atoms using ASP.

ASPAL: ASP encoding example

B

```
bird(a).          bird(b).
ability(fly).     ability(swim).
can(a, fly).      can(b, swim).
```

E^+

```
penguin(b)
```

E^-

```
penguin(a)
```

```
% Background
bird(a). bird(b).
ability(fly). ability(swim).
can(a, fly). can(b, swim).
```

```
% Examples
goal :- penguin(b), not penguin(a).
:- not goal.
```

```
% Skeleton Rules
penguin(V1) :- bird(V1), rule(1).
penguin(V1) :- bird(V1), not can(V1, C1), rule(2, C1).
penguin(V1) :- bird(V1), not can(V1, C1), not can(V1, C2), rule(3, C1, C2).

% Generate Hypotheses

0 {rule(1);  rule(2, fly);  rule(2, swim); rule(3, fly, swim) } 4.
```

ASPAL: ASP encoding

Given S_M, B, E^+, and E^-, we can encode the search for inductive solutions as an ASP program.

Definition 4. *Let T be the ILP_b task $\langle B, S_M, \langle\{\mathtt{e}_1^+, \ldots, \mathtt{e}_n^+\}, \{\mathtt{e}_1^-, \ldots, \mathtt{e}_m^-\}\rangle\rangle$, where S_M is characterised by the set of mode declarations M. Let Sk_M be the set of skeleton rules derivable from M. The ASPAL meta-representation is the program consisting of the following components:*

- B
- $\mathtt{h}$:- $\mathtt{b}_1, \ldots \mathtt{b}_{\mathtt{rl}}, \mathtt{R}_{\mathtt{meta}}$, *for each rule* $R \in Sk_M$*, where* R *is the rule* $\mathtt{h}$:- $\mathtt{b}_1, \ldots, \mathtt{b}_{\mathtt{rl}}$.
- *A choice rule* $0\{\mathtt{ab}_1, \ldots, \mathtt{ab}_\mathtt{k}\}\mathtt{k}.$*, where* $\{\mathtt{ab}_1, \ldots, \mathtt{ab}_\mathtt{k}\} = \{R_{meta} \mid R \in Sk_M\}$
- *The rule* goal :- $\mathtt{e}_1^+, \ldots, \mathtt{e}_n^+$, not $\mathtt{e}_1^-, \ldots,$ not $\mathtt{e}_m^-$.
- *The constraint* :- not goal.

If we add a weak constraint `:~` $\mathtt{R}_{\mathtt{meta}}$`.[|R|@1,` $\mathtt{R}_{\mathtt{meta}}$`]` for each R in Sk_M, the optimal answer sets represent the optimal solutions of *T*.

ASPAL: ASP encoding example

B

```
bird(a).        bird(b).
ability(fly).   ability(swim).
can(a, fly).    can(b, swim).
```

E^+

```
penguin(b)
```

E^-

```
penguin(a)
```

```
% Background
bird(a). bird(b).
ability(fly). ability(swim).
can(a, fly). can(b, swim).
```

```
% Examples
goal :- penguin(b), not penguin(a).
:- not goal.
```

```
% Skeleton Rules
penguin(V1) :- bird(V1), rule(1).
penguin(V1) :- bird(V1), not can(V1, C1), rule(2, C1).
penguin(V1) :- bird(V1), not can(V1, C1), not can(V1, C2), rule(3, C1, C2).
```

```
% Generate Hypotheses

0 {rule(1);  rule(2, fly);  rule(2, swim); rule(3, fly, swim) } 4.
:~ rule(1).[1@1, 1]
:~ rule(2, C1).[2@1, 2, C1]
:~ rule(3, C1, C2).[3@1, 3, C1, C2]
```

Brave Induction : Limitations

Consider a background knowledge:

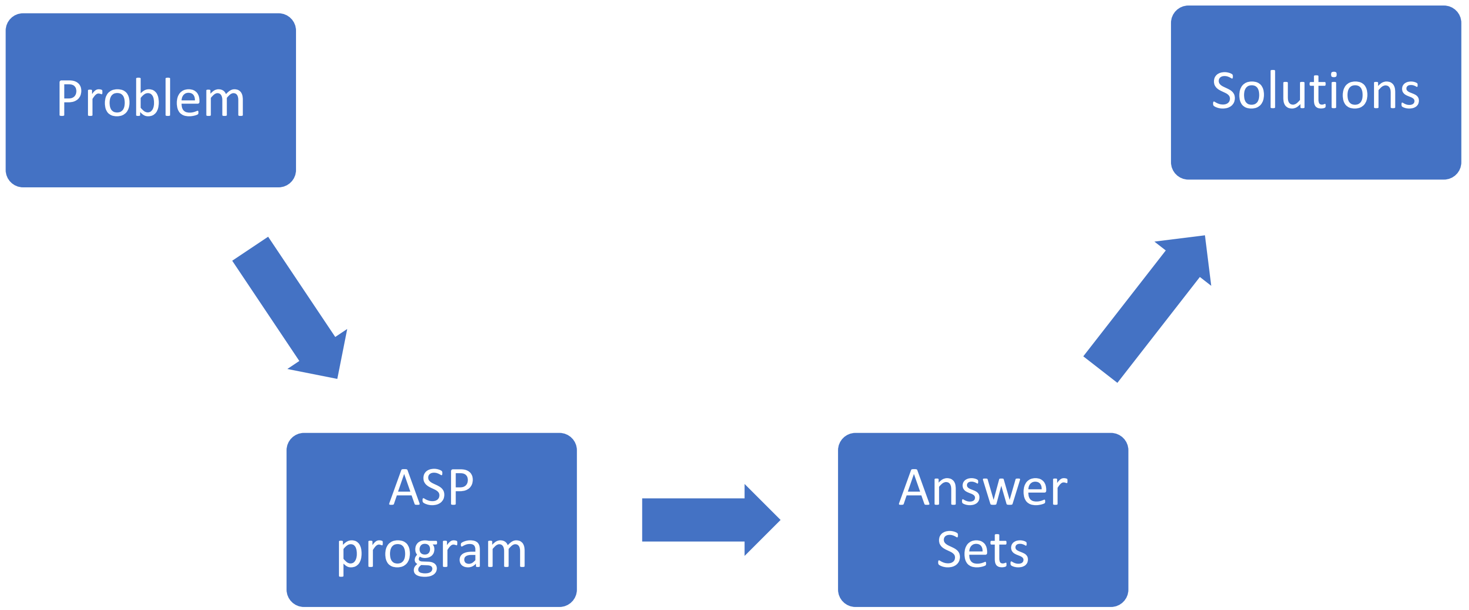
```
1 {value(C, heads); value(C, tails) } 1 :- coin(C).
coin(c1).
biased_coin(c1).
```

What examples could we give to learn the constraint

```
:- value(C, heads), biased_coin(C).
```

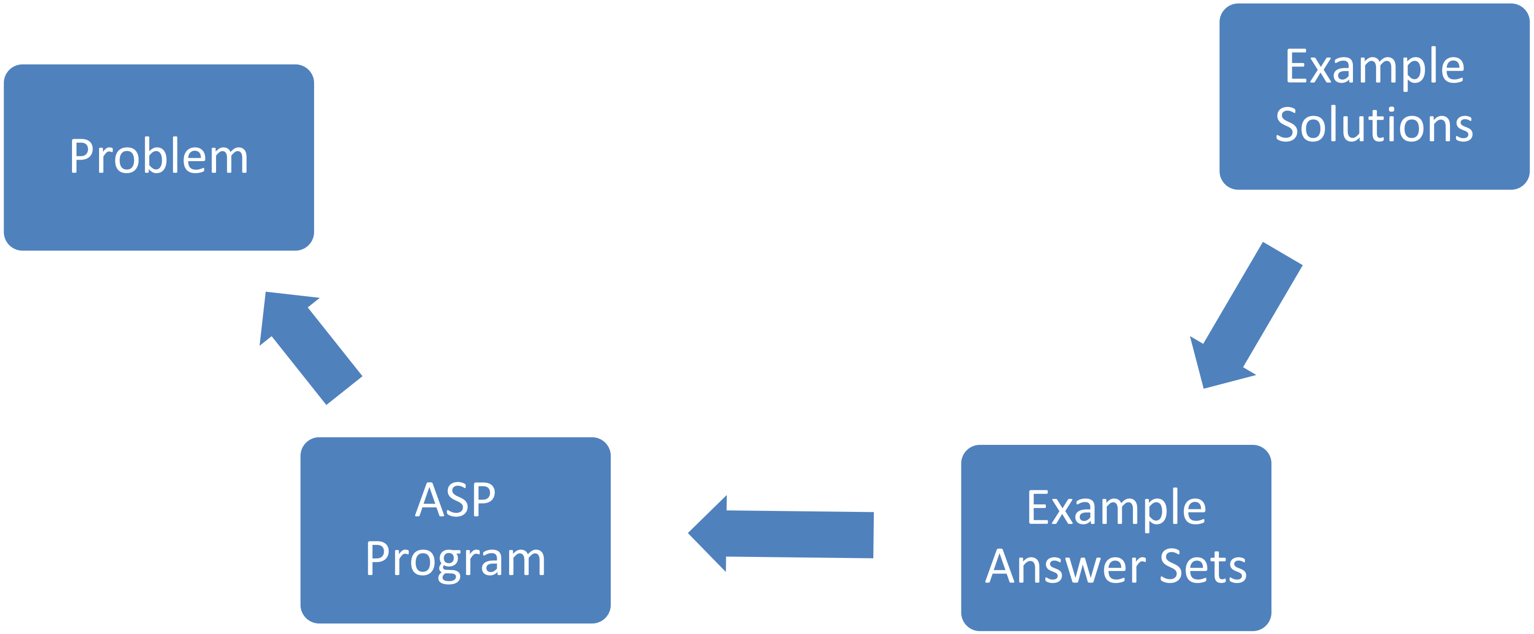
Learning from Answer Sets

Answer Set Programming



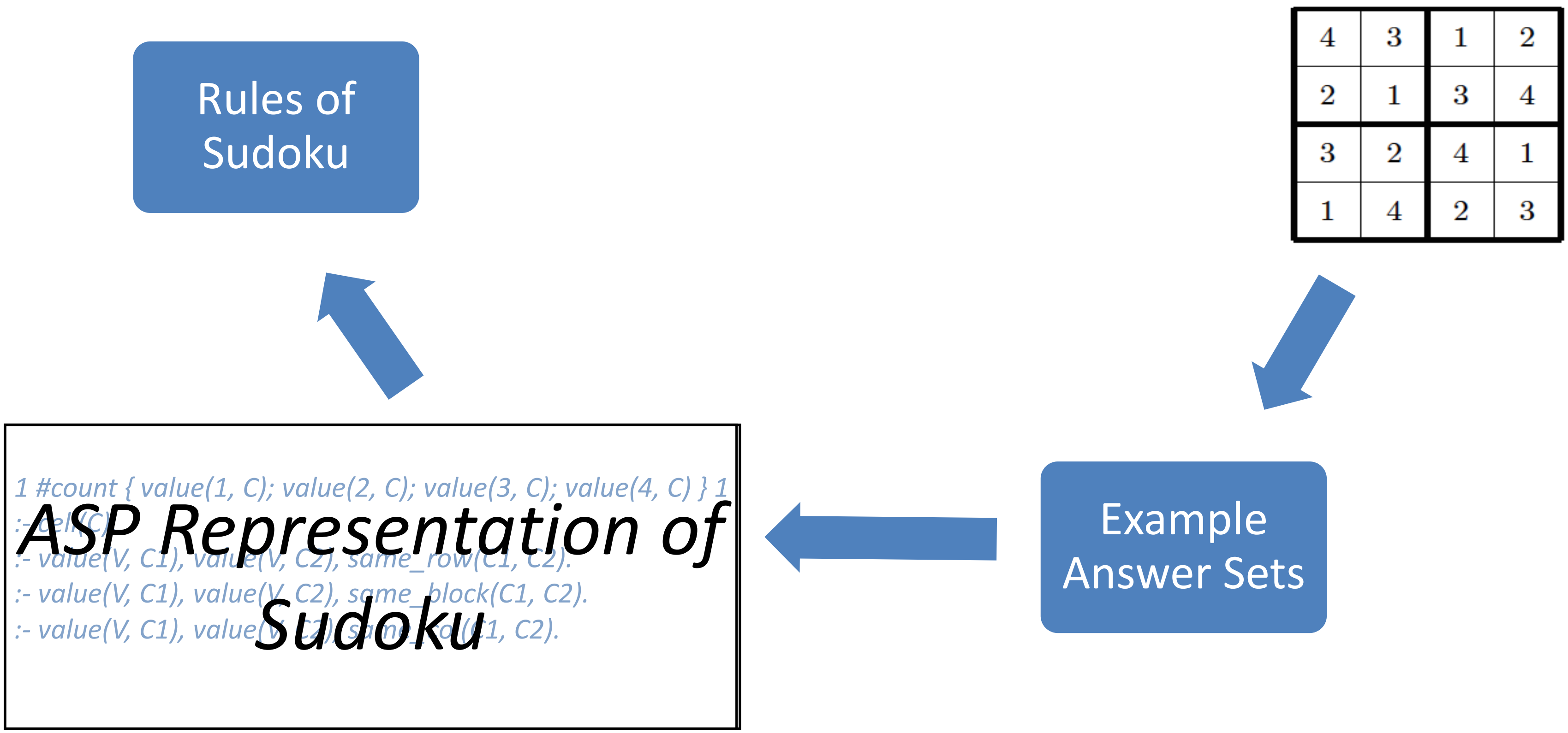
Inductive Learning of Answer Set Programs

From examples of what should/shouldn't be an Answer Set, we learn an appropriate hypothesis



Inductive Learning of Answer Set Programs

We can learn the rules of sudoku from examples boards



Partial Interpretations

A partial interpretation e is a pair of sets of atoms $<e^{inc}, e^{exc}>$ the *inclusions* and the *exclusions*.

A Herbrand Interpretation I *extends* a partial interpretation e if and only if:

$e^{inc} \subseteq I$

$e^{exc} \cap I = \varnothing$

`{ p, q }` and `{ p, q, s }` both extend `< { p, q }, { r } >`

Neither `{ p }` or `{ p, q, r }` do.

Learning from Answer Sets

LAS setting for ILP under the Answer Set semantics:

- Background knowledge *B* an ASP program
- Positive and negative examples E^+ and E^- (partial interpretations)
- Hypothesis space S_M (a set of normal rules, choice rules and constraints):
- Find a hypothesis *H* such that:
  - $H \subseteq S_M$

$$\forall e^+ \in E^+: \exists A \in AS(B \cup H) \text{ st } A \text{ extends } e^+$$

$$\forall e^- \in E^-: \nexists A \in AS(B \cup H) \text{ st } A \text{ extends } e^-$$

LAS: relation to brave induction

$$ILP_{brave}\langle B, E^+, E^- \rangle$$

$$\Downarrow$$

$$ILP_{LAS}\langle B, \{\langle E^+, E^- \rangle\}, \emptyset \rangle$$

Brave Induction Relationship : Example

Reconsider the Brave Induction task:

B

```
p :- not q.
q :- not p, not r.
s :- r.
```

E^+

```
s
```

E^-

```
q
```

`s.`	`:- not s.` `:- q.`	`p.` `s.`	`r.`
`{ p, s },` `{ q, s }`	NONE!	`{ p, s }`	`{p, r, s }`
✓	✗	✓	✓

What is the equivalent ILP_{LAS} task?

`< B, { <{s}, {q}> }, { } >`

LAS: relation to cautious induction

$$ILP_{cautious}\langle B, \{e_1^+, \ldots, e_m^+\}, \{e_1^-, \ldots, e_n^-\}\rangle$$

$$\Downarrow$$

$$ILP_{LAS}\langle B, \{\langle\emptyset, \emptyset\rangle\}, \{\langle\emptyset, \{e_1^+\}\rangle \ldots \langle\emptyset, \{e_m^+\}\rangle, \langle\{e_1^-\}, \emptyset\rangle \ldots \langle\{e_n^-\}, \emptyset\rangle\}\rangle$$

Positive Example

Negative Example

Cautious Induction Relationship: Example

Reconsider the Cautious Induction task:

B

```
p :- not q.
q :- not p, not r.
s :- r.
```

E^+

```
s
```

E^-

```
q
```

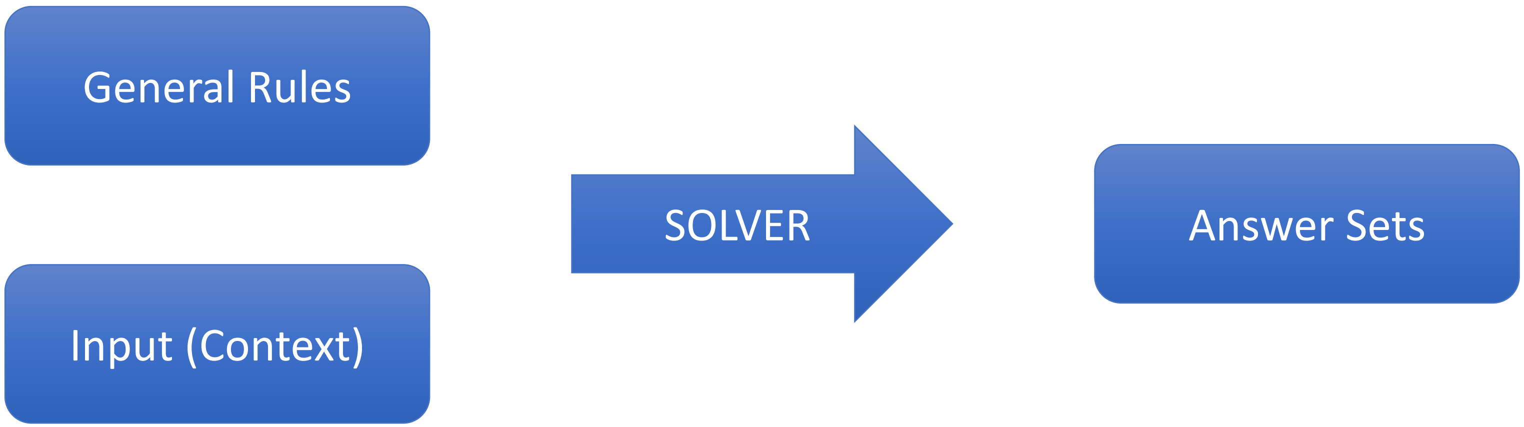
`s.`	`:- not s.` `:- q.`	`p.` `s.`	`r.`
`{ p, s },` `{ q, s }`	NONE!	`{ p, s }`	`{p, r, s }`
✗	✗	✓	✓

What is the equivalent $\boldsymbol{ILP_{LAS}}$ task?

< B, { <{}, {}> }, { <{}, {s}>, <{q}, {}> } >

Context-dependent Examples

Input to Answer Set Programs



% General rules:

```
0 { in(X, Y) } 1 :- edge(X, Y).
reach(1).     reach(Y) :- reach(X), in(X, Y).
:- node(X), not reach(X).
:- in(X, Y), in(X, Z), Y != Z.
```

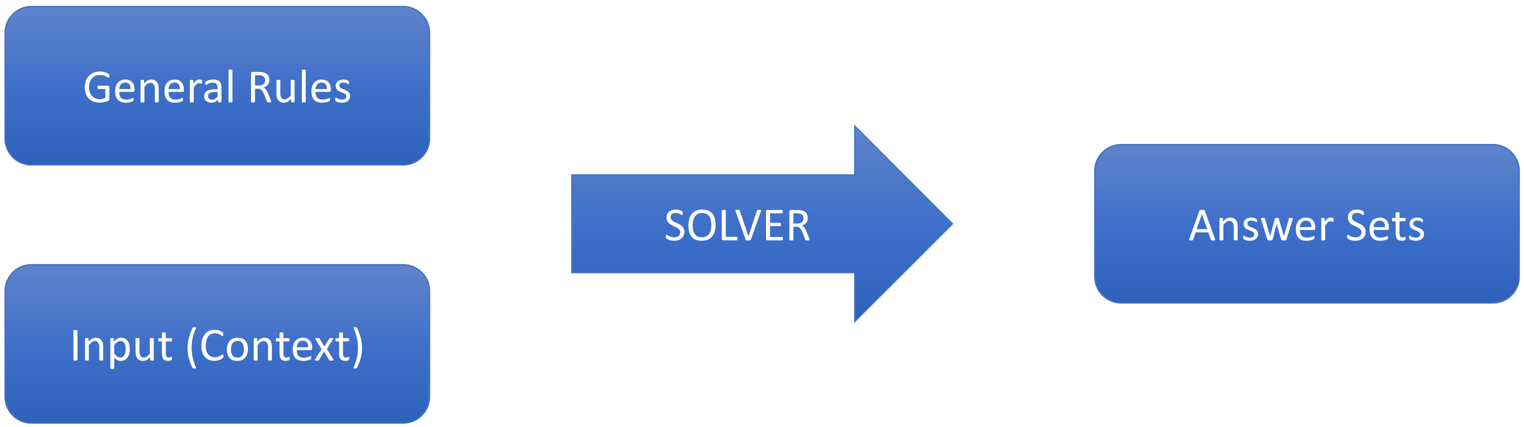
% Input context:

```
node(1..4).  edge(1, 2). edge(2, 3).
edge(2, 4).  edge(3, 4). edge(4, 1).
```

Answer Sets:

```
node(1..4), edge(1, 2), edge(2, 3),
edge(2, 4), edge(3, 4), edge(4, 1),
reach(1..4), in(1, 2), in(2, 3),
in(3, 4), in(4, 1)
```

Input to Answer Set Programs



% General rules:

```
0 { in(X, Y) } 1 :- edge(X, Y).
reach(1).     reach(Y) :- reach(X), in(X, Y).
:- node(X), not reach(X).
:- in(X, Y), in(X, Z), Y != Z.
```

% Input context:

```
node(1..4).  edge(1, 2). edge(2, 4).
             edge(3, 4). edge(4, 1).
```

Answer Sets:

```
UNSATISFIABLE
```

Context-dependent Examples

A *Context Dependent Partial Interpretation* (CDPI) is a pair $e = \langle e_{pi}, e_{ctx} \rangle$, where e_{pi} is a partial interpretation and e_{ctx} is an ASP program.

Given a program P and an interpretation I, I is an *accepting answer set of* e wrt P iff $I \in AS(P \cup e_{ctx})$ and I extends e_{pi}.

```
% P:
p :- not q.
```

1. $\{p\}$ **is** an accepting answer set of $\langle\langle\{p\}, \emptyset\rangle, \emptyset\rangle$ wrt P
2. $\{\}$ **is not** an accepting answer set of $\langle\langle\emptyset, \{p\}\rangle, \emptyset\rangle$ wrt P
3. $\{p\}$ **is not** an accepting answer set of $\langle\langle\{p\}, \emptyset\rangle, \{\ q.\ \}\rangle$ wrt P
4. $\{q\}$ **is** an accepting answer set of $\langle\langle\emptyset, \{p\}\rangle, \{\ q.\ \}\rangle$ wrt P

Context-dependent LAS

Context-dependent LAS setting:

- Background knowledge B (ASP program)
- Positive and negative examples E^+ and E^- (CDPIs)
- Hypothesis space S_M (normal/choice rules, constraints)
- Find a hypothesis H such that:

1. $H \subseteq S_M$
2. $\forall e^+ \in E^+$: ***at least one*** accepting answer set of e^+ wrt $B \cup H$
3. $\forall e^- \in E^-$: ***no*** accepting answer sets of e^- wrt $B \cup H$

Example

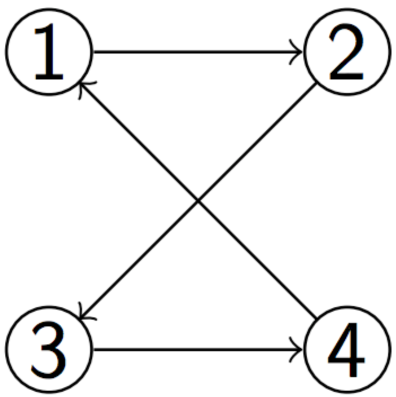
$ILP_{LAS}^{context}$ allows for a natural representation of contextual information (such as weather conditions).

$$B = \emptyset, \qquad E^{+} = \left\{ \begin{array}{c} \langle\langle\{go\_out\}, \emptyset\rangle, \emptyset\rangle \\ \langle\langle\emptyset, \{go\_out\}\rangle, \{raining.\}\rangle \end{array} \right\}, \qquad E^{-} = \emptyset$$

One solution is:

```
go_out :- not raining.
```

Hamilton in LAS



$$\left\langle \left\{ \begin{array}{l} \texttt{size(4)} \\ \texttt{edge(1, 2)} \\ \texttt{edge(2, 3)} \\ \texttt{edge(3, 4)} \\ \texttt{edge(4, 1)} \end{array} \right\}, \left\{ \begin{array}{c} \texttt{edge(1, 1)} \\ \texttt{edge(1, 3)} \\ \texttt{edge(1, 4)} \\ \ldots \end{array} \right\} \right\rangle$$

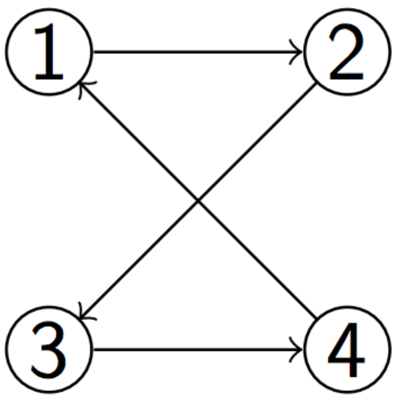
B:

```
1 { size(1..4) } 1.
node(1..N) :- size(N).
0 { edge(V0, V1) } 1 :- node(V0),
                        node(V1).
```

H:

```
0 { in(V0, V1) } 1 :- edge(V0, V1).
reach(V0) :- in(1, V0).
reach(V1) :- in(V0, V1), reach(V0).
:- node(V0), not reach(V0).
:- in(V0, V1), in(V0, V2), V1 != V2.
```

Efficient Hamilton in Context-dependent LAS



$$\left\langle \langle\emptyset, \emptyset\rangle, \left\{ \begin{array}{l} \texttt{node(1..4).} \\ \texttt{edge(1, 2).} \\ \texttt{edge(2, 3).} \\ \texttt{edge(3, 4).} \\ \texttt{edge(4, 1).} \end{array} \right\} \right\rangle$$

B:

```
% EMPTY
```

H:

```
0 { in(V0, V1) } 1 :- edge(V0, V1).
reach(V0) :- in(1, V0).
reach(V1) :- in(V0, V1), reach(V0).
:- node(V0), not reach(V0).
:- in(V0, V1), in(V0, V2), V1 != V2.
```

Logic-based Learning of Preferences

Preference Learning

There are many approaches to preference learning:

Collaborative filtering approaches identify similar users, and one user is recommended an item based on the actions of other users.

"Students with your chosen courses also took knowledge representation."

Object ranking approaches, aim to learn an ordering over a set of objects, based on examples of which objects are *preferred* to others.

In ASP, objects are represented by answer sets, and the preference ordering is represented by the weak constraints' ordering of the answer sets.

(Context-dependent) Ordering Examples

An ordering example is a pair of CDPIs $\langle e_1, e_2 \rangle$.

- Roughly speaking, the learned weak constraints should mean that e_1 is preferred to e_2.

There are two notions of coverage for ordering examples: *brave* and *cautious*.

- For an ordering to be *bravely respected*, there must be **at least one** pair of accepting answer sets A_1 and A_2 of e_1 and e_2 (wrt $B \cup H$) such that A_1 is preferred to A_2.
- For an ordering to be *cautiously respected*, **for each** pair of accepting answer sets A_1 and A_2 of e_1 and e_2 (wrt $B \cup H$), A_1 must be preferred to A_2.

Example

$$H = \left\{ \begin{array}{l} \texttt{:\sim mode(Leg, walk), crime\_rating(Leg, C), C > 4 . [1@3, Leg]} \\ \texttt{:\sim mode(Leg, bus) . [1@2, Leg]} \\ \texttt{:\sim mode(Leg, walk), distance(Leg, Distance) . [Distance@1, Leg]} \end{array} \right.$$

$$B = \emptyset$$

Does $B \cup H$ bravely respect the following example?

$$\left\langle\left\langle \langle \emptyset, \emptyset \rangle, \left\{ \begin{array}{c} mode(1,walk). \\ mode(2,bus). \\ distance(1,1000). \\ distance(2,4000). \\ crime\_rating(1,2). \\ crime\_rating(2,5). \end{array} \right\} \right\rangle, \left\langle \langle \emptyset, \emptyset \rangle, \left\{ \begin{array}{c} mode(1,bus). \\ mode(2,walk). \\ distance(1,3000). \\ distance(2,2000). \\ crime\_rating(1,4). \\ crime\_rating(2,5). \end{array} \right\} \right\rangle\right\rangle$$

✓

Example

$$H = \left\{ \begin{array}{l} \texttt{:~ mode(Leg, walk), crime\_rating(Leg, C), C > 4 . [1@3, Leg]} \\ \texttt{:~ mode(Leg, bus) . [1@2, Leg]} \\ \texttt{:~ mode(Leg, walk), distance(Leg, Distance) . [Distance@1, Leg]} \end{array} \right.$$

$$B = \emptyset$$

Does $B \cup H$ cautiously respect the following example?

$$\left\langle \left\langle \langle \emptyset, \emptyset \rangle, \left\{ \begin{array}{c} mode(1,walk). \\ mode(2,bus). \\ distance(1,1000). \\ distance(2,4000). \\ crime\_rating(1,2). \\ crime\_rating(2,5). \end{array} \right\} \right\rangle, \left\langle \langle \emptyset, \emptyset \rangle, \left\{ \begin{array}{c} mode(1,bus). \\ mode(2,walk). \\ distance(1,3000). \\ distance(2,2000). \\ crime\_rating(1,4). \\ crime\_rating(2,5). \end{array} \right\} \right\rangle \right\rangle$$

✓

Example

$$H = \left\{ \begin{array}{l} \texttt{:\sim\ mode(Leg, walk), crime\_rating(Leg, C), C > 4 . [1@3, Leg]} \\ \texttt{:\sim\ mode(Leg, bus) , not raining. [1@2, Leg]} \\ \texttt{:\sim\ mode(Leg, walk), distance(Leg, Distance) . [Distance@1, Leg]} \end{array} \right.$$

$$B = \{\ 0\{raining\}1.\ \}$$

Does $B \cup H$ cautiously respect the following example?

$$\left\langle\left\langle \langle \emptyset, \emptyset \rangle, \left\{ \begin{array}{c} mode(1,bus). \\ mode(2,bus). \\ distance(1,1000). \\ distance(2,4000). \\ crime\_rating(1,2). \\ crime\_rating(2,5). \end{array} \right\} \right\rangle, \left\langle \langle \emptyset, \emptyset \rangle, \left\{ \begin{array}{c} mode(1,bus). \\ mode(2,walk). \\ distance(1,3000). \\ distance(2,2000). \\ crime\_rating(1,4). \\ crime\_rating(2,4). \end{array} \right\} \right\rangle\right\rangle$$

✗

Example

$H = \left\{ \begin{array}{l} \texttt{:\~{} mode(Leg, walk), crime\_rating(Leg, C), C > 4 . [1@3, Leg]} \\ \texttt{:\~{} mode(Leg, bus) , not raining. [1@2, Leg]} \\ \texttt{:\~{} mode(Leg, walk), distance(Leg, Distance) . [Distance@1, Leg]} \end{array} \right.$

$B = \{\ 0\{raining\}1.\ \}$

Does $B \cup H$ bravely respect the following example?

$$\left\langle\left\langle \langle \emptyset, \emptyset \rangle, \left\{ \begin{array}{c} mode(1,bus). \\ mode(2,bus). \\ distance(1,1000). \\ distance(2,4000). \\ crime\_rating(1,2). \\ crime\_rating(2,5). \end{array} \right\} \right\rangle, \left\langle \langle \emptyset, \emptyset \rangle, \left\{ \begin{array}{c} mode(1,bus). \\ mode(2,walk). \\ distance(1,3000). \\ distance(2,2000). \\ crime\_rating(1,4). \\ crime\_rating(2,4). \end{array} \right\} \right\rangle\right\rangle$$

✓

Context-dependent LOAS

Context-dependent Learning from Ordered Answer Sets setting:

- Background knowledge B (ASP program)
- Positive and negative examples E^+ and E^- (CDPIs)
- Brave and cautious ordering examples (CDOEs)
- Hypothesis space S_M (normal/choice rules, hard/weak constraints)
- Find a hypothesis H such that:

1. $H \subseteq S_M$
2. $\forall e \in E^+$: *at least one* accepting answer set of e wrt $B \cup H$
3. $\forall e \in E^-$: *no* accepting answer sets of e wrt $B \cup H$
4. $\forall o \in O^b$: $B \cup H$ must *bravely respect* o
5. $\forall o \in O^c$: $B \cup H$ must *cautiously respect* o

Logic Based Learning from Noisy Examples

An unfortunate lack of perfection

Until now, we have assumed that all examples are *perfectly labelled*. In the real world some examples may be noisy.

Consider the ILP_b task $\mathrm{T} = \langle B, M, E^+, E^- \rangle$, where:

$$
\begin{aligned}
B &= \{\ t(1). \ldots t(10).\ \} \\
M &= \{\ \#modeh(q(+t)).\ \} \\
E^+ &= \{\ q(1), q(2), \ldots, q(9)\ \} \\
E^- &= \{\ q(10)\ \}
\end{aligned}
$$

This task is UNSATISFIABLE, but there is a hypothesis that covers all but one of the examples!

An unfortunate lack of perfection

Until now, we have assumed that all examples are *perfectly labelled*. In the real world some examples may be noisy.

Consider the ILP_b task $\mathrm{T} = \langle B, M, E^+, E^- \rangle$, where:

$$
\begin{aligned}
B &= \{\ t(1..12).\ \ f(2..4).\ \ mul(2, \ldots).\ \ mul(3, \ldots).\ mul(4, \ldots).\ \} \\
M &= \left\{ \begin{array}{c} modeh(q(+t)). \\ modeb\big(*, mul(\#f, +t)\big). \\ modeb\big(*, not\ mul(\#f, +t)\big) \end{array} \right\} \\
E^+ &= \{\, q(1), q(2), \ldots, q(5), q(7), \ldots, q(12)\,\} \\
E^- &= \{\, q(6)\,\}
\end{aligned}
$$

The only solution of this task is:

$$
\begin{array}{c}
\mathrm{q}(X) \leftarrow t(X), not\ mul(2, X). \\
q(X) \leftarrow t(X), not\ mul(3, X). \\
q(X) \leftarrow t(X), mul(4, X).
\end{array}
$$

There is a simpler hypothesis that covers all but one example:

$$\mathrm{q}(X) \leftarrow t(X).$$

Weighted/Penalised Examples

Given any ILP framework ILP_F, an $\mathrm{n}(ILP_F)$ task is an ILP_F task such that every example has been annotated with a *weight* – either a positive integer or ∞.

Consider the $n(ILP_b)$ task $\mathrm{T} = \langle B, M, E^+, E^- \rangle$, where:

$$
\begin{aligned}
B &= \{\ t(1..10).\ \} \\
M &= \{\ \#modeh(q(+t)).\ \} \\
E^+ &= \{\ q(1)@1, q(2)@1, \ldots, q(9)@1\ \} \\
E^- &= \{\ q(10)@1\ \}
\end{aligned}
$$

Given any hypothesis H, $S(H,T) = |H| + \sum_{e@w \in U} w$, where U is the set of examples in T that are not covered by H.

An inductive solution must have a finite score and an optimal inductive solution is a solution with minimum score.

What are the (optimal) inductive solutions of T?

∅ (with the score of 9)

$q(X) \leftarrow t(X).$ (with the score of 2)

Example

Consider the $n(ILP_b)$ task $\mathrm{T} = \langle B, M, E^+, E^- \rangle$, where:

$$
\begin{aligned}
B &= \{\ t(1..10).\ \} \\
M &= \{\ \#modeh(q(+t)).\ \} \\
E^+ &= \{\ q(1)@1, q(2)@1, \ldots, q(9)@1\ \} \\
E^- &= \{\ q(10)@\infty\ \}
\end{aligned}
$$

What are the optimal inductive solutions of T?

∅ (with the score of 9)

Example

Consider the $n(ILP_b)$ task $\mathrm{T} = \langle B, M, E^+, E^- \rangle$, where:

$$\begin{aligned}
B &= \{\ t(1..10).\ \} \\
M &= \{\ \#modeh(q(+t)).\ \} \\
E^+ &= \{\ q(1)@1, q(2)@\infty, \ldots, q(9)@1\ \} \\
E^- &= \{\ q(10)@\infty\ \}
\end{aligned}$$

What are the optimal inductive solutions of $\boldsymbol{T}$?

$UNSATISFIABLE$

Solving penalised brave induction with ASP

The ASPAL encoding of an ILP_b task $\langle B, M, \{e_1^+, \ldots, e_m^+\}, \{e_1^-, \ldots, e_n^-\}\rangle$ contains the rules:

$$\texttt{goal :- } \mathrm{e}_1^+, \ldots, \mathrm{e}_\mathrm{m}^+, \mathrm{not}\ \mathrm{e}_1^-, \ldots, \mathrm{e}_\mathrm{n}^-.$$
$$\texttt{:- not goal.}$$

The n(ASPAL) encoding of an $\mathrm{n}(\mathrm{ILP_b})$ task $\langle \mathrm{B}, \mathrm{M}, \{\mathrm{e}_1^+ @\mathrm{w}_1^+, \ldots, \mathrm{e}_\mathrm{m}^+ @\mathrm{w}_\mathrm{m}^+\}, \{\mathrm{e}_1^- @\mathrm{w}_1^-, \ldots, \mathrm{e}_\mathrm{n}^- @\mathrm{w}_\mathrm{n}^-\}\rangle$ instead contains the weak constraints:

$$:\sim\ \mathrm{not}\ \mathrm{e}_1^+. [\mathrm{w}_1^+ @1, \mathrm{e}_1^+]$$
$$\ldots$$
$$:\sim\ \mathrm{not}\ \mathrm{e}_\mathrm{m}^+. [\mathrm{w}_\mathrm{m}^+ @1, \mathrm{e}_\mathrm{m}^+]$$
$$:\sim\ \mathrm{e}_1^-. [\mathrm{w}_1^- @1, \mathrm{e}_1^-]$$
$$\ldots$$
$$:\sim\ \mathrm{e}_\mathrm{n}^-. [\mathrm{w}_\mathrm{n}^- @1, \mathrm{e}_\mathrm{n}^-]$$

Learning from Noisy Examples Demo

ILASP summary

Version	Efficient for tasks with			
	negative examples	many examples	noise	large hypothesis spaces
1	✘	✘	✘	✘
2	✓	✘	✘	✘
2i	✓	✓	✘	✘
3	✓	✓	✓	✘

All versions of ILASP are sound and complete, and therefore guaranteed to return an optimal solution of any satisfiable task.

ILASP4 is currently in development. The aim is to address scalability with respect to the size of the hypothesis space.

ILASP is available to download from www.ilasp.com

Summary

This tutorial has covered:

- Brave and cautious induction
- ASPAL
- Learning from Answer Sets
  - Context-dependent learning
  - Preference learning
- ILASP
- Learning from noisy examples

The lecture notes also cover:

- Other ASP-based ILP algorithms
- Complexity/generality of the learning frameworks

ILASP

Inductive Learning of Answer Set Programs

Inductive Learning of Answer Set Programs (ILASP) is the algorithm which was developed to solve LAS tasks.

A hypothesis $H \in positive\_solutions\langle B, S_M, E^+, E^-\rangle$ if and only if:

1. $H \subseteq S_M$
2. $\forall e^+ \in E^+ \; \exists A \in AS(B \cup H)$ st A extends e^+

A hypothesis $H \in violating\_solutions\langle B, S_M, E^+, E^-\rangle$ if and only if:

1. $H \subseteq S_M$
2. $\forall e^+ \in E^+ \; \exists A \in AS(B \cup H)$ st A extends e^+
3. $\exists e^- \in E^- \; \exists A \in AS(B \cup H)$ st A extends e^-

$$
\begin{aligned}
&ILP_{LAS}\langle B, S_M, E^+, E^-\rangle \\
&\qquad = positive\_solutions\langle B, S_M, E^+, E^-\rangle \setminus violating\_solutions\langle B, S_M, E^+, E^-\rangle
\end{aligned}
$$

ILASP: Summary

Algorithm 1 ILASP

procedure ILASP(T)
  $solutions = []$
  for $n = 0$; $solutions.empty$; n++ **do**
    vs = violating solutions of length n
    solutions = positive solutions of length n not in vs
  end for
  return $solutions$
end procedure

ILASP1/2 Demo

ILASP1/2 are slow…

Algorithm 1 ILASP

procedure ILASP(T)
  $solutions = []$
  for $n = 0$; $solutions.empty$; $n{+}{+}$ **do**
    $vs = AS(T^n_{meta} \cup \{\leftarrow \text{ not } violating; \quad ex(negative).\})$
    $ps = AS(T^n_{meta} \cup \{constraint(meta^{-1}(V)) : V \in vs\})$
    $solutions = \{meta^{-1}(A) : A \in ps\}$
  end for
  return $solutions$
end procedure

Both ILASP1 and ILASP2 use a meta representation whose grounding is proportional to the number of examples.

Relevant Examples

In real tasks, many examples may be explained by the same hypotheses.

- In ILASP1 and ILASP2, the grounding of the meta-program is proportional to the number of examples.

- As ILASP learns non-monotonic programs, it cannot iteratively learn a hypothesis using a traditional cover loop.

- Instead, ILASP2i iteratively builds a *relevant* subset of the examples, and in each iteration uses ILASP2 to solve a task with this (usually) smaller subset of the examples.