# Lecture Four: Answer Set programming - Semantics

301315 Knowledge Representation and Reasoning

# ASP Syntax Overview

- A **signature** $\Sigma$ consists of objects, functions, predicates, and variables, i.e., $\Sigma = \{\mathcal{O}, \mathcal{F}, \mathcal{P}, \mathcal{V}\}$.
- A **program** $\Pi$ of ASP consists of a signature $\Sigma$ and a collection of **rules** of this form:
  $l_0 \ or \cdots or \ l_i \leftarrow l_{i+1}, \cdots, l_m, not \ l_{m+1}, \cdots, not \ l_n,$

  where each $l$ is a literal of $\Sigma$.
- The set of ground instantiations of rules of $\Pi$ is called the **grounding** of $\Pi$.

# ASP Syntax Overview

## Example

Program $\Pi$ consists of the following rules:

$$tutor(X) \text{ or } lecturer(X) \leftarrow mark\_assignment(X).$$
$$mark\_assignment(alice).$$
$$mark\_assignment(bob).$$

$\Sigma = \{\mathcal{O}, \mathcal{F}, \mathcal{P}, \mathcal{V}\}$: $\mathcal{O} = \{alice, bob\}$, $\mathcal{F} = \emptyset$,
$\mathcal{P} = \{mark\_assignment, tutor, lecturer\}$, $\mathcal{V} = \{X\}$.

# ASP Syntax Overview

## Example

(Continued). The grounding of $\Pi$ is a program $\Pi'$ consisting of following rules:

$$tutor(alice) \text{ or } lecturer(alice) \leftarrow mark\_assignment(alice).$$
$$tutor(bob) \text{ or } lecturer(bob) \leftarrow mark\_assignment(bob).$$
$$mark\_assignment(alice).$$
$$mark\_assignment(bob).$$

# The Semantics of ASP - Satisfiability

- Recall that a **literal** is an atom or its negation, e.g., $mark\_assignment(X)$, $\neg tutor(X)$.
- A **ground literal** is a literal not containing any variables, e.g., $tutor(alice)$, $\neg lecturer(bob)$.
- In ASP, a program's semantics is provided by the set of ground literals.

# The Semantics of ASP - Satisfiability

When does a set of ground literals satisfy a rule $r$?

$$l_0 \ or \cdots or \ l_i \leftarrow l_{i+1}, \cdots, l_m, not \ l_{m+1}, \cdots, not \ l_n$$

## Definition

A set $S$ of ground literals *satisfies*:

1. $l$ if $l \in S$;
2. $not \ l$ if $l \notin S$;
3. $l_0 \ or \cdots or \ l_i$ if for some $0 \leq j \leq i$, $l_j \in S$ ;
4. a set $\{l_{i+1}, \cdots, l_m, not \ l_{m+1}, \cdots, not \ l_n\}$ if $S$ satisfies each $l_h$ and each $not \ l_k$ where $(i+1) \leq h \leq m$ and $(m+1) \leq k \leq n$;
5. $r$ if whenever $S$ satisfies $r$'s body, it satisfies $r$'s head.

# The Semantics of ASP - Satisfiability

## Example

Let $r$ be the following rule:

$$p(a) \text{ or } p(b) \leftarrow q(b), \neg t(c), \text{not } t(b).$$

Do any of these sets satisfy $r$?

- $\{\neg p(a), q(b), \neg t(c)\}$ no
- $\{q(b), \neg t(c)\}$ no
- $\emptyset$ yes
- $\{p(a)\}$ yes
- $\{p(b), q(b), \neg t(c)\}$ yes

# Informal Semantics - General Principles

- Program $\Pi$ can be viewed as a specification for **answer sets** - sets of beliefs that could be held by a rational reasoner associated with $\Pi$.
- We define a set $S$ of ground literals by the following principles:
  1. Each rule of $\Pi$ is satisfied in $S$, that is, the head of the rule is believed if the corresponding body is believed;
  2. Do not believe in contradictions;
  3. Adhere to a "rationality principle" - do not believe anything unless there is an explicit evidence to support it;
  4. $S$ is minimal in some sense.

# Informal Semantics - General Principles

Example

$p(b) \leftarrow q(a).$ "Believe $p(b)$ if you believe $q(a)$.

$q(a).$ "Believe $q(a)$.

Basically, a set $S$ of ground literals is an **answer set** of program $\Pi$ if $S$ obeys the principles described earlier.

This program has one answer set $A = \{p(b), q(a)\}$.

# Informal Semantics - About Classical Negation

Example

$\neg p(b) \leftarrow \neg q(a).$ "Believe that $p(b)$ is false if you believe that $q(a)$ is false."

$\neg q(a).$ "Believe that $q(a)$ is false."

This program has one answer set $A = \{\neg p(b), \neg q(a)\}$.

Note: There is no difference about negative literals.

# Informal Semantics - About Epistemic Disjunction

Example

$p(a)$ *or* $p(b)$. "Believe $p(a)$ or believe $p(b)$."

This program has two answer sets: $A_1 = \{p(a)\}$, $A_2 = \{p(b)\}$.

Note: The minimality principle eliminates the set $\{p(a), p(b)\}$.

# Informal Semantics - Reasoning by Cases

Example

$p(a)$ *or* $p(b)$.
$q(a) \leftarrow p(a)$.
$q(a) \leftarrow p(b)$.

This program has two answer sets: $A_1 = \{p(a), q(a)\}$, and $A_2 = \{p(b), q(a)\}$.

# Informal Semantics - About Constraints

Example

$p(a)$ *or* $p(b)$. "Believe $p(a)$ or believe $p(b)$."

$\leftarrow p(a)$. "It is impossible to believe $p(a)$."

This program has a unique answer set $A = \{p(b)\}$.

Note: A constraint limits the sets of beliefs an agent can have, but does not serve to derive any new information.

# Informal Semantics - Default Negation

- **Default negation** has been used by an agent to make a conclusion based on the absence of information.

Example

$p(a) \leftarrow not\ q(a).$ "If $q(a)$ does not belong to your set of beliefs, then $p(a)$ must."

If we cannot prove $q(a)$, then we believe $p(a)$. So this program has an answer set $A = \{p(a)\}$.

# Informal Semantics - Default Negation

Example

| | |
|---|---|
| $p(a) \leftarrow \mathit{not}\ q(a).$ | "If $q(a)$ does not belong to your set of beliefs, then $p(a)$ must." |
| $p(b) \leftarrow \mathit{not}\ q(b).$ | "If q(b) does not belong to your set of beliefs, then $p(b)$ must." |
| $q(a).$ | "Believe $q(a)$." |

This program has a unique answer set $A = \{q(a), p(b)\}$.

# Informal Semantics - Default Negation

10 min classroom exercise

## Example

Consider an ASP program $\Pi$ containing the following rules:

$$p(X) \leftarrow not\ q(X).$$
$$q(X) \leftarrow not\ p(X).$$
$$t(a).$$
$$t(b).$$

(a) What is the grounding of $\Pi$?

(b) Find all answer sets of $\Pi$

# ASP Entailment

## Definition

A program $\Pi$ *entails* a literal $l$, denoted as $\Pi \models l$, if $l$ is in *all* answer sets of $\Pi$.

It is important to note that the ASP entailment is **non-monotonic**.

# ASP Entailment

## Example

Consider program $\Pi$:

$$p(a) \leftarrow not\ q(a).$$

$\Pi$ has one answer set $A = \{p(a)\}$, so $\Pi \models p(a)$.

Now add $q(a)$ to $\Pi$ to form $\Pi'$:

$$p(a) \leftarrow not\ q(a).$$
$$q(a).$$

$\Pi' \not\models p(a)$, because $\Pi'$ has only one answer set $A' = \{q(a)\}$.

# Query Answering - Definitions

## Definition

A **query** is a conjunction or disjunction of literals. A **ground query** is a query where each literal is a ground literal.

## Example

Three queries:

$tutor(X) \wedge student(Y).$
$tutor(alice)$ *or* $lecturer(alice).$
$tutor(bob).$

Let $l$ be a literal, we use $\overline{l}$ to denote the negation of $l$, e.g.,

- for literal $tutor(X)$, $\overline{tutor(X)}$ is $\neg tutor(X)$;
- for literal $\neg tutor(X)$, $\overline{\neg tutor(X)}$ is $tutor(X)$.

# Query Answering - Definitions

## Definition

The **answer** to a query is defined as follows.

1. The answer to a ground conjunctive query, $l_1 \wedge \cdots \wedge l_n$, where $n \geq 1$, is
   - *yes* if $\Pi \models \{l_1, \cdots, l_n\}$;
   - *no* if $\Pi \models \overline{l_i}$ for some $l_i$ $(1 \leq i \leq n)$;
   - *unknown* otherwise.
2. The answer to a ground disjunctive query, $l_1$ *or* $\cdots$ *or* $l_n$, where $n \geq 1$, is
   - *yes* if $\Pi \models l_i$ some $l_i$ $(1 \leq i \leq n)$;
   - *no* if $\Pi \models \{\overline{l_1}, \cdots, \overline{l_n}\}$;
   - *unknown* otherwise.
3. The answer to a query $q(X_1, \cdots, X_n)$, where $X_1, \cdots, X_n$ is the list of variables occurring in $q$, is a sequence of ground terms $t_1, \cdots, t_n$ such that $\Pi \models q(t_1, \cdots, t_n)$.

# Query Answering - Examples

## Example

Answer to a Query

$$p(a) \leftarrow \mathit{not}\ q(a)$$

This program has one answer set $\{p(a)\}$. What does it answer to the following queries?

1. $p(a)$
2. $q(a)$
3. $p(a) \wedge q(a)$
4. $p(a)\ \mathit{or}\ q(a)$
5. $p(X)$

# Query Answering - Examples

## Example

Let us make a new program by adding a rule to the previous one:

$p(a) \leftarrow \mathit{not}\ q(a).$

$\neg q(X) \leftarrow \mathit{not}\ q(X).$ "If $q(X)$ is not believed to be true, believe that it is false."

This program has one answer set $\{p(a), \neg q(a)\}$. What does it answer to the following queries?

1. $p(a)$
2. $q(a)$
3. $p(a) \wedge q(a)$
4. $p(a)\ \mathit{or}\ q(a)$
5. $p(X)$

# Formal Semantics - Part I

Now it is the time to present the formal semantics for Answer Set Programming.

- The definition of answer sets has two parts.
- The first defines answer sets of programs without default negation.
- The second explains how to remove default negation so that we can apply the first part.

# Formal Semantics - Part I

Consistency

Pairs of literals of the form $p(t_1, \cdots, t_n)$ and $\neg p(t_1, \cdots, t_n)$ are called *contrary*. A set $S$ of ground literals is called **consistent** if it contains no contrary literals.

# Formal Semantics - Part I

## Definition

Let $\Pi$ be a program not containing default negation *not*. An **answer set** $S$ of $\Pi$ is a consistent set of ground literals such that

- $S$ satisfies the rules of $\Pi$, and
- $S$ is minimal, i.e., there is no proper subset of $S$ satisfying the rules of $\Pi$.

# Formal Semantics - Part I

## Example (1)

$\Pi$ consists of the following two rules:

$$p(b) \leftarrow q(a).$$
$$q(a).$$

We can check that $\{p(b), q(a)\}$ is the only answer set of $\Pi$.

What about entailment and answers to queries?

- $?q(a)$
- $?\neg q(a)$
- $?p(b)$
- $?\neg p(b)$

# Formal Semantics - Part I

## Example (2)

$p(a) \leftarrow p(b).$
$\neg p(a).$

Compute this program's answer set, and answer the following queries?

- $?p(a)$
- $?\neg p(a)$
- $?p(b)$
- $?\neg p(b)$

Note that this example demonstrates that $\leftarrow$ is not classical implication.

### Example (3)

Empty Set Answer Set.

$$p(b) \leftarrow \neg p(a).$$

What is the answer set of this program? What do we believe about $\neg p(a)$? How about $p(b)$?

Example (4)

Epistemic Disjunction.

$$p(a) \text{ or } p(b).$$

This program has two answer sets $A_1 = \{p(a)\}$ and $A_2 = \{p(b)\}$. Does this program entail $p(a)$?

What does the following program entail?

$$p(a) \text{ or } p(b).$$
$$q(a) \leftarrow p(a).$$
$$q(a) \leftarrow p(b).$$

### Example (5)

$p(a)$ *or* $\neg p(a)$ Is Not the Same As $p(a) \vee \neg p(a)$.

$$p(b) \leftarrow \neg p(a).$$
$$p(b) \leftarrow p(a).$$
$$p(a) \textit{ or } -p(a).$$

The addition of the last rule forces the agent to make a decision one way or the other, instead of remaining undecided. Instead of an empty set, we have two answer sets: $\{p(a), p(b)\}$ and $\{\neg p(a), p(b)\}$. What does this program entail?

### Example (6)

Constraint Revisited.

$$p(a) \text{ or } p(b).$$
$$\leftarrow p(a).$$

We have two answer sets from the first rule, but the second rule makes us exclude the possibility of $\{p(a)\}$ because it is impossible to satisfy an empty head if the body is satisfied.

So this program has a unique answer set $\{p(b)\}$.

# Formal Semantics - Part II

## Definition

Let $\Pi$ be an arbitrary program and $S$ be a set of ground literals. By $\Pi^S$ we denote the program obtained from $S$ by

1. removing all rules containing *not* $l$ such that $l \in S$;
2. removing all other premises containing *not*.

$S$ is an answer set of $\Pi$ if $S$ is an answer set of $\Pi^S$.

We refer to $\Pi^S$ as the **reduct** of $\Pi$ with respect to $S$.

### Example (1)

Default Negation Revisited.

$$p(a) \leftarrow not\ q(a).$$
$$p(b) \leftarrow not\ q(b).$$
$$q(a).$$

Let $S = \{q(a), p(b)\}$, then $\Pi^S$ is

$$p(b) \leftarrow not\ q(b).$$
$$q(a).$$

We see that $S$ is an answer set of $\Pi^S$, so $S$ is an answer set of $\Pi$, which is also the unique answer set of $\Pi$.

## Example (2)

No Answer Set.

$$p(a) \leftarrow not\ p(a).$$

It is silly to ask an agent to believe in something simply because it does not believe it.
We can check none of the following is an answer set of this program:

$$S_1 = \emptyset$$
$$S_2 = \{p(a)\}$$
$$S_3 = \{\neg p(a)\}$$
$$S_4 = \{p(a), \neg p(a)\}$$

## Example (3)

Other No Answer Set Cases.

Other inconsistent programs (programs that have no answer sets) include

$p(a).$
$\neg p(a).$

and

$p(a).$
$\leftarrow p(a).$

# Tutorial and Lab Exercises

1. Consider the following program $\Pi$:

   $p(a) \leftarrow not\ p(a).$
   $p(a).$

   Find all answer set(s) of this program.

2. Consider the following program:

   $p(a) \leftarrow not\ p(b).$
   $p(b) \leftarrow not\ p(a).$

   Find all answer set(s) of this program. Does $\Pi \models p(a)$ hold?

# Tutorial and Lab Exercises

3. Consider the following program written in ASP solver *clingo* syntax:

```
s(b).
-s(c).
r(a).
p(a) | p(b).
q(X) :- p(X), r(X), not s(X).
```

Run *clingo* command to compute *all* answer set(s) of this program, and then verify each answer set against the program, by following the answer set definition.

# Tutorial and Lab Exercises

4. We want to represent the statement that "all professors are adult" using an ASP program. There are different ways to encode this statement into an ASP program. Consider the following two programs:

$\Pi_1$:
$adult(X) \leftarrow prof(X).$
$adult(X)\ or\ \neg adult(X).$
$prof(X)\ or\ \neg prof(X).$
$prof(alice).$
$\neg adult(bob).$

$\Pi_2$:
$adult(X) \leftarrow prof(X).$
$\neg prof(X) \leftarrow \neg adult(X).$
$prof(alice).$
$\neg adult(bob).$

Rewrite $\Pi_1$ and $\Pi_2$ to comply *clingo* syntax, and run them using *clingo*. Do these two program have the same answer set?